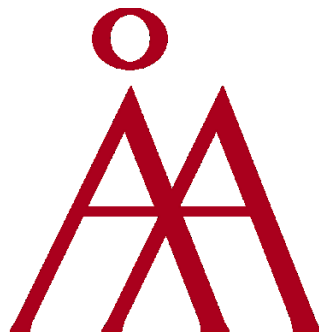


Master's Thesis:
GNSS multipath rejection based on environment
knowledge



Clément OZOUF

clement.ozouf@abo.fi

Student number: 42042

November 2018

Master of Science Thesis
Supervisor: Dr. Sébastien Lafond
University: Åbo Akademi University
Faculty of Science and Engineering
Embedded Systems Laboratory

Company: SAFRAN Electronics & Defense
Tutor Company: Marc-Emmanuel Coupvent des Graviers

I on X

© Propriété de Safran Electronics & Defense – Reproduction et Divulgateion interdites/Disclosure and copy prohibited

This document and the information contained herein are the property of Safran. They must not be copied or disclosed to third parties without prior Safran written authorization

ACKNOWLEDGMENTS

First, I would like to thank Sebastian LAFOND, senior lecturer at Åbo Akademi University, who supported me during my studies in Finland and supervised my thesis works.

Moreover, I would like to express my most sincere gratitude to Marc-Emmanuel COUPVENT DES GRAVIERS, system architect engineer at SAFRAN Electronics & Defense, to his advices, for challenging and motivating me during the project period.

Finally, I want to thank gratefully all the SAFRAN Electronics & Defense employees for assisting and allowing me to conduct my research for that project.

ABSTRACT

The idea of an autonomous car has long been a dream. Now, thanks to new technologies coming on the market this dream is becoming reality. In the same way as Star Trek imagined 3D printing, Knight Rider featured an example of an autonomous car.

Currently in this age of autonomous devices, many companies such as Waymo and Uber are developing their own autonomous cars using different specificities and development ideas.

Having a suitable and reliable system capable of moving safely on the roads of the world is now possible by combining different environmental information. A majority of systems use GNSS, LiDAR, IMU amongst others to obtain a safe device which meets the safety integrity level (SIL) standards.

The project CAMPUS is split due to the complexity of an autonomous car project.

One part of the problem is the positioning of the device. It is common to use the Global Navigation Satellite System (GNSS), even if GNSS suffers from signal properties such as reflection or refraction, transmission, absorption.

One of GNSS's major issues is its capacity to locate a car inside an urban canyon with an acceptable level of accuracy. Therefore, the focus of my thesis is to develop a software solution based on a multipath rejection algorithm, which has knowledge of the environment.

SAFRAN targets one of the best positioning systems for autonomous cars. The first targeted project is CAMPUS.

Keywords: GNSS rejection, Urban Canyons, LOS, NLOS Multipath, 3D-Map, Shadow matching, Position candidates.

III on X

© Propriété de Safran Electronics & Defense – Reproduction et Divulgence interdites/Disclosure and copy prohibited

This document and the information contained herein are the property of Safran. They must not be copied or disclosed to third parties without prior Safran written authorization

CONTENTS

| | |
|--|------|
| Acknowledgments..... | II |
| ABSTRACT..... | III |
| Contents | IV |
| Abbreviations and Terms..... | V |
| List of Figures | VIII |
| List of Tables | IX |
| List of SQL Requests..... | IX |
| List of Equations | X |
| 1. Introduction..... | 1 |
| 1.1 Scenario Introduction and Background..... | 2 |
| 1.2 Project Objective..... | 7 |
| 1.3 Thesis Structure..... | 10 |
| 2. Design stage..... | 11 |
| 2.1 Project requirements..... | 11 |
| 2.1.1 Hardware Environment..... | 11 |
| 2.1.2 Software and Firmware environments..... | 14 |
| 2.1.2.1 Software and firmware for the PIKSI Package..... | 14 |
| 2.1.2.2 Development tools | 14 |
| 2.2 Algorithm Explanation..... | 18 |
| 2.2.1 Algorithm's Design | 18 |
| 2.2.2 Overview of the Algorithm..... | 21 |
| 2.3 Algorithm Testing Design..... | 29 |
| 3. Implementation stage | 32 |
| 3.1 Development of the Algorithm under Python | 32 |
| 3.2 Code Testing | 51 |
| 4. Evaluation | 56 |
| 4.1 Code review..... | 56 |
| 4.2 Tests Review | 58 |
| 5. Conclusion | 60 |
| Bibliography | 62 |

IV on X

© Propriété de Safran Electronics & Defense – Reproduction et Divulgateion interdites/Disclosure and copy prohibited

This document and the information contained herein are the property of Safran. They must not be copied or disclosed to third parties without prior Safran written authorization

ABBREVIATIONS AND TERMS

Project

- **CAMPUS** Connected Automated Mobility Platform for Urban Sustainability

Positioning systems

- **GNSS** Global Navigation Satellite System
- **GPS** Global Positioning System: American GNSS
- **RTK GPS** Real-Time Kinematic Global Positioning System
- **GLONASS** GLObalnaya NAVigatsionnaya Sputnikovaya Sistema: Russian Global Navigation Satellite System
- **GALILEO** European Global Navigation Satellite System
- **BDS** BeiDou Navigation Satellite System, Chinese Global Navigation Satellite System

Signal Status

- **LOS** Line-Of-Sight
- **NLOS** Non-Line-Of-Sight

Positioning Protocols

- **SBP** Swift-Navigation Binary Protocol
- **CRC16-CCITT** Cyclic Redundancy Check following a sixteen polynomial order $(x^{16} + x^{12} + x^5 + 1)$ equation to check frame errors of the protocol

Coordinate systems standards

- **WGS84** World Geodetic System 1984
- **ECEF** Earth-Centered, Earth Fixed coordinate system
- **PZ-90.02** Parametry Zemli 1990: Russian equivalent to the *ECEF* coordinate system
- **LB-93** Lambert 1993: French cartography standard coordinate system

Geographical knowledges

- **Geodetic** From a straight plane to a curved plane
- **Geoid** Ocean's surface shapes of the earth under the earth gravity and rotation influence.

V on X

© Propriété de Safran Electronics & Defense – Reproduction et Divulgateion interdites/Disclosure and copy prohibited

This document and the information contained herein are the property of Safran. They must not be copied or disclosed to third parties without prior Safran written authorization

GNSS Data

- **GPS Ephemeris**
 - **ToE** List of orbital parameters from GPS
Time of Ephemeris, it represents the GPS Week number in second
 - **C_{rs}** Amplitude of the sine harmonic correction term to the orbit radius
 - **C_{rc}** Amplitude of the cosine harmonic correction term to the orbit radius
 - **C_{uc}** Amplitude of the cosine harmonic correction term to the argument of latitude
 - **C_{us}** Amplitude of the sine harmonic correction term to the argument of latitude
 - **C_{ic}** Amplitude of the sine harmonic correction term to the angle of inclination
 - **C_{is}** Amplitude of the sine harmonic correction term to the argument of latitude
 - **Δn** Mean Motion difference
 - **M₀** Mean anomaly at reference time
 - **Ecc** Eccentricity of satellite orbit
 - **a** Semi-major axis of orbit
 - **Ω_0** Longitude of ascending node of orbit plane at weekly epoch
 - **$\dot{\Omega}$** Rate of right ascension
 - **ω** Argument of perigee
 - **i** Inclination angle
 - **i'** Rate of inclination angle
- **GLONASS Ephemeris**
 - **ToE** List of orbital parameters from GLONASS
Time of Ephemeris, it represents the GPS Week number in second
 - **Γ** Relative deviation of predicted carrier frequency from nominal
 - **τ** Correction to the SV time
 - **$\Delta\tau$** Delay between L1 and L2 GLONASS band
 - **Pos** Position of the GNSS in PZ-90.02 coordinate system
 - **Vel** Velocity vector of the GNSS in PZ-90.02 coordinate system
 - **Acc** Acceleration vector of the GNSS in PZ-90.02 coordinate system
 - **Fcn** Frequency slot.
- **SNR(dB)** Signal-to-Noise Ratio in Decibels
- **Pseudo-range** Pseudo distance between a satellite and a receiver
- **C/N₀ (dB/Hz)** Carrier-to-Noise-Density Ratio in Decibels per Hertz
- **UTC** Universal Time Coordinated
- **Azimuth** Angular measurement in a spherical coordinate system
- **Elevation** Angular measurement between a point placed above the geoid and another placed above the first one such as a satellite.

VI on X

© Propriété de Safran Electronics & Defense – Reproduction et Divulgateion interdites/Disclosure and copy prohibited

This document and the information contained herein are the property of Safran. They must not be copied or disclosed to third parties without prior Safran written authorization

Sensors, devices and connectors

- **LiDAR** Light Detection And Ranging
- **IMU** Inertial Measurement Unit
- **PIKSI Multi** GNSS receiver manufactured by Swift Navigation company
- **PMEB** PIKSI Multi Evaluation Board
- **CPU** Central Processing Unit
- **RS-232** Recommended Standard **232** connector
- **USB** Universal Serial Bus connector
- **MMCX** Micro-Miniature CoaXial connector
- **SMA** SubMiniature version **A** connector
- **TNC** Threaded Neill–Concelman connector

Programming Languages and tools

- **Python** Interpreted High-Level programming language
- **SQL** Structured Query Language, domain-specific language to manage data from databases
- **Swift Console** Software provided by the Swift Navigation company
- **PostgreSQL** SQL interpreter to execute SQL requests.
- **PostGIS** PostgreSQL extension to operate geometrical SQL requests.
- **SRID** Spatial Reference system **ID**entifier number, related to a system of coordinate, *WGS84* = 4326 for instance.

Swift Navigation Binary Protocol registers

- **MSG_EPHEMERIS_GPS** Registers that contains orbital parameters of GPS
- **MSG_EPHEMERIS_GLO** Register that contains orbital parameters of GLONASS
- **MSG_OBS** Register that contains raw pseudo-range and carrier phase of satellites
- **MSG_POS_LLH** Register that contains the receiver position in WGS84
- **MSG_UTC_TIME** Register that contains UTC time data
- **MSG_RESET** Register to restart the PIKSI Multi

VII on X

© Propriété de Safran Electronics & Defense – Reproduction et Divulgateion interdites/Disclosure and copy prohibited

This document and the information contained herein are the property of Safran. They must not be copied or disclosed to third parties without prior Safran written authorization

LIST OF FIGURES

| | |
|--|----|
| Figure 1. LOS and NLOS signals in an urban canyon [6] | 2 |
| Figure 2. Design of the satellite visibility prediction [6] | 4 |
| Figure 3. 3D City model [7]..... | 5 |
| Figure 4. Different levels in a Self-Driving Car [9]..... | 8 |
| Figure 5. LHCP and RHCP reception from https://www.linksystems-uk.com | 11 |
| Figure 6. PIKSI package connection [12]..... | 12 |
| Figure 7. Black box View of the algorithm. | 18 |
| Figure 8. Azimuth and Elevation angles..... | 19 |
| Figure 9. White box View of an ephemeris data. | 21 |
| Figure 10. White box View of the computation from the ephemeris data..... | 22 |
| Figure 11. White box View of Elevation angle computation. | 23 |
| Figure 12. Example of map of the fifth arrondissement of Paris..... | 24 |
| Figure 13. White box View of the ray & buildings intersections. | 25 |
| Figure 14. Grid around the receiver point..... | 26 |
| Figure 15. Coefficient chart. | 27 |
| Figure 16. White box view of the shadow matching. | 28 |
| Figure 17. SBP frame [15]..... | 32 |
| Figure 18. Little endian format. | 34 |
| Figure 19. Planes projection. | 41 |
| Figure 20. Multi Polygon example | 44 |
| Figure 21. Intersection point inside a surface. | 48 |
| Figure 22. <i>SBP</i> and <i>Swift Console</i> data comparison. | 51 |
| Figure 23. GNSS Latitude and Longitude from the algorithm. | 52 |
| Figure 24. GNSS Latitude and Longitude from <i>in-the-sky.org</i> | 52 |
| Figure 25. Conversion of coordinate systems from the program..... | 52 |
| Figure 26. Coordinate systems conversion from <i>oc.nps.edu</i> website. | 53 |
| Figure 27. LOS and NLOS detection..... | 53 |
| Figure 28. Three best scores and GPS not visible. | 55 |
| Figure 29. Best candidate position..... | 55 |

VIII on X

© Propriété de Safran Electronics & Defense – Reproduction et Divulgateion interdites/Disclosure and copy prohibited

This document and the information contained herein are the property of Safran. They must not be copied or disclosed to third parties without prior Safran written authorization

LIST OF TABLES

| | |
|--|----|
| Table 1. Score setting of the shadow matching algorithm [6]. | 4 |
| Table 2. Scoring pattern. | 27 |
| Table 3. Harmonic correction terms | 35 |
| Table 4. Localization of the intersection point. | 49 |

LIST OF SQL REQUESTS

| | |
|---|----|
| SQL Request 1. Connection request | 17 |
| SQL Request 2. Query to recover the identifiers of the buildings. | 45 |
| SQL Request 3. Extraction into the <i>WGS84</i> coordinates system..... | 45 |

LIST OF EQUATIONS

| | |
|--|----|
| Equation 1. Plane Equation..... | 24 |
| Equation 2. Point belonging to a plane. | 25 |
| Equation 3. Definition of Earth's constants | 38 |
| Equation 4. Calculation of basic GPS parameters from [21]..... | 38 |
| Equation 5. True anomaly calculation from [21]..... | 39 |
| Equation 6. Eccentric anomaly calculation from the true anomaly from [21]..... | 39 |
| Equation 7. Eccentric anomaly definition..... | 40 |
| Equation 8. Ek iterative calculation [21]. | 40 |
| Equation 9. Argument of latitude from [21]. | 40 |
| Equation 10. Second Harmonic Perturbations from [21]..... | 41 |
| Equation 11. Corrected parameters calculation from [21]..... | 41 |
| Equation 12. Orbital plane projection from [21]. | 42 |
| Equation 13. Corrected longitude ascending node from [21]. | 42 |
| Equation 14. Corrected longitude ascending node from [21]. | 42 |
| Equation 15. Elevation angle equations [25]. | 43 |
| Equation 16. Definition of a ray | 47 |
| Equation 17. Coordinates of the point of intersection. | 47 |
| Equation 18. Rectangle area | 48 |
| Equation 19. Heron's formula. | 49 |
| Equation 20. William Kahan's formula..... | 49 |
| Equation 21. Score computation for each point..... | 50 |

X on X

© Propriété de Safran Electronics & Defense – Reproduction et Divulgation interdites/Disclosure and copy prohibited

This document and the information contained herein are the property of Safran. They must not be copied or disclosed to third parties without prior Safran written authorization

Chapter 1

1. INTRODUCTION

A key functionality of the **Connected Automated Mobility Platform for Urban Sustainability (CAMPUS)** project is the ability to detect **Line-Of-Sight (LOS)** signals and to distinguish between **LOS** and **Non-Line-Of-Sight (NLOS)** signals. The major goal is to determine the best position of a receiver located in an urban canyon.

An urban canyon is a location in a built-up area surrounded by a high density of buildings, just as a natural canyon is in a deep valley. Sites of location defined as urban canyons are common in big cities and cause a significant number of problems as to the choice of the positioning of devices.

Because **Global Navigation Satellite System (GNSS)** signals can be received as **LOS** or **NLOS** signals, they create issues. Indeed, the reception of **NLOS** signals is a common problem, which exists in the wireless communication field of study. Moreover, the reception of these **NLOS** signals produces a delayed propagation time due to the wave reflection phenomenon, i.e. signals are reflected on walls of buildings or various objects present in these environments.

Research and work conducted on the **CAMPUS** project aim to achieve a more accurate positioning estimation of an autonomous car moving in urban canyons situations using **GNSS** signals. However, due to atmospheric distortion and reflected signals, errors of several metres persist.

To compensate positioning errors, data from devices and sensors, such as **Light Detection And Ranging (LiDAR)** or **Inertial Measurement Unit (IMU)**, are merged to obtain the most accurate tracking device. In addition, algorithms and computations such as image processing treatment are developed to avoid conflicts of localization and to ensure user safety.

Respecting the requirements established in the project's specifications, the car has to drive on a road of two and a half metres and be able to detect and react according to its environment.

1.1 Scenario Introduction and Background

One aspect of the CAMPUS project aims to determine when a LOS and NLOS signals are received if a car is situated inside an urban canyon.

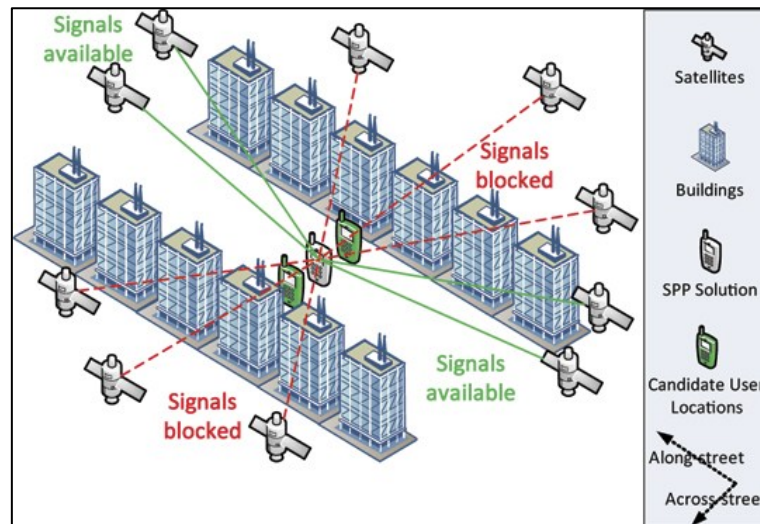


Figure 1. LOS and NLOS signals in an urban canyon [6]

According to several studies and research, such as references from [1] to [7], methods have already been implemented in the past to distinguish between LOS and NLOS signals. However, they took time and required a complex adaptation for the moving devices.

The following paragraphs offer a review of the existing methods. More than a state-of-the-art, it allows the designers of the project to build a system without making the same mistakes.

First, it is fundamental to understand the GNSS principles [1]. What makes the difference between signals and how it is possible to define and distinguish between LOS and NLOS signals are essential questions that need to be answered before conducting advanced research in that field of study.

On the one hand, according to [2], a receiver device receives LOS signals. Because the signal comes directly from a point A to a point B, there is no reflection of waves. On the other hand, NLOS signals cause multipath, a propagation phenomenon, which happens if a signal is received from two or more paths. This situation occurs when a wave bounces off a surface, such as a wall or a window.

From [2], Peyraud S, Bétaille D, Renault S, Ortiz M, Mougel F, Meizel D and Peyret F explain a smart way to make the difference. The principle is to inspect the **Signal-to-Noise Ratio (SNR (dB))**. In addition, they extrapolate conditions on the *SNR (dB)* behaviour of signals. For instance, if the ratio is mostly constant over time, the signal is potentially direct, and then it can be identified such as a LOS signal.

In contrast, in the case of a constant decibel level, if the SNR (dB) varies too much or disappears over time, the signals can be defined as NLOS signals.

Reinforcing the fact that a signal can be distinguished as either LOS or NLOS, Paul D. Groves in [3] explains that the values and behaviours of the **Carrier-to-Noise-Density Ratio** (C/N_0 (dB/Hz)) also changes along the time and can be related to the SNR (dB). Furthermore, he expresses in [3] and [4] that the SNR (dB) and the C/N_0 (dB/Hz) levels give indications on the states of the signals.

If these received ratios are low, it is probably due to a multipath reception of the signal. However, if the SNR (dB) or C/N_0 (dB/Hz) levels are high, this indicated a direct signal path reception.

Researchers Peyraud S, Bétaille D, Renault S, Ortiz M, Mougel F, Meizel D and Peyret F were inspired by the use of an environment map from the [4] reference. They decided to reuse this principle to merge several data outputs, by correlating SNR (dB) levels with a three-dimensional data map. Researchers were able to pinpoint precisely whether a signal comes from a direct or an indirect path.

Indeed, data of the map provide clues on the environment situation. Thus, it is possible to know if some objects are placed on the path drawn by the *receiver-GNSS* ray. In order to determine whether an object crosses the segment, they had to know both the positions and orientations of both the receiver and a GNSS. In addition, they had to know the azimuth and the elevation angles measured between both points. Without this information, they could not compute any results of a potential crossing point between two points and an obstacle.

However, knowing whether a satellite is visible or not is less complicated than an analysis of received signals. Indeed, it is extremely complicated to know whether a signal is reflected or directly received.

Many researchers and developers have tried to determine if signals come from multipath or direct paths. One method is the Bayesian approach explained in [5]. That approach is a probabilistic method. Based on signals filtering, Closas P, Fernandez-Prades C. and Fernandez-Rubio J proposed a Particle Filter algorithm to reduce the effect of multipath signals based on GNSS settings. In order to test and evaluate their algorithm, they used a simulation model. However, they were not totally convinced that solution was not the most efficient and they realized their method did not work in every case.

Concerning the approach expressed by researchers of [2], they based their works on the [3] and [4] methods, which were strongly related to the shadow matching one. P.D Groves named this method in [3] due to the 'shadow' provided by tall surfaces.

The principle summarized by authors of [4] in [6] is:

A GNSS receiver collects direct (LOS) and indirect (NLOS) signals. With the use of algorithms, the receiver can estimate its position. However, due to the multipath of signals and environmental parameters, the position can be more or less accurate. Because of the lack of precision, the shadow matching algorithm is run to predict the satellite visibility across the time in accordance with the position of the receiver. Then, the algorithm names these satellites as prospective candidates. Each candidate is predicted due to its potential visibility from a computation of azimuths and elevation angles according to a map of the environment.

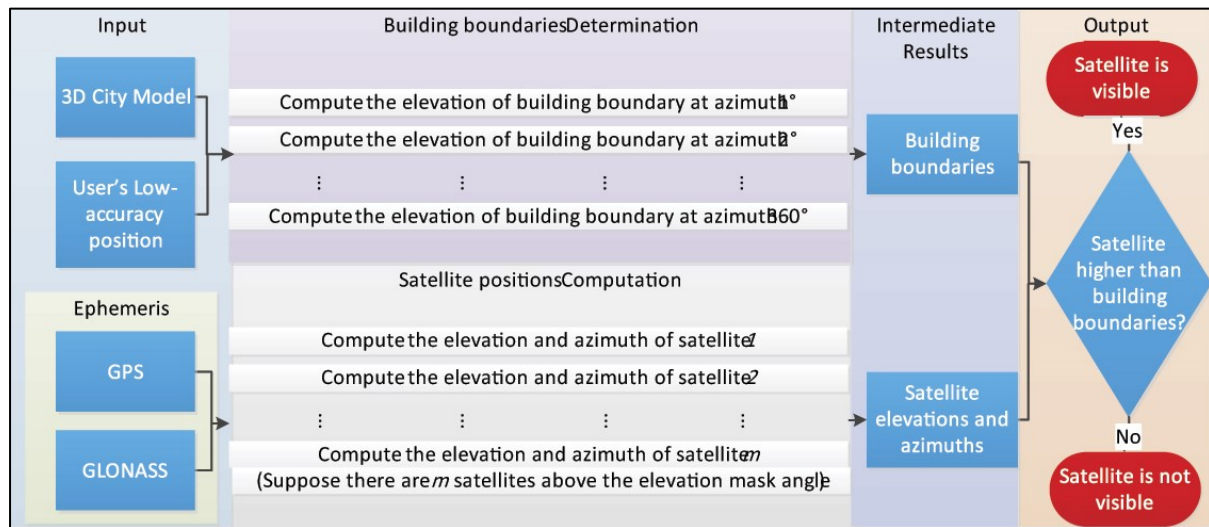


Figure 2. Design of the satellite visibility prediction [6]

From the previous prediction model and the real observation, visible GNSS at a time t . A match is made between the model and the reality.

A score is set for each candidate depending on the matching of the following conditions, between the prediction and the real observation.

If a GNSS from prediction and observation is:

| Observation/Prediction | Invisible | Visible |
|------------------------|-----------|---------|
| Untracked | 1 | 0 |
| Tracked | 0 | 1 |

Table 1. Score setting of the shadow matching algorithm [6].

A scoring evaluation is operated by summing up the previous matching scores. Then, the selection of one candidate is computed according to the highest shadow matching result.

This method of positioning correction is one of the most recent and best ways to identify NLOS signals and transmitters that provide too much noise and inaccurate position of receivers. However, it is not computed on real-time systems because of the time required to process the prediction algorithm. Furthermore, it requires several receivers, which is not the case of the CAMPUS project.

Other methods exist to identify if a signal suffers from multipath. Nicolas Viandier produced a thesis [7] in 2011 based on one possible method. His method seems to be more feasible if it is computed on a real-time system. From pages 83 to 87, he defines precisely all the terms and effects that can be caused by LOS and NLOS signals on a position estimation.

Next, he describes his own method which is a statistical method based on the pseudo-range errors and the identification of multipath.

As illustrated in Figure 3.1 of [7] or in Figure 3, he also uses a three-dimensional map model to determine to what extent a GNSS is masked by a building.

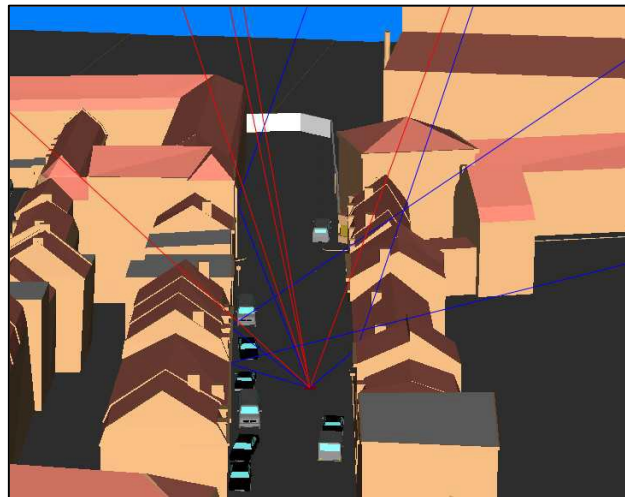


Figure 3. 3D City model [7]

Using a statistical model on an important amount of data, he succeeds in identifying whether or not a signal suffers from multipath. In order to identify reflected and NLOS signals, he prefers to use the Carrier-to-Noise-Density ratio, C/N_0 (dB/Hz) as the third reference, [3], did.

The C/N_0 (dB/Hz) level corresponds to the SNR (dB) level of a modulated signal. All types of local disruptions affect the electromagnetics characteristics of a signal, namely phase, amplitude and frequency. Therefore, when a disruption is detected, a signal loss is observed. That loss affects the C/N_0 (dB/Hz) level, e.g. a potential environmental disruption is related to the elevation angle defined between a receiver and a transmitter.

From table 3.2 of [7], Nicolas Viandier demonstrates a correlation between the C/N_0 (dB/Hz) level and the elevation angle. In addition, the C/N_0 (dB/Hz) level of a GNSS can be more or less fluctuating depending on its localization in space.

Past research has enabled the designers of the project to choose the best methods for the project goals and positioning objectives

1.2 Project Objective

Research has been conducted for the **Connected Automated Mobility Platform for Urban Sustainability** project (CAMPUS).

CAMPUS is a project involving **VALEO** and **SAFRAN** groups in association with the **French Environment & Energy Management Agency**, **FEEMA** or **ADEME** in French.

The ‘**Société d’Applications Générales d’Electricité et de Mécanique**’ also known as **SAGEM**, was created by Marcel MOME in 1925. SAGEM manufactured industrial and machine tools until the nineteenth. In 2005 and in accordance with ‘**Société Nationale d’Etude et de Construction de Moteurs d’Aviation**’ or **SNECMA** company, the SAFRAN group was created in 2005.

It is an international group regrouping several companies with activities ranging from the manufacturing of aeronautic or aerospace equipment to the development of algorithms of navigation. SAFRAN is also involved in the French Defence sector. Furthermore, the company in charge of the positioning system of CAMPUS is SAFRAN Electronics & Defense. More information is present in [8].

SAFRAN Electronics & Defense took part in a project named: ‘**Véhicule routier et mobilité du futur**’ which can be translated as ‘**Road vehicle and mobility for the future**’. CAMPUS is related to the goals and issues that concern autonomous vehicles for the future. It aims to identify, develop and deploy new functionalities for the autonomous driving in urban and suburban environments. In addition, technologies developed for CAMPUS have to introduce the first ‘real’ solutions of autonomous driving between 2021 and 2024.

A major goal of the project is to be able to build an urban high-end autonomous vehicle up to level three or four. Urban environment means that the vehicle speed is limited to 20 metres per second and buildings scramble GNSS signals.

In practice, various levels of automation exist and allow experts to classify autonomous devices depending on their features. Levels are defined from zero to five and must answer to different goals and purposes.

Figure 4, from [9] reference, summarizes levels of automation

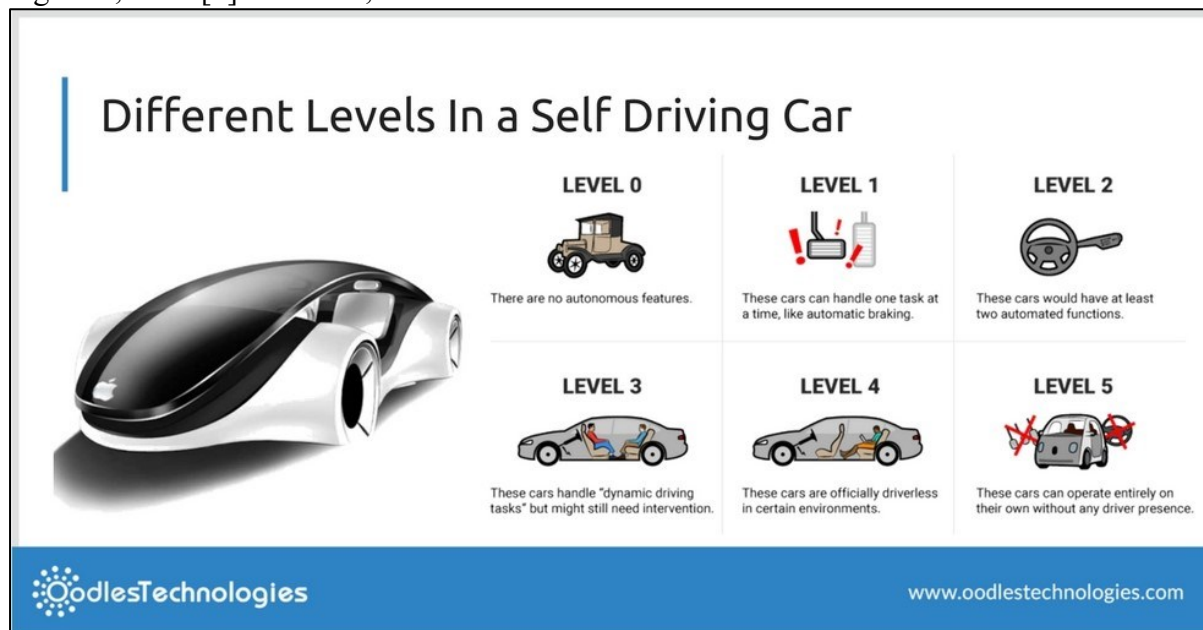


Figure 4. Different levels in a Self-Driving Car [9]

Worldwide companies are currently able to provide autonomous vehicles answering to a level two. However, they would like to produce autonomous vehicles at level 3 before 2022. Level 3 autonomous vehicles should drive in predefined environments, e.g. on a highway or in congested traffic – traffic jams. Nonetheless, the vehicle's driver has to be ready to take back control at any moment.

Consequently, systems that are more complex have to be developed to handle the high-density environments, such as people, vehicles or buildings. Compared with oldest releases, new autonomous cars should be able moving on a two-direction road.

In order to meet these goals, a significant number of different sensors are embedded in the autonomous car of CAMPUS.

One sensor set is a double detection 'cocoon', surrounding the car by 360 degrees. That 'cocoon' merges the images processing and the environment detection by using cameras and radars placed on the car.

One of the main sensors is the LiDAR, attached on the roof of the car to detect objects next to the car. The LiDAR technology uses the infrared light principle. A LiDAR sends and receives its own beams while it is rotating.

Finally, the car is equipped with a frontal camera to detect static and dynamic frontal objects, such as a monitoring station to identify vehicles or pedestrians.

Because the car has to be located and positioned in real-time, it is also supplied with an IMU monitoring station, which is a combination of several sensors, such as gyroscopes and accelerometers, to track position, speed and acceleration of the vehicle, and an odometer to monitor the speed of the car.

In order to obtain the most accurate position, the car is also equipped with a satellite receiver. The receiver selected is a PIKSI Multi [10], which is manufactured by the Swift Navigation company.

Ultimately, data from GNSS are manipulated to provide additional positioning measurement, independently of the equipment embedded in the car. The Research and Technology department of SAFRAN has put me in charge of producing and providing an independent software component as an accurate positioning solution. It uses the GNSS receiver to be implemented along other technologies still in development.

1.3 Thesis Structure

The core of the thesis contains three parts and it is organized as follows:

The first section: **Design stage** defines the project's requirements, explains the proposed algorithm and describes the test elaboration in accordance with the project design.

A second section: **Implementation stage** describes the development of the code based on the algorithm. It consists of a description of the Python programming language code and tests conducted during the project period.

The third section: **Evaluation** deals with an assessment of the code, using metrics to evaluate the pros and cons, and the results of tests carry out to check the validity of the solution for the project content.

As Chapter 2, the **Design stage** that follows three subsections:

- The requirements of the positioning solution. It details which materials and software are used to realize the data recovering and the operations processing.
- The algorithm explanation and overview.
- The tests carry out to ensure that the positioning solution provides the correct results.

As Chapter 3, the **Implementation stage**, subdivided into:

- The development of the software component, which explains the data gathering, processing and mathematical operations carried out all along the project.
- Tests and results of the final code.

As Chapter 4, the **Evaluation** that aims to either prove or disclaim tasks carried out during the project period.

The conclusion summarizes the major points, in accordance with the positioning solution development. It discusses future ideas of improvement and a line of thought concerning this field of research.

Chapter 2

2. DESIGN STAGE

2.1 Project requirements

CAMPUS is an embedded system project. It combines hardware and software, which allows providing a level three autonomous car for its final users.

2.1.1 Hardware Environment

The car of the CAMPUS project is equipped with an important number of sensors including a GNSS receiver. The PIKSI Multi from Swift Navigation [10] is selected because it is easily configurable and is still used on previous projects. The receiver's manufacturer is providing a suitable code for multiple programming languages, through a GitHub repository.

The positioning system uses a development kit for the PIKSI Multi. A survey antenna is provided along with it. This antenna picks GNSS signals located around the antenna's position.

The antenna receiving mode is important too. In truth, two receiving modes exist for antennas, either the **Left Hand Circular Polarized (LHCP)** or **Right-Hand Circular Polarized (RHCP)**. They are illustrated in Figure 5.

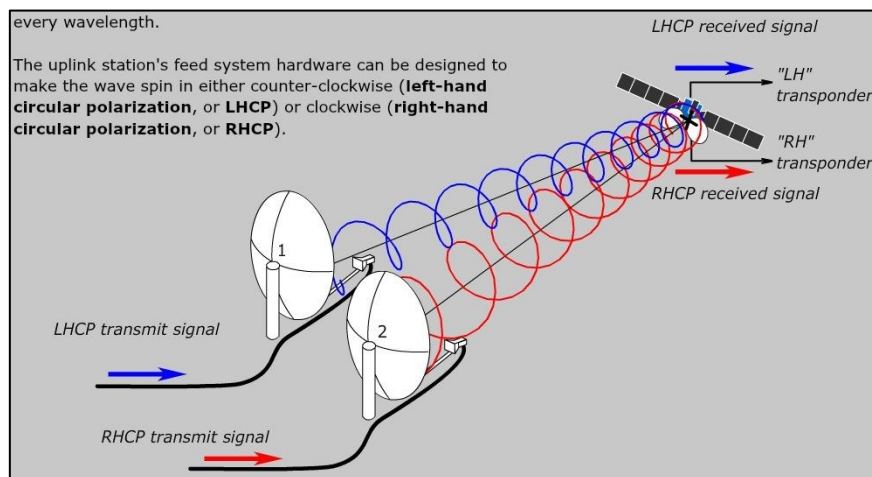


Figure 5. LHCP and RHCP reception from <https://www.linksystems-uk.com>

In accordance with the *SNR* (dB) error level, analyses performed through [11] conclude that the RHCP mode is more reliable than the LHCP.

Furthermore, the **Real Time Kinematic Global Positioning System (RTK GPS)** technology was created to enhance the reception of signals. It provides a costly correction of signals by

placing several reception bases around a studied location. Nevertheless, the project's car must move autonomously in any situation, even if a location is not totally equipped with bases. The car is a moving object and therefore the RTK GPS technology cannot be put into practice. Ultimately, the RTK GPS technology cannot be embedded in the car because it is a costly solution and cannot reach a wide audience.

Furthermore, the **PIKSI Multi Evaluation Board (PMEB)** [12] and [13] applies to connect the PIKSI Multi to a computer. The evaluation board provides multiple ways to connect the receiver to several external devices. It has six female ports including two **Recommended Standard 232 (RS-232)**, two **Control Area Network (CAN)**, one **Universal Serial Bus (USB)** and one Ethernet.

Moreover, there are two **Light-Emitting Diodes (LED)**, placed on the top of the **PMEB**, which sends visual feedback on a signal reception. **Pulse Per Second (PPS)** and **Position Valid (PV)** LEDs must be turned on to ensure the PIKSI Multi antenna receives data.

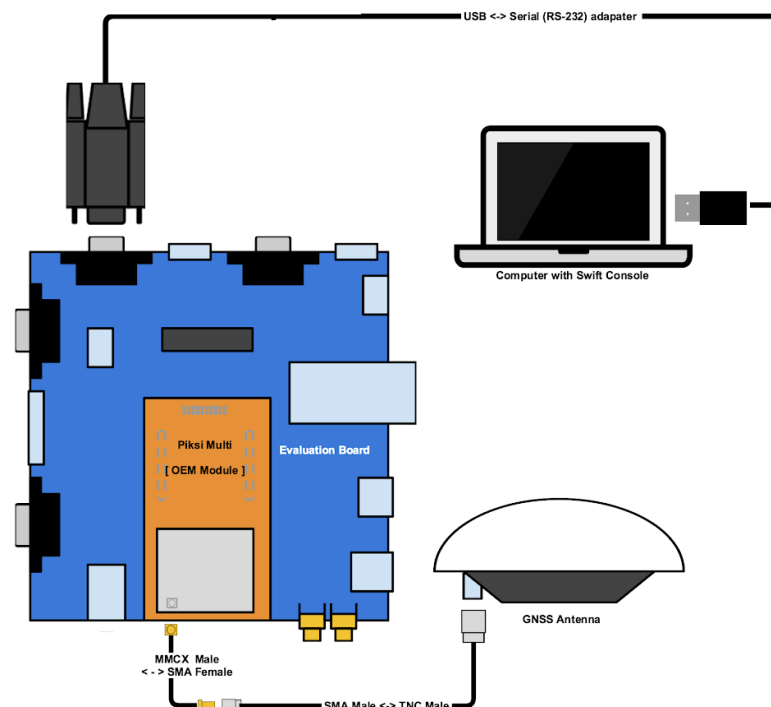


Figure 6. PIKSI package connection [12]

Thanks to the company, connecting cables are included with the PIKSI Multi. For example, connections are:

- From **RS-232** to **USB** wire, to connect a computer to the **PMEB**.
- From a **Micro-Miniature CoaXial (MMCX)** male to a **SubMiniature version A (SMA)** female connector and from a **SMA** male to a **Threaded Neill–Concelman (TNC)** male connector to enable the *PIKSI Multi-antenna* connection.

Every cable must be connected as shown in Figure 6 and on the Swift Navigation website [12].

Furthermore, the PIKSI Multi Hardware Specification [14], Swift Navigation Binary Protocol documentation [15] and an online specification document [16] are available on the website of Swift Navigation. They allow users to understand how to connect devices and what the protocol's data means.

A hardware system must not lead to errors. Thus, material issues have not to happen. The compliance of hardware to standards may avoid issues on a board. Furthermore, for the reliability of the system, there will be no need for any hardware change in the future.

Besides the PIKSI Multi package, there must be a processing unit that will process the data and run the algorithm. The development computer used for the Python prototyping is an HP EliteDesk 800 G3 TWR, a standard desktop.

Hardware conflicts and constraints are not the only possible causes of problems. Software components also generate their own issues.

2.1.2 Software and Firmware environments

This section focuses firstly on the software and firmware used to set, test and begin to work with the receiver hardware: the PIKSI package.

Secondly, it enumerates and explains the software and libraries employed to develop the code and to compute positioning, crossing and comparison operations.

2.1.2.1 Software and firmware for the PIKSI Package

Swift Navigation does not only provide to its clients the connection documentation, it also supplies a wide range of documents, i.e. software, firmware and testing tools documentation to configure and develop their devices.

To set the GNSS receiver, a specific version of the *Swift Navigation Firmware* and a Graphical User Interface (GUI), named *Swift Console*, are freely available on the company's website.

Thanks to the GUI, hardware is easily configurable. For instance, it is possible to choose which GNSS messages are sent from the receiver to the computer. The hardware settings enable users to avoid data floods and gaining computation time.

To ensure the program received the data from satellites, *Swift Console* was particularly helpful. It generates a readable comparison of information between its data and the potential program outputs.

2.1.2.2 Development tools

Python open source libraries are also used to perform processing such as reference conversion or data storage.

Database tools

A three-dimensional environment database of the fifth arrondissement of Paris was created for the purpose of another project. Because the fifth arrondissement of Paris meets the urban canyon criteria, the database is reused to distinguish between LOS and NLOS signals reception.

The database was built using *PostgreSQL*, a Structured Query Language (SQL), database system and a suitable geometrical SQL extension the system, *PostGIS*.

PostgreSQL

PostgreSQL is an open source object-relational database system. This system is able to import, connect and create tables and executes queries using *SQL* command lines. Using a Python library to communicate directly with the database, standard requests can be executed, such as *SELECT*, *FROM*, *ORDER BY* or *WHERE*.

PostGIS

The database is filled with buildings and roadside objects from the fifth arrondissement of Paris represented as geometrical objects, such as polygons or multi polygons.

The *SQL* queries that are executed must be adapted to the type of data. Thanks to *PostgreSQL*, a suitable system extension is available to deal with these types of queries. It is a spatial database extension named *PostGIS* which performs useful request functions, such as *ST_AsText(arguments)*, *ST_Transform(arguments)*, *ST_Force_3D(arguments)* and *ST_GeomFromText(arguments)*, detailed in section 3.1.

Python Libraries

Math Library

To compute mathematical operands and operations, python's *math* library is used. It provides many mathematical operands and functions such as trigonometric functions.

Numpy Library

Many developers around the world choose the Python programming language and an important community has emerged with time. Some of these users developed a usable and reliable way to manipulate the storage of data.

numpy manipulates N-dimensional lists, mathematical operands and computes operations from N-dimensional arrays.

Serial Library and Swift Navigation Binary Protocol explanation

A protocol of the Swift Navigation company has to be followed to read or write to the PIKSI Multi. This communication is done via a middleware library that is based on Python's *serial* library.

In order to interact with hardware, a serial link is enabled. Because it transfers data from *RS-232* to *USB*, the transmission has to operate through a specific port and the data communication must be configured using a 115,200 Hz baud rate.

Whenever an interaction between both pieces of hardware happens, an opening and closing request has to be sent. Depending on the data that should be extracted through

the serial communication, the corresponding *SBP* frame should be respected. The *SBP* frames and messages are described in [15] and useful data for the project are detailed in section 3.1.

Struct Library

To code or decode the Swift Navigation Binary Protocol faster, the *struct* library is chosen. The protocol is encrypted using undefined characters. Thanks to the *struct* library, it is possible to encrypt and decrypt messages.

To encrypt a message which meets the *SBP* requirements, it is necessary to input data as *integers*. Thus, output data are in the form of understandable characters. Moreover, it is possible to send an encrypted message from the computer to the PIKSI Multi through the serial communication via the `struct.pack` and serial command lines.

Concerning the decryption of a message, the opposite of the `struct.pack` function must be executed. The `struct.unpack` function takes *strings* as inputs and returns *integers*.

The Swift Navigation Binary Protocol [15] respects the little endian format. Each request using *struct* must be expressed with the `<` symbol placed at the beginning. Furthermore, some characters, summarized in [18], are associated to their matching *integers* depending on the variable's type. For instance, the *I* character corresponds to an *unsigned integer* with a size of four bytes.

Pyproj Library

To carry out conversions of coordinates, the python community has also developed a user-friendly library called *pyproj* [19]. It is a stable and tested solution to perform projections between coordinate systems using the right **Spatial Reference System Identifier (SRID)** of a coordinate system. The PIKSI Multi returns **World Geodetic System 1984 (WGS84)** coordinates and the database returns the **LamBert 1993 (LB-93)** coordinates. These systems are not directly usable for data computations. Moreover, outputs of calculations from PIKSI Multi data are expressed into the **Earth-Centered, Earth-Fixed coordinates (ECEF)**.

Using the *pyproj* library and the `pyproj.transform(arguments)` function, it is possible to go from one coordinate system to another.

Psycopg2 Library

SQL requests are transferred from the Python code to the three-dimensional database. The *SQLite* python library can be used to execute these *SQL* queries but because of the database's complexity, it was more reliable to use the *psycopg2* than *SQLite*. *Psycopg2*

is documented in [20]; it provides an **Object Relational Mapper (ORM)**, from *SQL* to Python, to interact with a database from the programming language.

First, a connection with the database must be established in order to execute *SQL* queries with Python on both sides of the script - database node.

SQL Request 1 illustrates which *psycopg2* structure is used to create the communication

```
Conn=psycopg2.connect ("dbname=database_name user=database_username  
host=host_name port=connection_port password=database_password")
```

SQL Request 1. Connection request

Re Module

Re is a preinstalled **Regular Expression** tester for Python. It allows users to select a specific expression and to check if there are any matches in the queried table. For instance, it can be used to search for matches between a list of coordinates and information returned by a database request.

2.2 Algorithm Explanation

Once hardware and software requirements are defined, it is time to explain the principle of the algorithm. It is able to provide the best signal detection and rejection. As mentioned in the Thesis Structure, section 2.2 is divided into three subparts. The first part is dedicated to the design of the algorithm. This is followed by an overview of the algorithm and finally the theory testing of the algorithm will be outlined.

2.2.1 Algorithm's Design

First, an algorithm able to meet decided upstream targets and goals is designed. The black box view exposed in Figure 7, illustrates the behaviour of the algorithm. It was developed to identify and reject NLOS signals, and determine the best position of a receiver:

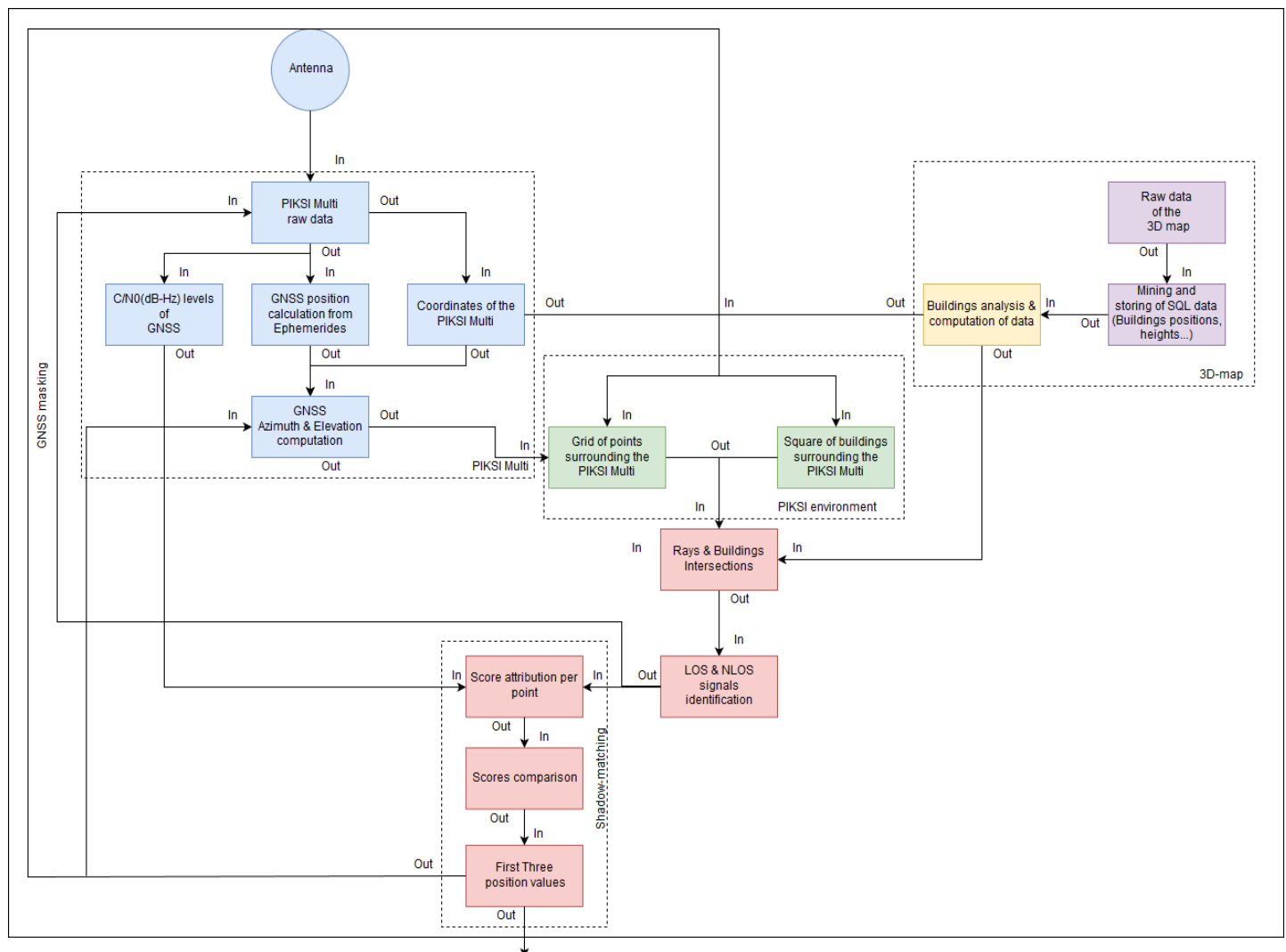


Figure 7. Black box View of the algorithm.

Raw data from the Swift Navigation Binary Protocol are received. These data are encrypted and respect a specific frame illustrated in Figure 17 (section 3.1). Data must be decoded according to the protocol. Once they have been processed, the data from the ephemeris of a satellite or the position of the receiver are available. Extracted data can be used to calculate useful information, such as the position of GNSS located in the sky above the receiver.

The most important extracted data are:

- The ephemerides of all the satellites.
- The position of the PIKSI Multi in the *WGS84* reference.
- The ‘observations’ from the satellites, i.e. pseudo-range and C/N_0 (dB/Hz) level.
- The Universal Time Coordinated (UTC).

From the data of the ephemerides, mathematical and orbital operations, X, Y and Z-coordinates, in the *ECEF* coordinate system are available for each GNSS surrounding the current PIKSI Multi position.

Subsequently, adequate mathematical operations are used in order to determine azimuth and elevation angles between the PIKSI Multi and the GNSS as illustrated in Figure 8. The results of these computations should indicate where satellites are located in space around Earth.

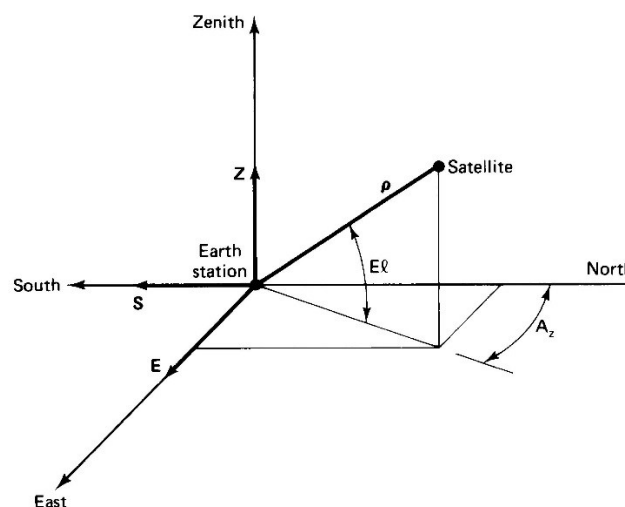


Figure 8. Azimuth and Elevation angles

In addition, encrypted data from the three-dimensional map database must be decoded and collected. The *PostgreSQL* database uses the *PostGIS* extension to store geometrical data from the environment.

Because *SQL* data respect a specific query architecture, corresponding command lines are executed to extract buildings information. When requests are executed, and *SQL* tables are queried from the database, they have to be analysed, assessed and sorted in order to acquire the best building outputs.

Because the receiver's and satellites' coordinates are now known, several rays can be cast between the receiver and the transmitters. Thanks to this information, it is possible to determine if a ray intersects a building's 3D model.

A signal status is established depending on previous results. Two potential cases are distinguished:

- If a point of intersection exists, the satellite is declared as hidden.
- If there is no intersection, the satellite is defined as visible.

Thus, a signal can be identified as LOS or NLOS. Thanks to the PIKSI Multi, it is possible to either enable or disable the GNSS listening through the Swift Navigation Binary Protocol.

In addition to the signal identification, a shadow matching algorithm has been designed in order to obtain the most accurate position.

As set forth in the state-of-the-art, authors of [4] and [6] use an algorithm of prediction and a scoring pattern. They used a score to compare results of their prediction model with the real measurements that come from GNSS.

Inspired by research conducted and compiled in [3] and [7], a correlation between the status of the signals and the C/N_0 (dB/Hz) levels of the satellites has been undertaken.

The 'real' position of the receiver is sent back using the GNSS triangulation principle. It can be more or less accurate due to the effect of multipath signals. Then, the project's developer decides to use a grid around the receiver position. A score has to be attributed to each point belonging to the grid depending on the previous correlation.

The grid size has been defined according to width of the road. It is represented as a seven-metre square. Thus, the grid is composed of forty-nine points named as potential candidates.

The intersection computations between each candidate and the buildings surrounding the PIKSI Multi will be processed. In addition, the C/N_0 (dB/Hz) level of each satellite will be used to help to determine a score (described in section 3.1). The goal of a score is to show whether a candidate is more or less reliable than another.

Once all scores are calculated, a comparison is undertaken. Because it is a score comparison, the highest score value identifies a point as the best position and the lowest score indicates the worst position. In order to always have the most precise position of the car, the best candidate affects the next forty-nine candidates.

Finally, this loop is run repeatedly, and sends the best candidate position to the next hardware chain until the car is stopped.

2.2.2 Overview of the Algorithm

Concerning the overview of the algorithm, the black box views are illustrated in section 2.2.1, Figure 7.

Section 2.2.2 is composed of the main white box views of the algorithm. Further, details about these white box views are expressed in section 3.1 in order to understand what is used to develop and implement the code.

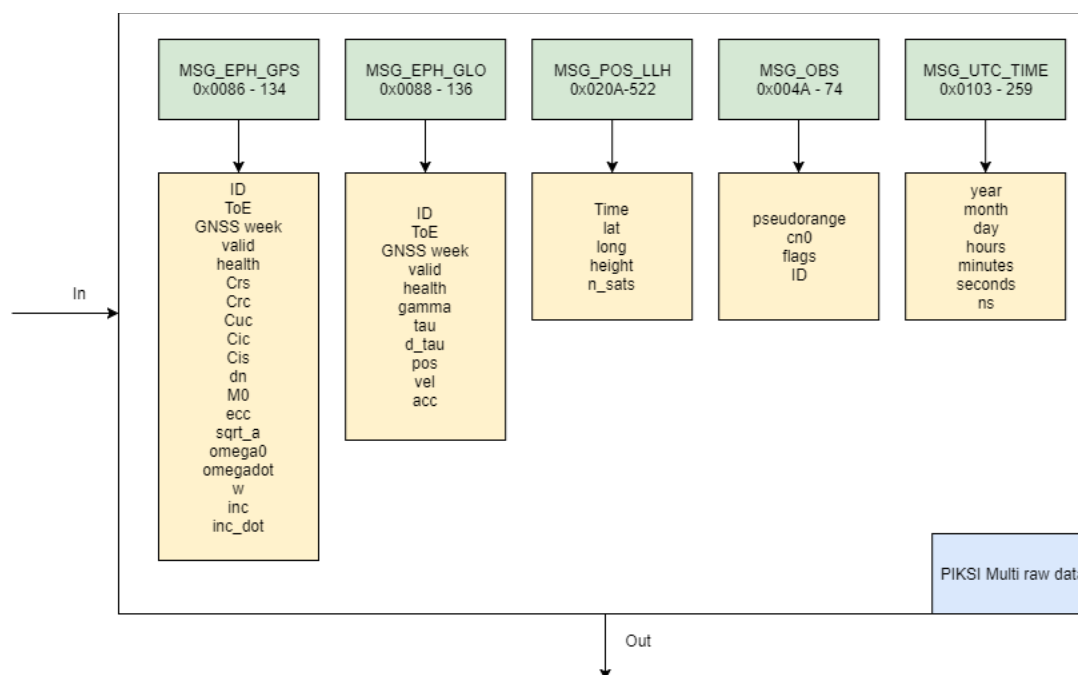


Figure 9. White box View of an ephemeris data.

The PIKSI Multi sends encrypted raw data that need to be decrypted. For instance, the Swift Navigation Binary Protocol returns GNSS ephemeris data as shown in Figure 9.

GNSS ephemeris data are tagged with the **MSG_EPHEMERIS_GPS** and **MSG_EPHEMERIS_GLO** message types. They are respectively associated to two hexadecimal values: $(0x0086)_{16} = (134)_{10}$ and $(0x0088)_{16} = (136)_{10}$.

In order to collect data from a GNSS ephemeris, its hexadecimal message type value is extracted from the header of the *SBP* protocol.

In addition, the PIKSI Multi sends back its own coordinates in the *WGS84* standard using the **MSG_POS_LLH** message type. The corresponding hexadecimal value is $(0x020A)_{16} = (522)_{10}$. As this message handles the computation of the receiver's position, it is not necessary to compute it again.

Concerning the observations data, they are stored into **MSG_OBS**, indicated by $(0x004A)_{16} = (74)_{10}$. It contains the pseudo-range, the C/N_0 (dB/Hz) level and the identification value of each satellite.

Finally, the **MSG_UTC_TIME** message type is achievable via the $(0x0103)_{16} = (259)_{10}$ hexadecimal value. It corresponds to the common time to calculate the position of satellites in the sky.

Once raw data have been collected and decrypted, the position calculation of a satellite can begin.

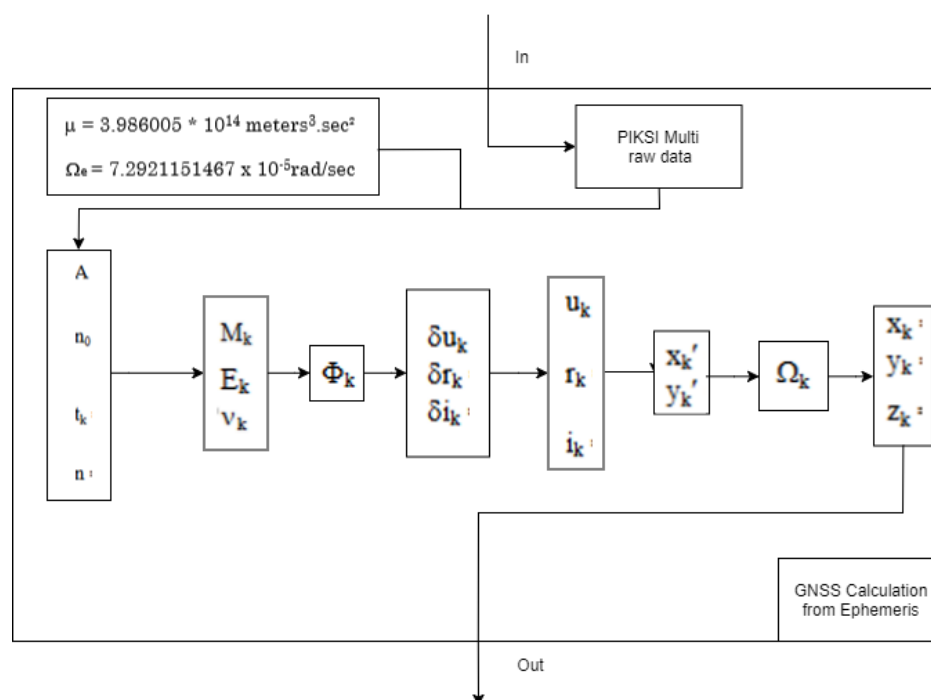


Figure 10. White box View of the computation from the ephemeris data.

Figure 10 provides a white box view of the calculation sequence used to determine the position of a satellite in the *ECEF* coordinate system. Physical constants and orbital parameters from a GNSS's ephemeris are taken into account to calculate a position. However, in order to keep clarity in the overview of the algorithm, the position calculation is detailed in the first subsection of the Chapter 3.

The next step of the algorithm is to determine the azimuth and elevation angles between the PIKSI Multi and each GNSS.

The azimuth angle is computed by a function from the *pyproj* Python library. The *pyproj* function takes the *WGS84* coordinates of two points, e.g. the PIKSI Multi and a satellite. The function returns the azimuth angles and the distance between both points. All instructions are detailed in Chapter 3, section 3.1.

The elevation angle computation follows the boxes and steps presented in Figure 11. The calculation steps were established from Figure 8 and [26].

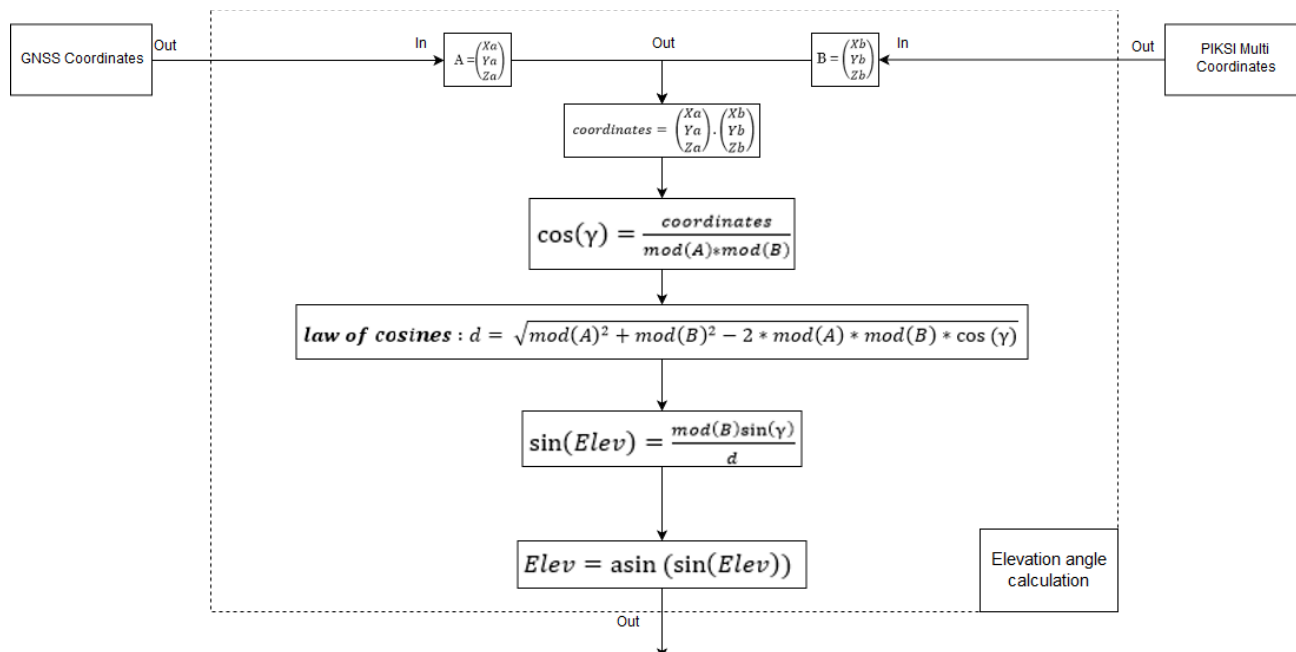


Figure 11. White box View of Elevation angle computation.

First, the coordinates of two points must be expressed in the same system of coordinates. Furthermore, the elevation angle is calculated using laws regulating the trigonometry, essentially the law of cosines. The detailed computation is explained in section 3.1.

Once raw data from the PIKSI Multi have been gathered, the database information is retrieved. The database corresponds to a three-dimensional map. Figure 12 illustrates an example of the map. Several data compose the database, such as the identification numbers, positions in the *LB-93* standard and the heights of each building located in the map.

These data are extracted from an *SQL* database and manipulated with the correct *SQL* requests. *Psycopg2* library is used to translate queries executed from Python code lines to understandable *SQL* queries.

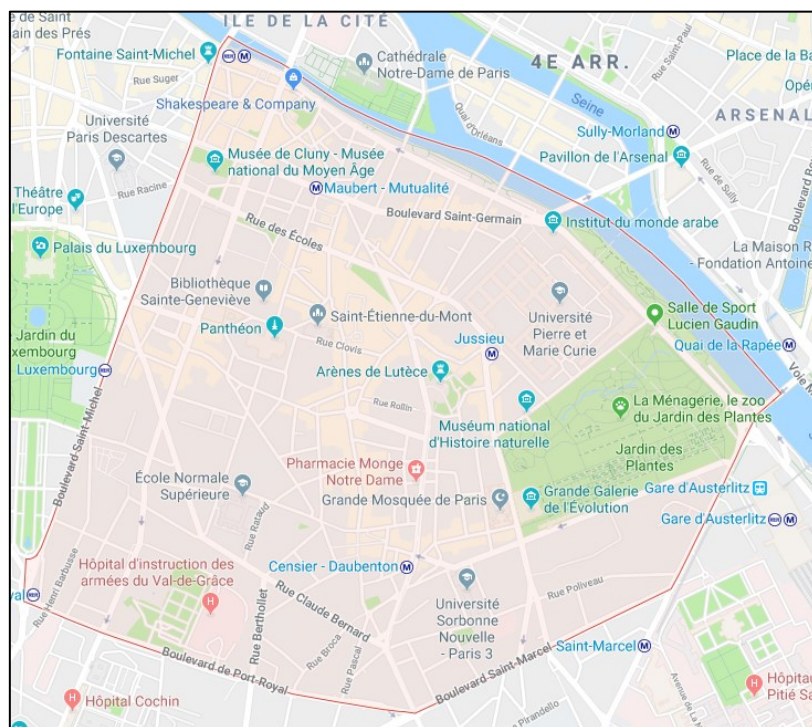


Figure 12. Example of map of the fifth arrondissement of Paris.

Moreover, surfaces of buildings can be considered as vertical planes around the PIKSI Multi. These planes follow four points of a building, two points that determine the floor and two others that define the roof of a building.

It has been decided to use surfaces rather than areas due to differences in performance between their size dimensions: two dimensions for a surface and three dimensions for an area. This method accelerates the overall execution time. Furthermore, the intersection point computation between a ray and a building is processed using this method.

Therefore, vertical planes are calculated from surfaces of buildings that surround the receiver. They respect the mathematical definition of planes, as a plane P is defined as:

Equation 1. Plane Equation

$$P = ax + by + cz + d \text{ such as } a, b, c \text{ and } d \text{ are the plane's parameters}$$

A plane is defined with a normal vector and three points of a surface. Once planes surrounding the PIKSI Multi are created, the intersection computations are run.

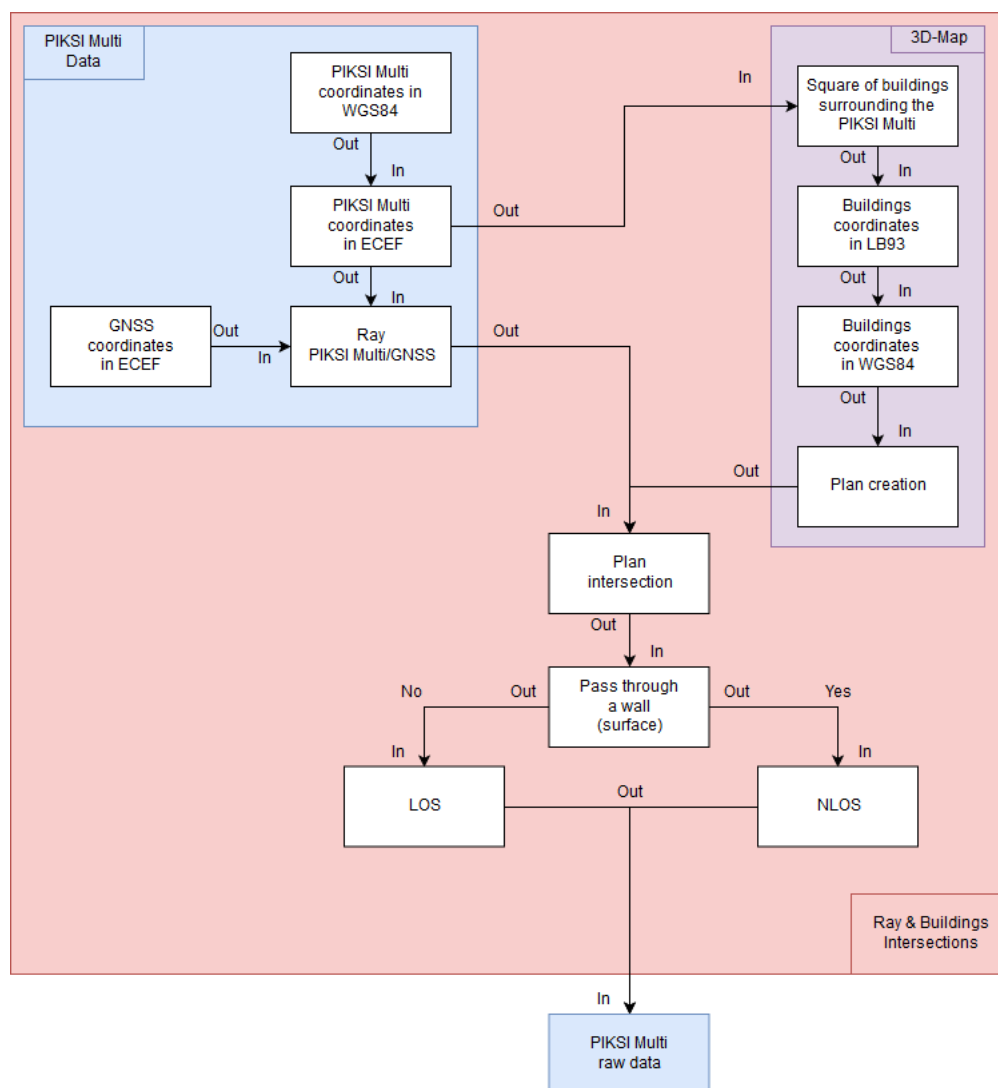


Figure 13. White box View of the ray & buildings intersections.

As shown in Figure 13, inputs of the *Plane Intersection* come from two different blocks. Outputs of the block *PIKSI Multi Data* represent the rays defined from both extremes. Planes previously created correspond to the outputs of the *3D-map* block.

In geometry, a point belongs to a plane, if and only if, the following equation is true:

Equation 2. Point belonging to a plane.

$$ax + by + cz + d = 0$$

Such as a, b, c and d are parameters defining a plane

x, y and z are coordinates defining a point

If an intersection point exists, however, it must cross the building surface, which defines a plane.

Using the mathematical operations detailed below, two options are conceivable

- If the intersection point crosses the surface, the signal is not direct and comes from a hidden GNSS.
- In other cases, the signal comes from an observable satellite and is identified as a visible GNSS.

At this algorithm stage, the signal identification is finished and the multipath rejection can be produced.

To reject a GNSS, it is designed to use a shadow matching method. This process consists of assigning a *score* to a candidate. The value of a *score* is based on the quality of a signal and on the status of a satellite, either LOS or NLOS. N -candidates are placed virtually around the receiver to obtain the best position.

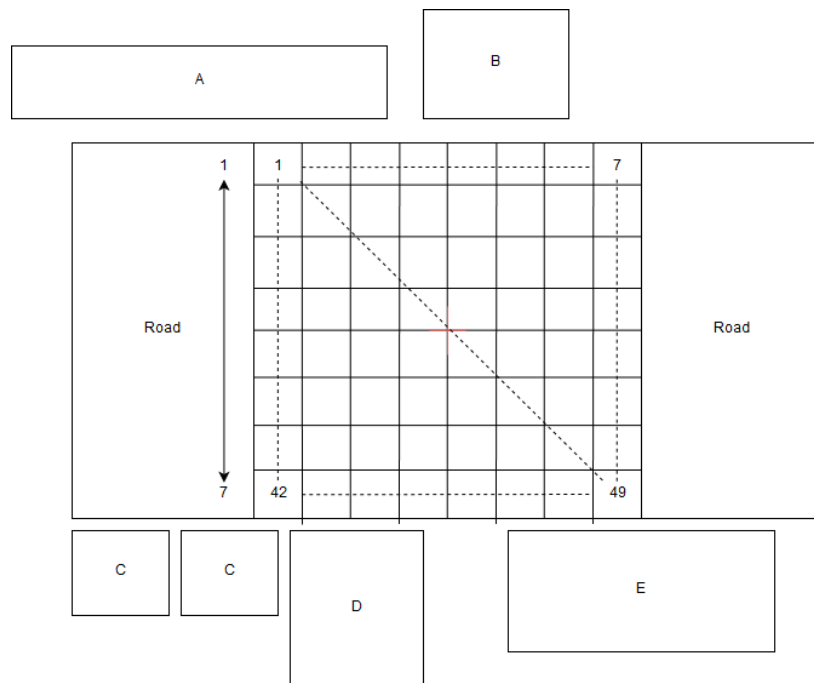


Figure 14. Grid around the receiver point.

As Figure 14 suggests, one small square of the grid represents a candidate next to the receiver. Each candidate is separated by one metre. Then, rays are defined between the position of each candidate and GNSS. An intersection point is calculated for each point of the grid, satellite and building surrounding the PIKSI Multi.

Because the receiver collects the Carrier-to-Noise-Density ratio (dB/Hz) of each satellite, it can be used to define a coefficient κ depending on the visibility of satellites.

Research conducted in [3] and [4] found that the C/N_0 (dB/Hz) level gives an indication on the quality of a signal and visibility of a GNSS. However, the C/N_0 (dB/Hz) level should not be taken into consideration alone without incorporating another factor.

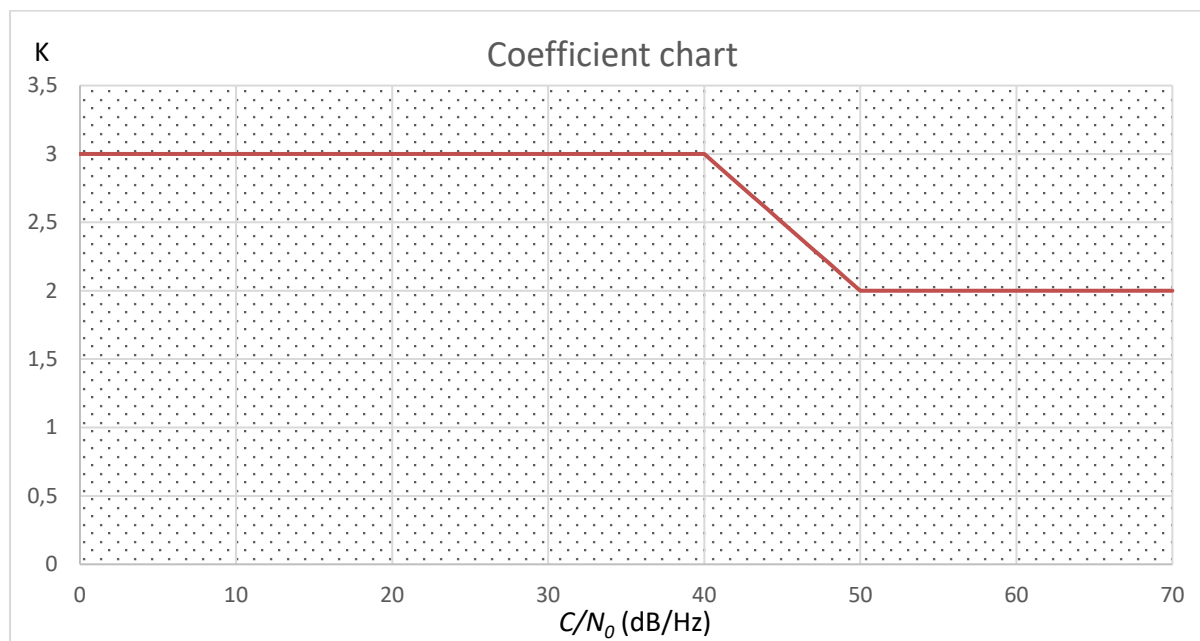


Figure 15. Coefficient chart.

As shown on the chart of Figure 15, coefficient κ is defined as:

$\kappa = 3$, if $C/N_0 \text{ (dB/Hz)} \leq 40 \text{ dB/Hz}$. It means a satellite is hidden.

$\kappa = a * C/N_0 \text{ (dB/Hz)} + b$, with a as the slope ratio and b as the y-intercept

Happens when $40 \text{ dB/Hz} < C/N_0 \text{ (dB/Hz)} < 50 \text{ dB/Hz}$. It means the visibility factor is ambiguous.

$\kappa = 2$, if $C/N_0 \text{ (dB/Hz)} \geq 50 \text{ dB/Hz}$. It means a satellite is visible

Depending on the status and $C/N_0 \text{ (dB/Hz)}$ levels of a GNSS, a score is attributed to each candidate. Table 2 indicates how a *score* is calculated.

Table 2. Scoring pattern.

| Scoring | $C/N_0 < 40 \text{ dB/Hz}$ | $C/N_0 > 40 \text{ dB/Hz}$ |
|---------|----------------------------|----------------------------|
| NLOS | κ | $-\kappa$ |
| LOS | $-\kappa$ | κ |

Once, scores computation is completed, each candidate is compared with each other. First three higher scores are revealed. They correspond to the best three positions.

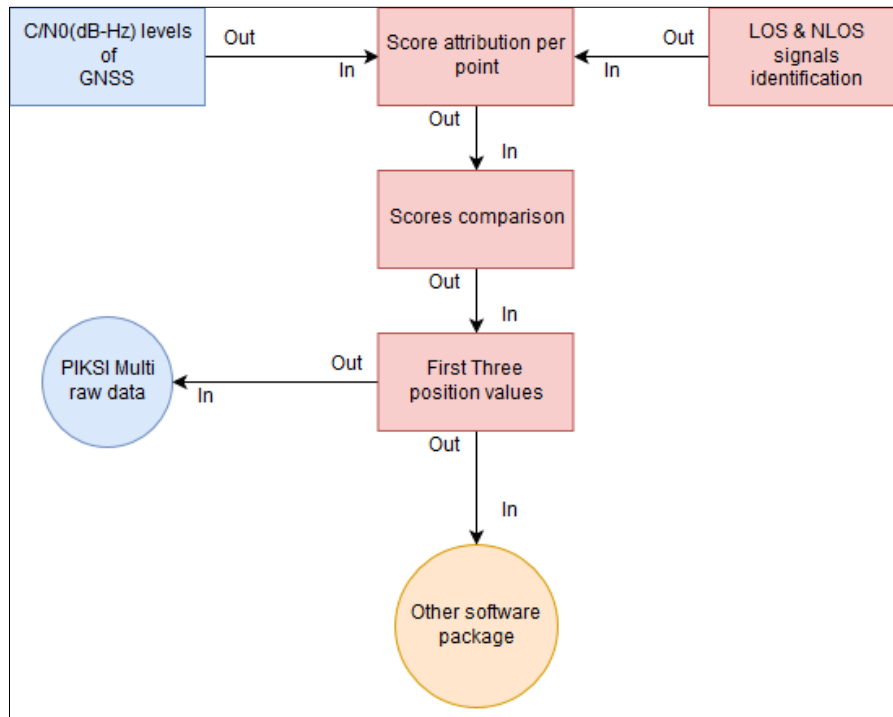


Figure 16. White box view of the shadow matching.

Ultimately, these candidates are returned by the shadow matching block and are either input into the next choice of candidates or inserted into the next stage of the software package as Figure 16 shows.

2.3 Algorithm Testing Design

Henceforth, the reader should have a better understanding and overview of the running of the algorithm. However, having a simple overview of its processing is not enough. Programs and algorithms must be tested and work properly. That is the reason why a brief testing approach is covered. The following paragraphs summarize how tests are designed and what is expected from them in practice.

Swift Navigation Binary Protocol Testing

The Swift Navigation Binary Protocol, or *SBP*, is a proprietary protocol from the company: Swift Navigation. Its understanding is available on [15]. Furthermore, a multilingual library, *spblib*, is developed and deployed on GitHub by the company. This library allows users to use the *SBP* directly.

Even if the library is available for the Python programming language, bespoke functions are developed to keep those that are the most suitable for the purpose of the project.

Validity of the data of the program is handled by a test, which compares raw data of the *Swift Console* and protocol reading. It should allow distinguishing difference between results.

If raw data are different, it means the protocol structure is not respected. In the other case, the test ensures that the program's function in charge of the data reading follows the *SBP* structure and sends the right requests to the PIKSI Multi.

GNSS Position Checking

Simulation of raw data is not mandatory because they come directly from the PIKSI Multi. However, an online tool is used to establish whether calculations executed during the receiver stage coincide with the reality. The positions, azimuths and elevations of GNSS from the functions of the code are compared with the outputs of the *in-the-sky.org* website [22]. It is a website developed, by Ford.D [22] that tracks GNSS moving around Earth. The comparison of results from computations and [22] must be similar or more accurate to ensure computations work well.

Coordinate System Conversions

Although it is simple to test satellites data, it is more complicated to validate that the positions are expressed into the good coordinate system. Data from the three-dimensional map follow the *LB-93* coordinate system and no direct conversion from *LB-93* to the *ECEF* system exists. Furthermore, raw data and calculations from the receiver give values expressed in, either *WGS84* or *ECEF*.

To avoid conversion problems during mathematical operations, it has been established that the data from the three-dimensional map have to be expressed in *WGS84* or *ECEF*. It means data go from the *LB-93* to *WGS84* and if it is necessary, from the *WGS84* to *ECEF*.

To ensure conversions compute well, two online tools are used. The first tool is the French government mapping website, *geoportail.gouv.fr*, [23]. Results from both sources are compared. It should validate that the *LB-93* to *WGS84* conversion has been properly computed. In addition, the *oc.nps.edu* [24] website is used to ensure the second conversion: from *WGS84* to *ECEF* has been well operated.

The test to valid the position of the PIKSI Multi consists of placing a specific point on an online map and converting the position from the *LB-93* to the *WGS84* standards. Then, it is possible to know whether expected values are equivalent to the reality.

LOS and NLOS Signals Identification

As far as LOS and NLOS identification testing is concerned, several PIKSI Multi positions should be simulated on the road close to the buildings appearing on the three-dimensional map. Because the positions of the GNSS can be inspected and compared with the *in-the-sky.org* [22], it is possible to track and assess the receiver-GNSS rays.

Thus, a predefined square that contains expected buildings is drawn, so the coordinates of the buildings located inside the square are extracted from the database. Then, the planes and surfaces are created. Ultimately, the intersection crossing process is run.

Signals status as LOS and NLOS can be expected. Furthermore, positions are defined and simulated upstream. It means, verifying the authenticity of results should be straightforward.

Write to the Swift Navigation Binary Protocol

To mask satellites, the program writes in a relevant register through a specific message of the *SBP*. However, it does not provide any flags when an action affects the receiver. Therefore, message should have a ‘visible’ effect. From the protocol detailed in [15], the PIKSI Multi is restarted via the message type $(0x00B6)_{16} = (182)_{10}$.

A restart proves that the message has been considered by the *SBP*. It proves equally that the sentence, which has been sent, followed the *SBP* structures. Therefore, any message can be sent if it follows this structure.

Determining the Best Position from the Shadow Matching Score

Test to check the veracity of the score computation is more complicated than pre-established tests. The score is strongly related to the coefficient κ . The coefficient must be well defined at the beginning of the design.

If the coefficient is not balanced enough between the status of satellites and their C/N_0 (dB/Hz) level, some problems can occur as regards the identification of the best position on the road. Supposing the visibility of GNSS is the most important factor in the identification of LOS and NLOS, it is designed that the coefficient κ has to take the level of C/N_0 (dB/Hz) of each satellite less into account than their visibility.

Once κ is defined, the test of the shadow matching score can take place. Always using a potential location and the square of buildings located in the three-dimensional area, candidates are created.

A score is assigned to each candidate. Due to the simulated aspect of the situation, the scores and best position outputs must be already known. On the one hand, if the program sends back expected positions, tests are deemed to have passed. On the other hand, if results of tests are different from expected solutions, they are considered to have failed.

The test definition is complete. Each part of the software component must pass all the designed tests. Otherwise, the code must be re-examined, redeveloped and retested in order to be one hundred percent safe.

Chapter 3

3. IMPLEMENTATION STAGE

3.1 Development of the Algorithm under Python

In this section a more detailed aspect of the algorithm outlined above is discussed.

Data from the PIKSI Multi

As Figure 17 illustrates, a Swift Navigation Binary Protocol frame is structured according to eight plus N bytes, where N is equal to the number of bytes needed to store a payload.

| Preamble | Message Type | Sender | Length | Payload | CRC16 CCITT |
|---------------|--------------|---------|--------|-----------|-------------|
| 1 byte | 2 bytes | 2 bytes | 1 byte | N bytes | 2 bytes |
| 8 + N bytes | | | | | |

Figure 17. SBP frame [15]

A frame is composed of three categories that can be subdivided.

- A header is fragmented in four subsections encoded on six bytes:
 - One byte is set to determine the preamble term.
 - Two bytes are used to identify the message type of a frame.
 - Two bytes are address to the identification number of the sender of the message.
 - One byte that indicates the length of the payload.
- The content of the payload is written using N bytes.
- The standard used as the checksum of a frame is the CRC16-CCITT. It is coded with two bytes.

The content of each section of the header is as follows:

- Header

The **preamble** value indicates if a message is received from the *SBP*. In accordance with the protocol [15], its value must always be equal to $(0x55)_{16} = (85)_{10}$.

To define which content is sent through a frame, a **message type** is transmitted to the receiver of the message. Based on the documentation [15], raw data have their own values. For example, the value $(0x020A)_{16} = (522)_{10}$ indicates that the **MSG_POS_LLH** is sent from the transmitter to the receiver of the message.

The **sender** value identifies the transmitter of the message. The identifier of the PIKSI Multi is preconfigured with the value $(0x7F5C)_{16} = (32,604)_{10}$ by Swift

Navigation in the private settings of the PIKSI Multi. However, hardware users can change the receiver identification number in the settings section of *System Info* or *Swift Console*, e.g. the value defining the computer of the project is $(0x4200)_{16} = (16,896)_{10}$.

The length corresponds to the N bytes of the payload. Its value varies depending on the message type and amount of data that must be transmitted through the protocol.

- Payload

It includes data of the message that must be sent. It is defined by the **message type** compiled in the *SBP* documentation [15]. Furthermore, the number of bytes corresponding to the payload can change greatly. Depending on the number of satellites tracked, S , it varies from zero to $17*S + 11$. For instance, the PIKSI Multi can receive thirty-two signals from GPS, the payload bytes range is extended until 555 bytes.

- CRC16-CCITT or Cyclic Redundancy checksum

It is calculated from the beginning of the message type to the end of the payload. The CRC16-CCITT checks the integrity of the message sent through the protocol.

It is a content validator and is calculated to accredit the transmission of a frame. It is transmitted from the sender Tx to the receiver Rx of a message. A comparison of CRC16-CCITT values is carried out to either accept or reject the transmission of a message.

This checksum follows the principle explained in [17]. However, multiple methods of checksum computation exist and the CRC16-CCITT is one of them. Each checksum responds to different rules of computation.

Once all the message types are identified, the payloads must be decrypted and parsed to gather expected raw data. Currently, two hexadecimal formats exist, the big and little endian. The *SBP* follows the second structure. Messages that are formatted with this standard are read by pairs of bytes from the tail to the head, i.e. from the right to the left, as Figure 18 shows.

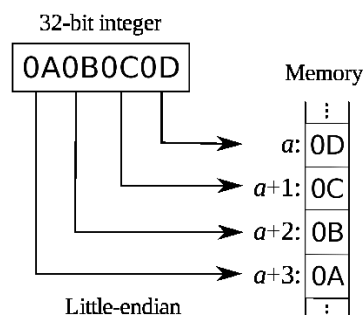


Figure 18. Little endian format.

Raw data from frames are extracted and are decrypted using the Python's libraries: *serial* and *struct*. Therefore, some mathematical and orbital calculations must be carried out to obtain intermediate usable data.

Raw Data Extraction

First, the PIKSI Multi receives the identification number (ID) of each satellite.

A GNSS sends back a quantity of data to the receiver located on Earth. GNSS information is public and enables interested parties to calculate the position of each satellite. In addition, the satellites send both navigation and observations messages to a receiver.

In order to compute the position of the GNSS, users can use the data from the almanacs or ephemerides. For instance, the position estimation of a GNSS is more or less precise depending on the data choice made between an almanac and an ephemeris.

The data from an ephemeris are valid for a period of four hours, whereas the data from an almanac are valid for ninety days. The last one are often stored in the GNSS receivers to speed up the first processes. Due to the message structures and transmission time of a GPS, it takes thirty seconds to receive the complete data from an ephemeris, and twelve and half minutes to receive those from an almanac.

Although an almanac is valid longer than an ephemeris, the last one provides much more orbital parameters. The ephemeris data are also more precise than the almanac data.

The almanacs and ephemerides data are obtainable via the *SBP*. However, the algorithm is designed to retrieve only orbital parameters from ephemerides.

The PIKSI Multi is able to receive messages from multiple GNSS, such as GPS (American), GLONASS (Russian), Galileo (European) and BeiDou Navigation Satellite System (BDS, Chinese). It is nonetheless designed initially to gather GPS and GLONASS ephemerides.

Parameters from GPS Ephemeris

- Time of Ephemeris, T_{oe} in seconds.

This parameter allows the user to access to the GPS week number. A GPS week number is not the same as the Julian calendar week. It is taken from a modified calendar, which began on the 6th of January 1980 at midnight.

- Harmonic corrections terms in metres.

The GPS radio waves are delayed when travelling through an atmosphere. The delay varies with the atmospheric conditions. Therefore, a GPS ephemeris provides six harmonic corrections terms, see Table 3. They are applied to the positioning calculation.

| | Symbols | Amplitude of sine and cosine correction terms |
|----------------------|------------------|---|
| Harmonic Corrections | C_{rs}, C_{rc} | To the orbit radius |
| | C_{uc}, C_{us} | To the angle of inclination |
| | C_{ic}, C_{is} | To the Argument of latitude |

Table 3. Harmonic correction terms

Signal perturbations must be corrected to calculate a spatial position of a GPS.

- Mean motion difference, Δn in semicircles per second.
It is the angular speed of a GPS to complete one orbit
- Mean anomaly at reference time, M_0 in semicircles.
It is the angular distance from the pericentre of an imaginary body moving around a circular orbit. The imaginary body moves at a constant speed, with the same period of revolution.
- Eccentricity of a satellite, Ecc without units.
It determines the amount of the orbit deviation from one object turning around another.
- Square root of the semi-major axis of an orbit, \sqrt{A} in metre $^{1/2}$.
- Longitude of ascending node of orbit plane at a weekly epoch, Ω_0 in semicircles.
It is the angle from a reference direction to the direction of the ascending node for a specific plane.
- Rate of right ascension, $\dot{\Omega}$ in semicircles per second.
It is the rate of change for the right ascension. The right ascension is the ‘angular distance measured only *eastward* along the celestial equator from the Sun at the March equinox to the (hour circle of the) point above the earth in question’ - Wikipedia.
- Argument of perigee or periapsis, ω in semicircles.
It is the angle from the ascending to its periapsis node of a GPS.

- Inclination angle of the GPS, i_0 in semicircles.
- Rate of inclination angle, i in semicircles per second.
As $\dot{\Omega}$, this rate represents an inclination angle change.

All these parameters are related to a GPS orbiting a specific star or planet, here, Earth.

Parameters from GLONASS Ephemeris

Some ephemeris information of GPS and GLONASS is the same, e.g. the T_{oe} or week number. However, GPS and GLONASS ephemerides do not totally follow the same structures.

Directly after the *header* of a GLONASS ephemeris, followed parameters are mentioned

- Relative deviation of predicted carrier frequency from nominal, Γ (dimensionless)
- Correction to the reference time, τ (seconds).
- Equipment delay between L1 and L2 GLONASS bands, $\Delta\tau$ (seconds).

Furthermore, a GLONASS ephemeris gives a direct access to the positions, velocities and accelerations of satellites. All these data are expressed in the *PZ-90.02* reference. It is an equivalent of the *ECEF* coordinate system.

- Pos array, as a double[3], which contain pos_x, pos_y and pos_z in metres.
- Vel array, as a double[3], which contain vel_x, vel_y and vel_z in metres per seconds.
- Acc array, as a double[3], which contain acc_x, acc_y and acc_z in metres per seconds square.

Raw data from a GLONASS ephemeris give a direct access to the position of a GLONASS that is orbiting around Earth. Therefore, no computation is necessary to obtain its position using the good standard of coordinates.

Position of the PIKSI Multi

Moreover, the PIKSI Multi sends back its coordinates in the *WGS84* system.

- Longitude is the angle, restricted between $[-180^\circ, 180^\circ]$, which specifies the position of a point in relation to the East. Longitude is expressed in degrees.
- Latitude is the angle, restricted between $[0^\circ, 90^\circ]$, which indicates the position of a point in relation to the North. It is measured in degrees
- Height or altitude is the position of the receiver above the *WGS84* ellipsoid, expressed in metres.

Universal Time Coordinates or UTC

This is the main time standard. It is based on one second of the mean solar time at 0° longitude. Interestingly, Greenwich Mean Time or GMT is a synonym, for English speakers, of UTC. The

UTC time is used to compute the positioning calculation of all the satellites surrounding the PIKSI Multi.

Observation Data

Data from an observation are also gathered through the protocol. An observation data are composed of the pseudo-range and Carrier-to-Noise-Density ratio.

The pseudo-range is the pseudo-distance, in metres, between a GNSS and a receiver. To determine its position, a receiver calculates its distance from a constellation of satellites at a transmitting time. It is possible to determine the position of an object on Earth because of the triangulation principle.

Triangulation Principle

Four satellites compose a constellation. Each satellite owns its atomic clock that has to be synchronized. All the satellites perform the same operation to calculate the time necessary for a signal to reach an Earth's receiver at a t instant, which is $T = t - t_{GPS}$.

Using the relation $d = T * C$, where C is the speed of light in a vacuum, each satellite defines a surrounding sphere.

By crossing surrounding spheres of satellites, it is possible to determine the position of the receiver.

The Carrier-to-Noise-Density ratio, C/N_0 (decibels/Hz), is the signal-to-noise ratio of a modulated signal when the only considered source of noise is the receiver. The unit of C/N_0 is the decibel per Hertz (dB/Hz) because it is a noise related to a modulated signal.

Raw data from the PIKSI Multi are collected through a serial communication using the Python's library *serial*. The communication is enabled on a specific input port and baud rate.

```
serial.Serial("COM4",baudrate=115200,bytesize=serial.EIGHTBIT,timeou  
t=10,xonxoff=False)
```

Mathematics is handled to calculate the positions of the GNSS from the raw data of ephemerides. Algebraic, geometric operations and equations are mainly used to determine the positions of the receiver and satellites.

Computed Data

Calculation of X, Y and Z-coordinates of GPS into the *ECEF* reference must be completed to determine the position of the PIKSI Multi in the same reference. *ECEF* means **E**arth-**C**entered, **E**arth **F**ixed, it is a coordinate system based on the shell of Earth. Thus $O_{ECEF} = (0, 0, 0)$.

Calculations that concern a satellite's position have been operated a certain number of times in the past. The equations that follow are well established and reliable.

Following the equations and instructions from page 99 to page 101 of 'Navstar GPS Space Segment/Navigation User Interfaces' [21], it is not too complex to compute operations.

GPS Position Computation

First, Earth's gravitation (μ) and rotation rate (Ω_e) constants are expressed in Equation 3. They are defined to calculate the first orbital parameters demonstrated in Equation 4.

Equation 3. Definition of Earth's constants

$$\begin{aligned}\mu &= 3.986005 * 10^{14} \text{ meters}^3.\text{sec}^2 \\ \Omega_e &= 7.2921151467 \times 10^{-5} \text{ rad/sec}\end{aligned}$$

Equation 4. Calculation of basic GPS parameters from [21]

$$\begin{aligned}A &= \left(\sqrt{A} \right)^2 \\ n_0 &= \sqrt{\frac{\mu}{A^3}} \\ t_k &= t - t_{oe} * \\ n &= n_0 + \Delta n\end{aligned}$$

The computed mean motion n_0 , in radians per seconds, represents the angular speed of a GPS to complete one orbit. Therefore, the time in seconds from an ephemeris reference epoch T_k that defines the time reference, must be computed. T_k is equal to the difference between the current UTC time and the T_{oe} .

In addition, the value of T_k depends on the time of the week of the calculation. A week is divided into halves. One from the beginning to the half, from Sunday at midnight to Wednesday at noon, of a week. Another from the half to the end, from Wednesday at noon to Saturday at midnight, of a week.

If T_k is calculated on the first half of a week, the result can be greater than 302,400 seconds. If it is the case, a subtraction of 604,800 seconds must be applied.

If T_k is computed on the second half, T_k can be less important than - 302,400 seconds. If it happens, an addition of 604,800 seconds must be made.

Because the reality cannot be as perfect as the theory, some corrections are required. One correction concerns the mean motion and it involves the addition of the computed mean motion and the mean motion difference.

The most difficult part of the GPS positioning calculation steps come from Equation 5 operations:

Equation 5. True anomaly calculation from [21]

$$\begin{aligned}
 M_k &= M_0 + nt_k \\
 M_k &= E_k - e \sin E_k \\
 v_k &= \tan^{-1} \left\{ \frac{\sin v_k}{\cos v_k} \right\} \\
 &= \tan^{-1} \left\{ \frac{\sqrt{1-e^2} \sin E_k / (1 - e \cos E_k)}{(\cos E_k - e) / (1 - e \cos E_k)} \right\}
 \end{aligned}$$

To determine the position of a body moving along a Keplerian orbit, an angular parameter must be calculated. It involves the true anomaly V_k .

However, the V_k operation uses a non-linear parameter, the eccentric anomaly E_k , which must be calculated first. Moreover, E_k must be computed in relation to a mean anomaly, M_k .

The M_k value is relatively simple to calculate, it is the addition of the mean anomaly at the reference time and the multiplication of the corrected mean motion at the time from an ephemeris reference epoch.

From the M_k result, the E_k calculation can be carried out using multiple methods. As is it demonstrated in Equation 5 and from the Navstar GPS Space Segment/Navigation User Interfaces, the eccentric anomaly can be solved using an iterative method. Moreover, it is possible to use the true anomaly V_k to compute the eccentric anomaly E_k , as shown in Equation 6.

Equation 6. Eccentric anomaly calculation from the true anomaly from [21]

$$E_k = \cos^{-1} \left\{ \frac{e + \cos v_k}{1 + e \cos v_k} \right\}$$

The iterative method is chosen for the project because it is the most suitable and documented method to determine E_k in relation to the operations carried out beforehand.

A maximal number of one hundred iterations is imposed at the beginning of the E_k computation. If the process cannot be completed before this limit, the iterative method is curtailed and an error is returned.

Secondly, E_0 , the first E_k value, a ΔE_k , with a tolerance equal $1 * 10^{14}$ and a counter equal to zero are initialized:

Equation 7. Eccentric anomaly definition

$$E0 = Mk + \left(-\frac{1}{2} * ecc^3 + ecc + \left(ecc^2 + \frac{3}{2} * \cos(Mk) * ecc^3 \right) * \cos(Mk) \right) * \sin(Mk)$$

$$\Delta E_k = \text{tolerance} + 1$$

The following computation will occur while the ΔE_k value is lower than the tolerance.

Equation 8. E_k iterative calculation [21].

$$\begin{aligned}
 t1 &= \cos(E0) & t2 &= -1 + ecc * t1 & t3 &= \sin(E0) & t4 &= ecc * t3 \\
 t5 &= -E0 + t4 + \text{norm}(Mk, 2p) & t6 &= \frac{t5}{\frac{1}{2} * t5 * \frac{t4}{t2} + t2} \\
 E_k &= E0 - \frac{t5}{\left(\frac{1}{2} * t3 - \frac{1}{6} * t1 * t6 \right) * ecc * t6 + t2} \\
 \Delta E_k &= \text{abs}(E - E0) & E0 &= E_k & \text{counter} &= \text{counter} + 1
 \end{aligned}$$

Operations used to compute E_k are included in Equation 8. These operations represent one iteration of the process. The final E_k value is obtained after N -iterations of Equation 8 and if one hundred iterations are not reached. The true anomaly term can now be calculated.

Consider the atmospheric perturbations is a good process to obtain the corresponding position of a GPS. A new parameter must be foremost calculated. It is the argument of latitude: Φ_k , expressed in Equation 9.

Φ_k is the result of the addition between the true anomaly and the argument of perigee. It is similar to a latitude of Earth but this argument of latitude is applied to the objects belonging to a Keplerian orbit.

Equation 9. Argument of latitude from [21].

$$\Phi_k = v_k + \omega$$

That new argument of latitude is used in the second harmonic perturbations computations, shown in Equation 10.

Equation 10. Second Harmonic Perturbations from [21].

| | |
|--|---------------------------------|
| $\delta u_k = c_{us} \sin 2\Phi_k + c_{uc} \cos 2\Phi_k$ | Argument of Latitude Correction |
| $\delta r_k = c_{rs} \sin 2\Phi_k + c_{rc} \cos 2\Phi_k$ | Radius Correction |
| $\delta i_k = c_{is} \sin 2\Phi_k + c_{ic} \cos 2\Phi_k$ | Inclination Correction |

Harmonic correction terms are multiplied by the square of cosine and sine of the new argument of latitude.

The harmonic perturbations should be taken into consideration because they can affect more or less the real argument of latitude, radius and inclination. In addition, to obtain the most accurate estimation of a position, all possible physical perturbations should be applied and included into further equations.

Once the second harmonic perturbations are calculated, they can be applied to initial parameters: argument of latitude, radius and inclination, as Equation 11 mentions.

Equation 11. Corrected parameters calculation from [21].

| | |
|--|--------------------------------|
| $u_k = \Phi_k + \delta u_k$ | Corrected Argument of Latitude |
| $r_k = A(1 - e \cos E_k) + \delta r_k$ | Corrected Radius |
| $i_k = i_0 + \delta i_k + (IDOT) t_k$ | Corrected Inclination |

Nearing the end of the process, it is important to carry out a projection into an orbital plane in order to convert the position of celestial bodies into the coordinate system of Earth.

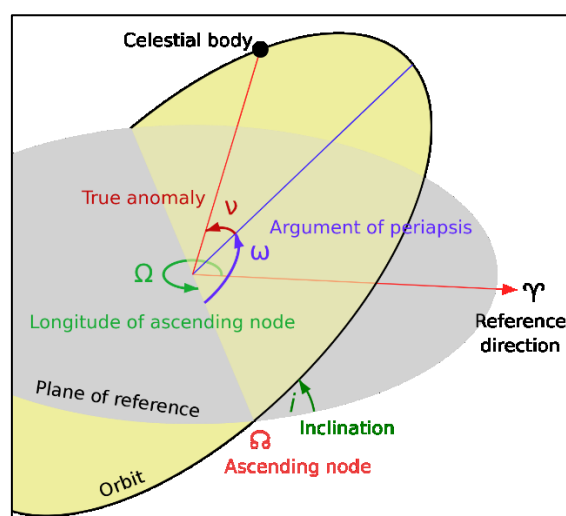


Figure 19. Planes projection.

The two first coordinates X'_k and Y'_k are the result of multiplying the corrected radius R_k by the cosine and sine of the corrected argument of Latitude U_k .

This projection is illustrated in Figure 19 and using Equation 12. However, coordinates are not projected into the *ECEF* system yet.

Equation 12. Orbital plane projection from [21].

$$\begin{aligned} x'_k &= r_k \cos U_k \\ y'_k &= r_k \sin U_k \end{aligned}$$

Next, the corrected longitude of the ascending node, Ω_k , has to be calculated. It helps to specify the origin of the orbit of an object placed in space. Ω_k is the ascending node, an angle from a reference direction to an ascending node direction. As Equation 13 demonstrates.

Equation 13. Corrected longitude ascending node from [21].

$$\Omega_k = \Omega_0 + (\dot{\Omega} - \dot{\Omega}_e) t_k - \dot{\Omega}_e t_{oe}$$

Finally, the positioning calculation of each GPS is solved for the *ECEF* reference, using the operations from Equation 14.

Equation 14. Corrected longitude ascending node from [21].

$$\begin{aligned} x_k &= x'_k \cos \Omega_k - y'_k \sin \Omega_k \\ y_k &= x'_k \sin \Omega_k + y'_k \cos \Omega_k \\ z_k &= y'_k \sin i_k \end{aligned}$$

If it is necessary, coordinates of GPS are available in *WGS84* and *ECEF*. Using the library of Python *pyproj*. It allows performing a reference conversion.

Azimuth and Elevation Angles Computation

These angles allow knowing how and where the GNSS are positioned in relation to the receiver's position.

An azimuth is an angle placed on the horizontal axis, respecting an interval between 0° and 360° . The elevation angle is located on the vertical axis and varies between 0° and 90° .

The azimuth angle is calculated with the help of the library *pyproj*. It contains an useful function that returns the forward, backward azimuth angles and the distance between two points. This function is named: *geod.inv* and it uses the latitudes and longitudes of both points as arguments.

However, in order to use this function, the desired coordinate system must be inputted. For instance, the *WGS84* is used with:

```
geod = pyproj.geod(ellps='WGS84')
```

Once a coordinate system has been defined, azimuth angles and distance can be collected with:

```
forward, backward, distance = geod.inv(longitude_A, latitude_A,
    longitude_B, latitude_B, radians = False)
```

The important returned value is *forward*. It represents the azimuth angle between a point A and a point B in decimal degrees. However, the *geod.inv* function returns a value between -180° and 180° . Because the azimuth is expressed between 0 and 360 degrees, 360 degrees has to be added when the result value is less than 0.

Concerning the elevation angle computation, no reliable library can provide a meaningful result. The following computation from [25] achieves to calculate the elevation angle.

Equation 15. Elevation angle equations [25].

$$\begin{aligned}
 A &= \begin{pmatrix} Xa \\ Ya \\ Za \end{pmatrix} & B &= \begin{pmatrix} Xb \\ Yb \\ Zb \end{pmatrix} \\
 coordinates &= \begin{pmatrix} Xa \\ Ya \\ Za \end{pmatrix} \cdot \begin{pmatrix} Xb \\ Yb \\ Zb \end{pmatrix} \\
 \cos(\gamma) &= \frac{coordinates}{mod(A) * mod(B)} & \gamma &= \arccos(\cos(\gamma)) \\
 \text{law of cosines : } d &= \sqrt{mod(A)^2 + mod(B)^2 - 2 * mod(A) * mod(B) * \cos(\gamma)} \\
 \sin(Elev) &= \frac{mod(B) \sin(\gamma)}{d} & Elev &= \arcsin(\sin(Elev))
 \end{aligned}$$

Calculation of the elevation angle is based on the trigonometry rules and law of cosines.

The elevation angle is used to know how is placed an object related to the position of another. A dot product, named *coordinates*, of both points gives a scalar value, i.e. the length of the projection of the first point onto the second one. Then, a division operation between *coordinates* and the multiplication of the norm functions of both points gives the cosine of γ .

The γ angle is related to the latitude of the point placed on Earth and the norm function attributes a physical size to a N -dimensions point.

Then, the law of cosines returns the distance d between both points. The sine of the elevation angle must be calculated. Using the division between the multiplication of the norm of the second point by the sine of γ , and the distance d . Finally, the inverse of the sine function: *asin*, results the expected elevation angle in degrees.

All geometrical data from the PIKSI Multi have now been processed. The next step is to extract, analyse and send the proper requests to the database to collect the three-dimensional data of the environment surrounding the PIKSI Multi.

Three-dimensional Map Data Extraction

The three-dimensional map gathers information of the fifth arrondissement of Paris, e.g. the roads, the road signs and the buildings. The table *bat_lo1* includes, for example, the buildings coordinates in the **LamBert 1993 (LB-93)** reference. It is the most important table to extract and analyse.

The *LB-93* was created in 1993. It is a conic map projection that is applied to the French Metropolitan territory. The continental drifts phenomenon incurs some localization errors when an international projection is realized on a local scale. In 1993, the French government decided to develop its own reference system to locate objects on its territory and avoid errors.

Table *bat_lo1* includes:

- The Identification number of a building is indicated with **building_ID**.
- Figure 20 illustrates the most important data. They are expressed with the type geometry of *PostGIS*.

The *SQL* column **building_floor** returns the coordinates of the buildings, which are defined as MULTIPOLYGON_ZM. They are composed of N points.

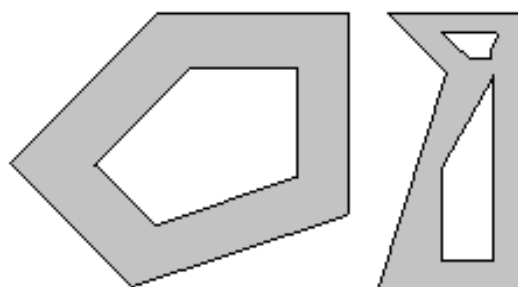


Figure 20. Multi Polygon example

- **z_min** is the z-coordinate of a building based on the floor.
- **h_max** is the maximal height of a building from the floor.
- **z_total** is the height of a building. It is expressed as the z-coordinate value from the floor to the roof of a building, i.e. $z_total = z_min + h_max$

PostgreSQL & PostGIS Requests.

To collect data for the project, *SQL* requests must be sent to the database. Command lines to communicate between the code and the database merge *PostgreSQL* and *PostGIS* requests.

Their goals are to collect the coordinates of the buildings located inside a two hundred-metre square around the PIKSI Multi. The size of the square is defined due to the capacity of the buildings to block a GNSS signal. Two hundred metres is an appropriate distance to collect the coordinates of the closest buildings without gathering those of the farthest buildings at the same time.

The identifiers of the buildings have nonetheless to be retrieved using SQL Request 2.

```
SELECT      building_ID      FROM      bat_lo1      WHERE
ST_Intersects(ST_Force_3D(ST_Transform(building_floor, 4326)),
ST_GeomFromText('POLYGON((%s %s %s, %s %s %s, %s %s %s, %s %s %s, %s
%s %s))', 4326));
```

SQL Request 2. Query to recover the identifiers of the buildings.

ST_Intersects(arguments) function of *PostGIS* takes one building and the square traced around the receiver as arguments. It returns **True** if a building is placed inside the square. However, a geometric conversion from *MULTIPOLYGON_ZM* to *MULTIPOLYGON* types is required because buildings were set as *MULTIPOLYGON_ZM*. The coordinates *M* does not correspond to a geometric measure, it is not necessary to keep it. The conversion is computed with the *ST_Force_3D*(arguments) *PostGIS* command.

Regarding the function *ST_Transform* (argument, Spatial Reference System or SRID number), it is carried out to gather and convert coordinates from the *LB-93* to *WGS84* coordinate systems.

The request *ST_GeomFromText* ('POLYGON ("coordinates of the point defining the polygon")', SRID number) declares the square surrounding the PIKSI Multi. A polygon composed of four points defines a square shape. The coordinates must be expressed into a specific reference depending on their *SRID* number.

Extraction of buildings coordinates in *WGS84* is carried out with the request illustrated in SQL Request 3 once ID of buildings have been collected by SQL Request 2.

```
Select  ST_AsText  (ST_Transform(bat_floor,4326))  as  buildings  FROM
database WHERE (ID = %s);
```

SQL Request 3. Extraction into the *WGS84* coordinates system.

However, SQL Request 3 is not enough to identify a signal. Distinguishing between LOS and NLOS requires using mathematical calculations.

The chosen method calculates mathematically an intersection point between a ray and a plane. Rays and planes must be created and crossed to manage the identification of a signal. The next mathematical operations return an intersection point if a ray crosses a plane. However, that potential crossing has to belong to the interval defining a building surface.

Mathematical operations

Planes Creations

Planes must be created in relation to the surfaces of the buildings. Surfaces used to create planes come from coordinates of the buildings that have been extracted.

Each surface is composed of four points:

- Points A and B defines the floor of a building.
- Points C and D defines the roof of a building.

A, B, C and D are expressed into the *ECEF* base and their X, Y and Z-coordinates will never be equal.

At least three points and a point of origin define a plane. For instance, A, B and C represent the point of a plane and D is set as the origin of a plane.

Following Equation 1, in section 2.2.2, parameters of a plane can be determined

$$P = ax + by + cz + d \text{ such as } a, b, c \text{ and } d \text{ are the plan's parameters}$$

To determine a, b and c values, the normal of a plane must be calculated. A *normal* in geometry is ‘an object such as a line or vector that is perpendicular to a given object’ - Wikipedia. The *normal* of a plane is noted as \vec{n} and is determined by the cross product of two vectors. However, the d value has not to be considered because it is related to vertical plane. The last parameter of a plane is equal to zero when the plane is vertical.

Therefore, both vectors are

$$\overrightarrow{AB} = \begin{pmatrix} Xb - Xa \\ Yb - Ya \\ Zb - Za \end{pmatrix}, \overrightarrow{AC} = \begin{pmatrix} Xc - Xa \\ Yc - Ya \\ Zc - Za \end{pmatrix}, \text{ the normal is } \vec{n} = \overrightarrow{AB} \wedge \overrightarrow{AC} = \begin{pmatrix} a \\ b \\ c \end{pmatrix} \text{ and } d = 0$$

Plane's coordinates $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ are expressed in relation to the coordinates of one point of the surface. For instance, coordinates of D $\begin{pmatrix} Xd \\ Yd \\ Zd \end{pmatrix}$ are chosen.

Ray Creation

A ray is drawn from an Earth's point to a satellite. Equation 16 defines the ray from a starting to an ending point.

Equation 16. Definition of a ray

$$\text{Ray}_{\text{position-GNSS}} = [P, G] = \left[\begin{pmatrix} X_{\text{position}} \\ Y_{\text{position}} \\ Z_{\text{position}} \end{pmatrix}, \begin{pmatrix} X_{\text{gnss}} \\ Y_{\text{gnss}} \\ Z_{\text{gnss}} \end{pmatrix} \right]$$

Once geometric objects have been declared, the intersection processing is carried out. To notice an intersection, a ray has to cross a plane.

Intersection Point Determination

If a ray crosses a plane, an intersection point is positioned between the beginning and the end of the ray. It must verify the correctness of the plane equation.

$$P = ax + by + cz = 0, \text{ with } \begin{pmatrix} x \\ y \\ z \end{pmatrix} \text{ as the coordinates of the intersection points, } I.$$

The intersection point I is calculated using one point belonging to the plane. Either A, B, C or D can be chosen. A, B and C points have already been used to define the plane and then D is selected.

Thus, Equation 17 compiles operations to calculate I .

Equation 17. Coordinates of the point of intersection.

$$I' = \frac{\vec{n} \cdot \overrightarrow{PD}}{\vec{n} \cdot \overrightarrow{PG}} = \frac{\vec{n} \cdot \begin{pmatrix} X_d - X_{\text{position}} \\ Y_d - Y_{\text{position}} \\ Z_d - Z_{\text{position}} \end{pmatrix}}{\vec{n} \cdot \begin{pmatrix} X_{\text{gnss}} - X_{\text{position}} \\ Y_{\text{gnss}} - Y_{\text{position}} \\ Z_{\text{gnss}} - Z_{\text{position}} \end{pmatrix}}$$

$$I = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \overrightarrow{GP} * I' + P = \begin{pmatrix} X_{\text{position}} - X_{\text{gnss}} \\ Y_{\text{position}} - Y_{\text{gnss}} \\ Z_{\text{position}} - Z_{\text{gnss}} \end{pmatrix} * I' + \begin{pmatrix} X_{\text{position}} \\ Y_{\text{position}} \\ Z_{\text{position}} \end{pmatrix}$$

The coordinates of I are injected into the plane's equation $ax + by + cz$ and three cases can happen

- (1) If the result is equal to zero, I crosses the plane P and I is returned.
- (2) If the result is not equal to zero, I does not cross the plane P and *None* is returned.

- (3) If the ray and the plane are parallel, the computation is curtailed, and *None* is returned.

When (1) happens, an intersection point is found. However, this point must be included within the interval describing the surface of a building.

It exists different methods to determine whether a point is belonging or not to a surface. One way is based on the question asks in [26]. It is answered by the user '*lab bhattacharjee*' of the *Mathematics Stack Exchange* website. The user's answer is summarized in Figure 21.

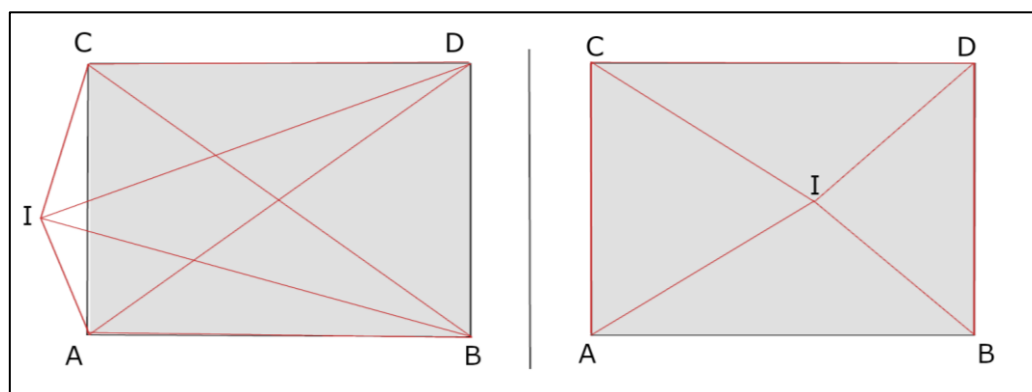


Figure 21. Intersection point inside a surface.

The method consists in calculating the sum of four triangle areas, e.g. AIB, BID, DIC and CIA. The sum of areas is then compared with the area of the ABCD rectangle. The comparison result leads to two possibilities.

- The sum result is greater than the area of ABCD, the intersection point *I* is therefore placed outside of the rectangle, i.e. the ray does not cross the building surface.
- The sum result is smaller than the area of ABCD, the intersection point *I* is located inside the rectangle and the ray crosses the building surface.

The surface of a building is represented as a rectangle. The area equation of a rectangle is simple as Equation 18 demonstrates.

Equation 18. Rectangle area

$$A_{ABCD} = width * length = |\vec{AC}| * |\vec{AB}|$$

$$= \sqrt{(Xc - Xa)^2 + (Yc - Ya)^2 + (Zc - Za)^2} * \sqrt{(Xb - Xa)^2 + (Yb - Ya)^2 + (Zb - Za)^2}$$

In contrast with the simplicity of the rectangle area calculation, the operation to compute a triangle area is more complex. It is possible to calculate a triangle area using the Heron's formula, as Equation 19 illustrates.

Equation 19. Heron's formula.

$$S = \sqrt{p(p-a)(p-b)(p-c)} \text{ With } p = \frac{a+b+c}{2}, a = AB, b = AC \text{ and } c = BC$$

An inaccurate result occurs nevertheless from the Heron's formula. It creates instability within the triangle area calculation if one edge of the triangle is relatively smaller than other edges.

William Kahan solved this issue at the end of the nineteenth century. He explains his method in [27]. The William Kahan's formula is summarized in Equation 20.

Equation 20. William Kahan's formula

$$S = \frac{1}{4} * \sqrt{[a + (b + c)] * [c - (a - b)] * [c + (a - b)] * [a + (b - c)]}$$

Where $a = AB$, $b = AC$ and $c = BC$.

To use the William Kahan's formula, triangle's edges must be labelled and sorted, such as $a > b > c$. The computation of the AID, DIC, CIB and IBA triangle areas follow the William Kahan's formula. Then, the sum of the four areas is operated:

$$S_{Triangles} = S1 + S2 + S3 + S4$$

Table 4 compiles and illustrates two possibilities:

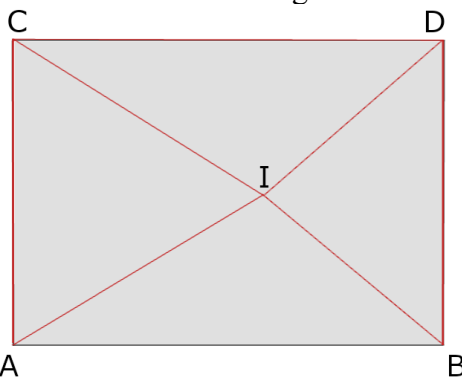
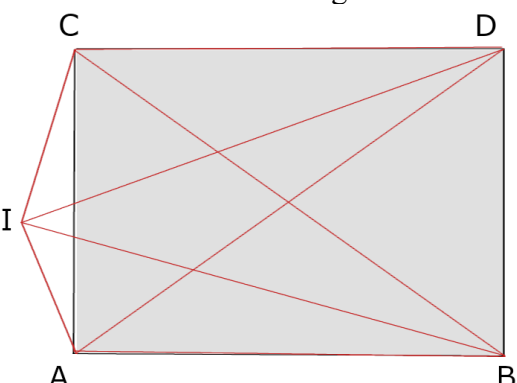
| $S_{Triangles} < A_{ABCD}$ | $S_{Triangles} > A_{ABCD}$ |
|---|--|
| Intersection point is placed inside the ABCD rectangle. | Intersection point is placed outside of the ABCD rectangle. |
|  |  |

Table 4. Localization of the intersection point.

These operations must be carried out whenever a ray crosses a surface of a building.

Finally, the visibility of a signal is defined such as visible or hidden by a building belonging to the square surrounding the PIKSI Multi.

Shadow matching score operations

As mentioned in section 2.2.2, a factor κ has to be established related to two parameters:

- The status of a signal, either LOS or NLOS.
- The C/N_0 (dB/Hz) level of a signal.

Moreover, a grid of forty-nine points is created to surround the PIKSI Multi. Potentially, one point of the grid is the best position to track for the next stages.

A factor κ is attributed to each satellite received by the receiver. A score follows rules defined in Table 2 of 2.2.2.

| Scoring | $C/N_0 < 40$ dB/Hz | $C/N_0 > 40$ dB/Hz |
|---------|--------------------|--------------------|
| NLOS | κ | $-\kappa$ |
| LOS | $-\kappa$ | κ |

Each point is submitted to Equation 21 afterwards:

Equation 21. Score computation for each point.

$$Score \text{ for a point} = \sum_{First \text{ GNSS ID}}^{Last \text{ GNSS ID}} \kappa$$

When all points own their scores, they are compared. The score comparison and sorting allows knowing which point is the best and next position to track.

Comparison of Score

A *for* loop is run forty-nine times to compare scores. The best three points are returned at the end of the loop according to the result of the comparison. The returned points are those which receive the most of direct signals. The first one is the next position to track.

Steps detailed above concern one run of the *while* loop behaviour. They are computed N -times until the PIKSI Multi is turned off or the program is stopped.

3.2 Code Testing

Orbital parameters of satellites help other car's devices to estimate its position. Therefore, any tests carried out must ensure that the *SBP* is properly decoded, the position of the satellites are accurately calculated, the shadow matching score is reliable, and the satellites are perfectly rejected.

Swift Navigation Binary Protocol Testing

It is preferable to use the *Swift Console* tool to test a frame decoding. Through a comparison of the *Swift Console* with the results of the code, the test must indicate if there are discrepancies.

As displayed in Figure 22, a comparison of the observation messages is undertaken.

| Code's outputs | | | Swift Console's outputs | |
|----------------|--------------|--|-------------------------|--------------|
| GPS ID | C/N0 (dB-Hz) | | PRN | C/N0 (dB-Hz) |
| 1 | 45,5 | | 1 (GPS L1CA) | 45.5 |
| 8 | 39,5 | | 3 (GPS L1CA) | 43.8 |
| 18 | 45,2 | | 8 (GPS L1CA) | 39.5 |
| 11 | 43 | | 11 (GPS L1C...) | 43.0 |
| 3 | 43,8 | | 14 (GPS L1C...) | 37.2 |
| 22 | 43,8 | | 18 (GPS L1C...) | 45.2 |
| 32 | 42,2 | | 22 (GPS L1C...) | 43.8 |
| | | | 27 (GPS L1C...) | 32.2 |
| | | | 32 (GPS L1C...) | 42.2 |

Figure 22. *SBP* and *Swift Console* data comparison.

The communication passes through the serial port and the receiver of the transmission receives the $C/N0$ (dB/Hz) levels.

Not only does this prove that the value of the CRC16-CCITT check is congruent on both sides of the *transmitter-receiver* node, but it also proves that a message content is similar. Therefore, the *SBP* is properly decoded and the test is validated.

GNSS Position Checking

Raw data are more or less accurate depending on the choice of their source. For instance, to determine the localization of a GNSS, either an almanac or an ephemeris can be used. An almanac provides nonetheless less orbital data than an ephemeris. So, the almanac data are not used for the purpose of the project. Nevertheless, they are adapted to estimate the position of a GNSS on a long period.

Tests to estimate the accuracy of the position of a satellite use the *in-the-sky.org* website. Figure 23 and Figure 24 display coordinates in the *ECEF* and *WGS84* references. Results

from Figure 23 come from the program whereas Figure 24 illustrates coordinates from the *in-the-sky.org* website.

```
Satellite 18 is placed at : X = 15726258070 Y = 336685648 Z = 20685355458
at 52,1806605223 N 12,0841101352 E
```

Figure 23. GNSS Latitude and Longitude from the algorithm.

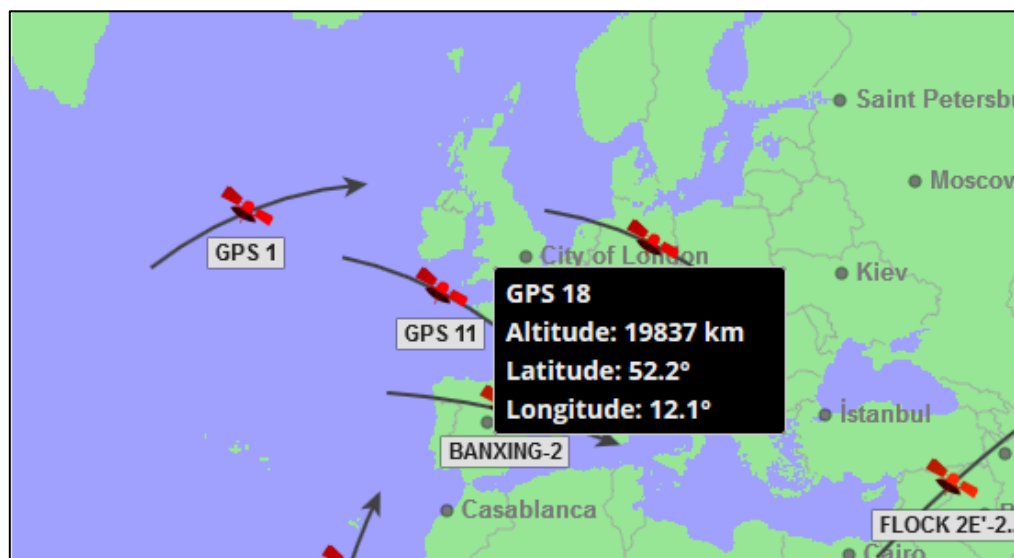


Figure 24. GNSS Latitude and Longitude from *in-the-sky.org*

Coordinates from Figure 23 are more accurate than coordinates from Figure 24.

A comparison demonstrates that the GPS coordinates are properly computed by the code and they are also more accurate than the online tool. Therefore, the test is validated.

It is easy to understand why there is a difference in precision. The number of digits taken into account by the online tool creates a gap of precision because it uses the almanac data and it must make some approximations.

Coordinate system conversions

Figure 25 shows the conversion from the *LB-93* to *ECEF*, through the *WGS84* references

```
Receiver is placed at 653217.473309 6860502.53247 in France in
LB-93 coordinates system
```

```
Receiver is positioned at: 48.8428710155 °N 2.36256882025 °E at
25.9 meters in WGS84
```

```
The ECEF conversion gives:
```

```
X = 42024049.17957 Y = 173368.106904 Z = 4779096.18122
```

Figure 25. Conversion of coordinate systems from the program.

As Figure 26 illustrates, the *oc.nps.edu* website displays the success of the conversion from the *WGS84* to *ECEF* systems.

| | | | | | | | | |
|--|-------------|-----------|-------------|--------|------|----------|--|----|
| Latitude | 48.84287101 | Longitude | 2.362568820 | Height | 25.9 | meters | | |
| ECEF - X | | km | ECEF - Y | | km | ECEF - Z | | km |
| LLH to ECEF | | | ECEF to LLH | | | | | |
| <p>ECEF from Latitude, Longitude, Height (ellipsoidal)</p> <p>X : 4202.049 km</p> <p>Y : 173.368 km</p> <p>Z : 4779.096 km</p> | | | | | | | | |

Figure 26. Coordinate systems conversion from *oc.nps.edu* website.

The comparison of results shows that the code converts better the coordinates than the website. This test meets the testing design requirements expressed in section 2.3.

The website '*oc.nps.edu*' operates some approximations to compute the conversions as does the website *in-the-sky.org*.

LOS and NLOS signals identification

To check if the status of a signal is well identified, the data from the database and the PIKSI Multi must be merged.

As Figure 27 roughly suggests, a point receives two signals. A building blocks one signal whereas the other signal is not. On the one hand, the point receives directly the first signal, i.e. there is no intersection between the *point-satellite* ray and the building. Thus, the signal is noticed as LOS. On the other hand, the building hides the second signal and it means there is an intersection on the path of the signal. This second signal is defined as NLOS.

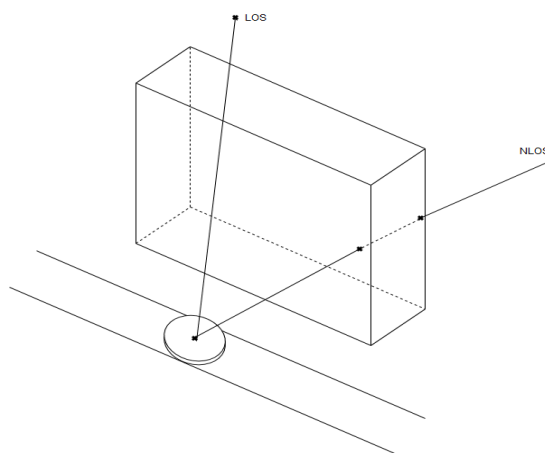


Figure 27. LOS and NLOS detection.

First, the buildings have to be extracted from the database depending on their position from the PIKSI Multi, i.e. if they are located inside the two hundred metres square around the receiver. Next, satellites that are surrounding the device, are detected and tracked. Finally, the piece of the program in charge of the intersection processing works well. Then, the PIKSI Multi stops to listen satellites which are not visible.

Write to the Swift Navigation Binary Protocol

The function to write to a PIKSI Multi register is easy to test because a *SBP* message follows the same header structure. Furthermore, the **MSG_RESET** register is used to check if the communication of data achieved.

The device restart is caused by **MSG_RESET**. The power LED turns off then turns on if the message is transmitted. This means it is possible to check the reliability of the contents of the header, payload and CRC16-CCITT.

When a message has to be written to the PIKSI Multi, the sender of a *SBP* message is the computer. The computer ID is configured as $(0x002A)_{16} = (42)_{10}$. However, data that must be encrypted and sent are:

- Message type: **MSG_RESET** = $(0x00B6)_{16} = (182)_{10}$
- Length of the payload: $(0x04)_{16} = (4)_{10}$.
- Payload content: $(0x00)_{16} = (0)_{10}$.
- CRC16-CCITT value that depends on the message content.

Once, all the fields of a frame are filled. The code line: `frame_without_preamble = struct.pack('<HHBL', MSG_RESET, computer ID, length, payload)` is run to encode the frame following the little endian standard and the [18] documentation.

The function `ser.write(argument)` encrypts a message which is sent from the computer to the PIKSI Multi through the serial communication. The PIKSI Multi has restarted. This proves that the message has been well encrypted and sent.

Determining the Best Position from the Shadow Matching Score

Shadow matching scores are attributed to each point belonging to the grid surrounding the signal receiver. It is simple to define a point as the best position according to the factor κ and the score comparison.


```
Score of the three best position is : 480.6 475.2 475.2  
List of GPS not visible set([22])  
Score of the three best position is : 948.6 943.2 943.2  
List of GPS not visible set([18, 22])  
Score of the three best position is : 1250.6 1241.2 1237.2  
List of GPS not visible set([17, 18, 22])
```

Figure 28. Three best scores and GPS not visible.

Figure 28 shows the masked satellites and the first three best scores.

A point which receives the most of LOS signals and the highest and most stable C/N_0 (dB/Hz) values has to have the highest score. The score comparison shows which score is the highest and consequently, which point is the best.

Figure 29 displays the best position related to the result of the score comparison.

```
Best candidate position is [4202046.495138522, 173364.49474502148, 4779098.481208794] in the ECEF coordinates system  
its latitude is : 48.842902441 ° N, its longitude is 2.36252104667 ° E, placed at 25.900000013 meters
```

Figure 29. Best candidate position

As a conclusion, the functions pass all the tests in *simulation* mode. The best point is determined and the wrong GNSS are rejected. Finally, reservations could be expressed on results of the tests which also have to be carried out in a real environment.

For instance, a car moves around the fifth arrondissement of Paris. Its locations and satellites positions are logged into a file throughout the driving time. The data collected are analysed to determine if the software component works and respects the structure and goals of the project.

Chapter 4

4. EVALUATION

The ‘Evaluation’ chapter describes and reconciles the results of the program and its tests. In order to keep clarity in the reading of this chapter, it is split into two parts.

The first part summarizes benchmarks for the responsiveness, portability, reliability, clarity and complexity of the code. The second part covers test results.

4.1 Code review

Responsiveness

Some parts of the code depend on the data sent by the PIKSI Multi. As this information is mandatory to complete some tasks, in the case that nothing is received, the code will simply not compute anything.

For instance, in order to compute the calculation of GNSS coordinates, either almanac or ephemeris data must be transmitted to the computing unit. Depending on the time it takes for the PIKSI Multi to receive and transmit data, the responsiveness of the program is affected. That is because it is not possible to set a period to send the *SBP* messages. Therefore, the program can possibly take a long time to receive the first required data.

Thus, no coordinate computation is carried out, in the same way that we wait for data from the PIKSI Multi before doing anything. Once all the required data are transmitted to the computing unit, it takes less than half a second to compute calculations and to obtain access to the first results.

Moreover, the *SBP*-conforming data are immediately decoded when they are transmitted. No deciphering errors can occur because of the definition of the *SBP* structure. A frame always follows the same pattern, such as header (preamble, message type, sender and length), payload and CRC16-CCITT. Finally, the received message cannot be erroneous.

Hence, the software component is fast when raw data have been transmitted to the unit in charge of the information processing.

Portability of the code

As long as the processing device is able to support an operating system, a Python compiler and external libraries, the code is executable and exportable anywhere.

The hardware targets to compute data must nevertheless dispose of an USB female port in order to communicate with the PIKSI Multi.

Reliability

Tests assess the reliability of the code. Extraction, decryption of frames and computational errors must not happen. To ensure that calculations and several parts of the code are executed and challenged against erroneous data multiple times.

Computation results are nonetheless strongly linked to coordinate systems. It is important for the reliability metric to take into account the choice of reference coordinates.

Clarity

To improve code clarity, the code follows the following guidelines:

Files and functions

Python doc is added to all files and functions with description of all inputs and outputs.

Variables and Python lines

Ambiguous names such as `a`, `b` or `c` are forbidden. Variables have explicit names such as `satellites_index` or `observations_from_GNSS`.

Magic Numbers

Magic number values are placed inside the *constants.py* file. It describes the meanings and origins of these numbers for the developers.

Complexity

Algorithm section

At the beginning of the project, a simple algorithm was described. The algorithm has to:

- Distinguish between two types of signals: LOS and NLOS.
- Associate a shadow matching score that depends on a factor κ to each point surrounding the receiver of signals.
- Sort scores and define the best score.
- Track the best point.

Mathematical section

Mathematical operations are not complex. They are based on basic mathematical and trigonometric operands such as the sums, subtractions, products, divisions, cosines, sines and tangents operations.

Matrix operations intervene to operate coordinates calculations and conversions or objects creation. These operations come from standard operands such as either addition, multiplication, dot or cross products of matrices.

Lack of Accuracy in Notation

Although program results are accurate, a major lack of precision comes from notation problems in the formulas, e.g. a $\cos^2(\alpha)$ is noted with $\cos 2\alpha$. Results of computations were sometimes wrong for that reason.

4.2 Tests Review

First, tests are carried out to cover predefined situations such as the urban canyon cases. For reference, see section 2.3.

The ‘Results of tests’ section summarizes tests executed in section 3.2. Furthermore, with the help of result analysis, it allows confirming or contradicting tests.

Section 3.2 proves that the code answers to the designed goals of the project.

Raw data, such as an ephemeris or the observations from satellites, are extracted thanks to the *SBP* frame decoding. Furthermore, computations of data cannot be carried out, if *SBP* messages are not well deciphered. The *Swift Console* software fulfils its role by ensuring that the extracted data fit.

However, raw data from GNSS are not the only challenged. Data from the three-dimensional map must be reliable and analysed in order to avoid misunderstandings on the next computations. Hopefully, the database is well designed and uses the suitable *SQL* extensions for geometric requests.

Once the raw data are collected in their entirety, coordinates computations and conversions can be done. Obtaining access to precomputed satellites coordinates is trivial because GNSS positions are public and coordinates equations are well known. Finally, it is easy to check if expected results are the right ones.

It is possible to determine whether a GNSS is LOS or not by computing whether a building is a blocking surface for a signal. The NLOS satellites can then be rejected from the GNSS listening.

The shadow matching algorithm aims to define all the potential next positions of the vehicle. Thus, the best candidate is used as the next position to track depending on the order of scores.

Through the Swift Navigation Binary Protocol, it was possible to stop the listening of hidden satellites. Thanks to the **MSG_RESET** message type of the *SBP*, the *writing on* function sends the right information via the serial communication.

As a conclusion of the result analysis, all of the program's components work properly. They send the next position to track, which is the major goal of the program.

No real-world urban canyon cases have been tested using the software component. Consequently, doubts can be raised about the code behaviour. Therefore, once the car embeds the software component, tests in real conditions must be undertaken in a location already compiled into the *SQL* database. Ultimately, the software component might be able to determine what the best position is and which satellite is not visible at any time.

Chapter 5

5. CONCLUSION

Several aspects of a project undertaken over six months have been presented and explained in this thesis - **GNSS multipath rejection based on environment knowledge**.

One section covers the hardware and software required to achieve the project goals. The hardware components have to be given serious consideration. The GNSS receiver, for example, needs to be able to receive and transmit the *SBP* frames. Receivers which match with *SBP* properties are accepted. However, if some adaptations are made to the code, another GNSS receiver can potentially be selected. Each individual must decide which software and tools are the most suitable for his or her project.

The algorithm has been overviewed and explained so that the reader may understand the algorithm's inner working. Furthermore, tests ensuring its truthfulness have been described and carried out. Tests are a significant side of any project and must be carefully considered.

A code review has been established to understand what was achieved during the project period. It includes raw data extraction, points and GNSS positioning calculations, objects creation, intersection computations, shadow matching and testing processes. It is important to take into account the fact that tests have been carried out in a *simulation* environment. Constraints applied to these tests are not the same as real-world constraints.

Finally, the code and its tests have been reviewed and discussed. The review shows that the code is perfectly readable, understandable and that it responds well when running in *simulation* mode.

The content of the thesis should be read as the program development logbook. It is recommended nonetheless to perform tests in a real urban canyon situation in order to ensure that the software component can be used in real-world conditions.

The multipath problem is one of the major issues of the GNSS technology. The physical properties of signals cause the multipath effect and prevent the use of GNSS in tight areas.

For years, developers and scientists have been trying to solve the effects of the multipath. It is a challenging problem, which is gradually being solved. However, the GNSS technology is not able to provide reliable data on its own yet.

For years, developers and scientists have been trying to solve the effects of the multipath. It is a challenging problem but solutions are gradually being identified. However, the GNSS is not able to provide reliable data on its own yet.

Future autonomous devices will not be able to rest on the GNSS technology. They will have to merge data from multiple sensors and choose the adequate information depending on their environment. For instance, GNSS data can be used in an open area but must be avoided when the device is approaching a city. In this kind of situation, autonomous vehicles must first trust LiDAR, cameras and IMU sensors.

Finally, GNSS technology cannot be trusted in a city environment because of the triangulation principle. A little improvement can be made by using a dynamic environmental map to constantly track surfaces around the signal receivers.

BIBLIOGRAPHY

- [1] Groves, P. D.(2008), Principles of GNSS, Inertial and Multisensor Integrated Navigation Systems, ArtechHouse, January 2008.
- [2] Peyraud, S., Bétaille, D., Renault, S., Ortiz, M., Mougél, F., Meizel, D. and Peyret, F. (2013). About Non-Line-Of-Sight Satellite Detection and Exclusion in a 3D Map-Aided Localization Algorithm. *Sensors*, 13(1), pp.829-847.
- [3] Groves, P. (2011). Shadow Matching: A New GNSS Positioning Technique for Urban Canyons. *Journal of Navigation*, 64(03), pp.417-430
- [4] Groves, P.D.; Jiang, Z.; Wang, L.; Ziebart, M. (2012) Intelligent Urban Positioning using Multi-Constellation GNSS with 3D Mapping and NLOS SignalDetection. In Proceedings of ION GNSS 2012, Nashville, TN, USA, 18–21 September 2012.
- [5] Closas, P., Fernandez-Prades, C. and Fernandez-Rubio, J. (2009). A Bayesian Approach to Multipath Mitigation in GNSS Receivers. *IEEE Journal of Selected Topics in Signal Processing*, 3(4), pp.695-706.
- [6] Paul D. Groves, Lei Wang, and Marek K. Ziebart. (2018). *Shadow Matching: Improved GNSS Accuracy in Urban Canyons*. [online] GPS World. Available at: <http://gpsworld.com/wirelesspersonal-navigationshadow-matching-12550/> [Accessed 24 Aug. 2018].
- [7] Viandier, N. (2018). *Modélisation et utilisation des erreurs de pseudodistances GNSS en environnement transport pour l'amélioration des performances de localisation*. [online] Tel.archives-ouvertes.fr. Available at: <https://tel.archives-ouvertes.fr/tel-00664264v1> [Accessed 24 Aug. 2018].
- [8] Safran. (2018). *Safran*. [online] Available at: <https://www.safran-group.com/> [Accessed 24 Aug. 2018].
- [9] Oodlestechnologies.com. (2018). *5 Different Automation Levels In a Self Driving Car*. [online] Available at: <https://www.oodlestechnologies.com/blogs/5-Different-Automation-Levels-In-a-Self-Driving-Car> [Accessed 24 Aug. 2018].
- [10] Swiftnav.com. (2018). *RTK, Piksi, High Precision GPS, GNSS, Centimetre-level relative positioning, Dual UART, USB, SBAS signals, Correlation*

Accelerator / SwiftNav. [online] Available at:
<https://www.swiftnav.com/piksi-multi>[Accessed 24 Aug. 2018].

- [11] Lighari, R., Berg, M., Kallankari, J., Parssinen, A. and Salonen, E. (2016). Analysis of the measured RHCP and LHCP GNSS signals in multipath environment. *2016 International Conference on Localization and GNSS (ICL-GNSS)*.
- [12] Swift Navigation. (2018). *Piksi Multi - RTK Position with Stationa....* [online] Available at:
<https://support.swiftnav.com/customer/en/portal/articles/2771177-piksi-multi---rtk-position>[Accessed 24 Aug. 2018].
- [13] Swift Navigation. (2018). *Piksi Multi Evaluation Board*. [online] Available at: <https://support.swiftnav.com/customer/en/portal/articles/2681333-piksi-multi-evaluation-board>[Accessed 24 Aug. 2018].
- [14] Swift Navigation. (2018). *Piksi Multi Specifications*. [online] Available at: <https://support.swiftnav.com/customer/en/portal/articles/2628580-piksi-multi-specifications>. [Accessed 24 Aug. 2018].
- [15] Swift Navigation. (2018). *Swift Binary Protocol*. [online] Available at: <https://support.swiftnav.com/customer/en/portal/articles/2492810-swift-binary-protocol>[Accessed 24 Aug. 2018].
- [16] Docs.swiftnav.com. (2018). *SBP - Swift Navigation Wiki*. [online] Available at: http://docs.swiftnav.com/wiki/SwiftNav_Binary_Protocol[Accessed 24 Aug. 2018].
- [17] Ross, N. (1993). *A PAINLESS GUIDE TO CRC ERROR DETECTION ALGORITHMS*. [online] Ross.net. Available at:
http://www.ross.net/crc/download/crc_v3.txt[Accessed 24 Aug. 2018].
- [18] Docs.python.org. (2018). *7.3. struct — Interpret strings as packed binary data — Python 2.7.15 documentation*. [online] Available at: <https://docs.python.org/2/library/struct.html> [Accessed 24 Aug. 2018]
- [19] PyPI. (2018). *pyproj*. [online] Available at: <https://pypi.org/project/pyproj/> [Accessed 24 Aug. 2018].

- [20] Initd.org. (2018). *Psycopg – PostgreSQL database adapter for Python — Psycopg 2.7.5 documentation*. [online] Available at: <http://initd.org/psycopg/docs/> [Accessed 24 Aug. 2018].
- [21] Gps.gov. (2018). [online] Available at: <https://www.gps.gov/technical/icwg/IS-GPS-200G.pdf> [Accessed 24 Aug. 2018].
- [22] Ford, D. (2018). *In-The-Sky.org*. [online] In-the-sky.org. Available at: <https://in-the-sky.org/> [Accessed 13 Sep. 2018].
- [23] Geoportail.gouv.fr. (2018). *Géoportail*. [online] Available at: <https://www.geoportail.gouv.fr/> [Accessed 13 Sep. 2018].
- [24] Oc.nps.edu. (2018). *Latitude/Longitude/Height ECEF via J-Script*. [online] Available at: <http://www.oc.nps.edu/oc2902w/coord/llhxyz.htm> [Accessed 13 Sep. 2018].
- [25] Userspages.uob.edu.bh. (2018). [online] Available at: http://userspages.uob.edu.bh/mangoud/mohab/Courses_files/ms-sat4.pdf [Accessed 14 Sep. 2018].
- [26] Mathematics Stack Exchange. (2018). *How to check if a point is inside a rectangle?* [online] Available at: <https://math.stackexchange.com/questions/190111/how-to-check-if-a-point-is-inside-a-rectangle> [Accessed 19 Sep. 2018].
- [27] Kahan, W. (2018). *Miscalculating Area and Angles of a Needle-like Triangle*. [ebook] Berkeley. Available at: <https://people.eecs.berkeley.edu/~wkahan/Triangle.pdf> [Accessed 19 Sep. 2018].