POLITECNICO DI TORINO

Master's Degree in Communications and Computer Networks Engineering

Master's Thesis

Outdoor Localization of an Assistive Autonomous Robot

Supervisors
Prof. Roberto Garello
Prof. Marina Mondin

Candidate
Alessandro Moro

Academic Year 2020-2021

Abstract

This thesis regards the design, development and testing of the localization system for Autonomous Mobile Robots in outdoor environments, a project carried out in collaboration with InnoTech Systems L.L.C. and California State University of Los Angeles. The ability to localize itself on the map is at the base of autonomous navigation, an imperative skill for any autonomous vehicle. The proposed solution to the outdoor localization problem is based on the combination of the current state-of-the-art GNSS technology with data from inertial sensors, encoders and stereo camera, where the fusion of multiple information is performed by means of a Kalman filter algorithm with the purpose of reducing the estimation uncertainty. In addition to the system development and validation, this work aims at showing the potential of sensor fusion for localization purposes. After a general introduction on autonomous mobile robots and on the localization problem, this document covers the theory behind the sensors and techniques at the core of the system, to conclude with the project development and the validation results.

Contents

List of Figures

Chapter 1

Introduction

1.1 Autonomous Mobile Robots

Autonomous Mobile Robots (AMRs) identify those computer-controlled machines that are able to move and accomplish complex tasks in complete autonomy, capable of making decisions and react based on the environmental stimulus they perceive, without any human interaction. The rapid increment in popularity in recent years makes the field of mobile robotics more relevant than ever before[1]. The recent progress made in the field of machine learning and computer vision allows mobile robots to improve in tasks such as object recognition[2] or autonomous navigation[3], making them suitable for more and more applications. AMRs are often used to substitute humans for repetitive or dangerous tasks in many different fields like industrial automation, transportation (humans or goods), construction, exploration, agriculture, medical care or any operation in hazardous or extreme environments, but also for simpler tasks like house cleaning or house monitoring. To perform its tasks, any AMR must have a number of sensors, either mounted on the robot or external sensors placed on the environment. Sensors can be categorized as proprioceptive or exteroceptive and active or passive. Proprioceptive sensors

measure values internal to the robot, like motor speed from encoders or angular velocity from a gyroscope, while exteroceptive ones measure external quantities from the environment, like the distance of the robot from a wall measured by an ultrasonic sensor. Active sensors measure the environmental response to the energy they emit like sonars or radars, while passive ones measure the received energy coming directly from the environment, like thermometers or touch sensors. At the basis of autonomous mobile robotics there are the following four fields: perception, cognition, locomotion and navigation.

- *Perception* is the ability to correctly interpret the surrounding environment from the sensors measurements. The information coming from sensors must be properly perceived in order to execute complex tasks such as localization or object detection.

- *Cognition* is the ability to properly take the actions required for the robot to complete its tasks, based on the perceived information.

- *Locomotion* is the act of the robot to move itself. It depends on the mechanical characteristics of the robot but also on its maneuverability, stability, on terrain conditions and so on. Based on the medium the robot moves it can be defined as an aerial, water or ground vehicle. Ground robots are generally wheeled, legged or hybrid, a characteristic that strongly impacts its locomotion.

- *Navigation* is the capability to successfully reach a destination on the map. In order to do so the robot must know where it is, where the destination is and how to reach it. It is evident how navigation, other than on the three fields reported above, strongly depends on aspects like localization, path planning and obstacle avoidance.

An efficient path planning algorithm is mandatory for an effective autonomous

navigation. Finding the best route in terms of shortness and simplicity can be very challenging, and parameters like length of the path, energy consumption and required time also affects the possible applications of the robot. Path planning differentiates in global algorithms for static or dynamic but known environments and local algorithms, used in dynamic and partially unknown environments. An important prerequisite for navigation is a collision-free trajectory, since collision with objects in the environment would compromise the achievement of the target destination, hence the obstacle avoidance aspect is vital for mobile robots. Fundamental requirement for navigation is the knowledge of the robot position and orientation in the map. A proper localization system must be able to continuously provide a reliable estimate of the robot's pose and orientation in order to achieve any navigation goal.

Path planning, obstacle avoidance and localization are some of the most challenging aspects in the design of mobile robots[4], making autonomous navigation one of the most difficult tasks for AMRs.

1.2 The Localization System

As stated in [5] the navigation system is the most important aspect in the design of autonomous mobile robots. In order to successfully navigate to a target destination, the robot must be able to localize itself on the map, hence the localization system represents an imperative part in the design of AMRs. The final goal of a localization system for a mobile autonomous robot is to allow the latter to autonomously and efficiently locate itself. The location of the robot is meaningful only if defined with respect to a reference. Such a reference is usually the origin of the map in which the robot is supposed to move, but it can also be the Earth's reference frame in case of global localization or it can be an arbitrary point on the map, stationary or not. Consider for example an application for which the robot

must follow a human; in this scenario the localization system should provide the position relative to the moving target in addition to the one relative to the stationary reference. Localizing the robot means to provide the position and orientation information with respect to the reference system. To effectively achieve this goal in complete autonomy the robot needs to be equipped with proper sensors which measure some quantities useful to localize it. All the data coming from the sensors must then be processed in some way to obtain the position and orientation of the robot in the map. The quality of the localization system's output directly depends on which sensors are used, on their quality and on how their measurements are combined.

1.2.1 Uncertainty in Localization

As stated above, the localization system estimates position and orientation of the robot starting from the information perceived by the sensors. The output of the system is an estimate of the true value whose accuracy depends first on the uncertainty the robot has to face when operating in the real world and second on the way it exploits the information perceived through the sensors. Uncertainty arises from the following sources:

- *Sensors*: sensors have limitations in range and resolution that make them unable to perfectly measure the true value of the target quantity. Moreover, sensors are always subject to noise which causes unpredictable errors in measurements and some sensors are also prone to nonuniqueness of sensor readings, or *sensor aliasing*.

- *Robot hardware*: the robot itself can be a source of uncertainty. Motors, for example, can be more or less accurate; the same control input may cause different motion resulting in different robot movements.

- *Mathematical models*: mathematical models are always abstractions of the real world. These approximated models, if used to predict the next state of the robot, produce approximated results.

- *Environments*: the real world always introduces uncertainty being highly dynamic and unpredictable.

Even if it is not possible to completely eliminate them, some of the uncertainties can be reduced. Higher quality sensors and actuators, for example, limit the uncertainty in the system. In the following we will introduce the idea of sensor fusion and how it can help limit the uncertainty of the final estimation. Later in the paper we will cover this topic in greater details.

1.2.2 Sensors and Techniques

Mobile robot localization can be achieved by different methods and techniques, some of which are more suitable in certain scenarios than others. Depending on the type of sensors on which these methods are based, localization can be divided into two categories: relative and absolute localization[6].

- *Relative localization* techniques - also called *dead reckoning* - estimate both position and orientation by integrating the information from one or more sensors continuously in time, starting from the initial state of the robot. Dead reckoning is reliable only for a short period of time since it is subject to cumulative errors. The estimated position and orientation are the results of multiple integrations of measurements that, as previously said, are not error-free, and therefore all the errors accumulate in the final estimation decreasing the accuracy over time. In mobile robotics the estimation of position from motion sensors is often called *odometry*, where measurements of velocities and accelerations usually come from encoders and/or inertial sensors.

- *Absolute localization* techniques instead estimate the robot position and orientation directly from the environment. No integration is performed and therefore the estimation error does not depend on time or covered distance. Nevertheless, such techniques often have limitations in frequency and are location-dependent. Some of these methods are based on global positioning systems, active beacons, magnetic compasses, landmark navigation or model matching.

Another important distinction can be done in terms of indoor or outdoor localization. While some techniques are completely independent from that, some others obtain good performance depending on the environment. In the following we will quickly present some techniques and relative sensors commonly used to localize mobile robots, covering advantages and disadvantages of each of them.

Odometry

Odometry is a simple technique exploiting mathematical equations to predict the next position and orientation of the robot based on the previous state and on velocity and acceleration measurements. Consider for example the two-wheel robot moving on a two-dimensional plane in figure 1.1. Both linear and angular velocity can be measured by means of wheel encoders and used in simple kinematic equations like the following:

$$x_{t+\Delta t} = x_t + cos(\theta_t) \cdot v_{x,t} \cdot \Delta t + 1/2 \cdot cos(\theta_t) \cdot a_{x,t} \cdot \Delta t^2 \tag{1.1}$$

$$y_{t+\Delta t} = y_t + sin(\theta_t) \cdot v_{x,t} \cdot \Delta t + 1/2 \cdot sin(\theta_t) \cdot a_{x,t} \cdot \Delta t^2 \tag{1.2}$$

to estimate the next position of the robot, whose accuracy not only depends on the measurements errors, but also on the frequency at which the sensors work since velocity and acceleration vary continuously in real word. The obtained error can be categorized as systematic or unsystematic error. Systematic errors derive

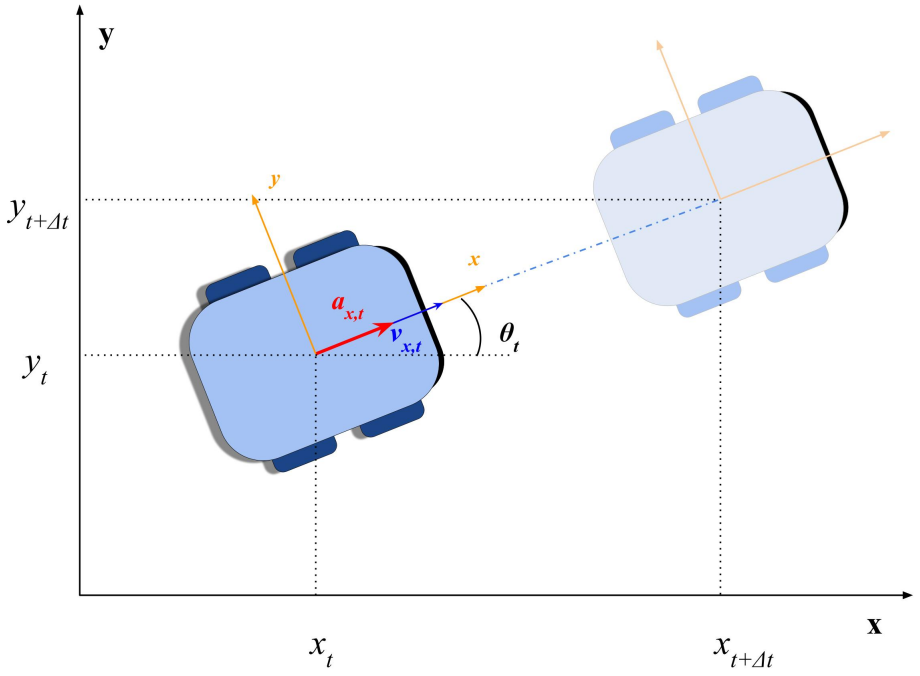

Figure 1.1. Position estimation in a 2D plane through odometry

from mechanical imperfection of the robot like differences between the wheels or from uncertainty in wheel diameters or wheel-to-wheel distance. Unsystematic errors are instead due to unpredictable interactions of wheels with the ground, like bumps or slippage. In such a system the largest error contribution usually comes from a poor precision in the orientation estimation from encoders. In a two-dimensional plane even few degrees of error cause a large displacement in the estimated pose even after short distances. For this reason inertial sensors are often exploited in addition to encoders for odometry. Even if still subject to drift problems, much more accurate orientation information can be estimated from accelerometers and gyroscopes and used to limit the error also in the pose. Combining accelerometer and gyroscope, Inertial Measurement Units are often exploited in mobile robotics for odometry. Despite the problem of unbounded cumulative errors, odometry remains an inexpensive way largely used in mobile

robot localization, often combined with absolute localization methods to correct eventual errors.

Active beacons

High reliability and high sampling rate, along with the minimal processing required for position estimation, have made active beacons commonly used in localization of ships, airplanes and robots, despite the costs for installation and maintenance. Active beacons usually broadcast radio or ultrasonic signals allowing the receivers to detect where they are and to estimate their own position. Trilateration and triangulation are the methods used to estimate the position from beacons.

- *Trilateration* consists in the estimation of the receiver position based on the measured distances from the known beacons positions. For a three-dimensional estimation a minimum of 4 transmitting beacons must be precisely placed in the environment, while the receiver is placed on the robot or vehicle to localize. Deriving the distances from the measured time-of-flight, this information is used to compute the intersection between multiple spheres and to obtain the area (ideally the point) where the robot stands.

- *Triangulation* consists in the estimation of the receiver position and orientation based on the measured angles between the robot's longitudinal axis and the beacons. By detecting at least three beacons, the respective angles can be used to estimate the x and y coordinates of the receiver in a two-dimensional map and its heading. Usually a rotating sensor mounted on the vehicle is used for this purpose.

The main drawback of both triangulation and trilateration is that, in order to work, a minimum number of beacons simultaneously detected by the robot is required. Moreover, the accuracy of the estimation depends on the geometry of

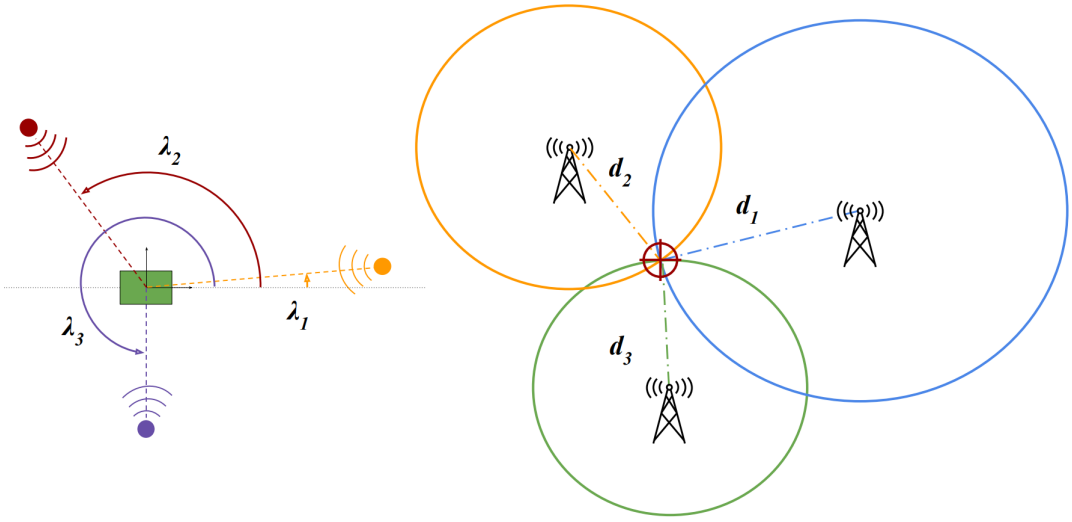

Figure 1.2. Two-dimensional triangulation (left) and trilateration (right)

the transmitters with respect to the receiver. Minimum number of beacons and "good" geometry are not always easy to satisfy, especially for indoor or large environments, or for low cost solutions where the overall number of beacons on the map is limited.

Global positioning

The Global Navigation Satellite System exploits different constellations of satellites to allow any receiver on Earth to estimate its own position in terms of latitude, longitude and altitude through trilateration. The satellite-receiver distance is estimated from the travel time of a RF signal broadcasted by the satellite, which also carries additional location and timing information. The GNSS is currently the most used positioning system in outdoor environments. It is largely used for navigation purposes by ground vehicles (including mobile robots), aircrafts, spacecraft, boats and ships, but also for land surveying, precision agriculture, geology and so on. Being a fundamental part of this work, GNSS technology will be covered in detail in chapter 2.

Magnetic compasses

Magnetic compasses provide a measure of absolute heading by measuring the Earth's magnetic field. These sensors are very important in navigation since an absolute reference heading would solve the problem of inaccurate orientation estimated from dead reckoning approaches. The main disadvantage of magnetic compasses is that they are subject to any close magnetic interference. Moreover, the Earth's magnetic field is distorted around steel structures or objects. Hence, magnetic compasses can only be used outdoors and may provide inaccurate measurements when mounted on mobile robots close to electrical components.

Landmark navigation

In landmark navigation the robot exploits distinct features of the environment to localize itself. Landmarks are fixed in known positions and may include additional information in form of bar code or QR code for example. Landmarks are usually categorized as natural and artificial ones. Natural landmarks are features already present in the environment with a primary function different from the robot navigation, like ceiling light or corridor edges for example. With respect to artificial ones, they do not modify the surrounding environment. Artificial landmarks are instead specifically designed for helping the robot navigation and are typically much easier to be recognized and more effective for position estimation, since they can be strategically placed in the environment and both shape and size are known. Regardless of the type of landmarks, the robot must store in advance their position and characteristics to be able to recognize them and estimate its own position. Moreover, the ability to recognize them and the accuracy of the estimation depends on aspects like the ambient light or the distance and angle between the robot and the marker. Landmark navigation is based on computer-vision capabilities requiring usually more processing power with respect to other

methods.

Model matching

In localization, model matching is a technique used by the robot to find its position and orientation in the map by comparing what the robot "sees" against a global map stored in memory. Model matching requires the robot to already have a map model stored in memory or to be able to build it while in operation. Map matching is then performed comparing the environmental features perceived from the sensors with the known map, and if a match occurs, the robot computes its position and orientation. A map-based localization algorithm does not require a perfect match, but it provides its belief on position and orientation. The process of simultaneously building the map while localizing the robot on it is called Simultaneous Localization And Mapping. SLAM algorithms have been the subject of much research in the past years, but thanks to the recent improvements in computer processing speed and high quality sensors such as cameras or LiDAR, this technique is nowadays successfully used in many fields. Model matching techniques require sensors with good accuracy to be able to properly extract map features, and work best in the case of easily distinguishable and stationary attributes. For these reasons, in mobile robot localization SLAM gives best results in indoor environments where the map is usually not too large and full of features.

So far we have introduced the most common techniques and the relative sensors used to solve the localization problem in autonomous mobile robots, highlighting the main strengths and weaknesses of each of them. What emerges from this paragraph is that there exists no unique sensor or technique which guarantees a reliable localization, since each of them is always subject to some kind of uncertainty. Some of the methods presented before only partially solve the localization

problem, properly working only with time or environmental constraints. In the following paragraph we introduce and justify the need for sensor fusion, an efficient solution to the uncertainty problem in localization.

1.3 The Need for Sensor Fusion

All the localization methods presented in the previous paragraph rely on sensors that are subject to some kind of uncertainty. Odometry is not reliable for long operation due to accumulation of systematic and random errors from encoders or inertial sensors. Trilateration or triangulation strongly depend on the geometry and number of beacons deployed on the map, and on certain environments their visibility from the robot is difficult to guarantee for limited budget solutions. GNSS localization is great for outdoor spaces but for precision localization it suffers in proximity of tall buildings or any kind of object obstructing the sky visibility. Landmark navigation alone is not suitable for continuous localization since it is subject to distance and angle constraints of the robot from markers. Model matching techniques are largely used for indoor mobile robots localization, but their performance drops in case of strongly dynamic or wide outdoor environments. Vision SLAM for example is limited by high computational costs and it is not as precise as LiDAR based SLAM, which on its side is much more expensive. Even if none of these systems guarantee a continuous and reliable localization regardless of the environment, a combination of them can overcome the uncertainty problem that each sensor or technique is subject to. If we want to improve the estimation accuracy a *sensor fusion* approach is needed, since it combines data coming from multiple sensors to reduce their uncertainty contribution to the final estimation. A known definition of data fusion has been provided in [7]: "*Data fusion techniques combine data from multiple sensors and related information from associated databases, to achieve improved accuracy and more specific inferences than*

could be achieved by the use of a single sensor alone". The integration of multiple sensors allows to compensate for the weaknesses of one with the strengths of another. For these reasons sensor fusion algorithms are widely used in robotics applications such as localization, object recognition and environmental mapping. Sensor fusion is the core of the localization system discussed in this thesis and it will be covered in greater detail in chapter 4.

1.4 Thesis Goal

This work is part of a bigger project of Innotech Systems L.L.C., a startup born in Los Angeles with the mission of providing Artificial Intelligence based solutions and services for a wide range of applications in different fields like robotics, transportation and healthcare. Innotech Systems L.L.C. is working on an autonomous mobile robot platform for delivery services of food, beverages, packages etc. in defined areas like airports, university campuses and so on, where the robot is required to move in complete autonomy. To allow such application, an efficient and reliable localization system is needed. For this reason, the thesis is focused on the design and development of the outdoor localization system for an assistive mobile robot. The core of the system consists of a sensor fusion node combining data from multiple sensors in order to estimate the position and orientation of the robot in the environment with the highest possible accuracy. Exploiting a state-of-the-art GNSS receiver for mobile applications as the main source of absolute measurements, the system also makes use of inertial sensors, encoders and cameras to make up for the GNSS weaknesses and to guarantee a continuous and reliable localization service. The goal of the thesis is not only to develop a working localization system, but also to show with real results how the localization problem can benefit from a multiple sensor fusion approach.

1.5 Thesis Organization

In this first chapter we provided an introduction on autonomous mobile robots and on the concept of autonomous navigation, focusing on the importance of the localization system and presenting some of the related techniques commonly used. Then we introduced the key role of sensor fusion and the goal of this thesis. Being a fundamental part of this project, in chapter 2 we cover the theory behind the GNSS technology, how the user position is derived from the signals coming from the satellites, the error sources affecting the estimation and the augmentation systems used to obtain high accuracy measurements. In chapter 3 we introduce the inertial sensors and how their measurements are combined to estimate the orientation of the device. In chapter 4 we explain the concept of sensor fusion and we go through the Kalman filter algorithms implemented in the localization system. Finally, in chapter 5 we describe the project development, the software and the hardware used to design the robot and the localization system, and we describe the tests used to validate the system, presenting the relative results. The thesis concludes with final considerations on the project and on possible future related works.

Chapter 2

Global Navigation Satellite System

Global Navigation Satellite System refers to the set of constellations of medium Earth orbit satellites that provide positioning and timing services on a global or regional basis. At the time of writing, the constellations providing global coverage are the United States' GPS, that with its 31 satellites currently in orbit is the world's most utilized satellite system, the Russian GLONASS, the European Galileo and the Chinese BeiDou, while the Indian NavIC is an autonomous system designed to provide coverage for the Indian region and the surrounding area, and the Japanese QZSS is used to complement the GPS to improve coverage in East Asia and Oceania. GNSS allows electronic receivers to derive their position in terms of latitude, longitude and altitude by using time signals transmitted on radio channels by the satellites. The signals also allow to precisely calculate the current local time. The achieved position accuracy may go from several meters in case of bad conditions (obstruction of the sky, low quality receiver ...) down to centimeter level with the current state-of-the-art technology.

The Global Navigation Satellite System exploits the concept of time-of-arrival (TOA) to determine the user position. The receiver estimates the propagation time of the signal coming from satellite (SV), then it multiplies it by the speed of light to derive the SV-to-user distance. Since the satellites positions are precisely known by the receiver thanks to the modulated data carried by the signals, through trilateration the user is able to estimate its position when a sufficient number of measurements are available. The fact that a satellite is theoretically visible does not mean that it is actually used for obtaining the position; the presence of obstacles or low quality of the signal may cause the receiver not to successfully process the signal. In theory, with at least four satellites in visibility, by intersecting the four spheres derived from the respective SV-to-user distances, the user is able to identify the point in the three-dimensional space corresponding to its position, as shown in figure 2.1. In practice, as we will see later, many sources of error affect the estimation process and usually more satellites are required to achieve a reasonable accuracy.

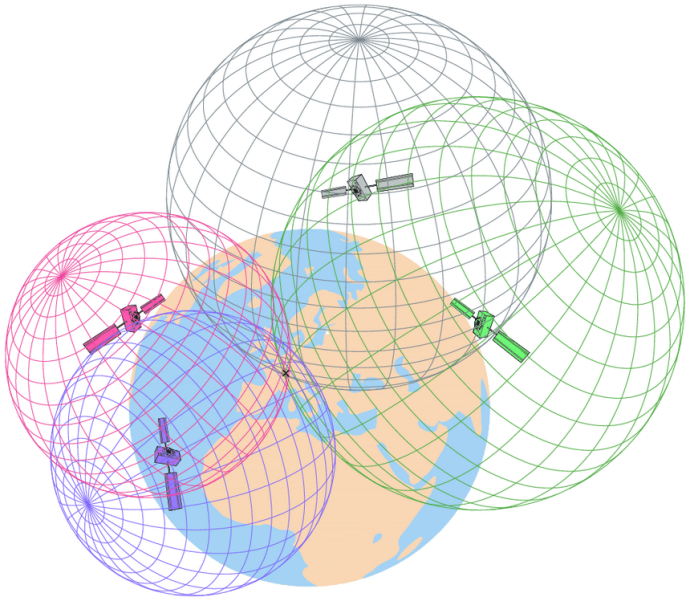

Figure 2.1. Three-dimensional trilateration with four satellites [8]

GNSS is composed of three segments: space, control and user segment.

- The Space segment is the constellations of satellites used as reference points in the space for the user positioning estimation. Each satellite broadcasts a PRN coded signal (pseudorandom noise) modulated to carry the satellite's position and timing information, from which the receiver derives the SV-to-user distance. From the user point of view the GNSS is a passive system, since the receiver passively receives the transmitted signals from the satellites, allowing an unlimited number of users to simultaneously utilize the service.

- The Control segment consists in a network of monitoring stations distributed all around the Earth responsible for maintaining the status of the satellites and signals. The satellite's orbital position, battery power level, clock and ephemeris (satellite's trajectory) are only some of the parameters that need to be periodically monitored by the control segment. Frequent updates of parameters like clock and ephemeris help in reducing the space segment and control segment contribution to the range measurement error. Furthermore, the control segment activates spare satellites when needed to maintain system availability and resolve possible anomalies. To accomplish all these tasks, the control segment comprises three components: the Master Control Station, monitor stations and ground antennas.

- The User segment is made of a wide range of different receivers with different performance levels, which estimate the position, velocity and time - PVT - of the user on the basis of the signals transmitted by the satellites.

Let us step a little deeper in the process of PVT estimation.

2.1 The Position Velocity Time solution

Above we briefly introduced the trilateration process to estimate the user position by intersecting multiple spheres, assuming to be able to precisely compute the SV-to-user distance neglecting all the possible errors. In reality this is never the case, and a different number of error sources affect the range measurements, like synchronization, ephemeris, relativistic, multipath errors and so on. The error introduced by non-synchronized clocks is generally much bigger than the other contributions, so to simplify the concept of range estimation here we neglect all the other sources of error, which will be covered in the next section.

Position

We want to determine the position of the user with respect to the Earth's reference frame. Reference frames will be discussed in detail in section 2.3, but for now we only need to know that we are interested in finding the coordinates of the user with respect to the ECEF frame, a Cartesian system centered at the center of the Earth. Our unknowns are then the coordinates (x_u, y_u, z_u) of the user, while the satellite position given by (x_s, y_s, z_s) with respect to ECEF are derived from ephemeris data broadcasted by the satellite. The SV-to-user range is derived from the propagation time of the signal in space. Taking as an example the baseline civil GPS service, each transmitted signal consists of a BPSK modulated binary sequence (named Coarse/Acquisition code) over the same carrier frequency, and each satellite is identified by a different code (CDMA scheme). Two ways to measure the propagation time are possible:

- *Code phase measurement*: the propagation time between SV and user is estimated measuring the Δt between a local replica of the C/A code and the received signal in space. This process is illustrated in figure 2.2. For example,

a specific code phase transmitted by the satellite at time t_1 gets at the receiver at t_2. The receiver generates an identical code replica at time t_1 with respect to user time, which is shifted in time until it achieves correlation with the satellite-generated code. If the receiver and SV were perfectly synchronized, the correlation process would give the true propagation time Δt. Multiplying it by the speed of light, the SV-to-user distance is obtained.

- *Carrier phase measurement*: with this method, the receiver evaluates the phase difference between the local carrier and the received one. We will talk about carrier phase measurements when covering the RTK technology.

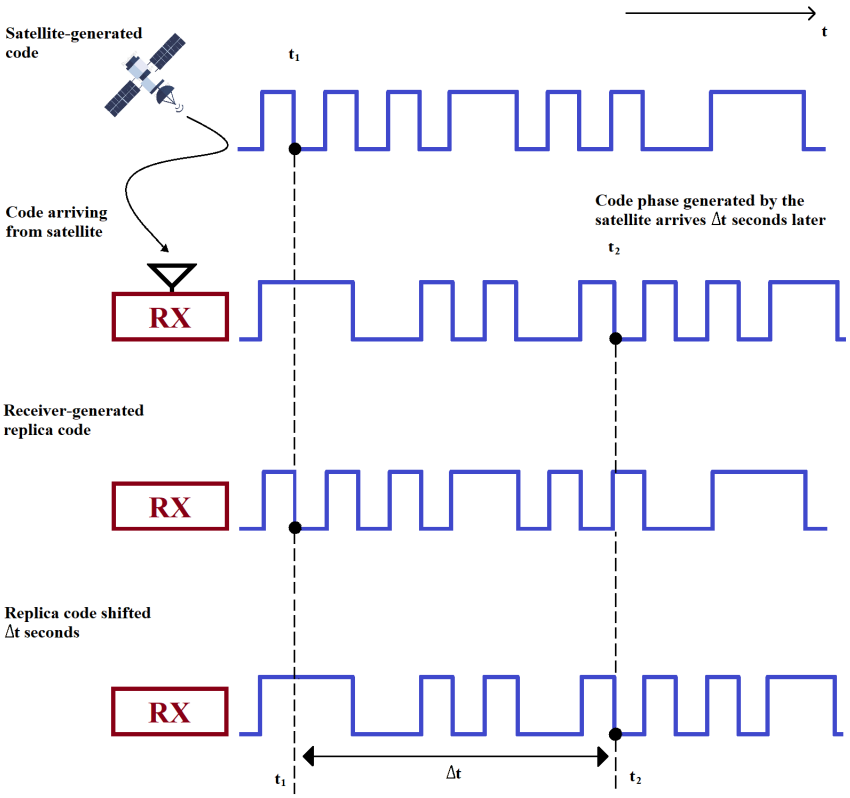

Figure 2.2. Use of replica code to determine satellite transmission time

The receiver and satellite clocks are not generally synchronized. Both clocks

have an offset from the ideal GNSS time. Figure 2.3 shows the time relationships of the range measurements, where:

T_s is the system time at which the signal left the satellite

T_u is the system time at which the signal reached the user receiver

δt_s is the offset of the satellite clock from system time

δt_u is the offset of the receiver clock from system time

$T_s + \delta t_s$ is the satellite clock reading at the time that the signal left the satellite

$T_u + \delta t_u$ is the user clock reading at the time the signal reached the user receiver

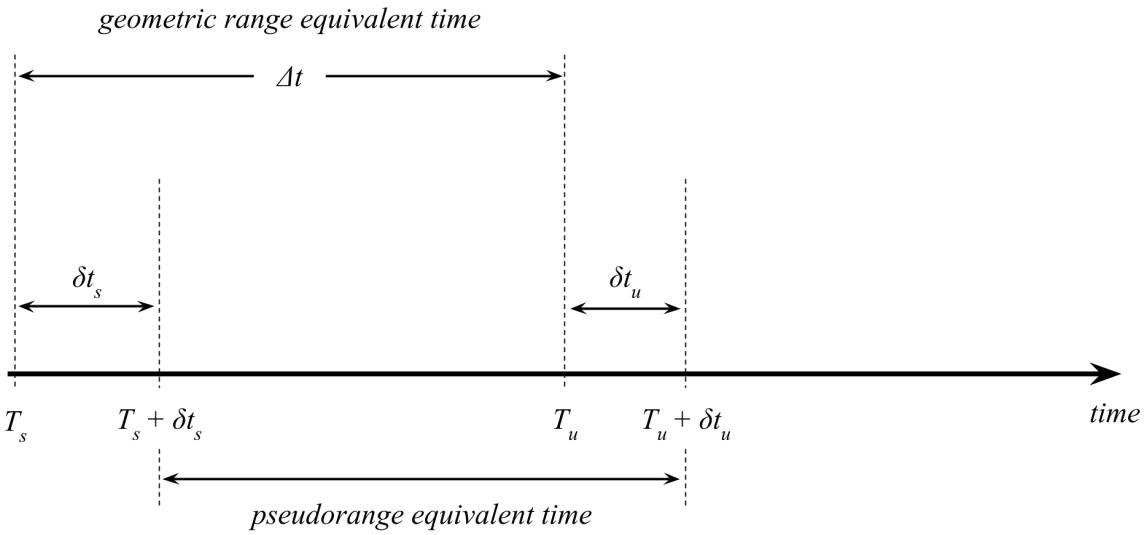

Figure 2.3. Time relationship of range measurements

While the satellite time offset δt_s is known by the control segment and then can be corrected, the receiver one can not. For this reason, we will assume δt_s negligible and we will only consider the receiver time offset δt_u in the following discussion. The measured distance is then different from the true geometric one and for this reason it is called *pseudorange*:

$$\rho = c \cdot (\Delta t + \delta t_u) \tag{2.1}$$

The user, by measuring at least four pseudoranges from four satellites (with known coordinates), can determine four unknowns: the user coordinates (x_u, y_u, z_u) and the user clock bias δt_u. Considering the generic j^{th} satellite, the corresponding pseudorange is given by:

$$\rho_j = \sqrt{(x_j - x_u)^2 + (y_j - y_u)^2 + (z_j - z_u)^2} + b_{ut} \tag{2.2}$$

where (x_j, y_j, z_j) is the satellite position (center of the pseudo-sphere), ρ_j is the pseudorange (radius of the pseudo-sphere), and b_{ut} is equal to $c \cdot \delta t_u$, the range bias due to the clock bias. To find the four unknowns, four equations of this kind are needed, so four satellites are the minimum required for this estimation, and they must of course be in *line-of-sight* with the receiver. For a more precise estimation more satellites and more frequencies can be used (multi-constellation and multi-frequency).

The non-linear equation above can be simplified considering the large SV-to-user distance ($R \geq 20000km$) to reduce the computational complexity. The pseudo-sphere equation can be linearized by approximating it through the Taylor expansion around a known location/time with coordinates $(\hat{x}_u, \hat{y}_u, \hat{z}_u, \hat{\delta t}_u)$. The trilateration process can be transformed in the intersection of the planes tangent to the pseudo-sphere in the user position.

$$\hat{\rho}_j = \sqrt{(x_j - \hat{x}_u)^2 + (y_j - \hat{y}_u)^2 + (z_j - \hat{z}_u)^2} + \hat{b}_{ut} \tag{2.3}$$

$$\begin{aligned}\rho_j = \hat{\rho}_j + \frac{\partial \rho_j}{\partial x_u}_{|x_u=\hat{x}_u} \cdot (x_u - \hat{x}_u) + \frac{\partial \rho_j}{\partial y_u}_{|y_u=\hat{y}_u} \cdot (y_u - \hat{y}_u) \\ + \frac{\partial \rho_j}{\partial z_u}_{|z_u=\hat{z}_u} \cdot (z_u - \hat{z}_u) + \frac{\partial \rho_j}{\partial b_{ut}}_{|b_{ut}=\hat{b}_{ut}} \cdot (b_{ut} - \hat{b}_{ut})\end{aligned} \tag{2.4}$$

Defining

$$\Delta x_u = x_u - \hat{x}_u \quad , \quad \Delta y_u = y_u - \hat{y}_u \quad , \quad \Delta z_u = z_u - \hat{z}_u$$

$$\Delta\rho_j = \hat{\rho}_j - \rho_j \quad , \quad \Delta b_{ut} = b_{ut} - \hat{b}_{ut}$$

we get the linearized equation which is function of the displacement with respect to the approximation point:

$$\Delta\rho_j = a_{xj}\Delta x_u + a_{yj}\Delta y_u + a_{zj}\Delta z_u - \Delta b_{ut} \tag{2.5}$$

where

$$a_{xj} = \frac{x_j - \hat{x}_u}{\sqrt{(x_j - \hat{x}_u)^2 + (y_j - \hat{y}_u)^2 + (z_j - \hat{z}_u)^2}}$$
$$a_{yj} = \frac{y_j - \hat{y}_u}{\sqrt{(x_j - \hat{x}_u)^2 + (y_j - \hat{y}_u)^2 + (z_j - \hat{z}_u)^2}}$$
$$a_{zj} = \frac{z_j - \hat{z}_u}{\sqrt{(x_j - \hat{x}_u)^2 + (y_j - \hat{y}_u)^2 + (z_j - \hat{z}_u)^2}}$$

The four unknowns $\Delta x_u, \Delta y_u, \Delta z_u, \Delta b_{tu}$ can be found by making ranging measurements to four satellites and by solving the following linear system:

$$\Delta\rho_1 = a_{x1}\Delta x_u + a_{y1}\Delta y_u + a_{z1}\Delta z_u - \Delta b_{ut}$$

$$\Delta\rho_2 = a_{x2}\Delta x_u + a_{y2}\Delta y_u + a_{z2}\Delta z_u - \Delta b_{ut}$$

$$\Delta\rho_3 = a_{x3}\Delta x_u + a_{y3}\Delta y_u + a_{z3}\Delta z_u - \Delta b_{ut}$$

$$\Delta\rho_4 = a_{x4}\Delta x_u + a_{y4}\Delta y_u + a_{z4}\Delta z_u - \Delta b_{ut}$$

Assuming to use n satellites, the following matrix equation holds:

$$\boldsymbol{\Delta\rho} = \boldsymbol{H}\boldsymbol{\Delta x} \tag{2.6}$$

where

$$\boldsymbol{H} = \begin{bmatrix} a_{x1} & a_{y1} & a_{z1} & 1 \\ a_{x2} & a_{y2} & a_{z2} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ a_{xn} & a_{yn} & a_{zn} & 1 \end{bmatrix} \quad , \quad \boldsymbol{\Delta\rho} = \begin{bmatrix} \Delta\rho_1 \\ \Delta\rho_2 \\ \vdots \\ \Delta\rho_n \end{bmatrix} \quad , \quad \boldsymbol{\Delta x} = \begin{bmatrix} \Delta x_u \\ \Delta y_u \\ \Delta z_u \\ -\Delta b_{ut} \end{bmatrix}$$

Since we are interested in finding the values $\boldsymbol{\Delta x}$, the solution is given by:

- if $n = 4$: $\boldsymbol{\Delta x} = \boldsymbol{H^{-1}\Delta\rho}$
- if $n > 4$: the solution is given by $\boldsymbol{\Delta x}$ that minimizes the square of the residual $\boldsymbol{R_{SE}}(\boldsymbol{\Delta x}) = (\boldsymbol{H\Delta x - \Delta\rho})^2$. Differentiating with respect to $\boldsymbol{\Delta x}$, the gradient $\boldsymbol{\nabla R_{SE}}$ is set to zero to search for the minimum value (Least Square solution). The final solution, provided that $(\boldsymbol{H^T H})^{-1}$ is non-singular, is:

$$\boldsymbol{\Delta x} = (\boldsymbol{H^T H})^{-1}\boldsymbol{H^T\Delta\rho} \tag{2.7}$$

A GNSS receiver solves this equation by recursive methods.

Velocity

GNSS allows the user to compute its three-dimensional velocity $\dot{\boldsymbol{x}}$. In some receiver the user velocity is computed as an approximate derivative of the position:

$$\dot{\boldsymbol{x}} = \frac{d\boldsymbol{x}}{dt} \approx \frac{\boldsymbol{x}(t_2) - \boldsymbol{x}(t_1)}{t_2 - t_1} \tag{2.8}$$

but this approach is reliable only if the velocity stays constant during the time interval and only if the errors in the positions are smaller with respect to the difference $\boldsymbol{x}(t_2) - \boldsymbol{x}(t_1)$. A more accurate method exploits the fact that the relative motion of the satellite with respect to the user causes a Doppler shift in the signal received by the receiver. This approach estimates the user's velocity from carrier phase measurements, from which the Doppler frequency of the received satellite signals can be precisely estimated [9].

Time

GNSS also provides the capability for time synchronization of users worldwide. Other than the synchronization needed for position and velocity estimations, GNSS time synchronization is very important for different applications in telecommunications, power grid or financial market for example. In section 2.3.2 we will talk about the time reference frame used for time synchronization.

To keep it as clear as possible, in this section we have described the range estimation process neglecting all the error sources except the receiver clock offset. In the following we will see that in reality the whole process is subject to many impairments which may introduce a significant error on the position estimate.

2.2 Error sources

From its generation to its reception the transmitted signal always experiences several impairments which produce variations on delay and power attenuation. These variations produce the pseudorange error and are caused by the following phenomena:

- Satellite generation impairments, which include ephemeris errors (difference between the expected and actual orbital position of the SV), on board clock biases, errors in code generation ...
- Satellite antenna gain pattern.
- Propagation losses and atmospheric impairments, which include:
  - Ionosphere effect: the propagation delay depends on the frequency and on the density of electrons along the path.
  - Troposphere effect: the propagation delay depends on the pressure, temperature, humidity of the air.
  - Multipath effects: reflections of the signals due to close obstacles.
- Interference.
- Receiver antenna gain pattern.
- Reception impairments, which include measurement errors, receiver noise, uncompensated relativistic effects.

Predictable and correctable biases aside, each contribution to the pseudorange error can be modelled as a gaussian random variable with zero mean and variance σ_j^2, independent and identically distributed on the different pseudoranges. According to this hypothesis, the standard deviation of the total pseudorange error, or *user equivalent range error*, is:

$$\sigma_{UERE} = \sqrt{\sum_j \sigma_j^2} \tag{2.9}$$

Taking now into account all the other error sources, equation (2.6) becomes:

$$\boldsymbol{\Delta\rho} + \delta\boldsymbol{\rho} = \boldsymbol{H}(\boldsymbol{\Delta x} + \delta\boldsymbol{x}) \tag{2.10}$$

$\delta\boldsymbol{x}$ represents the error in the position and time estimation due to all the impairments listed above, which contributions are included in the pseudorange error $\delta\boldsymbol{\rho}$. Moreover, $\delta\boldsymbol{x}$ also depends on the geometry of the satellites with respect to the user from which the position has been derived.

Dilution Of Precision

To understand the role of geometry in GNSS position estimation we have to recall that the final estimate is given by the intersection of several spheres in the three-dimensional space. In ideal conditions this intersection corresponds to a point with the real coordinates of the user, but in reality, due to the uncertainty on the spheres radius (i.e. the pseudoranges), the intersection results in an area in which the user can be located instead of a single point. The accuracy of the estimation is directly related to the size of this area, which depends on the pseudorange errors and on the satellites-user relative geometry. Assuming the same pseudorange error, the intersection of the pseudo-spheres gives different results based on the position of the satellites with respect to the user, resulting in a different position accuracy. Figure 2.4 reports an example in a two-dimensional scenario. The idea that the position error depends on the satellites-user geometry is called *dilution of precision*. The

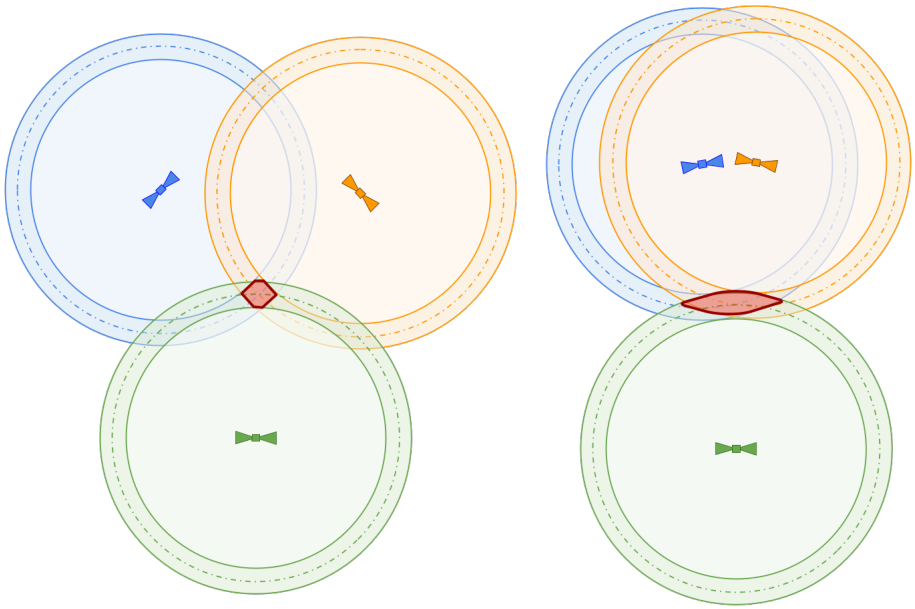

Figure 2.4. Higher accuracy, low GDOP (left) - Lower accuracy, high GDOP (right)

geometric dilution of precision (GDOP) is a parameter defined to characterized the amplification of the standard deviation of the measurements error caused by the geometry. Other DOP parameters are used to characterized the accuracy of various components of the PVT solution. Position dilution of precision (PDOP), horizontal dilution of precision (HDOP), vertical dilution of precision (VDOP) and time dilution of precision (TDOP). For details on the DOP derivation refer to [9].

Ephemeris error

In order to properly estimate its position, the user must know the location of the reference points in visibility. Since the satellite is not able to provide its own position at any time instant, it broadcasts its orbital parameters - *ephemeris* - to the users, which use them to compute the position of the satellite through an orbital function $f(t)$. Ephemeris are broadcasted from the satellite about every 30 seconds, but they are updated by the control segment every 2 hours. Possible

errors in the orbital parameters bring to wrong estimation of the satellite location, which couses errors in the estimeted user position.

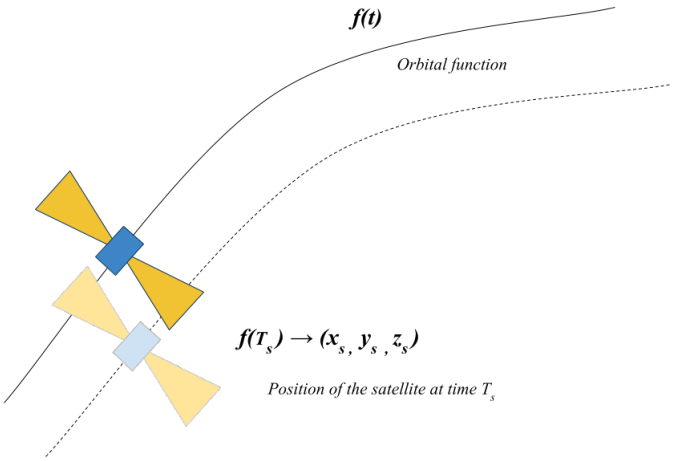

Figure 2.5. Ephemeris error

Ionosphere

The ionosphere is the Earth's upper atmosphere, from about 48 km to 965 km altitude, ionized by solar radiation. It has practical importance because, among other functions, it influences radio propagation to distant places on the Earth. Under a GNSS point of view, the ionization level of the ionosphere is measured by the *Total Electron Content*, which is defined as the number of electrons in a tube of $1m^2$ cross section from the receiver to the satellite. The TEC is usually greatest in the middle of the day and lowest at night. The strongest effect of the ionosphere on the radio signal is its speed variation, which depends on the number of free electrons along the path. This effect induces a pseudorange delay that gives a bias

$$I_\rho = \frac{40.3 \cdot TEC}{f^2} \tag{2.11}$$

measured in meters. This ionospheric error bias can be corrected at receiver side in two ways:

- it can be estimated and compensated using a double frequency receiver;
- a proper model must be employed by sigle frequency receivers, exploiting the parameters broadcatsed by the satellite.

Double frequency receiver Measuring the pseudorange at two frequancies from the same satellite we get:

$$\rho_{f1} = \rho^* + \frac{40.3 \cdot TEC}{f_1^2} \tag{2.12}$$

$$\rho_{f2} = \rho^* + \frac{40.3 \cdot TEC}{f_2^2} \tag{2.13}$$

where ρ^* is the ionosphere-free pseudorange. The set of equations can be solved for ρ^* and for TEC obtaining:

$$\rho^* = \frac{f_1^2}{f_1^2 - f_2^2} \cdot \rho_{f1} - \frac{f_2^2}{f_1^2 - f_2^2} \cdot \rho_{f2} \tag{2.14}$$

This approach cancels the bias due to ionosphere but *increases the variance of the noise* due to other error contributions.

Single frequency receiver In single frequency receivers it is not possible to obtain the iono-free pseudorange, so a model of the ionosphere has to be implemented, to estimate the iono delay depending on position and time of the day. Such a model can provide the expected delay or maps of the ionosphere and it can be either updated by the GNSS system or by some external aiding system.

Troposphere

The troposhpere is a *non-dispersive* part of the atmosphere which introduces a delay on the radio signal independently from the frequency. The troposhpere

delay consists in a dry and a wet contribution which depend on the refractive index n:

$$T = 10^{-6} \cdot \int (n-1) \cdot 10^{-6} dn = T_d + T_w \tag{2.15}$$

Relativistic effects

The satellite clock drift is affected by relativistic effects. In GPS the satellite clock frequency is adjusted so that the frequency observed by the user at sea level has the nominal value. Periodic effects due to the eccentricity of the satellite orbit introduce errors and must be taken into account:

- half of the error is due to the periodic change in the speed of the satellite relative to the ECI (Earth Centered Inertial) frame;
- the other half is caused by the satellite's periodic change in its gravitational potential:
  - at the perigee, the satellite velocity is higher and the gravitational potential is lower, and the satellite clock runs slower;
  - at the apogee, the satellite velocity is lower and the gravitational potential is higher, so the satellite clock runs faster.

Another relativistic effect is due to the rotation of the Earth during the signal transmission, and a clock on the Earth's surface will experience a rotation with respect to the reference frame during the propagation time.

Considering the error sources we have seen up to now, the receiver has to apply corrections on the raw pseudoranges for at least the following impairments: bias of the satellite clock, ionospheric delay, troposhperic delay, relativistic effect. The corrected measurements can then be used for the PVT computation.

$$Raw\, measurements \rightarrow Corrected\, measurements \rightarrow PVT\, computation$$

Some error is always present in the final estimated position, due to residual contribution of the previous sources or to sources that cannot be predicted (like receiver noise). Residual errors are modelled as gaussian random variables with zero mean and standard deviation σ_{UERE}. We can then re-write the equation (2.1) obtained from the j^{th} satellite as:

$$\rho_j = R_j + c \cdot (\delta t_u - \delta t_{s,j}) + I_{\rho_j} + T_{\rho_j} + \epsilon_{\rho_j} \tag{2.16}$$

where $R_j = c \cdot \Delta t$ is the true SV-to-user distance.

2.3 Reference Frames

In order to compute the distance between the user and the satellite it is necessary to have spatial and timing reference systems common for both. The position of the user is conventionally expressed in a coordinate system fixed to the Earth (so that a stationary object on the Earth remains fixed), while the satellite location is computed according to motion equations in an inertial system.

2.3.1 Positioning Reference Frames

The user reference system is the conventional terrestrial reference system called ECEF (Earth Centered Earth Fixed) shown in figure 2.6 in which :

- the origin is the center of mass of the Earth;
- the z-axis extends through true north (CTP - Conventional Terrestrial Pole), which does not coincide with the instantaneous Earth rotational axis;
- the x-axis intersects the sphere of the earth at 0° latitude (the equator) and 0° longitude (prime meridian in Greenwich). This means that ECEF rotates with the earth, and therefore coordinates of a point fixed on the surface of the earth do not change;

- the y-axis is in the equatorial plane completing a right-handed system.

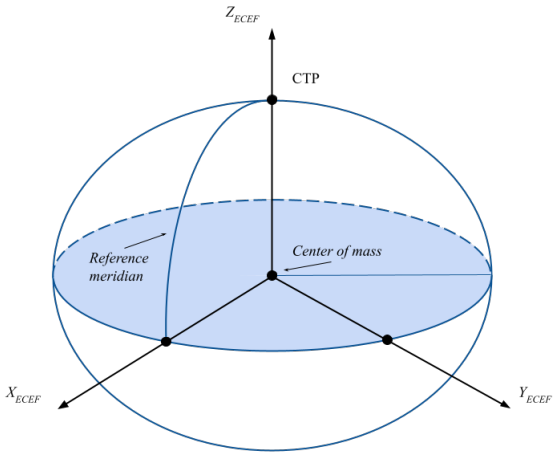

Figure 2.6. ECEF reference system

An inertial reference system is also needed for the satellite locations. The Conventional Inertial Reference System (CIRS) is realized through a catalog of position and proper motions with respect to fundamental stars and, contrary to ECEF, it does not rotate with the Earth.

GNSS provides a positioning system on a global basis, hence the user coordinates are given according to a terrestrial reference system, also called *geodetic datum*. For GPS the geodetic datum corresponds to the WGS84 datum, which includes the definition of the ellipsoidal coordinate system, the Earth gravitational model, the associated magnetic model and local datum transformations. GNSS then provides the user's position in ellipsoidal coordinates:

- *geodetic latitude* ϕ: is the angle in the meridian plane through the point P between the equatorial (x-y) plane of the ellipsoid and the line perpendicular to the surface of the ellipsoid passing in P (positive north from the equator);
- *geodetic longitude* λ: is the angle in the equatorial plane between the reference meridian and the meridian plane through P (positive east);

- *geodetic height h*: is measured along the normal to the ellipsoid through P.

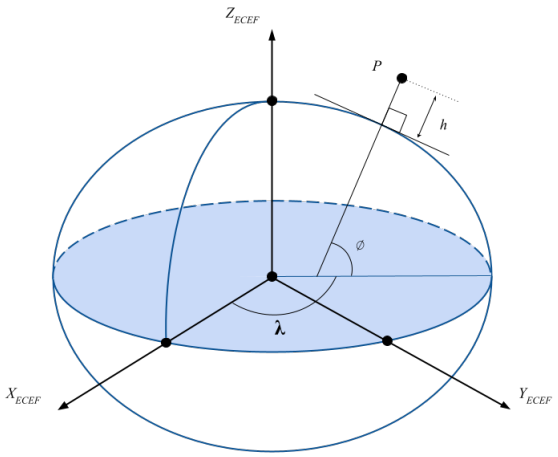

Figure 2.7. Ellipsoidal coordinates

Since the Earth's surface is not regular, the height h is defined with respect to an idealized mean see level. The surface used as the global zero reference for the height measurements is called *geoid* and it is defined as the locus of all the points with the same gravity potential best fitting the average see level globally. The geoid takes into account geological formation and topographic relief, then it is not regular and can be represented as a grid of point, mapped relative to the reference ellipsoid.

Several applications, like mobile robot localization, usually require to localize the user on a local map, a small area with respect to the Earth's surface. Local reference frames are then more suitable than ECEF or WGS84 in this scenarios. An example of local frame is the ENU frame (East for x, North for y and Up for z axis), which is a local Cartesian system with the x-y plane tangent on its origin to the Earth's surface. According the the specific application, the user's geodetic coordinates can be converted into the most suitable reference system. Figure 2.8 shows the relationship between geodetic, ECEF and ENU reference systems.

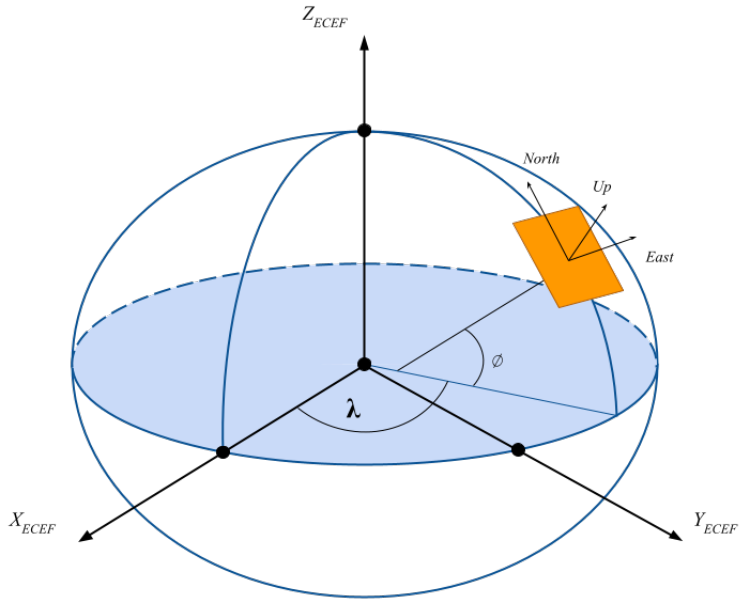

Figure 2.8. Geodetic, ECEF and ENU systems

2.3.2 Timing Reference Frames

Generation of precisely synchronised signals aboard spacecrafts and measurement of the transmission time is at the heart of GNSS. A precise and stable time scale for time-tagging the measurements is then needed. The *Coordinated Universal Time* (UTC) is a time scale based on SI seconds rearranged to keep into account the non uniform earth rotation and revolution around the sun. The difference between UTC and the Interantional Atomic Time is taken into account by introducing *leap seconds*. A specific time instant in GNSS is defined as an epoch. GNSS time is defined as counter of number of seconds from a reference date. This GNSS time has a countinuous time scale which differs from UTC (of about 18 seconds in GPS), and the bias of the satellite clock is computed relatively to this GNSS time. In GPS time for example (GPST), the zero is set at 0h, 6^{th} Janaury, 1980.

2.4 GNSS Signals

The signal broadcasted by the navigation satellites is usually denoted as *signal-in-space* and it must allow users to estimate the pseudo-distances user-SV, carry some useful data, be robust to the transmission through the atmosphere and uniquely identify the satellite. Most of the GNSS use the Code Division Multiplexing technique to obtain mutually orthogonal codes and to identify the satellite without ambiguity, so each of the SV can transmit on the same time-frequency resource. The unique code (*ranging code*) is broadcasted periodically and continuously from the satellite. GNSS systems exploit different carriers in the L band, which includes the radio spectrum from 1 to 2 GHz. L band electromagnetic waves are used for satellite navigation since they are able to penetrate rain, clouds, fog and vegetation, limiting the impact of atmospheric phenomena on the system. The signal-in-space transmitted by the satellite is a data sequence (navigation message) modulated on top of the unique ranging code at a carrier frequency f_0. The unique chip code for satellite i is a BPSK modulation of a sequence of bits of lenght p:

$$x^i_{code}(t) = \sum_{n=0}^{p-1} b^i_n r(t - nT_c) \tag{2.17}$$

where $b_n \in \{\pm 1\}$, *r(t)* is a rectangular pulse shape and T_c is the chip time (duration of one bit). Being periodically transmitted, the ranging code will then be:

$$c_i(t) = \sum_{n=-\infty}^{+\infty} x^i_{code}(t - nT_{code}) \tag{2.18}$$

with $T_{code} = pT_c$ is the chip code duration.

The ranging code $c_i(t)$ is a deterministic code on which the navigation message, carrying useful data, is modulated. It is still a BPSK modulation of a bit sequence where the bit duration T_b is an integer multiple of T_{code}:

$$d_i(t) = \sum_{n=-\infty}^{+\infty} d^i_n r(t - nT_b) \tag{2.19}$$

The unique bit sequence for the chip code must be able to uniquely identify the satellite in the constellation and must posses good autocorrelation and cross-correlation (with other chip codes) properties. Taking GPS as a reference, Gold Codes are used since they are *pseudo random noise* (PRN) codes with good autocorrelation and cross-correlation properties and can easily be generated through a *Maximal Length Linear Shift Register* implemented in digital circuit. The GPS signal-in-space consists of three components:

- Carrier, a RF sinusoidal signal with one of the designated frequencies of the L band.
- Ranging code, each SV transmits two codes: *Coarsial/Acquisition* code for civil users and *Precision* code (encrypted) for military applications.
- Data sequence, modulated on top of the ranging code, carrying the navigation message.

Figure 2.9 shows the modulation of data and C/A code on the GNSS signal.

2.5 Augmentations

The accuracy provided by the standard GNSS services is not sufficient in many important applications where the requirements may be very different. As we saw in section 2.2, the precision of the estimate depends on the pseudorange error and on the satellites-user relative geometry. A GNSS augmentation is any system external to the SV-user system which aims at the reduction of these error contributions, so the pseudorange error variance σ_{UERE} and the GDOP. A typical solution to reduce the GDOP are the so-called *pseudolites*. A pseudolite or *pseudo-satellite* is a terrestrial transmitter able to emit GNSS-like signals. The aim of the pseudolite is to give an additional reference point so to improve both *availability* and *geometry*.

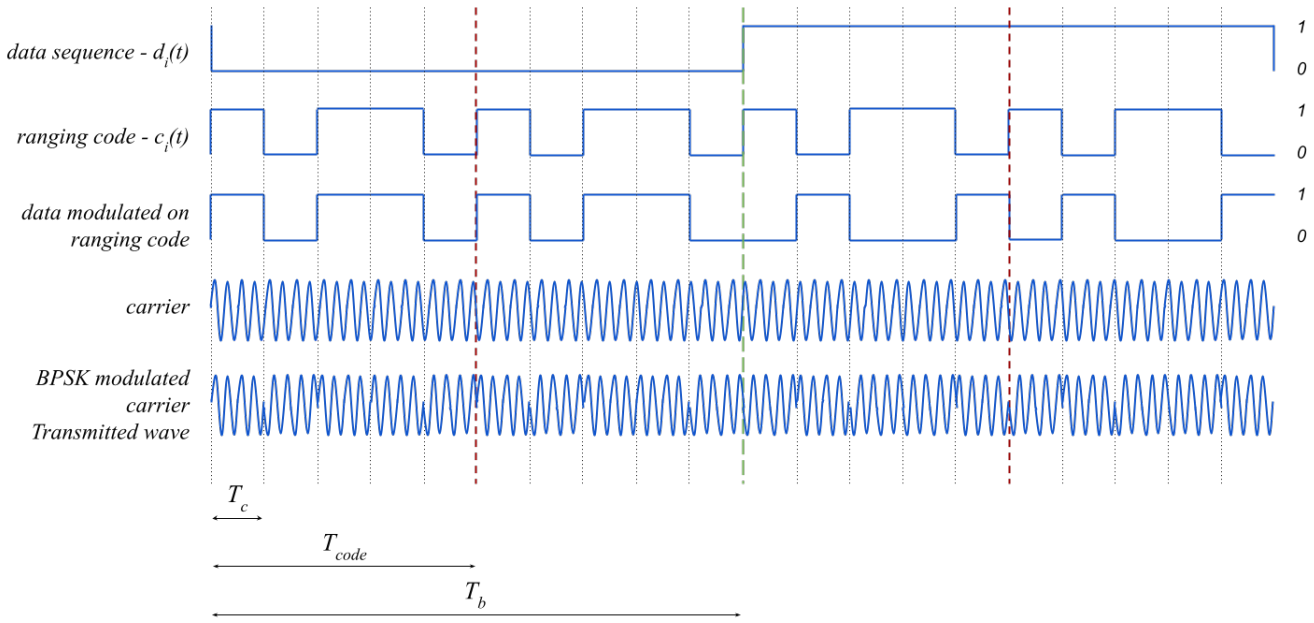

Figure 2.9. Example of a GNSS SIS modulation

Each pseudolite must be synchronized with the satellites and have its own PRN code. In addition to this, the pseudolite must be in line-of-sight with the receiver, so cannot be further than a few kilometers from the it, and should not disturb the other signals coming from other satellites (since they must operate on the same L band). This last problem is the most difficult one to solve since the pseudolite is much closer to the user than any satellite, and so its transmitted power must be carefully set to avoid any disturb on the other signals, but since the user moves, this power management should also be dynamic and very precise. For this reason pseudo-satellites are used only in specific areas, like airports, and cannot be used in crowded environments where many users require the same GNSS service.

Concerning now the reduction of the pseudorange error, excellent results are obtained with the Differential GNSS technology.

2.5.1 Differential GNSS

The fundamental idea behind Differential GNSS is to reduce the σ_{UERE} - *the standard deviation of the residual error on the pseudorange after the corrections* - so to improve the precision of the estimate. This can be done by exploiting a *reference station* (or a network of RS) which, being fixed on a known location, is able to precisely estimate the error and to broadcast the corrections to the users through a dedicated communication channel or through the internet. The idea is that the reference station, called *base*, receives the satellite signals and measures the pseudoranges using a GNSS receiver. Then it computes the pseudorange errors since the true position of the RS is known with geodetic precision, and the corrections are broadcasted to the users, called *rovers*, in the local area. The true position of the reference stations is manually set when known a priori, otherwise an initial setup time in survay mode is required for the RS to compute it. DGNSS can be applied in two different domain:

- *Position domain*: the RS sends to the user the position errors $(\Delta x, \Delta y, \Delta z)$ which are applied to the user estimated position. This approach is valid only if both user and RS use the same satellites (same GDOP) but the quality of the correction quickly degrades with the distance between them.

- *Range domain*: the RS sends to the user the pseudorange errors (for each satellite) and the corrections are applied to the user pseudoranges.

Each correction is valid for the time instant t_c in which the reference station computes the pseudorange. Due to latency in the transmission the user always applies the corrections at $t > t_c$ and then it must also receive its derivative to properly estimate the right correction when it is actually applied. If the base and rover are not too far, DGNSS works since the impairments experienced by the signals

while propagating through the atmosphere are very similar, and the resulting errors are almost the same. However, DGNSS cannot correct local errors, generated for example from receiver noise or multipath effects, since they are completely uncorrelated between base and rover. DGNSS properly works within some tens of kilometers from the reference station, with the quality of the corrections decreasing with the distance.

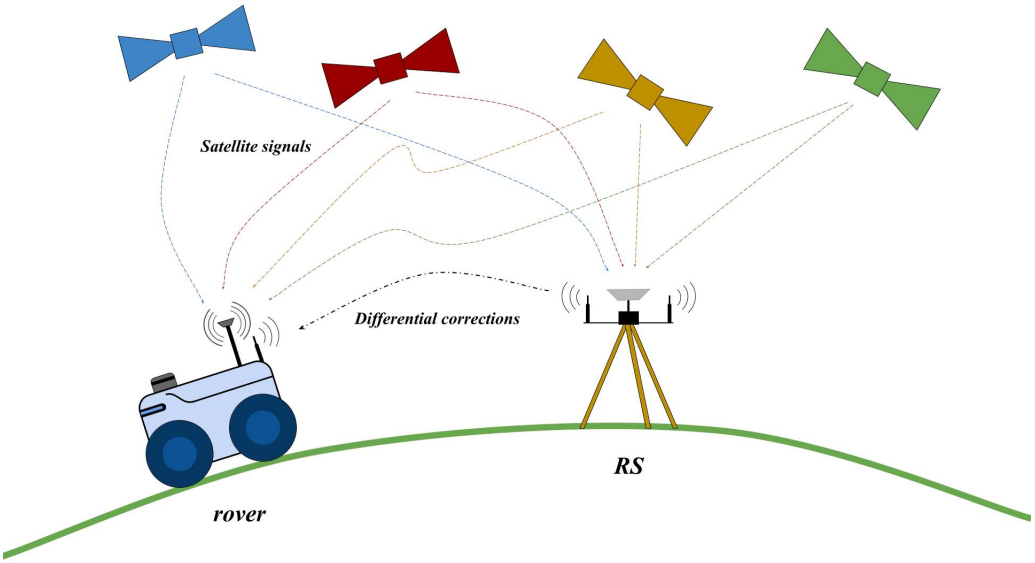

Figure 2.10. Differential GNSS

2.5.2 Real Time Kinematics

A very powerful technology exploiting the concept of differential GNSS is the *Real Time Kinematics* technique. What differentiate RTK technology from standard DGNSS is that, in the former, the pseudorange is computed by applying the *carrier-phase ranging* approach, while in the latter the *code-phase ranging* approach is used. Carrier-phase ranging exploits correlation at the carrier wave level instead of code level, and being the period of the sinusoidal wave much shorter

than the chip period, the resulting measurement has much higher precision. The SV-to-user distance is computed by multiplying the carrier wavelength times the number of whole cycles between transmitter and receiver and adding the phase difference. The most difficult part is the determination of the number of cycles, since the phase may be shifted by more than one cycle. The key feature of RTK is its ability to solve this carrier ambiguity in real time. Taking the L1 band as a reference ($f_0 = 1575.42MHz$), the corresponding wavelength is 19cm, resulting in a potential estimation accuracy relative to the reference station below the centimeter level. RTK corrections are valid up to 20km from the base station and, depending on the application and the environment, they can be transmitted on a dedicated radio link or through the internet. RTK technology finds application in surveying, precision farming, mobile robotics and related fields.

2.6 Conclusions

We have seen how GNSS is a very powerful technology able to provide positioning services at low cost, high availability and reliability on a global scale. For our specific application, GNSS augmentation systems are required to maximize the localization performance, since the accuracy of the stand-alone technology, around $1 - 2m$, is not enough to guarantee an efficient and safe delivery service with mobile robots in potentially small and crowded outdoor environments. For this reason, the Real Time Kinematics technology is exploited in our system allowing the localization to reach centimeter level precision. Despite the fact that GNSS represents the main source of absolute measurements for our system, alone it is not enough to solve the localization problem and additional sub-systems are required. In chapter 5 we will discuss how this technology is implemented in our project.

Chapter 3

Inertial Measurements

The term *inertial measurements* refers to the measurements of angular velocities and external forces applied to a body. Such measurements are performed by *inertial sensors*, i.e. *gyroscopes* for angular velocities and *accelerometers* for external forces. The term Inertial Measurement Unit, or IMU, refers to an electronic device which measures angular velocity, acceleration and sometimes orientation of a body by combining three mutually orthogonal accelerometers and three mutually orthogonal gyroscopes. In addition, IMU devices commonly include magnetometers, which do not perform inertial measurements but are used to improve the orientation estimation from accelerometers and gyroscopes. Nowadays, inertial sensors based on MEMS technology (microelectromechanical systems) are more and more widespread in IMU devices since their accuracy have increased over the years and they are inexpensive, light, small, have low power consumption and short start-time, allowing their use in devices with strict requirements like smartphones, wearable or drones. IMU sensors are frequently used in robots, cars, boat and airplanes for navigation purposes but also in fields like motion capture for the movie or gaming industries, bio-mechanical analysis, consumer electronics and so on.

As anticipated in section 1.2.2, one of the main purposes of IMU in navigation is the estimation of the vehicle's orientation and position. The orientation is estimated with a dead reckoning approach and is then subject to error accumulation, unless a magnetometer is used to obtain a reference heading. Position from IMU only is instead always subject to strong drifts since it is derived from the orientation and double integration of acceleration measurements, therefore it must always be combined with other type of sensors if used in localization. The main error sources that affects inertial sensors are biases, bias stability and thermo-mechanical noise [10]. *Biases* are the average values of the measured angular velocity and acceleration when no acceleration or rotation is actually experienced. Biases cause a systematic drift in position, velocity and orientation obtained from integrated measurements. This kind of error can be compensated by measuring the bias when the sensor is static and subtracting it during the operation. *Bias stability* defines how the bias changes in stable conditions. Temperature fluctuation and flicker noise are some of the causes of bias variation. The bias stability introduces a non-systematic error that cannot be corrected. *Thermo-mechanical noise* is a white noise which affect the position, velocity and orientation estimations introducing a random walk.

After going through the main sensors of an inertial measurement unit, in section 3.4 we present how the body's orientation can be estimated from their measurements.

3.1 Accelerometer

An accelerometer is an *inertial-frame sensor* that measures the proper acceleration it experiences. Proper acceleration is the rate of change in velocity of a body,

measured in m/s^2, in its rest frame, which is different from the acceleration experienced with respect to a fixed coordinate system. An accelerometer resting flat on a table on the Earth will measure an acceleration equal to the Earth's gravity ($g \simeq 9.81 m/s^2$) in upwards direction, corresponding to the force applied by the surface on the body to counteract the gravity. When instead the accelerometer is in free fall, an acceleration of $0 m/s^2$ will be measured in the falling direction. According to the Einstein's *equivalence principle*, the gravitational effect on an object is indistinguishable from any other experienced acceleration, therefore an accelerometer is not able to distinguish between being in deep space and accelerating at $1g$ or resting on the Earth's surface. For this principle, an application interested in linear accelerations of a body with respect to the Earth requires gravity compensation, usually obtained by calibrating the sensor while resting or from the knowledge of the gravity model. Accelerometers are also used to detect the gravity vector, which can be useful for some application, like creating a magnetic compass. Assuming gravity compensation on the Earth, from the provided acceleration one can get the velocity by integrating once in time and the position by integrating twice:

$$\boldsymbol{v} = \int \boldsymbol{a} \quad dt \qquad , \qquad \boldsymbol{x} = \int \boldsymbol{v} \quad dt$$

Due to measurements errors each integration creates drift, which causes inaccurate estimates, especially for position. The measurement coming from an accelerometer can be modelled as:

$$\hat{\boldsymbol{a}}_t = \boldsymbol{a}_t + \Delta \boldsymbol{a}_t + \boldsymbol{e}_{a,t} \tag{3.1}$$

where $\boldsymbol{a}_t$ is the true total acceleration applied to the body, $\Delta \boldsymbol{a}_t$ is a constant or slowly varying bias vector and $\boldsymbol{e}_{a,t} \sim \mathcal{N}(0, \Sigma_a)$ is the noise typically modelled as Gaussian, at time t. Accelerometers are widely used in many fields like engineering, medical applications, structural monitoring, consumer electronics etc. For certain applications accelerometers are less useful by themselves and so are often combined

with other sensors, like in IMU devices as we saw before for navigation purposes.

3.2 Gyroscope

A gyroscope is an *inertial-frame sensor* that measures the angular velocity relative to itself. The angular velocity is the rate at which the body rotates around a specific axis in its local frame and it is measured in rad/s. Gyroscopes exploit the *Coriolis effect* to perform such measurements. The Coriolis effect denotes the deflection of an object due to the *Coriolis force*, an inertial force acting on objects moving on a reference frame that rotates with respect to an inertial frame. Gyroscopes are mainly used as orientation sensors, to derive the orientation of the body from the angular velocities around the three orthogonal axes of its local frame. In aviation the rotation of an aircraft is called *roll*, *pitch* or *yaw* when occurring around the longitudinal (x), transversal (y) or vertical (z) axis respectively, but nowadays these terms are widely adopted also in the field of mobile robotics. The rotation angle around one of the axis is obtained by integrating the angular velocity from the gyroscope:

$$\theta = \int \omega \quad dt$$

but also in this case the resulting quantity will be affected by drift. Moreover, since the angular velocity may not be constant during this operation, the integration should be done as quickly as possible (reducing the integration time interval) to reduce the error due to non constant angular rates. The measurement coming from the gyroscope can be modelled as:

$$\hat{\omega}_t = \omega_t + \Delta\omega_t + e_{\omega,t} \tag{3.2}$$

where ω_t is the true angular velocity of the body, $\Delta\omega_t$ is a constant or slowly varying bias and $e_{\omega,t} \sim \mathcal{N}(0, \Sigma_\omega)$ is the noise typically modelled as Gaussian, at time t.

3.3 Magnetometer

A magnetometer is a sensor able to measure the magnetic field passing through it. Magnetometer are able to measure both magnitude and direction of the magnetic field vector, while sensors that only measure the direction of the field are usually called *compasses*. If no magnetic interference is present, magnetometers measure the Earth's magnetic field, which points toward magnetic North. Magnetometers are commonly used in navigation to obtain a reference heading and to correct the drift in the relative orientation estimated from gyroscopes, since an absolute heading is derived from the Earth's magnetic field. Magnetometers are often combined with accelerometers since the gravity vector is required to compute the orientation of the device. The reliability of the estimated heading of the device on the Earth's surface is subject to magnetic interference since a magnetometer measures the total magnetic field passing through it. This means that not only magnetic sources ruin the measurements, but also any ferromagnetic material in proximity of the sensor may distort the Earth's magnetic field, resulting in a wrong orientation estimation. For this reason these sensors are usually not suitable for indoor applications or environment where the Earth's magnetic field is highly distorted. Another source of error is represented by the high temperature, which causes increasing noise on the measurements, then magnetometers should be design to limit the temperature effect or should use compensation algorithms. The measurement coming from a magnetometer can be modelled as:

$$\hat{\boldsymbol{m}}_t = \boldsymbol{m}_t + \boldsymbol{e}_{m,t} \tag{3.3}$$

where $\boldsymbol{m}_t$ is the true magnetic field passing through the sensor and $\boldsymbol{e}_{m,t} \sim \mathcal{N}(0, \Sigma_m)$ is the noise typically modelled as Gaussian, at time t.

3.4 Orientation estimation

As we discussed above, each sensor has its own weaknesses when it comes to estimating orientation. Inertial Measurement Units combine those sensors to overcome their weaknesses and to produce a more precise and reliable estimation. Some IMU devices also come with an integrated microcontroller to directly provide the orientation estimate as output of a sensor fusion algorithm, in addition to the raw sensor data. Usually IMUs can work in different modes, employing different configurations of sensors, but in general the estimated orientation can be divided into *relative* or *absolute* orientation. Relative orientation is expressed with respect to the starting orientation of the device and it is computed from gyroscope, while absolute orientation is expressed with respect to the magnetic North, and it is computed from magnetometer and accelerometer.

Absolute orientation Let us assume we want to estimate the orientation of a robot while it stands still on the ground, with respect to the ENU frame (section 2.3.1). The orientation vector can be derived from the magnetic field vector and the gravity vector. If the robot does not move, the accelerometer measures a vector in the opposite direction of gravity, while the magnetometer measures the Earth's magnetic field passing through the robot (assuming no interference) which points toward magnetic North. Depending on the hemisphere, the lines of the magnetic field point North but also up or down. In the Los Angeles area for example, the magnetic field is angled about 60° down. Consider figure 3.1 as a reference. From the accelerometer we get the acceleration vector pointing in the same direction of the ENU *Up* axis, and the opposite of it is the *Down* direction. From the magnetometer we get the magnetic field vector, pointing toward *North* but also *Down*. From the cross product between these two vectors we get the *East* direction. Finally, we compute the *North* direction as the cross product

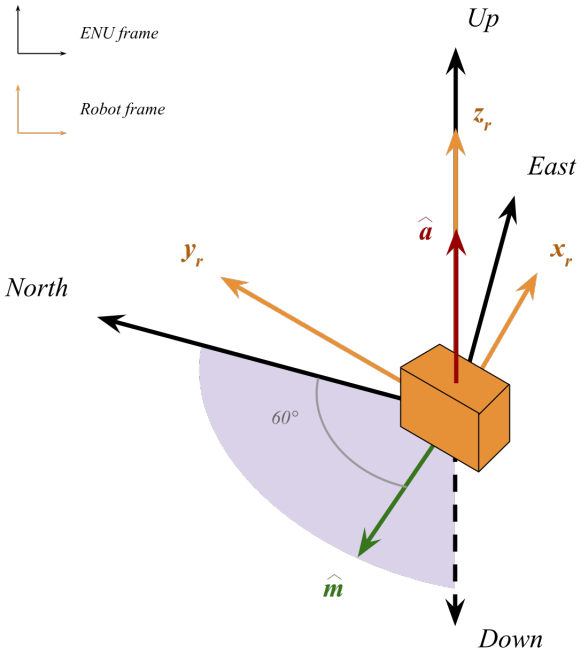

Figure 3.1. Orientation estimation from accelerometer and magnetometer

between *East* and *Down*. Denoting with **uN**, **uE**, **uD** the unitary vectors pointing respectively *North*, *East* and *Down* and with **uM**, **uG** the unitary vectors for the magnetic field and acceleration, we have:

$$\mathbf{uD} = -\mathbf{uG} \quad , \quad \mathbf{uE} = \mathbf{uD} \times \mathbf{uM} \quad , \quad \mathbf{uN} = \mathbf{uE} \times \mathbf{uD}$$

uN, **uE** and **uG** represent the ENU frame orientation in the robot's frame, from which we can derive the robot's orientation in the ENU frame. This approach is very good if the robot does not move and no magnetic interference affects it. If these conditions are not met, the following problems arise:

- The accelerometer measures all the acceleration on the robot, so if it moves we are no more able to isolate the gravity vector. In the case of human-controlled or autonomous vehicles, this problem can be solved by estimating the linear acceleration from the actuator commands. The vehicle is driven

by us or by an autonomous algorithm, either ways it is possible to estimate the accelerations that make it move and subtract them from the measured acceleration vector to isolate the gravity component.

- The magnetometer returns the magnetic field passing through the sensor, and if the device is subject to magnetic interference we are no longer able to isolate the Earth's one. However, if the Earth's magnetic field is only subject to constant biases, they can be compensated with calibration.

Relative orientation The same goal can also be reached with a completely different approach, exploiting the gyroscope alone. Measuring the angular velocities in the three orthogonal axes and knowing the time interval between measurements, we are able to compute the rotation angles and so the orientation with respect to the initial state. Figure 3.2 reports the scheme of the process. This approach is reliable only for a short interval since the drift caused by the integration grows with time.

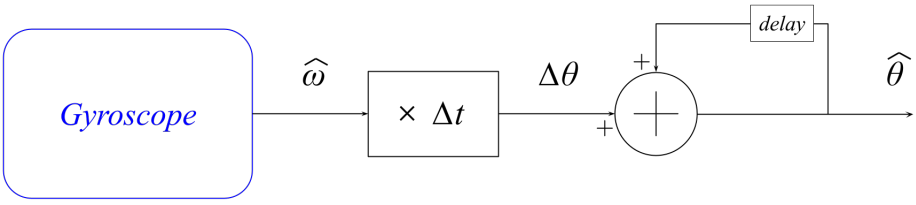

Figure 3.2. Orientation estimation from gyroscope

3.5 Conclusions

Summarizing, the orientation from accelerometer and magnetometer is not affected by error accumulation but its accuracy depends on external accelerations and magnetic interference. On the other side, the gyroscope is not affected by external

conditions but the measurements' noise causes an increasing error and the initial orientation of the device must be known. What is done in most IMUs is to combine the two systems so that one compensates the weaknesses of the other. Sensor fusion is commonly performed with a Complementary filer or a Kalman filer algorithm [11], where the orientation estimated from magnetometer and accelerometer is used to correct the drift introduced from the gyroscope. Regarding the localization problem, an IMU is a very powerful sensor for the estimation of the robot's heading and the measurement of accelerations, making it commonly used for dead reckoning in two or three dimensions.

Chapter 4

Sensor fusion

Sensor fusion, *data fusion* or *information fusion* are all synonyms used to identify any kind of combination of multiple sources with the aim of obtaining improved information. With improved information we mean that, with respect to the observation of a single sensor, the output of the fusion algorithm usually benefits from the following:

- *Uncertainty reduction*: fusion of data coming from multiple sources lower the unavoidable uncertainty characterizing each sensor.
- *Improved accuracy and reliability*: a system combining multiple sensors is able to provide accurate information even in case of partial failure.
- *Extended spacial and temporal coverage*: different sensors might cover different areas, complementing each other, both in space and time.

The multidisciplinary nature of sensor fusion and its application in several different fields make difficult the classification of the employed techniques. In the following we provide a classification based on the *relation between the data sources*, but other criteria are often used for the same purpose [12], like the input/output data type, level of abstraction, architecture type of algorithms and so on. We talk about:

1. *Complementary sensor fusion* when the data coming from different sources do not depend on each other but complement themselves. Each sensor capture a different part of the scene and the combination of them produce a more complete information. This type is the most used in mobile robot localization.

2. *Redundant sensor fusion* when the sensors independently measure the same quantity. In this case the data is used in a competitive way and fused to improve the confidence in the estimation. Redundant sensor fusion is often used in fault tolerant systems, enhancing their robustness against failures.

3. *Cooperative sensor fusion* when the data coming from multiple sensors is combined to produce a new, more complex information not obtainable with a sensor alone. An example is the combination of two images from slightly different points of view to derive a three-dimensional information.

One of the purpose of many sensor fusion algorithms is the estimation of the state of a target given the sensors observations or measurements. The estimation problem consists in finding the variables of the state vector that best fit the observed data. Identified as *state estimation methods*, they could be divided into *linear* or *nonlinear* methods. Linear state estimation algorithms are suitable when measurements and the target state equations are linear. Nonlinear algorithms are instead applied in case of nonlinear system dynamics and are generally more complex. Taking as example the robot localization problem, the state vector can be represented by the position, velocity and orientation of the robot. A part from the one-dimensional scenario, for both 2D or 3D environment the robot has a nonlinear dynamics and the problem must be solved by a nonlinear state estimation algorithm. The following section introduces the concept of state estimation and covers some of the most important related algorithms.

4.1 State estimation methods

With the term *state* we identify those variables defining the status or condition of a system at a specific time instant. For example, the state of a biological system can be represented by the body temperature, heart rate and blood pressure, while the state of a motor can be the speed and position of the shaft. *State estimation* tries to solve the problem of estimating these quantities from the sensors readings, quantities which are often not directly observable. Any area of engineering and science that involves mathematically modeled systems may benefit from state estimation techniques. A mobile robot cannot directly measure its position from the environment, but it must exploit sensor data carrying partial information about the target quantity. Moreover, all the measurements coming from sensors are affected by noise so state estimation algorithms often use a probabilistic approach to solve the problem. Instead of providing a deterministic output, probabilistic algorithms return the most likely state of the system at a given time instant as a probability distribution over a space of possible hypothesis, based on the available noisy measurements [13]. In this way it is possible to represent the ambiguity of the estimates due to the measurements uncertainty and use this information for the estimation itself.

Not being directly measurable, the estimate of the system state is inferred from sensor data and it is often called *belief* to distinguish it from the true state of the system. The *belief* is represented by a probability distribution conditioned on the available data. We define the belief for a state variable x_t as:

$$bel(x_t) = p(\, x_t \mid z_{1\to t}, u_{1\to t}\,) \tag{4.1}$$

which is a posterior probability distribution of x_t at time t conditioned on all the past measurements $z_{1\to t}$ and all the past controls $u_{1\to t}$. It is often useful to define

the posterior probability before including the last measurement z_t:

$$\overline{bel}(x_t) = p(\, x_t \mid z_{1\to t-1}, u_{1\to t}\,) \tag{4.2}$$

$\overline{bel}(x_t)$ is also called *prediction* since it predicts the state at time t based on the previous belief and current control, while $bel(x_t)$, being computed form the prediction and current measurement, is called *update*. In the next paragraph we introduce the most general algorithm for the calculation of beliefs, the *Bayes filter*, which is at the base of many state estimation algorithms used in practice, like the Kalman filter.

4.1.1 The Bayes filter

The Bayes filter is a recursive algorithm used to estimate the belief $bel(x_t)$ at time t from the belief $bel(x_{t-1})$ at time $t-1$. Taking as input the control and measurement at time t and the belief $bel(x_{t-1})$, it outputs the current belief $bel(x_t)$. The algorithm consists of two steps: *prediction* step and *measurement update* step.

Prediction In the prediction step the Bayes filter predicts the belief $\overline{bel}(x_t)$ at time t by integrating the product between the prior probability distribution assigned to x_{t-1} and the probability that the control input u_t produces x_t from x_{t-1}.

$$\overline{bel}(x_t) = \int p(\, x_t \mid u_t, x_{t-1}\,) \cdot bel(x_{t-1})\, dx \tag{4.3}$$

In other words, the belief $\overline{bel}(x_t)$ represents the estimated probability that the system is in the state x_t at time t immediately after the input u_t has been applied. This belief is then computed by combining the probability that the system moves from the previous state x_{t-1} to the current state x_t with the probability that the system actually was in the state x_{t-1} , i.e. $bel(x_{t-1})$. The probability $p(x_t|u_t, x_{t-1})$ is given by the mathematical model of the system that must then be known with

accuracy. To obtain the probability distribution of the state this operation must be done for all the possible states, which, in mathematical terms, results in an integration over the domain of x.

Measurement update In the update step the Bayes filter obtains the belief over the state x_t by multiplying the prediction $\overline{bel}(x_t)$ by the probability of the measurement z_t conditioned to the hypothetical state x_t.

$$bel(x_t) = \nu \cdot p(\, z_t \mid x_t \,) \cdot \overline{bel}(x_t) \tag{4.4}$$

The idea is to correct the prediction exploiting the measurement information. The probability $p(\, z_t \,|\, x_t \,)$ is given by the measurement model and tells us how likely it is to observe the measurement z_t when the system is in state x_t. If the sensor reading is close to the one expected from the measurement model, such probability will increase and the update step will confirm the validity of the prediction. If instead the sensor reading is distant from the expected measurement, the correction will decrease the probability to be in the predicted state. According to the *Bayes rule*, equation (4.1) can be written as:

$$bel(x_t) = p(\, x_t \mid z_t, u_t \,) = \frac{p(\, z_t \mid x_t, u_t \,) p(\, x_t \mid u_t \,)}{p(\, z_t \mid u_t \,)} = \frac{p(\, z_t \mid x_t \,) p(\, x_t \mid u_t \,)}{p(\, z_t \,)} \tag{4.5}$$

since z_t does not depend on u_t. Equation (4.4) is obtained from equation (4.5) by recalling that $\overline{bel}(x_t) = p(\, x_t \mid u_t \,)$ and denoting $\frac{1}{p(\, z_t \,)}$ as ν.

These two steps are executed recursively at each time instant starting from an initial condition, so an initial belief $bel(x_0)$ is required. If the state at time $t = 0$ (i.e. x_0) is precisely known, the belief $bel(x_0)$ should assign 1 to the probability of the known value of x_0 and zero probability anywhere else. If the initial value of x_0 is completely unknown, the initial belief should be a uniform distribution over the domain of the state at time $t = 0$. If instead x_0 is partially known, a proper

non-uniform probability distribution should be assigned to the initial belief. The following pseudo-code reports the general Bayes filter algorithm.

Bayes_filter($bel(x_{t-1})$, u_t , z_t)
  for all x_t **do**
    $\overline{bel}(x_t) = \int p(\, x_t \mid u_t, x_{t-1}\,) \cdot bel(x_{t-1})\, dx$
    $bel(x_t) = \nu \cdot p(\, z_t \mid x_t\,) \cdot \overline{bel}(x_t)$
  end for
  return $bel(x_t)$

The Bayes filter stated here can only be applied on simple estimation problems. For non-finite state spaces both the prediction integration and the update multiplication require a closed form to be computed. The following section introduces a widely used state estimation algorithm, the Kalman filter. Being a very important part of this work, we cover both linear and nonlinear Kalman filter algorithms.

4.1.2 The Kalman filter

The Kalman filter is one of the most important estimation algorithms, commonly used for tracking, localization, navigation, control systems and more. Based on uncertain measurements, the Kalman filter provides estimates of mean and covariance of hidden variables (the state of the system) and future prediction of them based on past estimates and on the knowledge of the system dynamics. Like all the estimators belonging to the family of Gaussian filters, the Kalman filter is based on the idea that beliefs are represented by multivariate normal distributions, characterized by the following probability density function:

$$p(x) = det(2\pi\Sigma)^{-\frac{1}{2}} \cdot e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)} \tag{4.6}$$

This *pdf* over the state x is characterized by the mean μ and the covariance Σ. The mean vector μ has the same dimension of the state x while the covariance Σ is a quadratic, symmetric and positive-semidefinite matrix with dimension equal

to the square of the dimension of x. Gaussian probability density functions are unimodal since they are characterized by a single maximum, hence Gaussian estimators are suitable for all those problems in which there exist a single hypothesis around the true state with a little margin of uncertainty, like many tracking problems in robotics.

The Kalman filter works by propagating the mean and covariance of the state through time, and if both the system dynamics and measurement model are linear functions, the produced output will always be Gaussian distributed. The Kalman filter does not work for nonlinear systems, for which extended versions of the algorithm are required (see later). Like the Bayes filter, the Kalman filter is a recursive algorithm composed by two main steps: the prediction step, in which the mean and covariance of the next state are predicted based on the past state and system model, and the update step, in which the predicted mean and covariance are corrected with the measurements.

Prediction The prediction step is composed by two equations used to *extrapolate* the mean and covariance of the state vector from the previous state and the system dynamics. The system dynamics is the law that links the future variables with the current ones, and it varies for different systems. However, we can write it in the following general form:

$$\hat{\boldsymbol{x}}_{n+1,n} = \mathbf{F}\hat{\boldsymbol{x}}_{n,n} + \mathbf{G}\boldsymbol{u}_n + \boldsymbol{w}_n \tag{4.7}$$

where $\hat{\boldsymbol{x}}_{n+1,n}$ is the predicted state vector at time $n+1$, $\hat{\boldsymbol{x}}_{n,n}$ is the estimated state vector at time n, $\boldsymbol{u}_n$ is the control input vector at time n, $\boldsymbol{w}_n$ is the process noise at time n (not measurable), $\mathbf{F}$ is the state transition matrix and $\mathbf{G}$ is the input transition matrix. The process noise $\boldsymbol{w}_n$ is a white Gaussian noise with covariance matrix $\mathbf{Q} = E\{\boldsymbol{w}_n \boldsymbol{w}_n^T\}$. This equation is called ***state extrapolation equation***

and performs the prediction for the mean. The prediction of the covariance matrix is instead performed by the so called ***covariance extrapolation equation***:

$$\mathbf{P}_{n+1,n} = \mathbf{F}\mathbf{P}_{n,n}\mathbf{F}^T + \mathbf{Q} \tag{4.8}$$

where $\mathbf{P}_{n+1,n}$ is the predicted covariance matrix at time $n+1$ and $\mathbf{P}_{n,n}$ is the estimated covariance matrix at time n.

After the prediction step, the obtained mean and covariance will be updated with the measurements. It is then required a ***measurement model***:

$$\boldsymbol{z}_n = \mathbf{H}\boldsymbol{x}_n + \boldsymbol{v}_n \tag{4.9}$$

where $\boldsymbol{z}_n$ is the measurement vector at time n, $\boldsymbol{x}_n$ is the true system state at time n (unknown), $\boldsymbol{v}_n$ is the random noise vector at time n and $\mathbf{H}$ is the observation matrix. The measurement noise $\boldsymbol{v}_n$ is a white Gaussian noise with covariance matrix $\mathbf{R}_n = E\{\boldsymbol{v}_n\boldsymbol{v}_n^T\}$. Many times the measured values are not the desired system variables, then a linear transformation $\mathbf{H}$ is needed to transform the system state into the measurements. If instead it is possible to directly measure the system variables, $\mathbf{H}$ will be the identity matrix.

Update Once the measurements are available, the predicted state mean is updated with the ***state update equation***:

$$\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1} + \mathbf{K}_n(\boldsymbol{z}_n - \mathbf{H}\hat{\boldsymbol{x}}_{n,n-1}) \tag{4.10}$$

where $\hat{\boldsymbol{x}}_{n,n-1}$ is the predicted state vector at time n and $\mathbf{K}_n$ is the ***Kalman gain*** at time n and indicates how much weight is assigned to the measurements against the predictions. In the same way also the covariance matrix is updated through the ***covariance update equation***:

$$\mathbf{P}_{n,n} = (\mathbf{I} - \mathbf{K}_n\mathbf{H})\mathbf{P}_{n,n-1}(\mathbf{I} - \mathbf{K}_n\mathbf{H})^T + \mathbf{K}_n\mathbf{R}_n\mathbf{K}_n^T \tag{4.11}$$

where $\mathbf{P}_{n,n-1}$ is the predicted covariance matrix at time n. As anticipated, the Kalman gain indicates the weights assigned to the measurements in the update step, and it is derived at each filter cycle based on the estimate covariance and measurement covariance:

$$\mathbf{K}_n = \mathbf{P}_{n,n-1}\mathbf{H}^T(\mathbf{H}\mathbf{P}_{n,n-1}\mathbf{H}^T + \mathbf{R}_n)^{-1} \tag{4.12}$$

The Kalman filter needs to be properly initialized, i.e. we need to set the initial state of the system and its covariance matrix $\mathbf{P}_{0,0}$ which must be estimated based on our confidence in the initial state. If we perfectly know the initial state, then $\mathbf{P}_{0,0} = \mathbf{0}$, if we have no idea about the initial state, then $\mathbf{P}_{0,0} = \infty \cdot \mathbf{I}$.

Let us clarify the concept with an example. Consider figure 4.1 as a reference: we want to estimate the position of the robot on a two-dimensional plane. Assume

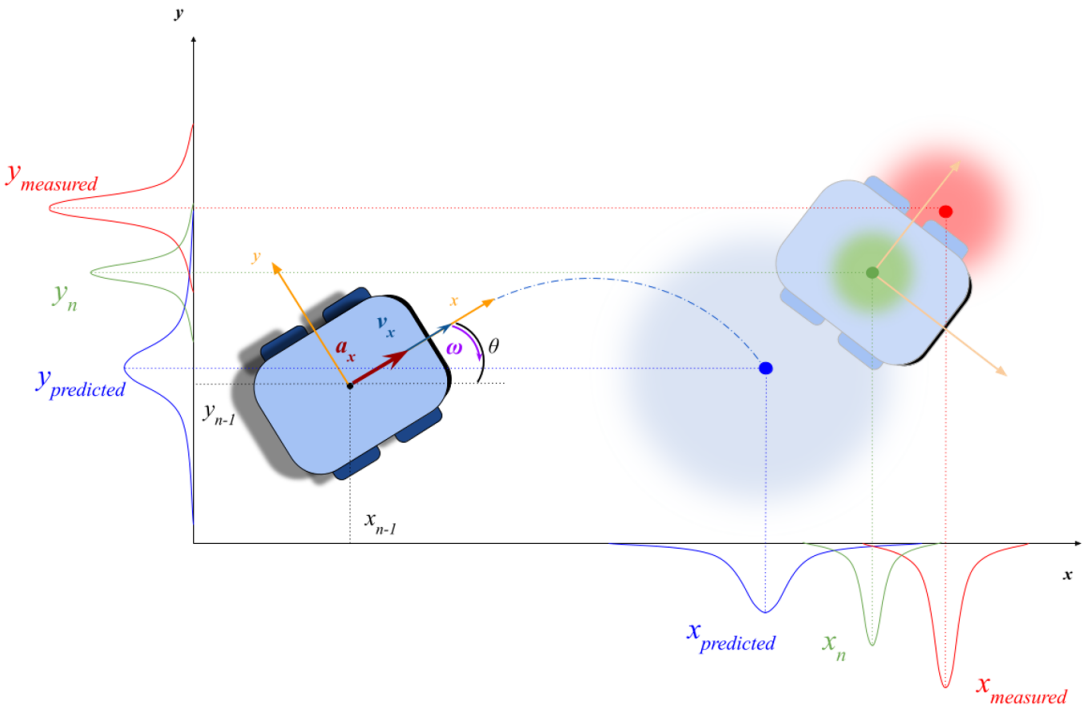

Figure 4.1. Example of a 2D pose estimation

the robot is equipped with sensors able to measure velocity, acceleration and heading as well as its absolute position (from GPS for example), all noisy measurements with a certain level of uncertainty. At start the robot is placed in a known position with respect to the map that we are able to measure, then the Kalman filter is initialized with our initial belief of it. The robot starts moving and the sensors report the respective quantities. As shown in the figure, at time n the robot belief is characterized by position, velocity, acceleration and heading estimated at the previous step. A prediction is then computed for time $n + 1$ based on this values and on the kinematic model with the *state extrapolation equation* and *covariance extrapolation equation*, represented with the blue dot and blue Gaussian distributions. Once the absolute position measurements are available (red dot and red Gaussian distributions), this prediction is updated with the *state update equation* and *covariance update equation* weighting the two contributions based on their level of uncertainty. The weights are computed at each time step based on the covariance of measurements and previous estimate and correspond to the Kalman gain. This sequence prediction $\rightarrow$ update is then repeated cyclically at each time instant. The Kalman filter algorithm for linear systems is summarized in figure 4.2. At each cycle the output of the filter will be the mean-covariance pair of the state. The mean is the result of a weighted average between the prediction and the measurements while the resulting covariance is significantly reduced with respect to the starting ones.

The algorithm just described is extremely important for many state estimation algorithm implemented in real systems. However, as already mentioned above, it only works for linear systems and therefore it does not achieve good performance in real applications, which mostly involve nonlinear systems. Being at the core of the fusion process of this project, in the following paragraph the two most popular nonlinear Kalman filter algorithms are explained: the *extended* KF and

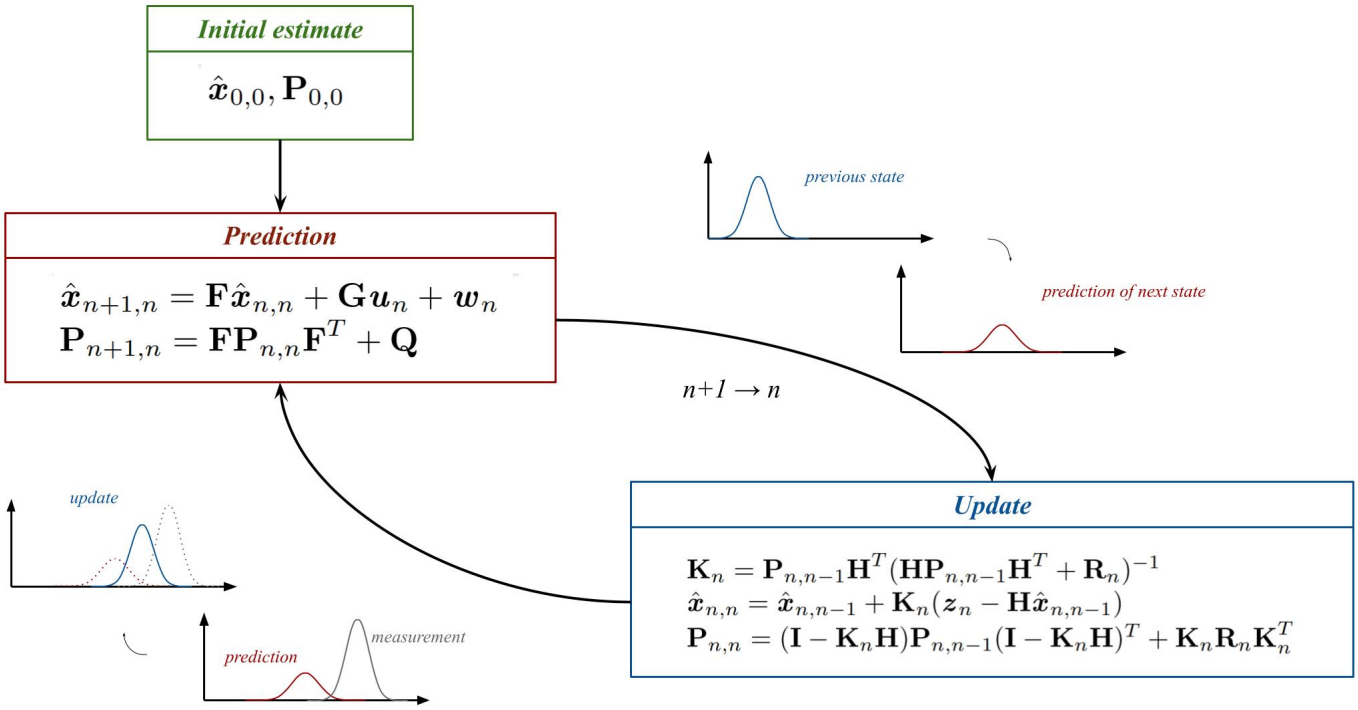

Figure 4.2. Kalman filter algorithm

the *unscented* KF.

Extended Kalman filter

As the name suggests, the extended Kalman filter is an extension of the previous algorithm used to estimate the state of a nonlinear system. The idea is to linearize the system around a nominal trajectory. The nominal trajectory is a guess of the future states, like for example the path that an autonomous vehicle is supposed to follow in the future, but many times it is not straightforward to determine it and so the estimated state itself is used as nominal trajectory. After the linearization the filter estimates the next state of the linearized system, which is used as the nominal trajectory around which the system will be linearized for the next step. This is repeated at each filter cycle. Consider the following general nonlinear system and

measurement model:

$$\boldsymbol{x}_{n+1} = \boldsymbol{f}_n(\boldsymbol{x}_n, \boldsymbol{u}_n, \boldsymbol{w}_n) \tag{4.13}$$

$$\boldsymbol{z}_n = \boldsymbol{h}_n(\boldsymbol{x}_n, \boldsymbol{v}_n) \tag{4.14}$$

where $\boldsymbol{w}_n \sim N(0, \mathbf{Q})$ and $\boldsymbol{v}_n \sim N(0, \mathbf{R})$. By expanding the nonlinear functions $\boldsymbol{f}(\cdot)$ and $\boldsymbol{h}(\cdot)$ in Taylor series around the nominal state $\boldsymbol{x}_n^* = \hat{\boldsymbol{x}}_{n,n}$, the nominal input $\boldsymbol{u}_n^* = \boldsymbol{u}_n$ (since we assume to exactly know the control input) and noise $\boldsymbol{w}_n^* = 0$, $\boldsymbol{v}_n^* = 0$, we get:

$$\begin{aligned}
\boldsymbol{x}_{n+1} &= \boldsymbol{f}_n(\hat{\boldsymbol{x}}_{n,n}, \boldsymbol{u}_n, \mathbf{0}) + \left.\frac{\partial \boldsymbol{f}_n}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}_{n,n}} (\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n}) + \left.\frac{\partial \boldsymbol{f}_n}{\partial \boldsymbol{w}}\right|_{\hat{\boldsymbol{x}}_{n,n}} \boldsymbol{w}_n = \\
&= \boldsymbol{f}_n(\hat{\boldsymbol{x}}_{n,n}, \boldsymbol{u}_n, \mathbf{0}) + \mathbf{F}_n(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n}) + \mathbf{L}_n \boldsymbol{w}_n = \\
&= \mathbf{F}_n \boldsymbol{x}_n + [\boldsymbol{f}_n(\hat{\boldsymbol{x}}_{n,n}, \boldsymbol{u}_n, \mathbf{0}) - \mathbf{F}_n \hat{\boldsymbol{x}}_{n,n}] + \mathbf{L}_n \boldsymbol{w}_n
\end{aligned} \tag{4.15}$$

$$\begin{aligned}
\boldsymbol{z}_n &= \boldsymbol{h}_n(\hat{\boldsymbol{x}}_{n,n}, \mathbf{0}) + \left.\frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}_{n,n}} (\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n}) + \left.\frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{v}}\right|_{\hat{\boldsymbol{x}}_{n,n}} \boldsymbol{v}_n = \\
&= \boldsymbol{h}_n(\hat{\boldsymbol{x}}_{n,n}, \mathbf{0}) + \mathbf{H}_n(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n}) + \mathbf{M}_n \boldsymbol{v}_n \\
&= \mathbf{H}_n \boldsymbol{x}_n + [\boldsymbol{h}_n(\hat{\boldsymbol{x}}_{n,n}, \mathbf{0}) - \mathbf{H}_n \hat{\boldsymbol{x}}_{n,n}] + \mathbf{M}_n \boldsymbol{v}_n
\end{aligned} \tag{4.16}$$

We have obtained a linear state space representation and a linear model, so also in this case we can estimate the state of the system using the standard Kalman filter equations. The EKF algorithm is reported in figure 4.3. The Extended Kalman filter is a great solution to apply the KF estimator on nonlinear systems, and it has been one of the most used nonlinear estimators in the past years. However, this algorithm may lead to unreliable estimates if the system is strongly nonlinear. The EKF relies on the propagation of both the mean and covariance of the system state according to the first-order linearization of the nonlinear model, which may introduce large errors on the predicted mean and covariance depending on the nonlinear system, causing sometimes poor performance or even divergence

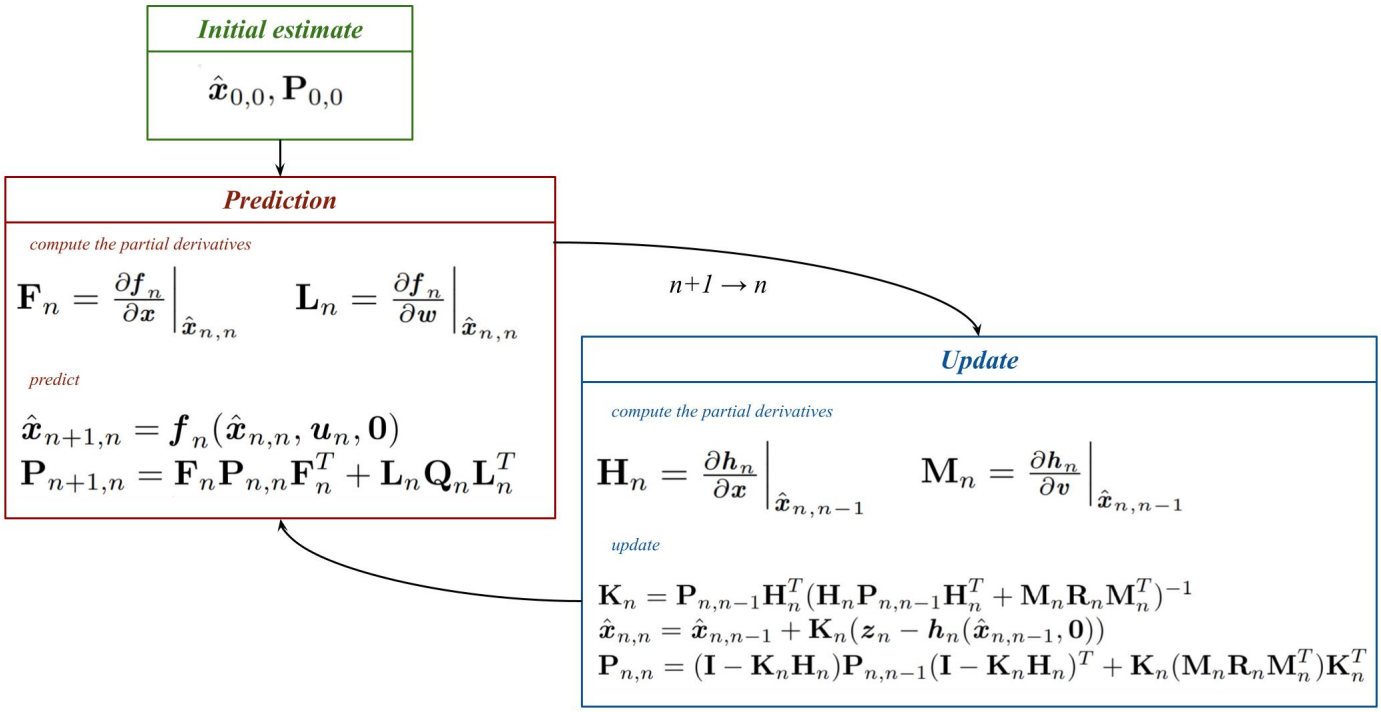

Figure 4.3. Extended Kalman filter algorithm

of the filter. In such cases it is required a more accurate estimation of the mean and covariance of the state when propagating through a nonlinear model. The Unscented Kalman filter is a nonlinear extension of the Kalman filter which reduces the linearization error of the EKF.

Unscented Kalman filter

The idea of the Unscented Kalman filter is to select a minimal set of possible system states whose mean and covariance truly represent the mean and covariance of the current state. When propagated through the nonlinear system, the mean and covariance of these carefully selected states accurately represent the true posterior mean and covariance for any nonlinear model. At the core of the UKF there is the *Unscented Transformation*, a method for the computation of mean and covariance

of a random variable (like the variables representing the state) after a nonlinear transformation. Consider a vector $\boldsymbol{x}$ with known mean $\overline{\boldsymbol{x}}$ and covariance matrix $\mathbf{P}$ and take a set of deterministic vectors called *sigma points* whose ensemble mean and covariance are exactly $\overline{\boldsymbol{x}}$ and $\mathbf{P}$. After applying the nonlinear function $\mathbf{y} = \boldsymbol{f}(\boldsymbol{x})$ to each deterministic vector, the ensemble mean and covariance of the obtained vectors will be a good estimate of the true mean and covariance of $\mathbf{y}$. Let us see the process a little more in detail: suppose we have a $m \times 1$ state vector that has to be transformed by a nonlinear function $\mathbf{y} = \boldsymbol{f}(\boldsymbol{x})$. Take $2m$ sigma points as follows:

$$\begin{aligned} \boldsymbol{x}^{(i)} &= \overline{\boldsymbol{x}} + \tilde{\boldsymbol{x}}^{(i)} \qquad i = 1, ..., 2m \\ \tilde{\boldsymbol{x}}^{(i)} &= (\sqrt{m\mathbf{P}})_i^T \qquad i = 1, ..., m \\ \tilde{\boldsymbol{x}}^{(i)} &= -(\sqrt{m\mathbf{P}})_i^T \qquad i = m+1, ..., 2m \end{aligned}$$

where $(\sqrt{m\mathbf{P}})_i$ is the i^{th} row of $\sqrt{m\mathbf{P}}$. Transform the sigma points by applying the nonlinear function:

$$\mathbf{y}^{(i)} = \boldsymbol{f}(\boldsymbol{x}^{(i)}) \qquad i = 1, ..., 2m$$

The approximated mean and covariance of $\mathbf{y}$ are given by:

$$\overline{\mathbf{y}}_{ut} = \frac{1}{2m} \sum_{i=1}^{2m} \mathbf{y}^{(i)} \tag{4.17}$$

$$\mathbf{P}_{ut} = \frac{1}{2m} \sum_{i=1}^{2m} (\mathbf{y}^{(i)} - \overline{\mathbf{y}}_{ut})(\mathbf{y}^{(i)} - \overline{\mathbf{y}}_{ut})^T \tag{4.18}$$

In [14] it is shown that $\overline{\mathbf{y}}_{ut}$ and $\mathbf{P}_{ut}$ respectively match the true value $\overline{\mathbf{y}}$ and $\mathbf{P}$ to the third order of Taylor expansion *for any nonlinearity*, while the EKF linearization only up to the first order. The UKF uses the Unscented transformation to propagate the mean and covariance of the state in both the prediction step and measurement step. The UKF algorithm is summarized in figure 4.4, with the assumption of both process and measurement noises to be additive.

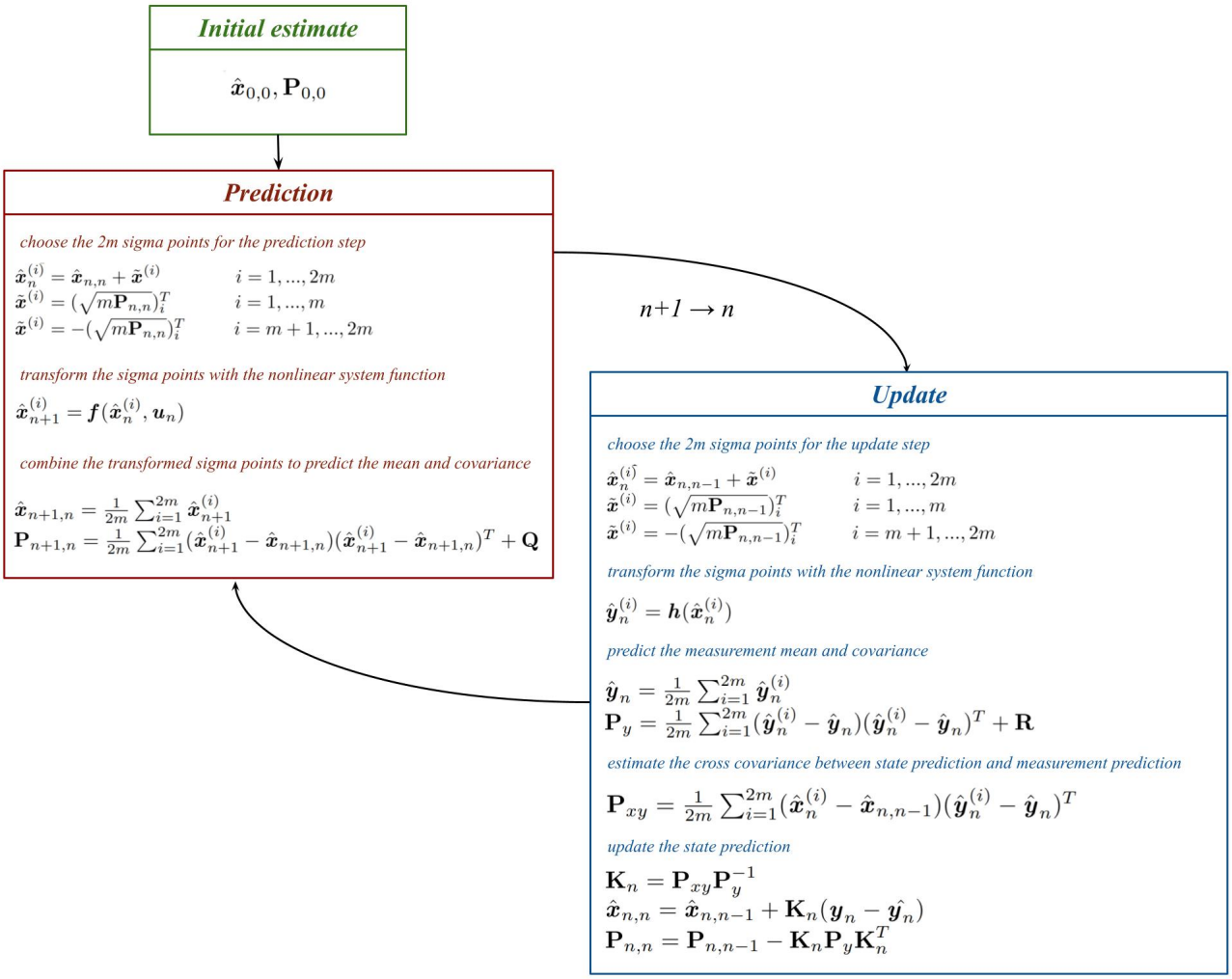

Figure 4.4. Unscented Kalman filter algorithm

4.2 Conclusions

Both the Extended Kalman filter and the Unscented Kalman filter are powerful algorithms able to estimate the state of a system from the knowledge of its dynamics and from noisy measurements, with the assumption that the state variables can be accurately represented by a multivariate Gaussian distribution. For strongly nonlinear systems the UKF can obtain greater performance with respect to the

EKF, being able to approximate any nonlinear model up to the third order of the Taylor expansion, against the first order of the extended version. The price for this improvement is that the unscented version generally requires more computational power. However, the EKF requires the computation of Jacobians (partial derivative matrices), which are numerically difficult to compute in case of systems not given in analytical form. For the specific application of localizing a robot on a 2D plane, we implemented and tested the system when using both algorithms and compared the respective performance. The obtained results are reported and discussed in the next chapter.

Chapter 5

Project development

As introduced in section 1.4, this work is part of a bigger project on autonomous mobile robots in collaboration with InnoTech Systems L.L.C and California State University of Los Angeles. As the name suggests, autonomous mobile robots are such because they are able to move in the environment and to complete assigned tasks in autonomy without any human interaction. Therefore, a crucial characteristic that any AMR must have is the ability to localize itself in the map in which it operates. As we saw at the beginning of the thesis, the robot localization problem and relative solution depend on several aspects, like the tasks the robot is suppose to execute, the environment in which it operates, the available budget etc., hence the localization system must be designed accordingly. As anticipated in the previous chapter, the robot localization is a state estimation problem where the state of the system is represented by the position and orientation of the robot with respect to a reference point (in general the origin of the map). This estimation is necessary since both position and orientation cannot be measured directly, but must be inferred from sensors data. Moreover, there exists no single sensor from which position and orientation can be accurately estimated, mainly due to

the uncertainty affecting the measurements and to the type of sensor whose performance depends on the surrounding environment. Therefore, multiple sensors are always required for the robot localization and the quality of the solution not only depends on the sensors themselves, but also on the way they are combined, on how the data is *fused* to provide position and orientation information. Given its complexity, the localization system is often composed by subsystems which are then combined by one or multiple fusion nodes to provide the required output. For example, if the robot is supposed to move both indoors and outdoors, the localization system is usually design to work according to this distinction, as some sensors and techniques are better suited for specific environments than others. Then, the two subsystems are combined to provide a seamless service.

The focus of this project is on the design, development and testing of the localization system in outdoor environment. The next section covers the development of the system and the reasons behind its design. Then, we quickly describe the robot we have built to test the localization and the used hardware. Finally, the results of testing and validation of the system are reported, with comments and conclusions.

5.1 Outdoor Localization System

As largely discussed in chapter 2, GNSS technology is currently the most used positioning system in outdoor environments for many different fields. Being a passive system from the user point of view, a small, cheap and energy efficient receiver is enough to obtain good quality position estimates, with few meters of error under good conditions. The state-of-the-art technology is nowadays able to reach a level of accuracy below the centimeter, opening its usage to an even wider set of applications, like mobile robotics, precision agriculture or land surveying,

where high precision in positioning estimation is mandatory. Moreover, GNSS provides a continuous service at global level, allowing any user on almost every place on the Earth to localize itself with very little effort. Nevertheless, this promising technology is not enough to solve the localization problem by itself for an autonomous mobile robot. To understand why, let us recall that a successful localization means finding the robot position and orientation with respect to a reference frame, continuously in time, with an estimation error always below a given threshold, which varies depending on the final application. For example, it may go from 1 meter in case of food delivery robots to 1 centimeter in case of seed sowing robots. At the time of writing, these requirements cannot be guaranteed by the global navigation satellite system for the following three main reasons:

1. A single receiver is not able to compute the orientation of the vehicle on which it is mounted.

2. The accuracy of the position estimates is not continuously guaranteed since it strongly depends on the dynamism of the environment, like weather conditions, buildings or general obstacles around the receiver antenna.

3. The continuity of the service itself cannot be guaranteed. If the sky visibility is strongly obstructed, like when crossing a tunnel or moving indoors, the GNSS signal may get completely lost.

As anticipated in section 1.4, in our system the GNSS receiver represents the main source of information for localization purposes, but additional sensors are employed to compensate for the above weaknesses. These sensors have been selected based on the needs and hardware characteristics of the robot. For now we only need to know that the localization system has been designed for differential drive wheeled robots. Additional information on the hardware are provided in the next section. Other than the GNSS receiver, the robot is equipped with an IMU device for inertial

measurements, a wheel encoder for each wheel and a stereo camera for recognition and tracking of augmented-reality tags. The system has been designed to also work in partial configurations, even if some of these sensors are not available. Two subsystems can be identified: the *position estimation* and *orientation estimation* process.

5.1.1 Position

Estimating the position on a two-dimensional plane means finding the x and y coordinates representing the displacement of the robot with respect to the origin of the map. In the following paragraphs we explain how each sensor contributes to the estimation of the robot's pose.

GNSS When using a global positioning system the receiver's position is estimated in terms of latitude, longitude and altitude, which identify a point on the Earth's surface. When localizing the robot on a 2D map, a local reference system is more appropriate than a global frame, then the geodetic coordinates need to be converted into local ones. As seen in section 2.3, this conversion consists in a two steps mapping: from the ellipsoidal coordinate system to the ENU one, passing through the ECEF frame. If performed with enough numerical precision, this geometric conversion does not cause any loss of accuracy, but the local approximation of the Earth's surface to a two-dimensional plane introduces an error. However, if the robot does not move too far from the origin, the introduced error is negligible for our purposes. Quantitatively, for a distance of about $1000m$ from the origin, the error due to the approximation is in the order of μm. The converted x and y represent absolute measurements of the position in the local frame.

Wheel encoders Wheel encoders - or *rotary encoders* - are electro-mechanical devices exploiting optical, magnetic or mechanical sensors to detect changes in

the rotational position of the motor shaft. By knowing the wheel's diameter and measuring the rotational rate of each motor, the linear and angular velocity of the robot can be derived through mathematical equations. These devices are widely used in mobile robotics to derive odometry information as explained in section 1.2.2. As we will see later, a much more accurate and reliable estimation of angular velocity is obtained from the IMU, then in our system wheel encoders are only used to estimate the linear velocity of the robot.

IMU As seen in chapter 3, IMUs provide absolute or relative orientation of the device in the environment, without the need of an external reference, and measure linear acceleration in the three orthogonal axes. Both information are used for odometry: orientation is fundamental in a 2D or 3D environment to understand in which direction the robot moves, while acceleration data improve the precision of the estimated covered distance as show in equation (1.1) and (1.2).

5.1.2 Orientation

The orientation of a coordinate frame must be defined with respect to another coordinate frame and expresses the rotation between them. There exist different ways to parametrize orientation: Euler angles, rotation matrices, quaternions ... all of them carrying the same information. Euler angles are usually not used to express the orientation of a free object in space, like the robot, due to the possible *gimbal lock* problem. Quaternions are commonly used in navigation problems since they are stable, efficient and with a compact representation. In the following paragraphs we explain how each sensor contributes to the estimation of the robot's orientation.

IMU The Inertial Measurement Unit mounted on the robot represents the main source of orientation data. This device can be configured to provide either an

absolute or a relative information based on the type of sensors used for the estimation. Relative orientation exploits the accelerometer and gyroscope, while absolute orientation requires the magnetometer in addition to the other inertial sensors. In optimal condition of no magnetic interference, the use of magnetometer allows to correct for the drift introduced by the inertial sensors and the IMU is able to provide a reliable orientation by itself. However, as we will see later, the robot used for the system validation has been built in such a way that magnetic interference cannot be excluded, due to the material of the chassis and to the electronic components mounted close to the sensors. For this reason, we decided to configure the IMU to provide a relative orientation, and to exploit external absolute measurements from GNSS and stereo camera to correct the eventual drift.

GNSS Other than position information, the GNSS measurements can also be used to estimate the robot's orientation. If specific conditions are met, the derived heading can be used as an absolute reference information to correct the drift in the IMU estimation. This technique is characterized by a delay in the estimation since the heading is computed from past GNSS measurements. Nevertheless, when specific constraints on GNSS accuracy, trajectory and speed of the robot are met, the obtained orientation is accurate enough to be used as an absolute reference information.

The localization system has been designed to get information also from artificial landmarks. The robot requires a 2D or 3D camera to be able to recognize them in the environment. Our robot is equipped with a stereo camera, a type of camera characterized by two lenses with a different image sensor for each lens. Thanks to this a stereo camera is able to simulate the binocular vision and to capture three-dimensional images, which can be used for many different purposes. In our robot this sensor is used as the main source of information for the docking and

obstacle avoidance algorithm, but it is a great source of data also for the localization problem. The relative position and orientation of the robot with respect to the augmented-reality tag can be derived with high precision thanks to computer vision techniques applied to the three-dimensional image. The absolute position and orientation can then be computed by knowing the location of the tag with respect to the map, which is then stored in the robot for each AR tag. Thanks to the high accuracy of the location information estimated with this technique, the obtained position and orientation are used to reset the state of the robot.

5.1.3 Sensor fusion

In this section we see how all the information derived from the sensors reported before are combined to solve the localization problem. Figure 5.1 depicts the scheme of the fusion algorithm. The pose measured from GNSS, orientation and acceleration from IMU and velocity from encoders represent the measurements of the state variables and are given as input to the Kalman filter. Exploiting the robot motion model in a 2D plane, the algorithm predicts the next state starting form the previous one and update the prediction when the measurements are available. Each time an AR tag is captured from the stereo camera, a reference pose and a reference orientation are computed by an external algorithm. A reference orientation is also computed from past GNSS measurements each time the required conditions are met, as previously described. Given the high precision of the location estimation from AR tag, and the stringent constraints we set to derive the robot heading from GNSS, we consider these reference informations accurate enough to reset the state of the system in the Kalman filter accordingly. Summarizing, the fusion algorithm combines absolute GNSS measurements with orientation, acceleration and velocity from inertial sensors and encoders to continuously estimate the state of the robot. Moreover, external algorithm are used to obtained accurate reference pose and

heading information to correct eventual drift in the estimation.

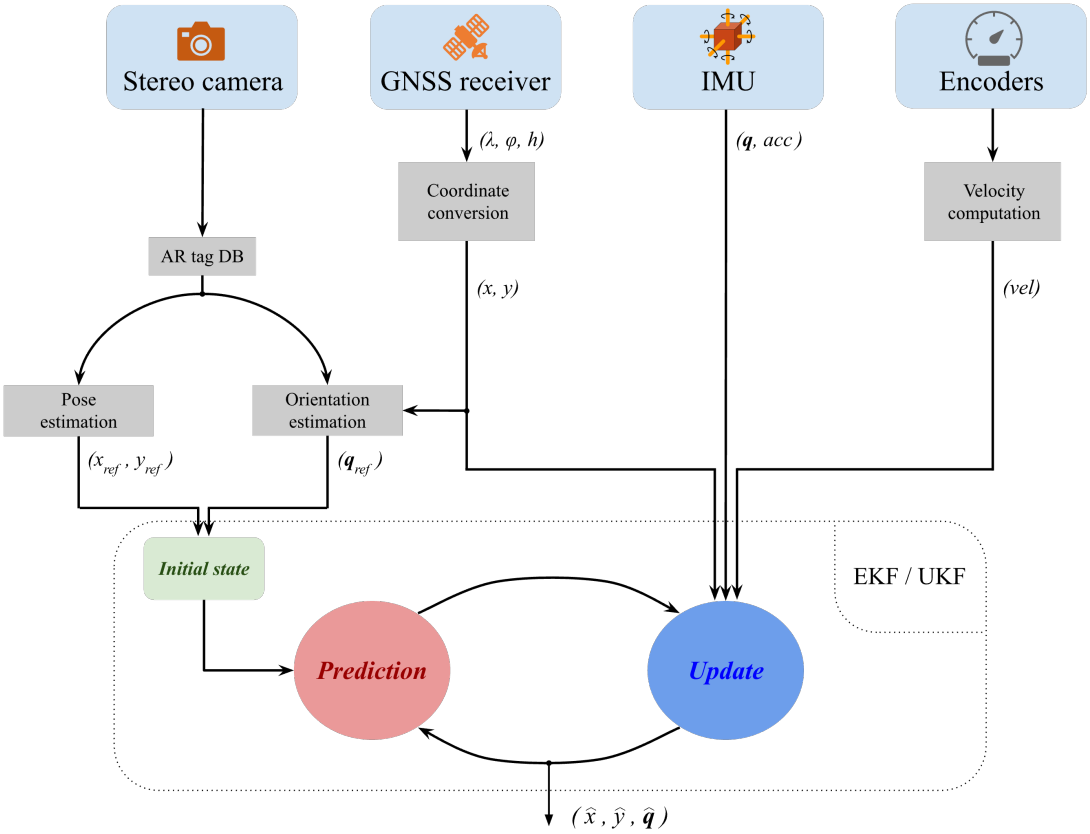

Figure 5.1. Data fusion scheme

5.1.4 Development framework

The outdoor localization system has been developed to work on the *Robot Operating System* [15]. ROS is an open-source meta-operating system running on Unix-based platforms. It provides hardware abstraction, implementation of commonly-used functionalities, low-level device control, message passing between processes, package management, tools and libraries to write, build and run code across multiple machines [16]. ROS implements a peer-to-peer network of processes, called *nodes*, coupled using the ROS communication infrastructure. The executables

written to work in ROS can be grouped into *packages* and *stacks* which can be easily shared and distributed. Different ways of communication between nodes are possible: synchronous communication over *services*, asynchronous streaming of data over *topics* and storage of data on a *parameter server*. ROS is language independent, even if the majority of the open-source code and libraries is written in C++ and Python. ROS is a software development kit for robotic applications, with the main aim of supporting code reuse in robotic research and development. Thanks to this, I have been able to develop the localization system integrating open-source packages and libraries with code written by me in C++. Moreover, ROS is a powerful framework to test the robotic system in simulation before developing ad implementing it in hardware. Our outdoor localization system is a set of nodes, each performing its own task, communicating together through topics. The software drivers reading from sensors, the processes to elaborate the measurements and the sensor fusion algorithm are all executable nodes exchanging messages between them, composing the localization system software.

The fusion node used in the system is provided by the *robot_localization* package [17], containing nonlinear state estimators for robot in 3D space. It provides both the Extended Kalman filter and Unscented Kalman filter estimator. This package allows the fusion of an arbitrary number of sensor data with highly customizable configurations. Regarding the detection of the augmented-reality tags in the map, in our project we used the *ar_track_alvar* package [18]. It contains tools to generate AR tags with different size, resolution and data encoding, and provides the software to identify and track the AR tag seen by a 2D or 3D camera.

A very important tool for the development of our localization system has been the *Gazebo* software [19]. Gazebo is a powerful software able to accurately and efficiently simulate robotic systems in complex indoor and outdoor environments. Thanks to the *gazebo_ros_pkgs* package, which creates the interface between ROS

and the Gazebo simulator, we have been able to test the system in a simulated environment before testing it in real scenarios.

5.2 Robot hardware

As described in the introductory chapter, the outdoor localization system has been designed taking into account the tasks the robot is supposed to accomplish. The aim of the project of InnoTech System is to develop a fully autonomous robot able to carry general objects and to provide delivery services, both in indoors and outdoors. For this purpose the company has designed a wheeled differential drive robot big enough to be able to carry food and small packages, and to be equipped with all the sensors and hardware needed by the different systems (figure 5.2). At the time of writing this platform is still under development, hence during this work we have built a four wheels robot for the purpose of testing and validate the localization system and the docking system, developed in a parallel project. The robot is composed by a base and two upper layers. The base houses the motors, the batteries, the on-board computer and the GNSS receiver module to keep the center of mass as low as possible. The second layer houses the motor drivers, the microcontrollers, the inertial measurement unit and the ultrasonic sensors. The stereo camera and the GNSS antenna have been mounted on the third layer, which is at 70 centimeters from the ground. This height is required to maximize the performance of the stereo camera for both the docking and obstacle avoidance system, other than improving the reception of the GNSS signals. All the sensors and microcontrollers are connected to the on-board computer, an Nvidia Jetson Nano running Ubuntu as operating system, in which the software is executed on the ROS environment. The following paragraphs describe the specific hardware used by the outdoor localization system.

Figure 5.2. Mobile robot designed and developed by InnoTech System

Nvidia Jetson Nano

The Nvidia Jetson Nano [20] is a powerful but small computer with a high energy efficiency, perfect for those applications requiring good graphical and computational capabilities at low power consumption, like mobile robotics. It is powered by a 128-core Maxwell GPU, a quad-core ARM CPU with 4GB of RAM memory, able to satisfy small and power-efficient AI systems, requiring from 5 to 10W. In our robot this on-board computer is the *"brain"* of the system, in charge of processing all the data coming from sensors and run the software required to complete the assigned tasks. The connection scheme is reported in figure 5.3. The GNSS

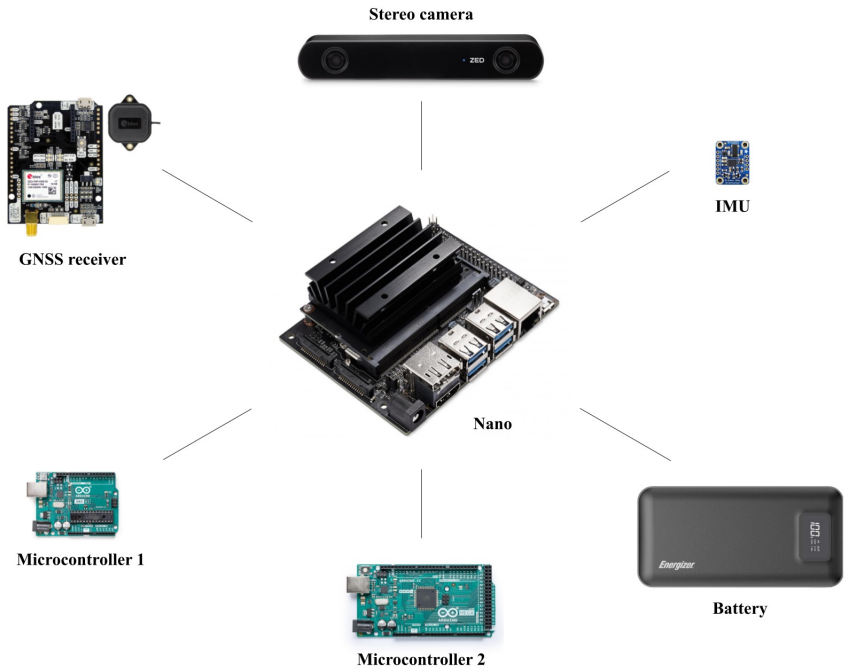

Figure 5.3. Jetson Nano connection scheme

receiver, IMU and stereo camera are connected to the Nano via USB port to continuously send the recorded measurements required by the localization system and docking algorithm. The two microcontrollers are used to read the data from wheel encoders and ultrasonic sensors and send it to the computer, and to control the motors according to the controls coming from the Nano.

GNSS modules

The applications in which our robot can be used require a high localization accuracy also in outdoor environments and the GNSS module must then be chosen accordingly. The performance of the outdoor localization system strongly depends on the quality and precision of the geodetic measurements. For our purposes we chose the Ardusimple simpleRTK2B board [21], based on the u-blox ZED-F9

module [22], which is a low power, multi band GNSS board supporting RTK technology to reach sub-centimeter position accuracy. The simpleRTK2B supports the following operating modes:

- *Standalone*: in this mode the board is able to reach $1m$ position accuracy in few seconds, without range limitations.
- *Base-Rover*: use a simpleRTK2B board as base station to send RTK corrections to one or multiple rovers to achieve $< 1cm$ accuracy with a maximum range of $35km$. In this mode each user must be equipped with its own simpleRTK2B module.
- *Standalone with RTK/SSR corrections*: allows to reach $< 4cm$ accuracy in standalone over large continental areas, receiving RTK-like corrections from a network of base stations.

We used the *base-rover* setup, configuring the second simpleRTK2B module as base station to send the RTK corrections through a dedicated radio link. For our application, where the robot operates in limited outdoor environments like university campus, the RTK covered area of $35km$ from the base is more than enough. The drawback of the radio communication is that a line-of-sight between base and rover is needed. In many different scenarios this condition is difficult to satisfy, hence another communication channel should be used. If an internet connection is available for both base and rover, the RTK corrections can also be distributed through an IP network thanks to the NTRIP protocol.

Inertial Measurement Unit

The Adafruit 9-DOF Absolute Orientation IMU has been chosen as the inertial measurements unit to be mounted on the robot. Based on the Bosch BNO055 System in Package [23], it integrates MEMS triaxial accelerometer, gyroscope and

magnetometer to provide inertial measurements and orientation data. Thanks to an ARM CORTEX-M0 based processor, absolute or relative orientation is obtained through a sensor fusion algorithm that combines all the data from the sensors. The BNO055 outputs the absolute or relative orientation as Euler vector of quaternions, the angular velocity vector, the acceleration vector, the magnetic field vector and the temperature.

Stereo camera

The stereo camera mounted on the robot is the ZED2 cam by Stereolabs [24]. It combines AI and 3D perception to detect and track objects with spatial context. The ZED2 is the first stereo camera that uses neural networks to reproduce the binocular human vision. In our robot this AI powered sensor is the main source of data for the docking and obstacle avoidance algorithm but it is also exploited by the localization system to track augmented-reality tags with high precision and to derive position and orientation information.

Rotary encoders

The rotary encoders used to measure the angular velocity of the motor shafts are based on the *Hall effect*, which allows to detect the presence and magnitude of a magnetic field. By sensing the variation of the magnetic field generated by a permanent magnet attached to the motor shaft, the Hall encoder measures the rotational rate with high precision, from which the linear velocity of our robot is derived.

Figure 5.4 reports the robot we have built during the project to test the developed systems.

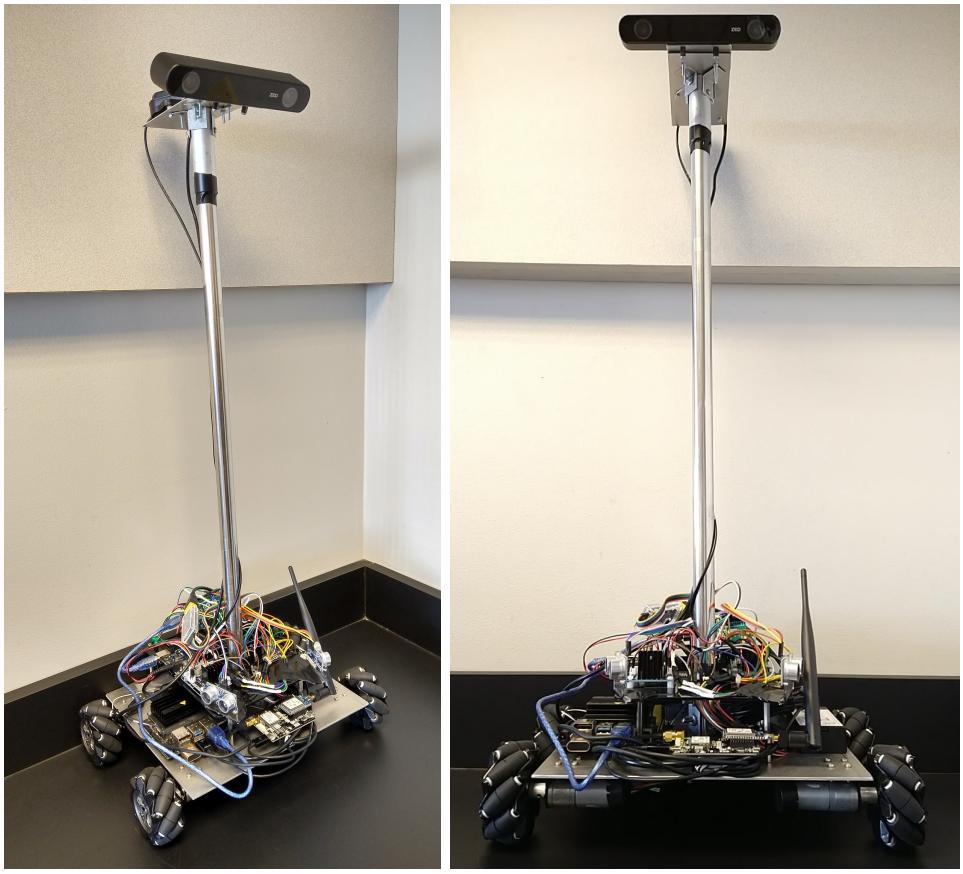

Figure 5.4. Mobile robot employed for the system validation

5.3 Testing

The robot described in the previous section has been specifically built in order to test the developed system under real working conditions. The real tests have been an important part of this project, allowing us to validate what has been developed and tested in simulation. In this way we have been able to understand the real performance and the weaknesses of the outdoor localization. Other than validate the proper functioning of the whole system, the aim of the tests is to demonstrate the benefits introduced by the multi sensor fusion approach: uncertainty reduction, reliability improvement and extension of temporal and spatial coverage in the final

estimate. Moreover, the experiments are also used to compare the performance between the Extended and Unscented Kalman filter as fusion node. Ideally, the best way to validate the localization system would be to compare the estimated position and orientation against the actual values at each time instant, but since in the real world it is almost impossible to know the ground truth of a moving robot as a function of time, the system has been tested by comparing the estimates against checkpoints on the map, known a priori. Consider a checkpoint A with coordinates (x_A, y_A, θ_A). Driving the robot in the exact position of A, assume that the estimated pose and orientation are $(\hat{x}_A, \hat{y}_A, \hat{\theta}_A)$. The estimation errors will be $\Delta\boldsymbol{x} = \sqrt{|x_A - \hat{x}_A|^2 + |y_A - \hat{y}_A|^2}$ for position, and $\Delta\theta = |\theta_A - \hat{\theta}_A|$ for the heading. The procedure followed during the tests is the following:

1. The robot starts its operation in the map origin pointing toward the x axis ($x_A = 0$, $y_A = 0$, $\theta = 0$).

2. Then it is driven following the predefined path, and it is stopped on each checkpoint for some seconds.

3. Meanwhile, all the sensors data is collected in a log file.

4. The test ends when the robot is driven back to the starting position.

The robot must be placed on the checkpoints with very high precision to obtain a meaningful comparison. The tests have been performed on the university campus to reproduce a realistic outdoor environment, moving the robot for several minutes to simulate a real use case. By collecting all the real time data in a log file, we have been able to evaluate the performance of the localization system in post processing, testing the following different configurations on the same data:

- Exploiting only wheel encoders for odometry.

- Exploiting wheel encoders and IMU for odometry.

- Introducing GNSS measurements in the previous configuration.

The stereo camera is used to derive the absolute position and orientation of the robot from the augmented-reality tags, and to reset the state of the system accordingly.

Considering a realistic application of the robot, e.g. to deliver food in complete autonomy to arbitrary points on the University campus, a meaningful test should be long enough in both covered distance and time to properly simulate a real operation. For this reason, the results presented below refers to a test consisting of around 10 minutes of continuous operation for a total of 150 meters covered by the robot. Figure 5.5 shows the outdoor environment with the map origin and

Figure 5.5. Checkpoints on the outdoor map - red axes: map origin and checkpoint 5 - blue icon: checkpoint 1 and 3 - green icon: checkpoint 2 and 4

the checkpoints used to test the accuracy of the system. Starting from the origin, the robot was driven through checkpoint 1(3), checkpoint 2(4) and the map origin

(checkpoint 5), twice in the same test. As anticipated before, the accuracy of the system is measured as the error in the estimated pose and heading of the robot - $\Delta \boldsymbol{x}$ and $\Delta\theta$ - computed as the difference between the output of the localization system and the truth coordinates of checkpoints on the map. Considering the configurations of the system listed before, I will compare their performance by plotting the obtained errors for each checkpoints for both pose and heading when using the Extended Kalman filter. After that I will compare the performance between the Unscented Kalman filter and Extended Kalman filter algorithm for the second and third configuration of the system.

EKF based localization system

Figure 5.6 plots the error in the estimated pose on the five checkpoints for the three configurations mentioned before. Starting from the first configuration where

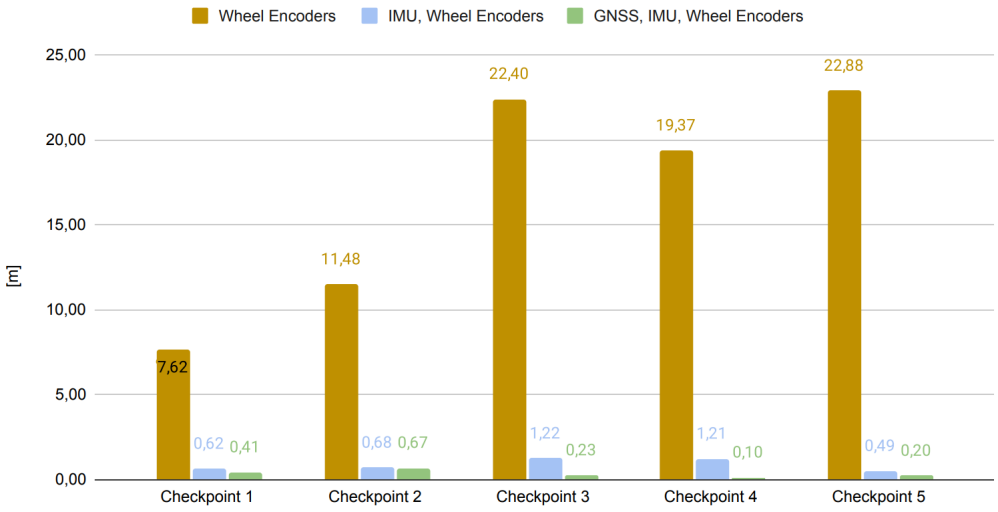

Figure 5.6. Extended Kalman filter pose estimation error

only wheel encoders are used, as expected the poor accuracy of the measurements from these sensors causes poor performance in the position estimation of the robot.

Being a dead reckoning approach, the constant and random errors on the measurements sum over time, causing an increasing estimation error. It is evident how the use of wheel encoders alone is not enough to obtain a valid estimation of the robot pose. Very large errors characterize the estimation immediately after the starting operation of the robot, making this configuration unsuitable for the localization. In this scenario the main source of error is the very low accuracy in the angular velocity. Even if the robot moves straight, the measured angular velocity is not zero but it tends to have a small positive value which causes the system to think the robot is slightly turning on its left when in reality it is not. This error comes from a not perfect balance of the robot, which causes the front right wheel to spin a little faster than the others. Improving the built quality of the robot would also improve the estimation accuracy from wheel encoders, but it is impossible to completely remove this kind of problems and the error on the estimation is only a matter of operational time. Unpredictable problems may always characterize the robot operation ruining the final estimation, like for instance:

- a not planar ground, which may cause the robot to lean on three wheels only, with the fourth one spinning faster than the others
- something stuck on a wheel, blocking it and making it slip on ground

Nevertheless, even if encoders alone are not enough for our purpose, they can help in the overall estimation when combined with other sensors, as we show in the following.

Considering now the second configuration where the IMU is used in addition to encoders, the obtained results are far better than before. The angular velocity is now taken from the gyroscope of the IMU, which provides a more accurate measurement with respect to wheel encoders. However, still being a dead reckoning approach, all the unavoidable small errors on the measurements coming from both

encoders and IMU end up in an increasing estimation error when integrated over time. For the operational time and covered distance of our test, the estimated pose reaches more than 1.2 meters of error, which is no more acceptable for the type of outdoor environment in which the robot is supposed to move. This configuration can be a valid solution only in the case of a short operation when the robot has no GNSS signal.

Moving to the third and last configuration, we obtain the greatest accuracy. The GNSS measurements, being absolute and very accurate, make the filter always correct the possible estimation errors from other sensors. With respect to before now the estimation error does not increase over time but stays always below $0.7m$ during the test. As already discussed, the great accuracy of the GNSS in RTK mode would be enough to estimate the pose of the robot, but since the continuity of the signal is not guaranteed, the multiple sensor fusion algorithm is required to keep the error low during the whole operation of the robot. Despite the short operational time, even during our test the GNSS service was not constant, but lower accuracy peaks were experienced at certain time instants. In the first two checkpoints we experienced a lower estimation accuracy since the GNSS started working in RTK mode only after some minutes from the beginning of the test. Moreover, we placed checkpoint 1 very close to a tall building to partially obstruct the sky visibility in that region and to make the GNSS work in sub-optimal conditions. Nevertheless, thanks to the fusion algorithm the estimated position remained accurate, avoiding discrete jumps of the robot pose due to low quality GNSS measurements. The set of pictures in figure 5.7 shows the estimated position of the robot coming from the localization system and the GNSS measurement at the same time, for consecutive instants around checkpoint 1. As we can see, when

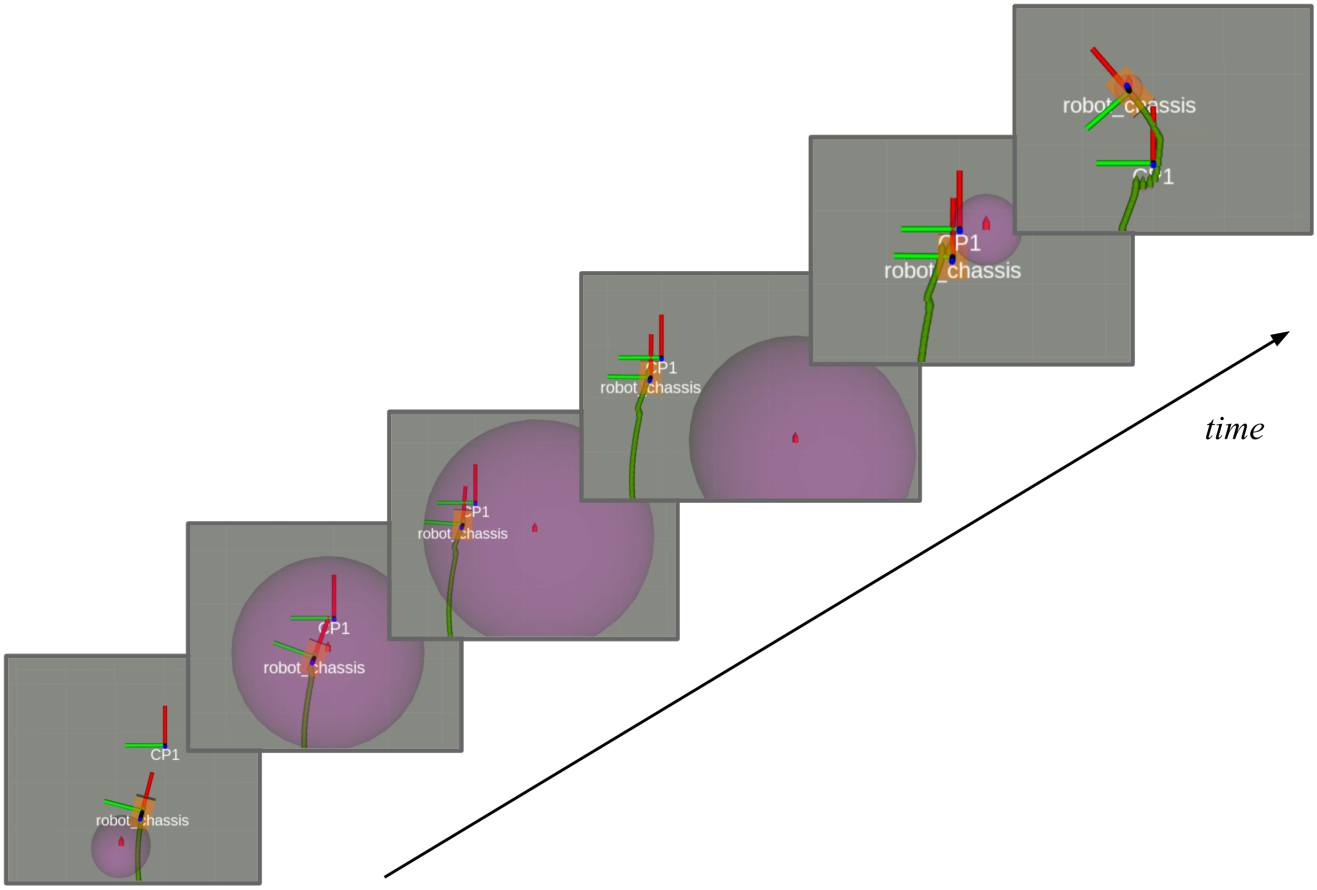

Figure 5.7. GNSS failure test: the robot is represented by the axes with orange box - Checkpoint 1 is CP1 - GNSS measured position is the red dot, with the purple cloud representing its variance - the robot trajectory is the green set of points.

the robot moves closer to checkpoint 1 (beside the building) the GNSS measurements lose accuracy and the given position moves of some meters with respect to the true position. However, the estimated position coming from the fusion node remains close to checkpoint 1, the true position of the robot. This is thanks to the Kalman filter algorithm which, as explained before, weights the contribution of each sensor based on the variance characterizing the measurements, and then the robot pose is estimated trusting more the data coming from IMU and wheel

encoders.

Moving now on the heading information, from figure 5.8 we can see how the estimation error is the same for both configurations and increases with time and distance. This is due to the fact that the robot's heading is uniquely derived from the same sensor - the IMU - for both systems, and the small errors accumulate over time causing a drift in the orientation. An important difference between the two configurations is evident in correspondence of checkpoint 5, before which the orientation from IMU has been corrected by the reference orientation computed from GNSS measurements. This method allows to have an absolute reference information of the robot heading and so to correct possible errors on its estimation. Without the reference orientation estimation techniques described in section 5.1.2,

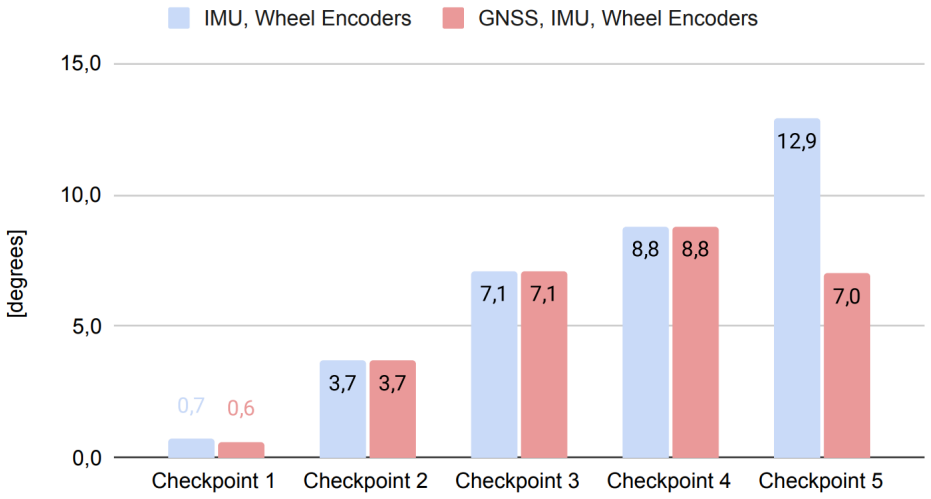

Figure 5.8. Extended Kalman filter heading estimation error

the error in the robot's heading would continuously increase along time as shown in the plot for the IMU + encoders configuration. This result demonstrates the importance of the fusion of GNSS measurements or landmark data in the orientation estimate, and in general the need to periodically have an accurate and absolute

heading information to avoid divergence in the estimated values during the robot operation.

Extended vs Unscented Kalman filter

Figure 5.9 plots the estimation errors for the five checkpoints obtained with the EKF and with the UKF algorithm. Both algorithms obtain very good results,

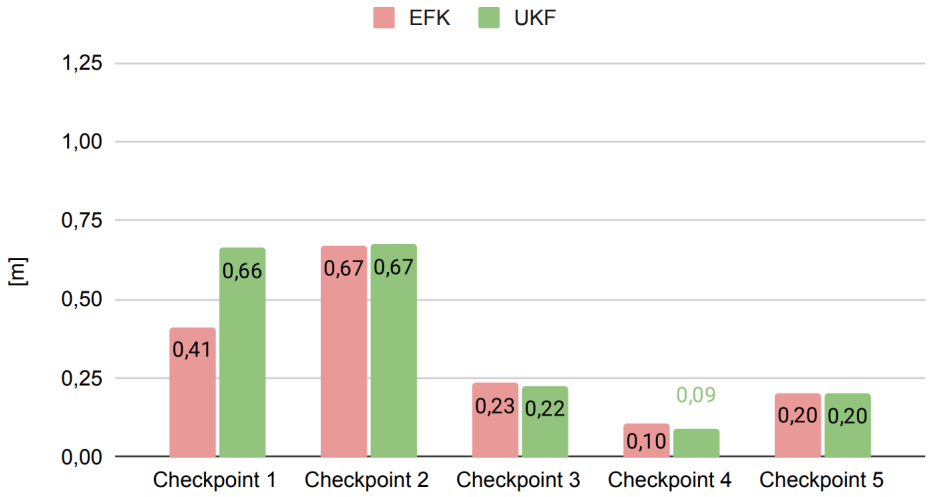

Figure 5.9. Comparison between Extended Kalman filter and Unscented Kalman filter pose estimation errors

especially for checkpoint 3, 4 and 5, corresponding to the RTK operation mode of the GNSS. Apart from the first checkpoint, the results obtained by the two algorithms are very similar and then they can be considered as equivalent in terms of performance. This is not true in general, since the EKF may lead to unreliable estimates if the system is strongly nonlinear, while the UKF, through a deterministic sampling approach, is able to accurately model any nonlinearity. The reason behind the obtained result is that the motion on the two-dimensional plane of our robot is greatly approximated by the linearization of the Extended Kalman filter algorithm, and the peculiarity of the Unscented version do not add any advantage

in terms of estimation accuracy. The same concept is valid for the heading estimation, where both EKF and UKF obtained the exact same results.

For what concerns landmarks localization, the position and orientation data derived from the three-dimensional image of an AR tag are not fused in the Kalman filter with the other measurements, but they are rather used to reset the state of the filter. This design choice is justified by the high precision in the estimated absolute position and orientation of the robot from the tag. Considering the optimal conditions during the test discussed above, with GNSS operating in RTK mode, the average estimation error in the output of the fusion algorithm was around $20cm$ for position and 7° for heading. The accuracy in the data derived from AR tags with the ZED2 camera mounted on our robot is instead around 5 to 10 cm and 3° respectively, lower then the average error in the output of the filter. Therefore, resetting the state with these reference values will lower the estimation error in the robot localization.

5.4 Conclusions

In this work we presented our solution to the problem of mobile robot localization in outdoor environments. We went through the development and implementation of the localization system and we presented the outcomes of the real tests used to validate our work. The results just presented in the previous sections demonstrate the improvements in the estimation accuracy obtained from a multiple sensor fusion approach. Combining the measurements from GNSS, IMU, encoders and stereo camera through an algorithm based on nonlinear Kalman filtering, we have been able to obtain position and orientation estimates accurate enough to allow a mobile robot to localize itself in outdoors, getting closer to the final aim of a completely autonomous navigation. Localizing the robot in outdoor environments

only solves half of the problem. An autonomous mobile robot must be able to localize itself regardless the type of environment in which it operates, accurately and continuously in time. InnoTech Systems has developed a localization system to satisfy these requirements in indoors, while in this document we described our solution for outdoors. Future steps in the project could regard the integration of the two systems to guarantee a seamless localization while moving from outdoor to indoor and vice versa.

Bibliography

[1] Uwe Jahn et al. «A Taxonomy for Mobile Robots: Types, Applications, Capabilities, Implementations, Requirements, and Challenges». In: *Robotics* 9 (Dec. 2020), p. 109. DOI: 10.3390/robotics9040109.

[2] Yeong-Hwa Chang, Ping-Lun Chung, and Hung-Wei Lin. «Deep learning for object identification in ROS-based mobile robots». In: *2018 IEEE International Conference on Applied System Invention (ICASI)*. 2018, pp. 66–69. DOI: 10.1109/ICASI.2018.8394348.

[3] Jing Xin et al. «Application of deep reinforcement learning in mobile robot path planning». In: *2017 Chinese Automation Congress (CAC)*. 2017, pp. 7112–7116. DOI: 10.1109/CAC.2017.8244061.

[4] Mary B. Alatise and Gerhard P. Hancke. «A Review on Challenges of Autonomous Mobile Robot and Sensor Fusion Methods». In: *IEEE Access* 8 (2020), pp. 39830–39846. DOI: 10.1109/ACCESS.2020.2975643.

[5] Francisco Rubio, Francisco Valero, and Carlos Llopis-Albert. «A review of mobile robots: Concepts, methods, theoretical framework, and applications». In: *International Journal of Advanced Robotic Systems* 16 (Mar. 2019). DOI: 10.1177/1729881419839596.

[6] J. Borenstein et al. «Mobile robot positioning: Sensors and techniques». In: *Journal of Robotic Systems* 14.4 (), pp. 231–249. DOI: https://doi.org/10.1002/(SICI)1097-4563(199704)14:4<231::AID-ROB2>3.0.CO;2-R.

[7] D.L. Hall and J. Llinas. «An introduction to multisensor data fusion». In: *Proceedings of the IEEE* 85.1 (1997), pp. 6–23. DOI: 10.1109/5.554205.

[8] François Peyret et al. «Better use of Global Satellite Systems for Safer and Greener Transport». In: (Sept. 2015).

[9] Christopher Hegarty and Elliott Kaplan. *Understanding GPS Principles and Applications, Second Edition*. 2005.

[10] Estefania Munoz Diaz, Dina Bousdar Ahmed, and Susanna Kaiser. «16 - A Review of Indoor Localization Methods Based on Inertial Sensors». In: *Geographical and Fingerprinting Data to Create Systems for Indoor Positioning and Indoor/Outdoor Navigation*. Ed. by Jordi Conesa et al. Intelligent Data-Centric Systems. Academic Press, 2019, pp. 311–333. ISBN: 978-0-12-813189-3. DOI: https://doi.org/10.1016/B978-0-12-813189-3.00016-2. URL: https://www.sciencedirect.com/science/article/pii/B9780128131893000162.

[11] Manon Kok, Jeroen D. Hol, and Thomas B. Schön. «Using Inertial Sensors for Position and Orientation Estimation». In: *Foundations and Trends® in Signal Processing* 11.1-2 (2017), pp. 1–153. ISSN: 1932-8346. DOI: 10.1561/2000000094. URL: http://dx.doi.org/10.1561/2000000094.

[12] Federico Castanedo. «A Review of Data Fusion Techniques». In: *The Scientific World Journal* 2013 (Oct. 2013), p. 19. DOI: 10.1155/2013/704504.

[13] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. ISBN: 9780262201629.

[14] Dan Simon. *Optimal State Estimation: Kalman, H-infinity and Nonlinear Approaches*. Wiley Interscience, 2006. ISBN: 9780471708582.

[15] *Robot Operating System*. URL: https://www.ros.org/.

[16] *Robot Operating System introduction*. URL: http://wiki.ros.org/ROS/Introduction.

[17] *robot_localization package*. URL: http://docs.ros.org/en/melodic/api/robot_localization/html/index.html.

[18] *ar_track_alvar package*. URL: http://wiki.ros.org/ar_track_alvar.

[19] *Gazebo*. URL: http://gazebosim.org/.

[20] *Nvidia Jetson Nano*. URL: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/.

[21] *Ardusimple simpleRTK2B*. URL: https://www.ardusimple.com/simplertk2b-receivers/.

[22] *u-blox ZED-F9*. URL: https://www.u-blox.com/en/product/zed-f9p-module.

[23] *Bosch BNO055*. URL: https://www.bosch-sensortec.com/products/smart-sensors/bno055/.

[24] *ZED2 camera by Stereolabs*. URL: https://www.stereolabs.com/zed-2/.