Imagem

Fábio Filipe da Costa Monteiro

LOCALIZATION OF FIELD ROBOTS
USING RTK-GNSS/INS SYSTEMS

Masters Dissertation in
Electrical and Computer Engineering
September 2016

· U C ·

UNIVERSIDADE DE COIMBRA

• U  C •

Localization of Field Robots Using RTK-GNSS/INS Systems

Dep. of Electrical and Computer Engineering at University of Coimbra

Masters Dissertation in Electrical and Computer Engineering

Fábio Filipe da Costa Monteiro

Supervisor: Professor Doutor Lino José Forte Marques

Co-Supervisor: Professor Doutor António Paulo Mendes Breda Dias Coimbra

Jury:

President: Professor Doutor Urbano José Carreira Nunes

Members: Professor Doutor Lino José Forte Marques

Professor Doutor Paulo José Monteiro Peixoto

September 2016

Acknowledgements

First of all, I would like to thank my parents, to whom, over the years were always my greatest support, being patient, encouraging, suffering, persevering, and friends. Without their help, this dissertation would not have been possible. And of course, all the other family members who were also interested and concerned about this phase of my life are not forgotten.

I want to manifest a sincerely gratitude to my supervisor, Professor Doutor Lino Marques, for all his help and guidance that he has given me over the last year. And also to Professor Doutor António Paulo Coimbra, for his advice and availability since the beginning.

To my true friends, who appeared in my life in different circumstances, occasions, and places, specially the ones that I have met in my early years in Coimbra. To João Carlos Horta, that I have met in the first year of university, went with me in Erasmus, and we've been friends since the first day.

Last but not least, for everybody that in some way had contributed to this dissertation, I want to express my deepest acknowledgments.

Resumo

Quando se utiliza robots autónomos umas das tarefas mais importantes é estimar a sua localização com exatidão e precisão, e ao mesmo tempo ter noção de tudo o que o rodeia. O sucesso de simples tarefas como a navegação e mapeamento depende bastante do bom conhecimento da posição do robot.

O Sistema de Navegação por Satélite (GNSS-Global Navigation Satellite System) e o Sistema de Navegação Inercial (INS-Inertial Navigation System) são dois sistemas de navegação muito usados nos dias de hoje para ajudar a estimar a localização. Ambos os sistemas têm vantagens e desvantagens, mas quando combinados num sistema único de navegação GNSS/INS, permite-nos combinar as vantagens de ambos os sistemas.

Nesta Tese de Mestrado vão ser explicados os principais problemas presentes em cada um destes sistemas de navegação e como os podemos reduzir. É também explicado e implementado um sistema de fusão dos dois sistemas, de modo a obtermos uma solução com maior precisão. Por fim no ultimo capítulo, o sistema que foi implementado é testado e comparado com outros Receptores GNSS que se encontram à venda no mercado com preços e características bastante diferentes.

Palavras-Chave: GNSS, RTK, IMU, Kalman, ROS

Abstract

When dealing with autonomous robots one of the most important tasks is to estimate an accurate and precise localization of the robot and his surroundings. The success of simple tasks as navigation and mapping depend on a good robot's position knowledge.

The Global Navigation Satellite System (GNSS) and the Inertial Navigation System (INS) are two navigation systems commonly used nowadays that can be used to determine the robot's localization. Both systems have advantages and disadvantages, but they can complement each other when fused together in a GNSS/INS Navigation System.

In this dissertation, it is going to explain the main problems when using this type of systems and how to reduce them. It's also explained and implemented a data fusion of both systems in order to achieve a higher precision. In the last chapter the implemented fusion is tested, and compared with others GNSS receivers available in the market with different prices and characteristics.

Keywords: GNSS, RTK, IMU, Kalman, ROS

Contents

List of Figures

List of Tables

Acronyms

ADC Analog-to-Digital Converter

AHRS Attitude and Heading Reference System

DD Double Differences

DGPS Differential Global Positioning System

DOF Degrees of Freedom

ECEF Earth-Centered-Earth-Fixed

ECI Earth-Centered-Inertial

EKF Extended Kalman Filter

EMI Electromagnetic Interference

ENU East-North-Up

GNSS Global Navigation Satellite System

GPS Global Positioning System

IEEE Institute of Electrical and Electronics Engineers

ILS Integer Least Square

IMU Inertial Measurement Unit

INS Inertial Navigation System

LLI Loss of Lock Indicator

LTE Long-Term Evolution

MEMS MicroElectroMechanical Systems

NED North-East-Down

NMEA National Marine Electronics Association

NWU North-West-Up

PPP Precise Point Positioning

PRN Pseudorandom Noise

RINEX Receiver Independent Exchange Format

RTCM Radio Technical Commission for Maritime Services

RTK Real Time Kinematic

SBAS Satellite-Based Augmentation System

SD Single Differences

SPS Standard Positioning Service

TDOA Time Difference of Arrival

UAV Unmanned Aerial Vehicles

UERE User Equivalent Range Error

1. Introduction

1.1. Motivation

In the last decades, research in autonomous vehicles has been growing and involving more and more people due to their vast application potentials. Every year more mobile robots and electronic devices are joining our daily life.

The improvements made in the processing power and the size reduction in the microcontrollers and similar processing boards, allowed the field of robotics to boost tremendously. These autonomous vehicles can have a tremendous importance for field robots and they can perform tedious and dangerous tasks to humans, with faster response times or in situations without human access. They can be used in scientific experiments using autonomous boats to patrol rivers, gathering water parameters and perform bathymetry studies [Fraga *et al.*, 2013]. In military operations using autonomous drones to perform mapping and surveillance of critical areas. Or in an opposite sense, to clear post-combat regions of land mines using the robots to map wide areas without human interaction [Cabrita *et al.*, 2015]. To perform planetary exploration with mission like the Mars Exploration Rovers, that uses autonomous robots to land, explore, and study the geology of the Martian surface. And of course, the most actual contest that every big brand in the tech and mobile vehicles manufacturers is trying to win, to create a true self driving car.

When these robots navigate outdoors they suffer from the unpredictability of everything that surround them. Objects that were stationary one moment ago, can start moving and becoming obstacles. Sometimes these objects have different contours than what we expect and out of the sensors range, which will lead to difficult situations for the robot to handle. The terrain conditions can change from day to day and undergo interactions by humans and weather conditions, making an easily travelled routed today inaccessible tomorrow. All these problems make outdoors navigation a lot more difficult than what we normally find indoors.

In order to be truly called autonomous, these robots must be able to sense its environment, navigate without human input, and be able to complete their tasks. To performing such critical tasks, one of the most important aspects needed for these robots is an accurate and repetitive localization.

On many critical occasions it is not allowed for a robot to drive by the same place twice and report different positions for that place, as such situation can lead catastrophic results.

A localization can be referred as an absolute or relative position depending on our reference point and the type of measurements that we are making. If our starting position is just a random point that we can't define its position globally, then our measurements will be relative to the reference point and known as a relative position. On the other hand, if we are starting from a reference point that we know the global position or using sensors providing global positioning, then we can call the position absolute.

Some of the most common techniques used to estimate a position include the odometry, the Global Navigation Satellite System (GNSS), and the Inertial Navigation System (INS). From all these systems, the only one that can obtain a truly absolute position is the GNSS, but at the same time it relies on third party services which can be disabled or the satellites it depends on be temporarily occluded by obstacles.

Most GNSS receivers commonly used in commercial applications have an User Equivalent Range Error (UERE) accuracy of about 6 meters within a 3σ deviation provided the measurements are under optimal conditions, and have low update rates. With this performance, these systems alone are not suitable to be used in robots and therefore other techniques need to be used.

To improve the position estimation, one should reduce the errors associated with the GNSS receivers and at the same time implement an Inertial Navigation System, so that our new positioning system can have two independent measurements sources. As the INS can provide higher accuracy in the short-term and also good update rates, the estimated position obtained from the data fusion has the advantage of being accurate in short and long term, and at the same time have good update rates for a robot to navigate dynamically.

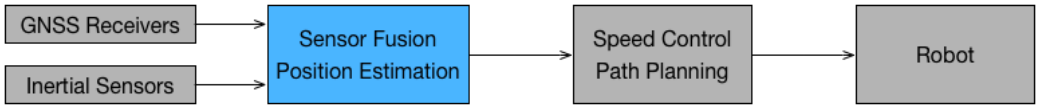


Figure 1: Position Estimation Block Diagram

1.2. Objectives

The purpose of this dissertation is to search and implement a sensor fusion algorithm that can combine the absolute measurements obtained from the GNSS receivers with the relative measurements from Inertial Sensors, in order to improve the accuracy of the estimated position. Before presenting the data fusion, both technologies are explained and their most common errors investigated as well as some techniques that can minimize them.

This dissertation will mainly focus on the blue block shown in Figure 1. The prime idea of this estimation block is that it will only deal with the GNSS and INS measurements, and predict an accurate position that can then be fed to a speed control and path planning algorithm.

The State of the Art in localization systems hasn't changed a lot in the latest years. Cost-effective systems with good accuracy have become more common due to advances in technology and consequently the price dropped, however these technologies have been used in high quality systems for some time.

The simplest technique to obtain an absolute positioning is to use a GNSS receiver, however these type of devices have some disadvantages. Firstly, they need sky view to be able to receive the satellite signals, therefore indoor usage is out of the question. Secondly, GNSS receivers have low update rate. Most receivers have updates rates between 1 and 5Hz, which in systems with fast dynamics can become problematic. Thirdly, the measurement errors are usually in the order of 3 meters even when the system is in a good environment. These last two errors can be attenuated when using higher quality receivers. Devices ranging from 300€ to 15.000€ can reduce the measurements errors to near half a meter by tracking more satellites, using multiple frequencies and antennas, different measuring techniques, and have update rates up to 100Hz.

To obtain solutions with higher accuracy it's necessary to implement algorithms as the Real Time Kinematic (RTK). This technique is able to overcome the errors inherent from the GNSS Receivers, allowing an accuracy up to a few centimeters when the system is operating in good conditions. To implement these kind of solutions and obtain low measurements errors, it's necessary to use two GNSS receivers operating within an area of $20\,km$, and with both receiver communicating between them. Theoretically, as both receivers are under similar conditions the errors arriving at

them are also similar and can cancel each other out. Low cost solutions with low update rates can be purchased from 1.000€, and higher quality ones can go up to 50.000€.

All these GNSS receivers and techniques suffer from a common problem, which is the satellite availability. If the receiver is near trees, high buildings, under bridges, or near other kind of obstacles that can momentarily block the satellite signals, the errors in the solution will start to increase.

To avoid these errors, started to appear localization system with multiple measurement sources and fusing them all together. These type of solutions depend highly from the environment the system is going to be used. Ground vehicles with wheels can fuse the GNSS measurements with Wheel Odometry, Marine vehicles can have measurements from Doppler Sonars, Unmanned Aerial Vehicless (UAVs) also have cameras and Visual Odometry. Simultaneously it can also be used an Inertial Measurement Unit which is more versatile to all applications, as the sensor can provide information about linear acceleration, angular velocity, and orientation. The improvement and price drop in Microelectronic Mechanical Systems (MEMS) devices allowed the introduction of Inertial Measurement Units to the localization systems. As the MEMS IMUs are usually cheap and have good update rates they are a good complement to GNSS receivers. Of course as in all the devices, higher quality MEMS IMUs with low measurements errors and drift rates can cost thousands of Euros.

The information from all these devices is then combined through a data fusion algorithm to estimate a single and accurate solution. The most common fusion algorithm is the Extended Kalman Filter [Fraga, 2012; Kelly, 1994; Lopes, 2011], however there are other possible fusion algorithms, such as the Unscented Kalman Filters (UKF) [Crassidis, 2006; Gustafsson, 2014], State-Dependent Riccati Equations (SDRE) [Nemra and Aouf, 2010], Particle Filters [Gustafsson, 2010], and the Extended or Unscented Rauch-Tung-Striebel smoother Algorithm [Elisson and Gassler, 2014].

In conclusion, a localization system using a RTK-GNSS Receiver fusing the measurements of an IMU and if possible other sensors can produce a final single solution without the three main problems described previously. The RTK-GNSS gives us an accurate solution in good sky view conditions, the introduction of an IMU to the system allows us to remove the problems from the satellite miscommunication, and temporary indoors trajectories.

1.3. Goals and Research Questions

When starting a project like this there is always questions that we ask ourselves and that we research in order to find the best answers. Some of them ended up having easy answers, other ones not so much. Here are some of them,

- How can I measure the accuracy of a localization system?
- What accuracy can I expect when using a budget GNSS receiver? Does a moving receiver loses a lot of accuracy when compared to a stationary?
- Should I go for RTK GNSS receivers or a GNSS/INS Sensor?
- If I want to upgrade my GNSS receivers, how does the accuracy improves based on the cost of the receiver? Do I get a better sensitivity? Better update rates? The ability to track new constellations? Integration with the information of other sensors?
- And if I want to upgrade my inertial sensors, does the accuracy improves based on the cost of the sensor? Do I get a better sensitivity? Better update rates? Lower random walks? Better update rates?
- How does the measurements errors from these sensors evolve over time? And what can I do to minimize them?

1.4. Outline

This dissertation is divided into 6 chapters. **The First Chapter**, contains a brief introduction to this project, explains the main sections where the dissertation is going to focus, and the existing localization methods. **Chapter Two**, describes the two Navigation Systems used: the Global Navigation Satellite System and the Inertial Navigation System. Some of the most common errors that can be expected when using these type of navigations are explained as well as the materials that are going to be used during this dissertation. **In Chapter Three** explains the various types of data fusion, cites similar projects, and the implemented Data Fusion Algorithm. **Chapter Four** shows us the results of the implemented algorithm compared with other solutions on the market. **Chapter Five** is the conclusion, where some ideas are given for future work and how the implemented data fusion could be improved. It also describes some of the challenges found during this dissertation. Finally, **Chapter Six** it's composed by all the extra information in the appendices.

2. Localization

This chapter covers the two types of navigation systems that are going to be used during this dissertation. It will list the electronics used, their characteristics, and similar devices that could be used as an alternative to improve the accuracy of the localization. It will also explain how both systems work, their most common errors and how to minimize them so the whole system can be more accurate. Finally, it demonstrate how both navigation systems were implemented.

When trying to choose the most appropriate localization system to implement, it's important to understand how the different localization methods obtain their measurements. Most of the localization systems can be divided in the following categories: Triangulation, Trilateration, and Multilateration.

Triangulation, consists in the process of determining the localization of a position by measuring the angles formed by known landmarks. It's the first localization method known and it's being used by the man since the antiquity, in situation as to find the height of the pyramids, or in nautical navigation using instruments as the sextant and the astrolabe (See Figure 2). Nowadays it can be implemented with multiple cameras to estimate a 3D position.

If we wants to measure the distance d or h using triangulation, we start by creating a triangle with a known baseline distance $S = A\bar{B}$, measure the angles formed by α and β, and the unknown distance can be found using

$$d = S \cdot \frac{\sin\alpha \, \sin\beta}{\sin(\alpha+\beta)} \qquad h = S \cdot \frac{\sin\alpha \, \sin\beta}{\sin(\alpha-\beta)} \tag{1}$$

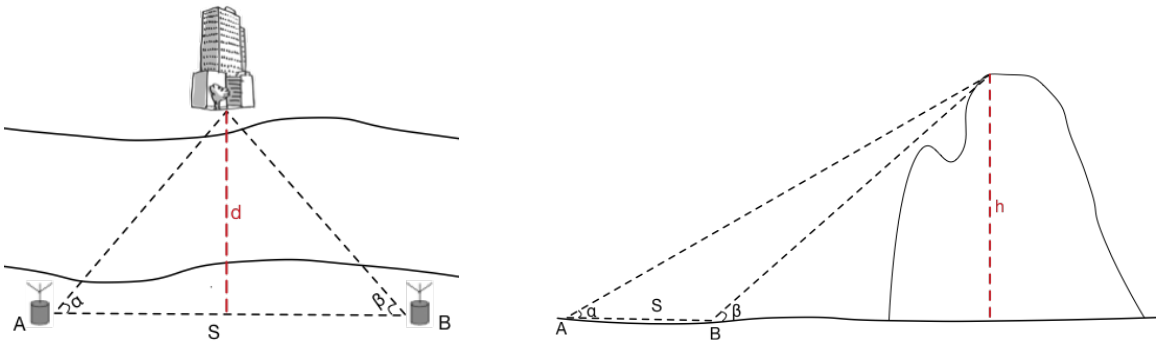


Figure 2: Triangulation

Trilateration, consists in the process of determining the localization of a position by measuring the distance to known landmarks using the geometry of circles and spheres (See Figure 3).

This is the method used by GNSS Receivers to estimate a localization on earth. If the receiver can track three satellites, then three theoretical spheres are created on the earths surface and their intersection represent the Latitude and the Longitude (2D Fix) of the receiver. If more satellites are being tracked by the receiver, then the solution can be estimated by more spheres and the receiver can evaluate the Latitude, Longitude, and Altitude (3D Fix).

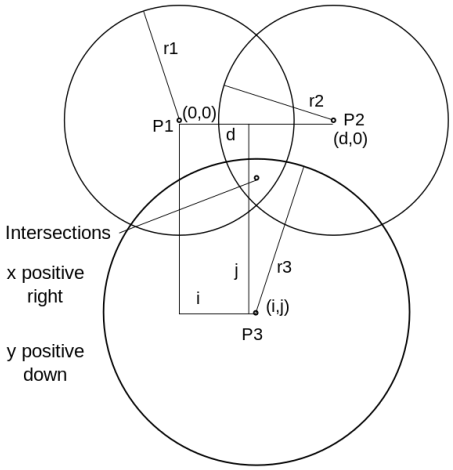


Figure 3: Trilateration

The position can be estimated in a 2D plane, using three spheres with centres $P1, P2, P3$, and radii $r1, r2, r3$. The equations for the three spheres are represented by

$$r_1^2 = x^2 + y^2 + z^2, \qquad r_2^2 = (x-d)^2 + y^2 + z^2, \qquad r_3^2 = (x-i)^2 + (y-j)^2 + z^2 \tag{2}$$

The intersection point located at (x, y, z) that satisfies all three equations, can be found using,

$$\begin{aligned} x &= \frac{r_1^2 - r_2^2 + d^2}{2d} \\ y &= \frac{r_1^2 - r_3^2 + i^2 + j^2}{2j} - \frac{i}{j}x \\ z &= \pm\sqrt{r_1^2 - x^2 - y^2} \end{aligned} \tag{3}$$

Multilateration, consists in the process of determining the localization of a position by measuring the Time Difference of Arrival (TDOA) of a broadcasted signal at multiple stations with known locations. The principle is quite similar to trilateration, except that now the triangles, circles, and spheres, are replaced by a hyperbola (2D), or a hyperboloid (3D) (See Figure 4).

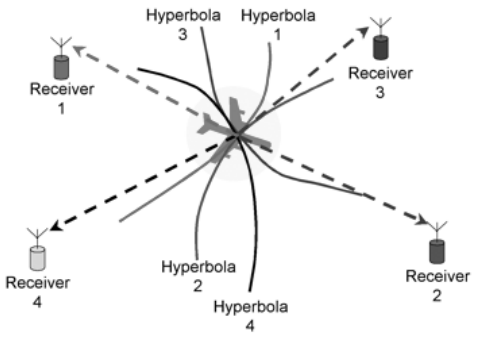


Figure 4: Multilateration

To obtain an accurate position it's also needed at least four stations, however the receivers don't need to know the instant the signal was broadcasted and hence there is no need for time synchronisation among transmitters and receivers. This method is commonly used to estimate the airplanes position.

Considering the emitter $\overrightarrow{E} = (x, y, z)$, and multiple receivers $\overrightarrow{P_i} = (x_i, y_i, z_i)$, the distance between

them is given by

$$R_i = \left| \overrightarrow{P_i} - \overrightarrow{E} \right| = \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} \tag{4}$$

If we have a receiver placed at the origin, its distance becomes

$$R_0 = \sqrt{x^2 + y^2 + z^2} \tag{5}$$

Which can also be found using the broadcasted wave speed (υ), and the transmitting time (T_i). Therefore, the TDOA equation for receivers P_i and P_0 is

$$\begin{aligned} \upsilon \tau_i &= \upsilon T_i - \upsilon T_0 \\ \upsilon \tau_i &= R_i - R_0 \end{aligned} \tag{6}$$

Equation 6 is the hyperboloid described previously, and when using 4 receivers it leads to 3 non-linear equations that allows to solve the three unknown values (x, y, z).

2.1. Global Navigation Satellite System (GNSS)

A GNSS is a localization method based on signals transmitted from satellites that are in orbit around the Earth. These signals are received by GNSS receivers and their messages decoded into information such as localization, speed, and time.

The most common system used nowadays, is the Global Positioning System (GPS) which is owned by the United States Department of Defense, but can be used freely by civilians and militaries [NAVSTAR, 2008]. As the GPS is owned by the EUA and can be deactivated if they choose to do so, other countries started to develop their own systems. Russia come up with the GLONASS, China with the BeiDou, and the European Union with the Galileo project, which aims to provide an independent high precision positioning system, that can not be disabled by governments. As some of the systems are relatively new their number of satellites is still low, which means that some places on earth can lack the visibility of their satellites (See Table 1).

These systems transmit their signals using multiple frequencies and different modulations, explaining them all in this dissertation would require a lot of time and would make the document rather extensive. Therefore, in the next sections it will only be explained the most common one, the Global Positioning System (GPS). For more informations on how these systems work there are good books explaining them, such as [Jeffrey, 2010] and [Petrovski, 2014].

Table 1: Number of operational satellites per constellation (Sep. 2016)

System	GPS	GLONASS	BeiDou	Galileo
Satellites	31	23	21	12

2.1.1. GNSS Receivers

The GNSS receivers are the main devices used to estimate a position of field platforms as they can provide absolute localization.

During this dissertation were used two u-blox NEO-6P receivers, and two Septentrio AsteRx4 receivers (Figure 5). The u-blox were chosen because they are inexpensive and can receive Raw Observable Messages from the satellites, which allows to use a technique known as Real Time Kinematic (RTK) Satellite Navigation. This technique consists of using two receivers at the same time, compare the signals from both receivers and cancel the common errors. The Septentrio are higher quality receivers that can track all satellite constellations, receives signals from multiple frequencies, has better updates rates, and better precision. The technical specifications for both receivers will be explained in Section 4.2.

There are a lot GNSS receivers can be used for this purpose, u-blox has the new NEO-M8 receivers which can receive signals from multiple constellations and some of them perform RTK solutions. There is also NVS, NovAtel, and Trimble, all of them are popular manufacturers with good quality receivers.

2.1.2. GPS Operation Principle

The satellite navigation uses trilateration to estimate a position by measuring the distance between the satellite and the receiver. This distance can be obtained by measuring how long a radio signal takes to reach the receiver after being transmitted by the satellite. Each satellite being measured creates a sphere of possible locations where the receiver can be located, and by intersecting at least four of these sphere the exact position can be found.

A broadcasted GPS satellite signal consists mainly on a message with the time of transmission and a Pseudorandom Noise (PRN) code. When the signal is received, the receiver generates an

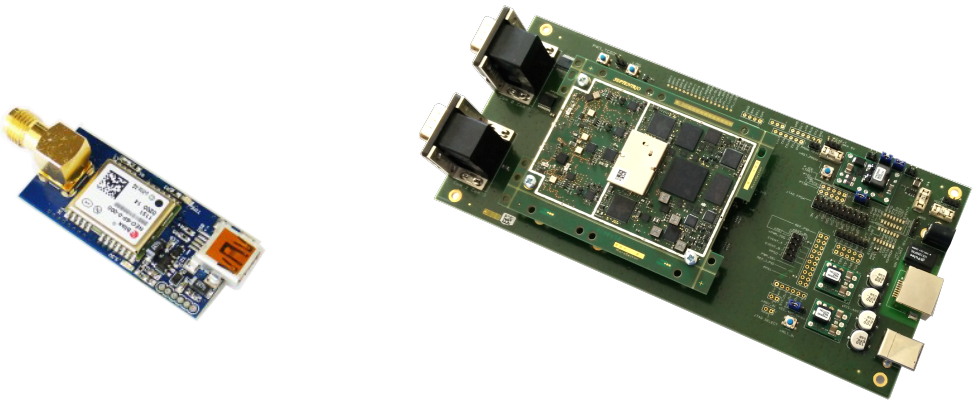


Figure 5: u-blox NEO-6P and Septentrio AsteRx4 GNSS Receivers

identical signal and shifts it in time until it correlates with the received code. The time shift corresponds to the propagation time of the signal, and multiplying the propagation time by the speed of light gives us the distance between the satellite and the receiver.

As the signals don't travel through vacuum, more complex models are needed to estimate the distance. Higher quality receivers use dual frequency measurements to estimate this value with higher accuracy. As different frequency signals take different times to travel the same path, by comparing the delays introduced by the different frequencies, the receivers can estimate with higher accuracy the position.

2.1.2.1. Signals

The GPS constellation broadcasts its signals using three different frequencies. They are generated by the onboard atomic clock and are known as L1 (1575.42 MHz), L2 (1227.60 MHz) and L3 (1176.45 MHz). Superimposed on these frequencies are a PRN code unique to each satellite and navigation messages.

2.1.2.2. Navigation Messages

The navigation message is divided in 25 frames of data, with each frame subdivided into 5 subframes. These messages are broadcasted at 50Hz, taking the receiver 6 seconds to receive a subframe, 30 seconds to receive one frame, and 12.5 minutes to receive all 25 frames.

To decrease the initial time to obtain a position, the first three subframes are identical in all the frames, allows the receiver to obtain all the critical satellite specific information within 30 seconds.
Subframe 1, contains clock corrections for the transmitting satellite and parameters describing the accuracy and health of the broadcasted signal.
Subframe 2 and 3, contain ephemeris parameters used to compute the location of the satellite.
Subframe 4 and 5, contain components of the almanac, but each frame contains only 1/25th of the complete almanac, so the receiver must retrieve all the 25 frames to be able to decode the data.

2.1.3. GPS Observables

There are two GPS observables known as Code Pseudoranges and Carrier Phases, whose measurement yield the distance between the satellite and the receiver.

2.1.3.1. Code Pseudorange

The Pseudorange is the geometric distance between the satellite at the time of signal transmission and the receiver at the time of signal reception. The simplest model for the pseudorange can be

obtained when the signal travels through vacuum and there are no clock errors involved. In this case, the pseudorange P_r^s between receiver r and satellite s can be determined by

$$P_r^s(t_r, t_e) = (t_r - t_e)c \tag{7}$$

where c is the speed of light, t_r is the reception time, and t_e is the emission time. The value of t_e is known from the navigation message, and the value of t_r is found out by the shifting technique described previously (See Figure 6).

Using the satellite position $r^s(t) = [x^s(t), y^s(t), z^s(t)]$ and the receiver position $r_r(t) = [x_r(t), y_r(t), z_r(t)]$, the pseudorange can be expressed as

$$\begin{aligned} P_r^s(t_r, t_e) &= \|r^s(t) - r_r(t)\| \\ &= \sqrt{(x^s(t_e) - x_r(t_r))^2 + (y^s(t_e) - y_r(t_r))^2 + (z^s(t_e) - z_r(t_r))^2} \\ &= \rho_r^s(t_r, t_e) \end{aligned} \tag{8}$$

Adding the clock errors to the pseudorange, it is then represented as

$$\begin{aligned} P_r^s(t_r, t_e) &= ((t_r + \delta t_r) - (t_e + \delta t^s))c \\ &= (t_r - t_e)c + (\delta t_r - \delta t^s)c \\ &= \rho_r^s(t_r, t_e) + (\delta t_r - \delta t^s)c \end{aligned} \tag{9}$$

where δt_r and δt^s are the clock offsets at the receiver and satellite respectively.

Taking into account the ionospheric effects, tropospheric effects, earth tide and loading tide effects, multipath and relativistic effects, as well as all the others remaining errors, the Pseudorange model can be completely represented by

$$P_r^s(t_r, t_e) = \rho_r^s(t_r, t_e) + (\delta t_r - \delta t^s)c + \delta_{ion} + \delta_{tro} + \delta_{tide} + \delta_{mul} + \delta_{rel} + \varepsilon \tag{10}$$

which is one of the most common models for the Code Pseudorange [Xu, 2003].

2.1.3.2. Carrier Phase

The Carrier Phase consists in measuring the phase between the received satellite signal and the receiver generated carrier phase signal. This method is usually used by higher quality receivers to

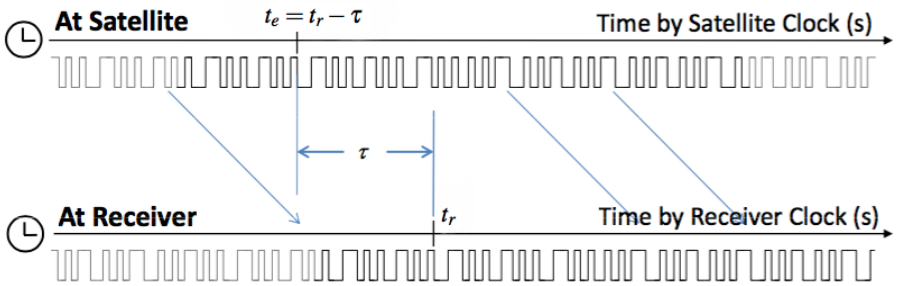


Figure 6: GPS Code Pseudorange [Takasu, 2015]

archive a better accuracy.

If the signals travels through vacuum and there are no clock errors involved, the measured phase ϕ_r^s between the receiver r and the satellite s can be written as

$$\phi_r^s(t_r,t_e) = \phi_r(t_r) - \phi^s(t_e) + N_r^s \tag{11}$$

where t_r and t_e are the reception and emission time of the signal respectively, ϕ_r is the phase of the receiver oscillator, and ϕ_s the phase of the emitted signal. The additional parameter N_r^s, is the number of full phase cycles that have passed between the satellite s and receiver r. This parameter is an ambiguity that the receiver needs to estimate (See Figure 7).

Assuming that the received satellite signal and the generated signal have a frequency f, their phases can be written as

$$\begin{aligned} \phi_r(t_r) &= f(t_r - t_0) + \phi_{r,0} \\ \phi^s(t_e) &= f(t_e - t_0) + \phi^{s,0} \end{aligned} \tag{12}$$

where $\phi_{r,0}$ and $\phi^{s,0}$ are the initial phases of the signal, and t_0 is the initial time. Introduce the clock errors to the Carrier Phase equation 11, becomes

$$\begin{aligned} \phi_r^s(t_r,t_e) &= (f(t_r + \delta t_r - t_0) + \phi_{r,0}) - (f(t_e + \delta t^s - t_0) + \phi^{s,0}) + N_r^s \\ &= f(t_r - t_e) + f(\delta t_r - \delta t^s) + (\phi_{r,0} - \phi^{s,0}) + N_r^s \end{aligned} \tag{13}$$

Expressing equation 13 using the wavelength ($f = c/\lambda$), allows the the Phase Range Φ_r^s to be defined as the Carrier Phase ϕ_r^s multiplied by the wavelength λ, as can be seen in equation 14.

$$\begin{aligned} \Phi_r^s(t_r,t_e) &= \lambda \phi_r^s(t_r,t_e) \\ &= c(t_r - t_e) + c(\delta t_r - \delta t^s) + \lambda(\phi_{r,0} - \phi^{s,0}) + \lambda N_r^s \\ &= \rho_r^s(t_r,t_e) + c(\delta t_r - \delta t^s) + \lambda(\phi_{r,0} - \phi^{s,0}) + \lambda N_r^s \end{aligned} \tag{14}$$

Adding the ionospheric effects, tropospheric effects, earth tide and loading tide effects, multipath and relativistic effects, as well as the others remaining errors into account, the complete model for the Carrier Phase becomes

$$\Phi_r^s(t_r,t_e) = \rho_r^s(t_r,t_e) + c(\delta t_r - \delta t^s) + \lambda(\phi_{r,0} - \phi^{s,0}) + \lambda N_r^s - \delta_{ion} + \delta_{tro} + \delta_{tide} + \delta_{mul} + \delta_{rel} + d\Phi_{br} + \varepsilon \tag{15}$$

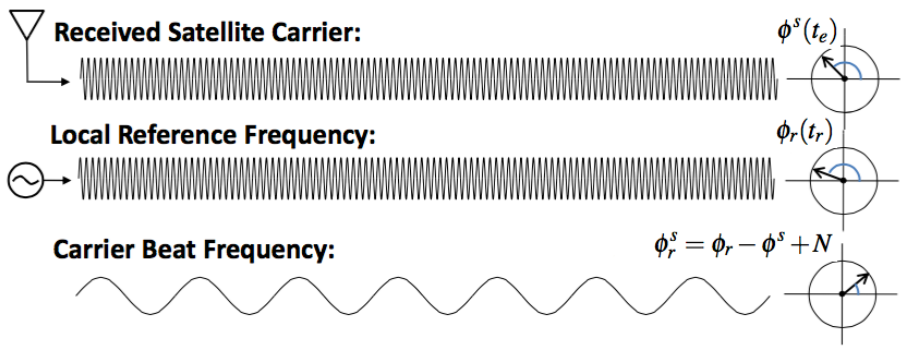


Figure 7: GPS Carrier Phase [Takasu, 2015]

where $d\Phi_{br}$ contains the receiver antenna phase offset, the antenna phase centre variation, station displacement and phase wind-up effect. When using good quality antennas, most of the $d\Phi_{br}$ can be neglected using the Antenna Calibration models available online [**?**].

It is worth to notice that, in the carrier phase model the sign of the ionosphere error is opposite to the signal used in the pseudorange model. This is because, the ionosphere delays the code signal transmission and advances the phase signal transmission [Hoque and Jakowski, 2012].

2.1.4. Real Time Kinematic

In situations where it is required to have an accurate position, it is common to use a differential positioning technique known as Real Time Kinematic (RTK).

This GNSS technique involves using two receivers placed over a small distance, typically less than $20\,km$ allowing to removed common errors arriving at receivers. As the satellites are so far out in space, the relatively low distance between the receivers become insignificant as the signals that reach both receivers will have traveled through virtually the same slice of atmosphere, and so will have virtually the same errors.

Usually, one of the receivers stays at a fixed position and is knows as the base station, the other one can move freely and is called as the rover. Depending on the satellites being observed by both receivers it is possible to create multiple types of linear combinations between the measurements of both receivers.

2.1.4.1. Single Differences

Single Differences (SD) are formed when both receivers are observing the same satellite at the same time, and can be defined as

$$SD^s_{br}(O) = O^s_r - O^s_b \tag{16}$$

where O is the raw bbservable data for the base station b and the rover r, when observing the satellite s. Using the pseudorange model described in equation 10, the SD are expressed as

$$\begin{aligned} P^s_{br} &= P^s_r - P^s_b \\ &= (\rho^s_r - \rho^s_b) + (\delta t_r - \delta t_b)c + (\delta t^s - \delta t^s) + (\delta^s_{r_{ion}} - \delta^s_{b_{ion}}) + (\delta^s_{r_{tro}} - \delta^s_{b_{tro}}) \\ &\quad + (\delta^s_{r_{tide}} - \delta^s_{b_{tide}}) + (\delta^s_{r_{mul}} - \delta^s_{b_{mul}}) + (\delta^s_{r_{rel}} - \delta^s_{b_{rel}}) + (\varepsilon_r - \varepsilon_b) \\ &= \rho^s_{br} + c\delta t_{br} + \delta^s_{br_{ion}} + \delta^s_{br_{tro}} + \delta^s_{br_{tide}} + \delta^s_{br_{mul}} + \delta^s_{br_{rel}} + \varepsilon_{br} \end{aligned} \tag{17}$$

And for the carrier phase model from equation 15 we get

$$\begin{aligned} \Phi^s_{br} &= \Phi^s_r - \Phi^s_b \\ &= \rho^s_{br} + c\delta t_{br} + \lambda\phi_{br,0} + \lambda N^s_{br} - \delta^s_{br_{ion}} + \delta^s_{br_{tro}} + \delta^s_{br_{tide}} + \delta^s_{br_{mul}} + \delta^s_{br_{rel}} + d\Phi^s_{br} + \varepsilon_{br} \end{aligned} \tag{18}$$

The satellite clock error δt^s was removed from both models due to this error being independent from the location of the receivers. In the carrier phase model it was also removed the satellite initial carrier phase $\phi^{s,0}$.

2.1.4.2. Double Differences

Double Differences (DD) can be formed when both receivers are observing two satellites at the same time, and is defined as

$$DD_{br}^{sz} = SD_{br}^{z}(O) - SD_{br}^{s}(O) \tag{19}$$

where s and z represent the two satellites. As the receiver clock offsets δt_{br}^{sz}, and the initial carrier phases $\phi_{br,0}^{sz}$, are constants and independent from the satellites, they cancel each other out. Using short baselines, the values for the ionospheric effects, tropospheric effects, earth tide and loading tide effects, and relativistic effects, are all close to zero making the code pseudorange and carrier phase models expressed as

$$\begin{aligned} P_{br}^{sz} &= \rho_{br}^{sz} + \delta_{br_{mul}}^{sz} + \varepsilon_{br}^{sz} \\ \Phi_{br}^{sz} &= \rho_{br}^{sz} + \lambda N_{br}^{sz} + \delta_{br_{mul}}^{sz} + d\Phi_{br}^{sz} + \varepsilon_{br}^{sz} \end{aligned} \tag{20}$$

2.1.5. Multipath

The multipath is one of the main contributors for the errors in the GNSS positioning, and can't be eliminated using Double Differences . The error is generated when the signal is reflected on nearby obstacles and then picked by the antenna together with the original signal (See Figure 8). These errors occur mainly when the receiver is located near tall buildings, under trees or bridges, or other object that can block the sky view.

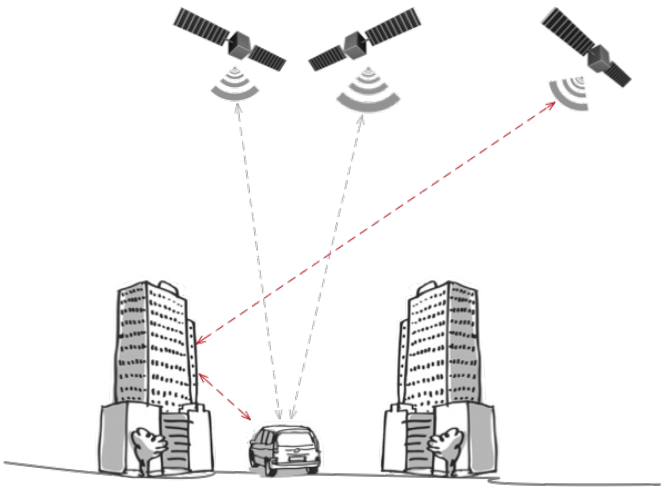


Figure 8: GNSS Multipath Errors

Table 2: GPS User Equivalent Range Error

GPS User Equivalent Range Error			
Error	**Pseudorange**	**Carrier Phase**	**RTK**
Signal arrival	3 m	3 mm	3 mm
Ionosphere	5 m	5m	0
Troposphere	0.5 m	0.5 m	0
Ephemeris	2.5 m	2.5 m	0
Satellite clock	2 m	2 m	0
Multipath	1 m	1 m	1 m
$3\sigma_R$	6.7 m	6 m	1 m

2.1.6. User Equivalent Range Error (UERE)

The term user equivalent range error (UERE) refers to the errors in the various components between the receiver and the satellite, and they are given as $\pm$ errors it is implied they have a zero mean. Table 2 shows estimated values for these errors. However, as they depend a lot from external factors as weather conditions, the receivers localization, and the satellites health, they don't completely illustrate the performance of GPS receivers.

2.2. Inertial Navigation System (INS)

An Inertial Navigation System uses inertial measurements to estimate the pose of a mobile platform. The INS usually contains three orthogonal accelerometers to measure linear accelerations, three gyroscopes to measure angular velocities, and three magnetometers to sense the Earth's magnetic field and use it as a digital compass. This sensor array is called an Inertial Measurement Unit (IMU), and allows us to calculate the current attitude, heading, velocity, and position of the system from a known initial origin. As opposed to the GNSS which depends on the availability of satellites, the INS can work by itself. However, the system is prone to noise and drift errors that accumulate with time, making its estimates unusable after a short period of time [Vectornav, 2015].

2.2.1. Inertial Measurement Units

During this dissertation it was used the InvenSense MPU-9150, which is a low cost 9-Axis MEMS IMU[1] that combines one accelerometer, one gyroscope, and one magnetometer. The idea is to use two of these sensors, placing them in fixed positions, and imposing restrictions in the allowed movements. Since the noise is not correlated, we want to use multiple units of this cheap IMU in order to improve the performance in $\sqrt{(}n)$. It was also used a Xsens MTi-G, which is a device that combines a MEMS accelerometer, gyroscope, compass, barometer, and a GNSS receiver. The

[1] https://www.invensense.com/products/motion-tracking/9-axis/mpu-9150/

Figure 9: InvenSense MPU-9150 and Xsens MTi-G

Xsens fuses all the sensors together using a proprietary Kalman filter algorithm and outputs its position and orientation (See Figure 9).

When choosing these devices it is important to know and understand the differences between the various designations given to them. When the device is referred as an IMU, it consists only in the sensors and it only provides their measurements. An Attitude and Heading Reference System (AHRS) is an IMU with an onboard processing system that provides additional attitude and heading information. When the device is called an INS, the processing system is more advanced and is able to perform navigation.

There are a lot of IMUs on the market that can be used for this propose. The biggest difference between them lies mostly on the type of device, the noise density, bias stability, and of course the price. Table 3 shows some of them, with their specifications and prices. Another interesting option can be one of the IMUs from SBG Systems. They have good quality standalone IMUs, some

Table 3: Alternative IMUs Specifications

IMUs PARAMETERS SPECIFICATION								
Device	**Type**	**Acc. FS** g	**Acc. Noise** $mg/\sqrt{Hz}$	**Gyr. FS** $^\circ/s$	**Gyr. Noise** $^\circ/s/\sqrt{Hz}$	**Mag. FS** G	**Mag. Noise** $mG/\sqrt{Hz}$	**Price**
Invensense MPU-9150	IMU	16	0.400	2000	0.005	12		5 €
Xsens MTi-G	GNSS/INS	5	0.200	300	0.050	0.75	1.500	
Xsens MTi-G 710	GNSS/INS	5	0.080	450	0.010	0.8	0.200	3.980 €
Xsens MTi-10	IMU	5	0.080	450	0.030	0.8	0.200	800 €
Xsens MTi-100	IMU	5	0.080	450	0.010	0.8	0.200	1.500 €
Analog Devices ADIS16488A	IMU	18	0.063	450	0.006	2.5	0.042	1.500 €
ST iNEMO-M1	IMU	16	0.220	2000	0.030	8		70 €
ST LSM6DSM	IMU	16	0.090	2000	0.004	16		
SBG Ellipse-E	INS	8	0.100	450	0.003	8	0.200	
VectorNav VN-100	AHRS	16	0.140	2000	0.003	2.5	0.140	$800
Inertial Labs OS3D	AHRS	2	0.300	1200	0.030	2	0.150	~$999
Inertial Labs OS3DM	AHRS	6	0.300	2000	0.009	2.5	0.015	~$999
Inertial Labs OS3D-FG	AHRS	6	0.200	2000	0.009	8	0.015	~$799
UAV Navigation POLAR	GPS/INS	16	0.520	2000	0.015			
MicroStrain 3DM-GX4-25	AHRS	5	0.100	300	0.005	2.5	0.100	
MicroStrain 3DM-GX3-35	AHRS	5	0.080	300	0.030	2.5	0.100	~$3075
MEMSENSE MS-IMU3050	IMU	10	0.054	480	0.002			
Gladiator Tech. LandMark 10	IMU	2	0.070	300	0.012			~$2495
Gladiator Tech. LandMark 70	IMU	10	0.055	100	0.001			
KVH DSP-3000	FOG-IMU			375	0.067			$4200

models with built in GNSS/INS integration, and also models that can be connected to external GNSS receivers like the Septentrio and do (RTK)GNSS/INS integration automatically. Similar to this, Septentrio also has their AsteRxi models which can be connected to external IMUs from Xsens and SBG, and the GNSS receiver can perform the (RTK)GNSS/INS integration automatically.

2.2.1.1. Accelerometer

MicroElectroMechanical Systems (MEMS) accelerometers are one of the most widely used type of MEMS devices. They are used in cost sensitive, low power, motion and tilt-sensing applications such as mobile devices, gaming systems, disk drive protection, image stabilisation, or in more complex systems such as air bag crash sensors, active suspensions, and anti-lock braking system (ABS).

The most common type of MEMS accelerometers are based on two capacitors with a suspended mass held between them. Without force applied to the system, the two capacitors are symmetrical. But when a small force is applied, the moveable mass shifts closer to one of the plates increasing its capacitance and moves further away from the other plate reducing its capacitance. This difference in capacitance is measured and amplified to produce a voltage proportional to the acceleration [O'Reilly *et al.*, 2009].

2.2.1.2. Gyroscope

MEMS gyroscopes work by measuring the Coriolis effect F_c that affects a reference frame with a mass m, rotates with an angular velocity ω, and a linear velocity v, as expressed in

$$F_c = -2m(\omega \times \upsilon) \tag{21}$$

When the device is undergoing a rotational motion, the generated force causes a differential out-of-plane deflection on the proof mass elements which is detected by capacitors plates, measured and amplified to produce a voltage proportional to the angular velocity.

2.2.1.3. Magnetometer

Magnetometers are important sensors when estimating the attitude of a robot in navigation systems, as they can be a low cost and accurate solution to estimate the heading of a vehicle. They work by comparing the magnetic field nearby with a vector representation of the Earth coordinates.

However, the magnetic field reading are sometimes corrupted by fabrication issues and the presence of ferromagnetic elements in the surrounding of the sensor. Therefore, proper calibration is really important to achieve accurate results.

2.2.2. MEMS Accelerometer/Gyroscope Error Characteristics

This section explains the most common errors in MEMS accelerometers and gyroscopes. As these errors are going to be integrated it is very important to reduce them [Woodman, 2007].

Constant Bias

The bias value is a constant offset that is present in the output signals and shifts the output from its real value.

For the **Accelerometer**, it's measured in m/s^2, and when double integrated causes an error in the position which grows quadratically with time.

For the **Gyroscope**, it's measured in $^\circ/h$ and when integrated creates an angular error which grows linearly with time.

$$\begin{aligned} Acc. \quad s(t) &= \varepsilon \cdot \frac{t^2}{2} \\ Gyr. \quad \theta(t) &= \varepsilon \cdot t \end{aligned} \tag{22}$$

Equation 22 can be used to estimate the error over time for both devices, where ε is the bias error and t is the integration time. The bias error can be estimated by long term averaging the sensor output when the device isn't experiencing the influence of any external movement, and then simply subtract its value from output.

Thermo-Mechanical White Noise - Velocity/Angle Random Walk

The output of a MEMS device is unsettled by a thermo-mechanical white noise sequence which oscillates at a much higher frequency than the sampling frequency of the sensor.

For the **Accelerometer**, this results in a velocity random walk whose standard deviation grows proportionally to $\sqrt{t}$, and is usually specified in $m/s/\sqrt{h}$.

For the **Gyroscope**, this results in a angle random walk whose standard deviation grows proportionally to $\sqrt{t}$, and is usually specified in $^\circ/\sqrt{h}$.

$$\begin{aligned} Acc. \quad \sigma_s(t) &\approx \sigma \cdot t^{3/2} \cdot \sqrt{\frac{\delta t}{3}} \\ Gyr. \quad \sigma_\theta(t) &= \sigma \cdot \sqrt{\delta t \cdot t} \end{aligned} \tag{23}$$

Equation 23 can be used to estimate the error over time for both devices. When double integrated to obtain the position, this creates a second order random walk with zero mean and a standard deviation $\sigma_s(t)$ which grows proportionally to $t^{3/2}$.

Flicker Noise - Bias Stability

The bias of a MEMS device drifts over time, and is caused by a flickering noise present in all the electronics. This noise has a $1/f$ spectrum which tends to be only observable at lower frequencies in electronic components. At higher frequencies it is usually obscured by the presence of white noise.

The bias stability is usually specified as a 1σ value and describes how the bias of a device change over time in fixed conditions. The flicker noise creates a random walk in the integrated signal signal, so its standard deviation grows proportionally to $\sqrt{t}$, being measured in $^\circ/h$, as seen in equation 24.

$$Acc. \quad Gyr. \qquad BRW(^\circ/\sqrt{h}) = \frac{BS(^\circ/h)}{\sqrt{h(t)}} \tag{24}$$

This flickering creates a second order random walk in velocity whose uncertainty grows proportionally to $t^{3/2}$, and a third order random walk in position which grows proportionally to $t^{5/2}$.

Temperature Effects

Temperature oscillations in the environment induces movements in the bias. There isn't a specific relationship between the bias and the temperature, as it changes from device to device and is often highly nonlinear for MEMS devices.

Calibration Errors

Calibration errors are expressed as scale factors errors, axis misalignments, and non-linearity outputs which produce bias errors that are only visible whilst the device is undergoing movements.

2.2.3. MEMS Magnetometer Error Characteristics

There are two types of errors that can be found in the magnetometer measurements. The first one represents instrumentation errors, and includes scale factors, sensor offsets, and the non-orthogonality of the sensor axes. These errors are present in the sensors due to fabrication imperfections. The second type is specific to magnetometers and consists of the magnetic deviation produced by the presence of ferromagnetic materials.

Scale Factors, are errors in the sensor measurements that corresponds to different constants of proportionality between the input and the output of the sensor. It is represented by

$$\mathbf{S} = diag\begin{pmatrix} s_x & s_y & s_z \end{pmatrix} \tag{25}$$

Bias, are simple offsets present in the sensor output and can be modelled as a scalar per axis

$$\mathbf{b}_{so} = \begin{bmatrix} b_{so_x} & b_{so_y} & b_{so_z} \end{bmatrix}^T \tag{26}$$

Non-Orthogonality errors are caused by limitations during the construction of the sensor which leads to the axes not being completely orthogonal to each other. It is represented by

$$\mathbf{K}_{nor} = \begin{bmatrix} cos(\alpha) & 0 & sin(\alpha) \\ sin(\beta)cos(\gamma) & cos(\beta)cos(\gamma) & sin(\gamma) \\ 0 & 0 & 1 \end{bmatrix} \tag{27}$$

where α, β, and γ are the misalignment in the X, Y, Z axes respectively.

Hard Irons distortions are produced by materials with a constant additive field to the earth's magnetic field, and therefore generate a constant additive value to the output of the magnetometer axes. They are represented by a bias

$$\mathbf{b}_{hi} = \begin{bmatrix} b_{hi_x} & b_{hi_y} & b_{hi_z} \end{bmatrix}^T \tag{28}$$

Soft Irons distortions are produced by ferromagnetic compounds that distorts the magnetic field, causing the intensity and the direction of the sensed field to change. The soft iron effect can be modelled by a 3x3 matrix

$$\mathbf{A}_{si} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{29}$$

Complete Error Model

Taking into account all the errors listed above, the relationship between the true geomagnetic field and the three-axis magnetometer output can be modelled as

$$\widehat{\mathbf{h}} = \mathbf{S} \cdot \mathbf{K}_{nor} \left(\mathbf{A}_{si} \cdot \mathbf{h} + \mathbf{b}_{hi} \right) + \mathbf{b}_{so} + \varepsilon \tag{30}$$

where $\mathbf{h}$ is the true geomagnetic field, $\widehat{\mathbf{h}}$ are the reading from the three-axis magnetometer output, and ε is a Gaussian noise $\sim \mathbf{N}\left(0, \sigma_{\varepsilon}^2\right)$. Introducing two new variables to equation 30, the magnetometer error model becomes

$$\widehat{\mathbf{h}} = A\mathbf{h} + b + \varepsilon \tag{31}$$

where values for the matrix $\mathbf{A}$ and the bias $\mathbf{b}$ can be found using

$$\begin{aligned} \mathbf{A} &= \mathbf{S} \cdot \mathbf{K}_{nor} \cdot \mathbf{A}_{si} \\ \mathbf{b} &= \mathbf{S} \cdot \mathbf{K}_{nor} \cdot \mathbf{b}_{hi} + \mathbf{b}_{so} \end{aligned} \tag{32}$$

the matrix $\mathbf{A}$ represents the scale factors, misalignments, and soft irons disturbances. The vector $\mathbf{b}$ represents the combined bias.

There are a wide variety of algorithms that can be used to estimate the values for these two

variables. In this case it was used the algorithm described by [Renaudin *et al.*, 2010], which uses an adaptive least mean square estimator to solve the ellipsoid fitting problem.

3. GNSS/INS Data Fusion

GNSS Receivers using Real Time Kinematics are able to accurately estimate the position in areas with good sky view. However as explained before, the satellite signals can be partially blocked, and the RTK solution will only work as long as the communication between the Base Station and the Rover is stable. The INS can provide us information about the orientation, linear acceleration, and angular velocity with higher update rates and availability.

A combination of both systems have the possibility of providing a better short and long term accuracy with higher update rates, and allows us to increase the reliability of the system if the RTK stops working or if there is a satellite occlusion. There are three types of GNSS/INS integrations: the Loosely coupled, the Tightly coupled and the Ultra-Tight coupled [Titterton and Weston, 2005].

Loosely Coupled Integration

In the Loosely coupled integration, the GNSS receiver and the INS sensors are treated as standalone devices that provide two separate measurements (See Figure 10). This allows the GNSS receiver and the inertial sensor to be replaced by other ones and only have to make minor changes in the data fusion algorithm. The implementation is simple and stable, however there is a need to track a minimum of 4 satellites to get an accurate positioning.

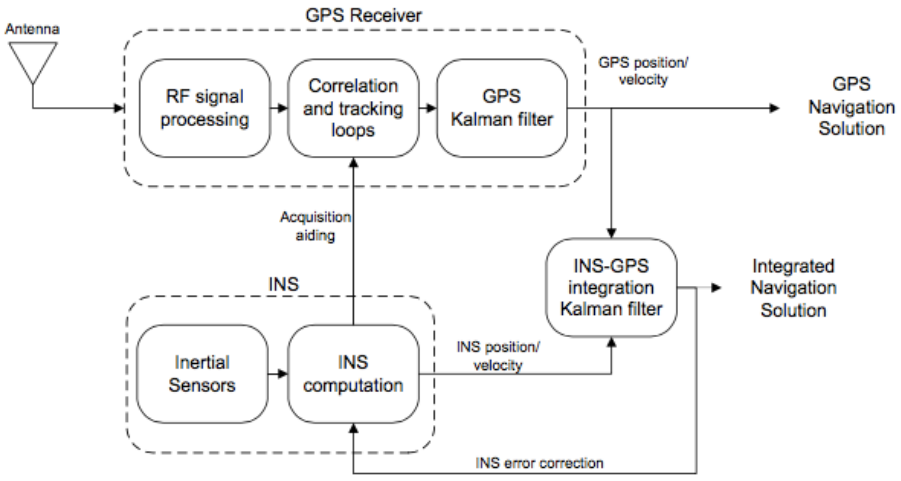


Figure 10: Loosely Coupled Integration [Titterton and Weston, 2005]

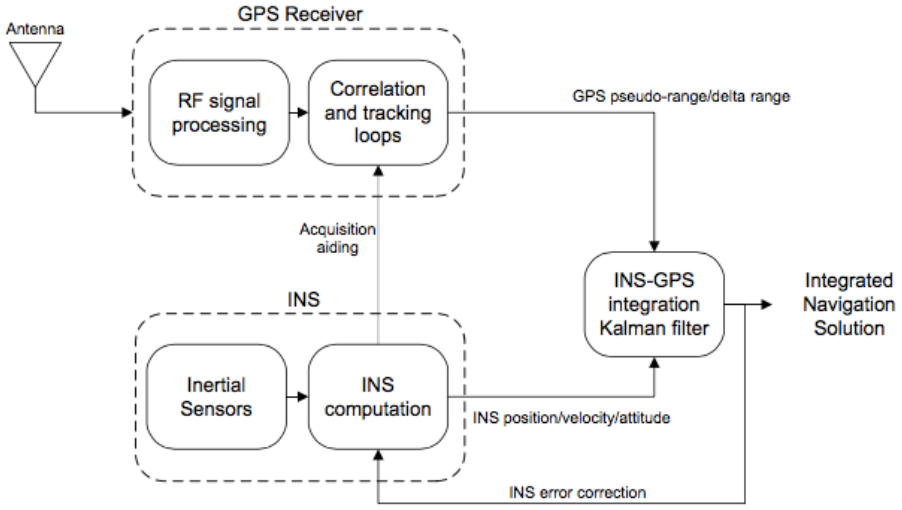


Figure 11: Tightly Coupled Integration [Titterton and Weston, 2005]

The problem with this type of integration is that each device has its own Kalman filter running, plus one more for the data fusion. When we are formulating the Kalman filter, it's implicitly assumed that the measurement errors aren't correlated, however in the cascaded Kalman filter configuration used by the loosely coupled integration, such assumption is not necessarily true. In some situations, the integration algorithm can sample the GNSS information faster than the provided by the receiver or receive signals arising through multipath effects, which can cause the Kalman filter measurement errors to become time correlated.

Tightly Coupled Integration

In the Tightly coupled integration, the measurements from the GNSS receiver and the INS are integrated at a deeper level and using a single Kalman filter (See Figure 11). This solution overcomes the problem of the loosely coupled where the measurement errors can become correlated.

In this system the measurements that arise from the inertial sensor are integrated in the pseudorange and the carrier phase signals received from the satellites, allowing the system to work with less than 4 satellites and thus keeping a good accuracy in urban areas. The main disadvantage with this type of integration is that it is more complex to implement and more hardware dependent.

An example of such type of integration can be found on the NovAtel Synchronized Position, Attitude and Navigation (SPAN) receivers.

Ultra-Tight Coupled Integration

In the Ultra-Tight coupled integration, the Kalman filter integration is done in the tracking loops of the satellite signals inside the receiver. This type of system provides improvements in the Signal to Noise Ratio, has less multipath influence, and a faster re-acquisition of interrupted signals. However,

it is a highly complex system to implement and therefore there aren't a lot of manufacturers using it.

3.1. Experimental Applications in the Literature

There are a lot of vehicles using these type of sensor fusion algorithms, so in this section some of the most interesting ones are presented.

From Gothenburg, Sweden [Magnusson and Odenman, 2012] implemented a sensor fusion algorithm for a GPS receiver, an IMU, and wheel odometry. Two years later, [Elisson and Gassler, 2014] also implemented a sensor fusion for GNSS/INS integration. Both projects where tested in cars in urban and rural areas, with some interesting results.

[Kreinar, 2013] also implemented a GPS, INS, and wheel odometry sensor fusion algorithm for an autonomous lawn mower robot with wheel slip detection. He also compares the robots behaviour with different EKF system models.

In Linköping, Sweden, [Hjelmare and Rangsjö, 2012] implemented a localization and mapping robot using the TurtleBot platform with a GPS receiver, a gyroscope, and a Microsoft Kinect.

From Instituto Superior Técnico in Portugal, [Lopes, 2011] implemented a low cost fusion algorithm modelled for an UAV, using a normal GPS receiver and an IMU with only a accelerometer and a gyroscope. The final test flight was done using a Xsens MTi-G to collect the data, and the fusing algorithm was able to achieve good results.

In his PhD Dissertation [Hol, 2011] talks about calibration methods, INS/Vision integration, INS/Ultra Wide-Band (UWB) integration, and GPS/INS integration which was later tested in a jet aircraft. It is a very complete thesis that shows other types of localization methods and some Xsens prototypes being developed.

[Carcanague, 2013] is a very interesting and complete PhD Dissertation on implementing low cost GNSS receivers in precise positioning algorithms to be used mostly in urban environments. He shows that low cost GNSS receivers can have large errors in their measurements, talks about the inter-channel biases that exists in the GLONASS satellite signals because of the signals being broadcasted at different frequencies, and has a lot more information about the GNSS.

Finally probably the most complete project, [Rodrigues *et al.*, 2013] implemented a robotic system composed by an Autonomous Surface Vehicle (ASV) piggybacking a multi-rotor Unmanned Aerial Vehicle (UAV). The ASV provides long-range transportation in all-weather conditions, and the UAV assures an augmented perception of the environment. The coordinated aerial, underwater, and surface level perception allows the navigation system to assess the near field to the far field, which is key for safe navigation and environmental monitoring data gathering. The localization of the ASV is done with RTK-GPS/INS integration and the UAV uses GPS/INS integration. The

navigation is based on a laser scanner, an underwater sonar, and multiple cameras in the ASV and the UAV.

3.2. Extended Kalman Filter

The Extended Kalman Filter is a modified version of the common Kalman Filter that allows to estimate a non-linear system such as the one this dissertation addresses. The filter consists of two distinct steps: In the first one, it is made an estimation of the of the system states based on the noise model, and the estimated covariance updated. In the second step, the measurements are updated and the estimation corrected. There are a lot of books and articles discussing the EKF and data fusion such as [Welch and Bishop, 2006; Ge and Lewis, 2006; Bar-Shalom *et al.*, 2004; Grewal and Andrews, 2015; Kelly, 1994; Magnusson and Odenman, 2012; Terejanu, 2008]. During this section it's mainly followed two books [Kelly, 1994; Grewal and Andrews, 2015].

The system can be modelled as a state space model in discrete time, with the following generic form,

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{f}_k(\mathbf{x}_k) + \mathbf{w}_k \\ \mathbf{z}_k &= \mathbf{h}_k(\mathbf{x}_k) + \mathbf{v}_k \end{aligned} \tag{33}$$

where $\mathbf{x}$ represents the system model, and $\mathbf{z}$ the measurements model. The process noise $\mathbf{w}$, and the measurement noise of the system $\mathbf{v}$, are assumed to be white, additive, zero mean, and uncorrelated with each other, as shown in equation 34.

$$\begin{aligned} E[\mathbf{w}_k] &= 0 \\ E[\mathbf{v}_k] &= 0 \\ \\ E\left[\mathbf{w}_k \mathbf{w}_i^T\right] &= \mathbf{Q}_k \\ E\left[\mathbf{v}_k \mathbf{v}_i^T\right] &= \mathbf{R}_k \\ \\ E\left[\mathbf{w}_k \mathbf{v}_i^T\right] &= 0, \quad \forall(k,i) \end{aligned} \tag{34}$$

In equation 33, $\mathbf{f}$ represents the non-linear function that defines the systems dynamics and relates the state from $\mathbf{t}_k$ to $\mathbf{t}_{k+1}$. The measurement function $\mathbf{h}$, is also a non-linear function and defines how the sensors measurements are related to the states.

Assuming the non-linearities in the dynamics and that the measurement model are smooth, the functions $\mathbf{f}$ and $\mathbf{h}$ can be expanded in Taylor Series, and approximating this way the prediction and the next estimation of $\mathbf{x}_k$. Neglecting the higher order terms in the Taylor Series Expansion, the

first order approximation coefficients are given by

$$\mathbf{F}_k = \left.\frac{\partial f}{\partial x}\right|_{\widehat{x}_k}, \qquad \mathbf{H}_k = \left.\frac{\partial h}{\partial x}\right|_{\widehat{x}_k} \tag{35}$$

that represent the system and the measurement Jacobians. The discrete time Extended Kalman Filter equations for the system model follows

---

Extended Kalman Filter - Prediction Step

$$\widehat{\mathbf{x}}_{k+1} = \Phi_k \widehat{\mathbf{x}}_k \tag{36}$$

$$\widehat{\mathbf{P}}_{k+1} = \mathbf{F}_k \mathbf{P}_k \mathbf{F}_k^T + \Gamma_k \mathbf{Q}_k \Gamma_k^T \tag{37}$$

The predicted estimate of the system state $\widehat{\mathbf{x}}$, and the associated covariance matrix $\widehat{\mathbf{P}}$, represent the best knowledge of the system state at $\mathbf{t}_{k+1}$ before making any observations. The transition matrix Φ_k relates the state vector from $\widehat{\mathbf{x}}_k$ to $\widehat{\mathbf{x}}_{k+1}$, and the process noise distribution matrix Γ_k transforms the vector $\mathbf{w}_k$ into the coordinates of $\widehat{\mathbf{x}}_k$.

The EKF correction step can be found using the following equations

---

Extended Kalman Filter - Correction Step

$$\mathbf{K}_k = \widehat{\mathbf{P}}_k \mathbf{H}_k^T \left(\mathbf{H}_k \widehat{\mathbf{P}}_k \mathbf{H}_k^T + \mathbf{R}_k \right)^{-1} \tag{38}$$

$$\mathbf{x}_k = \widehat{\mathbf{x}}_k + \mathbf{K}_k \left(\mathbf{z}_k - \mathbf{h}(\widehat{\mathbf{x}}_k) \right) \tag{39}$$

$$\mathbf{P}_k = \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k \right) \hat{\mathbf{P}}_k \tag{40}$$

In the previous equations, $\mathbf{K}_k$ is the Kalman Gain, a matrix that multiplies the error from the difference between the real measurement obtained at $\mathbf{t}_k$ and the predicted value. $\mathbf{x}_k$ represents the updated state estimate, and $\mathbf{P}_k$ the updated covariance matrix.

The equations aren't run all at once. The prediction step can be run at a high frequency and the correction step should only run when new measurements are available. There are several similar forms of the Kalman filter equations with different forms of systems models. However, this one is commonly used in the navigation industry since it avoids numerical problems in the Kalman gain matrix, and at the same time it also decreases the number of matrix inversions needed to the number of measurements only.

3.3. System Dynamics Model

When implementing a Kalman Filter it is very import to model correctly the system where the filter is going to be used, as a good dynamics model allows to reduce the navigation errors by implementing restrictions to the possible movements the physical system can do.

As the objective of this dissertation is to implement an EKF to be used in a wide variety of robots and other movable objects, the system model implemented was a standard 3D Model without kinematic restrictions, and follows the Loosely Coupled Integration explained earlier in this chapter.

The state vector of the filter is composed by the 3D components of the Pose, Orientation, Velocity, Angular Velocity, and Linear Acceleration, as described by equation 41.

$$\mathbf{x}_k = \left[x, y, z,\ \ \phi, \theta, \psi,\ \ \dot{x}, \dot{y}, \dot{z},\ \ \dot{\phi}, \dot{\theta}, \dot{\psi},\ \ \ddot{x}, \ddot{y}, \ddot{z}\right]^T \tag{41}$$

The transfer function can be found using the multiplication of the three rotation matrices that define the rotation of a 3D body using the Roll Pitch Yaw Convention.

$$\mathbf{R}_z(\psi) = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{R}_y(\theta) = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix}, \mathbf{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix} \tag{42}$$

Projecting the state forward using $\hat{\mathbf{x}} = \mathbf{f}(\hat{x}, t)$, the predicted state is found using the transfer function

$$\widehat{\mathbf{x}}_{k+1} = \Phi_k \widehat{\mathbf{x}}_k = \begin{bmatrix} I & 0 & \Phi_{13} & 0 & \Phi_{15} \\ 0 & I & 0 & \Phi_{24} & 0 \\ 0 & 0 & I & 0 & \Phi_{35} \\ 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & I \end{bmatrix} \cdot \widehat{\mathbf{x}}_k \tag{43}$$

where the sub-matrices are all 3x3 and

$$\Phi_{13} = \begin{bmatrix} \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \cdot \Delta t \tag{44}$$

$$\Phi_{15} = \frac{\Phi_{13}}{2} \cdot \Delta t, \qquad \Phi_{24} = \Phi_{13}, \qquad \Phi_{35} = \Delta t \cdot I \tag{45}$$

The System Jacobian can be obtained using equation 35, and is equivalent to the following matrix

$$\mathbf{F}_k = \begin{bmatrix} I & \mathbf{F_{12}} & \Phi_{13} & 0 & \Phi_{15} \\ 0 & \mathbf{F_{22}} & 0 & \Phi_{24} & 0 \\ 0 & 0 & I & 0 & \Phi_{35} \\ 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & I \end{bmatrix} \tag{46}$$

where $\mathbf{F_{12}}$ and $\mathbf{F_{22}}$ can be found in Appendix D and were calculated using MATLAB. With the Jacobian calculated, the covariance matrix $\widehat{\mathbf{P}}$ can then be estimated using equation 37.

The first correction step is initialised by feeding a starting "a priori estimate" $\widehat{\mathbf{x}}_k(0)$ and its covariance $\widehat{\mathbf{P}}_k(0)$ to the filter when the first measurements from the sensors arrive. The initial value of $\widehat{\mathbf{x}}_k(0)$ represents the starting position and orientation, and the value of $\widehat{\mathbf{P}}_k(0)$ was hand tuned using small covariance values. The value for the process noise covariance $\mathbf{Q}_k$ was also chosen using small covariance values, and for each cycle of the system model, the new values of the state transition matrix Φ_k and $\mathbf{Q}_k$ are estimated. The measurement matrix $\mathbf{H}_k$, and its covariance $\mathbf{R}_k$ are updated when new sensor measurements arrive to the filter.

If it is desired to work in a 2D environment, it is sufficient to zero the Z components of the position, velocity, and acceleration, roll, pitch, roll velocity, and pitch velocity.

3.4. Measurement Model

The measurement model implemented in the Kalman filter is used to define how the sensors measurements relate to the states. During this section it's important to know the different types of frames that exist, understand how they relate to each other, and how to make the conversions between them. A brief information about this can be seen in Appendix C or more detailed in [Crassidis, 2006].

3.4.1. GNSS Data

All the data processing related to the GNSS ceceivers is done outside of the data fusion algorithm using a separate ROS node and published in a ROS NavSatFix message. The EKF subscribes to these messages and starts by converting them from LLA to UTM. The measurement model for the GNSS receiver is then expressed as,

$$\begin{aligned}\mathbf{z}_{gps,k} &= \mathbf{H}_{gps,k}\mathbf{x}_k + \mathbf{v}_{gps,k} = \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{x}_k + \mathbf{v}_{gps,k} = \\ &= \begin{bmatrix} x_k \\ y_k \\ z_k \end{bmatrix} + \mathbf{v}_{gps,k}\end{aligned} \tag{47}$$

with the $\mathbf{v}_{gps,k}$ noise following a zero mean, white, gaussian distribution with a known covariance $\mathbf{R}_{gps,k}$, which is is filled directly from the values provided by the GNSS receiver or the RTKLIB covariance estimation.

3.4.2. INS Data

The measurement model of the INS Data is provided by the Inertial Measurement Units and pre-processed previously by the Stellaris Launchpad, a ROS node and then published in a ROS IMU message. Before feeding the measurements to the filter it's important to make sure all inertial sensors are reporting their data in the same body frames, and apply the correct rotations to align them properly if needed. As the IMUs are composed internally by three sensors, the measurement model is more complex and allows us to obtain more information.

$$
\begin{aligned}
\mathbf{z}_{imu,k} &= \mathbf{H}_{imu,k}\mathbf{x}_k + \mathbf{v}_{imu,k} \\
&= \begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix} \mathbf{x}_k + \mathbf{v}_{imu,k} \\
&= \begin{bmatrix} \phi_k & \theta_k & \psi_k & \dot{\phi}_k & \dot{\theta}_k & \dot{\psi}_k & \ddot{\phi}_k & \ddot{\theta}_k & \ddot{\psi}_k \end{bmatrix}^T + \mathbf{v}_{imu,k}
\end{aligned}
\tag{48}
$$

where the $\mathbf{v}_{imu,k}$ noise follows a zero mean, white, gaussian distribution with a known covariance $\mathbf{R}_{imu,k}$, broadcasted in the ROS IMU Message and hand tuned from similar devices.

3.4.3. Complete Model

The complete measurement model is composed by concatenating the measurement models above in a single matrix,

$$
\begin{aligned}
\mathbf{z}_k &= \begin{bmatrix} \mathbf{z}_{gps,k} \\ \mathbf{z}_{imu,k} \end{bmatrix} \mathbf{x}_k + \begin{bmatrix} \mathbf{v}_{gps,k} \\ \mathbf{v}_{imu,k} \end{bmatrix} \\
&= \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k
\end{aligned}
\tag{49}
$$

where the resulting matrix H has a dimensions of 15x15.

The measurement noise $\mathbf{v}_k$ has a corresponding covariance matrix $\mathbf{R}_k$,

$$
\mathbf{R}_k = \begin{bmatrix} \mathbf{R}_{gps,k} & 0 \\ 0 & \mathbf{R}_{imu,k} \end{bmatrix} \tag{50}
$$

where it is assumed that the measurement noises are independent from each other.

As the measurements and their errors are uncorrelated, these matrices can be subdivided in smaller ones and processed one at a time giving the same result as processing everything together. This is particularly important in real time systems with multiple asynchronous sensors, as it allows to only change the sub-matrices related to that sensor, and therefore reducing the computational power needed.

3.5. Other Platform Models

3.5.1. Differential Drive Steering

It is very common to find wheeled mobile robots using two wheels mounted on a common axis as a drive mechanism. These robots are known as differential drive, since each wheel can be driven independently in order to steer the platform (Figure 12).

Different wheels speeds causes the robot to follow circular trajectories with an Instantaneous Centre of Curvature (ICC). The differential drive imposes non-holonomic restrictions on the position, as the robot cannot move laterally.

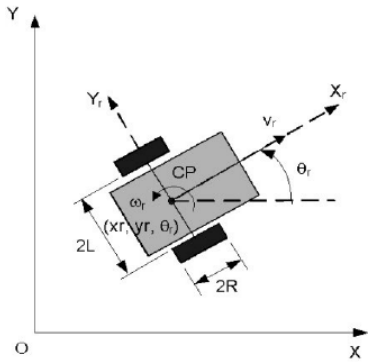


Figure 12: Differential Drive Steering [Kreinar, 2013]

A simple state vector for a differential drive can be expressed by

$$\mathbf{x}_k = [x, y, \theta, \upsilon, \omega]^T \tag{51}$$

The system model $\mathbf{f}(\mathbf{x})$, and the system Jacobian Matrix $\mathbf{F}_k$ can be represented by

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} x_{k-1} + \upsilon_{k-1} cos(\theta_k + \frac{\omega_{k-1}\Delta t}{2})\Delta t \\ y_{k-1} + \upsilon_{k-1} sin(\theta_k + \frac{\omega_{k-1}\Delta t}{2})\Delta t \\ \theta_{k-1} + \omega\Delta t \\ \upsilon_{k-1} \\ \omega_{k-1} \end{bmatrix} \tag{52}$$

$$\mathbf{F}_k = \begin{bmatrix} 1 & 0 & -\upsilon_k \cdot sin(\theta_k + \frac{\omega_k\Delta t}{2})\Delta t & cos(\theta_k + \frac{\omega_k\Delta t}{2})\Delta t & -\frac{\upsilon_k}{2} sin(\theta_k + \frac{\omega_k\Delta t}{2})\Delta t^2 \\ 0 & 1 & \upsilon_k \cdot cos(\theta_k + \frac{\omega_k\Delta t}{2})\Delta t & sin(\theta_k + \frac{\omega_k\Delta t}{2})\Delta t & \frac{\upsilon_k}{2} cos(\theta_k + \frac{\omega_k\Delta t}{2})\Delta t^2 \\ 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{53}$$

3.5.2. Skid Steering

Skid steering robots are also widely used because of their simple mechanism and robustness. However, their complex wheel-ground interactions and kinematic constraints, make it more difficult to understand and implement the kinematic model of the platform.

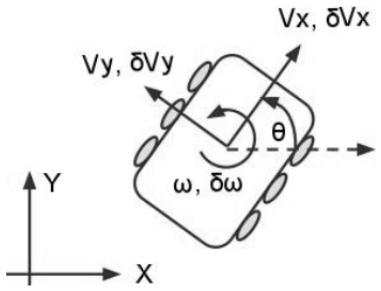


Figure 13: Skid Steering [Rogers-Marcovitz, 2010]

These vehicles typically use four wheel mechanically locked in synchronisation on each side which can be driven independently (Figure 13). As the wheels have no steering mechanism and are hold in a fixed straight alignment, by changing the speed of the left and right wheel pairs the robot turns by skidding its fixed-orientation wheels across the ground.

A simple state vector and input vector for the skid steering can be expressed by

$$\mathbf{x}_k = [x, y, \theta]^T \qquad \mathbf{u}_k = [V, \omega]^T \tag{54}$$

With the following kinematic differential equation for the vehicle's 2D position and heading,

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \left\{ \begin{bmatrix} V_x \\ V_y \\ \omega \end{bmatrix} + \begin{bmatrix} \delta V_x \\ \delta V_y \\ \delta\omega \end{bmatrix} \right\} \tag{55}$$

The input error is modelled as $\delta u = \left[\delta V_x, \delta V_y, \delta\omega\right]$. The Jacobian with respect to the slip velocities Γ, and the states $\mathbf{F}$ are represented by

$$\Gamma = \frac{\partial \dot{x}}{\partial \delta u} = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{56}$$

$$\mathbf{F} = \frac{\partial \dot{x}}{\partial x} = \begin{bmatrix} 0 & 0 & -(V+\delta V_x)sin(\theta) - \delta V_y cos(\theta) \\ 0 & 0 & (V+\delta V_x)cos(\theta) - \delta V_y sin(\theta) \\ 0 & 0 & 0 \end{bmatrix} \tag{57}$$

3.5.3. Ackerman Steering

The Ackerman Steering is one of the most common steering models used on two or four wheel vehicles, as road cars and bicycles. This steering principle defines a geometry for the correct turning angle of the steering wheels when cornering that minimises the tire slippage (Figure 14).

The Ackerman Steering also has non-holonomic restrictions, as the robot can only turn its front wheels. It can not move sidewise, nor rotate having a null linear velocity.

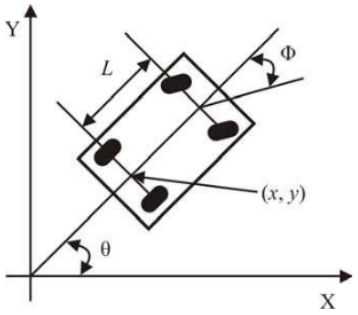


Figure 14: Ackerman Steering [Newman, 2003]

A simple state vector and input vector for the Ackerman steering can be expressed by

$$\mathbf{x}_k = [x, y, \theta]^T \qquad \mathbf{u}_k = [V, \phi]^T \tag{58}$$

With the following kinematic differential equation for the vehicle's 2D position and heading,

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_k + V_k \cdot cos(\theta_k)\Delta t \\ y_k + V_k \cdot sin(\theta_k)\Delta t \\ \theta_k + \frac{V_k \cdot tan(\theta_k)\Delta t}{L} \end{bmatrix} \tag{59}$$

The Jacobians with respect to the states, and the control noise are represented by

$$\frac{\partial f}{\partial x} = \begin{bmatrix} 1 & 0 & -V_k \cdot sin(\theta_k)\Delta t \\ 0 & 1 & V_k \cdot cos(\theta_k)\Delta t \\ 0 & 0 & 1 \end{bmatrix} \tag{60}$$

$$\frac{\partial f}{\partial u} = \begin{bmatrix} cos(\theta_k)\Delta t & 0 \\ sin(\theta_k)\Delta t & 0 \\ \frac{tan(\theta_k)}{L} & \frac{V_k \cdot sec^2(\theta_k)\Delta t}{L} \end{bmatrix} \tag{61}$$

4. Experimental Results

This chapter contains the practical part that was done during this dissertation, and explains how the INS and RTK was implemented, and has some tests of the GNSS/INS integration.

4.1. INS Implementation

As mentioned in section 2.2 during this dissertation was used two Invensense MPU-9150 IMUs and one Xsens MTi-G. Both devices differ quite a bit and their main specifications can be seen in Table 4. The Xsens provides drivers for their devices in their website, so interacting with the Xsens MTI-G quite easy and all it took was a little fiddling in the software.

However, to interact with the Invensense MPU-9150 and implement the INS, it was needed to develop the software for the device using one Texas Instruments Stellaris LaunchPad microcontroller with both inertial sensors connected to his I^2C ports. The microcontroller initialises the accelerometer and gyroscope, and sends them calibrations values. As the magnetometer doesn't have hardware registers for the calibration, it needs to be calibrated in software. It is very important to pay attention for the axes orientation when dealing with these type sensors, as they can have their axes swapped. The MPU-9150 has different axes orientations between the accelerometer/gyroscope and the magnetometer, as can be seen in Figure 15. Therefore, before using their measurements it is necessary to rotate the values properly.

The datasheet of the Invensense MPU-9150 states that in order to sample the magnetometer it is needed to follow a special sequence of commands with specific times between them, making the maximum update rate $8\,Hz$. Changing a little the command sequence and reducing the waiting times, it was possible to configure the microcontroller to run a timer interruption where the all

Table 4: Alternative IMUs Specifications

IMUs PARAMETERS SPECIFICATION													
Device	**Gyroscope**					**Accelerometer**					**Magnetometer**		
	FS	Lin.	Noise	Freq.	ADC	FS	Lin.	Noise	Freq.	ADC	FS	Freq.	ADC
Xsens MTi-G	300	0.1	0.05	512	16	5	0.2	0.2	512	16	0.75	512	16
Invensense MPU-9150	2000	0.2	0.005	8000	16	16	0.5	0.4	1000	16	12	8	13
	$^\circ/s$	$\%FS$	$^\circ/s/\sqrt{Hz}$	Hz	$bits$	g	$\%FS$	$mg/s/\sqrt{Hz}$	Hz	$bits$	G	Hz	$bits$

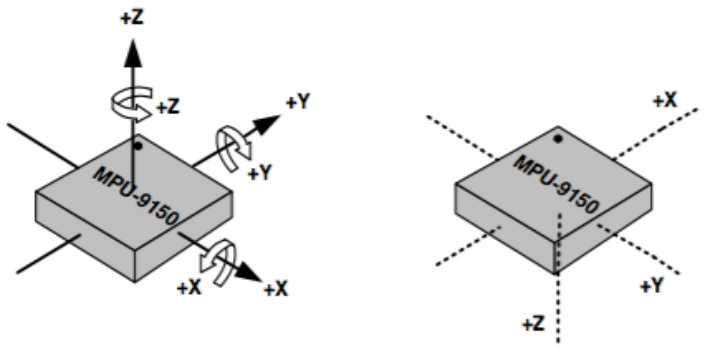


Figure 15: MPU-9150 Accelerometer/Gyroscope and Magnetometer Axes Orientation

the sensors are sampled at $100\,Hz$ without measuring errors. As the magnetometer values are still not calibrated, after being sampled they are corrected by the rotation matrix and the bias vector previously found using the model in section 2.2.3.

To calculate the IMU orientation, the calibrated values from the accelerometer, gyroscope, and magnetometer are fed to a Madgwick filter. This is a highly popular filter when using inertial sensors in microcontrollers because it doesn't require a lot of computational power and can obtain similar result as the Kalman filter or the complementary filter [Madgwick, 2010]. The filter receives the inertial 9 Degrees of Freedom (DOF) values, fuses them together and returns a normalised quaternion, which is a four-dimensional complex number used to represent the orientation of a ridged body or a coordinate frame in a three-dimensional space. A detailed explanation and comparison of the Madgwick filter, the Complementary filter and Kalman filter can be seen at [OlliW, 2013].

It is important to run the filter at a high frequency in order to the estimated orientation to be accurate when the sensor is undergoing movements. The Stellaris LaunchPad is able to run two instances of the filter, one for each inertial sensor, at $1600\,MHz$ and takes 10 to 15 seconds to initially estimate an accurate orientation. With everything ready, 3 messages per IMU (Equation 62) are created and sent to the serial port with a 460800 baudrate. These messages contain the quaternion, accelerometer, and gyroscope information and are sent at $50\,Hz$.

$$\begin{aligned} &1Q, 0.688973, -0.000998, -0.000075, -0.724955 \quad (44bytes) \\ &1A, -0.010777, -0.010777, 9.932511 \quad (34bytes) \\ &1G, -0.129699, -0.114440, 0.076293 \quad (34bytes) \end{aligned} \tag{62}$$

The transmission times for the messages can be calculated by

$$\begin{aligned} Quat. &= \frac{44 * 10\,bits/byte}{460800(bits/s)} = 0.955\,ms \\ Acc. &= \frac{34 * 10\,bits/byte}{460800(bits/s)} = 0.738\,ms \\ Gyr. &= \frac{34 * 10\,bits/byte}{460800(bits/s)} = 0.738\,ms \end{aligned} \tag{63}$$

It takes about $5\,ms$ for the complete 6 messages to be fully transmitted, giving a lot of free time in the $50\,Hz$ update rate used to send other debugging messages. In summary, the sensors are sampled at 100Hz, their values are corrected and fed to the Madgwick filter which runs at 1600MHz, and the information is sent to the serial port at $50\,Hz$. It is also important to notice that the quaternion is transmitted in the East-North-Up (ENU) Frame, which is useful information when fusing the IMUs with the GNSS.

The transmitted messages are received by the ODROID, which publishes two ROS topics "**imu/dataL**" and "**imu/dataR**" which are of the type "**sensor_msgs/Imu**" and hold the following information

- Quaternion
- Orientation Covariance
- Angular Velocity
- Angular Velocity Covariance
- Linear Acceleration
- Linear Acceleration Covariance

The Xsens MTi-G, is using a third party driver for ROS that is available in the official Xsens website and is configured to communicate with a 921600 baudrate, and to output the orientation, angular velocity, linear acceleration, and GPS information. The IMU information is published in a ROS topic "**imu/data**" at $48\,Hz$, and the GPS information in a "**gps/fix**" topic of the type "**NavSatFix**" at $4\,Hz$. When configuring the MTi-G it is very important to choose the right onboard filter, as some of them don't use the magnetometer in the fusion algorithm causing the orientation to drift a lot with time. In this case, it is being used the aerospace filter which uses all the sensors in the fusing algorithm, and the output orientation follows the North-West-Up (NWU) frame.

A schematic with all the components used and their respective connections can be seen in Appendix A. It was also done a small study on the inertial sensors stability using the Allan Variance, which can be read in the Appendix B.

4.2. RTK GNSS Implementation

As explained in section 2.1.4, the RTK technique involves using two receivers within a small distance to archive and higher accuracy. In this case it is being used two different systems so we can compare high end receivers with cheaper ones. The first system is composed by two u-blox NEO-6P receivers using Trimble Bullet antennas, and the RTK solution is processed separately using an open source software. The second system is composed by two Septentrio AsteRx-4 receivers with Septentrio PolaNt-x MF (Multi-Frequency) antennas, which can perform RTK solutions on its own.

Table 5: GNSS Receivers Specifications

GNSS RECEIVERS PARAMETERS SPECIFICATION								
Device	**Updates**	**GPS**	**GALILEO**	**GLONASS**	**BEIDOU**	**IRNSS**	**QZSS**	**Mode**
u-blox NEO-6P	5 Hz	L1	-	-	-	-	-	SPS SBAS PPP
Xsens MTi-G	4 Hz	L1	E1	-	-	-	-	SPS SBAS
Septentrio AsteRx-4	50 Hz	L1 L2 L5	E5 E6	L1 L2 L3	B1 B2 B3	L5	L1 L2 L5	SPS SBAS PPP RTK

As the systems have lot of differences, Table 5 shows some of the receivers specifications.

For the RTK system to work it is necessary to have some kind of connection between the two receivers in order for them to communicate with each other. To solve this problem it is commonly used an WiFi connection or other kind of RF communication directly between both devices. However, with these solution it is necessary to maintain a good line-of-sight in order for the connection to be stable, which limits the distance between the receivers and the places where the system can be used. Therefore in the final implementation of the RTK system, the base station was connected to the home internet connection and runs a TCP server broadcasting all the information. In order for the rover to access all the broadcasted information it is being used a smartphone with 4G Long-Term Evolution (LTE) high-speed internet to connect the rover to the internet and the base station. Figure 16 shows a diagram of the RTK implementation.

In a RTK system the base station is used to send corrections messages to the rover. There are a lot of messages and standards to broadcast this information, but in this case it was chosen the Radio Technical Commission for Maritime Services (RTCM) v3, and the messages shown in Table 6.

The u-blox receivers are using the open source software RTKLIB to estimate the RTK solution. The software was configured to transmit the messages at $5\,Hz$ which is the maximum update rate of the receiver. A brief explanation on how the software works can be read in Appendix E. The Septentrio receiver defaults the update rates for the corrections to every 2 seconds, but as the

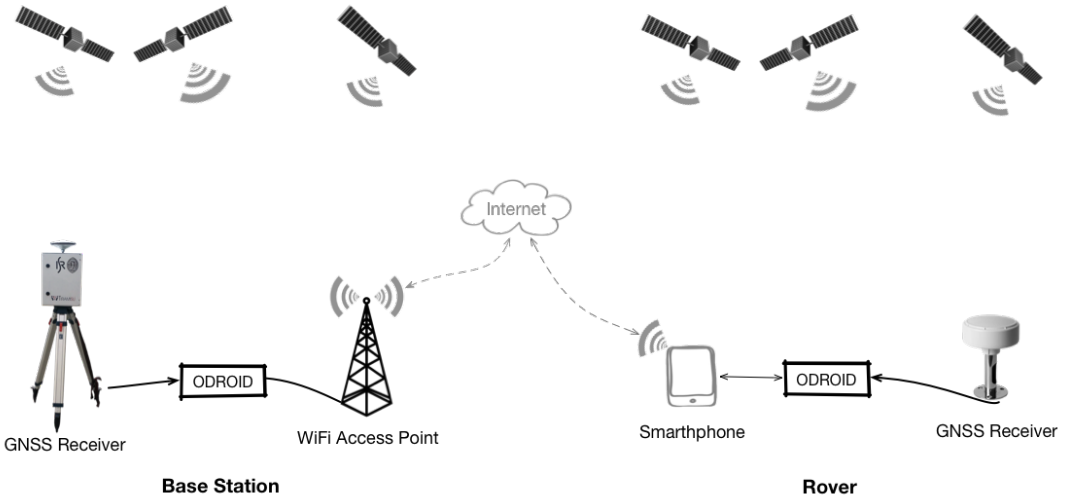


Figure 16: RTK-GNSS Connection Diagram

Table 6: Base Station Broadcasted Messages

RTCM Version 3.x Messages		
	Septentrio	u-blox
1002 - Extended L1 GPS RTK Observables		✓
1004 - Extended L1+L2 GPS RTK Observables	✓	
1006 - Stationary RTK Reference Station ARP with Antenna Height	✓	✓
1012 - Extended L1+L2 GLONASS RTK Observables	✓	
1013 - System Parameters, List of Transmitted Message Types and Update Rates	✓	✓
1019 - GPS Satellite Ephemeris Data	✓	✓
1020 - Glonass Satellite Ephemeris Data	✓	
1033 - Receiver and Antenna Descriptor	✓	
1044 - QZSS Satellite Ephemeris Data	✓	
1045 - Galileo F/NAV Satellite Ephemeris Data	✓	
1230 - GLONASS Code/Phase Bias Correction	✓	

implemented network infrastructure was stable the value as set to the same 0.2 seconds as the u-blox.

As explained during this section a RTK system is divided in two parts, the fixed part (base station) and the moving part (rover). As it is being used to 2 RTK systems, the base station is composed by one receiver and antenna of each type, one ODROID, and one router with internet connection (See Figure 17).

The moving part in this case has all the components necessary for the rover part of the RTK system, but also all the components used in the INS implementation. It is composed by an aluminium bar with secures the two antennas used by the Septentrio receiver, the Xsens MTi-G in the centre, two Invensense MPU-9150 secured to the bar with duck tape, and a Trimble antenna used by the u-blox receiver (See Figure 18). As well as, all the rest of the electric components like cables, the GNSS receivers, another ODROID, the Stellaris Launchpad, and a battery.

The first thing that needs to be done when performing RTK solutions is to find an accurate position for the base stations, as the position errors present here will offset the rover position. In order to have an accurate base station position and to study the random walk evolution of the GNSS receivers, both systems were left running for 24 hours logging all the messages to a Receiver

Figure 17: Real Time Kinematic - Base Stations

Figure 18: Aluminium Bar used in the Rover

Independent Exchange Format (RINEX) file. After the 24 hours logging period it was then waited another 24 hours for the more precise RAPID GNSS Orbit Ephemerides to become available online, and then the RINEX files were post-processed with the Canadian Spatial Reference System (CSRS) Precise Point Positioning (PPP). It is important to notice that as the Septentrio PolaNt-x MF are high-end GNSS antennas, it was used the calibration model available online, and the Antenna North Reference Point (NRP) was oriented toward the true north [**?**]. The estimated positions and errors are summarised in Table 7.

As the base stations are not in the same position, it is not expected for values to be the same. The distance between the two antennas was measured with a measuring tape and the difference between them was about ~68cm North, ~45cm East, and ~35cm Altitude, which is more or less the difference obtained from the Post-Processing. Figure 19 shows the estimated position random walk that occurred during the 24 hours logging period. The green crosses represent the initial position and the red ones the final position. The complete Post-Processing results can be found in the Appendix F and G, where other interesting plots can be analysed. Such as, it takes 9 hours of data collected for the Septentrio to have a good estimated position, however the u-blox needs 15 hours and still has some random walk after that.

It was also done a quick test to the three antennas used, the Septentrio PolaNt-x MF, the Trimble Bullet GG, and the patch antenna provided in the Xsens MTi-G Development Kit (See Figure 20).

Table 7: Base Stations Estimated Position - 24Hours Post-Processing

Base Stations Estimated Position						
	Septentrio			**u-blox**		
	Latitude (+n)	Longitude (+e)	Ell. Height	Latitude (+n)	Longitude (+e)	Ell. Height
ITRF08 (2016)	40°11’ 32.9839’’	-8°25’ 04.4044’’	100.609 m	40°11’ 33.0074’’	-8°25’ 04.3869’’	100.204 m
Sigmas (95%)	0.001 m	0.001 m	0.003 m	0.017 m	0.013 m	0.034 m
Universal Transverse Mercator (UTM) Estimated Position						
Zone 29	4449285.137m (N)	549549.062m (E)	---	4449285.864m (N)	549549.471m (E)	---

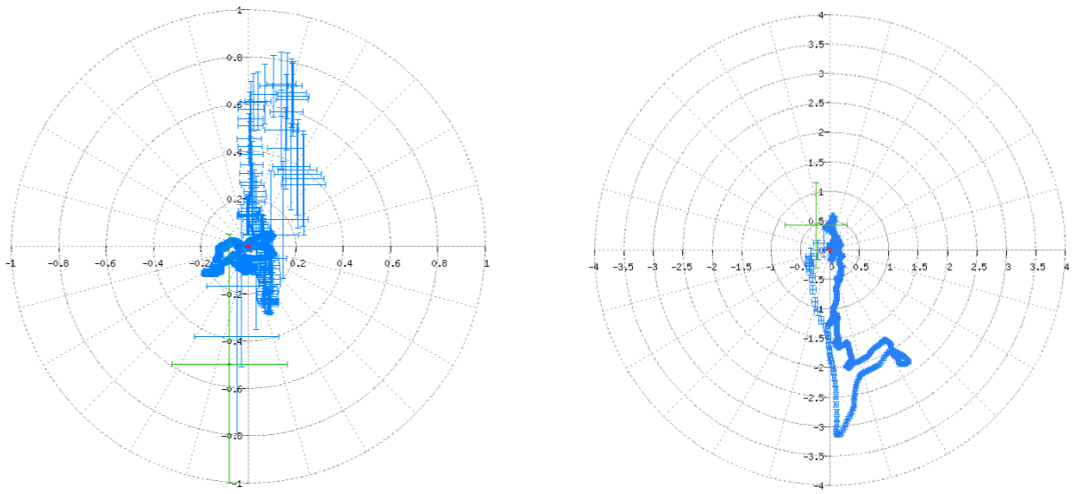


Figure 19: Septentrio (left) and u-blox (right) Base Stations Position Random Walk

To perform the test each antenna was placed in the same spot and connected to the u-blox NEO-6P GNSS receiver for 5 minutes, and after that the SNR for each satellite was written down. It isn't a scientific test and doesn't say that much about the quality of the antennas, because there are other important aspects other than SNR, but it was a quick and interesting test. All of them are active GPS L1 band antennas and all need the 3.3V DC supplied by the receiver. One aspect that should be noticed is that the Septentrio PolaNt-x MF, is a multi-frequency antenna with calibration models available and built-in ground planes, so it needs a good GNSS receiver to take fully advantage of its capabilities. Another aspect to notice is that the Xsens antenna has a built-in 3 meters long cable with an attenuation of 3.9 dB (3m), and for the other two antennas was used the same 1 meter long RG-58C/U cable which only has 0.63 dB per metre attenuation at 1.5GHz.

As can be seen in Figure 21 the PolaNt-x MF quality is undeniable as the antenna manages to obtain the highest SNR for all the satellites. However, the Trimble Bullet GG Antenna also has good quality results with only a few dBHz lower than the Septentrio. As far as the Xsens Antenna, the u-blox receiver struggled a lot to track satellites and get a solution, as most of the time it couldn't track that many satellites as displayed.

To test the fusion algorithm the aluminium bar with all the sensors and receivers was fitted on top of a car, and then driven around Coimbra while running the data fusion algorithm and at the same time recording all the data. The aluminium bar has one meter long, and its centre position is

Figure 20: Septentrio PolaNt-x MF (left), Trimble Bulet GG (center), Xsens MTi-G (right)

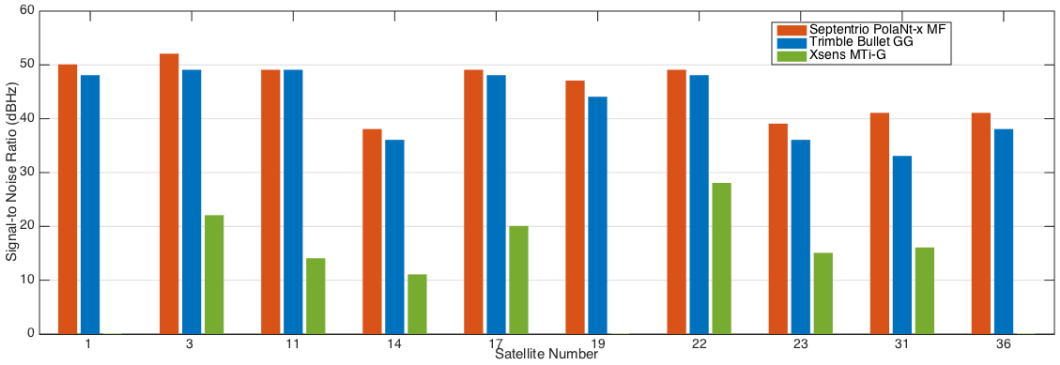


Figure 21: Antenna Quality SNR Test

represented by the Trimble antenna. The rest of the devices have the following offsets described in Table 8. As the bar was positioned relatively low (about 1 meter above the ground) the GNSS signals can be easily blocked and suffer from multipath errors. The test was done in an urban environment with lots of buildings, trees, overpasses, narrows streets, and some places without sky view. The route has 20 km takes about 30 minutes to complete, and was done in two separate days. In the first day, it was only tested the rover part of the system with the GNSS receivers working without the base stations. In the second day, the rover was connected to the base stations and the GNSS receivers were working in RTK.

4.3. GNSS/INS Integration

During the first test the following solutions were recorded:

1 - GNSS Positioning with the u-blox NEO-6P using Carrier Phase

2 - GNSS Positioning with the u-blox NEO-6P using Code Pseudorange

3 - GNSS Positioning with the Septentrio AsteRx4

4 - GNSS/INS Integration with the Xsens MTi-G

5 - INS Solution from 2 Invensense MPU-9150

6 - GNSS/INS Integration from the u-blox **(1)** and the 2 Invensense MPU-9150 **(5)**

7 - Low Speed Test with the u-blox NEO-6P using Carrier Phase

8 - Low Speed Test with the Xsens MTi-G

9 - Low Speed Test with the Septentrio AsteRx4

Table 8: Rover - Aluminium Bar Position Offsets

Rover Aluminium Bar Position Offsets			
	X (m)	Y (m)	Z (m)
Trimble Antenna for the u-blox receiver	0.00	0.00	0.00
Xsens MTi-G	0.13	0.00	-0.12
2x Invensense MPU-9150	0.12	±0.10	-0.13
2x Septentrio Antennas	0.13	±0.35	0.07

An overview of the route travelled during the test can be seen in Figure 22. This route was mainly chosen as an attempt to put together all kind of problems that can be expected when using these kind of devices. As it is difficult to show the errors and the small differences between all the estimated trajectories, Appendix H has a link to Google Maps where the respective routes can be analysed and also has some additional photos of this section. The Google Maps is a good tool to easily preview the solutions and estimate their accuracy, however when analysing them we must be careful as it can also yield wrong conclusions. This is because, the satellite imagery can have slight offsets from the real positions, and also because the latest satellite imagery available from the region are from January 2013.

The **u-blox NEO-6P (1)** allows to configure the output format of the solution between the proprietary UBX messages, or the more common National Marine Electronics Association (NMEA) messages. The NMEA-0183 messages are the easiest way to get all the information needed, as they are quite clear and can be easily logged through the serial port. The receiver was also configured in Precise Point Positioning (PPP), which helps stabilising the measured distance between the satellites and the receiver by using the carrier phase observables. It was also enabled the reception of signals from the Satellite-Based Augmentation System (SBAS) constellation, which helps archiving a better accuracy by receiving ionospheric corrections, clock drifts, and ephemeris errors. The final estimated position is then outputted using NMEA-0183 messages with an update rate of 5 Hz.

As a means of comparison it was also logged another u-blox NEO-6P **(2)**, but this time outputting the UBX RAW messages and with the solution being processed by the RTKLIB software in single mode. In this mode it is only used the pseudorange data which makes the precision quite low, and as the solution is being processed by the RTKLIB it isn't filtered by the u-blox chipset.

The **Septentrio AsteRx4 (3)** was configured in a similar way as the u-blox (1), as it receives

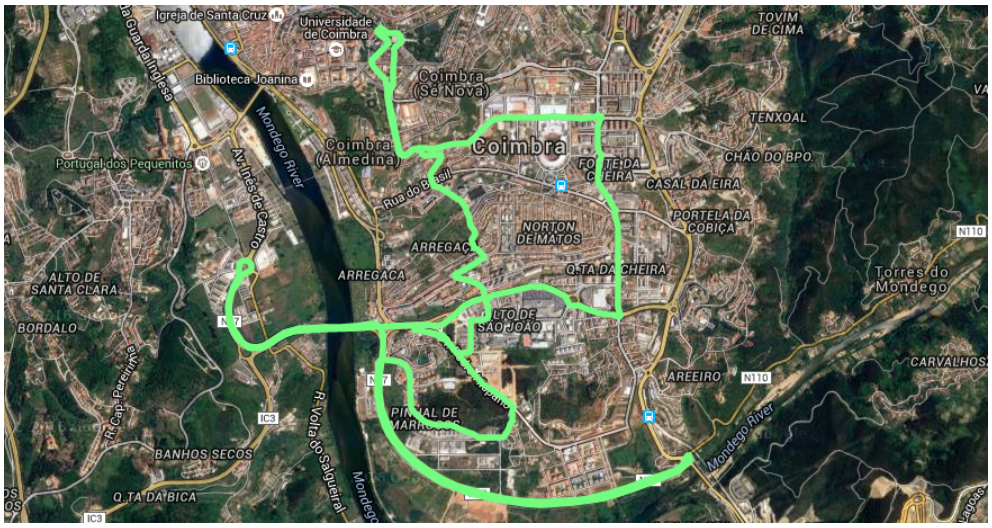


Figure 22: Overview of the route travelled

all the signals available, estimates the solution, and broadcasts it in NMEA-0183 messages at 50 Hz. The receiver is connected to 2 antennas so it can also estimate its attitude and use the heading and pitch angles to help estimating an accurate position. An interesting remark is that initially when the receiver was set up as Standard Positioning Service (SPS), it was able to receive signals from the GPS, GLONASS, Galileo and BeiDou constellations seamlessly. But when configured in Differential Global Positioning System (DGPS) to make use of the SBAS corrections, the receiver was only able to track the GPS satellites. After some time trying to circumvent the problem without luck, it was contacted the Septentrio support and after some emails, the receiver was updated with the latest firmware which solved the problem and allowed the receiver position error to halve.

The **Xsens MTi-G (4)** only has a 4 Hz update rate for the GNSS receiver, but with the internal GNSS/INS fusion the update rate can go up to 400 Hz. The device was configured to fuse the all its sensors and output the solution at 48 Hz.

The information from both **Invensense MPU-9150 (5)** is received from the Stellaris Launchpad at 50Hz and used to calculate the INS solution and the GNSS/INS integration. In the same Google Maps link provided previously it is also displayed 4 minutes of the INS solution without the aids of GNSS receivers. After 4 minutes and with the sensors moving at a relatively high speed, the estimated velocity starts to have a fairly amount of errors and therefore the estimated position drifts a lot from the real position. This solution allow us to have some visual information on how the errors grow in this type of sensors.

The **GNSS/INS Integration (6)** is estimated in real time using the solution obtained from the u-blox (1) and the information received from both Invensense MPU-9150 (5) using the fusion algorithm described in chapter 3.

The **Solutions 7, 8 and 9**, represent a secondary test done in a parking lot with good sky view to test the accuracy of the receivers in low speeds conditions.

As expected the solutions 1, 3, and 4 are quite similar in areas with good sky view, however when the conditions start to deteriorate, the errors increase and the solutions diverge.

One of the most remarkable aspects is the quality of the Septentrio solution when compared to the the other two receivers. The fact that it uses two high quality antennas, can track more satellites signals, uses multiples frequencies, and has a high update rate, allows the receiver to estimated an almost perfect trajectory even in areas with bad sky view and where multipath errors were expected.

As for the u-blox, in areas with good sky view it tends to yield a solution that follows quite close the generated by the Septentrio. However, in areas with big building and narrow streets the satellites being tracked starts to decrease and the accuracy of the solution rapidly degrades.

The Xsens MTi-G has a lower quality GNSS receiver when compared with the other two devices as it can only measure the code pseudorange, so the device strongly relies on a good GNSS/INS

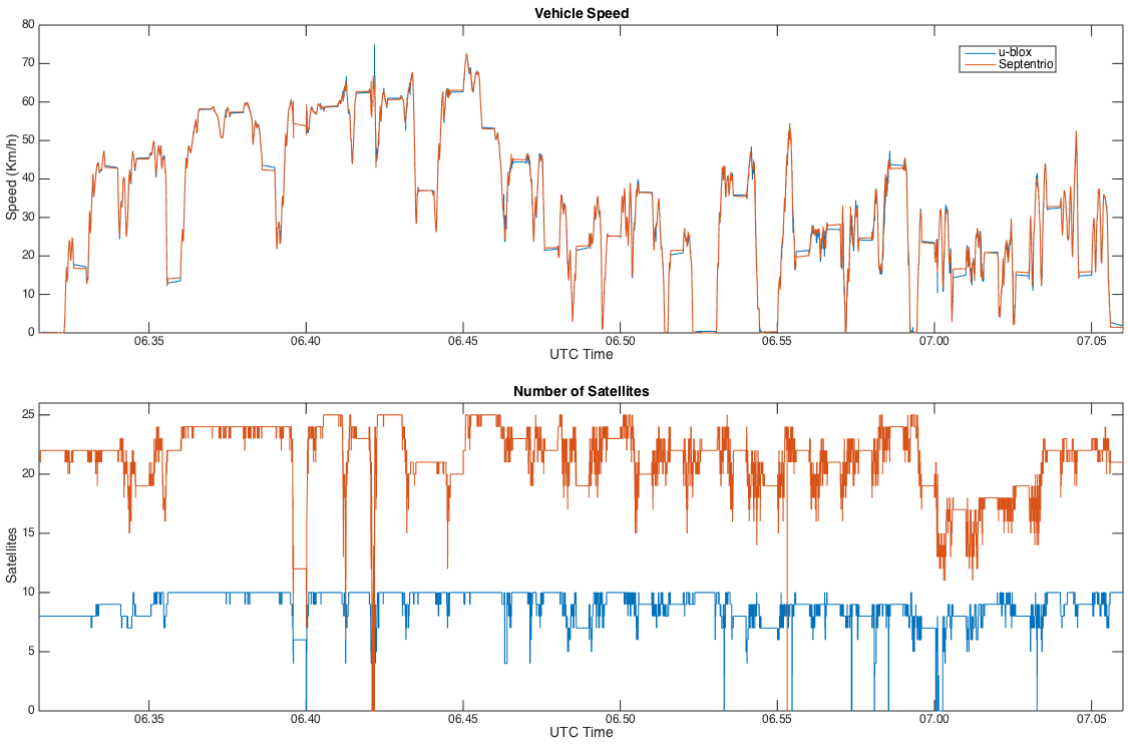


Figure 23: Vehicle Speed and Number of Satellites

integration. The solution generated with the Xsens differs quite a bit from the others receivers, this can be seen mostly after road turns (IMU rotations) where the solution tends to delay a bit and become parallel to the others solutions.

The following figures in this section represent a comparison between the u-blox (1) and the Septentrio (3), as the Xsens MTi-G doesn't share this information it's impossible analize it.

Figure 23 shows the speed of the vehicle and the number of satellites being used to estimated the position by the receivers during the route. As expected the reported velocity by both receivers is quite similar, and the drops in satellites tracked affect both receivers. Figure 24 shows the 1σ errors for the latitude, longitude and altitude measurements. Their mean value is estimated in the equation 64 and 65. A zoomed version of the 1σ errors can be seen in Figure 26 for two different situations of the route, one with good sky visibility and the other one not so much. In that figure, both plots have the same X-Axis scale (4 minutes) but the Y-Axis is different between them.

Analysing the plots, the NMEA messages, and the map, we can see that the first three big drops in satellites that occur between 06:39 and 06:42, correlates with three underpasses where the signals are partially blocked. As the Septentrio can track more satellites, these drop doesn't affect it that much even when reporting high errors. However, the first underpass makes the u-blox completely lose its positioning due to the lack of available satellites. The next big drop it is also an underpasses, but this time 150 meters long without sky view followed by a second small one. This time it is expected for the GNSS receivers to have some troubles with the signals. It is also a good time to

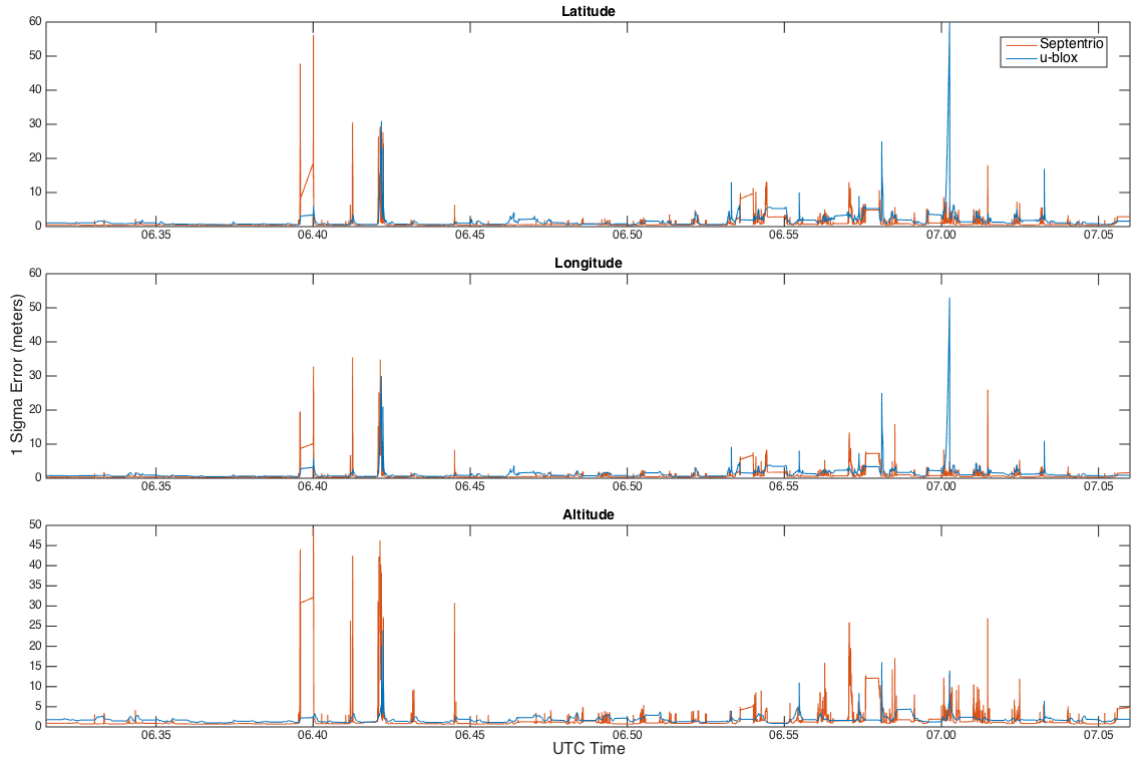


Figure 24: Latitude, Longitude and Altitude - 1 Sigma Errors

see how long they will take to recover and start reporting an accurate position again.

Analysing the NMEA messages and Google Maps, the u-blox was 9 seconds without being able to estimate a position, and then took 2.5 seconds to estimate an accurate position which was 50 meters after exiting the underpass. The Septentrio managed to get some signals in the middle of the underpass and was only 5 seconds without a position. The accurate position took only 1 second and was less than 25 meters after exiting the underpass. This can be seen in detail in Figure 27, where each green dot represent an estimated position with 3D Fix. In all these satellite drops the GNSS/INS integration seems to have taken care of the problem and made a good estimation of the real position. However, after the big underpass without sky view, the first values reported by the u-blox come with high errors and that makes the correction of the GNSS/INS integration to go off the road.

$$
\begin{aligned}
Septentrio \quad \overline{1\sigma_{LAT}} &= \frac{1}{n}\sum_{i=1}^{n} 1\sigma_{LAT}(i) = 0.9402m \\
\overline{1\sigma_{LON}} &= \frac{1}{n}\sum_{i=1}^{n} 1\sigma_{LON}(i) = 0.7245m \qquad (64) \\
\overline{1\sigma_{ALT}} &= \frac{1}{n}\sum_{i=1}^{n} 1\sigma_{ALT}(i) = 1.5829m \\
u-blox \quad \overline{1\sigma_{LAT}} &= \frac{1}{n}\sum_{i=1}^{n} 1\sigma_{LAT}(i) = 1.7310m \\
\overline{1\sigma_{LON}} &= \frac{1}{n}\sum_{i=1}^{n} 1\sigma_{LON}(i) = 1.4429m \qquad (65) \\
\overline{1\sigma_{ALT}} &= \frac{1}{n}\sum_{i=1}^{n} 1\sigma_{ALT}(i) = 1.8365m
\end{aligned}
$$

Figure 25: Detailed view from part of the route - 1 - u-blox (left) and Septentrio (right)

The next two drop in satellites happens after crossing the bridge and when passing a small overpass, each of this drops is the same overpass but in different directions. In this case both drops didn't cause a great effect in the the accuracy of the solution. After crossing the bridge again and at 06.45, starts a more urban area with the appearance of buildings, trees, and slower traffic. At 06.55 the receivers were stationary for some time with the signals being partially blocked by high building closed to the road and also by a tall bus standing near the antennas. Despite this the receivers were able to hold the solution with high measurements errors. However, shortly after that they both lose the solution in different places due to trees covering the sky. The same thing happened again in the u-blox at 06.57 when stopped at a traffic light under some trees. This section is very problematic for both the GNSS receivers and the inertial sensors, because it has lots of building and trees covering the sky and at the same time the road is made of old bricks which introduce a lot of vibrations in the car, the IMUs, and the antennas.

Finally at 07.00 starts a slow section with narrow roads, high building close to the road, and also a climb where the trees completely cover the sky. This is where high errors start to appear for the u-blox and the Xsens, and surprisingly the Septentrio keeps a good solution. Figure 27 shows the beginning of this path in detail where we can see that the u-blox completely loses the position for some time but the GNSS/INS integration is able to estimates it quite well. Later on, during the climb with the trees covering the sky, both receivers were able to hold some satellites, but the u-blox reports a wrong position and the Integration doesn't correct it.

Another way to quickly analyse the accuracy of a GNSS solution is by the fix quality indicator

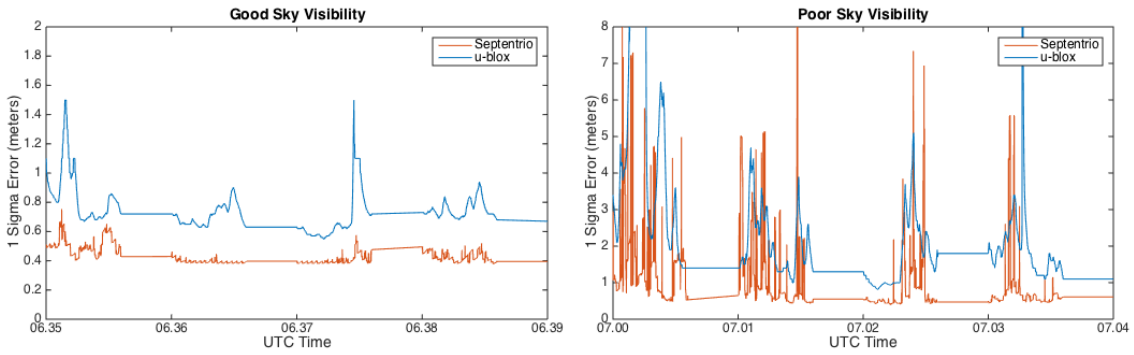


Figure 26: Latitude 1 Sigma Errors - Zoomed Version

Figure 27: Detailed view from part of the route - 2 - u-blox (left) and Septentrio (right)

broadcasted in the NMEA messages. When the value is 2, the receiver has a DGPS fix, meaning that is using the SBAS satellites and receiving corrections, which helps the accuracy. If the fix quality is 1 then the receiver only has a SPS fix, so the receiver is only using the common satellite and the accuracy will depend mainly from quality of the receiver and the antenna. Finally, if the fix quality drops to 0, then the number of satellites being tracked dropped bellow the minimum 3 and the receiver stopped being able estimate the position.

Low Speed Test

The low speed test allows to investigate how the GNSS receivers behave when used in vehicles with slower dynamics and at the same time use the parking spaces to estimate the accuracy of the solution and the Google Maps satellite imagery. The path has only 200 meters and consists mainly in 90 degrees turns and a small loop at the end. To estimate the real velocity was used used the NMEA messages broadcasted by the Septentrio which is higher accuracy GNSS receiver available (See Figure 29).

Figure 30 show the trajectory estimated by the receivers, the light blue path is from the u-blox, the orange from the Xsens, and the green from the Septentrio. It can be seen that the u-blox and

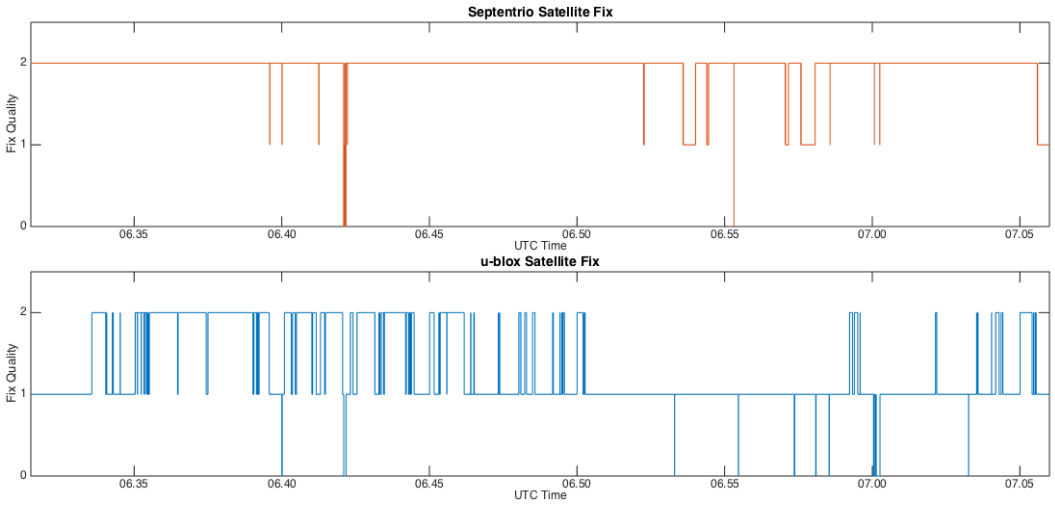


Figure 28: GNSS Receivers Fix Quality

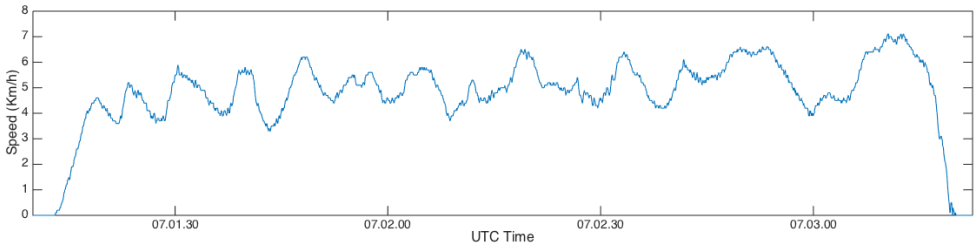


Figure 29: Low Speed Test - Velocity reported by the Septentrio AsteRx4 Receiver

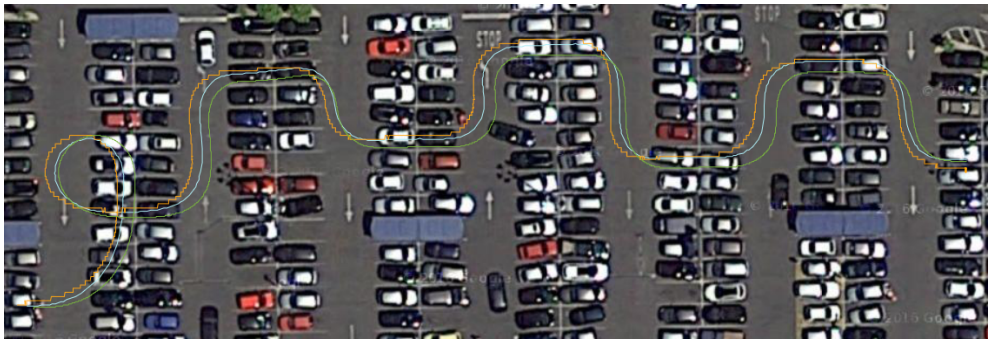


Figure 30: Low Speed Test - Satellite Image

the Septentrio started almost in the same place, with the Xsens having a little offset. Either way, none of the the solutions are starting in the right place. The real starting point is the middle of the white car, being the closest solution from the u-blox. Nevertheless, these are still good errors for the single GNSS receivers. After starting to move, it was always tried to pass as closely as possible to the middle of both lines of the parking spot. After the first two 90 degree turns, the u-blox started to have a solution closer to the Xsens, but there are still some differences between all of them. Interestingly, all the three solution report the final location pretty much in the same place.

To end this section, Figure 31 show us the estimated 1 sigma errors for the low speed test. Once again, the Septentrio's ability to track multiple satellite constellations in multiple frequencies allows the receiver to estimate a more accurate solution. Nonetheless, the u-blox still has a very good accuracy considering it is only using the GPS and SBAS constellations.

4.4. RTK-GNSS/INS Integration

During this test the following solutions were recorded:

1 - RTK-GNSS Positioning with 2 u-blox NEO-6P

2 - RTK-GNSS Positioning with 2 Septentrio AsteRx4

3 - INS Solution from 2 Invensense MPU-9150

4 - RTK-GNSS/INS Integration from the u-blox and the 2 MPU-9150

5 - Low Speed Test using 2 u-blox NEO-6P

6 - Low Speed Test using 2 Septentrio AsteRx4

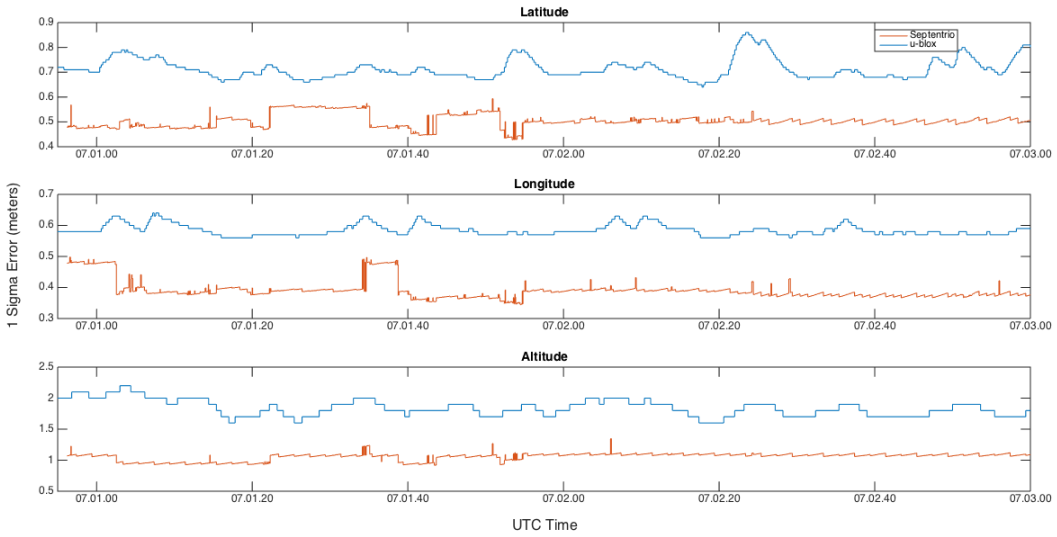


Figure 31: Low Speed Test - 1 Sigma Error

The **u-blox Solution (1)** is processed using the UBX RAW messages and the RTKLIB software in Kinematic mode. It is worth mentioning that it is being used the RTKLIB version from Tim Everett which fixes a lot of the bugs that exist in the software, improves the time needed to estimate the ambiguities, and also improves the overall stability of the RTK fix solution. His blog is really worth reading as he explains his modifications followed by images showing the improvements [Everett, 2016]. The software is outputting the solution to ROS topics as well as NMEA-0183 messages with an update rate of 5 Hz.

The **Septentrio Solution (2)** is quite similar to the previous one, the position mode was changed to RTK and the IP address from the base station was added as an input server. The receiver it is still configured to receive all the signals available, estimate the solution, and broadcast it in NMEA-0183 messages at 50Hz.

The **Solutions 3 to 6** are similar to the previous section, therefore there is no need to explain how they were generated again. Appendix I has the link to the new Google Maps with the RTK routes, as well as other figures about this section.

When generating RTK-GNSS solutions one of the most important things we should pay attention is the solution fix status. What does this mean?

The RTK solution can have two values, **"FLOAT"** which is the starting value for the solution and means that the receiver is getting data from the base station, performing single difference corrections, but still doesn't have the minimum carrier phase ambiguities solved to integer numbers, and therefore it is still not performing double differences. When the minimum number of satellites with their carrier phase ambiguities fixed is reached and the solution passes a quality check, the receiver starts performing double differences corrections and the solution is then called **"FIXED"**.

This information is broadcasted in the NMEA messages using the same parameter previously

mentioned as the fix quality indicator. However, in the RTK a float solution is represented by the number 5, and the fixed solution by the number 4. If at any time the rover stops receiving information from the base station, this can be easily identified in the fix quality indicator which in this case should have the values used for single GNSS positioning.

The first aspect that need to be noticed, is the time that it takes from booting up the system until obtaining a fixed RTK solution. For the Septentrio receiver this usually takes less than one minute, but the RTKLIB with the u-blox receivers using only GPS and SBAS can take from 5 to 30 minutes depending on the satellites being tracked, the atmospheric conditions, and the sky visibility.

There are two interesting parameters used in the RTKLIB software, the "minimum fixed satellites to obtain a fixed solution" and the "minimum fixed satellites to hold the fixed solution". In this case the chosen values were 5 and 4 satellites respectively, which is the lowest value possible and can have the drawback of generating wrong estimations in the internal EKF, but as the receivers only support GPS and SBAS satellites it is probably the best value use.

Why are these values important? If we think of one example where the base station is tracking 13 satellites and the rover 8 satellites, it is normal for the RTK solution to be only using 5 or 6 satellites due to the other ones being blocked by restrictions as the low SNR or low position in the sky. It was shown in section 4.3 that the number of satellites being tracked can drop a lot in some areas, and reflecting those values with this new information on how the RTK works, we can expect the u-blox to have problems holding the fixed solution. As for the Septentrio, with lots of satellites being tracked it can get the minimum satellites for the fixed solution faster and also hold it better.

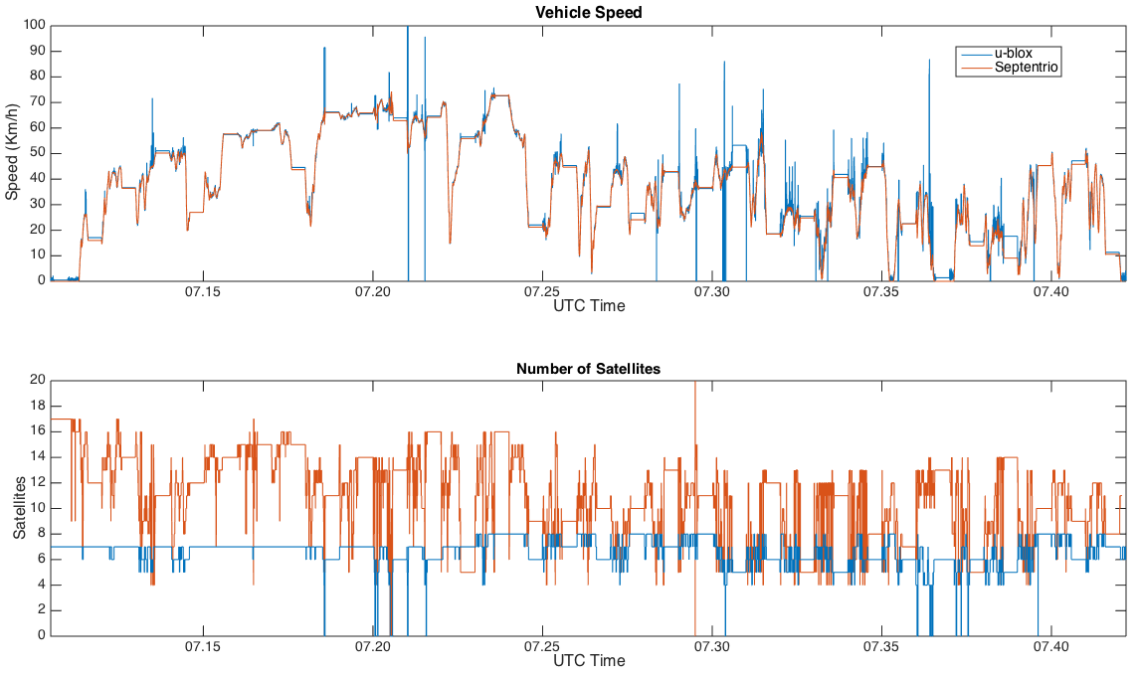


Figure 32: RTK Vehicle Speed and Number of Satellites

Figure 33: Detailed view from part of the route - 3 - RTK u-blox

As the route is the same, was performed in similar atmospheric conditions, and almost at the same time of the day, it is going to be interesting to see the differences between the RTK and the non-RTK solution. The first thing that can be noticed in Figure 32 is that the u-blox estimated speed has a lot more noise than the Septentrio.

Comparing Figure 23 and 32, we can see that both RTK solutions use fewer satellites to estimate the position as explained previously. For the u-blox receiver this is actually a problem, as this time the receiver actually stops tracking satellites in all the first three overpasses that appear during the route. (Figure 33). The Septentrio is also affected in the first overpass but only loses the position for 0.5 seconds. The big underpass also has impact in the RTK solution as can be seen in Figure 34. The u-blox has no satellite communication for 9 seconds, and after exiting the underpass takes 25 meters to estimate a new position. The Septentrio is able to track some satellites in the middle of the underpass once again, and therefore is only 3.9 seconds without signals. After that it only takes 4 meters for the receiver to broadcast new positions.

The RTK-GNSS/INS integration has no problems in the first three situations, as the the estimated route has a good approximation and follows what is to be expected from the previous measurements. In the big underpass the estimation starts appropriately but as the new GNSS measurements arrive with high errors and saying that we are moving in the opposite direction, this makes the estimation to move to the wrong side and go off the road.

Analysing the fix quality indicator in Figure 35, the Septentrio shows the expected switching between the various types of solutions as the number of satellites increases and decreases. However, the u-blox/RTKLIB doesn’t have this behaviour, and after loosing the RTK fix it still tries to recover

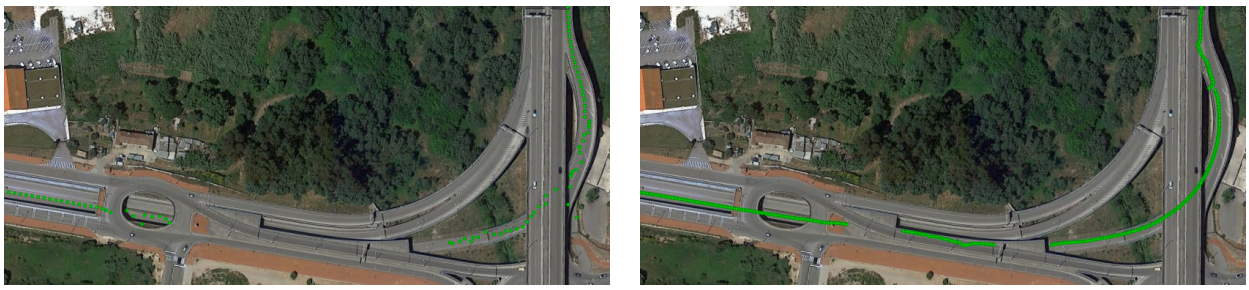


Figure 34: Detailed view from part of the route - 4 - RTK u-blox (left) and RTK Septentrio (right)

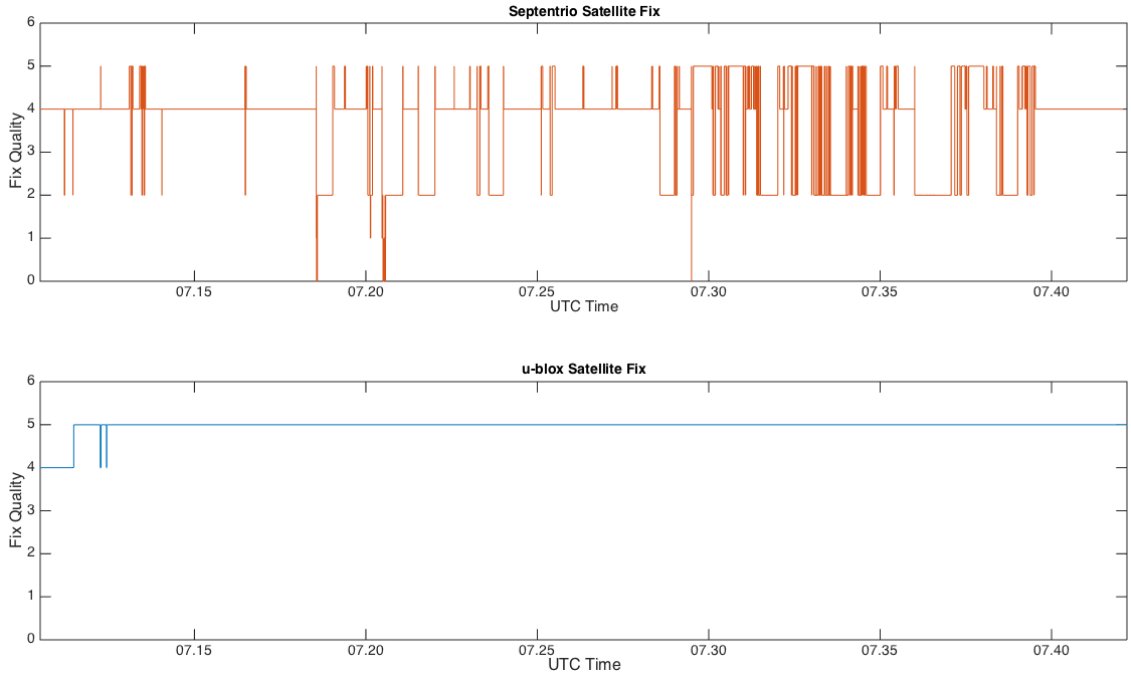


Figure 35: RTK GNSS Receivers Fix Quality

from the RTK float solution but never get it back. This problem will be investigated later on. The switching between RTK and standalone GNSS receiver it is also not reported by the RTKLIB, so it is almost impossible to have that kind of information. However, it is always possible to take some conclusions on how the system is working by looking at the satellites being tracked.

For the Septentrio the only thing interesting that happens during the rest of the route, is that this switching from float to fix to float can be actually seen in some places as it causes a small shift in the position. Apart from that, there isn't anything more that needs to be reported as the receiver does its job quite well. As for the u-blox, with good sky view it works fine, but otherwise the solution starts to have problems and the multipath errors are really easy to identify. Figure 36 and 37 are the 1 sigma errors reported by the receivers during the route, with their mean value estimated in equations 66 and 67.

$$\begin{aligned} Septentrio \quad RTK \quad \overline{1\sigma_{LAT}} &= \frac{1}{n}\sum_{i=1}^{n} 1\sigma_{LAT}(i) = 0.2883m \\ \overline{1\sigma_{LON}} &= \frac{1}{n}\sum_{i=1}^{n} 1\sigma_{LON}(i) = 0.2819m \\ \overline{1\sigma_{ALT}} &= \frac{1}{n}\sum_{i=1}^{n} 1\sigma_{ALT}(i) = 0.6522m \end{aligned} \tag{66}$$

$$\begin{aligned} u-blox \quad RTK \quad \overline{1\sigma_{LAT}} &= \frac{1}{n}\sum_{i=1}^{n} 1\sigma_{LAT}(i) = 1.0103m \\ \overline{1\sigma_{LON}} &= \frac{1}{n}\sum_{i=1}^{n} 1\sigma_{LON}(i) = 0.4051m \\ \overline{1\sigma_{ALT}} &= \frac{1}{n}\sum_{i=1}^{n} 1\sigma_{ALT}(i) = 0.8267m \end{aligned} \tag{67}$$

The values show a clear improvement from the previous section. However, the u-blox latitude error seems to be higher than expected when compared to the longitude and altitude.

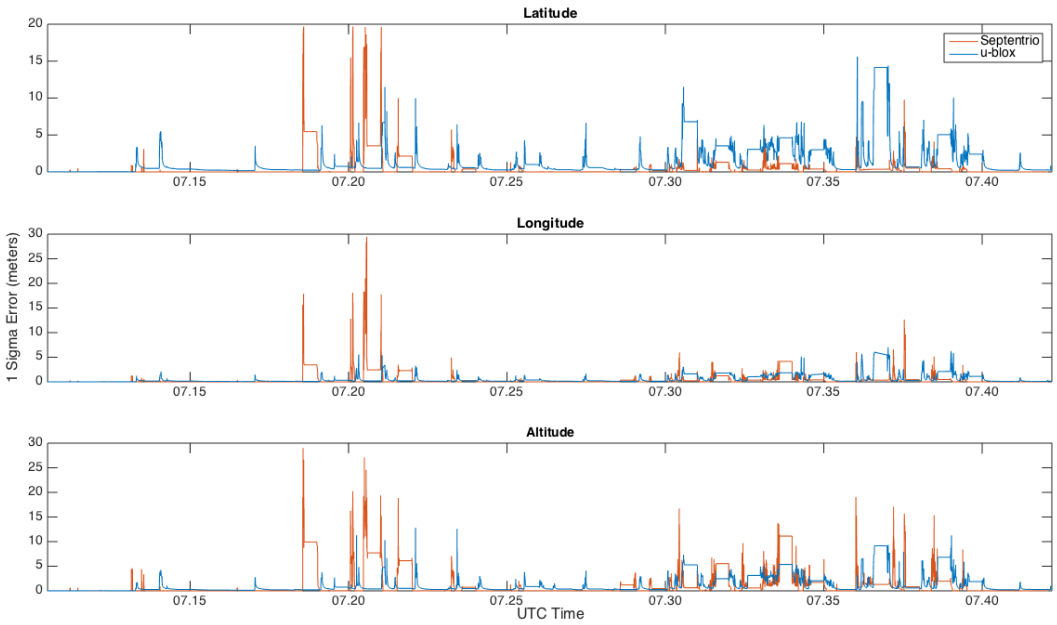


Figure 36: RTK Latitude, Longitude and Altitude - 1 Sigma Errors

The high error section of this route that starts at 07.30 and lasts until 07.40, is a little bit problematic for the RTK-GNSS/INS integration. Sometimes the GNSS estimations are all over the place, and it is difficult for the filter to identify what to follow. Another times the solution is accurate but the broadcasted covariance values are still high and again makes problems in the data fusion. Anyway, the estimated RTK-GNSS/INS integration has an overall good performance.

Another important aspect to take into account is the delay in the communications between the base station and the rover, and the RTK correction age used for the solution. Analysing Figure 38 and the NMEA messages, shows that RTKLIB sends every message received in the base station to the rover, and then processed the solution with all the information. As the receivers are working at 5Hz, the solution has 0.2 seconds of correction age with additional delay introduced in the datalink. The average value for the correction age in the u-blox solution is 0.267 seconds.

The Septentrio has a higher value for the correction age even when the base station is configured to send corrections at 5Hz. As the value seemed a little bit odd, both receivers were tested on a local network, and yet the values were oscillating between 13 and 15 seconds. As the receiver was getting a good accuracy and reporting that everything was fine with the communication, it was

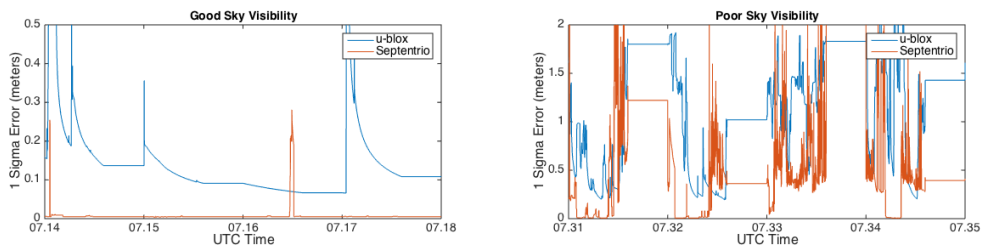


Figure 37: RTK Longitude 1 Sigma Errors - Zoomed Version

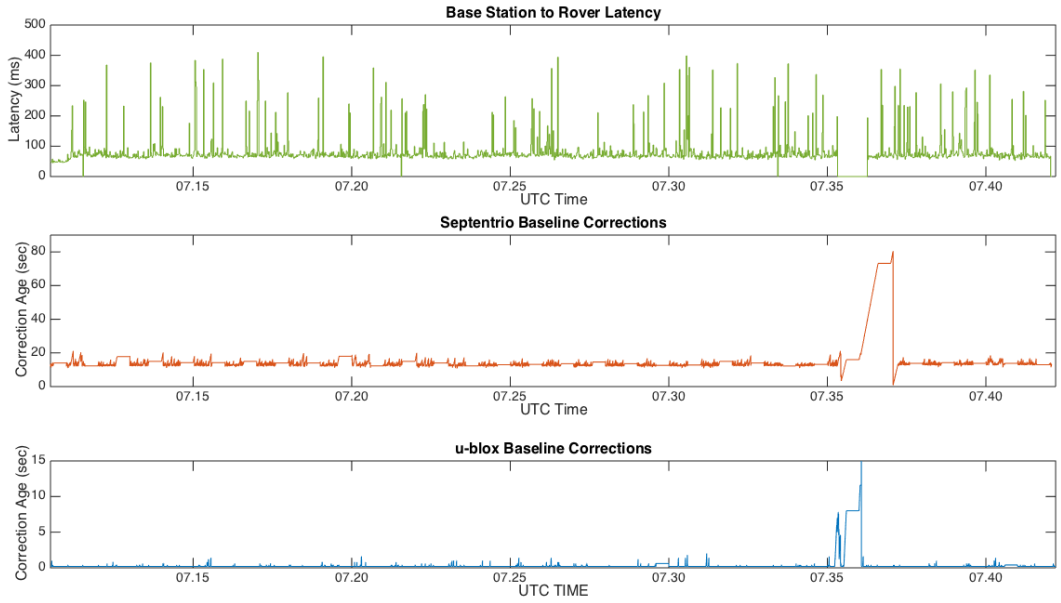


Figure 38: Latency between the Base Stations and the Rover, RTK Correction Age

supposed that this is its normal operation method. The average value for the correction age in the Septentrio solution is 14.490 seconds.

There is also a loss of network connection for 63 seconds, which corresponds to the problematic area where the u-blox GNSS solution had problems previously. This miscommunication with the base station wasn't a unique case, as the route was performed several times and in all of them the same thing happened in the area seen in Figure 39.

To try to figure out why the RTK solution from the u-blox wasn't being able to get past the Float solution, the base station and the rover were logged to RINEX files and then Post Processed. Figure 40 shows us the satellites being observed by both receivers, where the yellow lines represent the satellites that can be used to process the solution, and the grey lines satellites that were rejected because their position in the sky make a low angle to the receivers position.

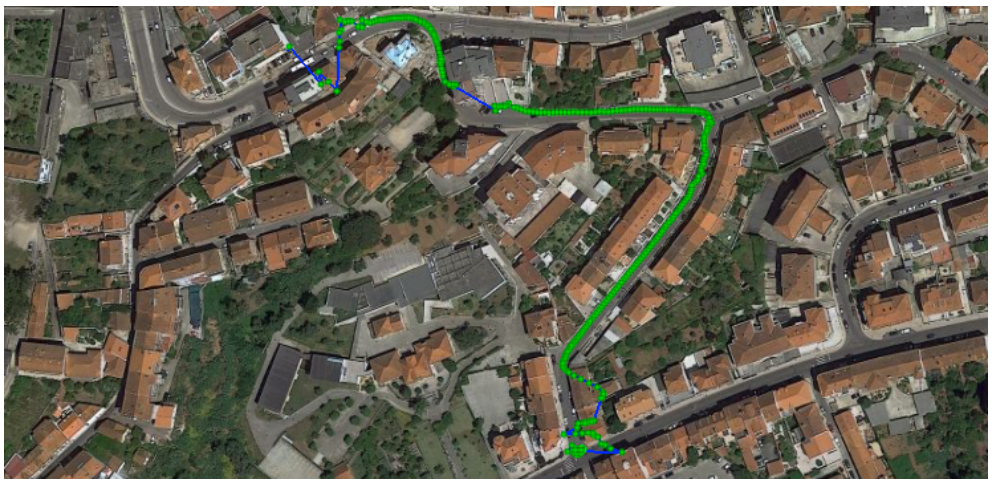


Figure 39: Detailed view from part of the route - 5 - RTK u-blox

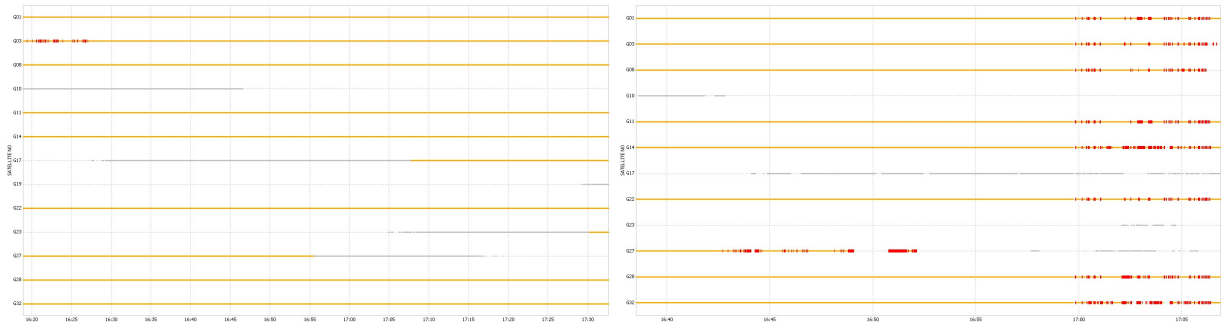


Figure 40: u-blox Satellite Cycle-Slips, Base Station (left) and Rover (right)

The most important part in Figure 40 are the small red strokes overlapped in the yellow lines. These strokes are the Loss of Lock Indicator (LLI) flag and represents a cycle slip. The cycle slips occur when the receiver loses the lock with a satellite and therefore loses count of the number of carrier phase cycles. Once a cycle slip occurs, the receiver has lost track of the cycle count and can not recover it, meaning that the Kalman filter state in RTKLIB that is estimating the cycle count for that satellite must be reset and reconverge again, which can take easily 30 seconds.

The cycle slips are usually caused by obstructions, reflections, high accelerations, vibration, Electromagnetic Interference (EMI), lack of antenna ground planes, or anything else that degrades the quality of the satellite signal. Figure 41 shows the measurement residual which are the difference between the expected measurement and the observed measurement, and also the estimated change in the rover position. The Post Processing was able to perform a fix solution (Green lines), but as the rover starts to see cycle slips on every satellite for every sample, the satellite phase biases are reset and the solution has to converge again causing it to always stay on float (Yellow lines).

The interesting part about these figures is that they show that the cycle slips start 38 seconds before the rover starts moving. The cause of this problem was not found, but as the problem starts before the rover starts moving its likely to be caused by a software problem or by an EMI caused by something inside the car.

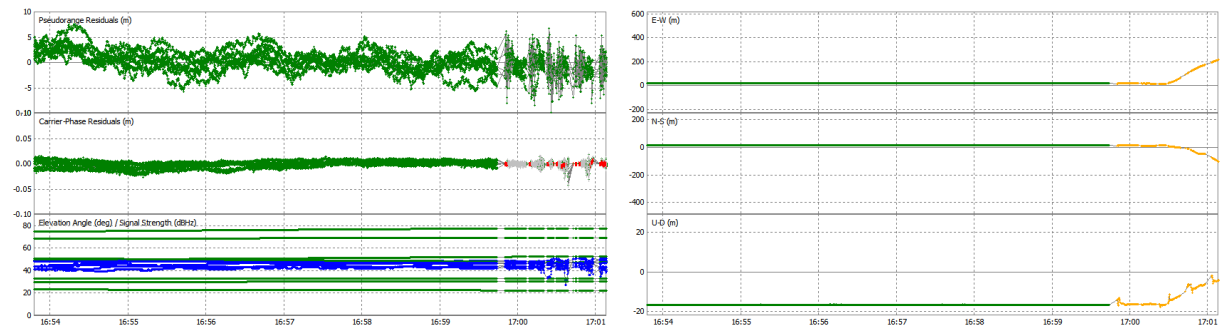


Figure 41: u-blox Residuals (left) and Position Estimation (right)

Low Speed Test

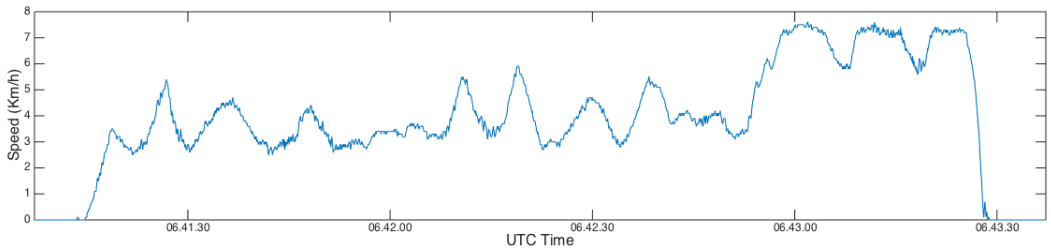


Figure 42: RTK Low Speed Test - Velocity reported by the Septentrio AsteRx4 Receiver

The RTK low speed test is quite similar to the one from the previous section, the path has 180 meters and consists mainly in 90 degrees turns, with the velocity shown in Figure 42.

The trajectory started in the right side at the same place as before, and the light blue path is from the u-blox and the green from the Septentrio (See Figure 43. This time the Septentrio starting position is actually being reported in the right place, the middle of the white car. The u-blox has a little error but we have to take into account that the receiver it is still not working properly, because the test was done with the fix/float problem still happening.

The accuracy of the solution can be seen by the Septentrio estimations that always go through the middle of the parking lines, as opposed to the u-blox that tends to keep its error. The 1 sigma error is shown in Figure 44.

Figure 43: RTK Low Speed Test - Satellite Image

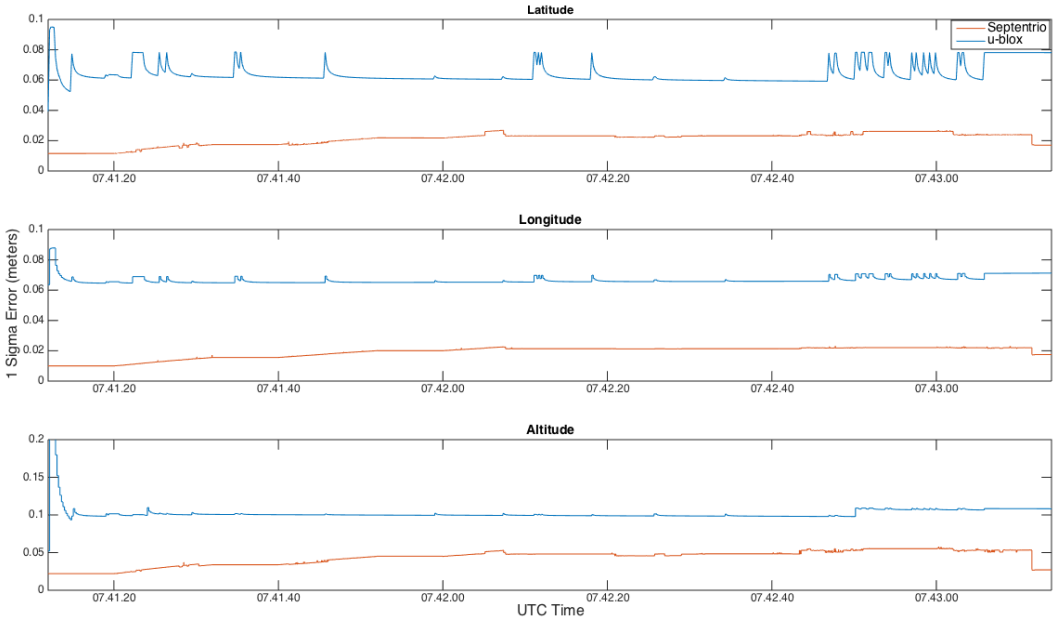


Figure 44: RTK Low Speed Test - 1 Sigma Error

5. Conclusion and Future Work

After a hard and long work, it is interesting to look back and compare the initial goals with what has been achieved. The main objective was fulfilled as we have a working localization method using RTK-GNSS/INS integration. At the same time, the project allowed to understand and have deeper knowledge on how the devices work and what to expect when using them.

It is possible to implement an INS to correct the GNSS positioning using low cost IMUs, however their quality will have a big impact on the time it takes for the solution to diverge. Higher quality IMUs allows the estimated solution to be accurate during more time without GNSS signals as they are composed by better sensor with lower noises and better bias stability. Nevertheless, low cost IMUs with using proper calibration methods and with good implementations can archive interesting results.

The developed INS solution with the two InvenSense MPU-9150 was able to yield a good solution for a couple of minutes before diverging and become unusable. In order to obtain such results its sensors had to be strictly calibrated and tested multiple times.

The u-blox NEO-6P used during the dissertation as able to obtain good results when measuring the carrier phases observables and working as a Standard Positioning Service (SPS) receiver, however it was not the best choice for Real Time Kinematic (RTK) solutions as it can only track GPS satellites. The Septentrio AsteRx4 proved to be a really good receiver in DGPS and RTK, and if coupled with good quality antennas it can reduce most of the expected errors. The GNSS implementation showed that even when using the same receiver, the type of observables being measured is really important as it completely changes the accuracy of the solution. Therefore, when using only one receiver to estimated the position of field robots, it should be chosen one receiver that measures carrier phases. If the robot is using RTK the receivers should be able to receive signals from multiple constellations.

The loosely coupled implementation used during the dissertation obtained good results, but there is still room for further improvements. As the system model implemented didn't correspond to the used vehicle, the solution had some errors that could have been eliminated if using a more restricted model.

Finally, the Google Maps proved to be a good tool to analize the accuracy of these solutions, as most of the times it was able to display positions which reflected the real world location.

As these kind of works are never finished and there is always something more that can be made and improved, here are some suggestions.

During the dissertation was bought an higher quality IMU, the Analog Devices ADIS16488A with the aim of implementing an INS solution more stable. As the time as short, it ended up never being implemented and tested.

Another interesting device to test is a low cost GNSS receiver capable of receiving multiple satellite constellations. It was proven that the number of satellites being tracked is very important for the accuracy and stability of the solution, so testing other receivers is always interesting.

The data fusion can also be improved and the suggestion here is to implement the an Ackerman model with other sensors in the fusion algorithm. The car speed is easily accessible throw the OBD-II port, and the Steering Angle Sensor (SAS) through the Controller Area Network (CAN) bus, so they are good sensors to integrate in the fusion algorithm.

6. Appendixes

Appendix A: Connections Diagram

Rover

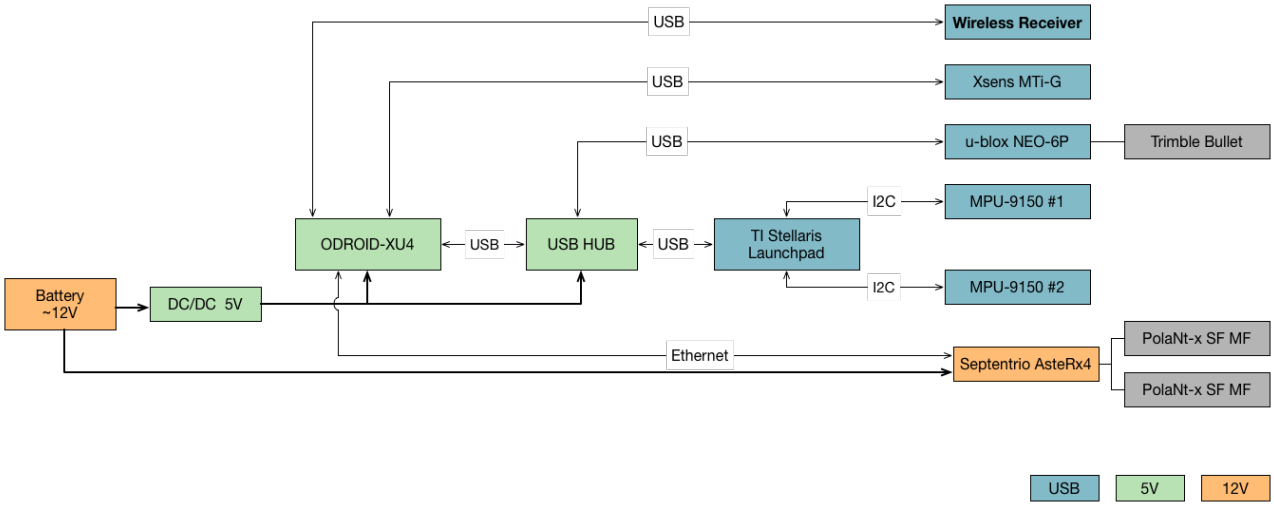


Figure 45: Rover Connections Diagram

Base Station

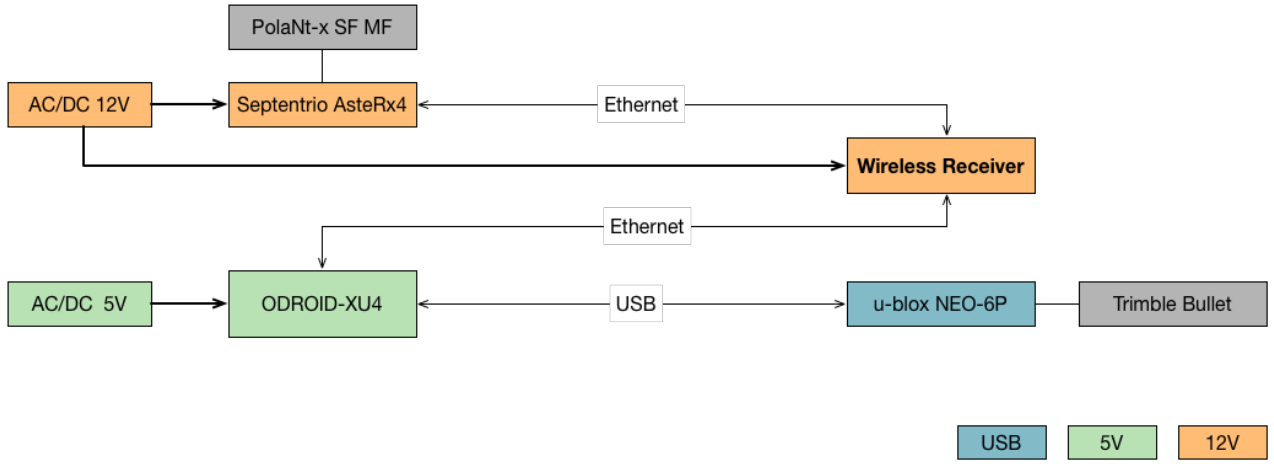


Figure 46: Base Station Connections Diagram

Appendix B: Allan Variance

The Allan Variance was developed by David Allan in 1966, and is a time domain analysis technique originally designed to characterise the noise and stability of clock systems. The technique has been adopted by the inertial sensor community for its simplicity when compared to others techniques like the Power Spectral Density (PSD) or the autocorrelation function. It is also recommended by the Institute of Electrical and Electronics Engineers (IEEE) as a standard to analyse accelerometers and gyroscopes signals [IEEE AES, 1997].

The method can be applied to any signal in order to determine the character of underlying noise processes, as they cause slopes with different gradients to appear on the plot. Furthermore, the different processes usually appear in different regions, allowing them to be easily identified (See Figure 48). A more detailed explanation and implementation of the Allan Variance method can be found at [El-Sheimy *et al.*, 2008].

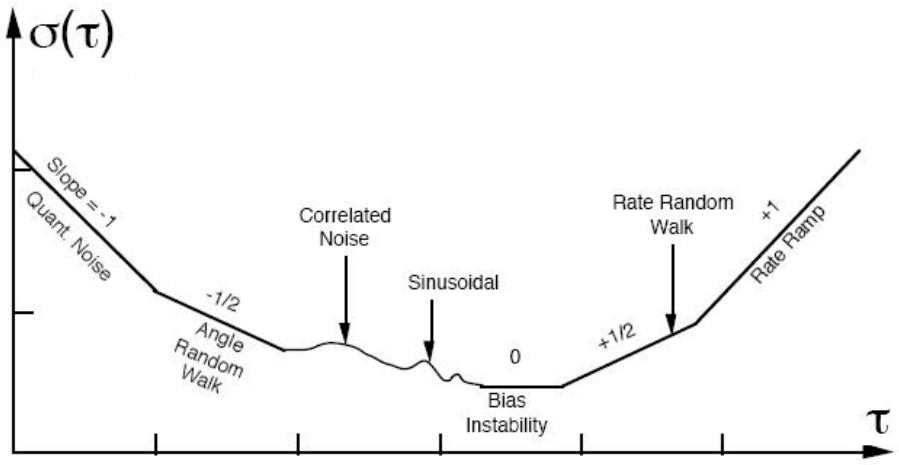


Figure 47: Allan Variance Plot [IEEE AES, 1997]

Quantization Noise

The quantization noise is a type of error introduced into the signal that results from inaccuracies in the Analog-to-Digital Converter (ADC), and it is caused by the small differences between the real amplitude of the point being sampled and the bits resolution of the ADC. It is represented by a slope of -1 in the log-log plot, and the magnitude can be read at $T = \sqrt{3}$.

Angle/Velocity Random Walk

The high frequency noise present in all electronic devices has a correlation time much shorter than the sampling time, which leads to the angle and velocity random walk. These errors are all

characterised by a white noise sequence on the signal output, and can be a substantial source for errors in the output signal. It is represented by a slope of -0.5 in the log-log plot, and the value can be obtained directly by reading the slope line at $T = 1$.

Bias Instability

The bias instability occurs due to the electronics being susceptible to random flickering noise as described previously. The flat region in the log-log plot can be used to estimate the limit of the bias instability.

Rate Random Walk

The rate random walk noise has an unclear origin, but it is likely a limiting case of an exponentially correlated noise with a very long correlation time. It is represented by a slope of $+0.5$ in the log-log plot, and the value can be obtained directly by reading the slope line at $T = 3$.

Drift Rate Ramp

The previously explained errors are all of random character, however it is also useful to determine the response of $\sigma(T)$ under deterministic errors. The drift rate ramp noise has a slope of $+1$ in the log-log plot and the amplitude can be obtained from the slope line at $T = \sqrt{2}$.

Implementation and Results

To plot the Allan Variance the inertial sensors were placed on a flat surface and left running during 10 hours without any external environmental disturbance. The sensors output were sampled at 100Hz and the measurements recorded to a file, which was later processed in Matlab. The log–log plot of the Allan standard deviation versus time can be seen in Figure 48 and 49.

During the logging period the temperature change was about 0.5°C, which being such a small value can be neglected. From the MPU-9150 datasheet, a change in 1°C in the sensors temperature will change the accelerometer output 0.02%, and about 0.04% for the gyroscope.

Tables 9 and 10 show the accelerometer errors for both Invensense MPU-9150 where it is easy to notice that the velocity random walk is the dominant error. As the sensors have a 16 bits ADCs the quantization noise isn't present in these IMUs. This is because as the number of bits increase the quantization error starts to have less influence. And as expected, the values for the Z-axis are almost double than the X and Y-Axis. Table 11 shows the same parameters for the Xsens MTi-G,

and as before the velocity random walk is the dominant error. The Xsens also has a 16 bits ADCs therefore the quantization noise it is also not present.

Table 9: Allan Variance - Accelerometer Data from MPU #1

MPU-9150 #1 - ACCELEROMETER PARAMETERS SPECIFICATION				
Acc.	Quantization Noise m/s	Velocity Random Walk $m/s/\sqrt{h}$	Bias Instability m/s^2	Rate Random Walk $m/s/s^2$
X	N/A	0.0866	2.304e-04	8.408e-04
Y	N/A	0.0874	2.651e-04	8.716e-04
Z	N/A	0.1174	3.491e-04	11.719e-04

Table 10: Allan Variance - Accelerometer Data from MPU #2

MPU-9150 #2 - ACCELEROMETER PARAMETERS SPECIFICATION				
Acc.	Quantization Noise m/s	Velocity Random Walk $m/s/\sqrt{h}$	Bias Instability m/s^2	Rate Random Walk $m/s/s^2$
X	N/A	0.0869	6.178e-04	8.557e-04
Y	N/A	0.0797	2.870e-04	7.634e-04
Z	N/A	0.1293	12.695e-04	12.695e-04

Table 11: Allan Variance - Accelerometer Data from Xsens MTi-G

Xsens MTi-G - ACCELEROMETER PARAMETERS SPECIFICATION				
Acc.	Quantization Noise m/s	Velocity Random Walk $m/s/\sqrt{h}$	Bias Instability m/s^2	Rate Random Walk $m/s/s^2$
X	N/A	0.0560	6.054e-04	7.737e-04
Y	N/A	0.0403	11.675e-04	5.231e-04
Z	N/A	0.0392	3.176e-04	4.706e-04

Reviewing the Tables 12 and 13 which summarise the gyroscope errors for both Invensense MPU-9150, yields a result very similar to the one experienced by the accelerometers, as the angle random walk is still the most dominant error. Table 14 shows us the same parameters for the Xsens MTi-G, and as before the angle random walk is the dominant error. In this case, the gyroscope inside the Xsens has a higher quality than the one used in the Invensense.

Table 12: Allan Variance - Gyroscope Data from MPU #1

MPU-9150 #1 - GYROSCOPE PARAMETERS SPECIFICATION				
Acc.	Quantization Noise $^\circ$	Angle Random Walk $^\circ/\sqrt{h}$	Bias Instability $^\circ/s$	Rate Random Walk $^\circ/s^2$
X	N/A	0.1903	7.610e-04	19.240e-04
Y	N/A	0.1847	8.402e-04	18.294e-04
Z	N/A	0.2106	7.248e-04	20.977e-04

Table 13: Allan Variance - Gyroscope Data from MPU #2

MPU-9150 #2 - GYROSCOPE PARAMETERS SPECIFICATION				
Acc.	Quantization Noise $^\circ$	Angle Random Walk $^\circ/\sqrt{h}$	Bias Instability $^\circ/s$	Rate Random Walk $^\circ/s^2$
X	N/A	0.2078	24.254e-04	20.279e-04
Y	N/A	0.2859	19.144e-04	28.643e-04
Z	N/A	0.2190	13.762e-04	23.248e-04

Table 14: Allan Variance - Gyroscope Data from Xsens MTi-G

Xsens MTi-G - GYROSCOPE PARAMETERS SPECIFICATION				
Acc.	Quantization Noise $^\circ$	Angle Random Walk $^\circ/\sqrt{h}$	Bias Instability $^\circ/s$	Rate Random Walk $^\circ/s^2$
X	N/A	0.0266	1.028e-04	2.752e-04
Y	N/A	0.0511	1.274e-04	7.284e-04
Z	N/A	0.0306	1.335e-04	3.053e-04

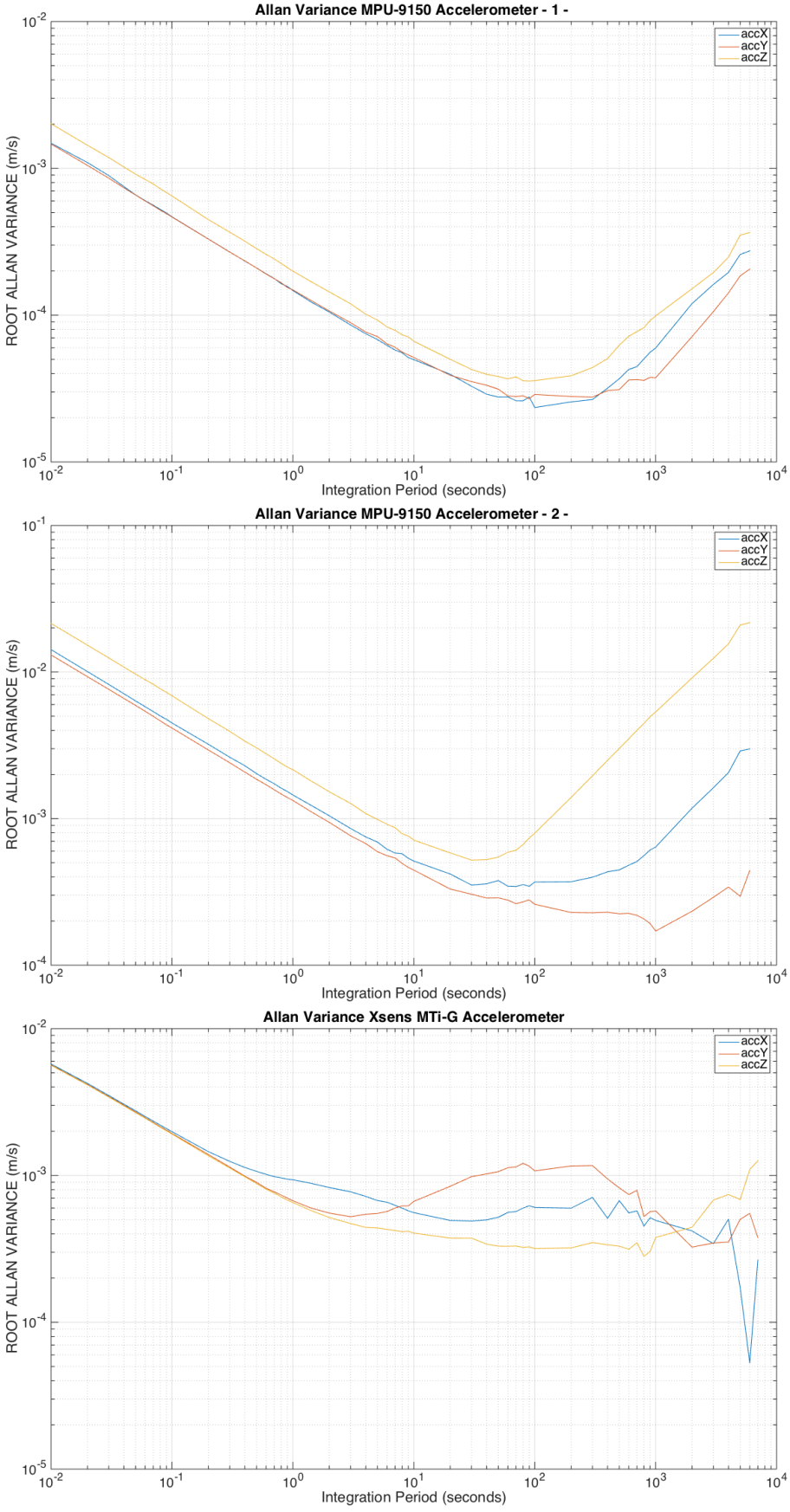


Figure 48: Allan Variance for Raw Accelerometer Data

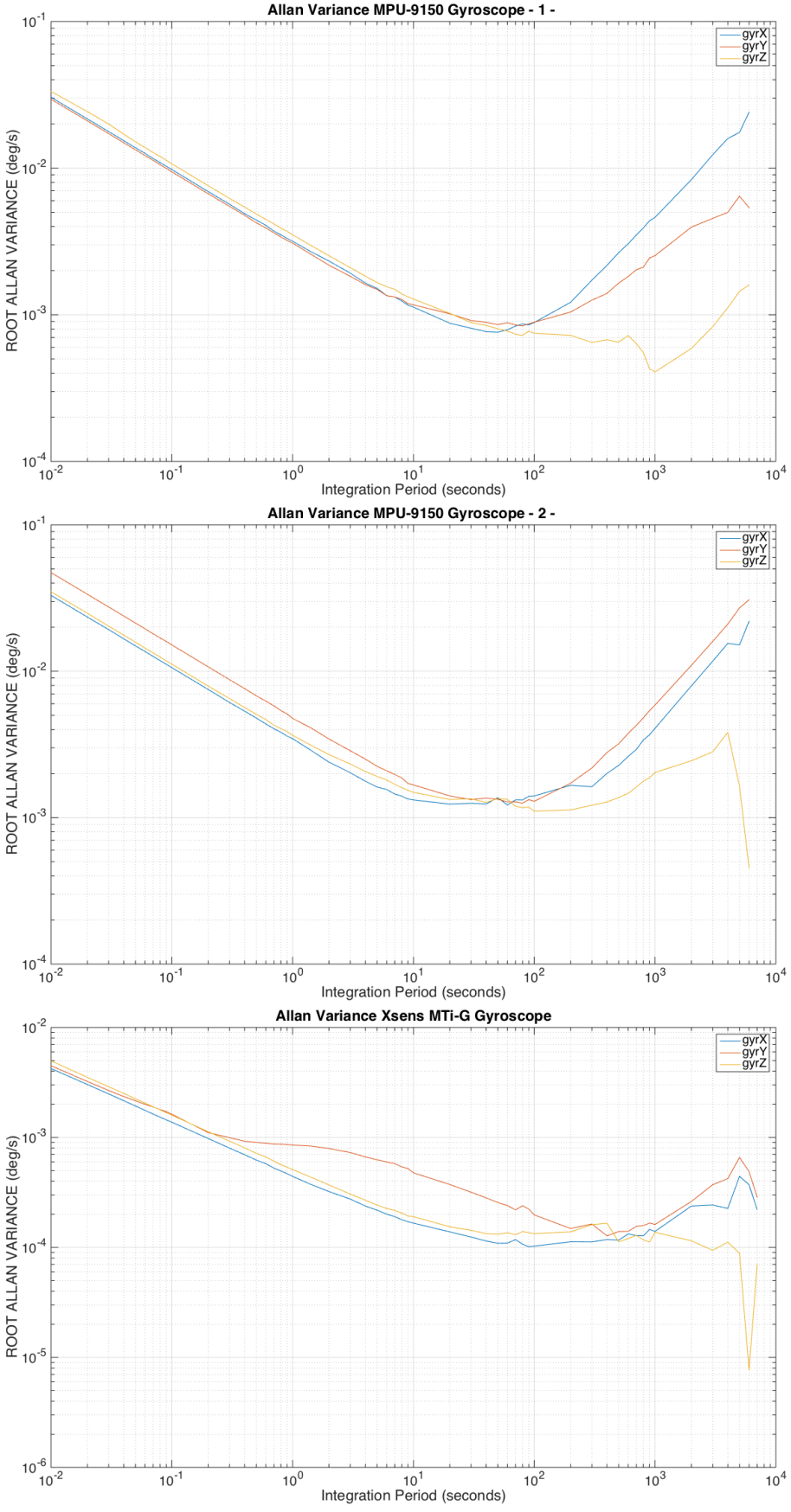


Figure 49: Allan Variance for Raw Gyroscope Data

Appendix C: Coordinate Frames and Earth Models

When using Satellite or Inertial Navigation Systems it's important to understand and know the differences between the coordinate frames that can be used to describe the position and attitude. These coordinate frames specify the reference points and axes which our measurements will use as their initial reference, and for that reason it's necessary to understand them and know which ones our devices use, so we don't have unexpected results in the data fusion.

Earth-Centered-Inertial (ECI) Frame has its origin in the earth's centre of mass, with the Z Axis passing through the north pole. The X Axis points to the sun and passes through the equatorial line. Finally, the Y Axis is found using the right hand rule. The ECI Frame it's mostly used for objects in space, as the motion equations that describe their orbital motion are easier in non rotating frames.

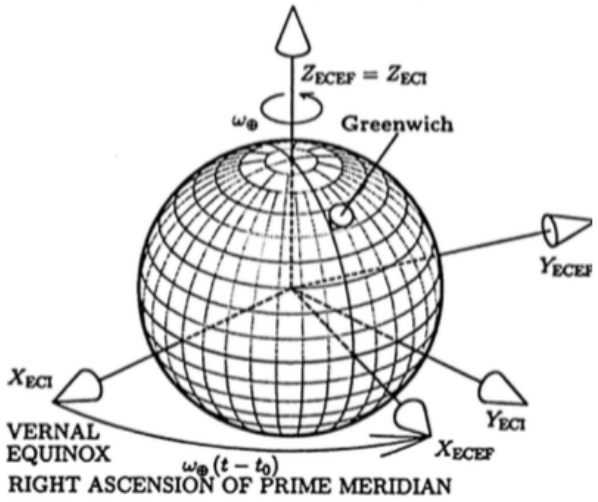


Figure 50: ECI and ECEF Coordinates Frames [Grewal *et al.*, 2007]

Earth-Centered-Earth-Fixed (ECEF) Frame also has its origin in the earth's centre of mass, with the Z Axis passing through the north pole. However, the X Axis is fixed to earth and intersects the Greenwich Meridian, with the Y Axis once again deduced by the right hand rule.

The ECEF Coordinates are usually expressed in the Geodetic Coordinates of Latitude, Longitude, and Altitude (LLA) or in the Universal Transverse Mercator (UTM).

The Geodetic Coordinates (LLA) is the most common one, and is measured in degrees for the Latitude and Longitude, and meters for the Altitude. The Latitudes values can change from 90° (North) to −90° (South) with the Equatorial line being represented by 0°. The Longitude values can go from 180° (East) to −180° (West), being 0° the prime meridian that passes through Greenwich. As for the Altitude, it's the estimated distance above the Earth's Ellipsoid defined by the World

Geodetic System Standard being used, which in this case is the WGS84.

The UTM Coordinates divides the world into 60 North-South zones, each one covering a strip of 6° wide in Longitude. Each zone is then divided into 20 horizontal bands of 8° wide in Latitude. This projection allows to represent the complete globe into a flat surface, where each zone/band combination is projected into a plane separately minimising the distortion.

More informations about the UTM Coordinates and conversions between then can be found at [DMATM 8358.2, 1989]. The ECEF Frame is used by GNSS Receivers with the LLA Coordinates.

Body and Sensor Frames, are frames fixed to the vehicle and used to represent navigation data such as velocity and orientation. The most commonly used systems are the North-East-Down (NED), the East-North-Up (ENU), and the North-West-Up (NWU). They are all right-handed systems where their origin could be defined anywhere, and their names identify the axes orientation. These local coordinate frames are the most basic ones and assume that the earth is flat, so they shouldn't be used over large distances.

The **NED Frame** is typically used for Air and Sea vehicles, where the X Axis points to the front of vehicle, the Y Axis to its right, and the Z Axis down.

The **ENU Frame** is mostly used for land vehicles, with the X Axis pointing to the front, the Y Axis to the left, and the Z Axis up.

The **NWU Frame** is similar to the ENU Frame, but rotated 90° degrees. It's worth mention this frame as some magnetometers report their data in this coordinates.

More information about Coordinate Frames and transformations between them can be found in [Crassidis, 2006].

Appendix D: System Dynamics Model Matrixes

$$\mathbf{F}_k = \begin{bmatrix} I & \mathbf{F_{12}} & \Phi_{13} & 0 & \Phi_{15} \\ 0 & F_{22} & 0 & \Phi_{24} & 0 \\ 0 & 0 & I & 0 & \Phi_{35} \\ 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & I \end{bmatrix}, \quad \mathbf{F_{12}} = \begin{bmatrix} F1_1 & F1_2 & F1_3 \\ F1_4 & F1_5 & F1_6 \\ F1_7 & F1_8 & F1_9 \end{bmatrix} \tag{68}$$

$$\begin{aligned}
F1_1 =& \Delta t\,(\dot{y}(\cos\phi\sin\theta\cos\psi+\sin\phi\sin\psi)+\dot{z}(-\sin\phi\sin\theta\cos\psi+\cos\phi\sin\psi)) \\
&+\frac{\Delta t^2}{2}(\ddot{y}(\cos\phi\sin\theta\cos\psi+\sin\phi\sin\psi)+\ddot{z}(-\sin\phi\sin\theta\cos\psi+\cos\phi\sin\psi)) \\
F1_2 =& \Delta t\,(\dot{x}(-\sin\theta\cos\psi)+\dot{y}(\sin\phi\cos\theta\cos\psi)+\dot{z}(\cos\phi\cos\theta\cos\psi)) \\
&+\frac{\Delta t^2}{2}(\ddot{x}(-\sin\theta\cos\psi)+\ddot{y}(\sin\phi\cos\theta\cos\psi)+\ddot{z}(\cos\phi\cos\theta\cos\psi)) \\
F1_3 =& \Delta t\,(\dot{x}(-\cos\theta\sin\psi)+\dot{y}(-\sin\phi\sin\theta\sin\psi-\cos\phi\cos\psi)+\dot{z}(-\cos\phi\sin\theta\sin\psi+\sin\phi\cos\psi)) \\
&+\frac{\Delta t^2}{2}(\ddot{x}(-\cos\theta\sin\psi)+\ddot{y}(-\sin\phi\sin\theta\sin\psi-\cos\phi\cos\psi)+\ddot{z}(-\cos\phi\sin\theta\sin\psi+\sin\phi\cos\psi)) \\
F1_4 =& \Delta t\,(\dot{y}(\cos\phi\sin\theta\sin\psi-\sin\phi\cos\psi)+\dot{z}(-\sin\phi\sin\theta\sin\psi-\cos\phi\cos\psi)) \\
&+\frac{\Delta t^2}{2}(\ddot{y}(\cos\phi\sin\theta\sin\psi-\sin\phi\cos\psi)+\ddot{z}(-\sin\phi\sin\theta\sin\psi-\cos\phi\cos\psi)) \\
F1_5 =& \Delta t\,(\dot{x}(-\sin\theta\sin\psi)+\dot{y}(\sin\phi\cos\theta\sin\psi)+\dot{z}(\cos\phi\cos\theta\sin\psi)) \\
&+\frac{\Delta t^2}{2}(\ddot{x}(-\sin\theta\sin\psi)+\ddot{y}(\sin\phi\cos\theta\sin\psi)+\ddot{z}(\cos\phi\cos\theta\sin\psi)) \\
F1_6 =& \Delta t\,(\dot{x}(\cos\theta\cos\psi)+\dot{y}(\sin\phi\sin\theta\cos\psi-\cos\phi\sin\psi)+\dot{z}(\cos\phi\sin\theta\cos\psi+\sin\phi\sin\psi)) \\
&+\frac{\Delta t^2}{2}(\ddot{x}(\cos\theta\cos\psi)+\ddot{y}(\sin\phi\sin\theta\cos\psi-\cos\phi\sin\psi)+\ddot{z}(\cos\phi\sin\theta\cos\psi+\sin\phi\sin\psi)) \\
F1_7 =& \Delta t\,(\dot{y}(\cos\phi\cos\theta)+\dot{z}(-\sin\phi\cos\theta)) \\
&+\frac{\Delta t^2}{2}(\ddot{y}(\cos\phi\cos\theta)+\ddot{z}(-\sin\phi\cos\theta)) \\
F1_8 =& \Delta t\,(\dot{x}(-\cos\theta)+\dot{y}(-\sin\phi\sin\theta)+\dot{z}(-\cos\phi\sin\theta)) \\
&+\frac{\Delta t^2}{2}(\ddot{x}(-\cos\theta)+\ddot{y}(-\sin\phi\sin\theta)+\ddot{z}(-\cos\phi\sin\theta)) \\
F1_9 =& 0
\end{aligned} \tag{69}$$

$$\mathbf{F}_k = \begin{bmatrix} I & F_{12} & \Phi_{13} & 0 & \Phi_{15} \\ 0 & \mathbf{F_{22}} & 0 & \Phi_{24} & 0 \\ 0 & 0 & I & 0 & \Phi_{35} \\ 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & I \end{bmatrix}, \quad \mathbf{F_{22}} = \begin{bmatrix} F2_1 & F2_2 & F2_3 \\ F2_4 & F2_5 & F2_6 \\ F2_7 & F2_8 & F2_9 \end{bmatrix} \tag{70}$$

$$\begin{aligned}
F2_1 =& \Delta t \left(\dot{\theta} \left(\cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi \right) + \dot{\psi} \left(-\sin\phi \sin\theta \cos\psi + \cos\phi \sin\psi \right) \right) + 1 \\
F2_2 =& \Delta t \left(\dot{\phi} \left(-\sin\theta \cos\psi \right) + \dot{\theta} \left(\sin\phi \cos\theta \cos\psi \right) + \dot{\psi} \left(\cos\phi \cos\theta \cos\psi \right) \right) \\
F2_3 =& \Delta t \left(\dot{\phi} \left(-\cos\theta \sin\psi \right) + \dot{\theta} \left(-\sin\phi \sin\theta \sin\psi - \cos\phi \cos\psi \right) + \dot{\psi} \left(-\cos\phi \sin\theta \sin\psi + \sin\phi \cos\psi \right) \right) \\
F2_4 =& \Delta t \left(\dot{\theta} \left(\cos\phi \sin\theta \sin\psi - \sin\phi \cos\psi \right) + \dot{\psi} \left(-\sin\phi \sin\theta \sin\psi - \cos\phi \cos\psi \right) \right) \\
F2_5 =& \Delta t \left(\dot{\phi} \left(-\sin\theta \sin\psi \right) + \dot{\theta} \left(\sin\phi \cos\theta \sin\psi \right) + \dot{\psi} \left(\cos\phi \cos\theta \sin\psi \right) \right) + 1 \\
F2_6 =& \Delta t \left(\dot{\phi} \left(\cos\theta \cos\psi \right) + \dot{\theta} \left(\sin\phi \sin\theta \cos\psi - \cos\phi \sin\psi \right) + \dot{\psi} \left(\cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi \right) \right) \\
F2_7 =& \Delta t \left(\dot{\theta} \left(\cos\phi \cos\theta \right) + \dot{\psi} \left(-\sin\phi \cos\theta \right) \right) \\
F2_8 =& \Delta t \left(\dot{\phi} \left(-\cos\theta \right) + \dot{\theta} \left(-\sin\phi \sin\theta \right) + \dot{\psi} \left(-\cos\phi \sin\theta \right) \right) \\
F2_9 =& 1
\end{aligned} \tag{71}$$

Appendix E: RTKLIB

To obtain the Real Time Kinematic solution was used the open source software RTKLIB. The software employs an Extended Kalman Filter (EKF) to perform the RTK solution and solve the ambiguities present in the code pseudorange and carrier phase models [Takasu and Yasuda, 2009]. The state vector $\mathbf{x}_k$ and the measurement vector $\mathbf{z}_k$, are represented by

$$\begin{aligned} \mathbf{x}_k &= [\mathbf{r}_r, \mathbf{v}_r, \mathbf{SD}_{L1}, \mathbf{SD}_{L2}, \mathbf{SD}_{L5}]^T \\ \mathbf{z}_k &= [\Phi_{L1}, \Phi_{L2}, \Phi_{L5}, \mathbf{P}_{L1}, \mathbf{P}_{L2}, \mathbf{P}_{L5}]^T \end{aligned} \tag{72}$$

where $\mathbf{r}_r$ and $\mathbf{v}_r$ are the rover antenna position and velocity in the ECEF frame, and $\mathbf{SD}_{Li}$ are the Single Difference carrier phase measurements. The measurement vector $\mathbf{z}_k$ is defined by the Souble Differences phase range and pseudorange measurements from the epoch t_k. Using the EKF Predict Step, the estimated state vector $\widehat{\mathbf{x}}$ and its covariance matrix $\widehat{\mathbf{P}}$ are found as

——— Extended Kalman Filter - Prediction Step ———

$$\widehat{\mathbf{x}}_{k+1} = \mathbf{F}_k \widehat{\mathbf{x}}_k \tag{73}$$

$$\widehat{\mathbf{P}}_{k+1} = \mathbf{F}_k \mathbf{P}_k \mathbf{F}_k^T + \mathbf{Q}_k \tag{74}$$

---

where $\mathbf{F}$ and $\mathbf{Q}$ are the transition matrix and the covariance matrix of the system noise from the epoch t_k to t_{k+1}. The EKF Correction Step can then be found using the following equations

——— Extended Kalman Filter - Correction Step ———

$$\mathbf{K}_k = \widehat{\mathbf{P}}_k \mathbf{H}_k^T \left(\mathbf{H}_k \widehat{\mathbf{P}}_k \mathbf{H}_k^T + \mathbf{R}_k \right)^{-1} \tag{75}$$

$$\mathbf{x}_k = \widehat{\mathbf{x}}_k + \mathbf{K}_k \left(\mathbf{z}_k - \mathbf{h}(\widehat{\mathbf{x}}_k) \right) \tag{76}$$

$$\mathbf{P}_k = \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k \right) \hat{\mathbf{P}}_k \tag{77}$$

---

where $\mathbf{h}$ represents the measurements model vector, $\mathbf{H}_k$ the matrix of partial derivatives, and $\mathbf{R}_k$ the covariance matrix of the partial errors. A more complete description of the EKF implementation can be found in the RTKLIB Manual [Takasu and Yasuda, 2009].

Solving the EKF equations with RTK-GPS Model (Equations 17 and 18) gives us the estimated rover position with the Single Differences carrier phase ambiguities. The next step to improve the accuracy of the rover position is to solve the float carrier phase ambiguities into integer values, and

implement the Double Differences. The floating solution obtained for the rover position with the Single Differences, is then transformed to Double Differences with

$$\begin{aligned}
\mathbf{G} &= \begin{pmatrix} \mathbf{I}_6 & & & \\ & \mathbf{D} & & \\ & & \mathbf{D} & \\ & & & \mathbf{D} \end{pmatrix} \\
\hat{\mathbf{x}}'_k &= \mathbf{G}\mathbf{x}_k = \left(\hat{\mathbf{r}}_r, \hat{\mathbf{v}}_r, \hat{\mathbf{N}}\right)^T \\
\mathbf{P}'_k &= \mathbf{G}\mathbf{P}_k\mathbf{G}^T = \begin{pmatrix} \mathbf{Q}_R & \mathbf{Q}_{NR} \\ \mathbf{Q}_{RN} & \mathbf{Q}_N \end{pmatrix}
\end{aligned} \tag{78}$$

being $\mathbf{G}$ the SD to DD transformation matrix, where the Single Differences carrier phase biases are transferred to Double Differences carrier phase form in order to eliminate the receiver initial phase terms $\phi^{sz}_{br,0}$, to obtain integer ambiguities $\hat{\mathbf{N}}$, and their covariance $\mathbf{Q}_N$. The most appropriate integer vector $\check{\mathbf{N}}$ for integer ambiguities is then found by solving the Integer Least Square (ILS) problem

$$\check{\mathbf{N}} = \min_{N \in Z} \left(\left(\mathbf{N} - \hat{\mathbf{N}}\right)^T \mathbf{Q}_N^{-1} \left(\mathbf{N} - \hat{\mathbf{N}}\right) \right) \tag{79}$$

To solve this problem in real time, the RTKLIB software uses a modified LAMBDA method, MLAMBDA [Teunissen, 1994]. The integer vector solution is then validated by using a simple ratio test defined as the weighted sum of the squared residuals by the second best solution $\check{\mathbf{N}}_2$ to the best $\check{\mathbf{N}}$, and is used to check the reliability of the solution

$$\mathbf{R} = \frac{\left(\check{\mathbf{N}}_2 - \hat{\mathbf{N}}\right)^T \mathbf{Q}_N^{-1} \left(\check{\mathbf{N}}_2 - \hat{\mathbf{N}}\right)}{\left(\check{\mathbf{N}} - \hat{\mathbf{N}}\right)^T \mathbf{Q}_N^{-1} \left(\check{\mathbf{N}} - \hat{\mathbf{N}}\right)} > \mathbf{R}_{thres} \tag{80}$$

If the solution ratio is higher than the defined threshold, the new position and velocity is finally obtained with

$$\begin{pmatrix} \check{\mathbf{r}} \\ \check{\mathbf{v}} \end{pmatrix} = \begin{pmatrix} \hat{\mathbf{r}} \\ \hat{\mathbf{v}} \end{pmatrix} - \mathbf{Q}_{RN}\mathbf{Q}_N^{-1}(\hat{\mathbf{N}} - \check{\mathbf{N}}) \tag{81}$$

and defined as a "FIXED" Solution. If the solution fails the ratio test, it is instead used the values of $\hat{\mathbf{r}}$ and $\hat{\mathbf{v}}$, and solution it is called "FLOAT".

When the receiver is locked to a satellite, the ambiguity that represents the unknown number of full cycles between the satellite and the receiver is constant. However, when a discontinuity in the continuous phase lock on the satellite signal occurs (from a power loss, a low SNR, severe ionospheric conditions, or the obstructions from buildings or trees), a cycle slip occurs in the RTK Solution. Once the signal from the satellite is re-established, a new ambiguity will exist and will have to be solved separately from the original ambiguity. Code Pseudorange measurements are

immune to cycle slips, but Carrier Phase accuracy rapidly degrades if cycle slips occurs. One of the advantages of using Post-Processing techniques is the ability to eliminate these Cycle Slips.

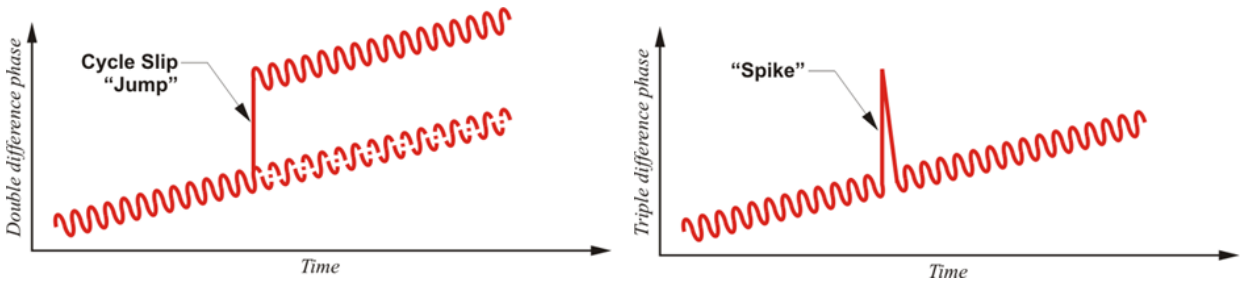


Figure 51: GNSS Cycle Slip [Sickle, 2014]

Appendix F: u-blox Base Station 24 Hours Post Processing Results

CSRS-PPP (V 1.05 34613)

base.obs

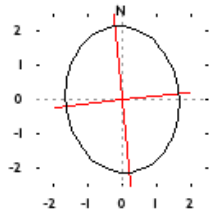
Data Start	Data End	Duration of Observations
2016-06-20 14:58:20.200	2016-06-21 15:04:26.900	24h 6m 6.70s
	Apri / Aposteriori Code Std	
	2.0m / 1.568m	
Observations	**Frequency**	**Mode**
Code	L1	Static
Elevation Cut-Off	**Rejected Epochs**	**Observation & Estimation Steps**
10.000 degrees	0.25 %	0.10 sec / 0.10 sec
Antenna Model	**APC to ARP**	**ARP to Marker**
	Ant. not in PPP (0 m)	0.000 m

(APC = antenna phase center; ARP = antenna reference point)

Estimated Position for base.obs

	Latitude (+n)	Longitude (+e)	Ell. Height
ITRF08 (2016)	40º 11’ 33.0074’’	-8º 25’ 04.3869’’	100.204 m
Sigmas(95%)	0.017 m	0.013 m	0.034 m
Apriori	40º 11’ 32.827’’	-8º 25’ 04.324’’	112.253 m
Estimated - Apriori	5.575 m	-1.499 m	-12.049 m

95% Error Ellipse (cm)
semi-major: 2.140cm
semi-minor: 1.655cm
semi-major azimuth: -5º 52’ 59.01’’



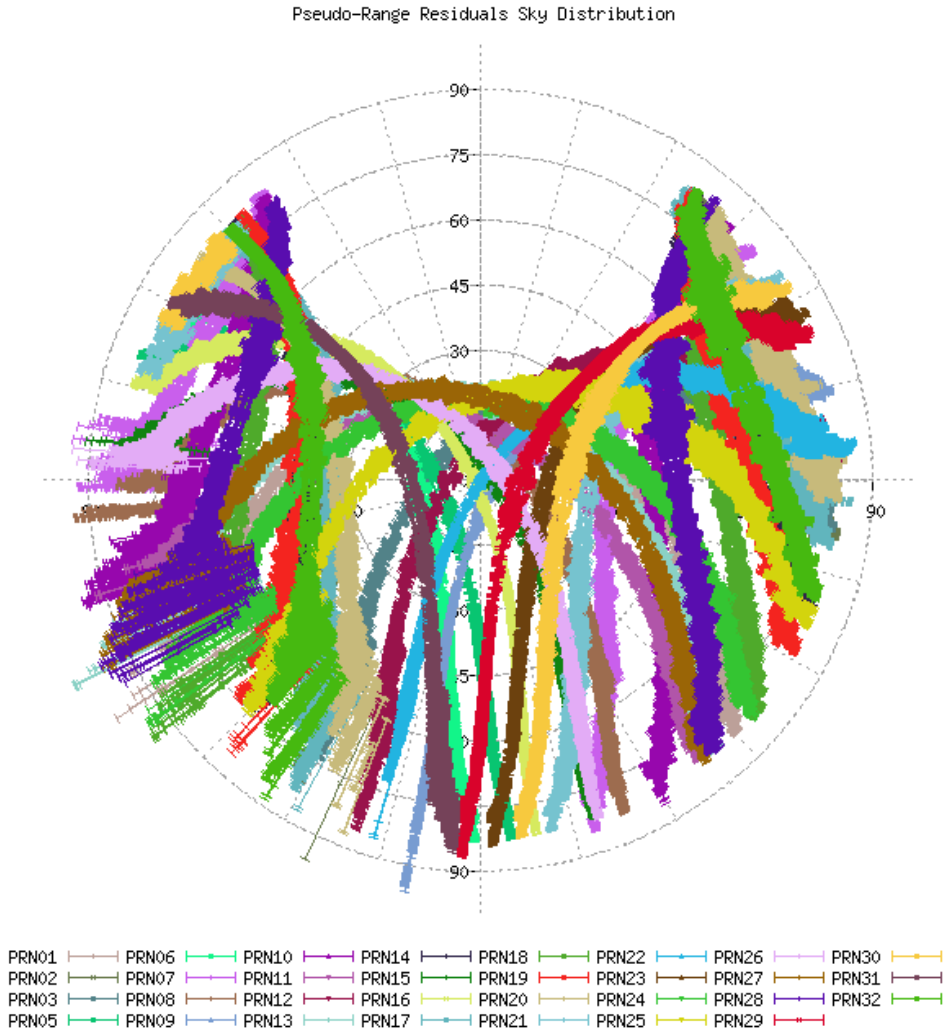
UTM (North) Zone 29

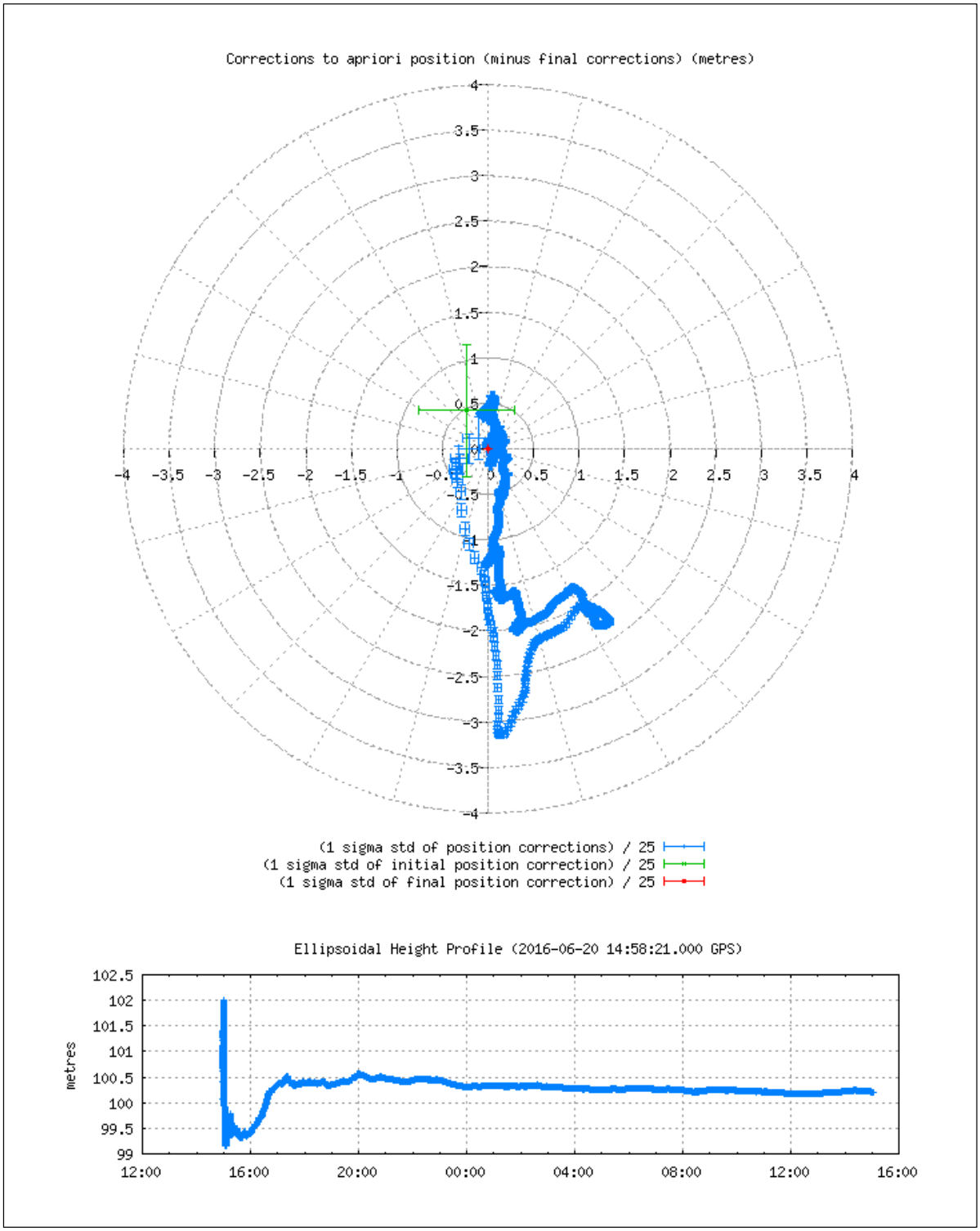
4449285.864m (N) 549549.471m (E)

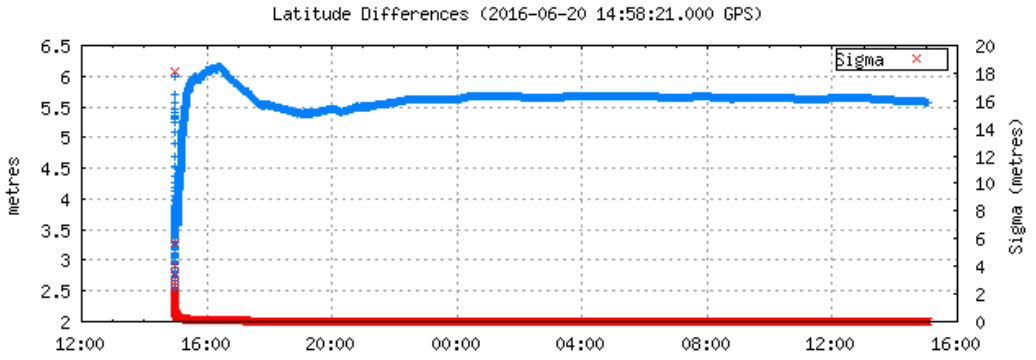
Scale Factors
0.99963022 (point)
0.99961450 (combined)

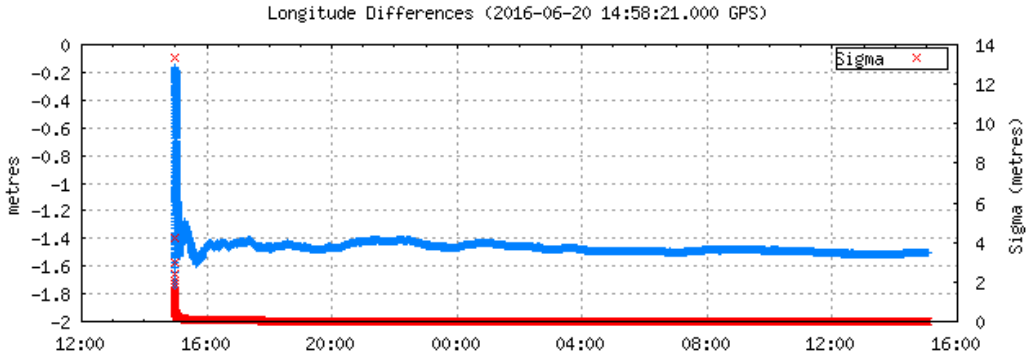
(Coordinates from RINEX file used as apriori position)

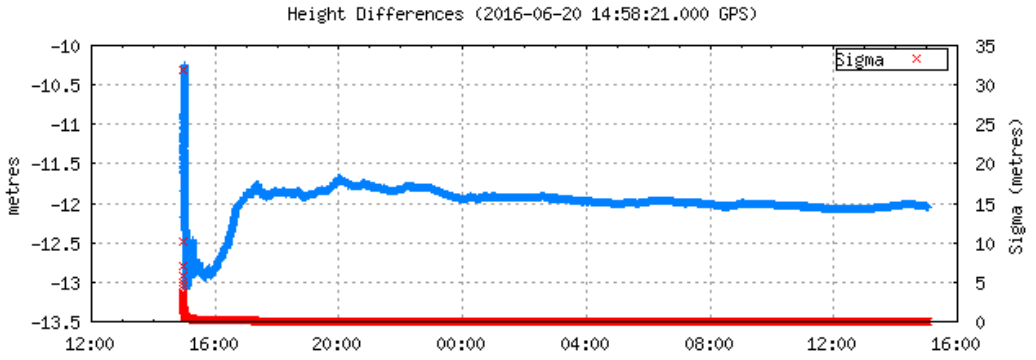
Estimated Parameters & Observations Statistics

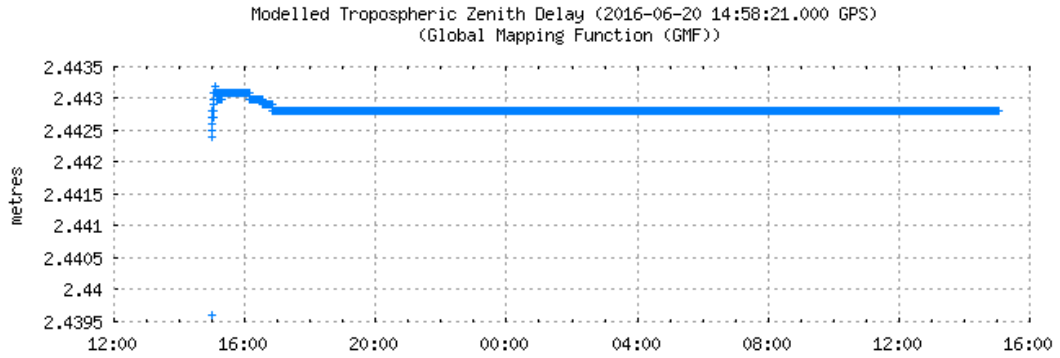


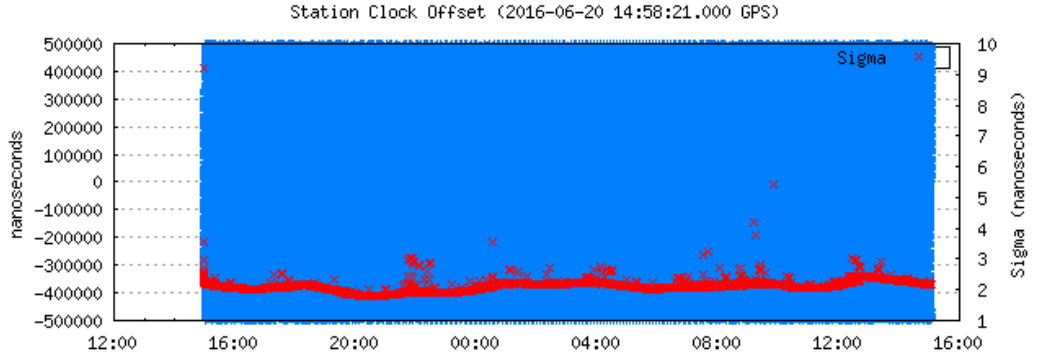

Corrections to apriori position (minus final corrections) (metres)
(1 sigma std of position corrections) / 25
(1 sigma std of initial position correction) / 25
(1 sigma std of final position correction) / 25
Ellipsoidal Height Profile (2016-06-20 14:58:21.000 GPS)
metres
12:00
16:00
20:00
00:00
04:00
08:00
12:00
16:00

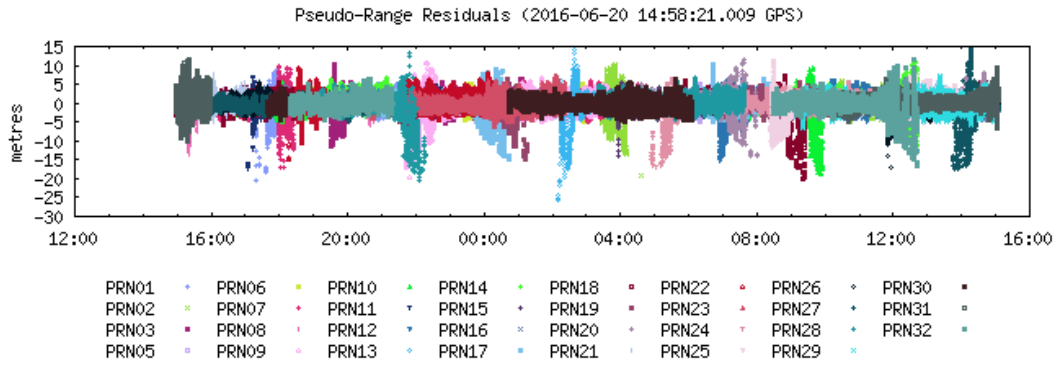

Latitude Differences (2016-06-20 14:58:21.000 GPS)
Sigma
metres
Sigma (metres)
6.5
6
5.5
5
4.5
4
3.5
3
2.5
2
20
18
16
14
12
10
8
6
4
2
0
12:00
16:00
20:00
00:00
04:00
08:00
12:00
16:00


Longitude Differences (2016-06-20 14:58:21.000 GPS)
Sigma
metres
Sigma (metres)
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-1.4
-1.6
-1.8
-2
14
12
10
8
6
4
2
0
12:00
16:00
20:00
00:00
04:00
08:00
12:00
16:00


Height Differences (2016-06-20 14:58:21.000 GPS)
Sigma
metres
Sigma (metres)
-10
-10.5
-11
-11.5
-12
-12.5
-13
-13.5
35
30
25
20
15
10
5
0
12:00
16:00
20:00
00:00
04:00
08:00
12:00
16:00


Modelled Tropospheric Zenith Delay (2016-06-20 14:58:21.000 GPS)
(Global Mapping Function (GMF))
metres
2.4435
2.443
2.4425
2.442
2.4415
2.441
2.4405
2.44
2.4395
12:00
16:00
20:00
00:00
04:00
08:00
12:00
16:00


Station Clock Offset (2016-06-20 14:58:21.000 GPS)
nanoseconds
Sigma (nanoseconds)
Sigma
500000
400000
300000
200000
100000
0
-100000
-200000
-300000
-400000
-500000
10
9
8
7
6
5
4
3
2
1
12:00
16:00
20:00
00:00
04:00
08:00
12:00
16:00


Pseudo-Range Residuals (2016-06-20 14:58:21.009 GPS)
metres
15
10
5
0
-5
-10
-15
-20
-25
-30
12:00
16:00
20:00
00:00
04:00
08:00
12:00
16:00
PRN01
PRN02
PRN03
PRN05
PRN06
PRN07
PRN08
PRN09
PRN10
PRN11
PRN12
PRN13
PRN14
PRN15
PRN16
PRN17
PRN18
PRN19
PRN20
PRN21
PRN22
PRN23
PRN24
PRN25
PRN26
PRN27
PRN28
PRN29
PRN30
PRN31
PRN32

Appendix G: Septentrio Base Station 24 Hours Post Processing Results

CSRS-PPP (V 1.05 34613)

SEPT

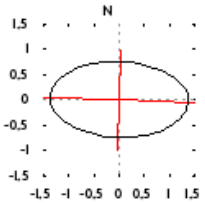
Data Start	Data End	Duration of Observations
2016-06-20 15:05:31.000	2016-06-21 15:11:29.000	24h 5m 58.00s
Apri / Aposteriori Phase Std		**Apri / Aposteriori Code Std**
0.015m / 0.012m		2.0m / 1.575m
Observations	**Frequency**	**Mode**
Phase and Code	L1 and L2	Static
Elevation Cut-Off	**Rejected Epochs**	**Observation & Estimation Steps**
10.000 degrees	0.00 %	1.00 sec / 1.00 sec
Antenna Model	**APC to ARP**	**ARP to Marker**
SEPPOLANT_X_MF NONE	L1= 0.058 m L2= 0.066 m	0.000 m

(APC = antenna phase center; ARP = antenna reference point)

Estimated Position for sept.16o

	Latitude (+n)	Longitude (+e)	Ell. Height
ITRF08 (2016)	40º 11’ 32.9839’’	-8º 25’ 04.4044’’	100.609 m
Sigmas(95%)	0.001 m	0.001 m	0.003 m
Apriori	40º 11’ 33.033’’	-8º 25’ 04.395’’	100.091 m
Estimated - Apriori	-1.516 m	-0.229 m	0.518 m

95% Error Ellipse (mm)
semi-major: 1.374mm
semi-minor: 0.748mm
semi-major azimuth: 91º 47’ 25.98’’



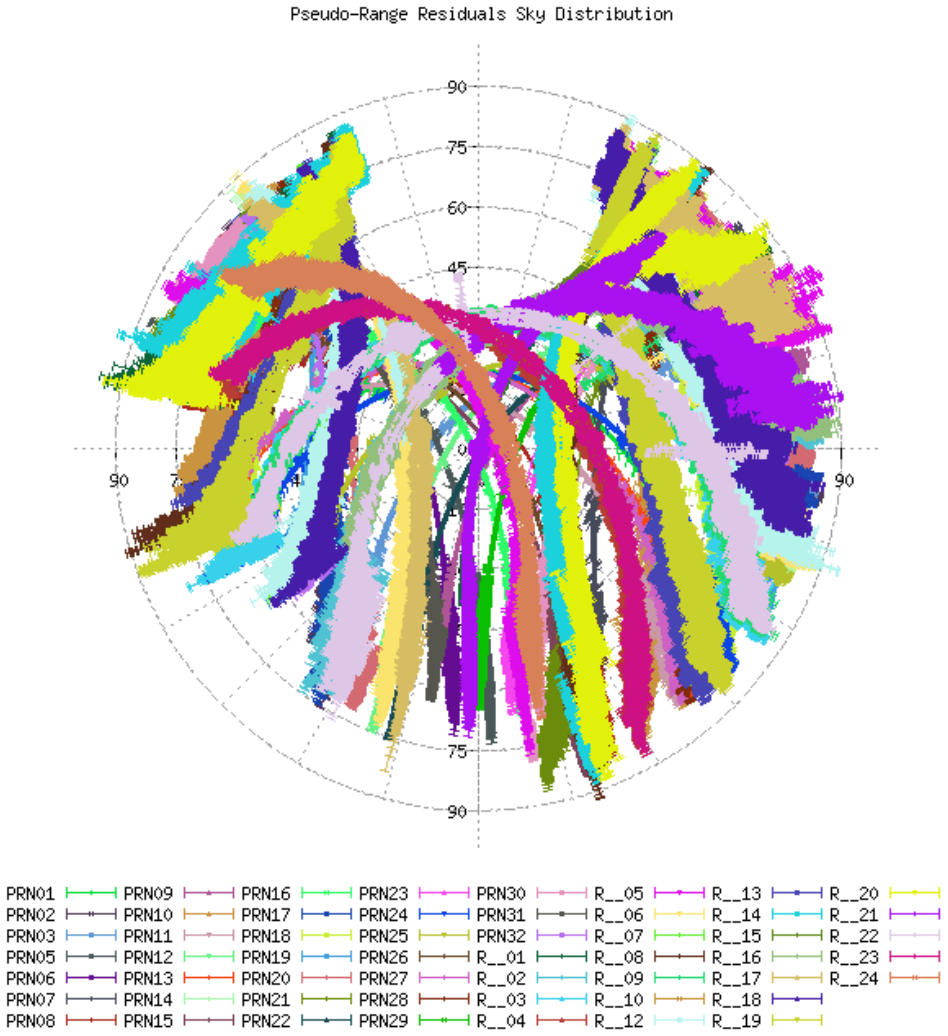
UTM (North) Zone 29

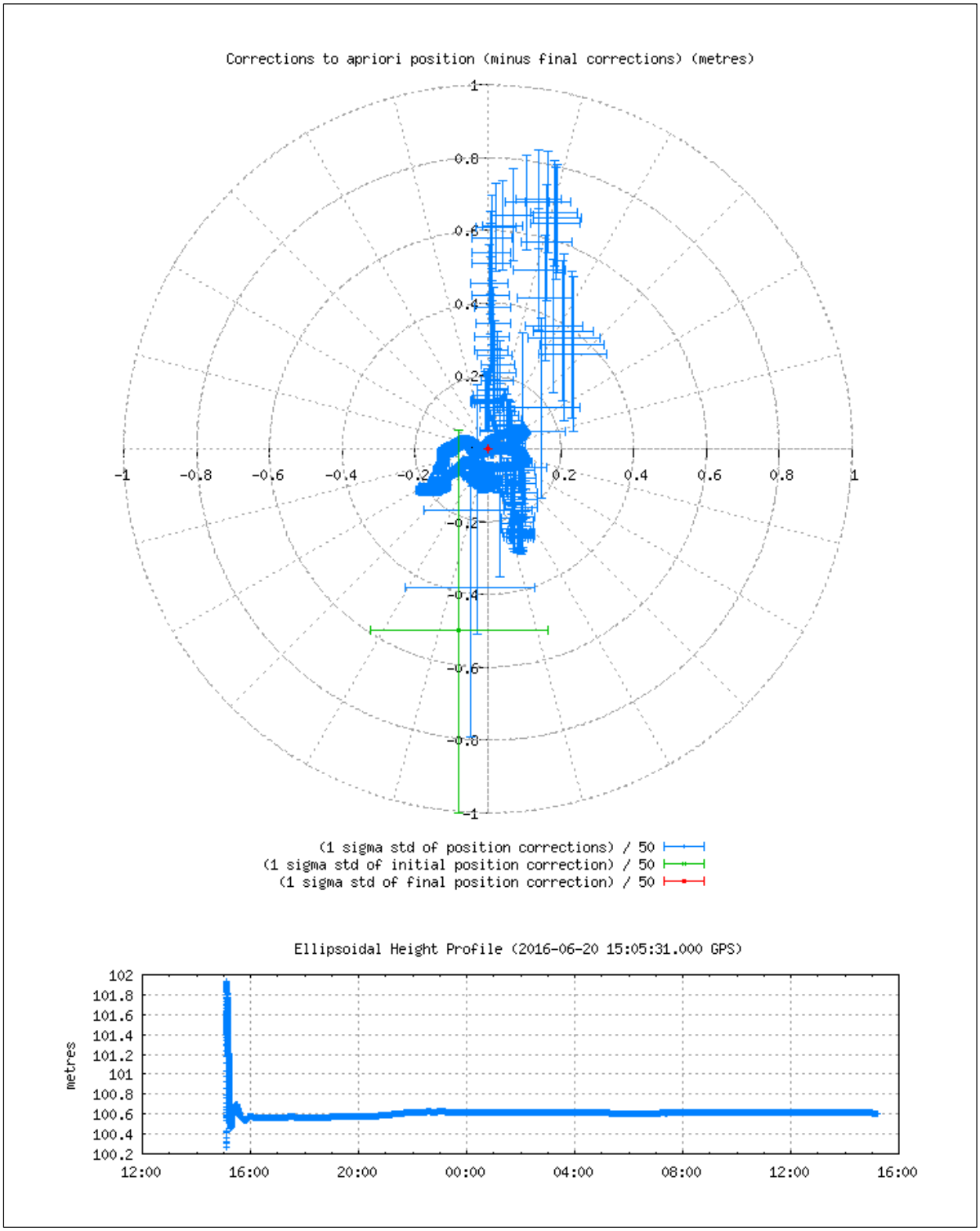
4449285.137m (N) 549549.062m (E)

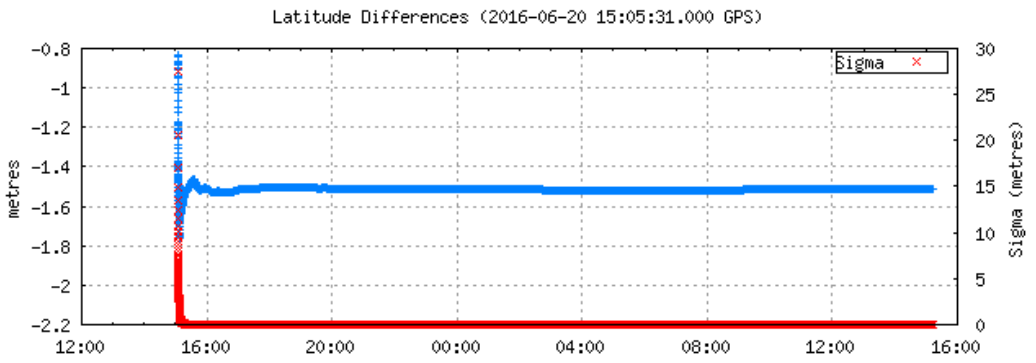
Scale Factors
0.99963022 (point)
0.99961443 (combined)

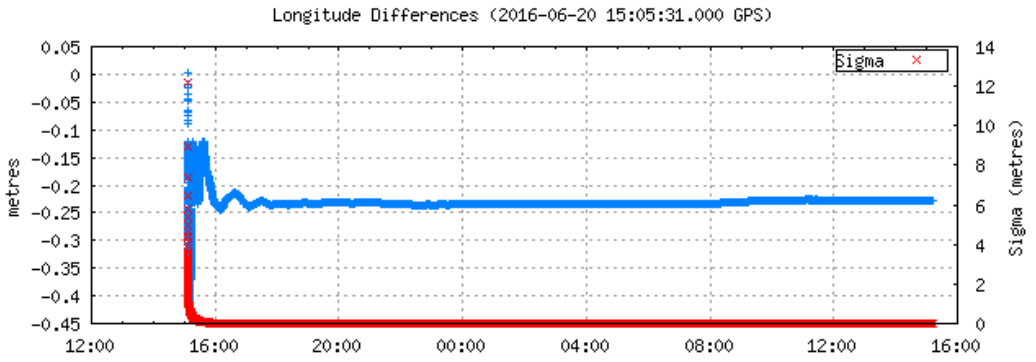
(Coordinates from RINEX file used as apriori position)

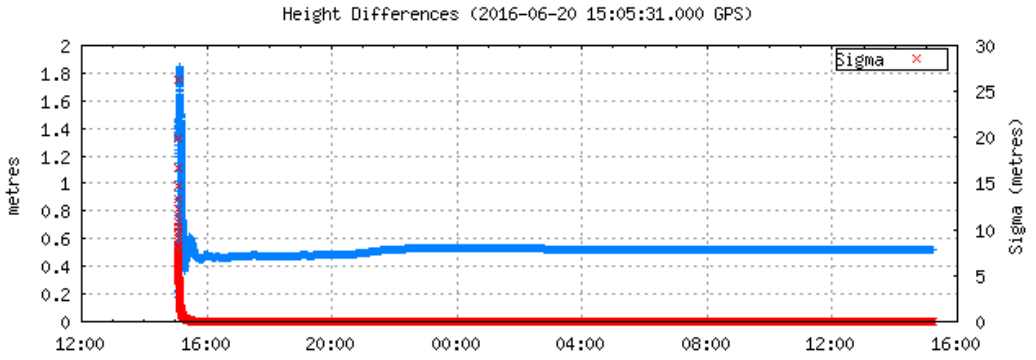
Estimated Parameters & Observations Statistics

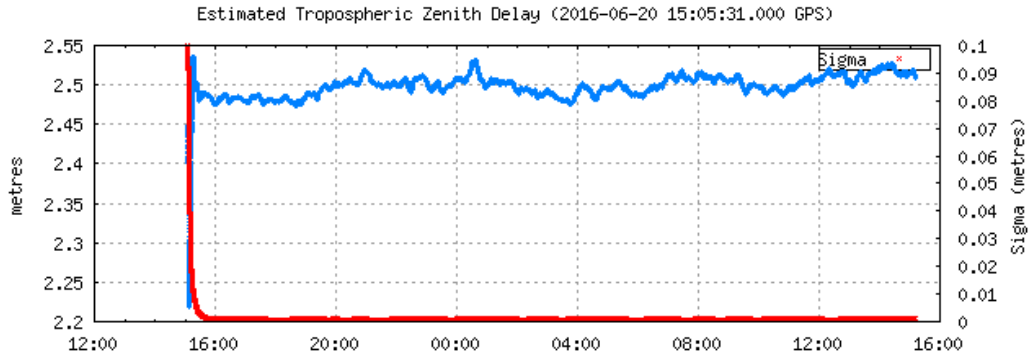


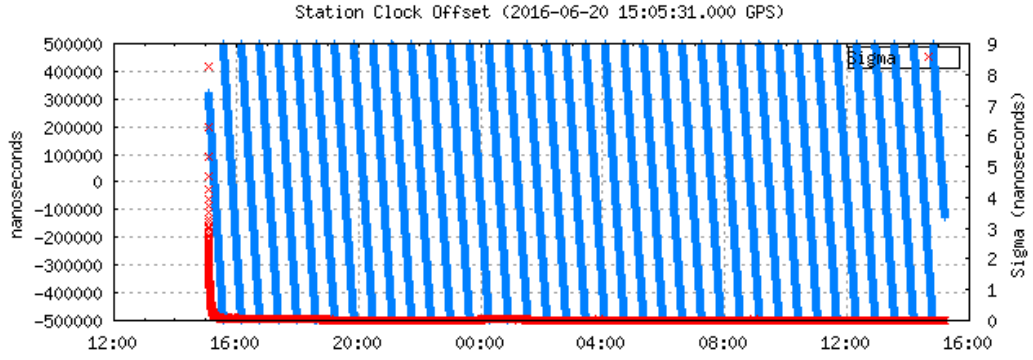

Corrections to apriori position (minus final corrections) (metres)
1
0.8
0.6
0.4
0.2
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.8
-0.6
-0.4
-0.2
0.2
0.4
0.6
0.8
1
(1 sigma std of position corrections) / 50
(1 sigma std of initial position correction) / 50
(1 sigma std of final position correction) / 50
Ellipsoidal Height Profile (2016-06-20 15:05:31.000 GPS)
metres
102
101.8
101.6
101.4
101.2
101
100.8
100.6
100.4
100.2
12:00
16:00
20:00
00:00
04:00
08:00
12:00
16:00

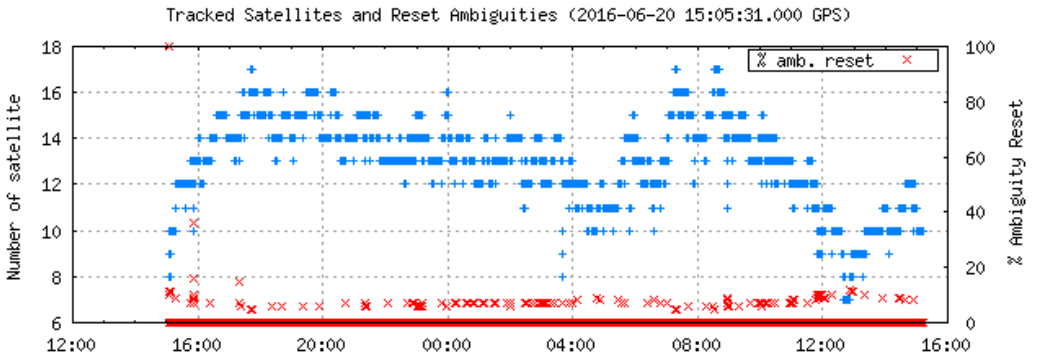

Latitude Differences (2016-06-20 15:05:31.000 GPS)
Sigma
metres
Sigma (metres)
-0.8
-1
-1.2
-1.4
-1.6
-1.8
-2
-2.2
30
25
20
15
10
5
0
12:00
16:00
20:00
00:00
04:00
08:00
12:00
16:00


Longitude Differences (2016-06-20 15:05:31.000 GPS)
Sigma
metres
Sigma (metres)
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
-0.3
-0.35
-0.4
-0.45
14
12
10
8
6
4
2
0
12:00
16:00
20:00
00:00
04:00
08:00
12:00
16:00


Height Differences (2016-06-20 15:05:31.000 GPS)
Sigma
metres
Sigma (metres)
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
30
25
20
15
10
5
0
12:00
16:00
20:00
00:00
04:00
08:00
12:00
16:00


Estimated Tropospheric Zenith Delay (2016-06-20 15:05:31.000 GPS)
metres
Sigma (metres)
Sigma
2.55
2.5
2.45
2.4
2.35
2.3
2.25
2.2
0.1
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
12:00
16:00
20:00
00:00
04:00
08:00
12:00
16:00


Station Clock Offset (2016-06-20 15:05:31.000 GPS)
nanoseconds
Sigma (nanoseconds)
Sigma
500000
400000
300000
200000
100000
0
-100000
-200000
-300000
-400000
-500000
9
8
7
6
5
4
3
2
1
0
12:00
16:00
20:00
00:00
04:00
08:00
12:00
16:00


Tracked Satellites and Reset Ambiguities (2016-06-20 15:05:31.000 GPS)
Number of satellite
% Ambiguity Reset
% amb. reset
18
16
14
12
10
8
6
100
80
60
40
20
0
12:00
16:00
20:00
00:00
04:00
08:00
12:00
16:00

Appendix H: GNSS/INS Integration

The routes can be seen with better quality using the Google Maps in the following address,

`https://drive.google.com/open?id=1qAzMKi3yE7Z3ePCLC49Nmok6no0&usp=sharing`

Appendix I: RTK-GNSS/INS Integration

The following paths can be seen with better quality using the Google Maps in the following address,

`https://drive.google.com/open?id=10ep9klfXXHQZyv4z90c8nV8kkiY&usp=sharing`

Bibliography

[Bar-Shalom *et al.*, 2004] Yaakov Bar-Shalom, X. Rong Li, and Thiagalingam Kirubarajan. *Estimation with Applications to Tracking and Navigation: Theory, Algorithms, and Software*. John Wiley & Sons, 2004.

[Cabrita *et al.*, 2015] Gonçalo Cabrita, Raj Madhavan, and Lino Marques. A framework for remote field robotics competitions. In *Autonomous Robot Systems and Competitions (ICARSC), 2015 IEEE International Conference on*, pages 192--197. IEEE, 2015.

[Carcanague, 2013] Sébastien Carcanague. *Low-cost GPS/GLONASS Precise Positioning Algorithm in Constrained Environment*. PhD thesis, Université de Toulouse, 2013.

[Crassidis, 2006] John L. Crassidis. Sigma-point kalman filtering for integrated gps and inertial navigation. *IEEE Transactions on Aerospace and Electronic Systems*, April 2006.

[DMATM 8358.2, 1989] DMATM 8358.2. The universal grids: Universal transverse mercator (utm) and universal polar stereographic (ups). *DMA Technical Manual*, September 1989.

[El-Sheimy *et al.*, 2008] Naser El-Sheimy, Haiying Hou, and Xiaoji Niu. Analysis and Modeling of Inertial Sensors Using Allan Variance, volume = 57. *Ieee Transactions On Instrumentation And Measurement*, 2008.

[Elisson and Gassler, 2014] Viktor Elisson and Gabriel Gassler. Low cost relative gnss positioning with imu integration. Master's thesis, Chalmers University of Technology, 2014.

[Everett, 2016] Tim Everett. Exploring ultra-low cost precision gps with rtklib and ublox receivers, 2016.

[Fraga *et al.*, 2013] Jorge Fraga, João Sousa, Gonçalo Cabrita, Paulo Coimbra, and Lino Marques. Squirtle: An ASV for Inland Water Environmental Monitoring. *ROBOT2013: First Iberian Robotics Conference*, pages 33--39, 2013.

[Fraga, 2012] Jorge Fraga. Trajectory Control for an Unmanned Surface Vehicle. Master's thesis, Faculdade de Ciências e Tecnologia da Universidade de Coimbra, 2012.

[Ge and Lewis, 2006] Shuzhi Sam Ge and Frank L. Lewis. *Autonomous Mobile Robots: Sensing, Control, Decision Making and Applications*. Taylor & Francis, 2006.

[Grewal and Andrews, 2015] Mohinder S. Grewal and Angus P. Andrews. *Kalman Filtering: Theory and Practice with MATLAB*. John Wiley & Sons, 2015.

[Grewal *et al.*, 2007] Mohinder S. Grewal, Lawrence R. Weill, and Angus P. Andrews. *Global Positioning Systems, Inertial Navigation, and Integration*. Wiley, 2nd edition, August 2007.

[Gustafsson, 2010] Fredrik Gustafsson. *Statistical Sensor Fusion*. Studentlitteratur, 2010.

[Gustafsson, 2014] Johan Gustafsson. Orientation estimation of a rigid multi body system using accelerometers, gyroscopes and the geometry. Master's thesis, Chalmers University of Technology, 2014.

[Hjelmare and Rangsjö, 2012] Fredrik Hjelmare and Jonas Rangsjö. Simultaneous localization and mapping using a kinect in a sparse feature indoor environment. Master's thesis, Linköping University, 2012.

[Hol, 2011] Jeroen Hol. Sensor fusion and calibration of inertial sensors, vision, ultra-wideband and gps. Master's thesis, Linköping University, 2011.

[Hoque and Jakowski, 2012] M. Mainul Hoque and Norbert Jakowski. *Global Navigation Satellite Systems: Signal, Theory and Applications*, chapter Ionospheric Propagation Effects on GNSS Signals and New Correction Approaches. InTech, 2012.

[IEEE AES, 1997] IEEE AES. IEEE Standard Specification Format Guide and test Procedure for Single-Axis Interferometric Fiber Optic Gyros. *IEEE Std. 852-1997 (R2008)*, 1997.

[Jeffrey, 2010] Charles Jeffrey. *An Introduction to GNSS: GPS, GLONASS, Galileo and other Global Navigation Satellite Systems*. NovAtel Inc., 2010.

[Kelly, 1994] Alonzo Kelly. A 3d state space formulation of a navigation kalman filter for autonomous vehicles. Technical Report CMU-RI-TR-94-19, Carnegie Mellon University, 1994.

[Kreinar, 2013] Edward Joseph Kreinar. Filter-based slip detection for a complete-coverage robot. Master's thesis, Case Western Reserve University, 2013.

[Lopes, 2011] Hugo Lopes. Attitude Determination of Highly Dynamic Fixed-wing UAVs - A MEMS-AHRS/GPS Integration. Master's thesis, Instituto Superior Técnico, 2011.

[Madgwick, 2010] Sebastian O.H. Madgwick. An efficient orientation filter for inertial and inertial/magnetic sensor arrays. Technical report, University of Bristol, April 2010.

[Magnusson and Odenman, 2012] Niklas Magnusson and Tobias Odenman. Improving absolute position estimates of an automotive vehicle using GPS in sensor fusion. Master's thesis, Chalmers University of Technology, 2012.

[NAVSTAR, 2008] NAVSTAR. *GPS Standard Positioning Service Performance Standard*. US-DOD, 4th edition, 2008.

[Nemra and Aouf, 2010] Abdelkrim Nemra and Nabil Aouf. Robust INS/GPS Sensor Fusion for UAV Localization Using SDRE Nonlinear Filtering. *IEEE Sensors Journal*, 10(4):789--798, 2010.

[Newman, 2003] Paul Michael Newman. C4b---mobile robotics. *Accessed*, 2, 2003.

[OlliW, 2013] OlliW. IMU Data Fusing: Complementary, Kalman, and Mahony Filter, 2013.

[O'Reilly *et al.*, 2009] Rob O'Reilly, Alex Khenkin, and Kieran Harney. Sonic nirvana: Using mems accelerometers as acoustic pickups in musical instruments. *Analog Dialogue*, 43, 2009.

[Petrovski, 2014] Ivan G. Petrovski. *GPS, GLONASS, Galileo, and BeiDou for Mobile Devices: From Instant to Precise Positioning*. Cambridge University Press, 2014.

[Renaudin *et al.*, 2010] Valérie Renaudin, Muhammad Haris Afzal, and Gérard Lachapelle. Complete triaxis magnetometer calibration in the magnetic domain. *Journal of Sensors*, 2010.

[Rodrigues *et al.*, 2013] Paulo Rodrigues, Franscisco Marques, Eduardo Pinto, Ricardo Pombeiro, André Lourenço, Ricardo Mendonça, Pedro Santana, and José Barata. RIVERWATCH - A Marsupial Surface-Aerial Robotic Team for Riverine Environmental Monitoring. *http://riverwatchws.cloudapp.net*, 2013.

[Rogers-Marcovitz, 2010] Forrest Rogers-Marcovitz. On-line mobile robotic dynamic modeling using integrated perturbative dynamics. Technical Report CMU-RI-TR-10-15, Carnegie Mellon University, 2010.

[Sickle, 2014] Jan Van Sickle. *GEOG 862: GPS and GNSS for Geospatial Professionals*. Penn State's Online Master of GIS, 2014.

[Takasu and Yasuda, 2009] Tomoji Takasu and Akio Yasuda. Development of the low-cost RTK-GPS receiver with an open source program package RTKLIB. *Laboratory of Satellite Navigation, Tokyo University of Marine Science and Technology*, 2009.

[Takasu, 2015] Tomoji Takasu. GNSS Precise Positioning and RTKLIB. 2015.

[Terejanu, 2008] Gabriel A. Terejanu. Extended kalman filter tutorial. *University at Buffalo*, 2008.

[Teunissen, 1994] P. J. G. Teunissen. A new method for Fast Carrier Phase Ambiguity Estimation. *IEEE Position Location and Navigation Symposium*, April 1994.

[Titterton and Weston, 2005] David Titterton and John L. Weston. *Strapdown Inertial Navigation Technology*. The Institution of Engineering and Technology, 2005.

[Vectornav, 2015] Vectornav. VectorNav Support Library, 2015.

[Welch and Bishop, 2006] Greg Welch and Gary Bishop. An Introduction to the Kalman Filter. Technical Report TR 95-041, University of North Carolina at Chapel Hill, July 2006.

[Woodman, 2007] Oliver J. Woodman. An introduction to inertial navigation. Technical Report UCAM-CL-TR-696, University of Cambridge, August 2007.

[Xu, 2003] Guochang Xu. *GPS Theory, Algorithms and Applications*. Springer, 2003.