POLITECNICO DI TORINO

Master's Degree in Communications and Computer Systems

Master's Degree Thesis

Study of advanced navigation algorithms for GNSS based navigation

Supervisors

Prof. Fabio DOVIS

Prof. Alex MINETTO

Dr. Stefano MARANGONI

Candidate

Cristian FERNANDEZ

December 2023

Abstract

This thesis, elaborated in collaboration with Aizoon, has the finality of studying and applying advanced GNSS-based algorithms(specifically Weighted Least Squares(WLS), Kalman filtering, and GNSS/INS integration) with the intention of comparing their performance on mobile devices.

In order to do this, first of all, there was a study of the state-of-the-art navigation algorithms. These algorithms are then applied on Matlab, using data recovered from smartphones(for the navigation data) and base stations across Italy(for the satellites' parameters). Other improvements to the algorithms are implemented, like the use of multiple GNSS constellations for the computation of the PVT solution and the ionospheric correction that can be applied to the pseudoranges and pseudorange rates. The results from the different algorithms will be compared, in conjunction with the results obtained by using the other improvements to the navigation data.

Acknowledgements

En primer lugar quiero agradecer al profesor Fabio Dovis por ponerme en contacto con Aizoon para llevar a cabo esta investigación.

En segundo lugar a Stefano Marangoni, por aceptar esta colaboración y guiarme en el proceso del desarrollo de este proyecto.

Quiero agradecer además a mis amigos aquí en Turín, que me apoyaron en cada momento de dificultad y que en este corto tiempo se convirtieron en algunas de las personas más importantes en mi vida. Agradezco también a mis amigos en Chile, que me apoyaron y me siguen apoyando a la distancia.

Finalmente quiero agradecer a mi familia. A mi madre, que ante a adversidad ha demostrado que siempre existe la opción de seguir adelante, independientemente de lo complicada que puede ser la situación. Mi hermana, de las personas que más me ha enseñado en esta vida y a quien quiero incondicionalmente. Y finalmente mi padre, que con su esfuerzo me permitió llegar a donde estoy hoy y a quien nunca voy a dejar de extrañar.

Table of Contents

List of Tables v

List of Figures vi

Acronyms viii

1 Fundamental of GNSS 1
- 1.1 GNSS signal . . . 2
- 1.2 Radionavigation principles . . . 4
  - 1.2.1 Reference systems . . . 5
  - 1.2.2 Satellite orbits . . . 6
- 1.3 Error sources . . . 7
  - 1.3.1 Satellite Clock Error . . . 7
  - 1.3.2 Receiver Clock error . . . 8
  - 1.3.3 Ionospheric Delay . . . 8
  - 1.3.4 Tropospheric Delay . . . 9
  - 1.3.5 Multipath Errors . . . 9
  - 1.3.6 Satellite Orbital Errors . . . 9
  - 1.3.7 Receiver noise . . . 9
  - 1.3.8 User equivalent Range error . . . 10
- 1.4 Position determination . . . 10
  - 1.4.1 Determining Satellite-to-User Range . . . 10
  - 1.4.2 Computation of the User Position . . . 12
- 1.5 Position estimation algorithms . . . 15
  - 1.5.1 WLS . . . 15
  - 1.5.2 Kalman filter . . . 16
  - 1.5.3 GNSS/INS integration . . . 16

2 Inertial Navigation System(INS) 17
- 2.1 Inertial Measurement Unit(IMU) . . . 18
  - 2.1.1 Accelerometer measurements . . . 18

2.1.2 Gyroscope measurements . . . 18
2.2 Inertial Sensor Errors . . . 19
2.2.1 Systematic Errors . . . 19
2.2.2 Random Errors . . . 20
2.3 Calibration . . . 20
2.3.1 Six-Position Static Test . . . 20
2.3.2 Importance of calibration . . . 22
2.4 Navigation . . . 22
2.4.1 INS Mechanization . . . 23
2.4.2 Computation of the parameters in the l-frame . . . 24

3 GNSS Navigation Algorithms 25
3.1 Kalman Filter . . . 25
3.1.1 Kalman filter model . . . 26
3.1.2 Kalman filter algorithm . . . 28
3.1.3 GNSS Kalman filter . . . 30
3.2 GNSS/INS integration . . . 32
3.2.1 Loosely Coupled INS/GNSS Integration . . . 34
3.2.2 Tightly coupled integration . . . 35
3.2.3 Ultra-tight integration . . . 36
3.3 Integration algorithm . . . 37
3.3.1 System Model . . . 37
3.3.2 Measurement Model . . . 38

4 Software Development 41
4.1 Data acquisition . . . 41
4.1.1 GNSSLogger . . . 42
4.1.2 RINEX files . . . 44
4.2 Data processing . . . 44
4.3 PVT calculations . . . 45
4.3.1 WLS . . . 45
4.3.2 Kalman filter . . . 46
4.3.3 GNSS/INS Kalman filter . . . 48
4.4 PVT improvements . . . 49
4.4.1 Multi-constellation PVT . . . 49
4.4.2 Ionospheric correction . . . 50

5 Results 51
5.1 WLS . . . 51
5.1.1 Multi-constellation PVT . . . 52
5.1.2 Ionospheric Corrections . . . 53

5.2 Kalman filter 58
5.3 GNSS/INS integration 68
5.3.1 Calibration 68
5.3.2 PVT solution 69

6 Conclusions and possible improvements 74

Bibliography 77

List of Tables

1.1 Ephemeris data 7

4.1 GNSS data delivered by GNSSLogger 43
4.2 INS data delivered by GNSSLogger 43

List of Figures

1.1 Main components of a GPS L1 C/A signal.[6] . . . 3
1.2 GPS, GLONASS, and Galileo navigational frequency bands.[6] . . . 3
1.3 Trilateration . . . 5
1.4 Vectors for trilateration [9] . . . 11

2.1 INS components[7] . . . 17
2.2 IMU[7] . . . 18
2.3 Inertial Navigation Concept[7] . . . 22

3.1 Kalman filter data flow[8] . . . 30
3.2 Typical INS/GNSS integration[7] . . . 32
3.3 Open-loop INS correction architecture.[8] . . . 33
3.4 Closed-loop INS correction architecture.[8] . . . 34
3.5 Block diagram of loosely coupled integration[8] . . . 35
3.6 Block diagram of tightly coupled integration[8] . . . 36
3.7 Block diagram of ultra-tight integration[8] . . . 37
3.8 Complete Loosely Coupled Integration implementation[7] . . . 40

4.1 GNSSLogger app . . . 42

5.1 WLS output . . . 51
5.2 Pseudorange and pseudorange change for GPS and Galileo . . . 52
5.3 WLS result with multiple constellations . . . 53
5.4 WLS solution with ionospheric corrections . . . 54
5.5 Axes of the WLS solution with ionospheric corrections . . . 54
5.6 Standard deviation of the PVT solution . . . 55
5.7 WLS solution with ionospheric corrections . . . 56
5.8 Axes of the WLS solution with ionospheric corrections . . . 56
5.9 Standard deviation of the PVT solution . . . 57
5.10 Uncalibrated Kalman filter PVT solution . . . 59
5.11 Uncalibrated Kalman filter solution . . . 59
5.12 Uncalibrated Kalman filter PVT solution . . . 61

5.13 Uncalibrated Kalman filter solution 61
5.14 First attempt on calibrated Kalman filter PVT solution 62
5.15 Calibrated Kalman filter PVT solution 63
5.16 Calibrated Kalman filter solution 64
5.17 Kalman filter PVT solution 64
5.18 Zoomed Kalman filter PVT solution 65
5.19 Kalman filter PVT solution for a noisier device 66
5.20 Kalman filter solution axes 66
5.21 Kalman filter PVT solution 67
5.22 Kalman filter solution axes 67
5.23 Data used for calibration 68
5.24 INS/GNSS PVT solution 69
5.25 INS/GNSS PVT solution axes 70
5.26 Zoomed INS/GNSS PVT solution axes 70
5.27 INS/GNSS PVT solution without a continuos GNSS input 71
5.28 Axes of INS/GNSS integration solution 71
5.29 INS/GNSS PVT solution with $f_{GNSS} = 0.2$ 72
5.30 Axes of INS/GNSS integration solution 72
5.31 INS/GNSS PVT solution with $f_{GNSS} = 0.1$ 73
5.32 Axes of INS/GNSS integration solution 73

Acronyms

GNSS
Global Navigation Satellite System

GPS
Global Positioning System

MEO
Medium Earth Orbit

ESA
European Space Agency

GEO
Geostationary Earth Orbit

PRN
Pseudo Random Noise

RNSS
Radio Navigation Satellite Systems

ECI
Earth Central Inertial

ECEF
Earth Centered Earth Fixed

IMU
Inertial Measurement Unit

INS

Inertial Navigation System

PVT

Position, Velocity and Time

RF

Radio Frequency

UERE

User Equivalent Range Error

DSSS

Direct Sequence Spread Spectrum

WLS

Weighted Least Squares

DOP

Dilution of Presition

DR

Dead Reckoning

KF

Kalman Filter

STD

Standard Deviation

Chapter 1

Fundamental of GNSS

A Global Navigation Satellite System (GNSS) involves a constellation of satellites orbiting at about twenty thousand kilometers altitude over the earth's surface, continuously transmitting signals that enable users to determine their three-dimensional position with global coverage.[1]

GNSS systems consist of 3 different segments:

- Space segment: Constellation of satellites transmitting radio signals to the users.
- Control segment: Global network of ground facilities that tracks, monitors, and sends commands to the GNSS satellites.
- User segment: Receivers that determine their own position, velocity, and time through the received signals.

As of 2023, four global systems are operational: the United States' Global Positioning System (GPS), Russia's Global Navigation Satellite System (GLONASS), China's BeiDou Navigation Satellite System, and the European Union's Galileo. These constellations aim to provide total world coverage on the Earth's surface.

The baseline GPS constellation consists of 24 satellites arranged in 6 orbital planes, equally spaced and 4 satellites per plane. Over the years new satellites have been added to the constellation to add up to a total of 31 satellites currently operating. These are placed in Medium Earth Orbit (MEO) at an altitude of approximately 20,200 km, with an inclination of 55 degrees with respect to the equatorial plane, and with a nominal orbital period of about 12 hours [2].

Having the same functioning principles as GPS the Globalnaya Navigazionnaya Sputnikovaya Sistema (GLONASS) constellation was pioneered by the Soviet Union. GLONASS is composed of 24 satellites arranged over 3 orbital planes, meaning that there are 8 satellites on each plane. These satellites operate on circular orbits

at an altitude of 19,100 km and an inclination of 64.8 degrees. It also has a similar orbital period to GPS being about 11 hours and 15 minutes [3].

Galileo was developed by the European Union and the European Space Agency (ESA). It consists of 30 satellites at an altitude of 23,222 km in MEO orbit, with 10 satellites in each of the 3 orbital planes, with an inclination of 56 degrees, and an orbital period of 14 hours 4 minutes and 42 seconds[4].

China's BeiDou Navigation Satellite System (BDS) consists of a constellation of 5 Geoestationary-Earth Orbit (GEO) satellites, 27 in MEO, and 3 in Inclined Geosynchronous Orbit (IGSO), summing up to a total of 35 satellites [5].

In this study, the used constellations will be GPS and Galileo, given the location of the used receivers (specifically Piemonte, Italy).

1.1 GNSS signal

GNSS satellites are constantly transmitting navigation signals in two or more frequencies in the L band. These signals always contain ranging codes and navigation data that are then used by the user to determine the distance from the satellite to the receiver 1.1. The GNSS signal is composed by:

- Carrier: Radio frequency sinusoidal signal at a given frequency.

- Ranging code: Binary sequences that allow the receiver to determine the travel time of radio signals from the satellite to the receiver. They are called Pseudo-Random Noise (PRN) sequences or PRN codes, given that they are structured like a random variable but are actually deterministic.

- Navigation data: Binary coded message that provides information on satellite ephemeris, clock bias, parameters, satellite health, etc.

As an example, the components of GPS L1 C/A signal are presented in figure 1.1.

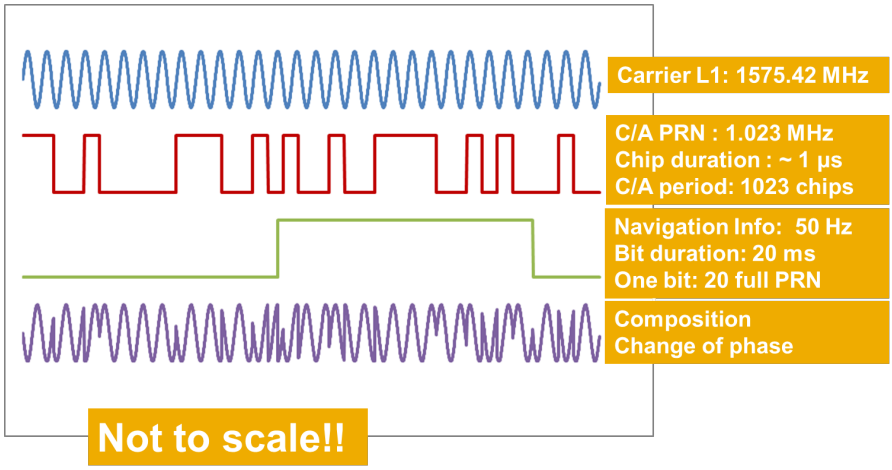


Figure 1.1: Main components of a GPS L1 C/A signal.[6]

The allocation of frequency bands allows multiple users and services to coexist in the same frequency range. The same frequencies can be allocated to different purposes in different countries. The different frequency bands used by each of the constellations are shown in figure 1.2.

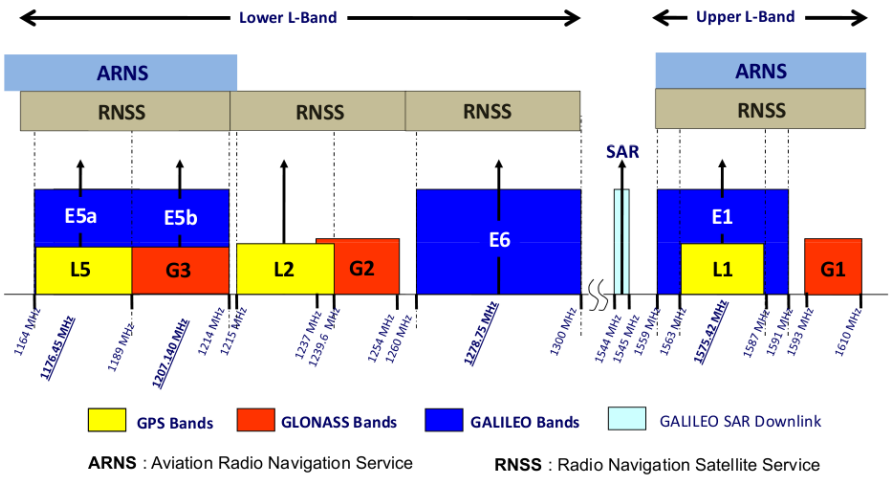


Figure 1.2: GPS, GLONASS, and Galileo navigational frequency bands.[6]

There are two sections that correspond to the Aeronautical Navigation system. These bands correspond to the upper L band (1,559 - 1,610 MHz), having GPS L1, Galileo E1, and GLONASS G1, and to the bottom lower L band (1,151 - 1,214 MHz) where GPS L5 and Galileo E5 are located. The frequencies between 1,215.6 MHz and 1,350 MHz are where you can find the GPS L2, GLONASS G2, and Galileo

E6 signals that are still available. These specific frequency bands were primarily assigned for ground radars and RNSS (Radio Navigation Satellite Systems). As a result, signals within these bands are more susceptible to interference compared to the earlier ones.

For this study, the important frequency bands are GPS L1, Galileo E1, GPS L5, and Galileo E5. Given that the study focuses on GPS and Galileo constellations and the used devices are capable of receiving those signals.

1.2 Radionavigation principles

GNSS systems utilize the concept of Time of Arrival(TOA), ranging in order to determine the user position. This term corresponds to the time it takes for a transmitted signal to reach the user's receiver. This time interval can then be multiplied by the speed of the signal in order to determine the transmitter-to-receiver distance.

The radius of the spheres, corresponding to the geometrical distance between the receiver and the transmitters can be estimated as:

$$R = c(T_{Rx} - T_{Tx}) = c\tau \tag{1.1}$$

Given the navigation data available for the transmitters and the previously described range it is possible to determine the user's position.

Depending on the estimated parameters of the signal, different positioning algorithms can be used. GNNS takes advantage of a process called trilateration, which consists of the intersection of spheres with their centers in the satellites' positions and with a radius equal to the computed range.

To better understand the problem an example of a possible system is shown in Fig1.3. As explained before, the position of the user corresponds to the intersection of the spheres(or circles in this example).

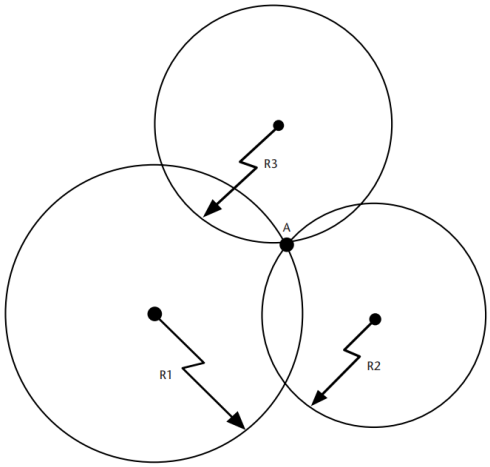


Figure 1.3: Trilateration

1.2.1 Reference systems

In order to determine the user position, it is necessary to use different reference systems, which are used on different parts of the process. It is also important to be able to transition between different systems, which permits one to better express the variables and relate them accurately.

Intertial frame(ECI or i-frame)

For the computation of the position of the GNSS satellites, it is convenient to use the Earth-centered inertial (ECI) coordinate system, which has the center of mass of the Earth as the origin and the direction of the axes fixed with respect to the position of the stars. In this reference system, the trajectory of a satellite obeys Newton's laws of motion and gravity. This makes it easier to determine and subsequently predict the GNSS satellite orbits.

Earth Centred Earth Fixed Frame (ECEF or e-frame)

On the other hand, for computing the position of the GNSS receiver the Earth-centered Earth-fixed system results are more convenient. This reference system rotates with the Earth, making it easier to compute the latitude, longitude, and height parameters that are commonly used for Earth positioning. In this reference system, the xy-plane coincides with the Earth's equatorial plane, while the x-axis points in the direction of 0° longitude and the y-axis points in the direction of 90°E longitude. In general, GNSS orbit software incorporates the transformation between ECI and ECEF systems, which is accomplished by applying rotation matrices to the satellite position and velocity vectors.

Finally, even if the computation is done in ECEF coordinates for simplicity, these coordinates are normally transformed to the latitude, longitude, and height of the receiver. In order to carry out this transformation it is necessary to have a physical model that describes the Earth. In the case of GNSS applications the used model is the DOD's World Geodetic System 1984 (WGS 84). This model provides an ellipsoidal model of the Earth's shape, used to determine the user's latitude, longitude, and height.

Body Frame (b-frame)

In the case that the system possesses an IMU, the body frame represents the orientation of the IMU axes. In stabilized platform systems, the IMU can be kept aligned to a particular navigation frame (for example ECEF), but for our application, the IMU is rigidly mounted to the phone, which means that the body frame can have an arbitrary orientation.

1.2.2 Satellite orbits

For position estimation, it is important to have accurate information about the position of the GNSS satellites. For this reason, it is necessary to understand the physics that characterize their orbits.

In order to determine this position, a number of parameters must be obtained from the satellite. This parameters are called Ephemeris parameters and are included in the GNSS navigation message. For this application, the Ephemeris parameters are recovered from base stations across Italy that provide them each hour.

The Ephemeris data corresponds to the one shown in table 1.1.

t_{oe}	Ephemeris reference time
$\sqrt{a}$	Square rot of the semimajor axis
e	Eccentricity
i_0	Inclination angle at the reference time
Ω_0	Longitude of the ascending node of the orbital plane at the beginning of the GPS week
ω	Argument of perigee
M_0	Mean anomaly at the reference time
Δn	Rate of change of the inclination angle
$\dot{\Omega}$	Rate of change of the RAAN(Right Ascension of the Ascending Node)
$\dot{i}(IDOT)$	Rante of change of the inclination angles
C_{uc}, C_{us}	Amplitudes of the cosine and sine harmonic correction terms for the computed argument of latitude
C_{rc}, C_{rs}	Amplitudes of the cosine and sine harmonic correction terms for the computed orbit radius
C_{ic}, C_{is}	Amplitudes of the cosine and sine harmonic correction terms for the computed inclination angles

Table 1.1: Ephemeris data

The movement of the satellites is described by Kepler's laws, having these parameters(which describe an elliptical movement) in conjunction with the GPS or Galileo current time(depending on the satellite), it is possible to estimate the satellite's position. For a complete explanation of how to compute the satellite position, refer to [7].

1.3 Error sources

As previously stated, GNSS systems use trilateration to determine the users' position, which requires the measurement of the ranges of at least four satellites. This measurements are affected by a plethora of error sources that are presented in this section.

1.3.1 Satellite Clock Error

Over time, the clocks on the satellites get unsynchronized with the GNSS clock system time. This error is estimated using satellite clock data provided by monitoring stations. After the estimation, the control segment uploads the correction of the parameters to the satellites, in order for them to correct the satellite clock.

1.3.2 Receiver Clock error

Given the characteristics of the cheap clocks available on common receivers, the accuracy of the receivers' clocks is normally low, containing a bias, that as we will see in the next sections, is computed in conjunction with the other three components required to determine the user's position.

1.3.3 Ionospheric Delay

In this report, the main focus will be on ionospheric propagation effects and possible ways of avoiding this effect.

The ionosphere extends from about 50 to 1,000 km above the Earth's surface and it is composed of gases that may be ionized, plasma of ions, and free electrons. This ionization of the gases is caused by solar radiation, and because of this, the characteristics of the ionosphere vary with time. The effects of the ionosphere on the signal are frequency-dependent [8].

The ionosphere is a dispersive medium, which means that the propagation velocity changes with the frequency, this means that the signal modulation is delayed. In the case of the ionosphere, the delay of the signal is approximately the same as the carrier phase meaning that

$$\delta\rho \approx -\delta\Phi \tag{1.2}$$

Being $\delta\rho$ the error in pseudorange generated by the ionosphere and $\delta\Phi$ the change in phase of the signal. More than 99% of the propagation delay varies proportional to f^2, being f the frequency of the carrier. The higher-order effects are negligible for this type of application.

In the case that the GNSS receiver tracks signals on more than one frequency, it is possible to combine them to eliminate the ionospheric propagation delay. In this document's application, in the case of GPS signals, the available frequencies are L1 and L5, frequency bands. The ionosphere-corrected pseudorange is computed as

$$\rho^{IC} = \frac{(f^{L1})^2\rho^{L1} - (f^{L5})^2\rho^{L5}}{(f^{L1})^2 - (f^{L5})^2} \tag{1.3}$$

Even if this correction gets rid of the bias generated by ionospheric delay, this combination brings a penalty in the form of increased noise. Assuming that the values of the pseudoranges are independent and Gaussian, the standard deviation of the corrected pseudorange is

$$\sigma^{IC} = \frac{\sqrt{(f^{L1})^4(\sigma^{L1})^2 + (f^{L5})^4(\sigma^{L5})^2}}{|(f^{L1})^2 - (f^{L5})^2|} \tag{1.4}$$

Where σ^{L1} and σ^{L5} are the code tracking error standard deviation for both signals. The error scales up as the frequencies get closer. Given that the ionosphere characteristics vary slowly over time, with correlation times close to half an hour, the ionosphere corrections can be smoothed over time using the previous estimates of the pseudorange to compute the new ones.

The process is explained in further detail in [8] section 9.3.2. This correction of the pseudorange not only eliminates the bias generated by the ionospheric delay, but also manages to generate a PVT solution that decreases its noise over time.

1.3.4 Tropospheric Delay

The troposphere is a lower part of the atmosphere that extends from 8 to 40 km above the earth's surface and is composed of mainly dry gases and water vapor. Tropospheric errors are consistent between different carrier frequencies. The tropospheric error can be modeled in order to eliminate it from the pseudorange measurements.

1.3.5 Multipath Errors

Multipath errors are common in urban environments where the signal is received from different paths. The alternate paths to a direct line of sight are generated by reflections of the signal on objects around the receiver. These received signals are delayed with a lower signal-to-noise ratio. Multipath generates distortion on the received signal because of interference between the original signal and its distorted reflected versions.

1.3.6 Satellite Orbital Errors

The satellite orbital error corresponds to the difference between the position of the satellite and the position estimated by the receiver using ephemeris data. These errors are predicted by the control segment and are uploaded to the satellites to be sent to users as ephemeris data.

1.3.7 Receiver noise

This noise is measurement noise intrinsic to GNSS receivers. It's caused by cumulative effects of antenna circuitry, cables, thermal noise, RF signal interference, signal quantization, and sampling. It can generate an incorrect measurement of the travel time of the GNSS signal.

1.3.8 User equivalent Range error

The combination of all these effects on the measurement of the pseudoranges is called user equivalent range error (UERE). Assuming that all the previously mentioned error sources are uncorrelated, the composite UERE for one of the pseudoranges can be expressed as a zero-mean Gaussian random variable with variance:

$$\sigma_{UERE} = \sqrt{\sigma_{eph}^2 + \sigma_{clk}^2 + \sigma_{ion}^2 + \sigma_{tro}^2 + \sigma_{mlt}^2 + \sigma_{rcv}^2} \tag{1.5}$$

where

- σ_{eph} is the range error due to ephemeris data
- σ_{clk} is the range error due to the satellite's clock
- σ_{ion} is the range error due to the ionosphere
- σ_{tro} is the range error due to troposphere
- σ_{mlt} is the range error due to multipath
- σ_{rcv} is the range error due to the receiver measurement

1.4 Position determination

GNSS satellites make use of direct sequence spread spectrum (DSSS) modulation. This modulation provides the structure for transmitting ranging signals and navigation data. The ranging codes correspond to PRN codes that are used to BPSK modulate the satellite carrier frequencies. These codes have a predictable and periodic pattern that can be replicated by a suitable receiver. Having a replica of the code running at the same time as the satellite, it is possible to determine the shift between the received code and the generated one, which will help to determine the distance between the satellite and the receiver.

1.4.1 Determining Satellite-to-User Range

Previously, it was explained that it was possible to determine the user position through ranging signals and the multiple spheres generated by their use with the trilateration process. In these examples, it was assumed that the receiver clock was perfectly synchronized to the system's time. This assumption is in general not the case. Even if other error sources affect the range measurement accuracy, these can be negligible in comparison to the errors experienced by non-synchronized clocks. Therefore, for the study of the basics of GNSS other errors are omitted [9].

In figure 1.4 we wish to determine the vector **u**, which corresponds to the receiver's position with respect to the ECEF coordinate system origin, which is unknown. **r** corresponds to the vector distance measured from the user to the satellite. Finally, **s** corresponds to the position of the satellite con the ECEF coordinate system. The value of **s** is determined by using ephemeris data broadcasted by the satellite.

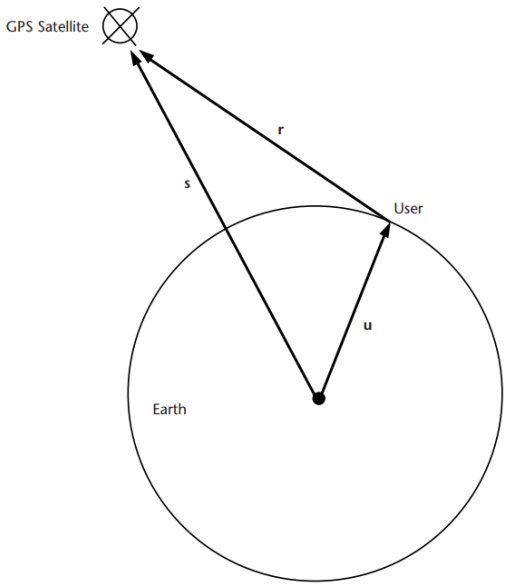


Figure 1.4: Vectors for trilateration [9]

Let r represent the magnitude of **r**, being the geometric range, we know that:

$$r = c(T_u - Ts) = c\Delta t \tag{1.6}$$

Where T_s is the time at which the signal left the satellite and T_u is the time at which it was received by the user. On the other hand, the pseudorange computed by the receiver corresponds to:

$$\rho = c[(T_u - t_u) - (T_S - \delta t)] \tag{1.7}$$

Being t_u the offset of the receiver clock from the system and δt the offset of the satellite clock from the system's time.

Using 1.6, 1.7 can be rewritten as:

$$\rho = r + c(t_u - \delta t) \tag{1.8}$$

It is also known that the range can be represented by:

$$r = ||\mathbf{s} - \mathbf{u}|| \tag{1.9}$$

Which can be rewritten using equation 1.8 as:

$$\rho - c(t_u - \delta t) = ||\mathbf{s} - \mathbf{u}|| \tag{1.10}$$

Finally, it is known that the value of δt is usually compensated by the system, so it can be expressed as a known quantity. Hence, equation 1.10 can be expressed as:

$$\rho - ct_u = ||\mathbf{s} - \mathbf{u}|| \tag{1.11}$$

1.4.2 Computation of the User Position

Using equation 1.10 it is possible to build a system of equations that can be solved in order to determine the user position. Also, the value of ct_u can be changed to b_u which expresses a bias in the value of the pseudorange.

$$\rho_j = ||\mathbf{s}_j - \mathbf{u}|| + b_u \tag{1.12}$$

Being j an available satellite. The position of the user is described by three dimensions (x_u, y_u, z_u) which are the unknown values in conjunction with the bias b_u. This means that in order to determine the variables, at least 4 equations are needed, these are:

$$\rho_1 = \sqrt{(x_1 - x_u)^2 + y_1 - y_u)^2 + z_1 - z_u)^2} + b_u \tag{1.13}$$

$$\rho_2 = \sqrt{(x_2 - x_u)^2 + y_2 - y_u)^2 + z_2 - z_u)^2} + b_u \tag{1.14}$$

$$\rho_3 = \sqrt{(x_3 - x_u)^2 + y_3 - y_u)^2 + z_3 - z_u)^2} + b_u \tag{1.15}$$

$$\rho_4 = \sqrt{(x_4 - x_u)^2 + y_4 - y_u)^2 + z_4 - z_u)^2} + b_u \tag{1.16}$$

where x_j, y_j, and z_j denote the jth satellite's position in three dimensions.

These nonlinear equations can be solved using a plethora of possible methods, but the focus of this thesis will be on Weighted Least Squares(WLS) and more importantly Kalman filtering, which will be discussed in the following chapters. For both methods, the linearization of the equations is utilized. This linearization is described in the following paragraphs.

As stated before, the value of a pseudorange can be represented by:

$$\rho_j = \sqrt{(x_j - x_u)^2 + (y_j - y_u)^2 + (z_j - z_u)^2} + b_u = f(x_u, y_u, z_u, b_u) \tag{1.17}$$

Using the value of an approximate location, which is a relatively fair assumption given that the approximate location of the receiver can be determined using past positions or the positions of other nearby stations, the approximate pseudorange is computed as:

$$\hat{\rho_j} = \sqrt{(x_j - \hat{x_u})^2 + (y_j - \hat{y_u})^2 + (z_j - \hat{z_u})^2} + \hat{b_u} = f(\hat{x_u}, \hat{y_u}, \hat{z_u}, \hat{b_u}) \tag{1.18}$$

The pseudorange can then be linearized by using the first-order Taylor expansion around known approximate location $(\hat{x_u}, \hat{y_u}, \hat{z_u}, \hat{b_u})$:

$$\begin{aligned} f(x_u, y_u, z_u, b_u) &= f(\hat{x_u}, \hat{y_u}, \hat{z_u}, \hat{b_u}) \\ +\frac{\partial f(\hat{x_u}, \hat{y_u}, \hat{z_u}, \hat{b_u})}{\partial \hat{x}_u}(x_u - \hat{x}_u) &+ \frac{\partial f(\hat{x_u}, \hat{y_u}, \hat{z_u}, \hat{b_u})}{\partial \hat{y}_u}(y_u - \hat{y}_u) \\ +\frac{\partial f(\hat{x_u}, \hat{y_u}, \hat{z_u}, \hat{b_u})}{\partial \hat{z}_u}(z_u - \hat{z}_u) &+ \frac{\partial f(\hat{x_u}, \hat{y_u}, \hat{z_u}, \hat{b_u})}{\partial \hat{b}_u}(b_u - \hat{b}_u) \end{aligned} \tag{1.19}$$

The partial derivates evaluate as:

$$\frac{\partial f(\hat{x_u}, \hat{y_u}, \hat{z_u}, \hat{b_u})}{\partial \hat{x}_u} = -\frac{x_j - \hat{x}_u}{\hat{r}_j} \tag{1.20}$$

$$\frac{\partial f(\hat{x_u}, \hat{y_u}, \hat{z_u}, \hat{b_u})}{\partial \hat{y}_u} = -\frac{y_j - \hat{y}_u}{\hat{r}_j} \tag{1.21}$$

$$\frac{\partial f(\hat{x_u}, \hat{y_u}, \hat{z_u}, \hat{b_u})}{\partial \hat{z}_u} = -\frac{z_j - \hat{z}_u}{\hat{r}_j} \tag{1.22}$$

$$\frac{\partial f(\hat{x_u}, \hat{y_u}, \hat{z_u}, \hat{b_u})}{\partial \hat{b}_u} = 1 \tag{1.23}$$

Being $\hat{r}_j = \sqrt{(x_j - \hat{x}_u)^2 + (y_j - \hat{y}_u)^2 + (x_j - \hat{y}_u)^2}$.
Then, substituting the partial derivatives into equation 1.19 and rearranging the variables yields:

$$\Delta\rho = \frac{x_j - \hat{x}_u}{\hat{r}_j}\Delta x_u + \frac{y_j - \hat{y}_u}{\hat{r}_j}\Delta y_u + \frac{z_j - \hat{z}_u}{\hat{r}_j}\Delta z_u - \Delta b_u \tag{1.24}$$

Being $\Delta\rho = \hat{\rho}_j - \rho_j$, $\Delta x_u = x_u - \hat{x}_u$, $\Delta y_u = y_u - \hat{y}_u$, $\Delta z_u = z_u - \hat{z}_u$, and $\Delta b = b_u - \hat{b}_u$.

For convenience, the terms can be simplified as:

$$\begin{aligned} a_{xj} &= \frac{x_j - \hat{x}_u}{\hat{r}_j} \\ a_{yj} &= \frac{y_j - \hat{y}_u}{\hat{r}_j} \\ a_{zj} &= \frac{z_j - \hat{z}_u}{\hat{r}_j} \end{aligned} \tag{1.25}$$

This means that equation 1.24 can be rewwritten as:

$$\rho_j = a_{xj}\Delta x_u + a_{yj}\Delta y_u + a_{zj}\Delta z_u - \Delta b_u \tag{1.26}$$

The unknowns in this new equation would be Δx_u, Δy_u, Δz_u, and Δb_u. And they can be determined by solving the set of linear equations:

$$\begin{aligned} \rho_1 &= a_{x1}\Delta x_u + a_{y1}\Delta y_u + a_{z1}\Delta z_u - \Delta b_u \\ \rho_2 &= a_{x2}\Delta x_u + a_{y2}\Delta y_u + a_{z2}\Delta z_u - \Delta b_u \\ \rho_3 &= a_{x3}\Delta x_u + a_{y3}\Delta y_u + a_{z3}\Delta z_u - \Delta b_u \\ \rho_1 &= a_{x4}\Delta x_u + a_{y4}\Delta y_u + a_{z4}\Delta z_u - \Delta b_u \end{aligned} \tag{1.27}$$

These equations can be put in a matrix form:

$$\Delta\rho = \begin{bmatrix} \Delta\rho_1 \\ \Delta\rho_2 \\ \Delta\rho_3 \\ \Delta\rho_4 \end{bmatrix} \qquad H = \begin{bmatrix} a_{x_1} & a_{y_1} & a_{z_1} & 1 \\ a_{x_2} & a_{y_2} & a_{z_2} & 1 \\ a_{x_3} & a_{y_3} & a_{z_3} & 1 \\ a_{x_4} & a_{y_4} & a_{z_4} & 1 \end{bmatrix} \qquad \Delta x = \begin{bmatrix} \Delta x_u \\ \Delta y_u \\ \Delta z_u \\ -\Delta b_u \end{bmatrix} \tag{1.28}$$

Being the relation between the vectors and the matrix:

$$\Delta\rho = H\Delta x \tag{1.29}$$

Which has the solution

$$\Delta x = H^{-1}\Delta\rho \tag{1.30}$$

This linear solution will work as long as the linearization point is within proximity to the actual user's position. This can be achieved by iteratively repeating the linearization process, solving the equation, and updating the user's estimated position until the value computed of Δx is small enough so that it is known that the estimated user position is very close to the actual user's position. It can also be

shown that for the estimation of the user's velocity, if the value of the pseudorange rate is known, the same matrix H is used for the computation.

As explained previously, in order to compute the user position, equation 1.29 must be solved by finding the inverse of the matrix H. In the previously discussed chapter, it is assumed that H makes use of four satellites' pseudoranges. This is not necessarily the case, given that there can be more satellites in the line of sight of the user. The use of more satellites can increase the accuracy of the solution, considering that the geometry of the system is improved. This concept is known as dilution of precision (DOP), this is a factor that affects the precision of the PVT solution depending on the geometry between the satellites and the user [9].

Because of this, from now on, when equations refer to the value H^{-1} it can be the value of the actual inverse of H if it is squared, or the pseudoinverse of H, in the case of an overdetermined system. This pseudoinverse can be determined using the Least Squares method or, as it will be described later, the Weighted Least Squares algorithm (WLS).

1.5 Position estimation algorithms

As previously stated, there are a lot of methods used for the determination of the user's position, velocity, and time. In this document, the focus will be on WLS and Kalman filtering, specifically using GNSS data with and without ionospheric corrections, and GNSS/INS integration.

1.5.1 WLS

At the moment of computing the inverse of the H matrix, it is assumed that the computed pseudoranges come from identically distributed signals, which in real-case scenarios is not an acceptable assumption. The Weighted least squares(WLS) algorithm makes use of the noise figures of the received signals from the satellites in order to compute a more accurate solution.

In order to do this it is assumed that the pseudorange errors are Gaussian and independent, because of this the covariance matrix R can be expressed as a diagonal matrix with the pseudorange variances as the values of the diagonals. The value of Δx can then be computed as

$$\Delta x = (H^T R^{-1} H)^{-}1 H^T R^{-1} \Delta p \tag{1.31}$$

This formula weights the importance of the satellites' pseudorange in the computation depending on how noisy the received signal is, which in turn yields a more accurate pseudoinverse of the matrix H.

1.5.2 Kalman filter

The Kalman filter is an estimation algorithm. It makes use of real-time estimates of the parameters of a system, in the case of GNSS position, velocity, and time. These estimates are updated using a stream of measurements that are subject to noise. The measurements that will be used are pseudoranges with and without ionospheric correction and measurements from an IMU.

The Kalman filter uses knowledge of the deterministic and statistical properties of the system parameters and the measurements to obtain optimal estimates given the available information. It is supplied with an initial estimate of the parameters and covariance. Then it operates recursively, updating the estimates as a weighted average of the previous values and new values derived from the latest measurements.

In order to perform an optimal weighting of the estimates, a Kalman filter uses a set of uncertainties in its measurements and a measure of the correlations between the errors in the estimates of the different parameters. These values are also updated on every iteration alongside the parameter estimates. It also takes into account the uncertainties due to noise. This filter will be further discussed in the following chapters.

1.5.3 GNSS/INS integration

In the case of having an Inertial Measurement Unit (IMU) available on the receiver, it is possible to apply Inertial Navigation.

Inertial navigation has a set of advantages that are complementary to those of GNSS navigation. INS works continuously without interruption at a frequency much higher than that of GNSS navigation (about a hundred times). However, an inertial navigation solution requires an initialization and its accuracy degrades with time, as the inertial errors are integrated through the mechanization equations [8].

On the other hand, GNSS provides a high long-term position accuracy with low error values, but with a low output rate, with high short-term noise in comparison to INS, and GNSS navigation is not able to compute attitude. Finally, GNSS signals are subject to obstruction, which means they cannot be relied on to provide a continuous navigation solution.

By integrating both systems it is possible to compute a solution that is continuous, high-bandwidth, and with high long and short-term accuracy. The specifics of this integration will be discussed in more detail in the following chapters.

Chapter 2

Inertial Navigation System(INS)

An inertial navigation system is a system that provides information about the position, velocity, and attitude of a device, using measurements from inertial sensors and applying dead reckoning (DR). DR is the determination of the current variables of a system based on the previous measurements.

Having a specified set of initial conditions, it possible to determine, for example, the position and velocity of an object if we can measure the acceleration, integrating it over the time period one time for velocity and two times for position.

An INS consists of three modules: an inertial measurement unit(IMU), a pre-processing unit, and a mechanization module. The three modules are shown in figure 2.1. The IMU delivers accelerometer and gyroscope information to the signal pre-processor, which filters the signals to eliminate disturbances and sends the filtered information to the mechanization algorithm, which converts these measurements into positional and attitude information.

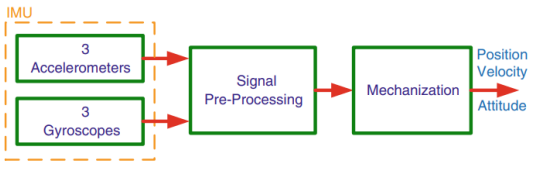


Figure 2.1: INS components[7]

2.1 Inertial Measurement Unit(IMU)

The measurements of acceleration and the rotation of the device are made by inertial sensors mounted in an IMU. The device possesses three accelerometers and three gyroscopes (one for each axis). The accelerometers measure linear acceleration while the gyroscopes measure angular velocity in three orthogonal directions. These sensor axes are fixed to the IMU's body, which means that their reference system is the b-frame. Apart from the inertial sensors, the IMU possesses electronics for calibration, sampling, and conversion of the data, as shown in figure 2.2.

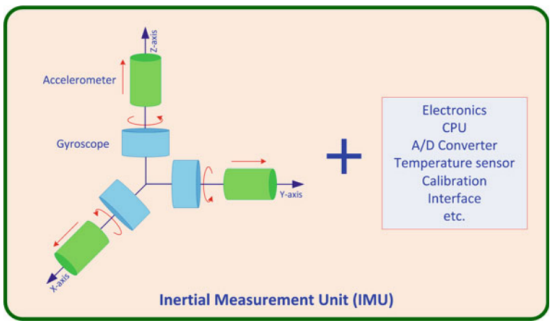


Figure 2.2: IMU[7]

2.1.1 Accelerometer measurements

An accelerometer measures translational acceleration. Its output is constantly affected by gravity, which means that its output will be equal to:

$$f = a - g \tag{2.1}$$

being a the acceleration of the body and g the gravitational vector. Acceleration a can be expressed as the second derivative of the position vector r

$$a = \frac{d^2r}{dt^2} = \ddot{r} \tag{2.2}$$

It is also important to note that the gravitational vector can be used for calibration of the IMU, given that it is approximately constant in this application[7].

2.1.2 Gyroscope measurements

Gyroscopes measure the angular velocity of a body with respect to the navigation frame, the rotation of the frame with respect to the Earth-fixed frame, and the rotation of the Earth as it spins on its axis with respect to the i-frame [7]. The measurement of the angular rate of the body frame can be then expressed as

$$w_{ib}^{b} = w_{ie}^{b} + w_{en}^{b} + w_{nb}^{b} \tag{2.3}$$

where

- w_{ib}^{b} is the rotation rate of the body with respect to the i-frame
- w_{nb}^{b} is the rotation rate of the body with respect to the navigation frame
- w_{en}^{b} is the rotation rate of the navigation frame with respect to the e-frame
- w_{ie}^{b} is the rotation rate of the Earth with respect to the i-frame

2.2 Inertial Sensor Errors

Inertial sensors are affected by a number of errors that limit the accuracy of the observable measurements. They are classified into two categories, systematic errors and stochastic errors [7].

2.2.1 Systematic Errors

These errors can be compensated by calibration of the system, the process of calibration will be discussed in future sections. The types of systematic errors are:

- Systematic Bias Offset: It is a bias offset presented in both accelerometers and gyroscopes. It is defined as the output of the sensor when its input is zero and it affects all the measurements with the same value.
- Scale factor error: Error proportional to the real value of the measured variable
- Non-linearity between input and output
- Scale Factor Sign Asymmetry
- Dead Zone: Range in which there is no output even if there is an input
- Quantization Error
- Non-orthogonality Error: This error appears when the axes of the sensors are not perfectly symmetric
- Misalignment Errors

2.2.2 Random Errors

Inertial sensors are also affected by errors that can be modeled as stochastic processes in order to mitigate them. These errors correspond to:

- Run-to-Run Bias Offset
- Bias Drift
- Scale Factor Instability
- White Noise

2.3 Calibration

Calibration is the process of comparing the instruments' outputs with a reference, with the intention of computing the coefficients that make the outputs agree with the desired output values. Calibration is used to determine and correct deterministic errors [7].

2.3.1 Six-Position Static Test

In this procedure, the device is placed with each of its axes pointing directly up or down, having 6 positions in total (two for every axis). With these sensor readings, it is possible to estimate the bias and scale factor of accelerometers and gyroscopes.

Accelerometer calibration

Accelerometers are calibrated by using gravity as a reference. In order to calibrate, each of the accelerometers is placed with its axis facing up for about 10-15 minutes. With this data, the mean value f_{up} is computed. The same process is then repeated with the axis facing down to compute f_{down}. Both of these values can be expressed as

$$f_{up} = b_a + (1 + S_a)g \tag{2.4}$$

$$f_{down} = b_a - (1 + S_a)g \tag{2.5}$$

The bias b_a is computed by adding both measurements and diving by two.

$$b_a = \frac{f_{up} + f_{down}}{2} \tag{2.6}$$

and the scale factor is obtained by subtracting the two measurements and two times the gravity and dividing by two times the gravity.

$$S_a = \frac{f_{up} - f_{down} - 2g}{2g} \tag{2.7}$$

This process is repeated for each of the three axes, to obtain the calibration values for the whole system. Having these values, the unbiased measurements of each of the accelerometers can be expressed as

$$f_{real} = \frac{f - b_a}{1 + S_a} \tag{2.8}$$

Gyroscope Calibration

The same procedure is followed for gyroscopes, using as a reference the rotation of the Earth. If we place a body frame at the surface of the Earth, with its z-axis pointing directly up, the gyroscope measures

$$\omega_z = \omega_e sin\phi \tag{2.9}$$

being ω_e the rotation rate of the Earth. Because of this, if we place one of the axes pointing up or down, the equations will be respectively

$$\omega_{up} = b_g + (1 + S_g)\omega_e sin\phi \tag{2.10}$$

$$\omega_{down} = b_g - (1 + S_g)\omega_e sin\phi \tag{2.11}$$

The bias and scale factor can be then computed as

$$b_g = \frac{\omega_{up} + \omega_{down}}{2} \tag{2.12}$$

$$S_g = \frac{\omega_{up} - \omega_{down} - 2\omega_e sin\phi}{2\omega_e sin\phi} \tag{2.13}$$

This is then repeated for each of the three axes, to obtain their respective bias and scale factors.

For low-cost gyroscopes, the axes can be rotated at a constant rate, and $\omega_e sin\phi$ is replaced by this value in the calibration.

2.3.2 Importance of calibration

As previously explained, inertial navigation is done by means of integration of the measured variables of the system. This means that calibration plays an important role in the accuracy of an INS system, given that if there exists any kind of bias or scale factor affecting the measurement, its corresponding error will grow with time.

In the case of acceleration, for example, a constant bias on its measurement will mean an error in velocity that grows linearly with time and for position an error that increases quadratically.

2.4 Navigation

As previously mentioned, the position of an object can be determined by double integration of its acceleration, sensed as a function of time in a defined reference system, and well-defined initial conditions. Mathematically it can be expressed as

$$\Delta P(t) = P(t) - P(t_0) = \int_{t_0}^{t} \int_{t_0}^{t} a(t) dt dt \tag{2.14}$$

where

- $P(t_0)$ is the initial point of the trajectory
- $a(t)$ is the acceleration measured by inertial sensors.

For navigation in three dimensions, three gyroscopes are required to measure the attitude of the body (pitch, roll, and azimuth), and three accelerometers to measure acceleration on the three axes. These measurements (and the initial conditions) can be delivered to the navigation algorithm in order to compute the PVT solution, as shown in figure 2.3.

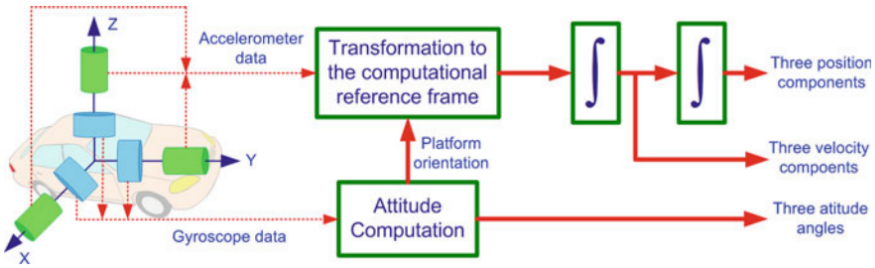


Figure 2.3: Inertial Navigation Concept[7]

For the estimation of the attitude of the device, also a digital compass can be used to aid the gyroscopes. The attitude information is provided in order

to determine the orientation of the device and, in turn, the direction in which the device is moving with respect to the ECEF frame. A complication of the measurement system is the sensed gravity by the accelerometers, which is always present in the measurements. In order to remove this component the attitude is used to compute the orientation of the device with respect to the vertical axis.

2.4.1 INS Mechanization

The process of mechanization corresponds to converting the output of the IMU (accelerations and rotation rates) and turning it into position, velocity, and attitude information. These measurements are done with respect to the i-frame, which will have to be later transformed [7].

Transformation between b-frame and l-frame

As shown in [7], the transformation between the body frame and the navigation frame can be computed using a rotation matrix R_b^l, which is calculated using the attitude values as

$$R_b^l = \begin{bmatrix} cosy\,cosr - siny\,sinp\,sinr & -siny\,cosp & cosy\,sinr + siny\,sinp\,cosr \\ siny\,cosr + cosy\,sinp\,sinr & cosy\,cosp & siny\,sinr - cosy\,sinp\,cosr \\ -cosp\,sinr & sinp & cosp\,cosr \end{bmatrix}$$

Where p, r, and y are pitch, roll, and yaw, respectively.

INS Mechanization equations

Taking into account this rotation matrix and the velocity and position estimation, the mechanization of the system of the system can be expressed as

$$\begin{bmatrix} \dot{r}^l \\ \dot{v}^l \\ \dot{R}_b^l \end{bmatrix} = \begin{bmatrix} D^{-1}v^l \\ R_b^l f^b - 2\Omega_{ie}^l v^l + g^l \\ R_b^l(\Omega_{ib}^b - \Omega_{il}^b) \end{bmatrix} \tag{2.15}$$

This result is obtained by following the process shown in [7]. In this equation the parameters are

- r^l is r the position vector in the l-frame
- v^l is r the velocity vector in the l-frame
- R_b^l is the rotation matrix to transform the values from the b-frame to the l-frame

- f^b is a vector with the measured accelerations
- Ω^b_{ib} is the skew-symmetric matrix computed from the measured angular rates.
- Ω^b_{il} is the skew-symmetric matrix that contains the rotation of the l-frame with respect to the i-frame.
- $\Omega^l_{ie}$is the skew-symmetric matrix which contains the rotation of the e-frame with respect to the i-frame, which is the rotation rate of the earth.
- D would normally transform the rectangular coordinates to curvilinear coordinates but for this application, the rectangular coordinates are kept, which means that D is the identity.

2.4.2 Computation of the parameters in the l-frame

The general procedure for the computation of the navigation parameters on the l-frame is done with the following steps:

1. Compute the attitude values of pitch, roll, and yaw (or azimuth). Also, compute the matrix R^l_b.
2. Use the value of R^l_b to transform the specific forces computed by the accelerometers.
3. Compensate for gravity and Coriolis forces (given the quality of the IMU the Coriolis forces are not compensated).
4. Compute the velocities by integrating the transformed specific forces.
5. Calculate the position value by integrating the velocities.

As it was previously stated, input data to the mechanization algorithm is calibrated before being used as part of the algorithm. For the update of the state vector, equation 2.15 is used to update the values of the velocity and position as follows:

$$\delta v^l = R^l_b \Delta t - g^l \Delta t \tag{2.16}$$

$$\delta r^l = v^l \Delta t \tag{2.17}$$

The simplification from equation 2.15 comes from the fact that the application directly provides an estimate of the attitude and that the used IMU is of very low precision, being unable to measure the Coriolis effect.

Chapter 3

GNSS Navigation Algorithms

In this chapter, the state-of-the-art navigation algorithms will be presented to be then applied in the following chapters. At first, Kalman filtering only using GNSS data is presented, to then show INS/GNSS integration.

The focus of the study is the comparison between the results obtained by using navigation algorithms versus positioning algorithms, like WLS, and also analyzing the performance of the use of the raw measurements, the corrected measurements, and the integration of INS and GNSS measurements.

3.1 Kalman Filter

As explained previously, the Kalman filter is an estimation algorithm, not exactly a filter. The technique is based on maintaining real-time estimates of a number of parameters on a system. This is done by using measurements that are functions of the estimated values and updating the previous estimates using the relation between the estimated variables.

The Kalman filter uses knowledge of the deterministic and statistical properties of the system parameters and the measurements to determine the optimal estimate given the available information.

It must be given an initial estimate of the variables, to then compute the following estimations recursively, updating the estimates as a weighted average of the model-updated previous values and new values derived from the measurement data. By being a recursive algorithm it allows the system to estimate the new values by just using the last estimate, not taking into account older estimates. This allows the system to discard old measurements, which helps with the computation.

3.1.1 Kalman filter model

First of all, it is assumed that the errors of the system can be modeled as systematic, white noise, or Gauss-Markov processes. They also may be linear combinations of these or integrations of them. Other types of errors have to be accounted for as variables in the model, in order to be estimated and subtracted from the actual variables that wanted to be estimated.

The set of parameters estimated by a Kalman filter is denoted by x and is known as the state vector. The Kalman filter estimate of the state vector is denoted by $\hat{x}$.

The estimation of the absolute properties of the system is known as total-state estimation, while the estimation of the errors in a measurement is known as an error-state implementation.

The state vector residual, denoted by δx, is the difference between the true state vector and the Kalman filter estimates.

$$\delta x = x - \hat{x} \tag{3.1}$$

In an error-state implementation, this vector represents the errors remaining in the system after the Kalman filter estimation has been used to correct them.

The error covariance matrix, denoted by P, defines the expectation of the square of the deviation of the state vector from the true value of the state vector. It is computed as

$$P = E((\hat{x} - x)(\hat{x} - x)^T) = E(\delta x \delta x^T) \tag{3.2}$$

The diagonal elements are the variances of the estimates, while the non-diagonal estimates are the covariances between the estimates. The values on the matrix can be expressed as

$$P_{ij} = P_{ji} = \sigma_i \sigma_j \rho_{ij} \tag{3.3}$$

where ρ_{ij} is the correlation coefficient, a value between 0 and 1 that depends on the correlation between the estimates. Note that for $i = j$, $\rho_{ij} = 1$.

In an error-state implementation, the initial state estimates are set to zero. On the other hand, in a total-state estimation, the initial values of the estimates may be set by the user or by using past estimates from when the system was previously used.

Given the definitions of the state vector and the error covariance matrix, it is important to distinguish between the values after the iteration of the Kalman filter and the ones used in the step between propagation and update. The time-propagated state estimates and covariance matrix are denoted by $\hat{x}_k^-$ and P_k^-, while the values using the measurement update are denoted by $\hat{x}_k^+$ and P_k^+.

The measurement vector is denoted by z and is the set of measurements of the system described by the state vector. The measurements can be range measurements or the difference between two navigation systems(in the case of loose integration for example). These measurements are modeled as

$$z = h(x) + w_m \tag{3.4}$$

being $h(x)$ a deterministic function that links the measurements to the state vector and w_m the measurement noise. This function $h(x)$, is normally linearized which returns a matrix H (measurement matrix), which relates the values of the state vectors x to the measurements. The relation is then

$$z = Hx + w_m \tag{3.5}$$

The measurement innovation, denoted by δz^- is the difference between the true measurement vector and the one computed from the state vector estimate prior to the measurement update. Thus, it is computed as

$$\delta z^- = z - h(\hat{x}^-) \tag{3.6}$$

In a real case application, it can be the difference between a set of measurements and a set of predicted measurements using the Kalman filter estimation. The measurement residual δz^+ is the difference between the true measurement vector and the vector computed from the updated state vector.

$$\delta z^+ = z - h(\hat{x}^+) \tag{3.7}$$

The standard Kalman filter assumes that these measurement errors form a Gaussian zero-mean distribution, that is uncorrelated in time and models their standard deviation with the measurement noise covariance matrix, denoted as R. The matrix is defined by the square of the measurement noise.

$$R = E(w_m w_m^T) \tag{3.8}$$

As well as in the P matrix, the values in the diagonal are the variances of the measurements, while the non-diagonal terms represent the correlation between different components of the noise. In general, the R matrix is diagonal, given that in most navigation applications the noise on each component is independent.

The transition matrix, denoted by Φ, defines how the state vector is updated with time given the dynamics of the system that is being modeled. This relation is

$$\hat{x}_k^- = \Phi_{k-1}\hat{x}_{k-1}^+ \tag{3.9}$$

Finally, the system noise covariance matrix, which is denoted by Q_{k-1}, defines how the uncertainties of the state estimates increase with time due to unknown changes in the values of the states, because of instrument noise for example.

3.1.2 Kalman filter algorithm

The Kalman filter algorithm is comprised of the following steps [8]:

1. Calculate the transition matrix, Φ_{k-1}
2. Calculate the system noise covariance matrix, Q_{k-1}
3. Propagate the state vector estimate from $\hat{x}_{k-1}^{+}$ and $\hat{x}_{k}^{-}$
4. Propagate the error covariance matrix from P_{k-1}^{+} and P_{k}^{-}
5. Calculate the measurement matrix, H_k
6. Calculate the measurement noise covariance matrix, R_k
7. Calculate the Kalman gain, K_k
8. Formulate the measurement, z_k
9. Update the state vector estimate from $\hat{x}_{k}^{-}$ to $\hat{x}_{k}^{+}$
10. Update the error covariance matrix from P_{k}^{-} to P_{k}^{+}

The first step is the calculation of the transition matrix Φ_{k-1}. In the case of GNSS the transition matrix is constant, given that the physical model that describes the estate estimate does not change with time.

Step 2 is the calculation of the system noise covariance matrix, Q_{k-1}. Depending on the application it can be modeled as either time-varying or constant and it depends on the used receiver's characteristics.

Step 3 is the propagation of the state vector estimate using

$$\hat{x}_{k}^{-} = \Phi_{k-1}\hat{x}_{k-1}^{+} \tag{3.10}$$

Step 4 is the propagation of the error covariance matrix. The standard form of this propagation is

$$P_{k}^{-} = \Phi_{k-1}P_{k-1}^{+}\Phi_{k-1}^{T} + Q_{k-1} \tag{3.11}$$

Step 5 is the calculation of the measurement matrix H_k. In a standard Kalman filter each measurement is assumed to be a linear function of the state vector, this means that $h(x)$ can be expressed as

$$h(x_k, t_k) = H_k x_k \tag{3.12}$$

In most applications the measurement matrix varies, so it has to be updated on each iteration of the Kalman filter. In navigation, it is a function of the kinematics and geometry of the transmitters.

Step 6 is the computation of the measurement noise covariance matrix, R_k. For most GNSS applications it can be computed as a function of signal-to-noise measurements.

Step 7 is the calculation of the Kalman gain matrix, K_k. This matrix is used to determine the weight of the measurement information in the update of the state vector. This matrix is computed as:

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \tag{3.13}$$

Step 8 is the formulation of z_k, the system measurements. In some applications(like GNSS navigation), the measurement vector is already present in the system model, but in other cases, the vector must be computed as a function of other system parameters.

Step 9 is the update of the state vector with the measurement vector

$$\hat{x}_k^+ = \hat{x}_k^- + K_k(z_k - H_k \hat{x}_k^-) = \hat{x}_k^- + K_k \delta z_k^- \tag{3.14}$$

Finally, step 10 is the update of the error covariance matrix

$$P_k^+ = (I - K_k H_k) P_k^- \tag{3.15}$$

Given that the state vector estimate is based on more information than before, the updated state uncertainties become smaller. The complete process is summarized in figure 3.1

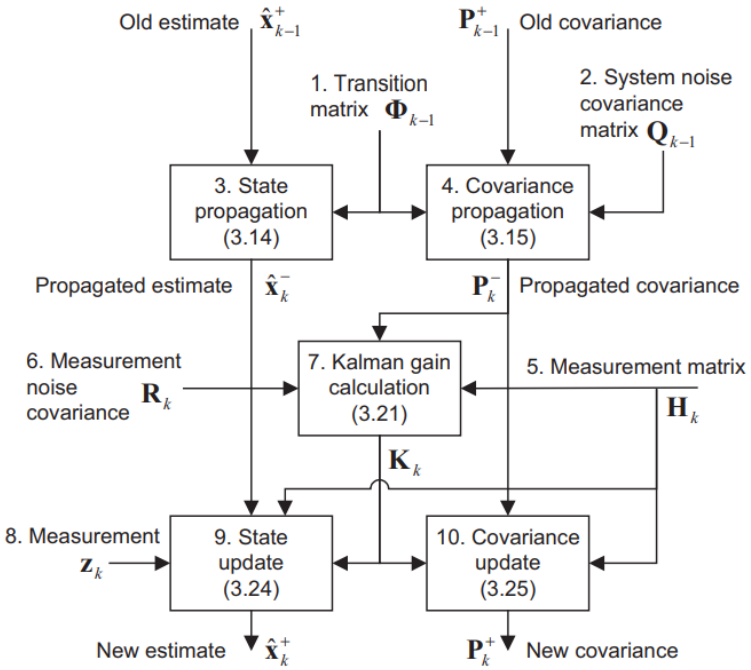


Figure 3.1: Kalman filter data flow[8]

3.1.3 GNSS Kalman filter

In this section, the Kalman filter utilized for this application is shown. It is important to note that the filter is applied on the ECEF coordinate system, to then transform these coordinates to longitude, latitude, and height.

First of all, the state vector x is composed of the position (x, y, z), the pseudorange bias(b), the velocities $(\dot{x}, \dot{y}, \dot{z})$ and the change of bias rate $(\dot{b})$. While the measurements vector z correspond to the pseudorange measurements, which can be at least 4, but can be more for a more precise result.

The vectors are then defined as

$$x = \begin{bmatrix} x \\ y \\ z \\ b \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{b} \end{bmatrix} \qquad z = \begin{bmatrix} \rho_1 \\ \rho_2 \\ \rho_3 \\ \rho_4 \\ \dots \end{bmatrix} \tag{3.16}$$

As stated in previous sections the measurement matrix corresponds to

$$H = \begin{bmatrix} a_{x_1} & a_{y_1} & a_{z_1} & 1 & 0 & 0 & 0 & 0 \\ a_{x_2} & a_{y_2} & a_{z_2} & 1 & 0 & 0 & 0 & 0 \\ a_{x_3} & a_{y_3} & a_{z_3} & 1 & 0 & 0 & 0 & 0 \\ a_{x_4} & a_{y_4} & a_{z_4} & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & a_{x_1} & a_{y_1} & a_{z_1} & 1 \\ 0 & 0 & 0 & 0 & a_{x_2} & a_{y_2} & a_{z_2} & 1 \\ 0 & 0 & 0 & 0 & a_{x_3} & a_{y_3} & a_{z_3} & 1 \\ 0 & 0 & 0 & 0 & a_{x_4} & a_{y_4} & a_{z_4} & 1 \end{bmatrix} \tag{3.17}$$

Knowing that the values of position are independent and that the second half of the variables are the derivatives of the first half, then it is fairly simple to conclude that the transition matrix in this case is expressed as

$$\Phi = \begin{bmatrix} 1 & 0 & 0 & 0 & \tau & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & \tau & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \tau & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \tau \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.18}$$

Being τ the time step between two different GNSS raw measurements.

For the value of the error covariance matrix P, it is only needed to give a value to the first iteration of the matrix P_0. Given that the exact value of the matrix is not available, this matrix is approximated as a diagonal matrix, assuming that the state vector variables are independent, with values taken from a standard GNSS receiver shown in [9]. It is important to note that this value can be changed as the Kalman filter is tuned, in order to achieve

On the other hand, the noise covariance matrix R can be approximately computed because GNSS receivers give an estimate of the variance of the computed pseudoranges. Having these values, R is computed as a diagonal matrix with the values of the estimated variances on the diagonal.

Finally, the system noise covariance matrix Q is taken from the example in [9], showing a standard GNSS receiver. This value is the one that is most used when tuning the Kalman filter, given that it intuitively changes the value of the Kalman gain, giving more importance to the kinematic model or the measurements. Subsequent chapters will delve deeper into this attribute of the Q matrix.

3.2 GNSS/INS integration

As explained in previous chapters the pros and cons of INS GNSS navigation are complementary, so it makes sense to integrate them in the same system. INS provides a high data frequency with good short-term accuracy and attitude information in addition to position and velocity. The problem is that for INS the long-term errors grow without bounds, given that previous errors are integrated into future estimates.

In contrast to INS, GNSS navigation provides good long-term accuracy but at a lower data frequency and a higher short-term accuracy. Also, GNSS navigation needs a direct line of sight to at least 4 satellites to be able to estimate the position, which is not always possible due to possible obstruction from other objects.

By exploiting their complementary characteristics, their integration overcomes their individual drawbacks and provides a more accurate and robust navigation solution than either could achieve on its own. For the integration of both systems optimal estimation techniques are used, normally Kalman filtering. GNSS prevents the inertial solution from drifting, while INS provides continuity in the navigational solution, attitude information, and bridges between points in which GNSS signals are not received. A typical INS/GNSS integration is depicted in figure 3.2.

The optimal estimator compares the outputs of the INS and GNSS and estimates errors in inertial position, velocity, and attitudes. The output is corrected using the estimated errors to produce the integrated navigation solution [7].

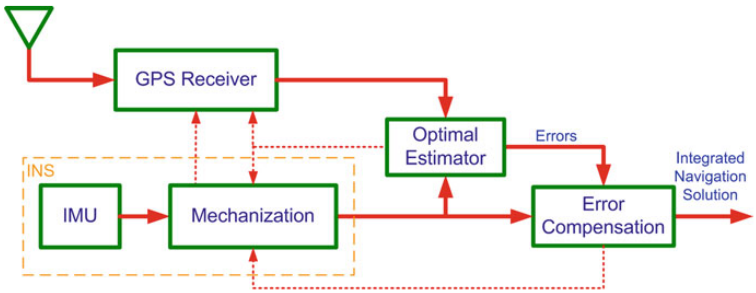


Figure 3.2: Typical INS/GNSS integration[7]

The architecture of an INS/GNSS integrated navigation system varies in three respects: how corrections are applied to the inertial navigation solution, what types of GNSS measurements are used, and how the INS and integration algorithm aids the GNSS user equipment.

The most widely used definitions for integration are

- Loosely coupled INS/GNSS integration

- Tightly coupled INS/GNSS integration
- Ultratightly coupled INS/GNSS integration

In all of these cases, the integrated solution is the corrected inertial navigation solution. In a conventional integration architecture using an error-state Kalman filter and separate inertial navigation processing, correction may be either open-loop or closed-loop.

In the case of open-loop architecture the estimated position, velocity, and attitude errors are used to correct the inertial navigation solution in the correction algorithm, but the results of the corrections are not fed back to the INS. Because of this, the integrated solution contains the Kalman filter estimates, while raw INS solutions are available for corrections. The complete process is shown in figure 3.3.

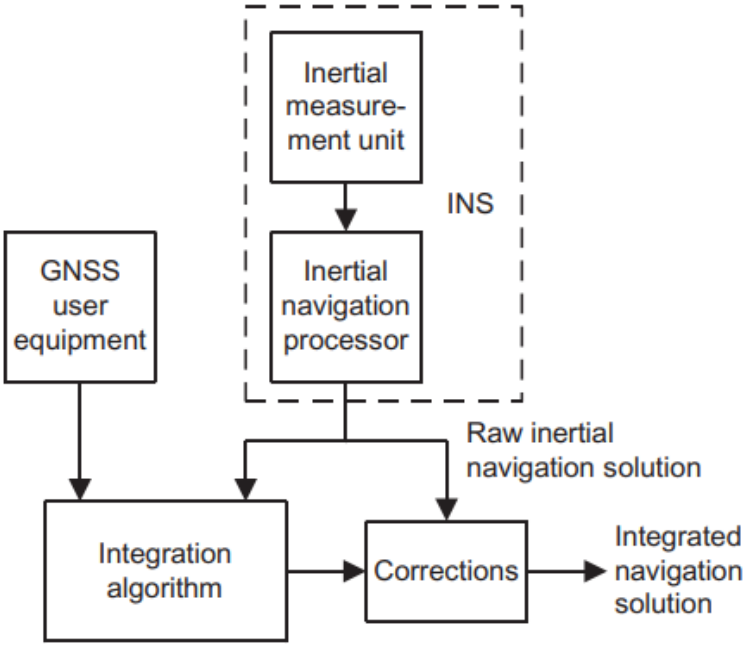


Figure 3.3: Open-loop INS correction architecture.[8]

In closed-loop correction architecture, GNSS is used to aid INS through the integration block. In this case, the estimated position, velocity, and attitude errors are fed back to the inertial navigation processor, where they are used to correct the INS solution. The feedback can be done on each iteration of the filter or at a lower frequency. These error estimations are zeroed after they are fed back. The architecture is shown in figure 3.4.

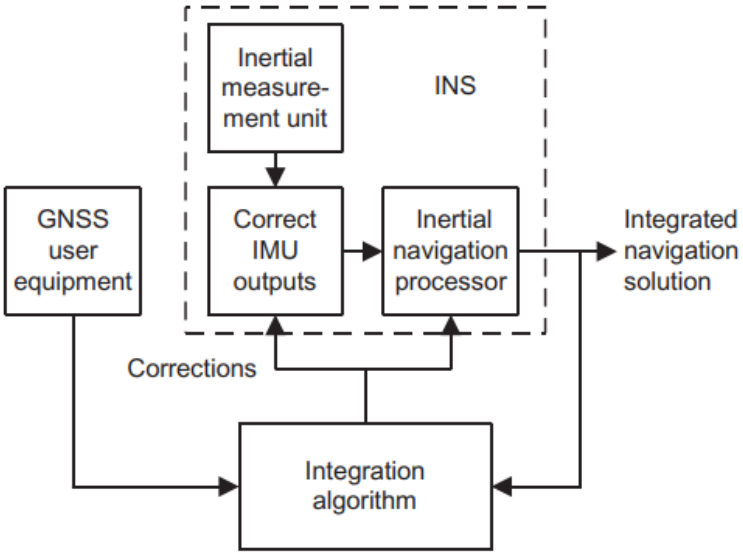


Figure 3.4: Closed-loop INS correction architecture.[8]

In the following sections the three previously presented types of integration will be shown, It can be noted that for all of these cases, the integration can be open-loop or closed-loop.

3.2.1 Loosely Coupled INS/GNSS Integration

In this type of integration, GNSS and INS operate independently and provide separate navigation solutions. Both of these solutions are fed to the integration Kalman filter and are used to determine the INS errors. The INS solution is corrected for these errors to produce the integrated navigation solution, given that it works at a higher frequency than the GNSS solution. The defining characteristic is the use of a separate filter for the GNSS data. A general diagram of loose integration is shown in figure 3.5.

Loosely coupled integration is the simplest to implement and is robust. The problem is that it is unable to provide GNSS-aiding when the number of on-sight satellites falls below 4.

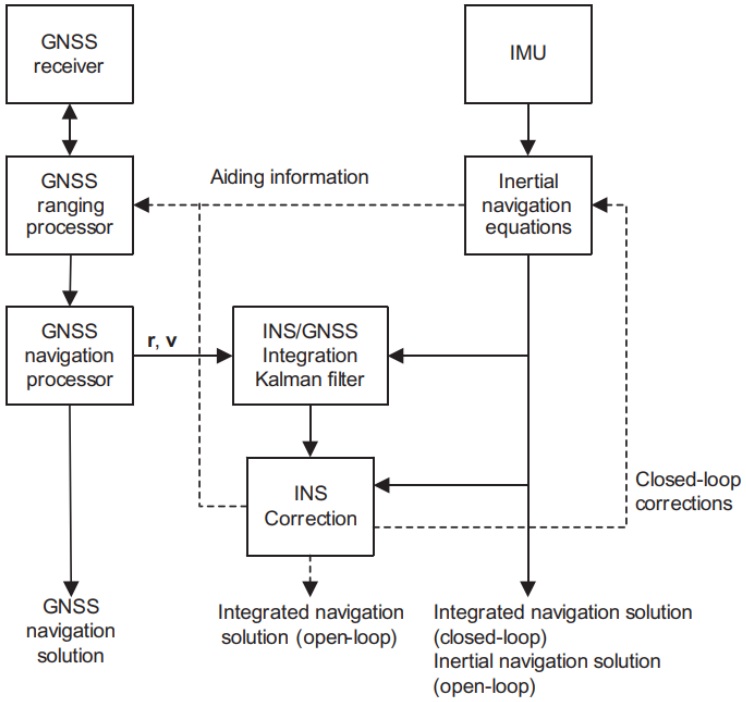


Figure 3.5: Block diagram of loosely coupled integration[8]

3.2.2 Tightly coupled integration

Figure 3.6 shows a tightly coupled INS/GNSS integration architecture. In this architecture, the GNSS navigation processor(which computes the position and velocity) is bypassed, using directly the values of the pseudoranges and pseudorange rates as inputs to the integration Kalman filter, as well as predictions of these values made from internal navigation equations, that are estimated with the IMU output values. Both of these values are then used to compute the errors in the INS, which are fed to the integration Kalman filter and then corrected with the higher frequency INS data, resulting in the integrated navigation solution. In a closed-loop model, the output of the INS corrections is fed back to the inertial navigation equations and the GNSS ranging processor, for further correction of the processed data.

This tightly coupled architecture has a number of advantages when compared with its loosely coupled counterpart. By having a centralized Kalman filter, possible cascading problems are eliminated. Also, the GNSS input is used to aid the INS when insufficient satellite signals are being received. Finally, having the same inertial measurements and the same GNSS equipment, a tightly coupled INS/GNSS integration should be more accurate and robust than the loosely coupled option.

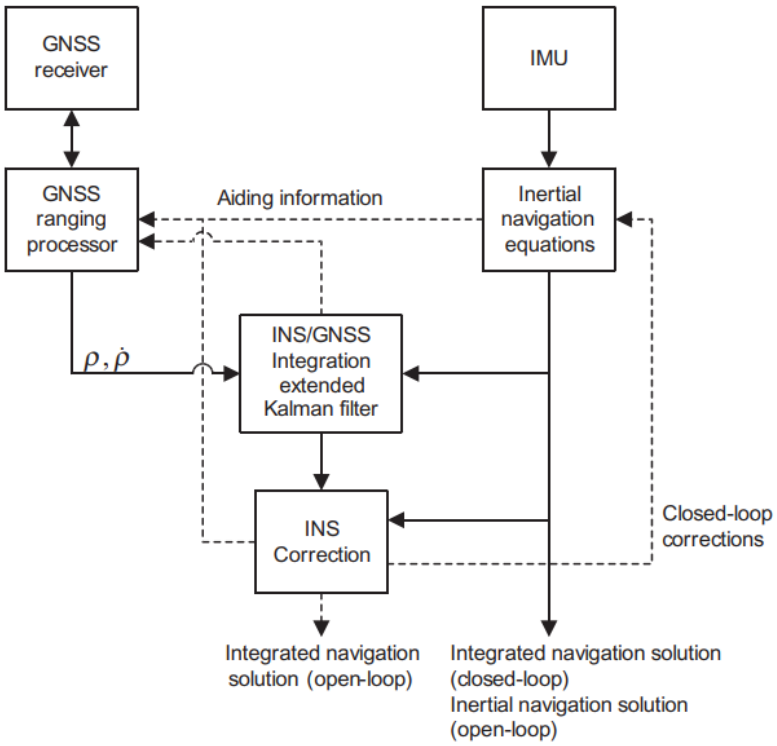


Figure 3.6: Block diagram of tightly coupled integration[8]

3.2.3 Ultra-tight integration

This kind of integration also called deep integration, is shown in figure 3.7. In this integration architecture, the information from the INS is used as an integral part of the GNSS receiver, meaning that the INS and GNSS are no longer independent navigators. This architecture uses INS navigation parameters to improve the raw measurements, obtaining a more precise solution than the previous architectures.

Compared with tightly coupled integration, deeply coupled lowers the tracking bandwidth and is more resistant to jamming. This type of integration can also work at lower C/N_0 and provides a solution even when the number of on-sight satellites falls below four.

The main limitation of this architecture is its complexity and the fact that it has to track the GNSS parameters. In comparison with the other architectures, it is not convenient to use on simple applications, like for example the focus of this thesis which is mobile devices.

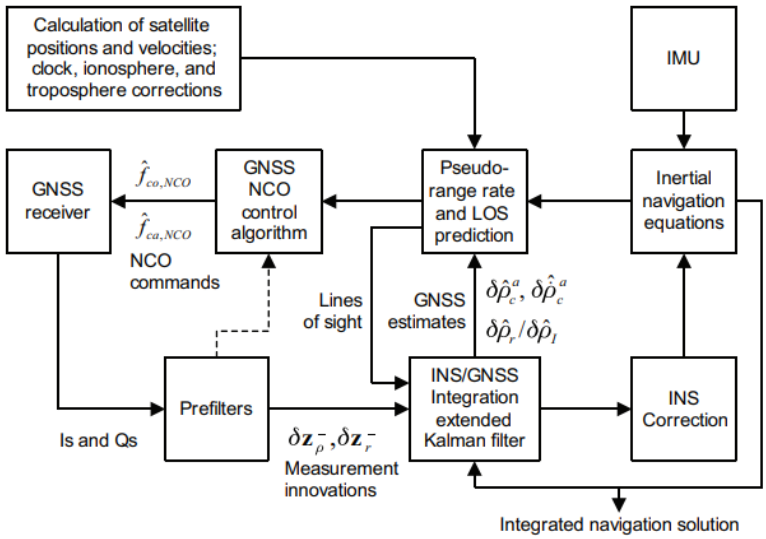


Figure 3.7: Block diagram of ultra-tight integration[8]

3.3 Integration algorithm

In this specific case, we will be using the loosely coupled integration, given that it is the simplest one to apply and is robust, giving accurate solutions even when a GNSS PVT solution is not provided, being because of a lack of reception or not enough satellites being in sight.

In this section, we will explain how to model and apply this architecture to obtain an INS/GNSS integrated solution. At first, the coordinate transformation equations will be presented, to then show the integration Kalman filter model.

3.3.1 System Model

The system model of the integrated KF for loosely coupled integration is given by

$$\delta\dot{x} = F\delta x + Gw \tag{3.19}$$

The state vector is conformed by error components of position, velocity, attitude, acceleration, and angular rate.

$$\delta x_{15x1} = [\delta r_{3x1}, \delta v_{3x1}, \epsilon_{3x1}, \delta\omega_{3x1}, \delta f_{3x1}]^T \tag{3.20}$$

where

- $\delta r = [\delta x, \delta y, \delta z]^T$ is the position error vector.

- $\delta v = [\delta v_x, \delta v_y, \delta v_z]^T$ is the Earth-referenced velocity error vector
- $\epsilon = [\delta p, \delta r, \delta A]^T$ is the attitude error vector
- $\delta\omega = [\delta\omega_x, \delta\omega_y, \delta\omega_z]^T$ is the gyroscope error vector
- $\delta f = [\delta f_x, \delta f_y, \delta f_z]^T$ is the accelerometer error vector

w is a vector of unit variance white Gaussian noise. On the other hand, G is the noise distribution vector, which has the variances associated to the state vector

$$G = [\sigma_{r,1x3}, \sigma_{v,1x3}, \sigma_{\epsilon,1x3}, \sigma_{\omega,1x3}, \sigma_{f,1x3}] \tag{3.21}$$

The term F contains the INS error models for the position, velocity, attitude, and inertial sensors' outputs. F can be written as

$$F = \begin{bmatrix} 0_{3x3} & F_r & 0_{3x3} & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & F_v & 0_{3x3} & R_b \\ 0_{3x3} & F_\epsilon & 0_{3x3} & R_b & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & 0_{3x3} & F_\omega & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} & F_f \end{bmatrix} \tag{3.22}$$

The definition of the submatrices can be found on [7]. Given these matrices, the discrete form of equation 3.21 can be written as

$$\delta x_k = (I + F\Delta t)\delta x_{k-1} + G\Delta t w_{k-1} \tag{3.23}$$

3.3.2 Measurement Model

The equation that describes the measurement model is expressed as

$$\delta z_k = H_k \delta x_k + \nu_k \tag{3.24}$$

Given that δx_k contains the errors in the INS, the measurement vector δz_k corresponds to the difference between position and velocity values computed by the INS mechanization and the ones computed by the GNSS KF

$$\delta z_k = \begin{bmatrix} r_{INS} - r_{GNSS} \\ v_{INS} - v_{GNSS} \end{bmatrix} = \begin{bmatrix} x_{INS} - x_{GNSS} \\ y_{INS} - y_{GNSS} \\ z_{INS} - z_{GNSS} \\ v_{x,INS} - v_{x,GNSS} \\ v_{y,INS} - v_{y,GNSS} \\ v_{z,INS} - v_{z,GNSS} \end{bmatrix} \tag{3.25}$$

ν_k is a measurement noise vector with zero mean and covariance R_k. H_k is the measurement matrix at time t_k and, as described in previous chapters, it describes the relation between the measurements and the state variables in the absence of noise. Given this type of architecture, H_k takes a simple form, being expressed as

$$H = \begin{bmatrix} I_{6x6} & 0_{6x9} \end{bmatrix} \tag{3.26}$$

For the covariance matrices, the measurement covariance matrix R_k is expressed as

$$R_k = \begin{bmatrix} \sigma_x^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_y^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_z^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{v_x}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{v_y}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{v_z}^2 \end{bmatrix} \tag{3.27}$$

On the other hand, the prediction covariance matrix P_k has the variances of the predicted states on its diagonal. The rest of the values are cross-correlation values between the other state values. The matrix can be expressed as

$$R_k = \begin{bmatrix} \sigma_{r,3x3}^2 & . & . & . & . \\ . & \sigma_{v,3x3}^2 & . & . & . \\ . & .\sigma_{\epsilon,3x3}^2 & . & . & \\ . & . & . & \sigma_{\omega,3x3}^2 & . \\ . & . & . & . & \sigma_{f,3x3}^2 \end{bmatrix} \tag{3.28}$$

Having these equations it is possible to apply the Kalman filter in the same way as it was described in section 3.1. The full implementation of the loosely coupled INS/GNSS integration is shown in figure 3.8

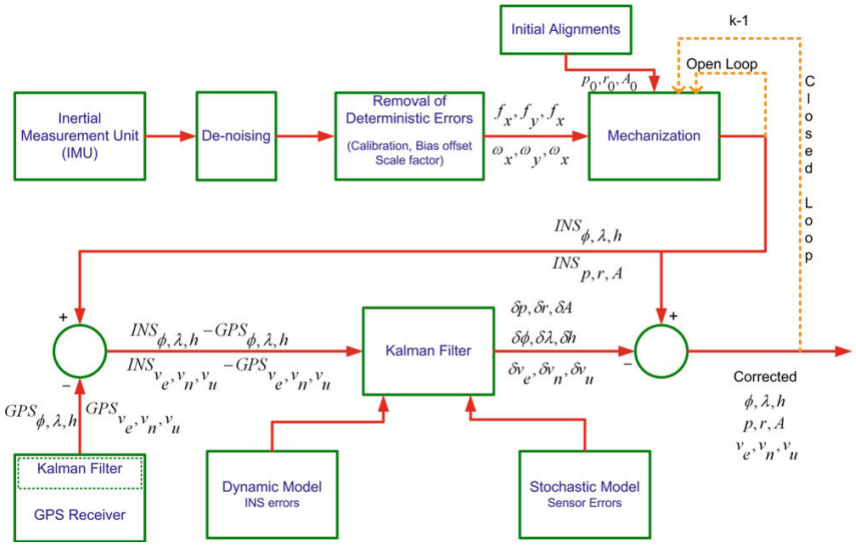


Figure 3.8: Complete Loosely Coupled Integration implementation[7]

Chapter 4

Software Development

By using the content studied in previous chapters related to advanced navigation algorithms, the finality of this study is to be able to apply the algorithms to a set of real data. To this end, data from GNSS and INS data provided by smartphones has been used in order to apply and compare the different algorithms.

These measurements have been provided by the Android application GNSSLogger, which provides information on the pseudoranges from different constellations and inertial data from the inertial measurement unit(IMU) of the phone, while the ephemeris information has been supplied by RINEX files available on [10]. This ephemeris information is computed by different base stations across Italy.

Having the files that contain the measurements, these are then processed by Matlab and used to compute the user's PVT with the different presented algorithms.

In this chapter the methods of acquiring and processing the data will be shown, to then explain the application of this data on various algorithms and, in the next chapter, show the results of these on the different cases.

4.1 Data acquisition

The first step of the process is data acquisition. Not all smartphones are capable of providing GNSS raw measurements and different phones provide different levels of information about navigation. Some examples of phones that provide GNSS measurements are shown on [11]. For this purpose, a Samsung Galaxy S20 was used for acquisition, apart from providing raw GNSS measurements, it is also equipped with an L5 receiver and it is capable of detecting multiple GNSS constellations. Normally, phones on the market are only equipped with L1-only GNSS chipsets, now having L1 and L5 as available frequencies it is possible to apply the ionospheric corrections that were discussed in a previous chapter.

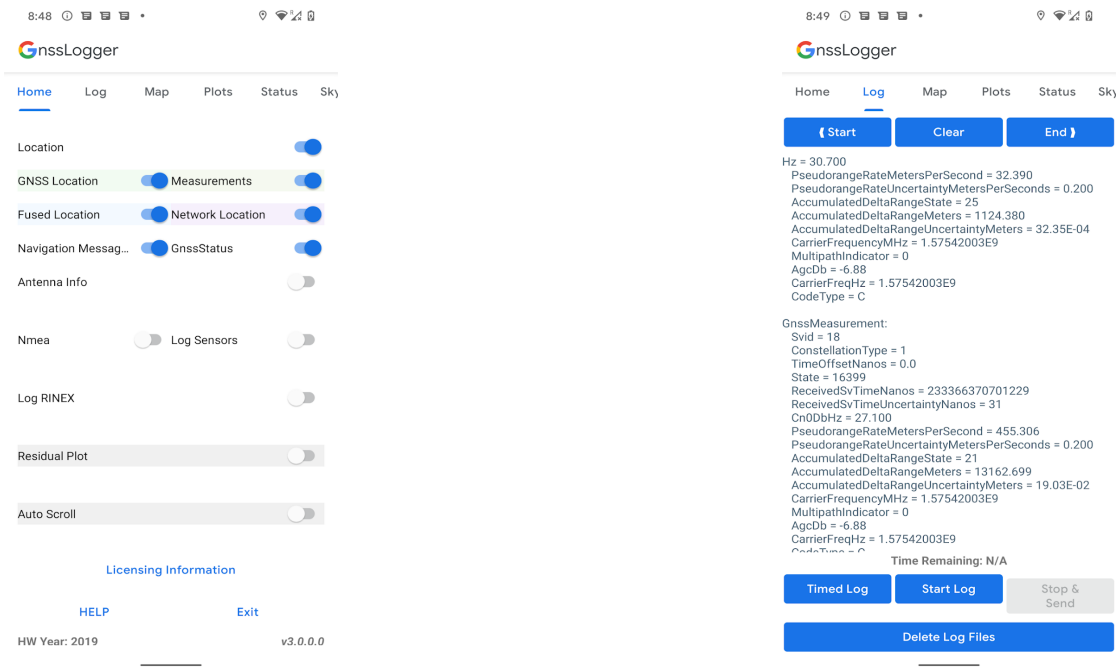
It is important to note that most of the information related to the specifications

of the various sensors found in smartphones is not available to the public, given the privacy policies of the manufacturers. This means that some of the obtained noise values have to be estimated and are not the exact ones.

4.1.1 GNSSLogger

GNSSLogger is an Android application that provides users with measurements. Among the most important variables are the pseudoranges and pseudorange rates, accompanied by their respective uncertainty measurements. Also, on the latest versions of the application, it also provides IMU measurements, which are used for INS/GNSS integration[11].

The application looks as it is shown in figures 4.1a and 4.1b. In order to get GNSS information, the user has to select the desired outputs. For this case, the information used for processing are the location measurements and the log sensors. After selecting these options, the user has to change to the log window and press "Start Log". The user can then stop the log at any time by pressing "Stop & Send" which will give the user the possibility to save the information.



(a) Home window **(b)** Log window

Figure 4.1: GNSSLogger app

All the navigation data is saved as a text file, having one measurement for

each line, with its corresponding important values. This text file will have to be processed in order to obtain the data required by GNSS navigation algorithms. The main pseudo-range related measurements provided by the application are shown on 4.1, which is originally taken from [12].

FIELD	DESCRIPTION
TimeNanos	GNSS receiver's internal hardware clock value in nanoseconds
BiasNanos	Clock's sub-nanosecond bias
FullBiasNanos	Difference between TimeNanos inside the GPS receiver and the true GPS time since 0000Z, 6 January 1980
DriftNanosPerSecond	Clock's drift
HardwareClockDiscontinuityCount	Count of hardware clock discontinuities
LeapSecond	Leap second associated with the clock's time
ConstellationType	Constellation type
Svid	Satellite ID
State	Current state of the GNSS engine
ReceivedSvTimeNanos	Received GNSS satellite time at the measurement time
AccumulatedDeltaRangeMeters	Accumulated delta range since the last channel reset
Cn0DbHz	Carrier-to-noise density
TimeOffsetNanos	Time offset at which the measurement was taken in nanoseconds
CarrierCycles	Number of full carrier cycles between the satellite and the receiver
CarrierFrequencyHz	Carrier frequency at which codes and messages are modulated
PseudorangeRatemetersperSecond	Gets the Pseudorange rate at the timestamp

Table 4.1: GNSS data delivered by GNSSLogger

On the other hand, for inertial navigation, in its latest versions, GNSSLogger provides the user with accelerometer, gyroscope, and attitude data. The available data is shown on 4.2.

FIELD	DESCRIPTION
utcTimeMillis	UTC time in milliseconds
elapsedRealTimeNanos	internal hardware clock value in nanoseconds
AccelMps2	Acceleration in m/s^2
GyroRadPerSec	Angular rate in rad/s
OrientationDeg	Attitude data (yaw, roll, pitch)

Table 4.2: INS data delivered by GNSSLogger

4.1.2 RINEX files

As previously mentioned, the RINEX files are available on the "Regione Autonoma Friuli Venezia Giulia" webpage[10]. The data contained in these files is acquired by several base stations across Italy. These files are essential to the correct functioning of the navigation algorithms because they provide the information to estimate the satellite's position, which is then used in the navigation algorithms as a reference to compute the user's position and velocity.

The files have different names depending on the time the acquisition started, so the user must make sure that the downloaded RINEX file is on the correct date and time. The used name convention for the flies is "ssssdddf.yyt", in this convention, the first four digits of the name of the file are used as a designator of the name of the station from which the data is taken and the next three digits correspond to the day of the year. In the extension of the file, the first two digits show the year of the creation of the file and the last digit is the type of file. For this application, the used files are navigation files, the letter in that case depends on the used constellation. The ones available for use in this script are GPS and Galileo, with l for GPS and n for Galileo.

The files must be placed in the same folder as the GNSSLogger file, in order to be read by the script. When running the script section by section, before the program tries to read the RINEX file, it prints the required date and time that is required by the algorithm in the format "YYYY MM DD hh mm ss", which refers to the year, month, day, hour, minutes and seconds. Having this information, the corresponding file has to be searched in [10] and downloaded to be placed with the rest of the files.

4.2 Data processing

The first used function is called "ReadGnssLogger()", it receives the path of the file and its name. This file takes the names of all the parameters of the GNSS measurements and makes them the fields of a structure called "gnssRaw". From this structure it is possible to get information about the time of the measurements, so we can match the measurements with the required RINEX file.

This data is then further processed by the "ProcessGnssMeas()" function, which receives the GnssRaw structure. This function aims to process the information presented on the “gnssRaw” structure, organizing it in a more concise and useful structure and filtering useless information, affected too much by noise or errors in the acquisition.

This function also computes the most important value for the PVT computation, which are the values of the pseudoranges. It does this by using the values of the values of the transmitted and received time, also subtracting from this time the

bias and offset that are computed by the gnssLogger app.

4.3 PVT calculations

Having the pseudorange and satellite information structured in a convenient way, it is then possible to compute the PVT solution of the acquired data. This can be done with the previously presented algorithms. For each one of the different algorithms, a different function for processing the information has been developed.

4.3.1 WLS

The created function for the WLS PVT computation is called "GpsWlsPvt" and it receives the "gnssMeas" structure and the ephemeris information.

First of all, the initial value of the state vector(called in this case xo), is defined. Given that there is not any available information about the initial position of the user, the state vector is a vector of zeros.

Having an initial value, the algorithm iterates over the time stamps and checks the available satellites, saving the value of the pseudorange, pseudorange standard deviation, pseudorange rates, and pseudorange rates standard deviation for their later use.

In case that less than 4 satellites are available at a given time, the algorithm skips to the next time step. On the contrary, when there are enough satellites for the computation, the function "WlsPvt" is called, this function receives the pseudoranges for this iteration, the ephemeris, and the state vector and outputs an error state vector which is later added to the state vector.

The function "WlsPvt" first uses the values of the standard deviations to generate the weighted matrices W_{pr} and W_{rr}. It then starts a while loop that iterates over the value of the error state vector, called dx in this case. The while loop is exited once the value of dx is small enough, meaning that the PVT computation is close to the actual result. The process of computing the PVT solution for one measurement is shown in listing 4.1 and it is done as explained in section 1.5.1.

Listing 4.1: WLS code

```
while norm(dx) > threshold
    v = xyz0 - svXyzTrx;% vector from xo to satellite
    range = sqrt( sum(v.^2) );
    v = v./(ones(3,1)*range); % line of sight unit vectors from sv to
     xo

    %calculate the estimated range error
    prHat = range(:) + bc - LIGHTSPEED*dtsv;

```

```
    zPr = pr-prHat;
    H = [v', ones(numVal,1)]; % H matrix = [unit vector,1]

    % We use the weighted pseudo inverse to compute dx
    dx = pinv(Wpr*H)*Wpr*zPr;

    % update xo, xhat and bc
    xHat=xHat+dx;
    xyz0=xyz0(:)+dx(1:3);
    bc=bc+dx(4);

    %Now calculate the a-posteriori range residual
    zPr = zPr-H*dx;
end
```

Having the desired values for the position, it is also possible to compute the values of the velocity states and the velocity bias. Both codes are consecutive, and the output of the whole operation is $xHat$, which is then added to the state vector.

Listing 4.2: Velocity update

```
rrMps = zeros(numVal,1);
for i=1:numVal
    %range rate = [satellite velocity] dot [los from xo to sv]
    rrMps(i) = -svXyzDot(i,:)*v(:,i);
end
prrHat = rrMps + xo(8) - LIGHTSPEED*dtsvDot;
zPrr = prr-prrHat;
%z = Hx, premultiply by W: Wz = WHx, and solve for x:
vHat = pinv(Wrr*H)*Wrr*zPrr;
xHat = [xHat;vHat];

z = [zPr;zPrr];
```

It is important to note that these operations are done on the ECEF coordinate system and are then transformed to the geographic coordinate system to be plotted, given that the Matlab functions for position plotting use this reference system's coordinates.

4.3.2 Kalman filter

This filter follows a similar procedure to the one described for WLS. The function "GpsKalmanPvt" is the one used in this case for the PVT computation and it is very similar to "GpsWlsPvt". The state vector has the same characteristics and it also checks for the availability of at least 4 satellites. For the first iteration of the algorithm, in order to get an initial state vector for the Kalman filter, the function "WlsPvt" is called. After getting this initial position, the rest of the iterations are

done using the "KalmanPvt" function.

This function receives the same values as the "WlsPvt" function in addition to an error covariance matrix(P) that is updated on each iteration and the time step(δt) that will be used for the Kalman filter. In order to compute the Kalman filter estimate, the values of the noise covariance matrix(Q), the transition matrix(transMat), and the measurement covariance matrix(R) are defined. The transition matrix is defined as shown in 3.1, while the value of the noise covariance matrix is determined with trial and error, given that the manufacturers do not always provide the noise figures of their devices. Finally, the matrix R is defined using the standard deviation values provided by the navigation data.

The update of the state vector is done as shown in listing 4.3.

Listing 4.3: Kalman filter code

```
%calculate line of sight vectors and ranges from satellite to xo
v = xyz0 - svXyzTrx;
range = sqrt( sum(v.^2) );
v = v./(ones(3,1)*range); % line of sight unit vectors from sv to xo

prHat = range(:) + bc -LIGHTSPEED*dtsv;

zPr = prs-prHat;

% Compute velocities
rrMps = zeros(numVal,1);
for i=1:numVal
    rrMps(i) = -(svXyzDot(i,:))*v(:,i);
end

prrHat = rrMps + bc_dot - LIGHTSPEED*dtsvDot;

zPrr = prs(:,jPrr) - prrHat;

dz = [zPr;zPrr];

h_pre = [v', ones(numVal,1)];

H = [h_pre, zeros(size(h_pre)); zeros(size(h_pre)), h_pre];

P_ext = transMat*P*transMat' + Q;

K = P_ext*H'*pinv(H*P_ext*H' + R);

x_ext = transMat * xo;

xo = x_ext + K*dz;

P = (eye(8) - K*H)*P_ext;
```

4.3.3 GNSS/INS Kalman filter

Lastly, the GNSS/INS integration system is implemented as shown in section 3.2. The implementation of this integration requires new navigation information, specifically IMU data. This data is also provided by the GNSSLogger app.

Calibration of IMU data

In order to correctly use the IMU data, calibration is needed, given that biases are always present in IMU measurements. In this specific application, before taking the actual measurements, it is required that the user carries out a measurement of the device facing the six different directions of the accelerometer, having each of the directions be pointed down for one minute. This data is then stored next to the actual navigation file with the name "calibration.txt". This information is read by the function "ReadGnssIMU", and the output is then given to the "calibrationIMU" function, which returns the values of the bias and scale factors, these values are computed as shown in section 2.3. These values are then given to the "calibrateIMUdata" function, which changes the IMU data delivered by the navigation file.

It is important to note that the calibration for the gyroscope is not done, given that GNSSLogger directly provides the user with attitude data.

Navigation algorithm

Having the calibrated IMU data at our disposal, this information, along with the GNSS data is provided to the "GpsKalmanIMUPvt", which returns the value of the PVT solution.

As in the previous cases, the state vector is initialized as a vector full of zeros. As in the Kalman filter, this vector is first given to the WLS algorithm, which computes the first solution for the PVT solution. Following GNSS solutions are computed using the "KalmanPvt" function, as done for the Kalman filter algorithm. For integration the "KalmanIMUPvt" function is used, this function receives first of all an updated state vector(xIns) which is computed using the navigation equations presented in section 2.4, in conjunction with the previous state vector(xo), the error covariance matrix for the integration(Pimu) and the time step size.

The integration is done as shown in listing 4.4.

Listing 4.4: INS Kalman filter code

```
1 dz = [xIns(1:3) - xo(1:3);xIns(4:6) - xo(5:7)];
2
3 H = [eye(6) zeros(6)];
4
5 F_aux = [
```

```
6       zeros(3),eye(3),zeros(3),zeros(3);
7       zeros(3),zeros(3),zeros(3),generateMat(xIns(7),xIns(8),xIns(9));
8       zeros(3),zeros(3),eye(3)*beta_or,zeros(3);
9       zeros(3),zeros(3),zeros(3),eye(3)*beta_acc
10      ];
11
12  transMat = eye(12) + F_aux*delta_t;
13
14  P_ext = transMat*P*transMat' + Q;
15  K = P_ext*H'*pinv(H*P_ext*H' + R);
16
17  xHat = K*dz;
18
19  P = (eye(12) - K*H)*P_ext;
```

The function "generateMat" builds a matrix that translates the accelerations measured by the IMU into the ECEF reference system. This is done because the state vector is referenced to the ECEF reference system. Also, as in the case of the Kalman filter, the noise figures are not provided by the manufacturer, so the initial values for the noise figures are taken from other papers using IMU units present in smartphones [13]. After this, these values are tuned to better fit the specific smartphones.

The final PVT solution is computed using this integration progress. The outputs of the "KalmanIMUPvt" function are the xHat value, which is subtracted from the state vector xIns to obtain the PVT solution, and the error covariance matrix P, which is used in the following iterations of the algorithm. It is important to note that the IMU data has a much higher frequency than the GNSS data, which makes this solution the one with the most amount of data in its solution, being much more usable for real applications.

4.4 PVT improvements

In this section, some improvements to the base algorithms are proposed in order to obtain better results, using some of the methods presented in previous chapters. These are based on the pre-processing of the available raw data or the addition of new data in order to obtain cleaner signals and better geometry for the problem to be solved.

4.4.1 Multi-constellation PVT

For multi-constellation PVT, multiple satellite constellations are used to compute the PVT solution. The data in this specific application corresponds to GPS and Galileo constellations.

This kind of integration helps obtain better accuracy on the computed PVT solutions, given that on each of the iterations of the PVT algorithm, the algorithm will get more pseudoranges, which may lead to a better geometry and, in turn, higher accuracy on the PVT solution.

In order to attain this, both the GPS and Galileo RINEX files at the corresponding time must be in the same folder as the GNSSLogger file. From these files, the satellites' ephemeris information is obtained, in order to integrate both files into the computation of the satellite solution, the user must input the ephemeris information as a cell with both of the constellations' information.

Having this information, the PVT solution can be computed using more satellites, which normally provides a better dilution of precision(DOP) which in turn, gives the solution a higher accuracy. Apart from this, in some cases, the solution to the PVT solution is not available from only one constellation, given that it may happen that there are less than 4 satellites available at that given time. In these occasions, having multiple constellations available can provide the PVT solution with a more continuous PVT solution, not skipping as many steps as in the case in which only one constellation is being used.

4.4.2 Ionospheric correction

As presented in section 1.3.3, for satellites that transmit at multiple frequencies, it is possible to apply ionospheric corrections, which eliminates a bias generated by the propagation of the signal through the ionosphere. GNSSLogger provides reliable dual-frequency GNSS signals for L1(1575.42 MHz) and L5(1176 MHz) frequencies.

As previously stated, and as we will see in the results, this ionospheric correction by itself is not always ideal, given that, even if it eliminates the bias, it also increases the standard deviation of the noise of the signal, making the resulting PVT solution noisier. For this reason, further processing of the ionospheric correction might be necessary.

To obtain this correction, first of all, the navigation data is used as input to the "dualFreqProcess" function. For each time step provided by the GNSS data, this function checks if there is available GNSS data for both of the used frequencies. In case both frequencies have available data at the time step, the correction shown in section 1.3.3 is applied to the data. On the other hand, if only one of the frequencies has GNSS data at that specific time, the function saves the data with the value given by that present frequency. In order to show the change between the corrected and uncorrected data, the different sections are plotted with different colors in the PVT solution's results.

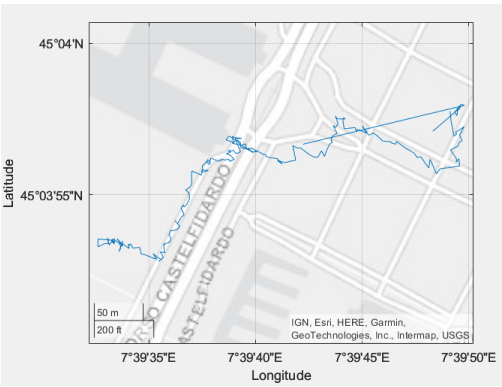
Chapter 5

Results

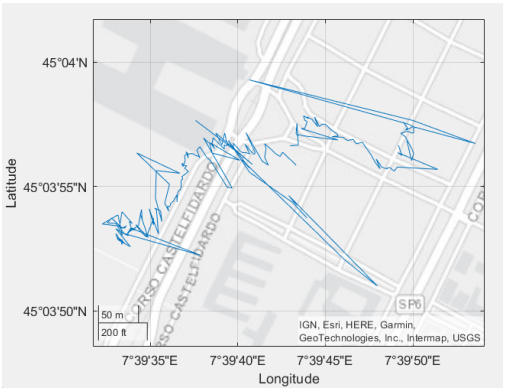
In this chapter, the results and progress in the development of the navigation algorithm developed in Matlab are shown. The WLS solution will be used as a base case to show the progress made when adding some improvements and then compare it to the navigation algorithms.

5.1 WLS

As stated previously, we will use the Weighted Least Squares(WLS) solutions in order to build on top of this solution and have a point of comparison with the other navigation algorithms. First of all, we check the output using WLS on a test done while walking on the streets of Turin. The results of using WLS only with GPS satellites and then only with Galileo satellites are shown in figures 5.1a and 5.1b.



(a) WLS with GPS constellation

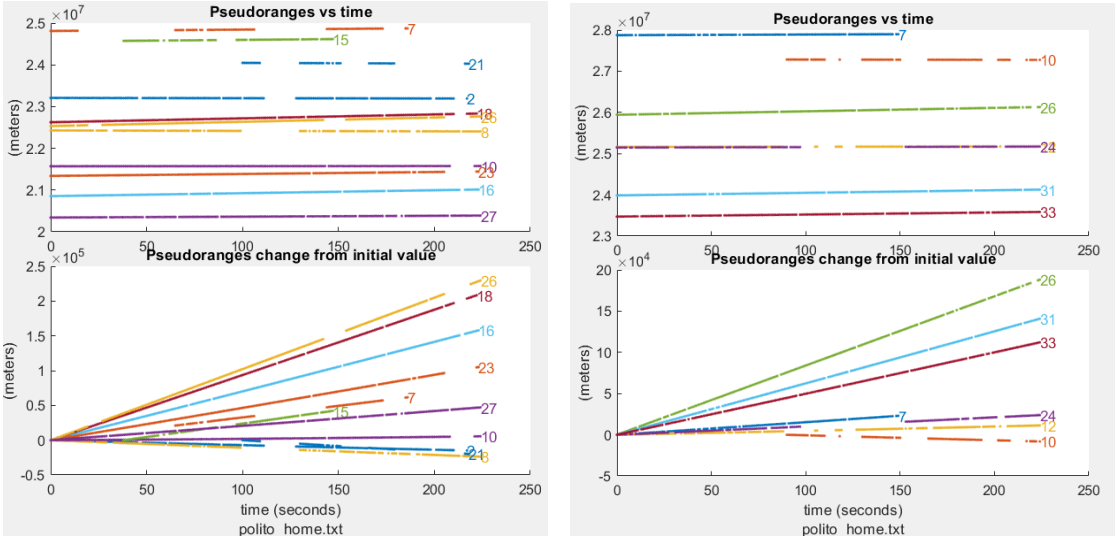


(b) WLS with Galileo constellation

Figure 5.1: WLS output

As we can see, both constellations generate noisy results, having some quick changes in position that wouldn't make sense if the physical model was taken into consideration. The longer "jumps" can be explained by the noisy measurements of some of the satellites at the time instant, in conjunction with the lack of enough satellites to compensate for the measurement.

It is important to note that when comparing both results, it is obvious that Galileo outputs a less precise result, this can be explained by looking at 5.2a and 5.2b, which show the available pseudoranges for both constellations. In the case of the Galileo constellation, fewer satellites are available at any given moment which means that the geometry of the solution is less ideal, resulting in a higher GDOP and a less precise PVT solution.



(a) GPS **(b)** Galileo

Figure 5.2: Pseudorange and pseudorange change for GPS and Galileo

5.1.1 Multi-constellation PVT

In order to obtain an even better geometry for the solution to this problem, it is possible to use satellites of multiple constellations. The result of this integration of both constellations is shown in 5.3. As expected, the PVT solution shows a more precise solution, not having any sudden change in position in contrast to the two previous solutions. Also, the variance in the position seems to have decreased, making for a cleaner trajectory overall.

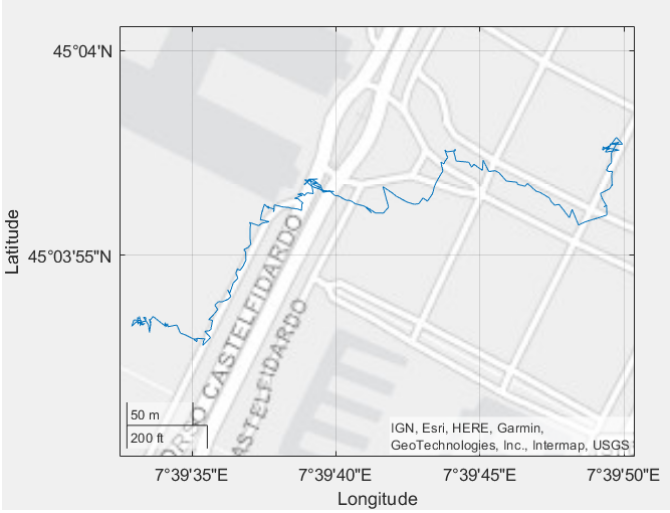


Figure 5.3: WLS result with multiple constellations

Following the same argument as before, the use of both constellations means a higher number of satellites being used for computation, which in turn means a better geometry (DOP) and a better PVT solution.

To further show the capabilities of the algorithm, another example is computed. In this case, the acquired information corresponds to a walk around a block of Turin with

For the following results, we will consider both constellations for the computation of the PVT solution, given that it always returns a better output than the use of only one of the constellations.

5.1.2 Ionospheric Corrections

As explained in section 1.3.3, as the signal gets propagated through the atmosphere, the ionosphere generates a bias on the pseudorange measurement, which can be corrected. It is important to note that for this application, the sampling frequency of the pseudoranges does not allow the removal of the bias to be seen when plotting the values of the obtained pseudoranges, this is because the change in the value of the pseudorange on each sampling moment is too great in comparison to the bias correction itself. For this reason, the comparison between corrected and non-corrected results will be shown in the PVT solution.

An example of a PVT solution using this correction is shown in figures 5.4, 5.5a, and 5.5b. These results show that most of the time, the PVT solution with iono-corrected pseudoranges provides a very similar output as just using the plain pseudoranges, having a difference of a few meters in some areas. The problem appears in the last part of the trajectory, after the 200-second mark, where the

solutions stop having the same shape, especially looking at the x-axis.

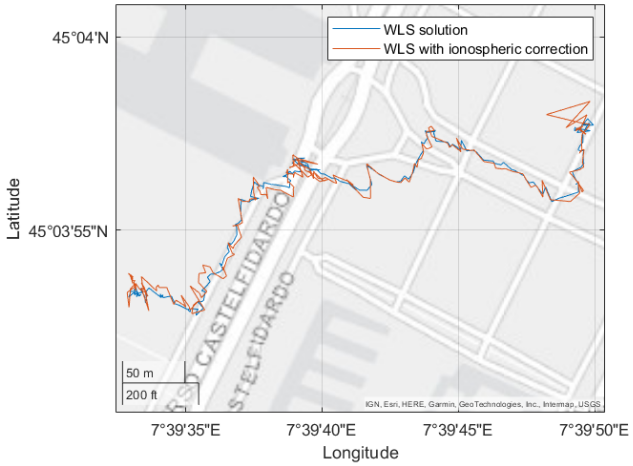
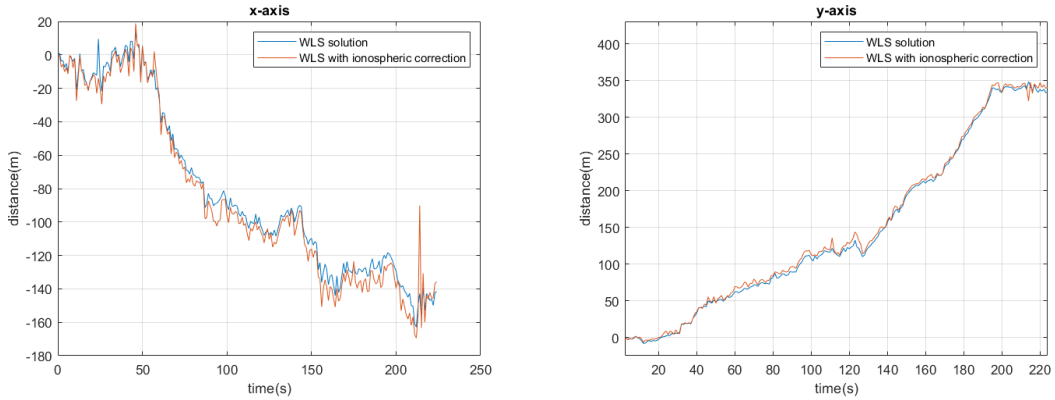


Figure 5.4: WLS solution with ionospheric corrections



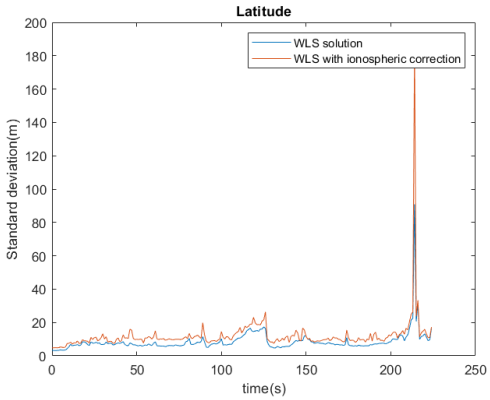
(a) WLS with GPS constellation **(b)** WLS with Galileo constellation

Figure 5.5: Axes of the WLS solution with ionospheric corrections

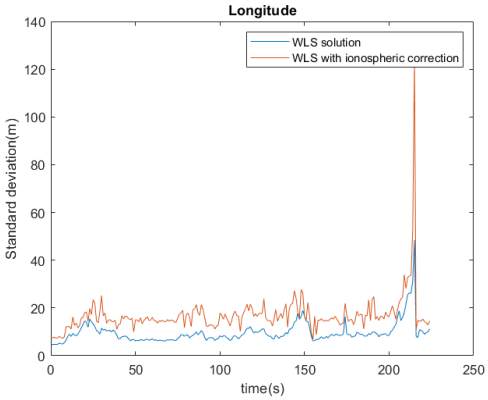
It is important to note that the signals for both frequencies are not always available, this means that in the PVT solution when there are no dual-frequency pseudoranges available, the system uses whatever pseudorange is available at any given moment.

Also for these solutions, the standard deviation of the computed solution is proportioned. The values of the standard deviation for the altitude, longitude, and altitude are shown in figures 5.6a, 5.6b, and 5.6c, respectively. The standard

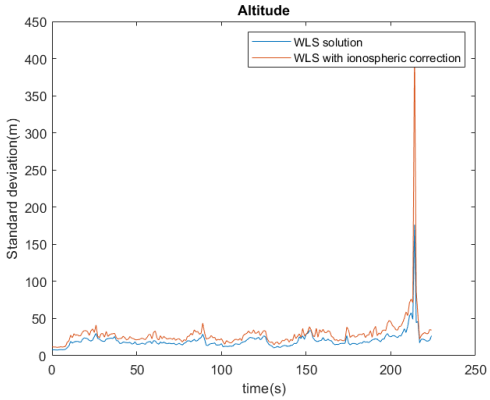
deviation of the iono-corrected solution is always at least as high as the one with no correction. For most of the path, this difference doesn't affect the end result but, as was noted previously, at the end of the trajectory the difference becomes too great given the increase in the standard deviation of the result. While for the plain WLS PVT solution the output follows a more or less plausible trajectory, the x-axis of the iono-corrected solution deviates too much from the followed path until that moment.



(a) Latitude standard deviation



(b) Longitude standard deviation



(c) Altitude standard deviation

Figure 5.6: Standard deviation of the PVT solution

An example of another trajectory is shown in figures 5.7, 5.8a, and 5.8b while the value of the standard deviations is shown in figures 5.9a, 5.9b, and 5.9c. In this case, even if the value of the standard deviation never gets to extreme values, the accuracy of the PVT solution is affected by this effect. This effect can be seen in figure 5.8a, in the measurements between the 60 and 100 seconds marks, the iono-corrected solution's values vary in a greater amount. This is reflected in figure

5.6b, in which the std has a higher value at those specific times.

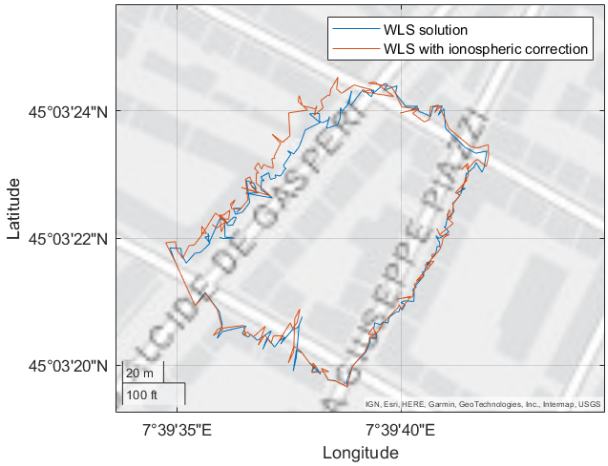
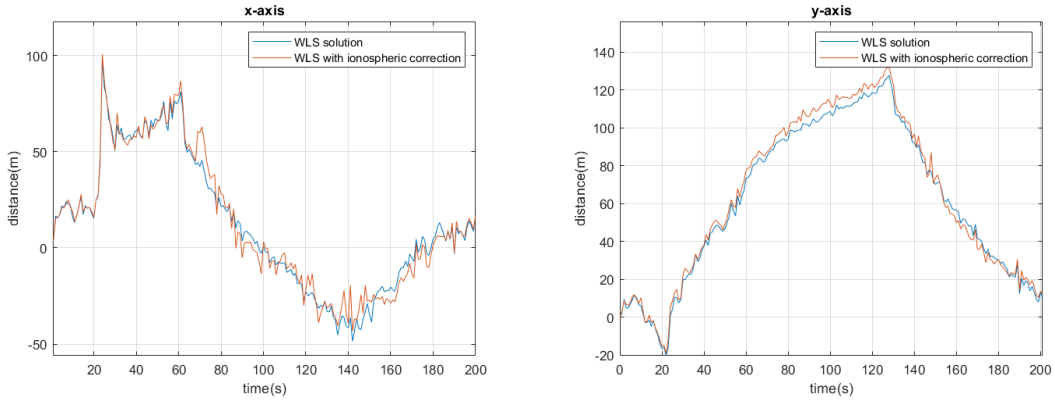
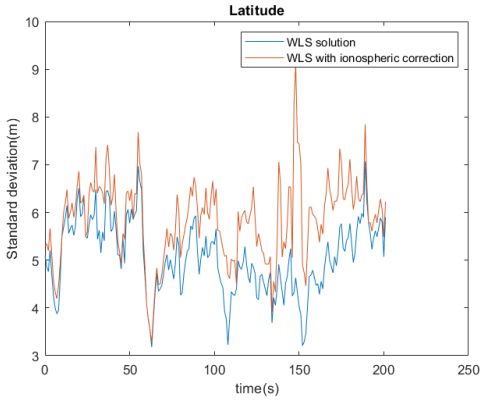


Figure 5.7: WLS solution with ionospheric corrections

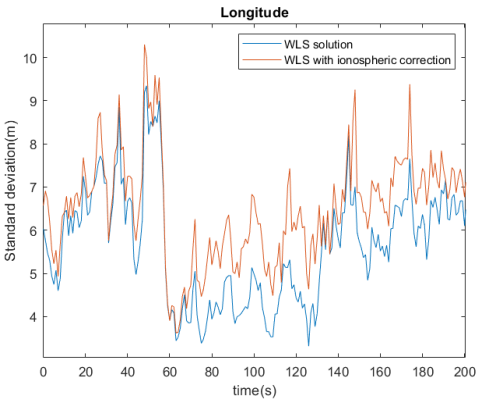


(a) WLS with GPS constellation **(b)** WLS with Galileo constellation

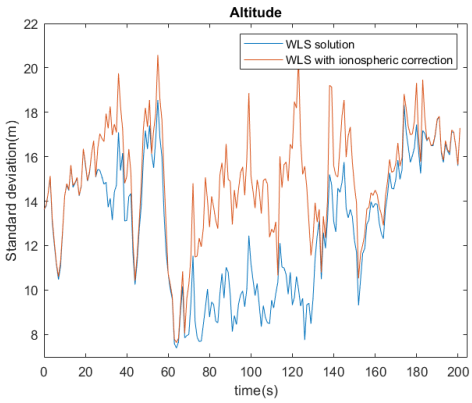
Figure 5.8: Axes of the WLS solution with ionospheric corrections



(a) Latitude standard deviation



(b) Longitude standard deviation



(c) Altitude standard deviation

Figure 5.9: Standard deviation of the PVT solution

For this specific solution, the standard deviation is comparable to the ionospheric correction at all times, which means that the final result is similar for both the corrected and not corrected data, having a low difference because of the removal of the bias.

It must be taken into account that even if this correction can generate an increase in the standard deviation of the PVT solution, this doesn't mean that the correction should not be applied in other applications. For example, in a situation in which the system can guarantee pseudorange measurements with lower amounts of noise, this correction means a constant correction of the ionospheric bias with no repercussions on the solution.

Given that most mobile devices cannot guarantee this scenario, as previously mentioned, a technique can be used that takes advantage of the fact that the changes in the ionosphere are slow. This process is further explained in [8] and it could be applied in future works.

In the following results, this correction will not be applied to the pseudoranges. This is because the possible change in the variance of the results can be too great in comparison with the proportioned bias correction.

5.2 Kalman filter

As it was stated before the system's noise information is not provided by the manufacturers of the phones. This means that some of the noise figures must be approximated, or "calibrated" as it will be referred to from this point on. In this specific case, the most important calibration value is the system covariance matrix Q.

In order to obtain the values for the Q matrix, a set of initial values obtained from [8] was used, as it is stated that it is a typical value for commercial-level receivers, such as the one used on a mobile device. The value of the Q matrix is as follows

$$Q = \begin{bmatrix} 0.01 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.01 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.1e-3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1e-5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1e-5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1e-5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1e-4 \end{bmatrix} \tag{5.1}$$

With this value of the Q matrix, the computed PVT solution is as shown in 5.10. As done previously, this solution is shown in conjunction with the WLS solution. Also in figures 5.11a and 5.11b the change on the x and y axis of the ECEF coordinates in meters are shown.

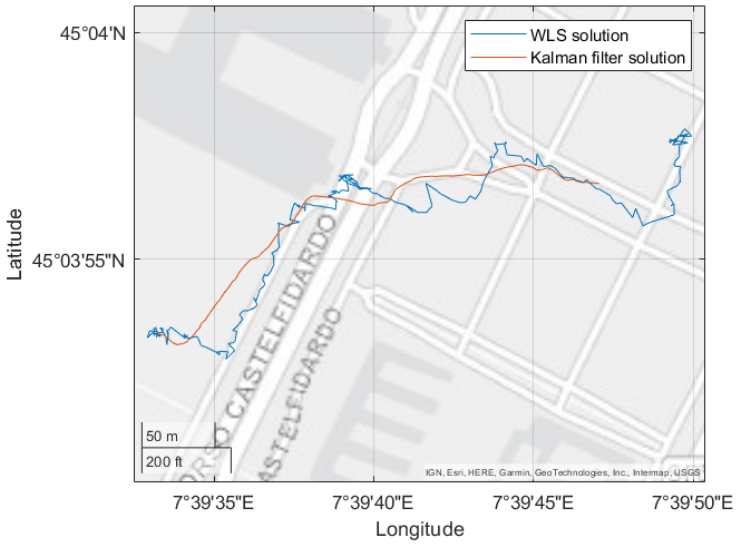
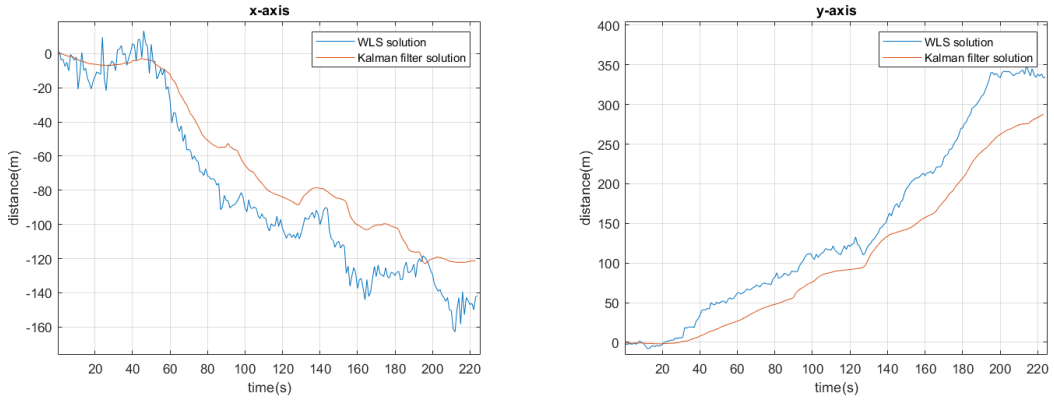


Figure 5.10: Uncalibrated Kalman filter PVT solution



(a) x-axis **(b)** y-axis

Figure 5.11: Uncalibrated Kalman filter solution

Having this first value for the Q matrix, the solution is a lot smoother than the one obtained using plain WLS, this happens because the model takes into consideration the physical model that describes the movement of the user. The problem that appears with this solution is that it may be excessively reliant on the physical model of the problem, giving the curve a shape overly smooth to describe real movement. Also for this specific case, the solution obtained with the Kalman filter does not get close to the final estimation done by the WLS solution and, as

shown in figures 5.11a and 5.11b probably because the update on the solution takes too much into account the estimated speed of the user, which is this case is low given that the user is walking.

As it can be seen in equation 3.14 the larger the value of the Kalman gain, the less the updated solution relies on the physical model. According to equation 3.13, the value of the Kalman gain(K_k) depends on the value of P_k^-, which in turn gets its value from the Q matrix according to equation 3.11. This means that the higher the value of the Q matrix, the closer the value of K_k gets to the identity, and on the contrary, as the Q matrix gets smaller, the value of the solution is closer to the WLS solution. To further prove that this value for the Q matrix does not fit the problem, we test with a Q matrix with a lower value for the Q matrix.

$$Q = \begin{bmatrix} 1e-4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1e-4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1e-4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.1e-5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1e-7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1e-7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1e-7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1e-6 \end{bmatrix} \tag{5.2}$$

The obtained solution for this value of the matrix is shown on 5.12. As expected, the obtained PVT solution is even more different than the previous solution to a possible trajectory followed by a real user. In figures 5.13a and 5.13b it can be seen that the PVT solution follows the WLS solution shape, but it has even softer transitions in position.

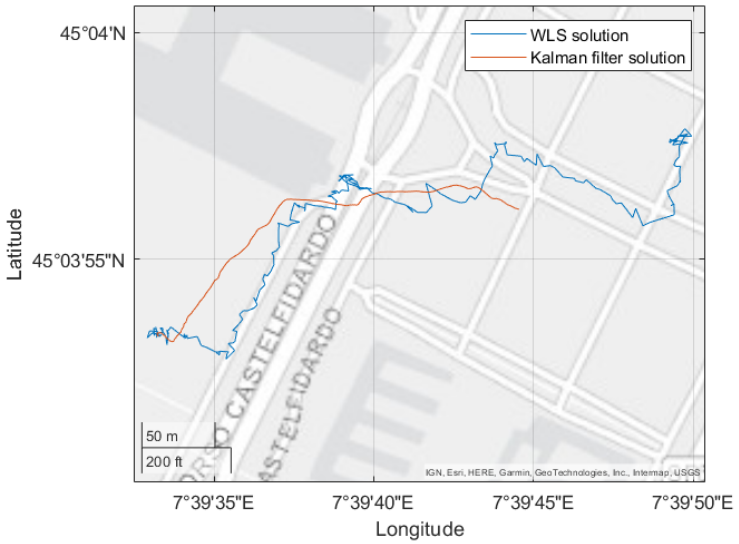
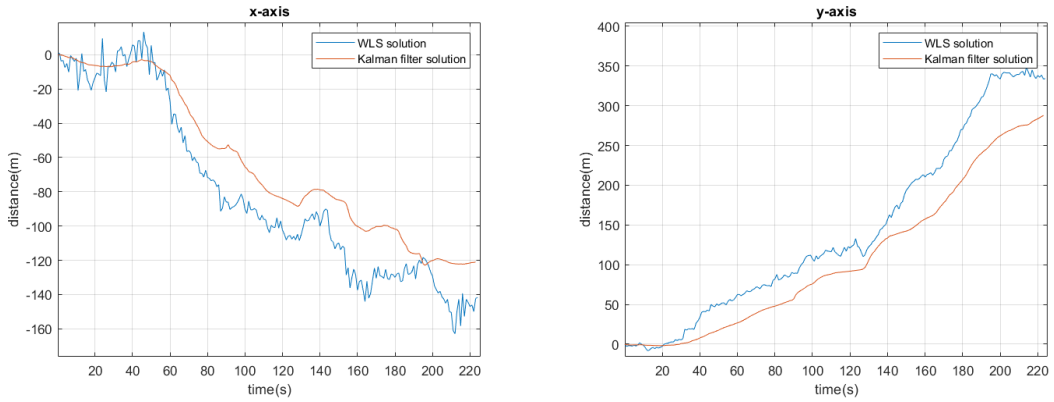


Figure 5.12: Uncalibrated Kalman filter PVT solution



(a) x-axis **(b)** y-axis

Figure 5.13: Uncalibrated Kalman filter solution

Taking this into account, the value of Q is lowered to:

$$Q = \begin{bmatrix} 0.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.1e-3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1e-4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1e-4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1e-4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.04 \end{bmatrix} \tag{5.3}$$

The PVT solution using this new Q matrix is shown on 5.14.

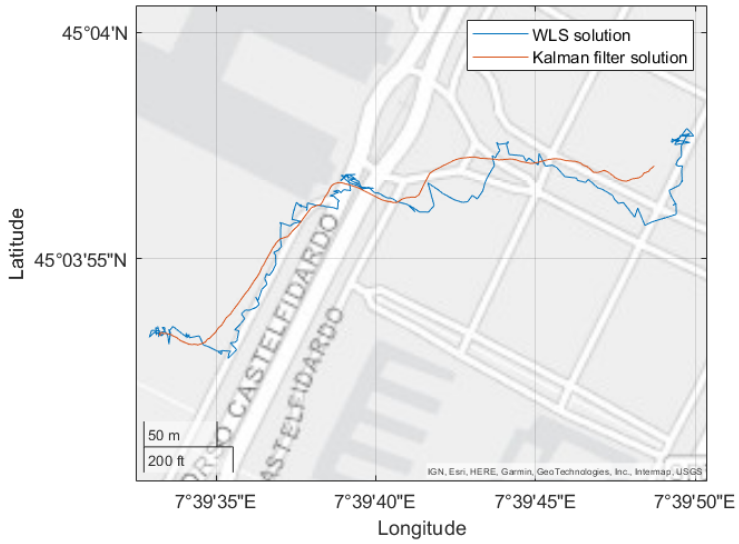


Figure 5.14: First attempt on calibrated Kalman filter PVT solution

With this Q matrix, the trajectory followed by the user is more realistic, but still far from an actual trajectory followed on the street. After some iterations, a value that accurately fits the system is obtained.

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.11 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1e-3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1e-3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1e-3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.01 \end{bmatrix} \tag{5.4}$$

The solution after applying a correctly tuned Kalman filter is shown in figures 5.15, 5.16a and 5.16b. It is clear that in comparison to the previously generated PVT solutions, this presents a cleaner and continuous navigation solution. This is explained by the fact that this result takes into consideration the physical model that describes the movement of the user.

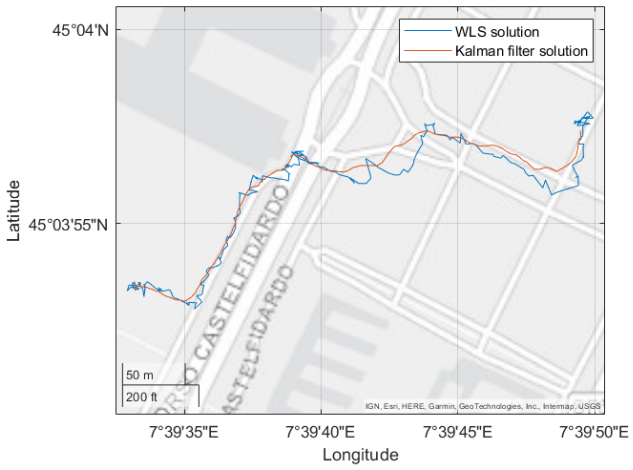
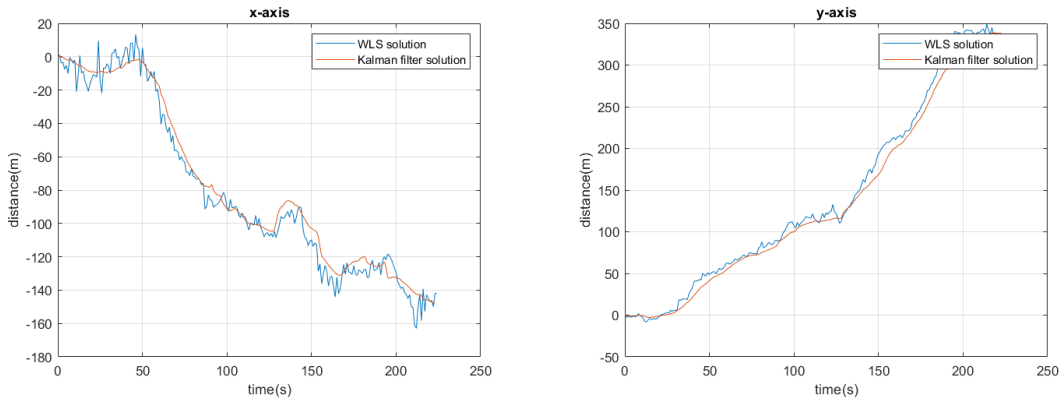


Figure 5.15: Calibrated Kalman filter PVT solution



(a) x-axis **(b)** y-axis

Figure 5.16: Calibrated Kalman filter solution

To further show the effectiveness of this navigation technique an example under other circumstances is shown in figure 5.17, the Q matrix is kept with the same value. In this example, the user of the mobile device moves a longer distance by bike, meaning that the speed of the movement is higher, which in turn means that this movement is heavily affected by inertia.

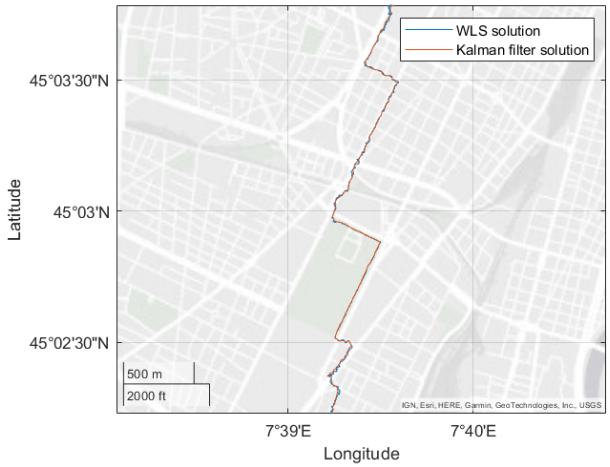


Figure 5.17: Kalman filter PVT solution

To better understand the result a zoomed version of the result is shown in figure 5.18. As in the previous solution, the solution correctly shows the possible movement of the user.

But a problem arises, given the fact that the user is moving at a higher speed,

this means that the user should not be able to make such sudden turns. With this in mind, the value of the Q matrix is changed to better fit this scenario.

$$Q = \begin{bmatrix} 0.2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2.2e-2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2e-4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2e-4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2e-4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2e-3 \end{bmatrix} \tag{5.5}$$

Figure 5.18 shows the result using this new matrix. This trajectory better suits the trajectory of a bike given the inertia of the movement. This means that depending on the speed of the user, different Kalman filters can be applied to the received navigation data. For even greater speeds this phenomenon is even more noticeable.

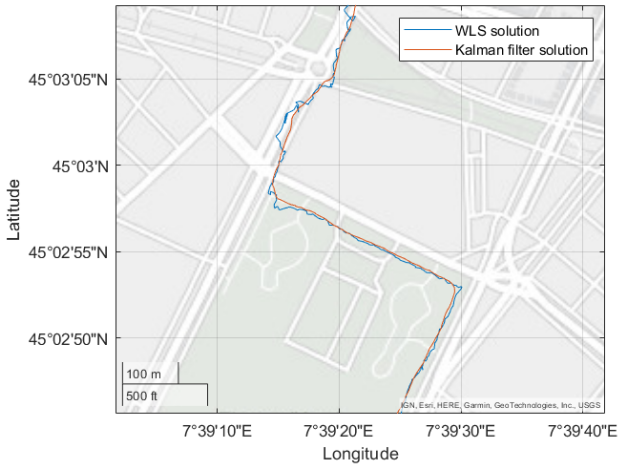


Figure 5.18: Zoomed Kalman filter PVT solution

Finally, an example using an older device is computed. For this example, the trajectory followed by the user is a walk around Politecnico di Torino, starting on the inside of a building, to then walk inside a building, which returns a less precise solution. The Q matrix in this example is the same as the one used in the previous example that was done at walking speed. The result of these simulations is shown in figures 5.19, 5.20a and 5.20b.

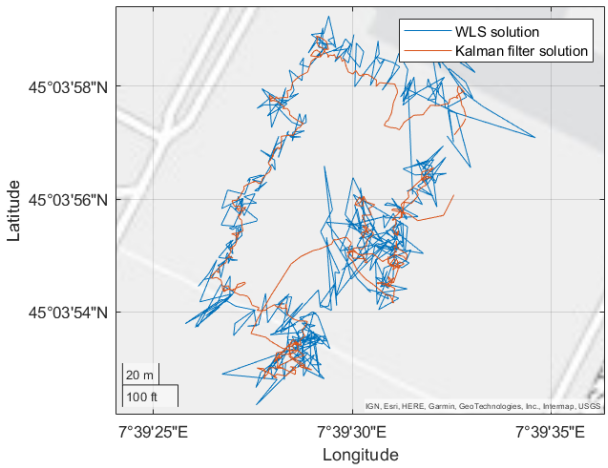
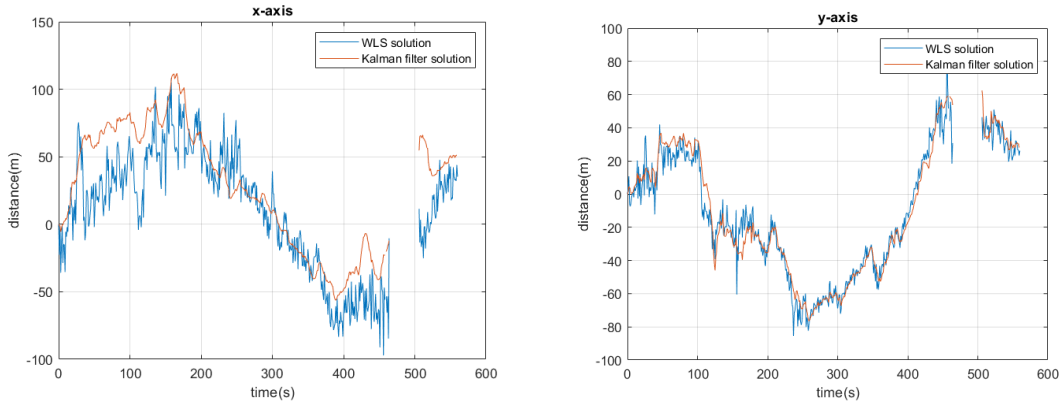


Figure 5.19: Kalman filter PVT solution for a noisier device



(a) x-axis **(b)** y-axis

Figure 5.20: Kalman filter solution axes

It can be seen that even for an older device, with less precise receivers and fewer constellations available, it is still possible to obtain a solution that for the outdoors is acceptable for a phone GNSS application. On the other hand, the available signals indoors appear to be greatly affected by the effects of low SNR and multipath, resulting in a noisier PVT solution which is not reliable.

In the case of a newer device, with better components for reception of the GNSS data, the obtained results are shown on 5.21, 5.22a and 5.22b.

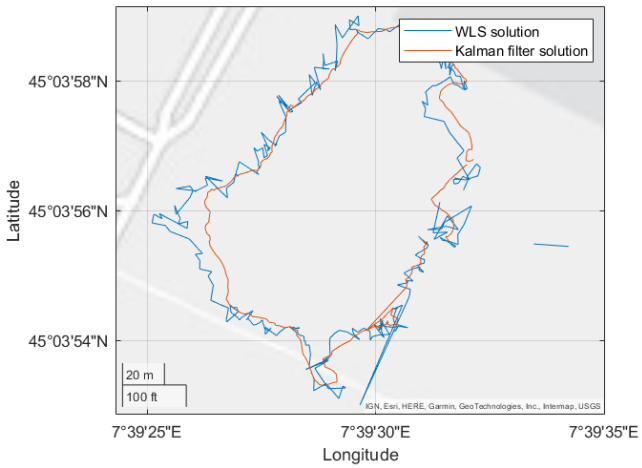
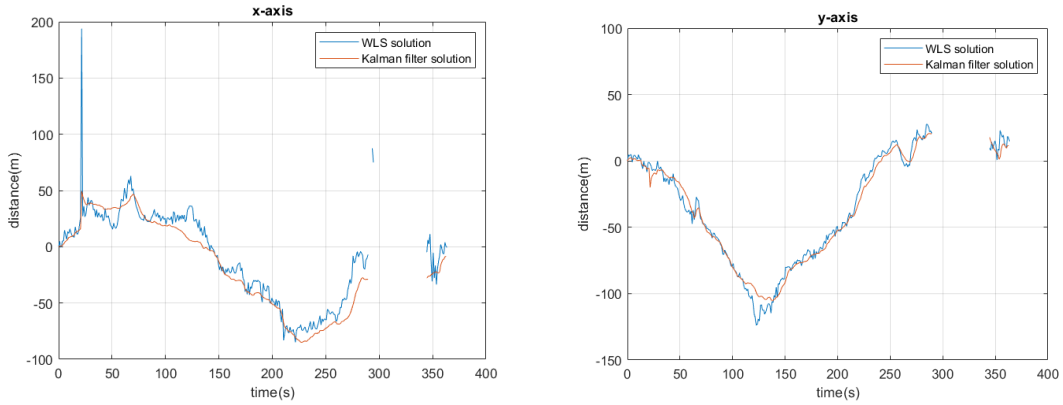


Figure 5.21: Kalman filter PVT solution



(a) x-axis **(b)** y-axis

Figure 5.22: Kalman filter solution axes

It can be seen that for the same trajectory, the Kalman Filter on a device with less noise also provides a great improvement in the accuracy and precision of the navigation solution. It is important to note that in both of these cases, in some parts of the trajectory, there were not enough satellites on site and the algorithm was still able to recover the next values of the solution when the satellites were available again.

5.3 GNSS/INS integration

5.3.1 Calibration

As previously explained, before being able to use the IMU data for the INS/GNSS integration, this must be properly calibrated. For this application, calibration is carried out as it was previously explained in section 2.3. With this in mind, before each GNSS/INS integration measurement, a calibration file is created. This calibration file contains measurements like the ones shown in figure 5.23 which corresponds to the calibration file of the first example shown in 5.1. This figure represents the measured accelerations on each one of the axes when facing the six different directions. As expected the value of the measured acceleration on each of the axes is close to the gravitational acceleration.

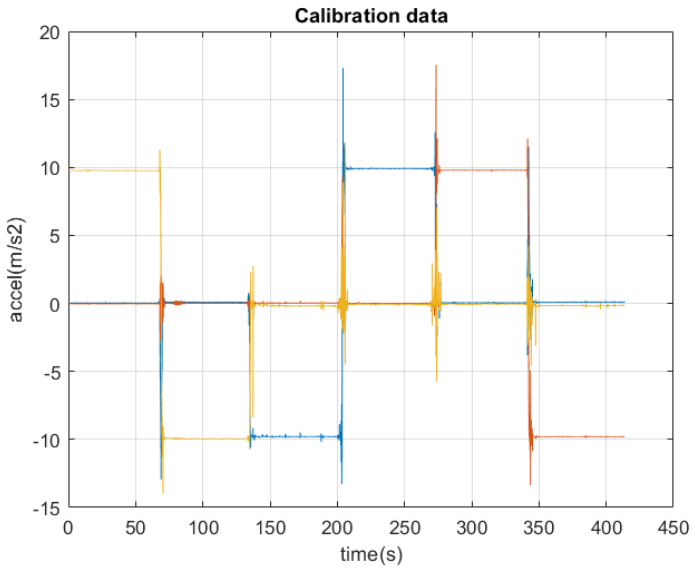


Figure 5.23: Data used for calibration

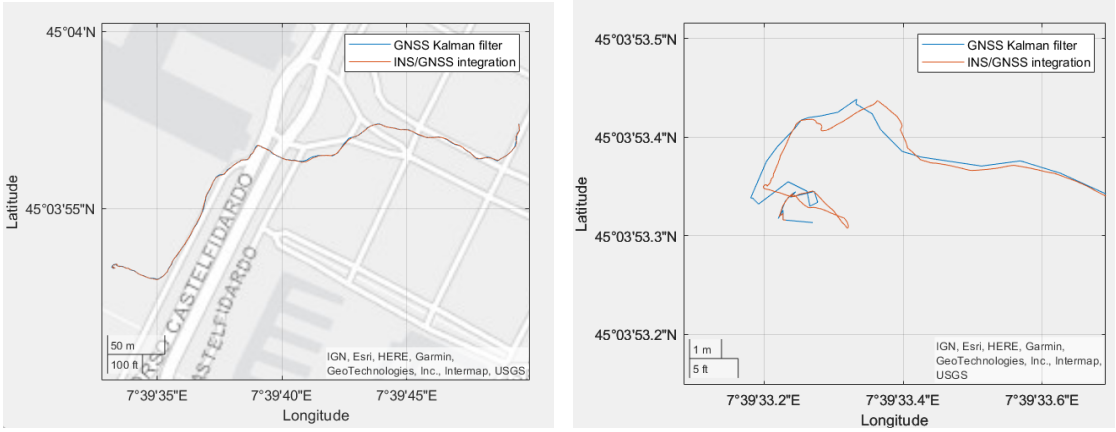
This specific measurement was acquired before the example shown in section 5.1.

It must be taken into account that, given the available resources for the experiments, the calibration is not very precise and the IMU data recovered is in all cases very noisy, thus the Kalman filter used for this cases is heavily reliant on the GNSS Kalman filter PVT solution.

5.3.2 PVT solution

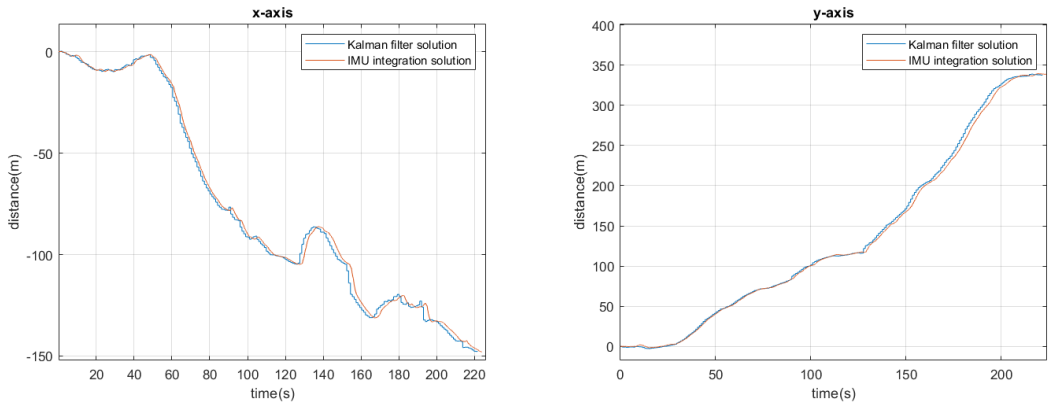
As described in previous chapters, the selected integration is loosely coupled integration. This means that the Kalman filter used in the previous section is maintained, but a new one is developed for the integration. With this new filter, a new process of calibration of the Q matrix is required. After this calibration, the obtained Q matrix for the integration Kalman filter is diag([1e-8, 1e-8, 1e-8, 1e-9, 1e-9, 1e-9, 1e-9, 1e-9, 1e-8, 1e-8, 1e-8]), which is a diagonal matrix with the values of this vector on the diagonal.

Figures 5.24a, 5.24b, 5.25a and 5.25b show the PVT solution of the integrated system, in conjunction with the simple Kalman filter result. Both cases present a similar solution, with the difference that the INS/GNSS integration case has a greater amount of estimated points, resulting in a smoother curve.



(a) INS/GNSS PVT solution **(b)** Zoomed map

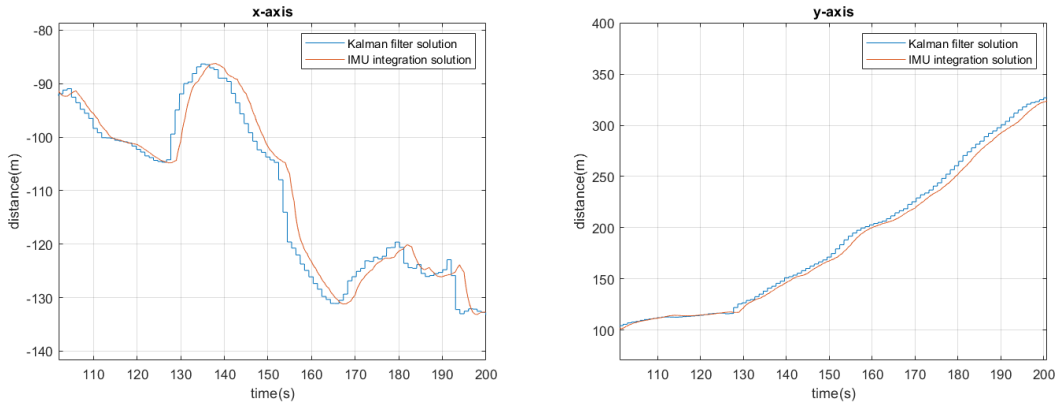
Figure 5.24: INS/GNSS PVT solution



(a) x-axis **(b)** y-axis

Figure 5.25: INS/GNSS PVT solution axes

Figures 5.26a and 5.26b show a zoom of both navigation solutions sampled at the same frequency, this makes apparent that the Kalman filter solution provides a lower frequency solution when being compared with the integrated solution. This makes the INS/GNSS integrated implementation better for real-time applications.



(a) zoomed x-axis **(b)** zoomed y-axis

Figure 5.26: Zoomed INS/GNSS PVT solution axes

As stated in previous chapters, the presence of both GNSS and IMU measurements allows the system to have a continuous PVT solution, thanks to the fact that the INS solution is constantly being computed. On the other hand, the PVT solution provided by the GNSS system is not constant given that it depends on the availability of the GNSS signals. The figures 5.27, 5.28a, and 5.28b show an

example of a PVT solution when the GNSS signals are not always available. In this example, around the 280-second mark the signal is lost, this means that for a short period, the solution is computed only by the INS system. When comparing this to the Kalman filter solution, it is apparent that the integrated solution can maintain a continuous solution while the Kalman filter one has a discontinuous solution.

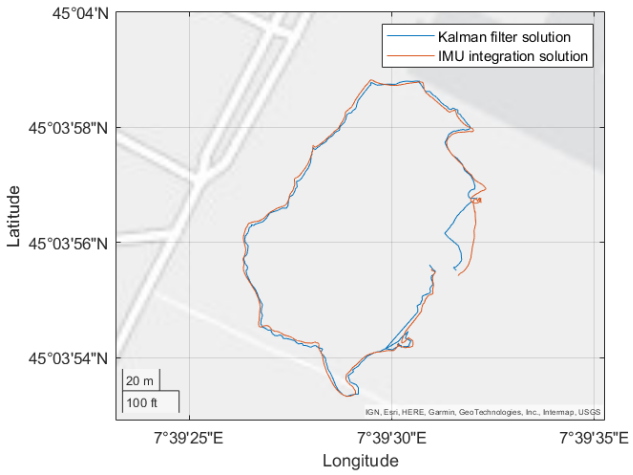
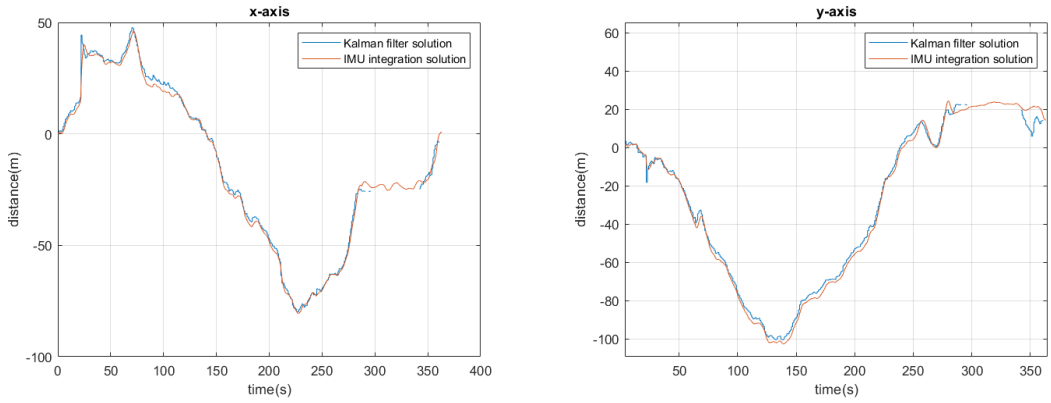


Figure 5.27: INS/GNSS PVT solution without a continuos GNSS input



(a) x-axis **(b)** y-axis

Figure 5.28: Axes of INS/GNSS integration solution

It is also important to note that even if the INS provides a solution at a higher frequency, the precision rapidly decreases with time. In contrast, the results provided by the GNSS solution have a lower frequency but are a lot more precise.

In order to show this, the frequency of the GNSS measurements is changed in the simulations. For the first example, the frequency is reduced to 0.2Hz, from the previous sampling frequency of 1Hz. The results of this process are shown in figures 5.29, 5.30a, and 5.30b. It can be seen that even if the solutions are more or less similar in shape, the use of the less accurate INS measurements makes the solution deviate from the trajectory when comparing it to its more precise counterpart.

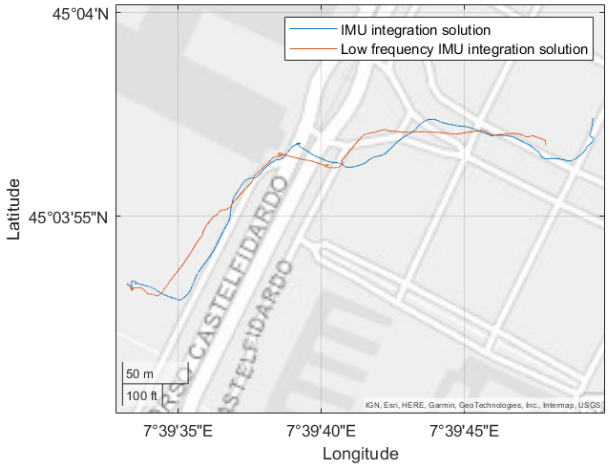
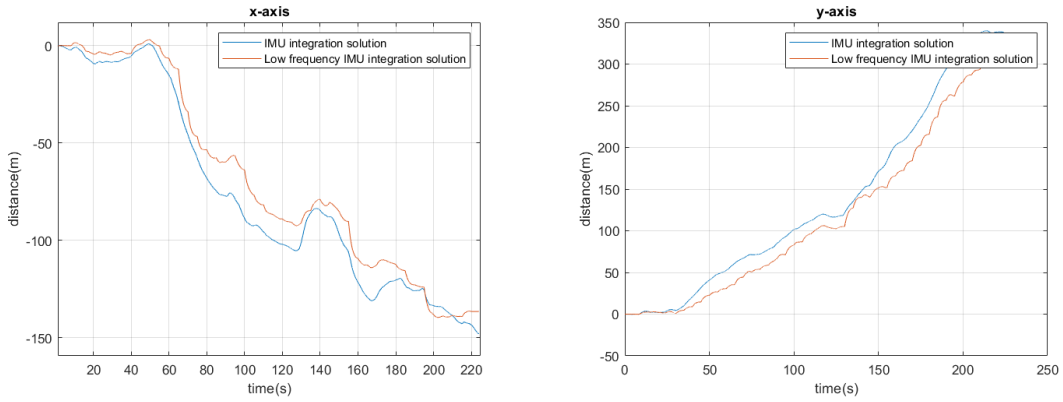


Figure 5.29: INS/GNSS PVT solution with $f_{GNSS} = 0.2$



(a) x-axis **(b)** y-axis

Figure 5.30: Axes of INS/GNSS integration solution

Finally, as a last example, the frequency is even further reduced to 0.1Hz. The solution is shown in figures 5.31, 5.32a, and 5.32b. This case, being more extreme

than the previous one, has the values of the solution deviate even further from the more precise solutions.

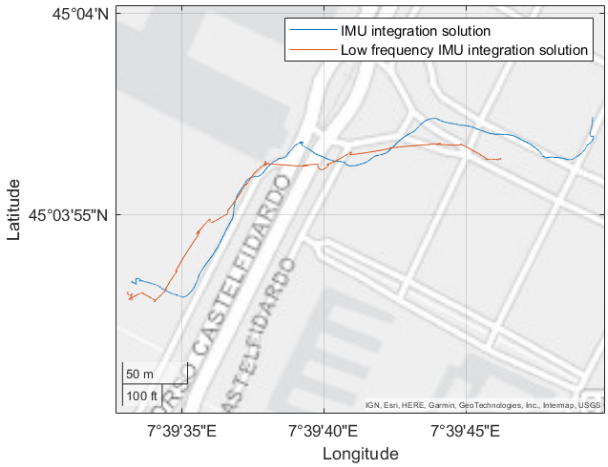
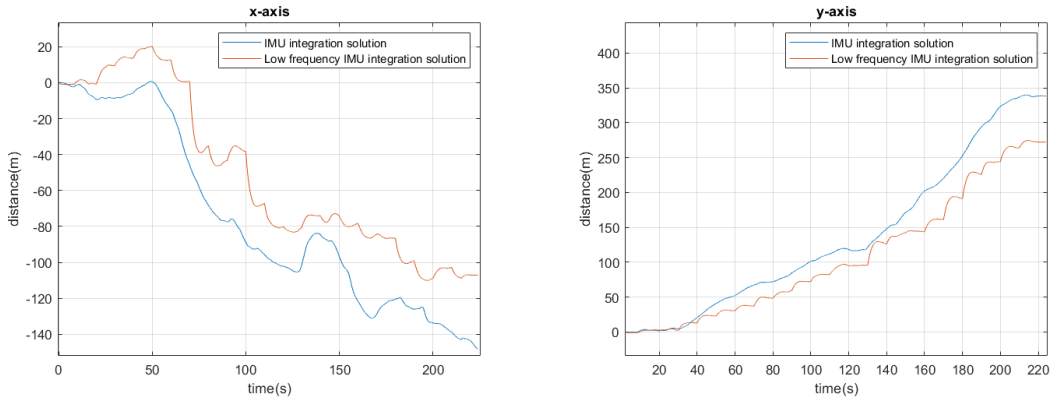


Figure 5.31: INS/GNSS PVT solution with $f_{GNSS} = 0.1$



(a) x-axis **(b)** y-axi

Figure 5.32: Axes of INS/GNSS integration solution

Chapter 6

Conclusions and possible improvements

In this work, a review of different navigation techniques has been provided. Between these techniques, the application of a Kalman filter and INS/GNSS integration rise as the most important ones in this study.

All the tests were performed using data coming exclusively from smartphones, using the GNSSLogger app, while the processing of the data and computation of the PVT solution was done using Matlab. As expected for a mobile device, the obtained pseudoranges are heavily affected by noise given the low-end receivers that are used in this type of device. This is where the navigation techniques present themselves, being solutions to cope with this noisy data and obtain a usable solution for mobile applications.

Going in order, the first algorithm used is WLS, which provides the user with a PVT solution that is completely affected by the noise of the navigation data. Even if the solution might have a high accuracy, its precision is very low, returning a solution that is not usable for most applications. In order to solve this problem, a Kalman filter can be applied to smooth the results. This is done by taking into account the physical model that describes the movement of the user. In order to obtain an acceptable solution for the Kalman filter, the correct values for the used matrices have to be estimated. The change in these values describes the proportion in which the measured data is used in comparison with the data provided by the physical model. For this study these correct values were estimated, obtaining usable solutions for mobile devices' applications.

Having the Kalman filter PVT solution, the other major enhancement to the PVT solution was done by adding loosely coupled GNSS/INS integration to the solution. In order to do this, IMU data was also provided by the GNSSLogger app. This data had to be correctly calibrated before being used, but the result is

a PVT solution with higher accuracy and a much larger frequency, which makes this option optimal for navigation applications. The addition of satellites to the computation provides also a significant improvement to the PVT solution. This is because a higher number of satellites for the computation means a better geometry for the problem and, in turn, a solution with a lower standard deviation.

The final implemented improvement is the ionospheric correction applied to the received pseudoranges. This is only possible for devices that can acquire signals in both L1 and L5 frequency bands. These frequency bands provide the user with two different pseudorange values that can be used in order to estimate the bias generated by the ionosphere. As shown in this study, the ionospheric correction of the data increases the standard deviation of the received pseudoranges, which is why the technique was shown but not used in most of the cases.

Regarding possible future work and improvements on the used algorithms, various techniques can be applied in order to provide a better PVT solution for the specific case of a cellular phone application.

First of all, the use of more constellations for the computation of the solution is the obvious choice, given the fact that for this study only the GPS and Galileo constellations were used. Navigation data from BEIDOU and GLONASS could further improve the GNSS server, making it possible to obtain enough sufficient GNSS data in some areas or obtain a better GDOP given that these satellites can provide the problem with a better geometry.

Also, as was previously mentioned, the ionospheric corrections can be further improved by using a method shown in [8], in which past values of the ionospheric correction can be used in order to filter the newly computed values, solving the problem of the increase of the standard deviation of the received pseudorange values. Apart from this, filtering the measurements with higher SNR can be an option for some systems, affecting the continuity of the measurements but giving the result a more stable trajectory.

Taking a look at the possible improvements of the Kalman filter, an adaptive Kalman filter can be implemented, in which the values of the used matrices change depending on the speed of the user. This can make the PVT solution better fit every situation, obtaining a more precise solution that properly uses the physical model describing the movement and the provided navigation data.

It is also important to note that in this application, only the loosely coupled integration was implemented. For further improvement of the navigation solution tightly coupled or ultra-tight coupled navigation integration systems can be implemented. These implementations

Finally, thinking about strictly software-based techniques, the initial position used for estimation can be given by previous uses of the device or by other devices that communicate with it. Also, as shown in [14], map matching is commonly used for mobile applications, providing the user with useful navigation data given the

common situations in which the GNSS system is used by normal consumers.

Bibliography

[1] *Official U.S. government information about the Global Positioning System (GPS).* https://www.gps.gov/systems/gps/space/. Accessed: 2023-05-22 (cit. on p. 1).

[2] *ESA - Navipedia. GPS Space Segment.* https://gssc.esa.int/navipedia/index.php/GPS_Space_Segment. Accessed: 2023-06-16 (cit. on p. 1).

[3] *ESA - Navipedia. GLONASS Space Segment.* https://gssc.esa.int/navipedia/index.php/GLONASS_Space_Segment. Accessed: 2023-06-16 (cit. on p. 2).

[4] *ESA - Navipedia. Galileo Space Segment.* https://gssc.esa.int/navipedia/index.php/Galileo_Space_Segment. Accessed: 2023-06-16 (cit. on p. 2).

[5] *ESA - Navipedia. Beidou Space Segment.* https://gssc.esa.int/navipedia/index.php/BeiDou_Space_Segment. Accessed: 2023-06-16 (cit. on p. 2).

[6] *ESA - Navipedia. GNSS signal.* https://gssc.esa.int/navipedia/index.php/GNSS_signal. Accessed: 2023-06-16 (cit. on p. 3).

[7] Jacques Georgy Aboelmagd Noureldin Tashfeen B. Karamat. *Fundamentals of Inertial Navigation, Satellite-based Positioning and their Integration.* Springer, 2013 (cit. on pp. 7, 17–20, 22, 23, 32, 38, 40).

[8] P. D. Groves. *Principles of GNSS, Inertial, and Multi-Sensor Integrated Navigation Systems.* Artech House, 2007 (cit. on pp. 8, 9, 16, 28, 30, 33–37, 57, 58, 75).

[9] Christopher J. Hegarty Elliot D. Kaplan. *Understanding GPS: Principles and Applications (2nd ed).* Artech House Mobile Communications, 2006 (cit. on pp. 10, 11, 15, 31).

[10] *Dati Rinex per post-processing.* https://gnss.regione.fvg.it/dati-rinex/. Accessed: 2023-06-16 (cit. on pp. 41, 44).

[11] *Android developers guide on Raw GNSS Measurements.* https://developer.android.com/guide/topics/sensors/. Accessed: 2023-05-22 (cit. on pp. 41, 42).

[12] The GSA GNSS Raw Measurements Task Force. *Using GNSS raw measurements on Android devices*. `https://www.gsa.europa.eu/system/files/reports/gnss_raw_measurement_web_0.pdf`. Accessed: 2023-06-16 (cit. on p. 43).

[13] V. Gikas and H. Perakis. «Rigorous Performance Evaluation of Smartphone GNSS/IMU Sensors for ITS Applications». In: *MDPI, Sensors* 16 (Aug. 2016). DOI: `10.3390/s16081240` (cit. on p. 49).

[14] T. Wenxin and Y. Wang. «Real-Time Map Matching: A New Algorithm Integrating Spatio-Temporal Proximity and Improved Weighted Circle». In: *Open Geosiences* 11 (2019). DOI: `https://doi.org/10.1515/geo-2019-0023` (cit. on p. 75).