THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Institut National Polytechnique de Toulouse (INP Toulouse)*

Présentée et soutenue le *10 Février 2023* **par :**

Thomas VERHEYDE

Positionnement Précis Coopératif de Mobiles Low-Cost en Milieu Urbain

Precise Cooperative Positioning of Low-Cost Mobiles in an Urban Environment

JURY

Juliette Marais	Directrice de Recherche, HDR	Rapporteur
Fabio Dovis	Professeur	Rapporteur
Christophe Macabiau	Professeur	Directeur
Antoine Blais	Enseignant-Chercheur	Co-directeur
François-Xavier Marmet	Ingénieur	Invité
Damien Serant	Docteur	Invité

École doctorale et spécialité :
MITT : Domaine STIC : Réseaux, Télécoms, Systèmes et Architecture
Unité de Recherche :
École Nationale de l'Aviation Civile (ENAC) - TELECOM
Directeur(s) de Thèse :
Christophe MACABIAU et *Antoine BLAIS*
Rapporteurs :
Juliette MARAIS et *Fabio DOVIS*

Manuscrit dédié à :
Joanna, Franck, Fabienne, Lisa,
Pierre, Christelle, Chloé, Lydie,
Jacek, Dorota et Paulina.

À la mémoire de :
Romain, Maurice et Renata

Acknowledgments

First and foremost, I would like to express my gratitude to Thales Alenia Space and to the French Space Agency (CNES) for financing this research thesis and for their technical expertise throughout our working collaboration. The success of this study was made possible with the precious guidance from Christophe Macabiau (Thesis director) and from Antoine Blais (Thesis co-director). Moreover, I sincerely acknowledge François-Xavier Marmet (CNES) for its valuable support and proficiency all along this research project. I also thanks Damien Serant (Thalès Alénia Space) for its insights and advices. At last but not least, I would like to send many thanks to the TéSA research laboratory, led by Corinne Mailhes, for their assistance and professional counseling.

This research thesis was also made possible by the great support received from colleagues at the Signal Processing and Navigation ENAC research laboratory. A huge thank to Paul Thevenon and Jeremy Vezinet for their technical and practical assistance in various tasks and projects. They inspired my dedication and passion to pursue my career in the GNSS domain. I would like also to cite permanent members of ENAC's research laboratory (without any classification) Anais, Alexandre, Remi, Hèlene and Christophe Mo with who I shared everyday coffee habits and saw the evolution of this research study (supposedly with excitement). Plus, I would like to include PhD student and Post-doc (Maxandre, Anne-Marie, David, Guillaume and everyone else) for standing together in difficult times, but mainly with whom I enjoyed great moments.

Enfin, j'aimerais prendre le temps de remercier ma famille et mes amis qui m'ont accompagné, supporté et soutenu durant toutes ces années. Tout d'abord, prendre le temps (pour une fois) de remercier de tout mon cœur mes parents qui m'ont permis de m'épanouir en m'offrant des opportunités de vie incroyables afin que je puisse réaliser mes rêves. À Joanna, qui partage ma vie et m'encourage à être meilleur tous les jours pour elle : *Kocham cię.* À ma famille proche, qui me soutient et m'encourage en permanence dans les bons comme dans les moments difficiles. Nous serons toujours là, les uns pour les autres. J'adresse également une belle passe vrillée "à la fidjienne" (cad: une chistera champagne) à la *BOC* composée de Brice et de Guillaume, capable de me suivre dans les expéditions et projets les plus fous. Sans vous tous, rien de tout ça ne serait possible.

Merci !

Summary

In recent years, our society has been preparing for a paradigm shift toward the hyper-connectivity of urban areas. This highly anticipated rise of connected smart city centers is led by the development of low-cost connected smartphone devices owned by each one of us. In this context, the demand for low-cost, high-precision localization solutions is driven by the development of novel autonomous systems. After Google announced the release of Android GNSS raw data measurements on mobile devices, the enthusiasm around those low-cost positioning devices quickly grew in the scientific community. The increasing need of Location Based Services (LBS) provoked the rapid evolution of smartphones embedded low-cost Global Navigation Satellite System (GNSS) chipsets within the last few years. Most Android devices are now equipped with multi-constellation and multi-frequency positioning units. Preliminary studies explored the implementation of advanced positioning algorithms aiming at answering the demand for precise navigation and positioning on mobile devices.
However, various drawbacks prevent the realization of above-mentioned techniques on hand-held mobiles. Smartphones positioning capabilities are limited by the tight-integration of hardware components within the device. Integrated low-cost components, such as the linearly polarized antenna, are unoptimized for acquiring multi-frequency GNSS signals and their operation in constrained environment quickly becomes a challenge for mitigating disruptive multipath events. Moreover, due to a fierce technological competition between chipset manufacturers, embedded GNSS receivers have been conceived to act as ”black-box” processes. The receiver parameterization is kept confidential and only GNSS raw data measurements are outputted to the user.

In order to overcome those difficulties, this research work ambitions to develop a collaborative network positioning system between smartphones. A collaborative system is defined as a set of inter-connected users exchanging GNSS data in order to enhance network's users positioning performance. The implementation of a cooperative smartphone network takes advantage of the tremendous number of connected Android devices present in today's city centers for refining and improving users position accuracy and integrity in urban environment.

This research thesis presents a thorough analysis of Android GNSS raw data measurements aiming at lifting the ambiguity generated by receivers' ”black-box” processes on a

wide variety of Android smartphone brand and models. A wide data collection campaign, on 7 different smartphone models in real-life urban conditions, has been conducted for assessing the positioning performance of those contemporary low-cost devices.
After grasping the receivers' mechanisms, the implementation of Android GNSS raw data measurements in collaborative positioning algorithm has been investigated. An innovative smartphone-based double code difference method has been employed to compute the inter-phone distance between network's users, named Inter-Phone Ranging (IPR). This technique was tested for nominal and urban scenario cases and has demonstrated its reliability for collaborative positioning implementation.
Finally, a smartphone-based cooperative engine, called *SmartCoop*, was developed and evaluated. This software-based engine is integrated within the cooperative network infrastructure for delivering accurate positioning solutions to network's users. This collaborative estimation technique exploits the previously computed IPR ranges in a non-linear constrained optimization problem. An experimental protocol has been put in place in order to determine the estimation method efficiency through a series of simulation runs for both nominal and urban scenarios. The presented results analysis supports our hypothesis that smartphone-based collaborative engine enhances Android positioning performance in urban canyon.

Résumé

Ces dernières années, l'usage fait de notre téléphone mobile a beaucoup évolué. En effet, grâce aux développements technologiques ainsi qu'à l'hyper-connectivité de nos activités quotidiennes, ces appareils connectés ont pris une place centrale dans notre société. De nombreuses applications et services (Location Based Services : LBS) utilisent nos données de positionnement afin de répondre au mieux à la demande de services géolocalisés et personnalisés. Depuis de nombreuses années, nos smartphones sont équipés d'un récepteur GNSS donnant accès à la position de l'utilisateur. Aujourd'hui, la majorité de ces téléphones sont dotés d'un récepteur multi-constellations et multifréquences de plus en plus puissant.

Après l'annonce faite par Google, d'une mise à jour Android permettant de récupérer les mesures GNSS brutes, les smartphones sont rapidement devenus attrayant pour la communauté scientifique, en tant que récepteur GNSS bas-coût grand public. Les premières études menées sur ces appareils consistaient à transposer les algorithmes GNSS avancés (RTK et PPP) sur ces périphériques.
Cependant, l'implémentation de ces techniques se heurte à la faible qualité des données de positionnement mobiles. En effet, en raison d'une architecture restreinte et de composants auxiliaires à bas-coût, la performance du positionnement sur mobile est rapidement impactée par différents biais d'erreurs. Ce phénomène s'accentue en milieu urbain, notamment à cause de l'antenne interne du téléphone dont les spécifications sont inadaptées aux traitements de signaux multifréquences et au positionnement en environnement contraint. De plus, les paramètres et réglages de ces récepteurs embarqués sont tenus secrets par les constructeurs et seules les données GNSS brutes sont fournies à l'utilisateur, rendant leur utilisation ambiguë.

Afin de surmonter ces difficultés, ce projet de recherche ambitionne le développement d'un algorithme collaboratif dédié aux smartphones. La création de ce réseau coopératif permettrait de tirer avantage du nombre croissant de téléphones mobiles connectés agglomérés dans les rues des grands centres-villes.
Cette thèse présente une analyse complète et détaillée des mesures Android GNSS brutes afin de lever l'ambiguïté créée par les procédés de "boite-noire" employés par les récepteurs embarqués. Une grande campagne de collecte de données fut organisée pour évaluer la performance du positionnement sur téléphone en milieu urbain. Cette campagne a été

réalisée sur 7 smartphones en conditions réelles. À la suite de cette étude, une méthode de caractérisation des données GNSS brutes fut créée afin de couvrir le spectre de la plupart des modèles de téléphone mobile Android.
Après avoir analysé les mécanismes de positionnement internes aux smartphones, la mise en place d'un algorithme collaboratif a été instauré utilisant les données GNSS d'Android. Une méthode de double différence sur les mesures de code a été proposée afin de permettre l'estimation des distances entre utilisateurs d'un réseau coopératif. Cette technique innovante a été baptisée "Inter-Phone Ranging" (IPR). La fiabilité et la précision de cette méthode d'estimation sont démontrées par plusieurs études couvrant plusieurs environnements. Enfin, après avoir méthodiquement caractérisé la mise en place d'un réseau collaboratif de téléphones mobiles, un algorithme de positionnement collaboratif appelé "SmartCoop" est présenté. Ce dispositif permet d'exploiter les mesures d'inter-distances entre utilisateurs du réseau afin de résoudre un problème d'optimisation non-linéaire à contraintes. Cette méthode d'estimation a pour but d'améliorer la précision et de diminuer la dispersion de la position de tous les utilisateurs du réseau. Ce système coopératif a été validé en simulation. L'analyse des résultats obtenus nous permet de penser que cet algorithme coopératif innovant participe à l'amélioration globale de la performance du positionnement sur téléphone mobile en milieu urbain.

Contents

List of Figures

List of Tables

1 Introduction

Contents

1.1 Research Context & Motivations

In recent years, the rapid development of satellite positioning systems observed with the growth of GNSS constellations created an urge for precise positioning. The need for high-precision localization solutions have been driven by the development of novel autonomous systems in urban and constrained environments.
Nevertheless, classical GNSS-based positioning solutions have been proved to be highly impacted by urban environment. In constrained environments made of canopy, narrow streets and tall buildings, satellites signals reception quality often deteriorate. Indeed, GNSS signals suffer from time varying signal blockage, multipath and NLOS occurrences that directly impact the positioning estimation process. To this day, urban positioning remains a challenge for GNSS-based systems and algorithms.

In the meantime, an exponential increase of wireless signals being used in today's busy urban areas was observed. The apparition of new SoOP signals created opportunities for achieving robust precise urban positioning. The current development of the 5G technology is the perfect example of the previous statement. These signals were naturally envisaged for assisting GNSS algorithms in urban environments [20]. However, this promise relies on the implementation of heavy hardware infrastructure which are not available at this early deployment stage. On the other hand, smartphone devices capitalized on the hyper-connectivity characteristics of our society and packed a plethora of sensors within their core. This technological tool regroups the necessary capabilities and connectivity for hybridizing GNSS measurements.

Furthermore, this research work has been motivated by the release of GNSS measurements on Android devices and by observing the exponential development of LBS on those connected devices. In May 2016, Google announced during their I/O conference that GNSS raw data measurements would be accessible on any Android devices running Android Nougat (7.0) and above [21]. This announcement unlocked opportunities, for developers and the scientific community, to capitalize on the embedded positioning chipset inside smartphone devices. This trend reached chipsets manufacturers that engaged into a ferocious technological race for developing the most precise and accurate location device. This innovation race led to the normalization of multi-frequency and multi-constellation chipset embedded GNSS receivers within Android device. Thus, making smartphones positioning capabilities and specifications compete with COTS low-cost GNSS receivers.

The arrival on the market of smartphone devices with embedded low-cost GNSS receiver created a paradigm shift in measurements availability, density and computational power. This PhD thesis ambitions to develop a collaborative network positioning system between Android smartphone devices. Cooperative positioning localization has been recently studied for applications in the autonomous vehicular transportation and the robotic domains [22]. Only few studies have been made on smartphone-based networking. However, research works introduced preliminary groundwork concerning cooperative ranging between smartphones showcased in [23] as well as a collaborative positioning technique in [14]. Our intended collaborative network will take advantage of the tremendous number of connected devices present in highly frequented city centers. In the context of smartphone hyper-connectivity, we assume that a reliable, secure and efficient communication link is made available to network's users.

1.2 Objectives

The main objective of this thesis is to define, design and develop a collaborative positioning algorithm based on Android smartphone devices made for improving user urban positioning performance. In order to achieve this goal, the presented research work has been segmented around 3 phases:

1. **Characterizing Android GNSS Raw Data Measurements**
    - Recording Smartphone GNSS Measurements
    - Evaluation of Android GNSS Raw Data Measurements
2. **Designing a Smartphone-based Collaborative Network**
    - Defining a Specific Smartphone Cooperative Network
    - Developing a GNSS-based IPR Method
    - Establishing a Smartphone-based Collaborative Positioning Engine
3. **Evaluating a Smartphone-based Collaborative Positioning Engine**
    - Characterizing IPR Method in Multiple Environment
    - Simulating and Analyzing Collaborative Positioning Solutions in an Urban Environment

1.3 Contributions

This research work contributed to the publication and presentation of multiple papers as listed below:

[**2021**] Verheyde. T, Blais. A, Macabiau. C, Marmet. François-Xavier, *"SmartCoop Algorithm: Improving Smartphones Position Accuracy and Reliability through Collaborative Positioning"* IEEEXplore, vol.2021, International Conference on Localization and GNSS (ICL-GNSS 2021), pp. 1–6, doi: 10.1109/ICL–GNSS49 876.2020.9 115 564, 2021.

[**2020a**] Verheyde. T, Blais. A, Macabiau. C, Marmet. François-Xavier, *"Analyzing Android GNSS Raw Measurements Flags Detection Mechanisms for Collaborative Positioning in Urban Environment"* IEEEXplore, vol.2020, International Conference on Localization and GNSS (ICL-GNSS 2020), pp. 1–6, doi: 10.1109/ICL–GNSS49 876.2020.9 115 564, 2020.

[**2020b**] Verheyde. T, Blais. A, Macabiau. C, Marmet. François-Xavier, *"An Assessment Methodology of Smartphones Positioning Performance for Collaborative Scenarios in Urban Environment"* Proceedings of the 33rd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2020), pp. 1893–1901., Sep. 2020.

[**2019**] Verheyde. T, Blais. A, Macabiau. C, Marmet. François-Xavier, *"Statistical Analysis of Android GNSS Raw Data Measurements in an Urban Environment for Smartphone Collaborative Positioning Methods"* International Navigation Conference (INC 2019), Edinburgh, UK. 2019

1.4 Thesis Outline

This research work is organized in two parts:

1. **Part I: Android Smartphone Positioning**
   The first part introduces the concept of Android smartphone positioning. A preliminary analysis defines the positioning capabilities of Android mobiles and identifies the strengths and weaknesses associated to this contemporary low-cost platform. The newly accessible Android GNSS raw data measurements are characterized and analyzed in order to provide an optimal groundwork for integrating smartphone measurements in a collaborative system.

   - *Chapter 2: Smartphone Positioning Overview*
     This chapter provides a global assessment of Android devices architecture. A first study compares the smartphone characteristics to COTS low-cost GNSS receivers. Multiple smartphone's sensors measurements are then presented, identifying the strength of this low-cost device. On the other hand, smartphone hardware analysis provides a root cause for the weaknesses associated with smartphone positioning. Finally, a complete overview of Android GNSS raw data measurement is displayed and highlights the retrieval process for accessing

smartphone GNSS measurements. Current Android software-based algorithms are presented and characterized.

- *Chapter 3: Smartphone Positioning Techniques*
  This chapter is dedicated to the positioning techniques and algorithms that can be implemented for taking advantages of the embedded GNSS receiver. A data collection campaign is presented where data measurements have been collected from various collaborative scenarios. Finally, retrieved Android GNSS raw data measurements are characterized for both nominal and urban environments. This chapter will be concluded by an analysis defining the suitability of Android data for collaborative purposes.

2. **Part II: Collaborative Positioning**
   The second part of this research thesis explores the existing collaborative positioning algorithms. Cooperative techniques aim at improving network's users positioning performance by processing additional shared GNSS measurements. The resulting collaborative algorithm is referred as an engine. A smartphone-adapted collaborative engine and network structure will be selected and discussed. Then, a double code difference GNSS-based algorithm is developed for estimating inter-phone distances within the defined collaborative network. Finally, our proposed smartphone-based collaborative engine called *SmartCoop* is presented and analyzed.

   - *Chapter 4: Definition & Methodology*
     This chapter is a throughout literature review concerning collaborative positioning techniques. The aim is to assess the compatibility of existing collaborative positioning methods with the implementation of a smartphone-based cooperative network. The structure of the selected method is presented in this chapter.

   - *Chapter 5: GNSS-based Collaborative Ranging*
     This chapter provides an adapted solution for computing Inter-Phone 3D Ranges (IPR) within a collaborative network of smartphones. The presented method is defined as a double differentiation technique on GNSS code measurements. The innovation is supported by a new approach taking into account Android GNSS raw data measurements specificities and by an adaptive algorithm.

   - *Chapter 6: Smartphone Collaborative Positioning*
     This chapter is a synthesis of previously presented analysis and algorithms regrouping smartphone positioning capabilities into a collaborative system. Our proposed solution encapsulates a non-linear optimization problem constrained by estimated IPR. A detailed analysis is displayed, underlining significant positioning performance improvements.

Figure 1.1 illustrates the thesis organization sequence and highlights the relation between each chapter.

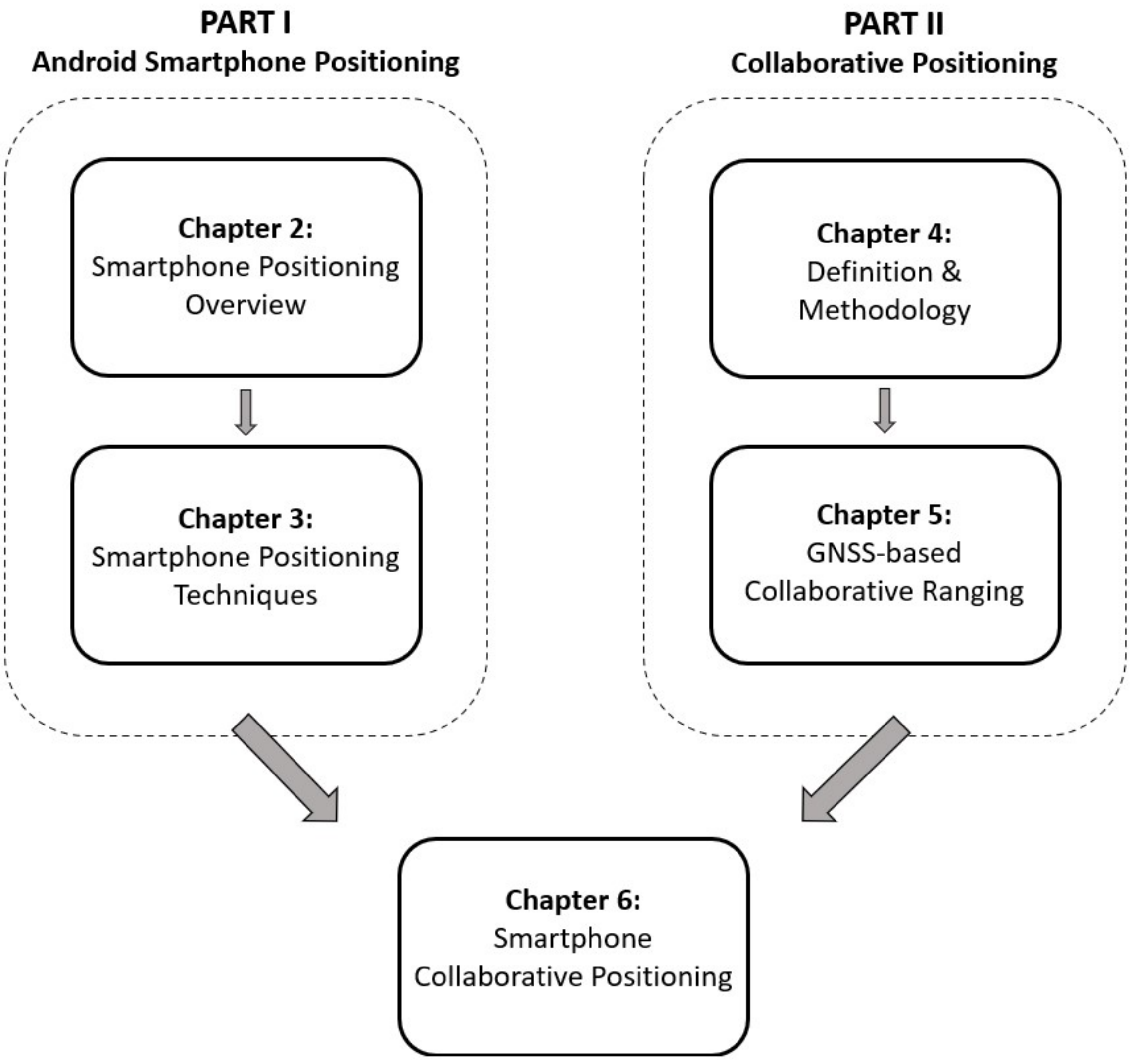

PART I
Android Smartphone Positioning
Chapter 2:
Smartphone Positioning Overview
Chapter 3:
Smartphone Positioning Techniques
PART II
Collaborative Positioning
Chapter 4:
Definition & Methodology
Chapter 5:
GNSS-based Collaborative Ranging
Chapter 6:
Smartphone Collaborative Positioning

Figure 1.1: Block Diagram of the Thesis Outline

Part I

Android Smartphone Positioning

2 Smartphone Positioning Overview

Contents

2.1 GNSS Low-cost Receivers

A GNSS receiver process signals from one designated or multiple satellites that are part of a navigation system. The role of the receiver is to estimate its position, velocity and time characteristics. A detailed description of GNSS receiver architecture and processes is given in appendix chapter A. In recent years, the popularity of GNSS applications were boosted by the appearance of mass-market GNSS receivers. Smartphones' embedded GNSS receivers became the norm for everyday positioning and thus potentially unlocking the access to a vast connected network perfectly suitable for opportunistic collaborative

positioning network.
In this chapter, a panorama of low cost receivers will be made by analyzing the origin of such device and by looking at their characteristics. Smartphones will be also presented as the newly developed low-cost GNSS devices made for daily applications and for Location Based Services (LBS). Then, Android smartphone data features will be explained and detailed. Those modern compact devices pack multiple sensors that can be recorded and used for positioning, navigation and timing (PNT) purposes. Thereafter, a detailed investigation of smartphones hardware will be performed through characteristics assessment, performance analysis and embedded components characterization. Finally, Android GNSS raw data measurements will be thoroughly presented. This final section will detail the raw data characteristics and assessment. A complete understanding of Android GNSS raw data measurements will be provided from how to access the data, to recording options available to smartphone users.

2.1.1 Definition

Historically, GNSS receivers has been developed for military purposes in the late seventies. The next step towards low-cost receivers was the first civilian GNSS receiver made by Standford in 1980 for scientific and technical works [24]. By the end of the 20th century, more and more civilian receivers had been developed. However, their prices varied between 25.000$ and 300.000$ making those instruments extremely expensive and hard to acquire [1]. Another major change for the GNSS community happened on May 2nd, 2000. The selective availability feature on GPS was turned off. This feature, added by the United States Department of Defense (DoD), would add a time varying error to the broadcasted civilian measurements, intentionally. By removing this induced error and with the rapid technology improvement, especially in the field of microelectronics, receivers became smaller and smaller and were offered at a cheaper price for everyone as a personal Position, Navigation and Timing (PNT) solution component (e.g personal GPS for car navigation).

This change of users community made the manufacturers realized that the GNSS receiver business market size was significantly increasing. Following that, constructors made

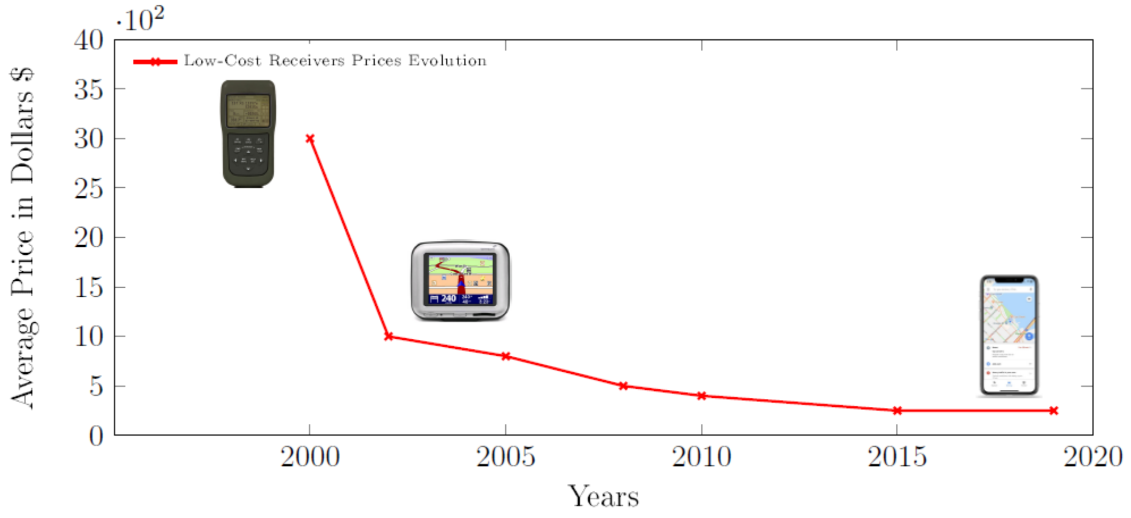


Figure 2.1: Chronological Prices Evolution of GNSS Low-Cost Receivers. [1] [2]

cheaper (few hundreds of dollars) receivers for an everyday use. Those kind of units were defined as low-cost, opposed to high-end receivers used for geodetic purposes for example. Nowadays, GNSS low-cost receivers are widely available. Those cheap devices can be used for multiple application techniques such as: car navigation, pedestrian navigation, positioning, search & rescue and more.

Smartphones' embedded GNSS receiver is the most popular low-cost receiver. Everyone has now access to positioning and navigation in the palm of their hands. GNSS low-cost techniques are now part of many LBS and are part of the idea behind a more connected world: the Internet of Things (IOT). By definition, GNSS low-cost receivers are cheap hardwares priced at a few tens or hundreds of dollars. Figure 2.1 shows a graph on prices evolution in time of GNSS low-cost receivers. This figure represents the retail price of integrated GNSS receivers, their price evolution was mainly driven by the cost of microelectronics constituting their hardware.

Low-cost receivers have been brought to market by well-implanted companies in the GNSS domain. Off the shelf low-cost receivers usually comes on a circuit board with connectors that surrounds the GNSS chipset module. Such low-cost receivers are paired with standard patch antenna that are characterized as easy-to-use and reliable for standard positioning solutions. Nowadays, those devices costs tens of dollars and can deliver around a meter accuracy positioning solution. As an example, we can cite Ublox receivers with their ZED-F9P application board. According to the receiver description provided by the company [25], this multi-constellation and multi-frequency receiver can be used for advanced GNSS algorithm (Precise Point Positioning (PPP), Real-Time Kinematic (RTK)). In the literature, multiple research works test, analyze and evaluate performances of GNSS low-cost receivers [26], [27].

Finally, a low-cost GNSS receiver is defined by its retail price and its performances. These GNSS instruments were initially developed for fitting a market segment of layman consumers looking for PNT services without a need for high-precision solutions. Nowadays, the gap between high-end COTS devices and low-cost GNSS receivers is shrinking by bringing scalable and affordable meter-level positioning accuracy performance. These progresses have been driven by the study and implementation of advanced positioning algorithms on those devices [28].

2.1.2 Smartphones

Smartphones contain the most recent technology embedded in their hardware due to a ferocious competition between manufacturers. This rivalry pushes subcontractors such as GNSS chip manufacturers to create innovative GNSS embedded products. The objective in the GNSS world is to be able to achieve centimeter level precision in mass-market devices such as smartphones. This goal will be soon obtained thanks to the new generation of GNSS chips and their characteristics. However, positioning processes are not prioritized over other smartphone features (e.g: Camera performance, battery saving, social experiences, etc.) Critical positioning components, such as antennas and processors, are not developed exclusively for positioning and navigation purposes and tend to be multipurpose instead. Therefore, impacting the overall positioning performances. A detailed characterization of smartphone data features is given below in section 2.2. Thereafter in section 2.3, a review of smartphone hardware will be exposed, highlighting architecture limitations and choices.

In the past years, technological improvements made on smartphone embedded GNSS receivers made those devices appealing to the scientific community in order to be used as low-cost receivers. Furthermore, commercial opportunities rose from this technological achievement, that led mobile manufacturers to compete in order to obtain the world's most precise smartphone. Multiple phone companies joined the race, releasing dozens modern smartphones equipped with various chipset models, that are multi-frequency and multi-constellation capable. Those technological progresses potentially unlocked the access of a wide crowd-sourced and connected network of embedded smartphone receivers. Indeed, the particularity of those low-cost GNSS receivers is the incredible rising number of devices in city centers. Key figures and numbers shows the hyper connectivity of today's urban areas and the role played by the smartphone industry in the positioning domain. 3 billions mobile applications rely on positioning capabilities, 277 Galileo enabled smartphone models have been released and most recently European Union Agency for the Space Program (EUSPA) claimed that an approximate number of 2 billion Galileo enabled devices have been sold up to this date [29]. Those numbers show the degree of involvement from the smartphone industry regarding the development of low-cost positioning devices made available for everyone. Those industrial developments were initiated by the released of Android GNSS raw data measurements in late 2016, adding to the list of Smartphone data features.

2.2 Android Smartphone Data Description

Android smartphone devices are data featured packed mobiles. In recent years, smartphones have been equipped with a multitude of sensors that enable third-party applications and additional features. Smartphones' positioning estimation algorithm gain from the variety of sensors present on mobile devices. Data ranges from GNSS raw data measurements, IMUs, Signals of Opportunity (SoOP) to smartphone specific sensors. This section will detail the type of data that can be recorded on Android smartphone devices. In the purpose of this research we will solely focus on sensors and data that could be used in navigation and positioning algorithms.

2.2.1 GNSS

The pre-requisite data necessary for positioning capabilities of modern smartphones is GNSS data. Android GNSS raw data measurements is derived from the interpretation of received satellite signals by the embedded positioning receiver. The recognized signal model of GNSS signals is presented in equation 2.1.

$$S_T(t) = \Re(A\, d(t)\, c_m(t)\, e^{-j2\pi f_L t}) = A\, d(t)\, c_m(t)\, cos(2\pi f_L t) \tag{2.1}$$

with:

- A is the Amplitude of the transmitted signal
- $d(t)$ represents the data component of the signal, carrying the binary data information
- $c_m(t)$ is the PRN sequence of the m^{th} satellite

- f_L represents the carrier frequency

Classical GNSS receiver architecture and processing are detailed in appendix A. This appendix chapter exposes the processes required to exploit satellite signal for positioning purposes. The model shown is directly applicable for smartphone GNSS signal processing.

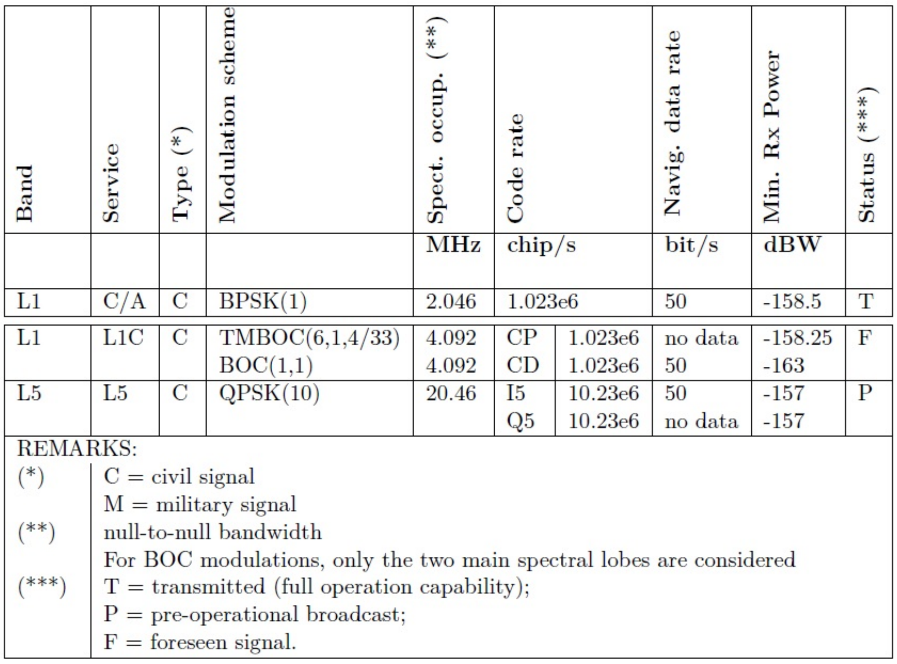
Smartphone GNSS processes have generally been executed by the embedded System-on-a-Chip (SoC). Theoretically, SoC systems are made of different processor units such as: Central Processing Unit (CPU), Graphics Processing Unit (GPU), Digital Signal Processor (DSP) and including GNSS receivers. The processor responsible for processing GNSS signals will be referred in this thesis as: Smartphone embedded GNSS receivers. Further details about smartphone's GNSS receivers are provided in section 2.3.1.
Modern smartphones are now equipped with multi-constellation and multi-frequency chipsets. The following subsections will present the type of signals received by Android smartphones divided per constellation. The next description is based on the most recent chipset specifications (Qualcomm Snapdragon 888, released end of 2020.) [30].

2.2.1.a GPS Signals

GPS signals are tracked by most Android smartphones. With the arrival of multi-frequency capable Android devices, smartphone embedded chipsets are able to process GPS L1 C/A and GPS L5 signals. The GPS L1 band is transmitted on a carrier frequency centered at 1545.42 MHz whereas L5 signals are broadcasted on a carrier frequency centered at 1176.45 MHz. Signals characteristics are detailed for the GPS constellation in table 2.2.

2.2.1.b Galileo Signals

Android smartphone embedded GNSS receivers are able to process Galileo E1 and E5a signals. The Samsung S8 mobile was the first device capable of recording and processing Galileo signals with the embedded Qualcomm Snapdragon 835 SoC when it was released in late 2017. Signals characteristics are detailed for the Galileo constellation in table 2.3.

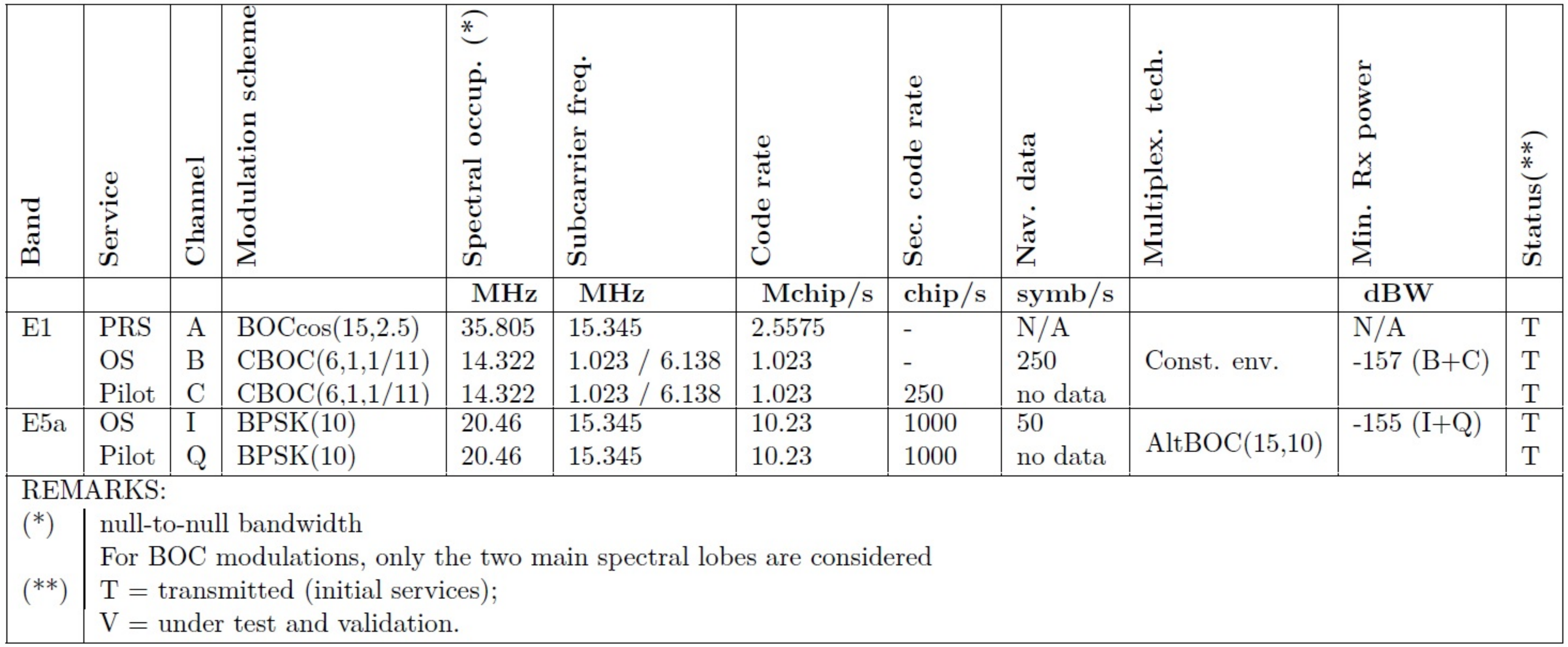


Band	Service	Type (*)	Modulation scheme	Spect. occup. (**)	Code rate		Navig. data rate	Min. Rx Power	Status (***)
				MHz	chip/s		bit/s	dBW	
L1	C/A	C	BPSK(1)	2.046	1.023e6		50	-158.5	T
L1	L1C	C	TMBOC(6,1,4/33)	4.092	CP	1.023e6	no data	-158.25	F
			BOC(1,1)	4.092	CD	1.023e6	50	-163	
L5	L5	C	QPSK(10)	20.46	I5	10.23e6	50	-157	P
					Q5	10.23e6	no data	-157	

REMARKS:

(*)	C = civil signal
	M = military signal
(**)	null-to-null bandwidth
	For BOC modulations, only the two main spectral lobes are considered
(***)	T = transmitted (full operation capability);
	P = pre-operational broadcast;
	F = foreseen signal.

Figure 2.2: GPS Signals Characteristics Recordable by Android Smartphone. Table modified from [3]



Band	Service	Channel	Modulation scheme	Spectral occup. (*)	Subcarrier freq.	Code rate	Sec. code rate	Nav. data	Multiplex. tech.	Min. Rx power	Status(**)
				MHz	MHz	Mchip/s	chip/s	symb/s		dBW	
E1	PRS	A	BOCcos(15,2.5)	35.805	15.345	2.5575	-	N/A	Const. env.	N/A	T
	OS	B	CBOC(6,1,1/11)	14.322	1.023 / 6.138	1.023	-	250		-157 (B+C)	T
	Pilot	C	CBOC(6,1,1/11)	14.322	1.023 / 6.138	1.023	250	no data			T
E5a	OS	I	BPSK(10)	20.46	15.345	10.23	1000	50	AltBOC(15,10)	-155 (I+Q)	T
	Pilot	Q	BPSK(10)	20.46	15.345	10.23	1000	no data			T

REMARKS:

(*) null-to-null bandwidth
For BOC modulations, only the two main spectral lobes are considered

(**) T = transmitted (initial services);
V = under test and validation.

Figure 2.3: Galileo Signals Characteristics Recordable by Android Smartphone. Table modified from [3]

2.2.1.c BeiDou Signals

Android smartphones are now able to record BeiDou signals, B1C and B2a. The first signal is part of the L1 band frequency plan with a central carrier frequency of 1575.42 MHz. BeiDou Navigation Satellite System (BDS) B2a has a central frequency at 1176.45 MHz, equivalent to the L5 band of Galileo and GPS frequency plan.

2.2.1.d GLONASS Signals

Globalnaya Navigatsionnaya Sputnikovaya Sistema (GLONASS) Frequency Division Multiple Access (FDMA) L1 signals can also be tracked by smartphone embedded GNSS receivers. However, the GLONASS L1 band does not coincide with the GPS and Galileo L1 band. Satellites from the Russian GNSS constellation emitting on the L1 band with a frequency ranging from 1592.9525 MHz to 1610.485MHz can be used on modern smartphone device for positioning purposes.

2.2.1.e Other Signals

Recently, Quasi-Zenith Satellite System (QZSS) and Navigation Indian Constellation (NAVIC) GNSS radio signals can also be processed by embedded smartphone devices.

2.2.2 Inertial Measurement Unit (IMU)

The integration of Inertial Measurement Unit (IMU) within smartphone devices was firstly motivated by consumer needs (i.e: step counting, screen rotation capability). Sensor fusion techniques, already present in the GNSS literature, were implemented in a second time on smartphones taking advantage of the presence of low-cost IMU components. Modern Android devices are equipped with Micro-Electro Mechanical Systems (MEMS) IMU which role is to assist the positioning and navigation solutions in difficult situations. A typical IMU consists of a 6-axis MEMS unit, combining a 3-axis gyrometer (on x, y and z) and a 3-axis accelerometer (on x, y and z). Moreover, a 3-axis magnetometer is usually added to the unit.

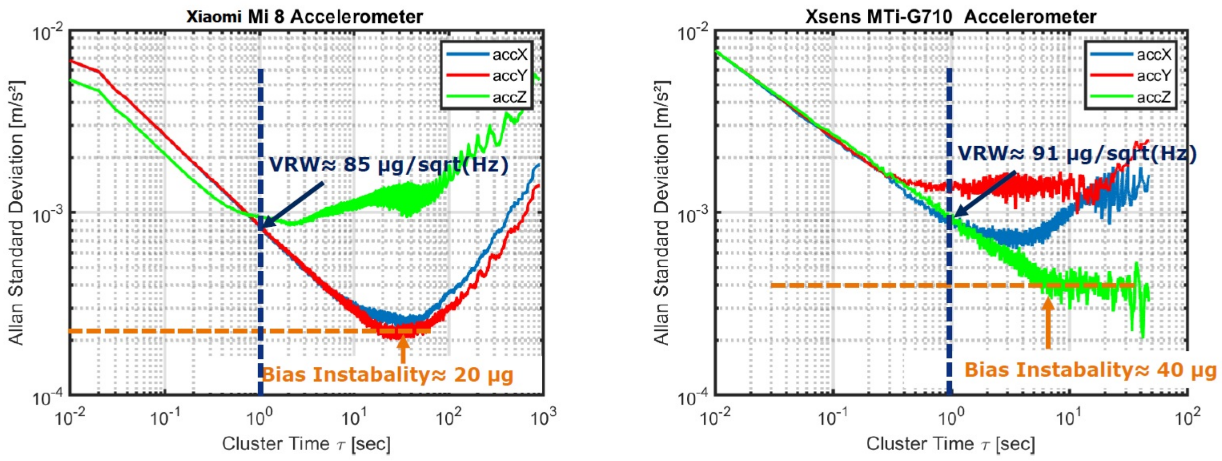


Figure 2.4: IMU Accelerometer Allan Standard Deviation Comparison between 1€ Smartphone's IMU (Left) and a Higher-end 2000€ MEMS IMU (Right) [4]

Multiple research papers characterized the cheap IMU components against higher-end products [31]. Figure 2.4 shows the Allan standard deviation comparison between a Xiaomi Mi8 embedded IMU and a high-end Xsens IMU. Two parameters are studied here: the bias instability and the Velocity Random Walk (VRW). It has been found out that smartphone's MEMS IMU demonstrates a high level of bias stability. However, a clear difference can be seen on the z-axis stability component. This behavior is explained in [32]. It is argumented that the manufacturing process of the three-axis MEMS accelerometer on smartphone devices differ due to the lack of space in mobile device hardware. High-end MEMS IMU device are incorporated in a set of 3 accelerometer chips orthogonally arranged for a more accurate definition of 3D forces. Further research work [33], validated the stability and reliability behaviors of IMU to be integrated within RTK fusion algorithms.

2.2.3 Smartphone Duty Cycle

As exposed above, positioning services are not the primary functions of a smartphone device. In order to support low power consumption and battery drainage, a technique of duty cycle power saving mode as been widely used on smartphones. The goal of this feature is to minimize the computational loads needed by the embedded GNSS chipset during inactive periods. This function puts to sleep the positioning hardware and thus affects the reception of GNSS carrier phase observation. Multiple studies demonstrated that this mode increases cycle slip occurrences in the positioning engine by up to 90% [34] [35]. Fortunately, the possibility to turn off this feature was introduced by the Android P (9.0) Open Service (OS). Turning off duty cycle allows the embedded receiver to continuously track GNSS signals and thus reducing the number of ”battery-saving induced” cycle slips. In recent devices, the option to turn off duty cycle is directly prompted to the user will being asked to turn on the *High-Precision Positioning* option.

2.2.4 Signals of Opportunity (SoOP)

Signals of Opportunity (SoOP) are defined as non-GNSS signals that can be derived from their primary function and used as a positioning signal. Those signals are only used to improve the positioning performance and not as a substitute of GNSS radio signals in positioning algorithms. The promise behind the use of SoOP would be to provide resilient positioning and navigation solutions in any environment. A plethora of positioning techniques can be implemented such as: Angle-of-Arrival (AOA), Time-of-Arrival (TOA), Time Difference of Arrival (TDOA), Frequency Difference of Arrival (FDOA) or also Received Signal Strength (RSS). The main drawback from exploiting SoOP signals for positioning purposes would be the complex hardware needed for processing multiple signals ranging a wide frequency spectrum. However, the use of SoOP signals in smartphone can be easily envisioned since the hardware necessary to capture a variety of signals is already present on our complex mobile devices. The two main signal of interest that can be used in smartphone-based positioning are Bluetooth and WiFi.

2.2.4.a Bluetooth

Bluetooth wireless technology is widely available on Android devices. Two types of data sharing can be made via this technology. The first one, called Classic, is used for streaming audio. The second, Bluetooth Low Energy Consumption (LE), is used to transmit data packets in battery operated sensors devices and thus is widely used on smartphone devices. Bluetooth operates at frequencies around 2.4 GHz and are following the IEEE 802.15.1 standard. Traditionally, Received Signal Strength Indicator (RSSI) techniques are used to exploit Bluetooth signals for positioning purposes. Recently with the arrival of Bluetooth 5.1, a direction finding feature was introduced allowing higher accuracy solutions in the positioning domain. Such techniques and algorithms are used for improving indoor positioning, way finding, assets tracking and more [36] [37]

2.2.4.b WiFi

WiFi is characterized as a technology using wireless radio signals based on the IEEE 802.11 standards. Wireless Local Area Networking (WLAN) signals operate in a 2.4 and 5 GHz frequency bandwidth. This technology has been popularized for its used on computers and smartphone. Multiple positioning methods can be used: Trilateration, fingerprinting, TOA and more [38]. Recently, an emerging positioning technique is used for localization purposes using WiFi signals. The Round-Trip Time (RTT) measures a distance between the device and a nearby router without requiring time synchronization between the two instruments and is specified by the IEEE standard 802.11 mc. Since the release of Android 9.0 (Pie), smartphones can provide WiFi RTT measurements using RTT-ready surrounding devices. Such methodology can be implemented for phone-to-phone localization and for aiding traditional positioning methods in harsh conditions [39].

2.2.5 Smartphone Sensors

Other type of data can be retrieved from Android smartphone. Specific smartphone sensors and features can be relevant for positioning and navigation algorithms. Those extra sensors are being implemented by smartphone industries in order to empower customers with additional information and data to be shared on social medias. Furthermore, the technological battle fought by mobile constructors led to technological innovations being introduced to mass-market consumers.

2.2.5.a Motion

Motion sensors are typically derived from MEMS IMU device. Although, raw measurements can be retrieved, recorded and used for the purpose of localization. Moreover, through the use of its Application Programming Interface (API), Android gives access to motion derivative type of information. User activity (standing, walking, running, biking) can be retrieved, thus providing accurate users motion states that can be adapted to the system dynamic.

2.2.5.b Environmental

Environmental awareness can become an important parameter for advanced positioning algorithm. Android smartphones provide information data of four parameters:

- Ambient air and internal temperature [°C]
- Pressure [hPa]
- Relative Humidity [%]

- Light [lx]

An an example, we can the barometric measurements given by smartphone environmental sensors for demonstrating their benefit in positioning computations. Pressure measurements can be used for estimating user dynamic and relative vertical movement. It has been shown, [40], that the derivative of the barometric measurements is highly reliable for estimating user dynamic.

2.2.5.c LiDAR

The innovation war fought by smartphone manufacturers allows users to be gifted with surprising features to their newest smartphone. In march 2020, Samsung released a smartphone with an embedded LiDAR. LiDAR stands for Light Detection and Ranging, it is a remote sensing method that utilizes pulsed laser for measuring distances. LiDAR positioning methods are employed in the automotive industry and could be exported and applied to the smartphone domain [41].

2.3 Smartphone Hardware

Smartphone can be described as complex hardware devices. In order to better apprehend Android GNSS raw data measurements, a study of smartphone hardware specification will be made in this section. Smartphones are defined as portable device that combines communication and computing functions. They are distinguished from their features packed and hardware capabilities. GNSS positioning and navigation capabilities are one of many features, it has to be reminded that smartphone are not GNSS specific devices. This precision is important to be made in order to understand the technical and architecture choices made by mobiles constructors.

By definition, smartphone hardware main limitation is the lack of space on the main Printed Circuit Board (PCB). Therefore the integration of new feature components is often made in detriment of other components. Since GNSS positioning and navigation are not the smartphone's main feature, it can be expected that GNSS receivers components takes a smaller space on the portable device's motherboard. Furthermore, components selected for such positioning operations are often cheap and not optimal for position computations. Figure 2.5 shows the hardware architecture of a Samsung S8 that as been dismounted for the purpose of this research. The Samsung S8 flagship smartphone was the first Galileo ready device, equipped with a SoC Snapdragon 835 chipset. The list of components have been studied [42] in order to identify GNSS receiver components highlighted in figure 2.5. The first conclusion that can be drawn is the size of the GNSS receiver components compared to the camera module for example.
This section will explore the specification of a GNSS embedded receiver. This analysis will be focused on the embedded chipset receiver and on the antenna characteristics.

2.3.1 Embedded Chipset

Computations and operations are made on the embedded SoC processor of the smartphone device. SoC are considered by most the brain of the smartphone. SoC chipsets are built around multiple processing units. The main processing units are listed below:

- CPU: Runs the Android Operating System (OS).

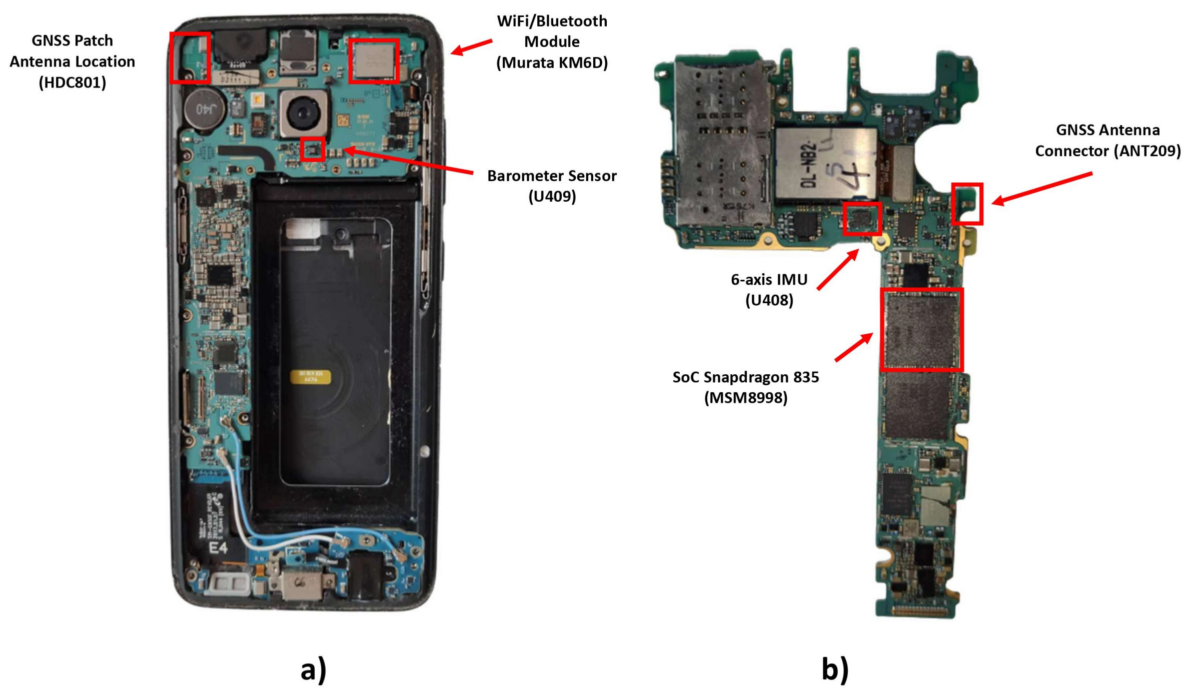

GNSS Patch Antenna Location (HDC801)
WiFi/Bluetooth Module (Murata KM6D)
Barometer Sensor (U409)
GNSS Antenna Connector (ANT209)
6-axis IMU (U408)
SoC Snapdragon 835 (MSM8998)
a)
b)

Figure 2.5: Samsung S8 Hardware - Motherboard Architecture Key Components Layout. a) Smartphone Backside with a Frame Mounted Motherboard. b) Front-face of the Standalone Motherboard

- GPU: Handles graphic related content.
- Image Processing Unit (ISP): Process phone's camera data
- DSP: Records and process the numerous digital signals received by the device.

SoC chipsets are then paired with GNSS sensor hub microcontroller. The Microcontroller Unit (MCU) is tightly integrated with multiple sensors for achieving precise positioning. The goal of such hardware architecture is to limit the MCU PCB footprint and to keep low system-level power consumption. Figure 2.5 shows the SoC of the smartphone, paired with various sensors.
Currently, smartphone embedded GNSS receivers have multi-constellation and multi-frequency capabilities achieving meter-accuracy positioning. Constructors are achieving such performances while keeping the price of their SoC chipsets at a couple dollars baseline, making those devices very low-cost GNSS receivers. Broadcom, Qualcomm and HiSilicon are the market leaders among integrated GNSS chip manufacturers. Their MCU chipsets are equipping most Android smartphone.

2.3.2 Antenna Design

The main constraint to overcome in smartphone-based positioning is to comply with a very low grade GNSS embedded antenna. Typical GNSS antenna are designed taking into account GNSS signal characteristics. Most of the time, such antenna are Right-Hand Circular Polarization (RHCP) with a spherical spatial reception pattern. This configuration allows the reception of satellite signals from any azimuthal directions. On the other hand, smartphone antenna have a patch configuration. Smartphone antenna

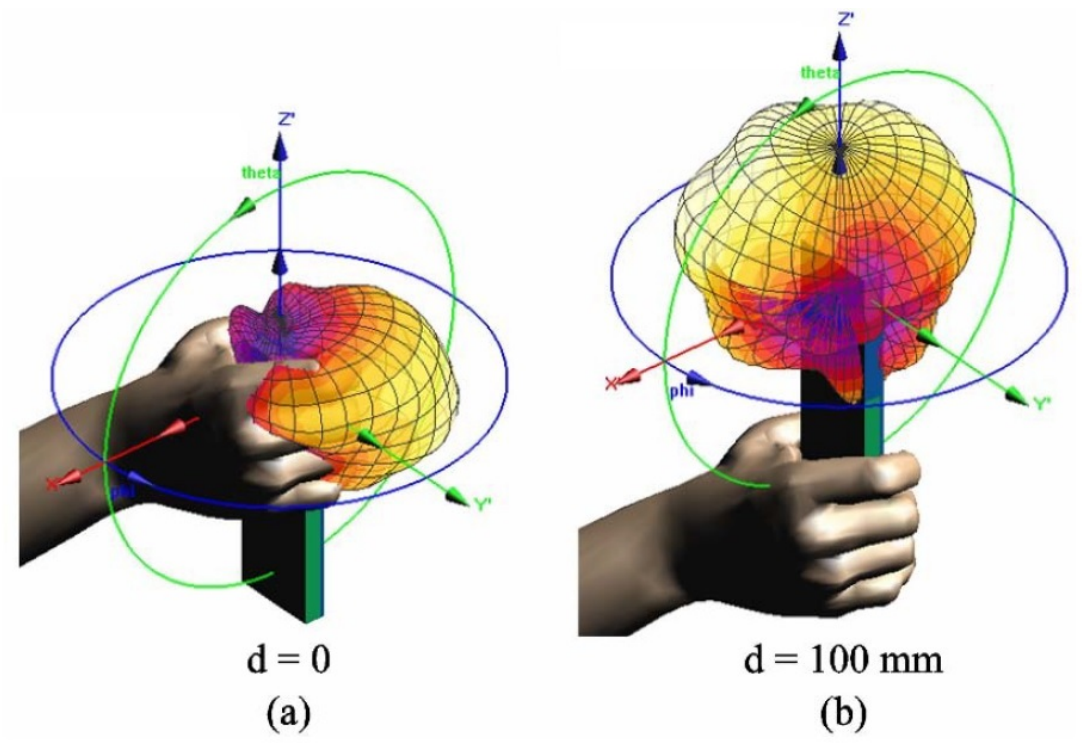


Figure 2.6: Simulated 3D Radiation Pattern of a Smartphone Antenna in hand-held Conditions at 1795 MHz. a) $d = 0mm$. b) $d = 100mm$. Figure taken from [5]

are low-profile thin microstrip embedded with the motherboard PCB. An example of such antenna can be seen on figure 2.5 on the top left corner of the left picture. This antenna specification is considered linearly polarized due to the previously explained configuration with the motherboard. Linearly polarized antennas tend to under perform compared to RHCP antenna, especially in urban areas where multipath signal distortion greatly impact linearly polarized antenna performances.

Multiple studies were made in order to characterize smartphone's antenna performances. In his paper [43], Pesyna tested and characterized antenna's behavior compared to different antenna classes. It has been found that smartphone's antenna loses between 5 and 15 dB in sensitivity compared to other products. Such results demonstrate the poor quality of linearly polarized patch antenna and the anticipated difficulty encountered by the receiver to retain lock of GNSS signals. Figure 2.7 represents the gain pattern of a Samsung S3 embedded antenna. This analysis, [6], has been made in reference to a survey-grade antenna. The antenna radiation pattern seems to be irregular. Patches of blue colors indicate a difference of at least 10 dB-Hz between the signal strength characteristics received by the smartphone. Even when the smartphone is not hand-held, it appears that the effect of local multipath cannot be mitigated by the smartphone embedded GNSS patch antenna. Further analysis have been made in order to characterize real life application conditions, namely when the smartphone is hand-held [5]. Figure 2.6 shows the simulated 3D radiation pattern of the antenna while the phone is being held by the users (represented by the distance d on the image). It is now clear that the smartphone embedded antenna design is not optimal for recovering and tracking GNSS signals. However, this design choice can be explained by the tight integration characteristics of the small motherboard device.

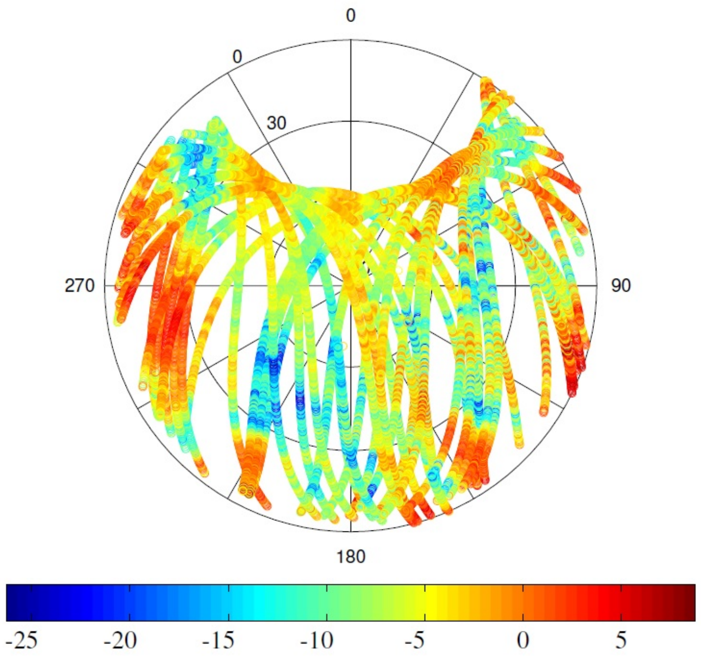


Figure 2.7: Relative C/N_0 Skyplot for a Samsung S3 Antenna with respect to a High Grade Antenna [dB-Hz]. Figure extracted from [6]

2.4 Android GNSS Raw Data

A promising initiative, led by Google, allowed developers and the GNSS community to access GNSS raw data measurements from Android smartphones devices. This innovation granted access to the embedded GNSS chipset data outputs. For years before that, smartphone positioning solutions were given to the user as an output of a "black-box" process made by the device OS.
In may 2016, during the "I/O" conference, Google announced that GNSS raw data measurements will be accessible with their updated smartphone operating system (OS) Android Nougat (7.0). In previous Android versions, only GPS status, satellite PRN and the final position could be used by potential applications and users. The chipset embedded in the smartphone could then choose to select GNSS constellations and PVT algorithms based on protocols directly implemented in the hardware [7]. The arrival of GNSS raw data measurements permits users to unlock the full capacities of their GNSS chipset. Even if users do not have a direct control of the chipset itself, they have access to decisive GNSS informations such as pseudoranges, carrier-phase and the estimated Doppler frequency in order to derive a more accurate position [44]. GNSS Raw data measurements corresponds to the input data of the PVT algorithm. Having control over this information could lead the design of a controlled environment where a data quality monitoring could be made. This initiative has been warmly welcomed by the GNSS community as it was a real opportunity to obtain sub-metric positions on smartphones.

The section will provide the description of the Android *Location* API structure before introducing GNSS raw data measurements that can be recorded and processed by an Android Smartphone. Finally, a review of data recording options will be presented; including ENAC's own Android application called *SmartLogger*.

2.4.1 Android *Location* API Structure

The Android *Location* API allows smartphone users to retrieve smartphone's GNSS raw data measurements. An API is defined as a collection of protocols and functions listed in a package class accessible by developers. Since the launch of the Android 7.0 OS, GNSS raw data measurements are accessible from the *android.location* API. As discussed in Appendix chapter A, section A.1.3, GNSS raw data measurements are the data output of the signal processing block of GNSS receivers. Figure A.4 shows the graphical representation of a typical GNSS receiver highlighting the chain process for recovering those raw data measurements.
According to Google [21], currently 84% of existing Android devices have the possibility of recording GNSS raw data measurements. As of today (end of 2021), the technical support of Android raw data measurements is mandatory for all new smartphones since Android 9.0. The diagram description of the API structure is depicted in figure 2.8. This structure has been implemented since Android 7.0 and is still used today with more recent OS Android updates. As a reminder, the recorded GNSS data comes directly from the embedded GNSS chipset unbiased. Users do not have a direct access to the chipset itself.

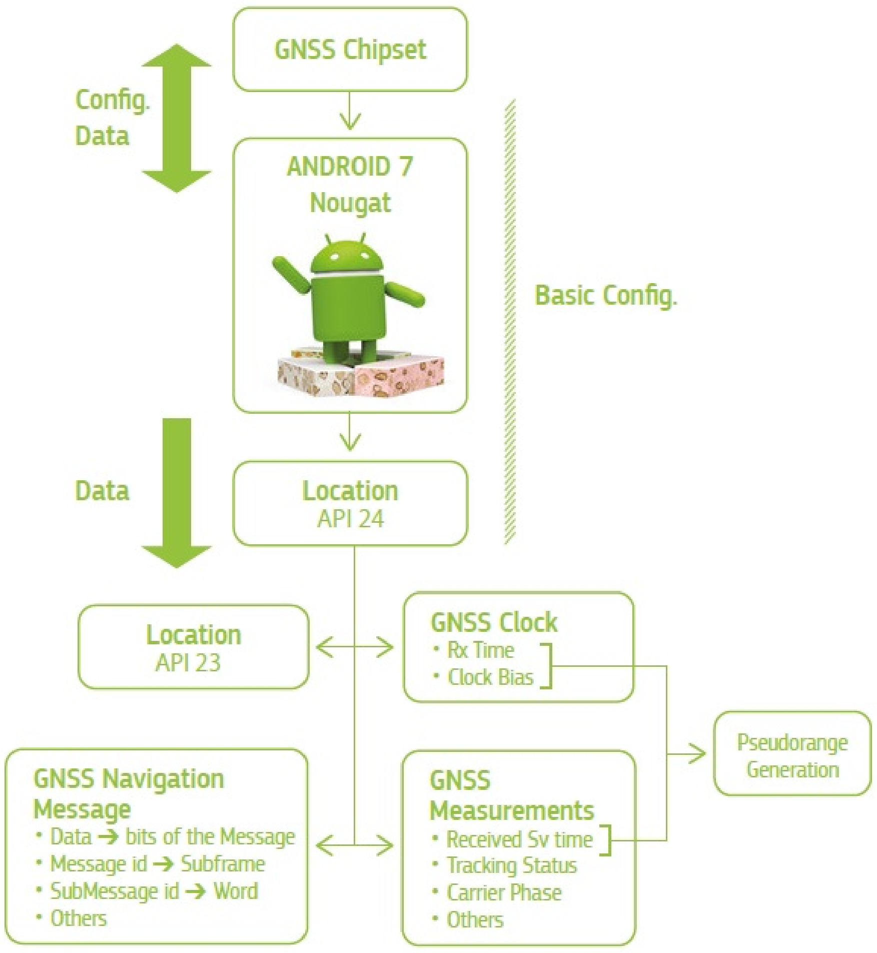


Figure 2.8: Android *Location* API Diagram. Valid since Android 7.0 (Nougat) [7]

2.4.2 GNSS Raw Measurements Description

The Android API gives access to the smartphone embedded GNSS receiver raw data measurements. These GNSS raw data measurements can be recorded, replayed and used by the scientific community or by software application developers for developing advanced positioning and navigation products. The interaction between users and the chipset is limited to basic executable commands (e.g: ON/OFF, Cleaning Assisted GNSS (AGNSS) data, Turn OFF Duty Cycle). On the other hand, configuration settings and positioning estimation algorithm are driven by the chipset itself. Measurements data given by the Android device are directly outputted by the embedded GNSS receivers. Raw Measurements can be divided into different categories, GNSS Measurements, Clock measurements, Antenna Information data and Navigation messages. A detailed description of recordable data can be found in table 2.10 and 2.11. Most typical GNSS data measurements needed for Position, Velocity and Timing (PVT) computations are not directly provided by the Android API (e.g: Pseudoranges, GNSS Time, Carrier-Phase and Doppler Frequency). Thus, GNSS data measurements needs to be estimated with the raw measurements provided by the Android API. GNSS data measurements computations are provided below for the main parameters necessary in positioning algorithm. Their associated models are shown in appendix chapter B.1. Classical measurements model can be applied to smartphone-based positioning.

2.4.2.a GNSS Time Generation

The generation of GNSS time is defined by the equation 2.2. The following expression is given in nanoseconds. The internal hardware clock and the bias to the true GPS time is provided by Android. We also have to take into consideration that all GNSS constellations have different time of reference. All reference time are calculated based on the GPS time. Once the GPS time is calculated, adjustments are made to fit the constellation of the tracked satellite.

$$t_{GPS} = \mathit{TimeNanos} - (\mathit{FullBiasNanos} + \mathit{BiasNanos}) - \mathit{InterSystemsBias} \tag{2.2}$$

where the italic text represents the field of Android raw measurements. All parameters are retrieved from the GNSSClock Android Class.

2.4.2.b Pseudoranges Computation

Pseudoranges represents the pseudo distance between the receiver and the transmitting satellite. This distance can be affected by diverse biases in the propagation channel. Pseudoranges are calculated by differentiating the transmitted time (t_{Tx}) and the received time (t_{Rx}) at the speed of light c. The pseudorange is then computed as:

$$\rho = (t_{Rx} - t_{Tx}) \times c \tag{2.3}$$

where:

- t_{Tx} is the *ReceivedSvTimeNanos* in nanosecond.
- t_{Rx} is computed as follows:

$$t_{Rx} = \mathit{TimeOffsetNanos} + t_{GPS}(1) \tag{2.4}$$

- *TimeOffsetNanos* represents the offset between the GPS TOW (Time of Week) and the current measurement epoch.

In the smartphones' embedded GNSS receiver case, pseudoranges generation methods assume a common reception time. This assumption implies that $t_{GPS}(1)$ indicates that the first value of *FullBiasNanos* and *BiasNanos* are used in the computation until the phone is turned off. t_{Tx} and t_{Rx} must be in the same time reference, usually in GPS time.

2.4.2.c Carrier-Phase Measurements

Carrier-phase measurements are directly provided by the Android API under the name *getAccumulatedDeltaRangeMeters* in meters. A set of flags have been created to detect cycle slips and are provided under the name *AccumulatedDeltaRangeState*. These flags insures a sanity check from the obtained measurements.

2.4.2.d Doppler-Shift

The Doppler shift, resulting from satellite movement and/r receiver movement and/or frequency offset, can be computed as follows:

$$DopplerShift = PseudorangeRateMeterperSecond \times \frac{Carrier\ FrequencyHz}{c} \tag{2.5}$$

2.4.3 Smartphone GNSS Data Collection

Nowadays, numerous Android smartphones with accurate GNSS chipset are available on the market. A non-exhaustive list of Android smartphone GNSS capabilities can be found in [21]. Following the release of GNSS raw data measurements, Google released a Matlab based program, GNSSAnalysis [11], in order to analyze and evaluate GNSS raw data measurements. The program and the open source code is free and available for everyone. In [45], a detailed explanation is provided on how to use and interpret raw data parameters using Google's source code. The collection of raw data measurements can be made directly from any smartphones via an Android application. Multiple applications exists allowing users to record smartphone's data and features (refer to section: 2.2). Concerning GNSS data, two format types are used: Raw data measurements and Receiver Independent Exchange Format (RINEX).

2.4.3.a Raw Measurements Format

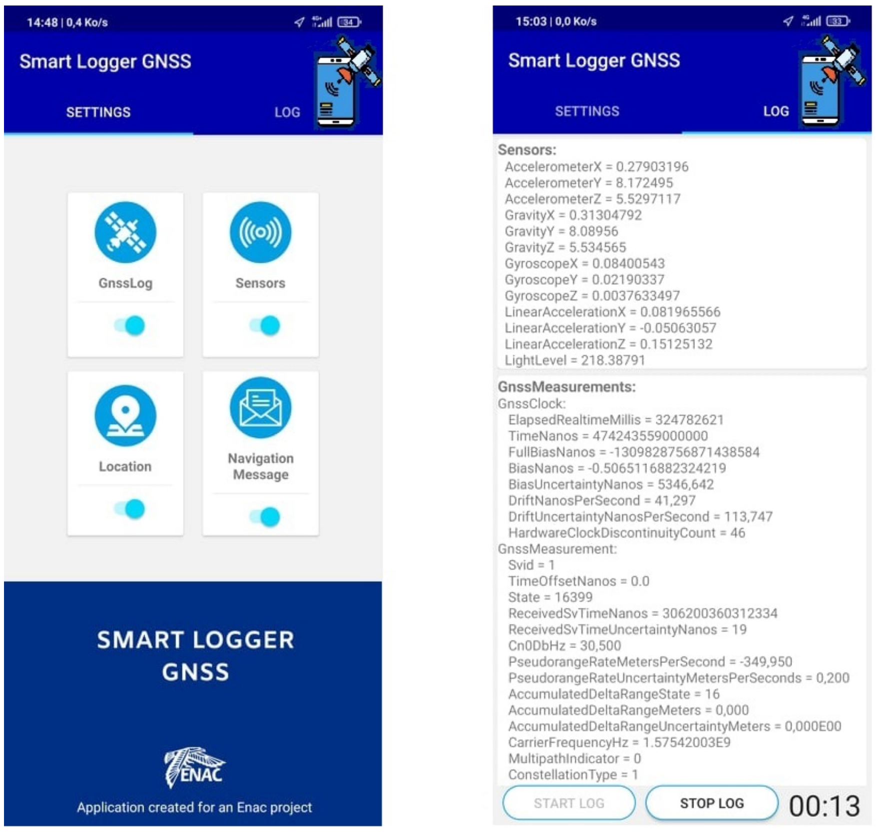
Raw measurements is a simple format that logs and records each individual smartphone data measurements (exposed in table 2.10 and 2.11) for every epoch at 1 Hz frequency. This data is then stored in a *.csv* file. This file configuration is straightforward and easy to use in any post-processing program. Furthermore, since every single chipset output is recorded by this logging format, it is considered the most adequate recording format. Google released an application called, *GNSSLogger*, that is capable of recording smartphone's GNSS raw data measurements in a *.csv* file. Other commercial applications can be found directly on the Google Play store.

2.4.3.b RINEX Format

By definition, the RINEX format is a well-known file format exploiting GNSS data independently of the receiver. The file format is universally used by the scientific community for logging and processing GNSS data. Unfortunately, the strict formating policy of this recording format do not allow to report all GNSS information given by the smartphone. The newest version (v3.1.0.0) of the *GNSSLogger* application give users the choice to record or convert smartphone's data using the universal RINEX format. *Geo++ RINEX Logger* application also allow user to log GNSS data from their smartphone exclusively into a RINEX file.

2.4.3.c *SmartLogger*, an ENAC Android APP

For the purpose of this research, an Android application has been developed. The goal was to better understand and apprehend the functionality and usage of the Android API. Furthermore, we wanted to be able to record any smartphone's data and features using a single application only. Indeed, this ENAC's Android application allows users to record GNSS raw data measurements from the *android.location* API, plus recording additional sensors data (including but not limited to: IMU, barometer and Light parameter measurements) synchronously. This application has been developed by Laetitia KESSAS as a part of an internship, supervised by Thomas VERHEYDE and Antoine BLAIS. The application is in the beta development phase and is ambitioned to be soon published on the Google Play Store for free.

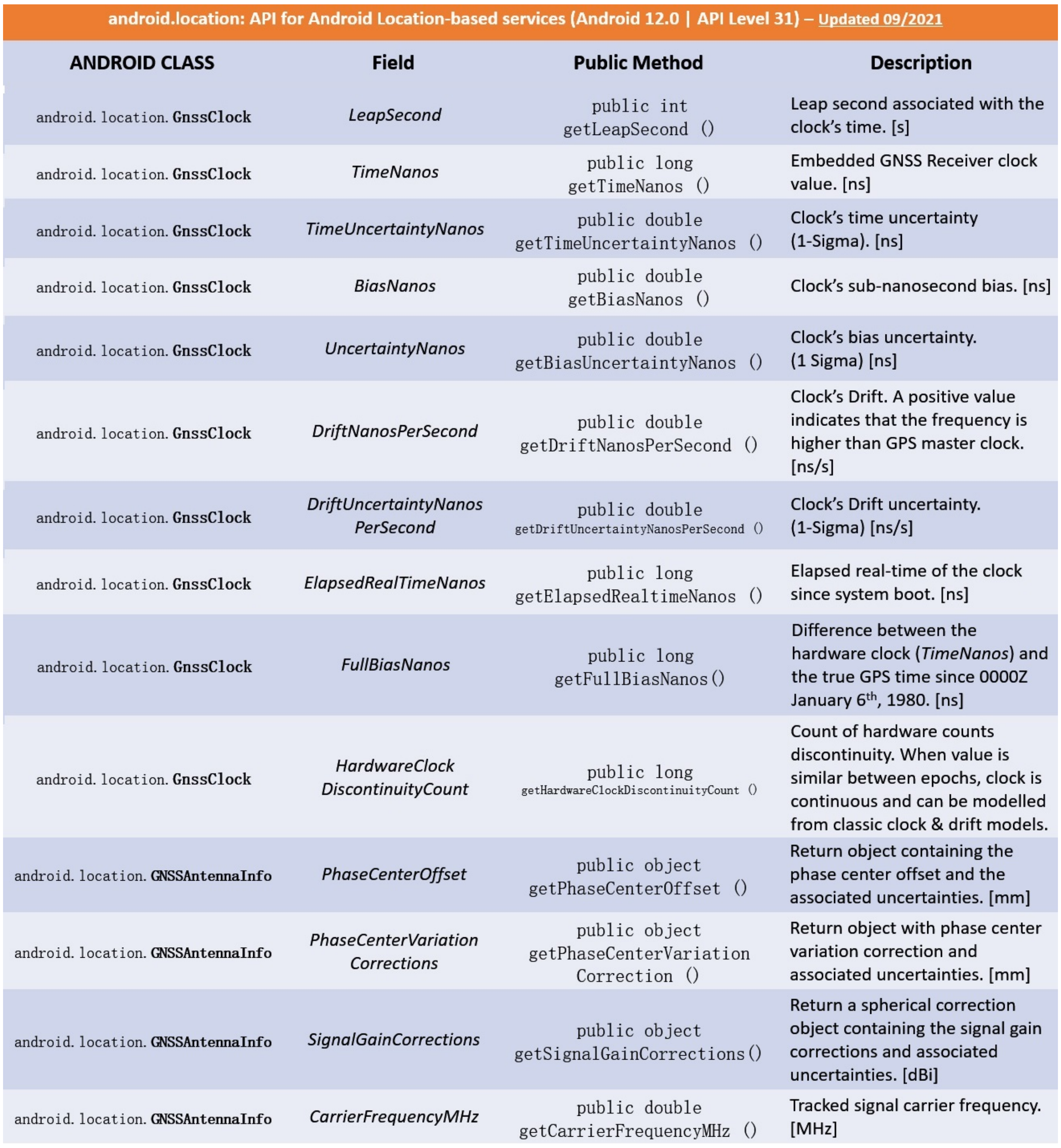


```
Sensors:
  AccelerometerX = 0.27903196
  AccelerometerY = 8.172495
  AccelerometerZ = 5.5297117
  GravityX = 0.31304792
  GravityY = 8.08956
  GravityZ = 5.534565
  GyroscopeX = 0.08400543
  GyroscopeY = 0.02190337
  GyroscopeZ = 0.0037633497
  LinearAccelerationX = 0.081965566
  LinearAccelerationY = -0.05063057
  LinearAccelerationZ = 0.15125132
  LightLevel = 218.38791
GnssMeasurements:
GnssClock:
   ElapsedRealtimeMillis = 324782621
   TimeNanos = 474243559000000
   FullBiasNanos = -1309828756871438584
   BiasNanos = -0.5065116882324219
   BiasUncertaintyNanos = 5346,642
   DriftNanosPerSecond = 41,297
   DriftUncertaintyNanosPerSecond = 113,747
   HardwareClockDiscontinuityCount = 46
GnssMeasurement:
   Svid = 1
   TimeOffsetNanos = 0.0
   State = 16399
   ReceivedSvTimeNanos = 306200360312334
   ReceivedSvTimeUncertaintyNanos = 19
   Cn0DbHz = 30,500
   PseudorangeRateMetersPerSecond = -349,950
   PseudorangeRateUncertaintyMetersPerSeconds = 0,200
   AccumulatedDeltaRangeState = 16
   AccumulatedDeltaRangeMeters = 0,000
   AccumulatedDeltaRangeUncertaintyMeters = 0,000E00
   CarrierFrequencyHz = 1.57542003E9
   MultipathIndicator = 0
   ConstellationType = 1
```

Figure 2.9: SmartLogger Android Application - Settings and Logging Interfaces - Github Project Link for *.apk* Beta Application Download in [8]

2.5 Chapter Conclusions

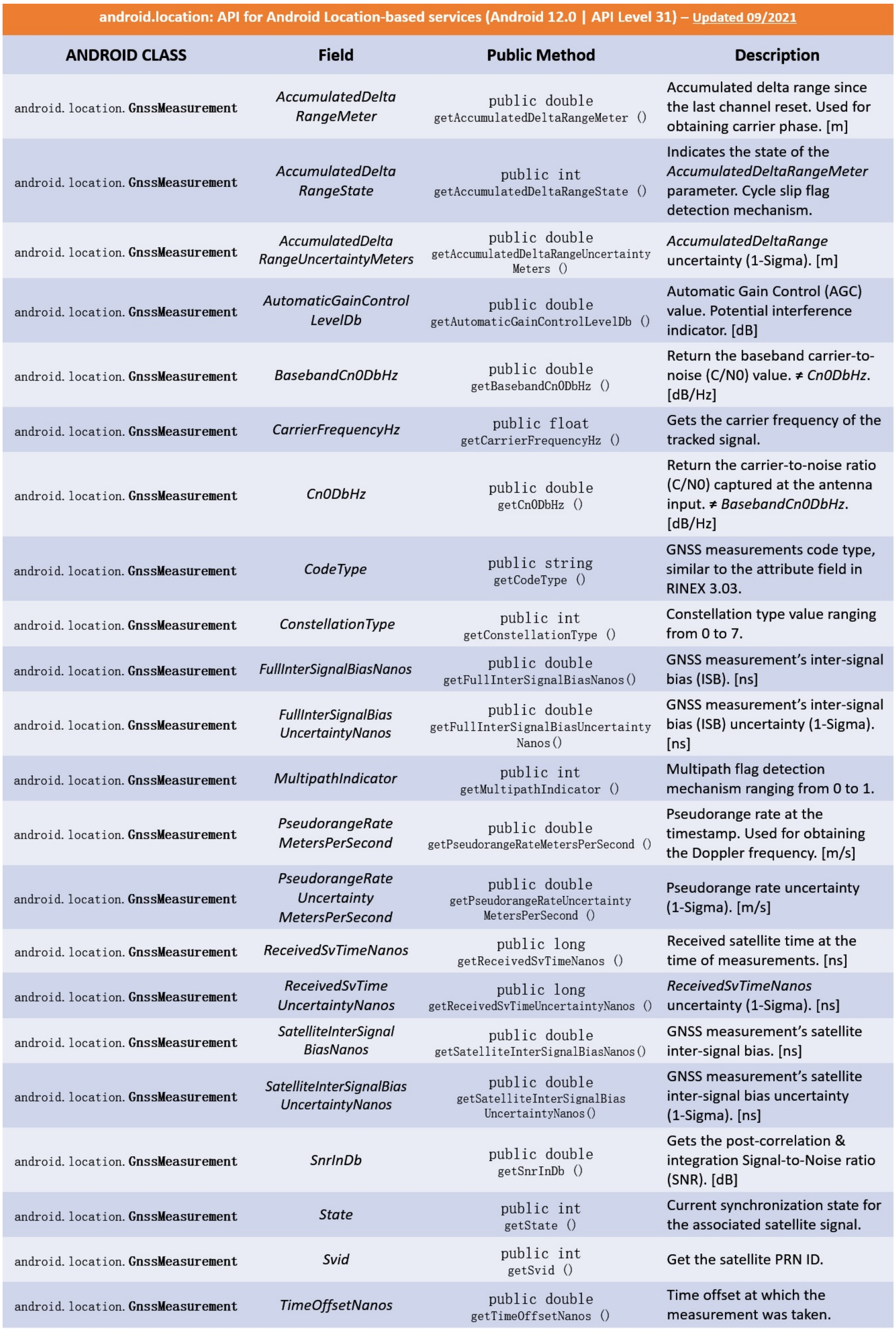
This chapter depicted the current situation of the low-cost GNSS receivers market. It has been noted that low-cost receivers quality and accuracy is improving with time. For the scope of this research, the growing market of embedded smartphone GNSS receivers could potentially unlock an opportunity to create a wide collaborative network. Thanks to the recent accessibility of GNSS raw data measurements on smartphones, GNSS parameters are easily record-able and could be easily transferable or exchanged. Smartphones location capabilities and accuracy are improving, and sub-metric positioning is envisioned. However, cheap hardware components in smartphone (such as the antenna) prohibit the easy transition to high-end geodetic-like smartphone positioning.



android.location: API for Android Location-based services (Android 12.0 | API Level 31) – Updated 09/2021

ANDROID CLASS	Field	Public Method	Description
android.location.**GnssClock**	*LeapSecond*	public int getLeapSecond ()	Leap second associated with the clock's time. [s]
android.location.**GnssClock**	*TimeNanos*	public long getTimeNanos ()	Embedded GNSS Receiver clock value. [ns]
android.location.**GnssClock**	*TimeUncertaintyNanos*	public double getTimeUncertaintyNanos ()	Clock's time uncertainty (1-Sigma). [ns]
android.location.**GnssClock**	*BiasNanos*	public double getBiasNanos ()	Clock's sub-nanosecond bias. [ns]
android.location.**GnssClock**	*UncertaintyNanos*	public double getBiasUncertaintyNanos ()	Clock's bias uncertainty. (1 Sigma) [ns]
android.location.**GnssClock**	*DriftNanosPerSecond*	public double getDriftNanosPerSecond ()	Clock's Drift. A positive value indicates that the frequency is higher than GPS master clock. [ns/s]
android.location.**GnssClock**	*DriftUncertaintyNanos PerSecond*	public double getDriftUncertaintyNanosPerSecond ()	Clock's Drift uncertainty. (1-Sigma) [ns/s]
android.location.**GnssClock**	*ElapsedRealTimeNanos*	public long getElapsedRealtimeNanos ()	Elapsed real-time of the clock since system boot. [ns]
android.location.**GnssClock**	*FullBiasNanos*	public long getFullBiasNanos()	Difference between the hardware clock (*TimeNanos*) and the true GPS time since 0000Z January 6th, 1980. [ns]
android.location.**GnssClock**	*HardwareClock DiscontinuityCount*	public long getHardwareClockDiscontinuityCount ()	Count of hardware counts discontinuity. When value is similar between epochs, clock is continuous and can be modelled from classic clock & drift models.
android.location.**GNSSAntennaInfo**	*PhaseCenterOffset*	public object getPhaseCenterOffset ()	Return object containing the phase center offset and the associated uncertainties. [mm]
android.location.**GNSSAntennaInfo**	*PhaseCenterVariation Corrections*	public object getPhaseCenterVariation Correction ()	Return object with phase center variation correction and associated uncertainties. [mm]
android.location.**GNSSAntennaInfo**	*SignalGainCorrections*	public object getSignalGainCorrections()	Return a spherical correction object containing the signal gain corrections and associated uncertainties. [dBi]
android.location.**GNSSAntennaInfo**	*CarrierFrequencyMHz*	public double getCarrierFrequencyMHz ()	Tracked signal carrier frequency. [MHz]

Figure 2.10: Android Clock and Antenna Data Measurements Description [9], [10].



android.location: API for Android Location-based services (Android 12.0 | API Level 31) – Updated 09/2021

ANDROID CLASS	Field	Public Method	Description
android.location.**GnssMeasurement**	*AccumulatedDeltaRangeMeter*	`public double getAccumulatedDeltaRangeMeter ()`	Accumulated delta range since the last channel reset. Used for obtaining carrier phase. [m]
android.location.**GnssMeasurement**	*AccumulatedDeltaRangeState*	`public int getAccumulatedDeltaRangeState ()`	Indicates the state of the *AccumulatedDeltaRangeMeter* parameter. Cycle slip flag detection mechanism.
android.location.**GnssMeasurement**	*AccumulatedDeltaRangeUncertaintyMeters*	`public double getAccumulatedDeltaRangeUncertaintyMeters ()`	*AccumulatedDeltaRange* uncertainty (1-Sigma). [m]
android.location.**GnssMeasurement**	*AutomaticGainControlLevelDb*	`public double getAutomaticGainControlLevelDb ()`	Automatic Gain Control (AGC) value. Potential interference indicator. [dB]
android.location.**GnssMeasurement**	*BasebandCn0DbHz*	`public double getBasebandCn0DbHz ()`	Return the baseband carrier-to-noise (C/N0) value. ≠ *Cn0DbHz*. [dB/Hz]
android.location.**GnssMeasurement**	*CarrierFrequencyHz*	`public float getCarrierFrequencyHz ()`	Gets the carrier frequency of the tracked signal.
android.location.**GnssMeasurement**	*Cn0DbHz*	`public double getCn0DbHz ()`	Return the carrier-to-noise ratio (C/N0) captured at the antenna input. ≠ *BasebandCn0DbHz*. [dB/Hz]
android.location.**GnssMeasurement**	*CodeType*	`public string getCodeType ()`	GNSS measurements code type, similar to the attribute field in RINEX 3.03.
android.location.**GnssMeasurement**	*ConstellationType*	`public int getConstellationType ()`	Constellation type value ranging from 0 to 7.
android.location.**GnssMeasurement**	*FullInterSignalBiasNanos*	`public double getFullInterSignalBiasNanos()`	GNSS measurement's inter-signal bias (ISB). [ns]
android.location.**GnssMeasurement**	*FullInterSignalBiasUncertaintyNanos*	`public double getFullInterSignalBiasUncertaintyNanos()`	GNSS measurement's inter-signal bias (ISB) uncertainty (1-Sigma). [ns]
android.location.**GnssMeasurement**	*MultipathIndicator*	`public int getMultipathIndicator ()`	Multipath flag detection mechanism ranging from 0 to 1.
android.location.**GnssMeasurement**	*PseudorangeRateMetersPerSecond*	`public double getPseudorangeRateMetersPerSecond ()`	Pseudorange rate at the timestamp. Used for obtaining the Doppler frequency. [m/s]
android.location.**GnssMeasurement**	*PseudorangeRateUncertaintyMetersPerSecond*	`public double getPseudorangeRateUncertaintyMetersPerSecond ()`	Pseudorange rate uncertainty (1-Sigma). [m/s]
android.location.**GnssMeasurement**	*ReceivedSvTimeNanos*	`public long getReceivedSvTimeNanos ()`	Received satellite time at the time of measurements. [ns]
android.location.**GnssMeasurement**	*ReceivedSvTimeUncertaintyNanos*	`public long getReceivedSvTimeUncertaintyNanos ()`	*ReceivedSvTimeNanos* uncertainty (1-Sigma). [ns]
android.location.**GnssMeasurement**	*SatelliteInterSignalBiasNanos*	`public double getSatelliteInterSignalBiasNanos()`	GNSS measurement's satellite inter-signal bias. [ns]
android.location.**GnssMeasurement**	*SatelliteInterSignalBiasUncertaintyNanos*	`public double getSatelliteInterSignalBiasUncertaintyNanos()`	GNSS measurement's satellite inter-signal bias uncertainty (1-Sigma). [ns]
android.location.**GnssMeasurement**	*SnrInDb*	`public double getSnrInDb ()`	Gets the post-correlation & integration Signal-to-Noise ratio (SNR). [dB]
android.location.**GnssMeasurement**	*State*	`public int getState ()`	Current synchronization state for the associated satellite signal.
android.location.**GnssMeasurement**	*Svid*	`public int getSvid ()`	Get the satellite PRN ID.
android.location.**GnssMeasurement**	*TimeOffsetNanos*	`public double getTimeOffsetNanos ()`	Time offset at which the measurement was taken.

Figure 2.11: Android GNSS Raw Data Measurements Description [11].

3 Smartphone Positioning Techniques

Contents

Smartphones contain the most recent technology embedded in their hardware due to a ferocious competition between manufacturers. This rivalry pushes subcontractors such as GNSS chip manufacturers to create innovative GNSS embedded products. The objective of the GNSS community is to be able to achieve centimeter level precision in mass-market devices such as smartphones. This goal will be soon obtained thanks to the new generation of GNSS chips and their improved characteristics (explored in chapter 2). Contrastingly, it is well understood that smartphones are mainly used for communication and entertainment purposes. Components, such as antenna and SoC processor, will never be prioritized for positioning purposes and will impact the overall positioning performance. Since the release of Android GNSS raw data measurements, efforts have been made toward high-precision smartphone-based positioning performance. Inspired by the GNSS literature, advanced positioning algorithm have been implemented and studied for smartphone-based positioning in order to achieve sub-metric position solutions.

This chapter will describe various smartphone-based positioning algorithm and performance. Firstly, a section will review the state of the art for Android smartphone positioning techniques and algorithms for past, current and future devices. The next section will present the data collection campaign organized for studying smartphones behaviors in urban environment. Furthermore, in anticipation to our research work, we conducted specific collaborative scenarios that will be detailed here. Finally, a study of smartphone main positioning parameters will be made for nominal and urban conditions. This study will also be the occasion to assess the quality of Android GNSS raw data measurements in constrained conditions.

3.1 State of the Art: Smartphone Positioning

A complete state of the art on smartphone positioning techniques is explored in this section. The objective of this section is to present the various methods employed for smartphone-based positioning. Historically, AGNSS technique was adapted to smartphone devices for enhancing their positioning performances. Nowadays, hybrid positioning methods are used for determining the user position. In the meantime, the adaptation of well-known GNSS advanced algorithms to mobile devices are currently under development. This analysis will be divided between conventional and advanced smartphone-based algorithms.

3.1.1 Conventional Methods

A complete overview of current smartphone positioning algorithms is given here. It turns out that AGNSS fit perfectly the need and constraints linked with smartphone-based positioning. This method was one of the first smartphone positioning algorithms to be adopted on those devices. Later on, Google improved smartphone localization capabilities by hybridizing multiple sensors and data that can be retrieved on modern smartphone. It is today considered by most, the reference method for smartphone-based positioning.

3.1.1.a Assisted GNSS (AGNSS)

AGNSS is a method that improves standard GNSS capabilities of a GNSS receiver. Satellite data are transmitted via an alternative communication channel, different than the GNSS signal propagation channel. Those transmitted information allow the receiving user to faster compute his location. Aiding data given by the AGNSS protocol encompasses acquisition assistance, almanac and ephemeris [12]. AGNSS assists GNSS receivers in obtaining a position estimation more efficiently by reducing Time To First Fix (TTFF) from approximately 1 minute to 1 second [46]. Furthermore, an assisted GNSS receiver increases its sensitivity enabling it to acquire signals at much lower signal strength.
This method has been exported on mobile smartphones mainly for the three following reasons. First of all, smartphone devices are by definition always connected to a cellular network. Thus, providing an ideal communication channel for transmitting and receiving AGNSS data. A secure protocol has been put in place for transmitting satellite data to multiple users in real-time. The Secure User Plane Location (SUPL) protocol is an IP based protocol for AGNSS. A smartphone embedded GNSS receiver uses SUPL server to download up-to-date almanac and/or satellite ephemeris data. The most used and trusted SUPL server today is the one operated by Google. Secondly, the increase in embedded receiver sensitivity allowed by this method, makes acquiring weak signals in constrained environment much easier. Finally, this method also allows smartphone device to reduce

their energy consumption by saving the need to decode navigation data from received satellite signals that are already obtained via the AGNSS network.

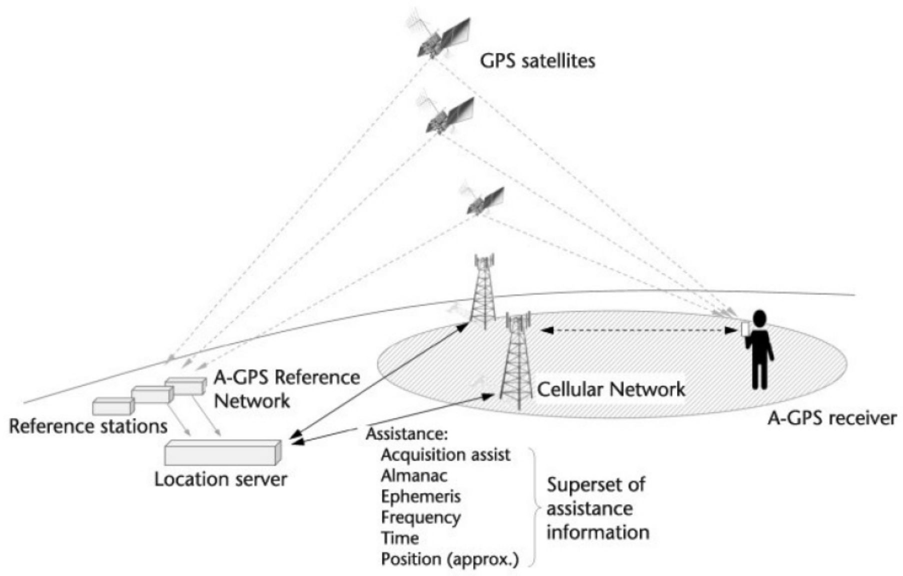


Figure 3.1: AGNSS Graphical System Representation. Figure extracted from [12].

3.1.1.b Fused Location Provider (FLP)

The Fused Location Provider (FLP) positioning service is only made available on Android device via a specific location API. Google describes this service as an intelligent method that combines different signals for producing the most accurate location solution. This algorithm is often described as a tight integration design between cellular, Wi-Fi, Bluetooth and the IMU data. In the future, it is expected that FLP service will be improved with the arrival of 5G technology and the rapid development of WiFi RTT protocol [47]. Originally, the FLP engine was defined as the combination and improvement of two API services: the GPS Location Provider (GLP) and the Network Location Provider (NLP) [48]. FLP output positions are intended to be the best available positioning solution obtainable by a given Android device.
However, the black-box processes used by Android for obtaining this solution while combining GNSS, cellular network and sensors informations, makes them ambiguous and unreliable for scientific analysis. Indeed, Google describes the conception of FLP position estimation as a complex process that automatically changes the appropriate system settings to what was requested by the user. The main motivations for Android to keep this method as a ”black box” process for the users and developers are privacy and limitation of power consumption. By keeping FLP processes unknown they ensure the privacy protection of their users while accepting in-application requests for this API. They also keep the upper hand on the trade-off that can be applied between the device power consumptions and accuracy. On the other hand, Android FLP positioning solutions provide a good reference for intrinsic quality of smartphones positioning capabilities. This position solution can be accessible by every Android smartphone that will be tested later on during this research study, and Android FLP position estimation will be used for reference purposes.

3.1.2 Advanced Techniques

Advanced smartphone positioning techniques are GNSS positioning methods that differentiate themself from the previously presented classical smartphone-based positioning solutions. These advanced method are derived from well-known GNSS algorithm, adapted to smartphone devices. Advanced techniques can also be referred as high-precision positioning algorithms. High precision positioning is defined as a technique capable of positioning a user/receiver with centimetric precision level. Those methods employ a combination of carrier-phase and pseudoranges GNSS measurements in order to achieve accurate results. Advanced positioning algorithms have been developed for application that requires higher accuracy solutions, such as surveying. The next subsections will present the most known advanced algorithms applied in smartphone-based positioning, followed by a brief presentation of future algorithms being developed for answering smartphone positioning limitations.

3.1.2.a PPP and RTK

The two most used advanced positioning algorithms in the GNSS domain today are PPP and RTK. These two correction-oriented type methods take advantage of the measurements that can be made on the carrier wave of the GNSS signal. Carrier-based positioning algorithms rely on the physical proprieties of the carrier wave allowing for precise localization estimation. However, measurements of the carrier phase is ambiguous since the number of cycles is unknown. Resolving these carrier phase ambiguities is key and a prerequisite for both PPP and RTK in order to achieve precise positioning. The description of the two methods is described below:

- **Real-Time Kinematic (RTK)**
  RTK is a high-precision Differential GNSS (DGNSS) positioning technique. This method is articulated around the use of carrier-phase measurements from a rover station and the transmission in real-time of corrections coming from a nearby fixed base station. Measurements differentiation between the rover and the base station allows for mitigating most propagation errors. The base station is considered as a reference GNSS receiver with an established location. This reference station, then, broadcast its position and the code and carrier measurements of all in-view satellites. This process will allow the rover to fix phase ambiguity and thus to determine a precise localization estimation.

- **Precise Point Positioning (PPP)**
  PPP positioning technique is based on precise clock and orbit corrections sent to the rover. Corrections are generated in real-time from a network of GNSS reference stations located all over the world. As an example, the International GNSS Service (IGS) organization ensures open access, high-quality GNSS data products daily with a network of more than 500 stations. Precise PPP position solutions are obtained through the combination of undifferentiated GNSS observables with precise satellite orbits, positions and clocks.

RTK Vs PPP
This direct comparison between these two techniques will allow to better draw the advantages and disadvantages for both methods. RTK provides at the moment the highest level of accuracy at the local level while the PPP generated solution are more versatile on a global scale. However, most RTK solutions requires additional hardware which is not adapted for smartphone users. Moreover, PPP algorithms requires long convergence time and are also not compliant with smartphone standards.

Nowadays with the arrival of new smartphone embedded technologies and the permanent emulation of the scientific community for higher positioning performance of mass-market devices, the implementation of advanced positioning techniques in the smartphone domain has been extensively studied [32] [33] [4]. Modern smartphones embedded GNSS receivers are able to reach very impressive standards for both static or kinematic positioning, opening the doors to an enormous quantity of application domains and research fields. The release of carrier phase measurements and dual-frequency chipset on Android smartphones accelerated the efforts made toward the integration of advanced GNSS algorithms on Android smartphone device.
Current works mostly focus on the feasibility for integrating PPP and RTK algorithms on smartphone platform [49]. Post-precessing tests have been carried out testing smartphones ability to perform under different environment [50]. Results from the literature show that smartphone raw data measurements are compatible with the development of advanced GNSS algorithms. However, it is pointed out that smartphone-based positioning constraints cannot be mitigated by the application of such advanced algorithms. Cycle slip events tends to be frequent on Android portable devices, [51], which makes the procedure for carrier ambiguity solving challenging. Smartphone hardware components and usual operating environments produces drawbacks (such as: multipath occurrences, signal blockage, Non-Line of Sight (NLOS) and more) that do not comply with advanced GNSS algorithms operation requirements.

3.1.2.b Other Smartphone-based Positioning Research Fields

Additional research fields have been explored in the domain of smartphone-based positioning. The following research topics represent leads that have been explored toward higher-precision positioning capabilities for smartphone.

- **Visual Positioning System (VPS)**
  During the I/O Tech Conference organized by Google in 2018, a new technology for pedestrian urban navigation called Visual Positioning System (VPS) was introduced [52]. The VPS algorithm uses smartphone's camera in order to recognize places and buildings characteristics that surround the users during urban navigation. Such characteristics are compared and correlated with Google's image database built within the *Street View* service. The embedded algorithm, then, uses an hybridization process between the a priori position given by the GNSS chip and the image point features analyzed by the camera. Thus, allowing smartphone users to follow navigation instructions on his phone thanks to an Augmented Reality display. This feature has been integrated to Google Map and is accessible to most Android users under the name "*Live View*". Other research works utilize smartphone's camera for implementing VPS techniques but strictly for answering indoor positioning problematics [53] which is outside the scope of this research project.

- **3D City Mapping Aided Positioning Algorithms**
  One of the main constraints of smartphone-based positioning is that most smartphone activities are achieved in urban and sub-urban areas. The regular environments in which portable devices are putted in, induce multiple constraints that need to be addressed. Deep urban navigation has become one of the toughest challenge for the GNSS community. This type of environment is favorable to the generation of errors due to NLOS tracked signals and multipath events.
  The main 3D Mapping Aided (3DMA) methods, exploitable by smartphone devices, are presented below. All the following methods requires the access to a detailed 3D city model of the user city.

  Shadow Matching
  Shadow matching technique was firstly introduced by Groves in [54]. The principle of the presented method is that at a specific epoch in time, each visible satellite signals can be predicted given a rough estimation of the user street position. A set of visible signals is compared to the predicted one in order to incrementally estimate user final position. Research work implemented this method on Android smartphone device finding out that cross-street positioning error can be significantly improved [55].

  3DMA Pseudoranges Matching
  3DMA pseudoranges matching is a similar technique to shadow matching except that we are here predicting expected satellite pseudoranges. In order to do so, a grid is created around a first rough position estimate. Then, at each point of the grid a comparison between the observed pseudoranges and the predicted ones, including NLOS signals, is made. A correlation statistical analysis delivers the final position solution. This solution was explored in [56] and has not yet been tested for smartphone-based positioning in constrained environment.

  Google's Approach on 3DMA
  The above 3DMA method presented a unique flaw for smartphone-based positioning. The computational power required for exploiting 3D city models and generating ray tracing processes cannot be managed by the phone internal hardware. It is in this optic that Google decided to release their own solution to smartphone urban positioning. One of Google's advantage is the abundance of 3D city models used for other services, on Google Maps and Google Earth, covering more than 4000 cities worldwide. The second advantage from them, is the availability of the processing power needed for delivering such solutions, on dedicated Google servers. By using similar 3DMA techniques as the one presented above, Android directly sends corrections to the embedded smartphone GNSS chipset plus integrates 3DMA correction to their FLP service (see 3.1.1.b). During their testing campaign, Google claimed that wrong-side-of-street events have dropped by 75% [57]. This feature was launched on Google Pixel devices (4a and 5) in December 2020. However, it seems that this method was not adapted on other Android devices.

3.2 Data Collection Campaign

As presented in chapter 2, smartphone embedded GNSS receivers have a unique architecture trade-off design compared to Commercial Off-the-Shelf (COTS) GNSS receivers. Advanced Smartphone-based positioning became a trend when Google announced the release of Android GNSS raw data measurements.
Multiple studies were made in the smartphone positioning domain aiming at applying advanced GNSS-based positioning techniques, as presented in section 3.1.2. Most studies drawn their conclusions based on one smartphone brand and model in optimal open-sky conditions. Although, most smartphone-based positioning activities are achieved in urban and sub-urban areas. In urban conditions, signals are degraded from disruptive multipath and NLOS interferences that represent the main challenge associated with urban positioning. Apprehending those difficulties become even more challenging when using a low-grade smartphone antenna (in section 2.3.2) and coping with the embedded receiver black-box processes (see: 2.3.1). Characterizing different smartphone and embedded chipset models in urban environment remains an unaddressed challenge.

Part of this research study aims at addressing smartphone-based urban positioning and at characterizing smartphones positioning parameters. A data collection campaign has been orchestrated in the interest of outlining multiple smartphone behaviors in various urban scenarios. This data collection has been conducted with the active support of ENAC and the French Space Agency (CNES). Both entities provided hardware equipments, personnel, and vehicles for the success of this campaign. The following list underlines the goals and highlights the contributions for this data collection campaign.

- Validate a reliable recording techniques for Android GNSS raw data measurements
- Characterize multiple smartphone and embedded chipset brands and models
- Assess smartphones' behaviors in various conditions (urban/open-sky & static/dynamic)
- Assess and analysis the impact of smartphone's embedded antenna on recorded GNSS raw data measurements using an unique asset.
- Create collaborative scenarios for post-processing analysis
- Compare smartphone and low-cost COTS GNSS receivers positioning performance in "real-life" environments

The first subsection will present the experimentation protocol put in place for achieving a successful data collection campaign. The selected group of tested smartphones will be presented. Secondly, the presentation of vehicles and equipments used for this data collection campaign will be exposed. Finally, pre-selected collaborative scenarios will be introduced in anticipation for the implementation of cooperative techniques later on during this research work.

3.2.1 Experimentation Protocols

The success of any data collection campaign comes from the redaction of strict protocols for ensuring the correct replayability of the collected data during post-processing and analysis cycles. Therefore, this data collection campaign took place in urban areas around Toulouse in France, for a total duration of 2 hours and 10 minutes. Two research

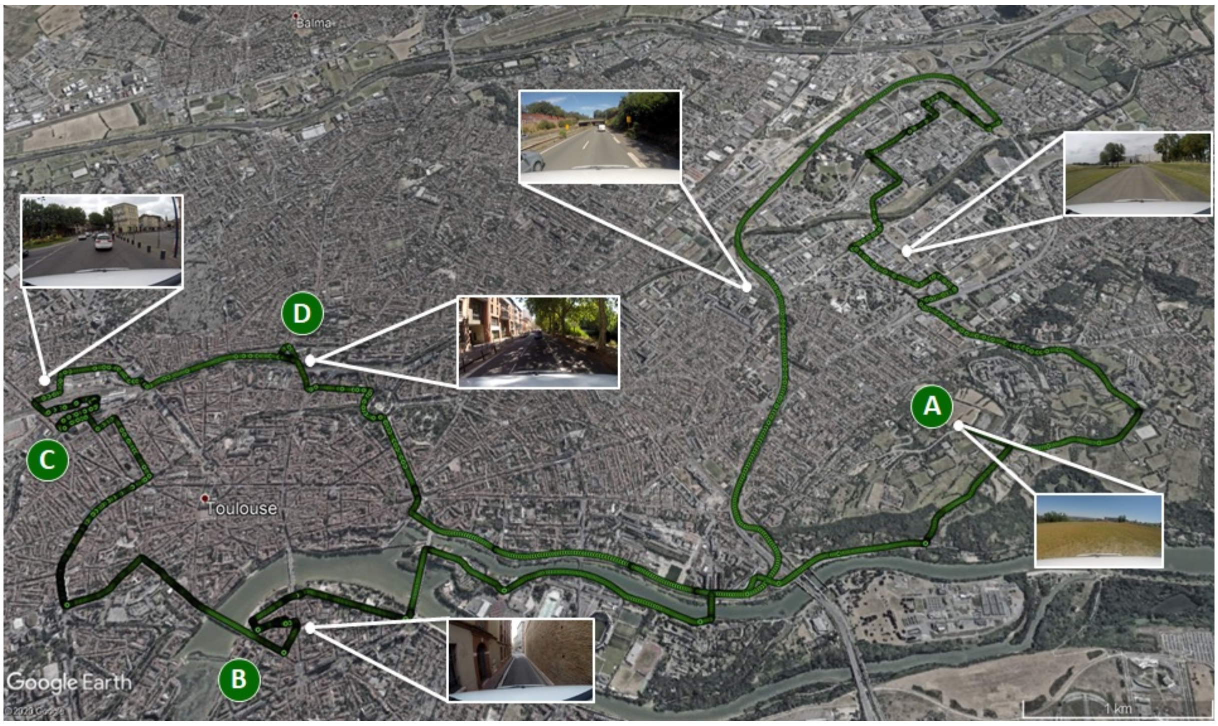


Figure 3.2: Data Collection Campaign Trajectory. Letters and encapsulated photos depict the locations and conditions encountered for the different collaborative scenarios.

vehicles were used during this experimentation. The main benefit for using two cars was to simulate two different users in a collaborative setting. Both vehicles embarked high-end GNSS receivers for reference purposes. Tested smartphones were splited into two groups and placed on the roof of each vehicle. A total of 7 smartphones have been tested during this data collection campaign. The selected array of smartphones represent the variety of brands and models found in today's market. Finally, the chosen campaign trajectory has been selected for its versatility in terms of reception conditions and constraints that can be found in modern city centers. Along this trajectory, pictured in figure 3.2, collaborative scenarios have been created and are depicted by letters on the above mentioned figure.

3.2.1.a Smartphones Selection

During this data collection campaign, the array of selected smartphones needed to best represent the diversity of Android smartphone brands and models available on the market. All the 7 tested Android devices are shown in picture 3.3. Moreover, most SoC and smartphone embedded GNSS receiver manufacturers are represented by our analysis (Qualcomm, HiSilicon and Broadcom). All smartphones were running Android Q (10.0) OS. Similarly, all of them were multi-constellation and multi-frequency (except for the Google Pixel 3 that was single-frequency).

Although, one device presents a particularity that need to be explained. The Xiaomi Mi 8 devices was the world's first dual-frequency and multi-constellation GNSS capable smartphone. In order to obtain such achievement, Xiaomi decided to fit two SoC units to their device. The phone was mainly developed around the Qualcomm Snapdragon technology. Whereas, the Broadcom BCM47755 chipset was fitted exclusively for its

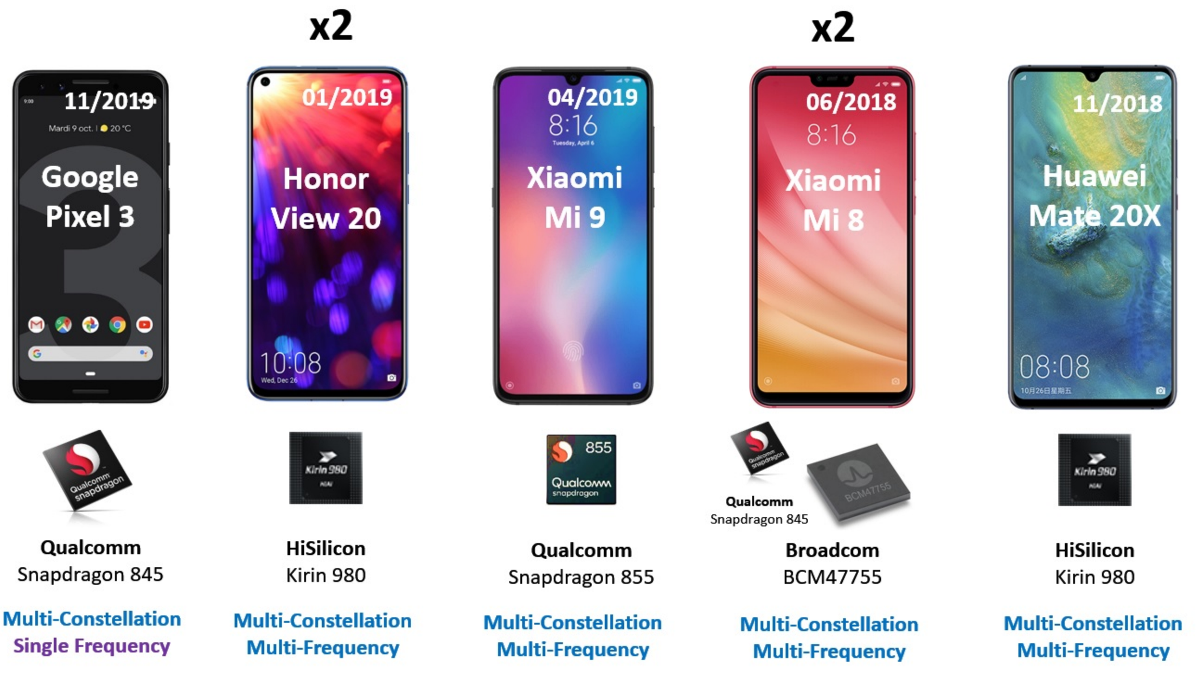


Figure 3.3: Smartphones Selection Tested during the Data Collection Campaign

positioning capabilities. Thus, the Xiaomi Mi 8 obtained the title of world's first dual-frequency and multi-constellation smartphone making the device the benchmark reference in the smartphone-based positioning domain.

3.2.1.b Vehicles Configuration Setup

As part of the data collection campaign, two research vehicles were used. These vehicles allowed to carry heavy equipments but also to carry collaborative scenarios in low-dynamic urban scenarios. During the entire collect, both cars were following each other. Static and dynamic scenarios were performed in both urban and sub-urban environment, thus replicating best real life events encountered by smartphone users. The sketch shown in figure 3.4 represents the overall configuration of both cars. This plan is supported by the picture below, 3.5, that shows the rooftop setup on both experimental vehicles. Both figures are linked by the number identifying each car.

For reference and comparison purposes, multiple COTS hybridized GNSS receivers were fitted inside both vehicles. Two high-end COTS GNSS receivers, *NovAtel SPAN Inertial Navigation System*, were assigned to each car and used as positioning and navigation reference solution. Additionally, three Ublox F9P low-cost GNSS receivers have been fitted inside the vehicles (as shown on figure 3.4) and were used for comparison purposes. Other equipments present in the car were either used as backup receivers or were exploited for other research purposes, outside the scope of this thesis.
Smartphones were split into two groups, in function of their brand and model, and were attached to the roof of both cars. Figure 3.5 shows the exact layout configuration. All smartphones were recording Android GNSS raw data measurements (as a .csv file) using the *GNSSLogger* application [58]. Moreover, all sensors information available were recorded using the *AndroSensor* application at minimum frequency of 1 Hz up to 10 Hz for specific phones. One of the unique feature of this smartphone data collection

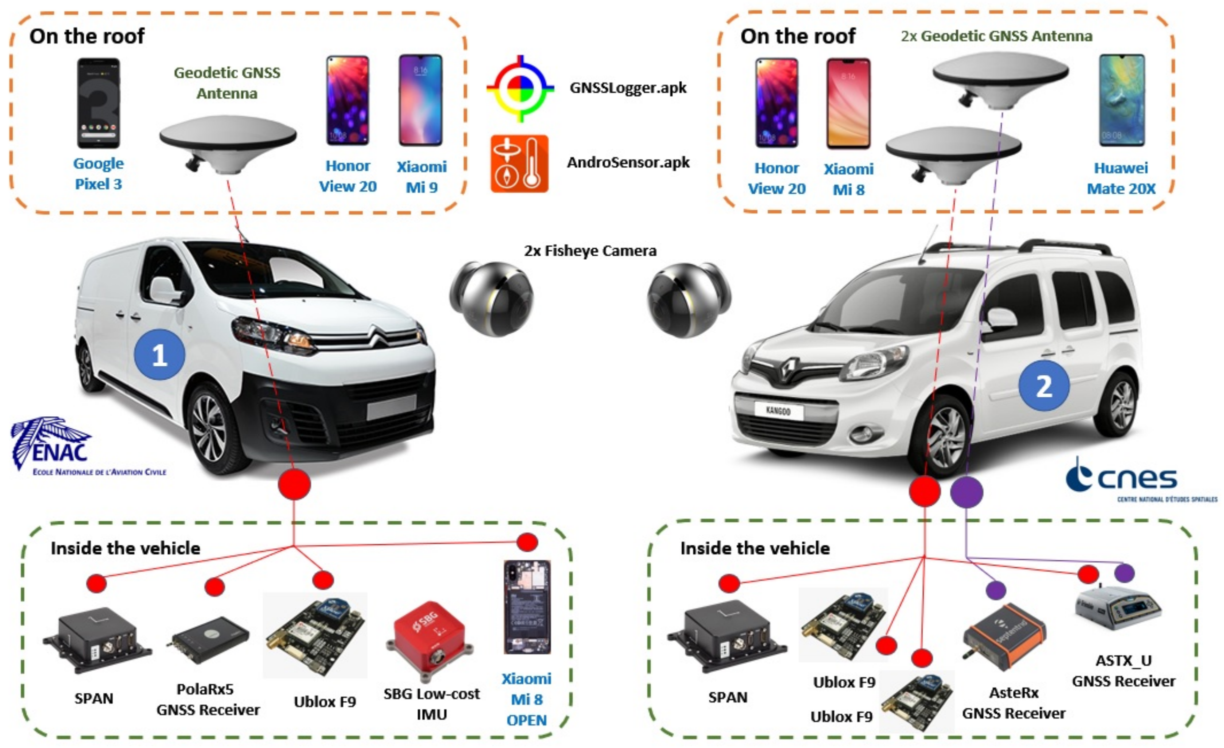


Figure 3.4: Schematic Representation of Both Research Vehicles Configuration used for the Smartphone Data Collection Campaign

campaign was to assess and characterize the impact of the smartphone antenna on the recorded GNSS raw data measurements. One of the Xiaomi Mi8 was open, exposing its internal components, while still being able to function. An antenna adaptator was weld into place at the smartphone embedded antenna output port, thus allowing the roof-mounted geodetic antenna to feed signals to the smartphone embedded GNSS receiver. Furthermore, three geodetic antennas were placed also on the top of the cars, GNSS signal splitters redistributed captured signals to the on-board equipments as shown by the connected lines. Finally, two fisheye cameras aiming straight at the road were placed on both vehicles to gage for cars trajectories and scenarios length in post-processing.

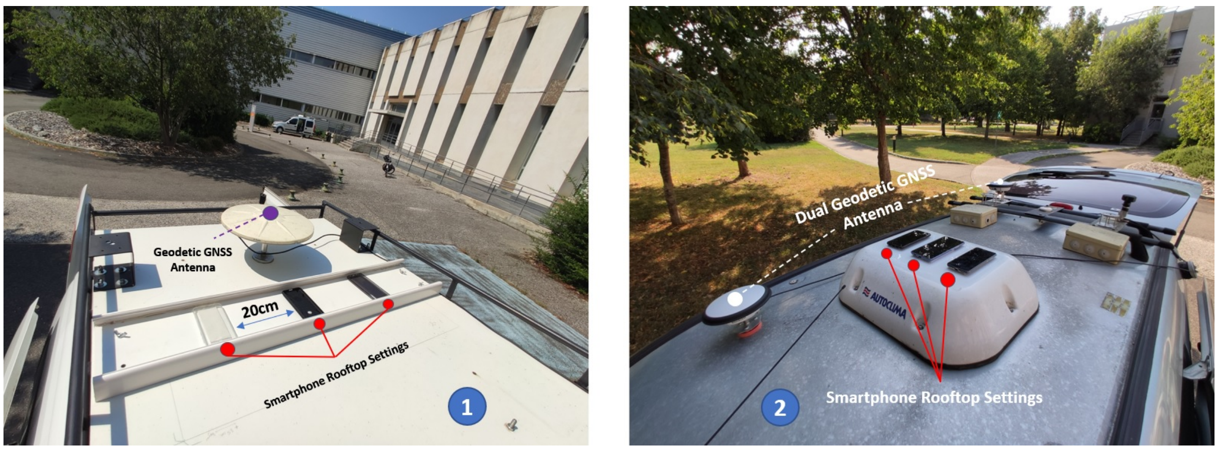

Geodetic GNSS Antenna
20cm
Smartphone Rooftop Settings
1
Dual Geodetic GNSS Antenna
Smartphone Rooftop Settings
2

Figure 3.5: Vehicles Rooftop View Showing Smartphones Layout Configuration

3.2.2 Collaborative Scenarios

Early on in this research work and after carefully reviewing the state-of-the-art of smartphone-based urban positioning (see 3.1), it has been decided that the implementation of collaborative algorithms would be the best fit for overcoming and mitigating most urban smartphone-based positioning issues. It was ambitioned to develop a collaborative user network taking advantage of the tremendous number of connected Android devices in today's busy city centers. A whole thesis segment, part II, is dedicated to argument the benefits of collaborative algorithms applied for urban smartphone-based positioning, and also include a complete literature review of existing collaborative process in the positioning domain 4.
In order to achieved this objective, various collaborative scenarios needed to be studied for the implementation of smartphone cooperative techniques. Four collaborative scenarios, depicting real-life cases, were implemented along the trajectory during the data collection campaign. The two vehicles would represent two cooperating smartphone users. The creation of scenarios was based on urban pedestrian low dynamic positioning, referencing real-life positioning application mostly used by smartphone users. The following items describe the implemented collaborative scenarios and their respective real-life application case.

- **Collaborative Scenario A:** Nominal-Static in Open-Sky
  Scenario A has been thought to be the baseline case scenario for collaborative smartphone-based positioning. Both vehicles were in an open-sky environment while being static. Collaborative scenario A lasted for 20 min during the data collection campaign.

- **Collaborative Scenario B:** Low Dynamic in Deep Urban Canyon
  Scenario B represents a collaborative structure between two users in a deep urban canyon environment. Both users were close to each other and navigated around a block of flats. The recording of the collaborative scenario B lasted for 10 min.

- **Collaborative Scenario C:** The French *Bistro Café*
  Collaborative scenario C, so called French *Bistro Café*, characterizes a real-life situation where one user would be static in a reasonably open-sky environment, while the second user is navigating around him in a more constrained urban conditions. This case scenario is inspired from real-life situations where people would be sitting in a city square or park (e.g: in open-sky reception conditions) and who could cooperatively help other network users navigating in deep urban environment nearby. The implementation of this scenario was achieved by setting one of the car on the final open floor of a silo parking lot in Toulouse city center, while the second car was navigating streets around the same car park. This collaborative scenario lasted for 20 min during the data collection campaign.

- **Collaborative Scenario D:** Low Dynamic in Urban Environment
  The final collaborative scenario D, depicts a low dynamic situation (pedestrian velocities) under a path surrounded by both trees and buildings, blocking satellite signals. This collaborative scenario lasted for 10 min during the data collection campaign.

Collaborative scenarios locations selected during the data collection campaign are depicted on figure 3.2 by environmental snapshots.

3.3 Android GNSS Raw Measurements Evaluation

Smartphone positioning engine architecture advantages and flaws have been presented above, in chapter 2. The state of the art review made on smartphone advanced positioning techniques suggested that the implementation of such algorithm was achievable on Android devices. A focus on collaborative positioning techniques will be made by this research work, taking advantage of the increasing number of connected smartphones in busy city centers, for mitigating positioning errors in urban environment on Android smartphones. Thus, characterizing Android GNSS raw data measurements remains an important task for ensuring the reliability of collaborative transmitted GNSS data within a network of smartphone users. Smartphone's measurements recorded during the data collection campaign, detailed in section 3.2, will be used for developing this characterization method in both nominal and urban environments. Few studies developed assessments analysis on Android GNSS raw data measurements, [59] [60]. Plus, most of them drawn their conclusions based on one smartphone brand and model in optimal open-sky conditions despite the fact that most smartphone-based positioning activities are achieved in urban and sub-urban areas.

The following, in depth, characterization of smartphone's GNSS measurements will aim at analyzing multiple smartphones embedded GNSS receivers behaviors and performances in ”real-life” urban scenarios. The first objective will be to better cope with Android GNSS raw measurement and better understand their impact on the final positioning solution while navigating deep urban environment. A study of Android GNSS raw data measurements output will be made, as well as a thorough investigation on Android flag detection mechanisms will be provided. Then, smartphones nominal positioning performances will be tested, supported by a statistical analysis. After, smartphone positioning characterization will be made for urban environment positioning, while discussing on the feasibility of implementing urban collaborative smartphone algorithms.

3.3.1 GNSS Observables Preparation

One of the main pillar for characterizing smartphone positioning performance parameters is to exhaustively understand Android GNSS raw data measurements and how to properly manipulate its derivatives. Android GNSS raw data measurements are the outputs of the signal processing block of the embedded chipset receiver. This subsection will present the post-processing algorithm developed for converting those raw data measurements into usable GNSS observables. The following analysis will show in details the particularities and features associated with the manipulation of Android GNSS raw data measurements.

3.3.1.a Pre-processing Procedure

The creation of a Matlab script was initially developed for fully exploiting Android GNSS raw data measurements and to convert them into GNSS observables. This algorithm has been inspired and modified from the open-source code developed by Google via their *GNSS Analysis Tool*. This process allows for keeping full control on the interpretation of the recorded measurements and thus, completely apprehending recorded smartphone measurement from the output of the chipset to the post-processing use of the generated observables.

Android GNSS raw data measurements are recorded in a log file, in a Comma-Separated Values (CSV) format. This file is written and saved by an application called *GNSSLogger*. This application has been developed by Google exclusively for recording the embedded chipset measurements outputs. The main advantage is that, the application directly transcribes GNSS raw data measurements in an unbiased way. The file CSV format is standard; with firstly a header describing file's elements followed by the data. In this document, every line starting with the word "`RAW`" corresponds to the registered information for one particular received signal. The nature and number of raw data measurements recorded are listed in 2.4.3.a on figures 2.10 and 2.11. The Matlab post-processing algorithm global architecture is described step by step below:

1. Extract and reading raw data measurements from the log file
2. Inspection of measurements quality for clock and raw GNSS data
3. Filter each received signal measurement individually
4. Convert Android raw measurements into GNSS observables
5. Store GNSS observables into output structures epoch by epoch

Moreover, during step 3, a particular Android field measurement is specifically used for assessing the current signal synchronization state for the associated satellite. This Android field is called `State` and comes from the *GNSSMeasurement* Android class (see figure 2.11). This class will return an integer value that indicates the current tracking state of the signal. The final value returns by `State` is a combination state status, each represented by a unique integer number. The list of state status can be found in [11]. If `State` parameter takes any value above 16384, then the current signal measurements is reliable and can be processed. Whereas on the other hand, if `State` takes any value below the threshold value, then the current signal measurements is discarded. This numerical value signifies that at least the current measurement Time-of-Week (ToW) has been decoded and is known. As a general rule of thumb, only in 5% of the time a GNSS measurement is ruled out due to its `State` status. This number increases in urban environment where loss of lock events become more frequent.

The converted GNSS observables measurements are stored in a structured format. The output array is organized into two categories. First, an header array is created regrouping general observations and data such as: number of observations, observation types (C1, L1 and more...), system time reference and the first epoch time. The second category is formed into an array structure that includes GNSS observables. The generation of GNSS observables is made in step 4 of the algorithm process and is presented in detail in section 2.4.2. Equation 3.1 provides a template of the observation array structure for outputting GNSS observables.

$$\underset{N\times M}{\text{OBS\_OUTPUT}} = [\texttt{t, SvId, C1, C5, L1, L5, S1, S5, D1, D5, Flags}] \tag{3.1}$$

where:

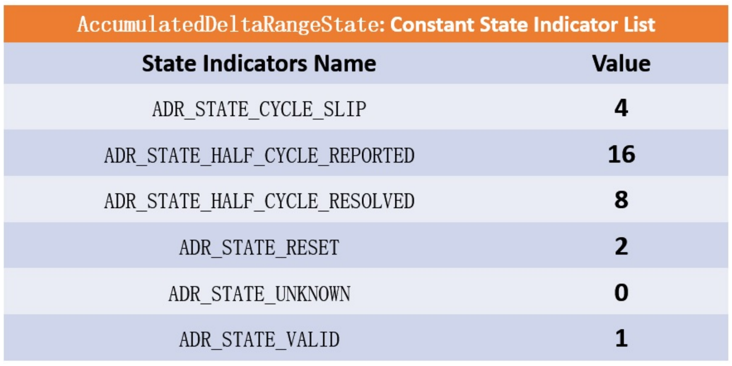
- `N` is the total number of epochs.
- `M` represents the number of GNSS observable types.
- `t` stands for the receiver time of the current observables in GPS time.
- `SvId` is the satellite identifier that includes the constellation and PRN numbers for each satellite
- `C` represent pseudorange measurements per frequency band (L1 or L5).
- `L` are carrier phase measurements per frequency band (L1 or L5).
- `S` stands for signal strength indicators per frequency band (L1 or L5).
- `D` are Doppler offsets per frequency band (L1 or L5).
- `Flags` represent extra state parameters indicating measurements quality (multipath and cycle slip)

The output structure can also be reconfigured into a RINEX type format. This ensure the replayability of the collected smartphone data by other researchers on various existing algorithms. However, due to the strict nature of the structure of a RINEX file, multiple measurements and information are lost while using this configuration (e.g: Multipath and cycle slip flags).

3.3.1.b Processing Android State Flag Mechanisms

Smartphones' embedded GNSS receiver architecture is mostly similar to COTS GNSS receivers, from capturing the signal to estimating its position. Android allowed their users to have access to raw data measurements outputted by the baseband signal processing unit of the chipset receiver. Raw data measurements range from the most basic parameters (i.e code, phase, Doppler, C/N0) to more complex ones (i.e Automatic Gain Control (AGC), signals states and indicators). Among these complex measurements it was found that the `Multipath Indicator` and the `AccumulatedDeltaRangeState` parameters. Few information are released by Android and/or by chipset manufacturers concerning their computation algorithms. In a time where smartphone GNSS receiver's technology advance rapidly, it became crucial to understand and evaluate those flags reliability in order to better characterize smartphones' positioning performances. Android GNSS raw data measurements are obtained through the use of the *Android.location* API. Within this API, a public class called *GnssMeasurement* contains GNSS data coming directly from the embedded chipset. This class is divided into two data groups. The first one, called *Public methods*, regroups all GNSS raw data measurements. The second one named *Constant*, gathers information about received signals characteristics. Within this second group, it was found that a `Multipath Indicator` and an `AccumulatedDeltaRangeState` that provide multipath and cycle slip flags detection mechanism to the informed user. A thorough study has been conducted for determining the nature and efficiency of Android detection mechanisms [61].

1. **Multipath Flag Detection Mechanism**
   The Android multipath indicator state flag can take three different values. If the flag takes the value of 1, a multipath interference has been detected for that measurement. On the other hand, when the indicator is set to the value 2 it signifies that multipath was not detected. Moreover, the indicator can also take the value of 0, meaning that the presence or absence of multipath is unknown. In our study, the multipath detection mechanism is simply activated when the indicator shows



AccumulatedDeltaRangeState: Constant State Indicator List	
State Indicators Name	Value
ADR_STATE_CYCLE_SLIP	4
ADR_STATE_HALF_CYCLE_REPORTED	16
ADR_STATE_HALF_CYCLE_RESOLVED	8
ADR_STATE_RESET	2
ADR_STATE_UNKNOWN	0
ADR_STATE_VALID	1

Figure 3.6: `AccumulatedDeltaRangeState` Constant States Indicator List. Adapted from [11], updated on 03/08/2021.

a value of 1. It has to be noted that only Honor View 20 smartphones reported signals being unaffected by multipath (i.e Multipath Indicator = 2).

2. **Cycle Slip Flag Detection Mechanism**
   Android phase measurement characterization is based on the combination value of six state indicators. Each indicator corresponds to a constant value, and the overall addition of those states is prompted to the user by the `AccumulatedDeltaRangeState` parameter. Those states are listed below in table 3.6. Processing cycle slip flag detection is done by identifying *ADR STATE CYCLE SLIP* and *ADR STATE RESET* constants presence in the final `AccumulatedDeltaRangeState` value.

$$\text{Valid\_State} = [1, 8, 9, 16, 17, 24, 25] \tag{3.2}$$

   Thus, after checking every possible combination values, an array of "Valid State" has been determined and it is shown in equation 3.2.
   If the current `AccumulatedDeltaRangeState` indicator value falls out of our selection then a flag is raised on our current measurement indicating a cycle slip.

3.3.2 Android Flag Reliability Analysis

Previously, the Android state flag parameters were introduced. Two types of flag are provided by the Android API: multipath and cycle slip detection mechanisms. However, a performance assessment is needed to determine the reliability of the provided information. This section aims at analyzing the performance of the Android flag mechanisms through correlation of detection events and an evaluation of the detection performance.

3.3.2.a Correlating Android Flags Detection Events

Figure 3.7 shows a preliminary signal analysis over the entire data collection campaign. In this example we took one of our tested smartphone the Huawei Mate 20X. Due to the rapid evolution of the user-to-satellite propagation channel, fast varying C/N0 values was observed. The top graph of Figure 3.7 illustrates those C/N0 fast fluctuations observed over time. For each smartphone, the minimum, median and maximum C/N0 value has been computed in function of each individual received signal (considering all frequencies and all constellations simultaneously) for every epochs. For the Huawei Mate 20X, the median value of C/N0 ranges between 30 and 35 dBHz. The bottom graph of

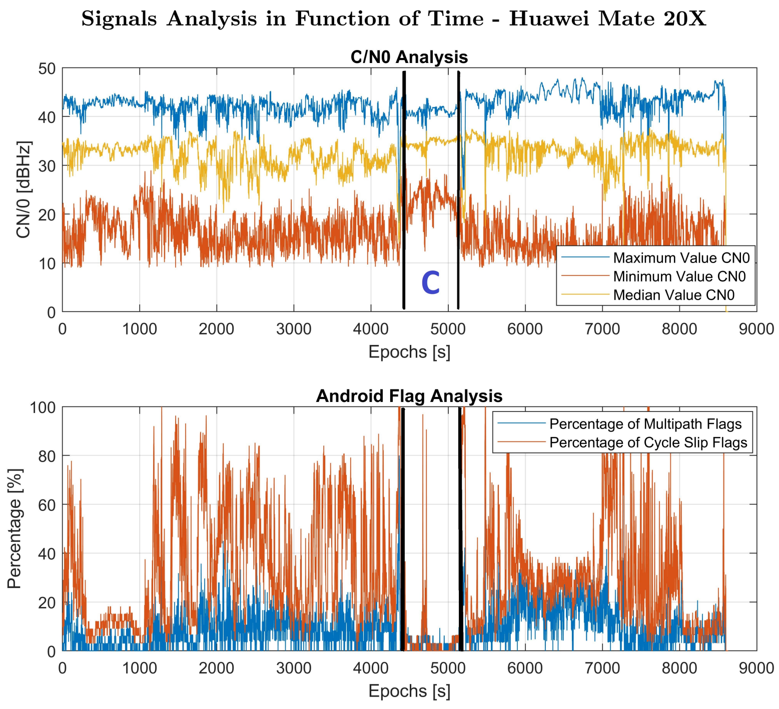


Figure 3.7: Signal Analysis of the Huawei Mate 20X. a) Upper Figure: C/N0 Analysis. b) Lower Figure: Percentage of Android Flags Activation

figure 3.7 shows the percentage of signals where a multipath and/or a cycle slip detection has been recorded in function of time. The percentage computation was obtained by dividing the number of flags detected by the total number of received signals for that specific epoch. The first observation made here is that cycle slip events seem to be often detected by the embedded smartphone receiver whereas multipath detections remain less frequent. The mark, labeled **C**, on figure 3.7 highlights collaborative scenario C. During this time, the second car was parked on the last floor of a parking garage in open-sky condition. C/N0 values of all signals improved, while both cycle slip and multipath flag detection decreased as expected. On the other hand, uncorrelated situations between C/N0 and flags detection have been observed during multiple occasions. This situation can be observed on figure 3.7 between epoch 400 and 1000, where median signal strength remains constant during that time period whereas flags activation numbers suddenly decrease.

Similar analysis has been performed for all tested smartphones. It is worth noting that the Google Pixel 3 and the Xiaomi Mi 9 did not record any phase measurements data, due to constructors choices that decided not to release this data in their units. It is then safe to state that cycle slip detection is not possible for those devices. Moreover, no multipath flags have been raised during our data campaign by either phone. This evidence suggests that the *Multipath Indicator* algorithm is not a naive linear correlation of C/N0 variation but exploits the phase measurement to detect multipath.

Independently of those phones, it appeared that both Xiaomi Mi 8 and the Huawei Mate 20X exhibit similar behaviors. However, both Honor View 20 (equipped with the same chipset as the Huawei Mate 20X) generated fewer cycle slip flags. All related figures for this analysis will be exposed in Appendix C.

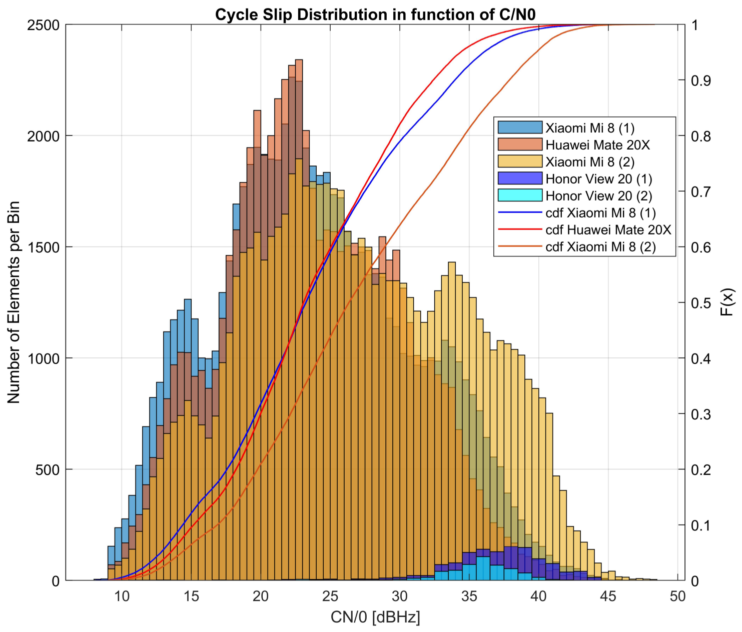


Figure 3.8: Cycle Slip Flags Detection Distribution in function of C/N0

Following our preliminary analysis, multiple basic GNSS measurements have been tested through a series of correlation events. The following hypothesis was made: multipath and cycle flag detection algorithms are not solely linearly correlated to C/N0. To validate this hypothesis, flags distributions in function of C/N0 and elevation were analyzed. Figure 3.8 represents the cycle slip flag detection distribution in function of C/N0. Histograms and cumulative density functions (cdf) are drawn here. Cycle slip detection distribution seems to be quite uniformly distributed over C/N0 values. Even though our tested smartphones are not equipped with the same chipset component, they tend to have similar detection behaviors (increased detection activity between 13 and 16 dBHz, before peaking around the C/N0 value of 22 dBHz). However, Honor View 20s did not detect as many cycle slip flags as other devices and their distributions are surprisingly shifted towards higher C/N0 values (around 35 dBHz). Multipath flag detection distribution in function of C/N0 have similar characteristics as the one observed in the case of cycle slip flag distribution. The distribution of multipath and cycle slip flags have also been studied in function of satellite elevation. Those figures are listed in Appendix C.
Overall, Android flag detection systems are not naively only interpolated from C/N0 and satellite elevation value of the current signals. Detection mechanisms might be as complex as the one found in modern COTS GNSS receivers. All smartphone brands and models shown similar distribution patterns making us believe that those estimation algorithms may be using the same detection techniques at the chipset level and/or that these flags might be computed at a common low-level Android layer.

3.3.2.b Android Flag Detection Performance

Furthermore, in order to get a sense on those indicators efficiency, the Code-Minus-Carrier (CMC) of the two highest satellites for each smartphone has been computed. An analysis made on satellite reception conditions (C/N0 and elevation) defined the Galileo PRN 12 and GPS PRN 27 satellites as the best available satellites on both frequencies for the study of Android device in dense urban canyons. Both satellites have been selected because they were at the highest elevation angle and visible during an extensive part of our data collection campaign. The CMCs are then computed to visualize potential large cycle slips and multipath degradation. Pseudorange model are described in equation 3.3 and phase measurement models in equation 3.4 [7]. Equations 3.5 and 3.6 model the difference between the code and phase measurements for a satellite SV at epoch i. Both ϵ_{Phase} and $\epsilon^{\phi}_{Multipath}$ terms have been neglected since $\epsilon_{Code} >> \epsilon_{Phase}$ and $\epsilon^{\rho}_{Multipath} >> \epsilon^{\phi}_{Multipath}$.

$$\rho_i^{SV} = r + c(t_{rx} - t_{tx}) + \epsilon_{Iono} + \epsilon_{Tropo} + \epsilon_{code} + \epsilon^{\rho}_{Multipath} \tag{3.3}$$

$$\varphi_i^{SV} = r + c(t_{rx} - t_{tx}) - \epsilon_{Iono} + \epsilon_{Tropo} + N\lambda + \epsilon_{Phase} + \epsilon^{\varphi}_{Multipath} \tag{3.4}$$

$$CMC_i = \rho_i^{SV} - \varphi_i^{SV} = 2\epsilon_{Iono} - N\lambda + \epsilon_{code} + \epsilon^{\rho}_{Multipath} \tag{3.5}$$

$$CMC_i^{detrended} \approx \epsilon_{code} + \epsilon^{\rho}_{Multipath} \tag{3.6}$$

where:

- r = User-satellite range
- $(t_{rx} - t_{tx})$ = Receiver minus satellite clock offset
- ϵ_{Iono} = Ionospheric error
- ϵ_{Tropo} = Tropospheric error
- ϵ_{Code} = Delay Lock Loop (DLL) Jitter
- ϵ_{Phase} = Phase Lock Loop (PLL) Jitter
- $N\lambda$ = Ambiguity term
- $\epsilon_{Multipath}$ = Multipath error

Equation 3.5 presents also the detrended CMC model where the influence of the ionospheric error and the ambiguity terms have been removed. The ambiguity term $N\lambda$ has been fixed by computing a sliding CMC mean for continuous observation segments and rounding to the nearest integer. Thus, satellite continuous tracking segments being shorts, our mean ambiguity fixing computation also corrects for the ionospheric term (ϵ_{Iono}) since ionospheric error is a slow varying component. The remaining parameters of the detrended CMC expression in 3.5 are then multipath errors plus Gaussian noise.
Figure 3.9 exhibits CMC computations for one of the tested Xiaomi Mi 8. Other smartphones CMC plot analysis can be found in Appendix C. On this graph, the top plot represents the CMC evolution in time, while applying the sliding mean fixing method on segments where the satellite was physically visible by the receiver. This implies that cycle slip should still be visible on that plot (e.g. red boxes on figure 3.9), and red dots show where a cycle slip flag activation has been reported by Android. Thereafter, the bottom plot illustrates the computed CMC values still corrected by the sliding mean fixing method per segments. However this time, segments were said to be continuous if the satellite was physically visible *and* if the Android flag algorithm did not detect any cycle slip. In this case, visible cycle slips remaining on the figure would mean that Android failed to correctly detect cycle slips. Theoretically at this stage, cycle slips should have been removed, leaving multipath and noise characteristic behaviors on the CMC plot. Purple dots indicates Android multipath flag detection events.

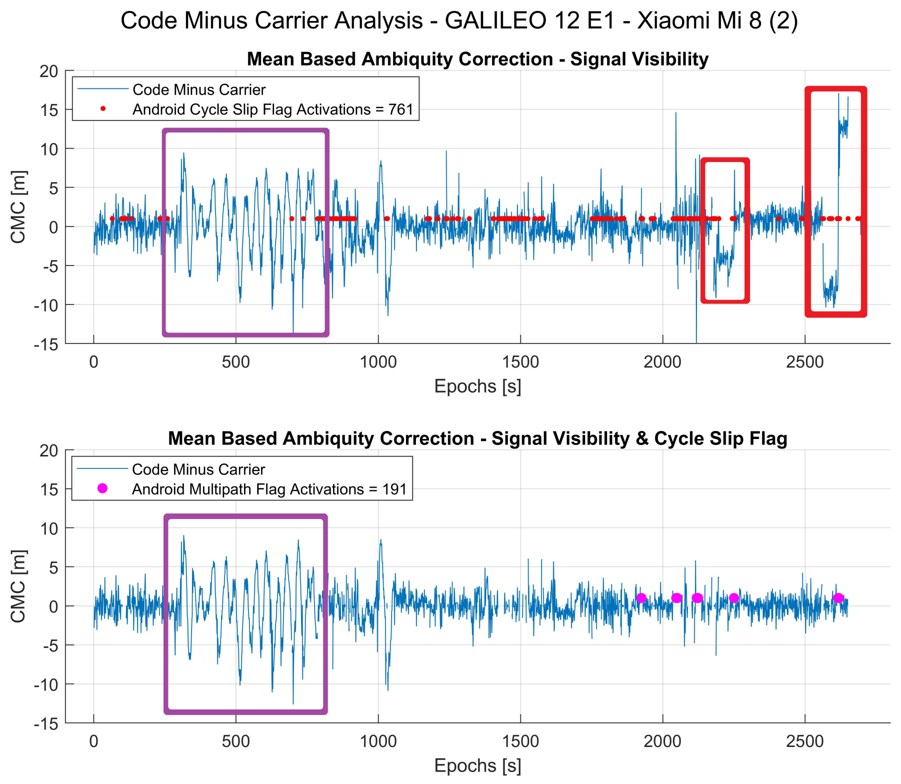


Figure 3.9: Code-Minus-Carrier Correlation Analysis between real Multipath and Cycle Slip Occurrences and Android Flags Detection Mechanisms

On the top figure 3.9, cycle slip flags have been activated 761 times over 8350 seconds. The overall flag activation seems to be over proportionate and too strict to detect real occurrences. However, the few cycle slips that happened during our data collection seem to have been successfully detected by Android. Cycle slips occurrences, shown by red boxes, can be identified on figure 3.9 top plot whereas they do not appear on the bottom graph. Figure 3.9 also highlights the use of multipath flags. Firstly, multipath flag algorithm only detected 191 occurrences. This number is supposedly underestimating the reality of our deep urban environment data collection. Moreover, a significant multipath event is visible on both top and bottom graph. A typical multipath oscillation can be seen (depicted by purple boxes) and not being detected at any moment by the Android algorithm. This phenomenon was often seen for other satellites and other smartphones, implying that the multipath indicator is not triggered by a simple threshold on the CMC.

In conclusion, the computation processes of Android flags mechanisms seem to not be exclusively based on a naive interpolation of C/N0 or satellite elevation parameters. Moreover, the similarities observed between smartphone brands make us believe that detection algorithms might be computed at a low-level Android layer. Multipath flags tend to be inconsistent whereas cycle slip flags were proven to be coherent despite their apparently high false alarm rate. However, Android multipath and cycle slip indicators might not be used as reference parameters to qualitatively assess smartphone positioning performance.

3.3.3 Quality Assessment in Nominal Conditions

In order to explore the feasibility of smartphone collaborative positioning, accurate and realistic simulations must be developed. Classic GNSS error models measurements needs to be re-evaluated for smartphone-based positioning. Android GNSS raw data measurements are outputted by embedded chipsets receiver that use undisclosed signal processing techniques. Those ”black-box” processes do not allow to specifically characterize measurements errors based on classical criterion. Therefore, a statistical analysis of retrieved smartphone GNSS measurements will be developed in open-sky nominal conditions. Plus, smartphone positioning performance parameters will be evaluated against low-cost GNSS receivers.

3.3.3.a Measurements Error Statistical Analysis

The characterization of Smartphones’ GNSS measurements starts with the statistical analysis of measurements errors derived from the internal hardware processes. As exposed in 2.3, smartphone embedded GNSS receivers operates, for the most part, similarly to any other GNSS receivers. A typical GNSS receiver architecture and processes are shown in Appendix A.2. The main goal of our nominal conditions analysis is to determine whether Android GNSS measurements are suitable to be used for smartphone-based collaborative applications. The focus here is put on measurements related errors that are directly generated by the receiver instrumentation. This low-level error analysis will provide valuable and accurate information in the perspective of simulating smartphone-based collaborative techniques. Theoretical error models on GNSS receivers tracking loop parameters have been extensively studied and developed in the literature [24] [62] [63]. However, those standard models depends on specific receiver architecture information that is not always available. In the smartphone positioning domain, receiver architecture and tracking loops parameters are generally kept secret for confidentiality reasons. Hence, classical measurements models cannot be applied to characterize smartphone’s measurements. Moreover, due to the large plurality of embedded GNSS receivers chipset brands and models, it is impossible to predict or simulate all smartphones’ positioning behaviors.
Consequently, the evaluation of receiver-originated measurements errors for smartphone GNSS receivers black-box processes will be made from a statistical analysis of real-life recorded measurements. From there, an assessment method will be introduced for facilitating the characterization of GNSS measurements for future and present Android devices. This analysis has been made in nominal open-sky environment in order to easily isolating receiver instrumentation errors.

This study will mainly focus on the main smartphone receiver’s parameters, driving the smartphone positioning performances. GNSS receivers highly rely on the quality of their embedded Numerically-Controller Oscillators (NCO). This receiver’s element is used for generating a signal replica correlating the incoming received signal. Loop filters, Phase Lock Loop (PLL) and Delay Lock Loop (DLL) are used for controlling the generated NCO signal code and carrier. Those components are recognized to be the main source of receiver-based error within the outputted raw data measurements. Therefore, this study will focus of the modeling of the uncorrelated errors of the PLL, DLL and Frequency Lock Loop (FLL) also referred as jitter. Moreover, a review of smartphone clock stability will be made. The characterized receiver’s parameters of this study are detailed below:

- **Phase Jitter**: σ_{PLL}
- **Code Jitter**: σ_{DLL}
- **Doppler Jitter**: σ_{FLL}
- **Clock Drift**: $c\dot{\delta t}$

This characterization is based on field measurements extracted mainly from the data collection campaign in section 3.2. Only the scenario A will be considered in this analysis. As a reminder, both vehicles and thus all smartphone were recording Android raw data measurements in an open-sky/static case. Additionally, two more sets of raw data measurements were captured in clear, open-sky environment. Those recordings were made after the data collection campaign and were used for assessing and comparing measurements error of newly acquired Android smartphones. Thus, the Samsung S10+ and a third Xiaomi Mi8 will be analyzed, totaling 9 tested smartphones.

1) Error Modeling: Clock Drift

GNSS receivers provides the capabilities to compute user position, timing as well as velocity. In most GNSS receivers, including smartphone devices, Doppler measurements are obtained through the processing of carrier-phase measurements. Receiver's clock drift can be derived from the estimation of user's velocity. Detailed derivation of user velocity estimation can be found in [64]. Smartphones raw data measurements give access to primary GNSS observable data (including: Pseudoranges, Pseudorange Rate and Doppler). The pseudorange rate can be modeled according to the equation 3.7.

$$\dot{\rho_i}^{SV} = v_{xi} \cdot a_{xi} + v_{yi} \cdot a_{yi} + v_{zi} \cdot a_{zi} + c\dot{\delta t} + \epsilon_{\dot{\rho}_i} \tag{3.7}$$

where:

- $\mathbf{v}_{ui}^{SV} = [v_{uix}, v_{uiy}, v_{uiz}]$: user-satellite relative velocity vector knowing that:
  - $\mathbf{v}_{u}^{SV} = [v_{ux}, v_{uy}, v_{uz}]$: user velocity vector
  - $\mathbf{v}_{i}^{SV} = [v_{ix}, v_{iy}, v_{iz}]$: satellite velocity vector
- $\mathbf{a}_{i}^{SV} = [a_{xi}, a_{yi}, a_{zi}]$: user-satellite direct line of sight unit norm
- $c\dot{\delta t}$ = receiver clock drift
- $\epsilon_{\dot{\rho}_i}$ = observation residual error

After scaling the pseudorange rate equation, we set d_i^{SV} representing the Doppler shift (Equation 3.8) taking the form of the difference between the observed Doppler and the predicted Doppler parameter. In our case, the predicted Doppler value can be computed since the exact position of the user is known (estimated from the high-end COTS GNSS receiver, the *NovAtel SPAN*, on-board both vehicles). Finally, the design-matrix of the Doppler-based velocity model can be expressed, as in equation 3.9.

$$d_i^{SV} = \dot{\rho_i}^{SV} - \mathbf{v}_i^{SV} \cdot \mathbf{a}_i^{SV} \tag{3.8}$$

Equation 3.9 is written in function of four unknown parameters v_{ux}, v_{uy}, v_{uz} and $c\dot{\delta t}$. This equation can be solved using well-known estimation techniques such as the Kalman

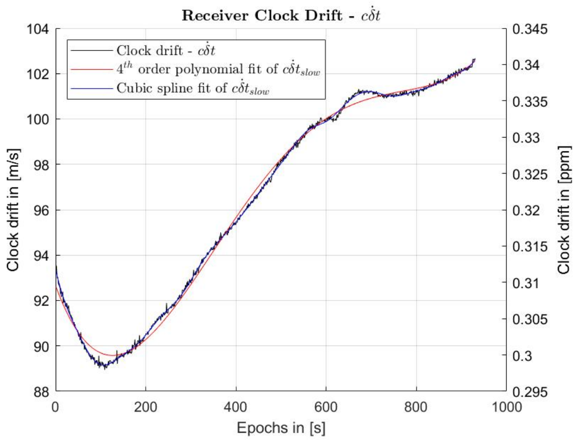


Figure 3.10: Receiver Clock Drift in [m/s] and [ppm] ($c\dot{\delta t}$)

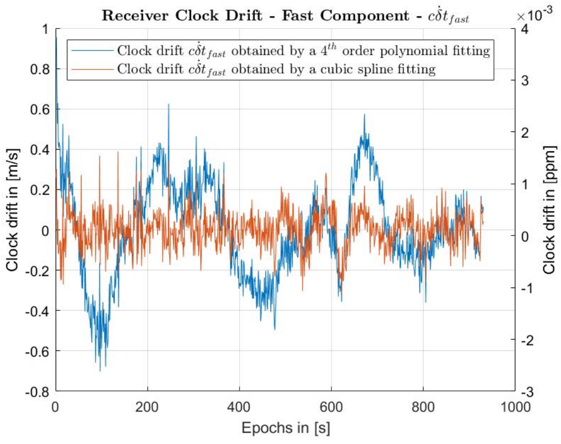


Figure 3.11: Clock Drift Fast Component ($c\dot{\delta t}^{fast}$)

Filter (KF) or the Weighted Least Square (WLS). As a results we obtained the estimated receiver's clock drift parameter. Moreover, due to the static property of our study, a quick verification of our estimation method was to confirm that our estimated velocity were approximately equal to zero.

$$\begin{pmatrix} d_i^1 \\ d_i^2 \\ \vdots \\ d_i^n \end{pmatrix} = \begin{pmatrix} -a_{x_1} & -a_{y1} & -a_{z_1} & 1 \\ -a_{x_2} & -a_{y2} & -a_{z2} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ -a_{x_i} & -a_{y_i} & -a_{z_i} & 1 \end{pmatrix} \cdot \begin{pmatrix} v_{ux} \\ v_{uy} \\ v_{uz} \\ c\dot{\delta t} \end{pmatrix} + \begin{pmatrix} \epsilon_{\dot{\rho}1} \\ \epsilon_{\dot{\rho}2} \\ \vdots \\ \epsilon_{\dot{\rho}_i} \end{pmatrix} \tag{3.9}$$

The estimated clock drift is assumed to impose the same clock error regardless of the constellation. Therefore, we can rewrite our clock drift term $c\dot{\delta t} = c\dot{\delta t}_{GPS} = c\dot{\delta t}_{Galileo} = c\dot{\delta t}_{GLONASS} = c\dot{\delta t}_{Beidou}$. The estimated clock drift, presented in figure 3.10, shows two clear components: the fast and the slow varying component. It is assumed that the slower component of the clock drift is due to a satellite clock offsets that develop slowly in time. Thus, the estimated $c\dot{\delta t}$ can be divided in two terms (equation 3.10). As stated above, we are interested in the receiver-based error that reside in the clock drift fast component.

$$c\dot{\delta t} = c\dot{\delta t}^{fast} + c\dot{\delta t}^{slow} \tag{3.10}$$

Results obtained by this characterization analysis are shown in figure ranging from 3.10 to 3.14. The presented results are the one obtained for the Honor View 20 smartphone. All of the other tested Android devices statistical results presented similar characteristics and distribution. Figure 3.10 shows the estimated receiver clock drift in both [m/s] and [ppm] units. On this image, the distinction between the fast and slow varying component are clearly visible. This first observation validates our previous hypothesis on the clock drift component division. Hence, the red and blue lines represent the fitting strategies used for estimating and removing the slow-varying component of $c\dot{\delta t}$. As a result, figure 3.11 presents the clock drift fast component $c\dot{\delta t}^{fast}$. Two fitting methods were tested in order to accurately estimate the slow-varying part of the clock drift: the polynomial fitting and the spline fitting. A third order spline fit seems to better match the slow-carying

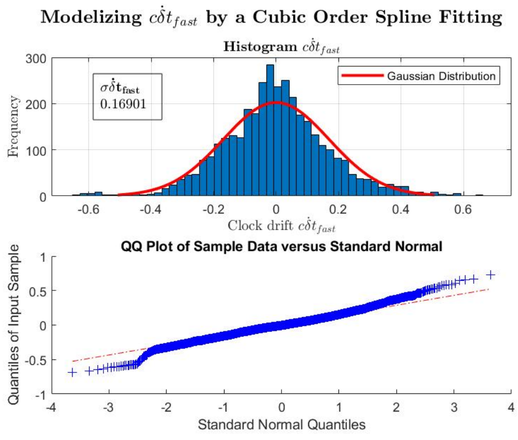


Figure 3.12: Cubic Spline Fitting of Clock Drift Fast Component Distribution

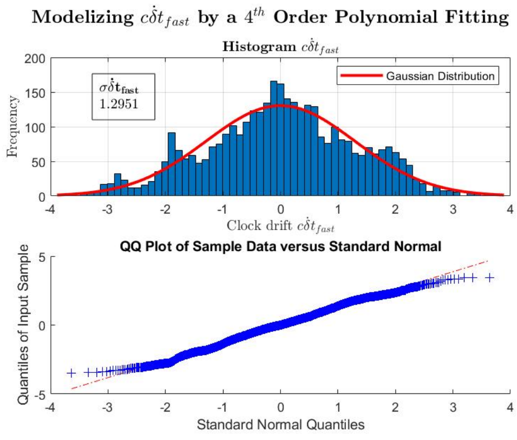
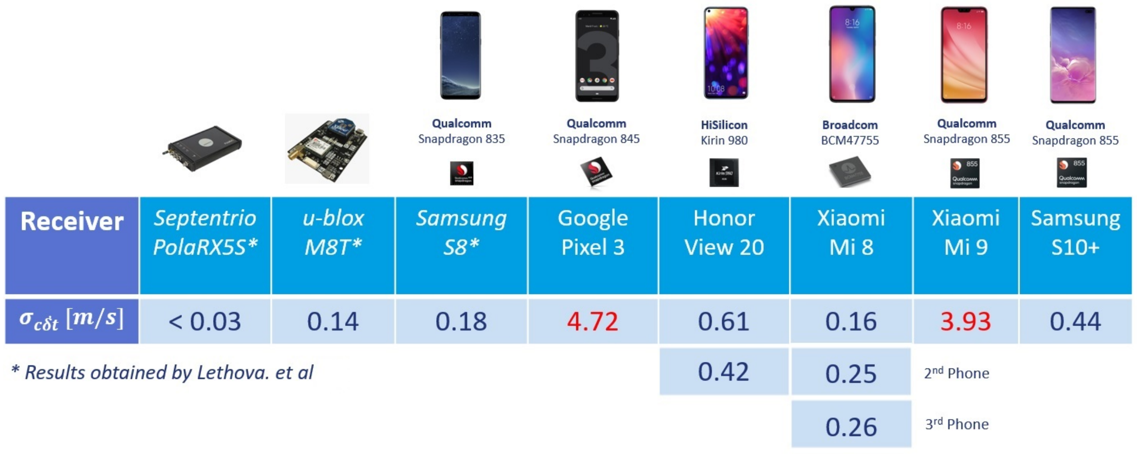


Figure 3.13: 4th Order Polynomial Fitting of Clock Drift Fast Component Distribution

component whereas the polynomial fitting still exhibits residues of the slow-varying error. Figures 3.12 and 3.13 present the distribution of clock drift error according to the chosen fitting technique. Both histogram distribution reflect Gaussian-like distribution features. The standard deviation of the clock drift error is presented on those figures for the Honor View 20.

Table 3.14 represents the estimated standard deviation of the receiver's clock drift for all tested smartphone. Moreover, the comparison is complemented by a previous analysis in [65] indicated by the asterisks. Both results from the Google Pixel 3 and the Xiaomi Mi 9 are highlighted in red due to their inconsistency with the rest of the tested devices. It is important to note that both Android devices do not have and do not grant access to phase measurements. Thus, the error biases present in their recorded Doppler measurements can explain the error repercussion on the estimated clock drift error. On the other hand, all other tested smartphones seem to have the same magnitude of clock drift error as a low-cost COTS GNSS receiver, the u-blox M8T. This results is encouraging and shows that embedded smartphone GNSS receivers have similar clock performance error resulting in accurate velocities solutions. Finally, similar smartphone models displays consistent



			Qualcomm Snapdragon 835	Qualcomm Snapdragon 845	HiSilicon Kirin 980	Broadcom BCM47755	Qualcomm Snapdragon 855	Qualcomm Snapdragon 855
Receiver	*Septentrio PolaRX5S**	*u-blox M8T**	*Samsung S8**	Google Pixel 3	Honor View 20	Xiaomi Mi 8	Xiaomi Mi 9	Samsung S10+
$\sigma_{c\dot{\delta}t}$ $[m/s]$	< 0.03	0.14	0.18	4.72	0.61	0.16	3.93	0.44
					0.42	0.25	2nd Phone	
						0.26	3rd Phone	

** Results obtained by Lethova. et al*

Figure 3.14: Estimated Standard Deviation of Receiver's Clock Drift ($\sigma_{c\dot{\delta}t}$ in [m/s])

results (e.g: Both Honor View 20 and the Xiaomi Mi 8 trio). This observation comfort the measurements modeling hypothesis made above and proves that the implementation of a strict assessment method for characterizing smartphone positioning performance is realistic.

2) Error Modeling: DLL Jitter

The receiver-based code measurement error, or DLL jitter, will be analyzed and quantified in this study. This error can be estimated by taking the time derivative of the difference between the code and phase measurements. Code and phase measurements models have been exposed in equation 3.3 above. Plus, the CMC model remains unchanged and is shown in equation 3.5. After simplification, the remaining terms of the CMC model are two times the ionospheric error and the code error. Indeed, the error related to the phase measurement has been neglected since the error magnitude of the code measurements is larger. Also, the multipath error is disregarded since our measurements were taken in an open-sky environment where the impact of multipath interferences is limited.

$$\text{CMC} = (\rho_i^{SV} - \varphi_i^{SV}) = 2\epsilon_{Iono} + \epsilon_{Code} \tag{3.11}$$

Equation 3.11 exposes the remaining terms of the taken difference between the phase and code measurements and the first order time derivative of the CMC model is described in equation 3.12.

$$\frac{d}{dt}(\rho_i^{SV} - \varphi_i^{SV}) = \frac{d}{dt}\epsilon_{Code} \tag{3.12}$$

The time derivative of the CMC allows to mitigate the ionospheric error since ionospheric error variation are considered slow ($\frac{d}{dt}(\epsilon_{Iono}) \approx 0$). Thus, leaving us with the expected DLL jitter estimation. Results analysis are presented on figure 3.15 and on 3.16. DLL jitter error distribution appears to follow a Gaussian distribution. According to the literature standard and for simulation purposes, DLL jitter has been computed in function of the C/N0. The obtained DLL errors appear to be exponentially decreasing with the

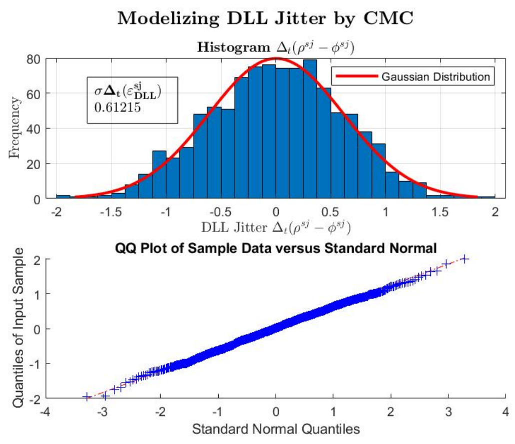


Figure 3.15: DLL Jitter Error Distribution

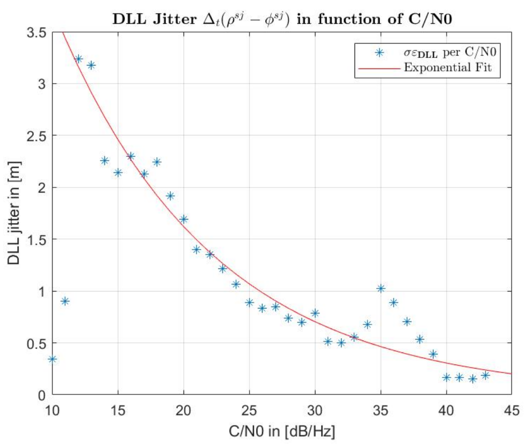
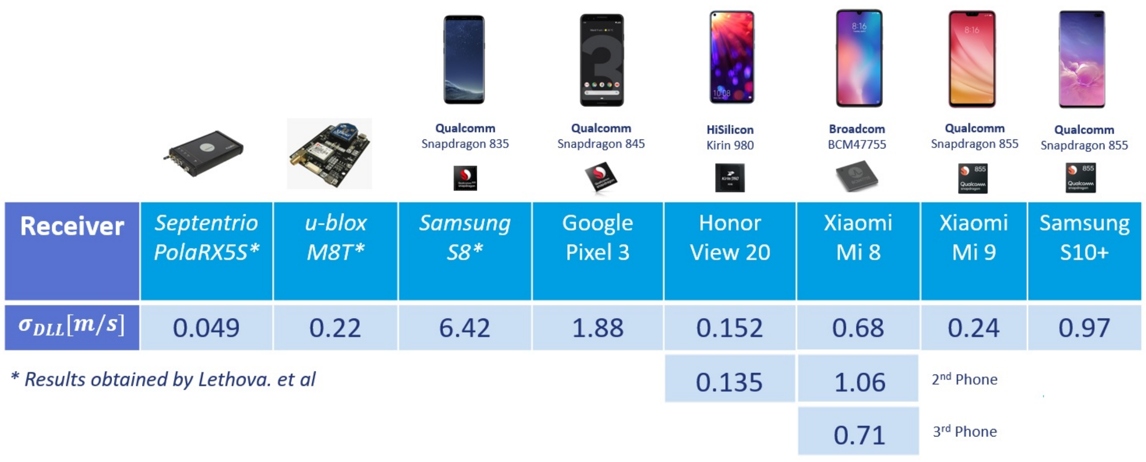


Figure 3.16: DLL Jitter in function of C/N0 (ϵ_{Code} in [m])



			Qualcomm Snapdragon 835	Qualcomm Snapdragon 845	HiSilicon Kirin 980	Broadcom BCM47755	Qualcomm Snapdragon 855	Qualcomm Snapdragon 855
Receiver	*Septentrio PolaRX5S**	*u-blox M8T**	*Samsung S8**	Google Pixel 3	Honor View 20	Xiaomi Mi 8	Xiaomi Mi 9	Samsung S10+
$\sigma_{DLL}[m/s]$	0.049	0.22	6.42	1.88	0.152	0.68	0.24	0.97
					0.135	1.06	2nd Phone	
						0.71	3rd Phone	

** Results obtained by Lethova. et al*

Figure 3.17: Estimated Standard Deviation of Receiver's DLL Jitter ($\sigma_{\epsilon_{Code}}$ in [m/s])

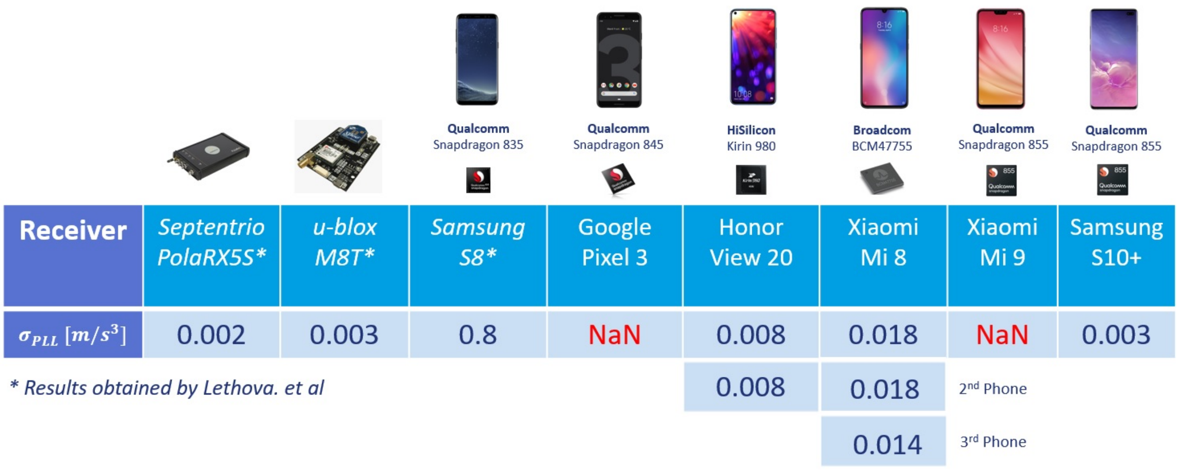
increasing signal strength. The exponential fit $f(C/N0)$, corresponds to a thermal model of the DLL jitter [66]. However, we can note a "bump" on the curve centered at 35 dB/Hz that remains unexplained at this time. Table 3.17 shows the estimated receiver's DLL jitter for all tested smartphone. Once again, those results shows the consistency and the performance level of the tested Android device. It appears that smartphone code measurements are of the same magnitude as the one on low-cost COTS GNSS receivers. However, smartphones equipped with similar chipset models (the Xiaomi Mi9 and the Samsung S10+ stocked with a Qualcomm Snapdragon 855) shows differences. The estimated error is 4 times lower on the Xiaomi Mi9 device. It is suspected that the Xiaomi OS operates and manage the chipset raw measurements differently. This would signify that two smartphones with identical chipset could perform and handle positioning duty differently.

3) Error Modeling: PLL Jitter

The receiver-based phase measurement error, or PLL jitter will be analyzed and quantified in this study. This error originate from the PLL tracking loop in charge of keeping the input and output phase in lock. Firstly, the PLL noise can be estimated by taking the triple time difference of the phase measurements (also known as: PLL jerk) [67]. Starting from the phase measurement model (φ_i^{SV}) of equation 3.3, the triple time difference is modeled as in equation 3.13.

$$\frac{d^3\varphi_i^{SV}}{dt^3} = \frac{d^3}{dt^3}\Big((r + c(t_{rx} - t_{tx})) - \epsilon_{Iono} + \epsilon_{Tropo} + N\lambda + \epsilon_{Phase} + \epsilon^{\varphi}_{Multipath}\Big) \tag{3.13}$$

The above mentioned equation can be simplified by setting the following hypothesis. The first term of the equation $\frac{d^3}{dt^3}r$ is mitigated since the observations are made in a static scenario. Furthermore due to the slow variability of tropospheric and ionospheric errors, these terms $\frac{d^3}{dt^3}\epsilon_{Iono}$ and $\frac{d^3}{dt^3}\epsilon_{Tropo}$ are dismissed. In this scenario we will also assume that the ambiguity are fixed and that cycle slips occurrences have been detected, removing the term $\frac{d^3}{dt^3}N\lambda$. Then, we assume that the measurements have been collected in an open-sky



			Qualcomm Snapdragon 835	Qualcomm Snapdragon 845	HiSilicon Kirin 980	Broadcom BCM47755	Qualcomm Snapdragon 855	Qualcomm Snapdragon 855
Receiver	*Septentrio PolaRX5S**	*u-blox M8T**	*Samsung S8**	Google Pixel 3	Honor View 20	Xiaomi Mi 8	Xiaomi Mi 9	Samsung S10+
σ_{PLL} $[m/s^3]$	0.002	0.003	0.8	NaN	0.008	0.018	NaN	0.003
					0.008	0.018	2nd Phone	
						0.014	3rd Phone	

** Results obtained by Lethova. et al*

Figure 3.18: Estimated Standard Deviation of Receiver's PLL Jitter ($\sigma_{\epsilon_{Phase}}$ in $[m/s^3]$)

environment thus the term $\frac{d^3}{dt^3}\epsilon^{\varphi}_{Multipath}$ can be neglected.
Finally, equation 3.13 can be approximated by the third time derivative of the phase measurement error $\frac{d^3}{dt^3}\epsilon_{Phase}$ as the sole unmitigated term. The standard deviation of the PLL jitter $\sigma_{\epsilon_{Phase}}$ can thus be estimated by applying a normalization factor of $\frac{1}{\sqrt{20}}$ as demonstrated in [67]. This estimation process is resumed in equation 3.14

$$\sigma_{\epsilon_{Phase}} \approx \frac{\frac{d^3\varphi_i^{SV}}{dt^3}}{\sqrt{20}} \tag{3.14}$$

The presented PLL jitter estimation technique assumes that cycle slips are not present in the phase observations. In this dissertation, we assume that the Android cycle slip flag detection mechanism can be used for determining phase segment without cycle slips; despite the high false alarm detection rate discussed in section 3.3.2.b.
Figures exposing the PLL jitter histogram distribution and PLL jitter as a function of C/N0 are displayed in [68]. The table synthesizing the results and the associated figures are also presented in [68]. Table 3.18 shows the estimated standard deviation of the PLL ($\sigma_{\epsilon_{Phase}} = \sigma_{PLL}$). As expected, the error level of the phase measurements is a thousand times smaller than the one for the code measurement. Again, similar conclusion can be drawn concerning the characterization of the phase measurements being similar to the one of a low-cost receiver in nominal, static and open-sky conditions.

3.3.3.b Chipsets Performance Variability

The results obtained on the characterization method needs to be balanced with smartphone positioning performance variability. Among the tested Android devices, two of them did not record phase measurements during the entirety of the data collection campaign. As exposed in our analysis above, the Xiaomi Mi 9 and the Google Pixel 3 did not record phase measurements of any tracked signals. This missing data is a deliberate choice from the chipset manufacturer to not include and/or track phase measurements in their receiver hardware. The most disturbing fact being that the Xiaomi Mi 9 is the direct evolution model of the Xiaomi Mi 8, who was the first-ever multi-constellation and multi-frequency smartphone and had phase measurements recording capability. Therefore, a

characteristic analysis based on smartphone brand cannot be envisaged given the differences that can be encountered. An assessment method will be developed for characterizing any Android smartphone following statistical results emanating from this complete smartphone study. Similar observations and conclusions have been drawn in [69].
Figure 3.19 highlights an other smartphone GNSS measurements feature characteristics of smartphone-based positioning. This figure displays the captured signals C/N0 along the collaborative scenario B during a short time (≈ 5 min). This shows a large variability of measured C/N0 for all captured signals. This characteristic have been observed on all tested Android smartphones. Moreover, few captured signals seems to be tracked at very low C/N0. This might be due to the high sensitivity of the receiver setting explained in 3.1.1.a. Signal consistency seems to be one of the main issues that we can observed on smartphone embedded chipset. This signal and measurements analysis will be now carried on urban environment.

3.3.4 Quality Assessment in Urban Environment

Urban environment positioning constitutes a great challenge for GNSS receivers. Signals can easily get degraded either by disruptive multipath and by NLOS signals reception. Those disruptions are mainly due to environment around the user made of tall buildings and usually referred as "urban canyons". These limiting factors are aggravated by smartphone's components. Indeed, the linearly polarized patch antenna, mounted on a tight logic board inside our smartphones, is not optimized for acquiring circularly polarized GNSS signals in disruptive conditions. Our analysis aims at recognizing potential limiting factors for developing a smartphone collaborative system. In the first place, a global positioning performance study will be presented for all our tested devices. This will allow to better draw the main picture of smartphone-based urban positioning. A specific section will detail the impact of smartphone's antenna in urban conditions thanks to the unique setup of our data collection campaign. Finally, a short conclusion will assess the suitability for Android GNSS raw data measurements for the future implementation of collaborative positioning techniques.

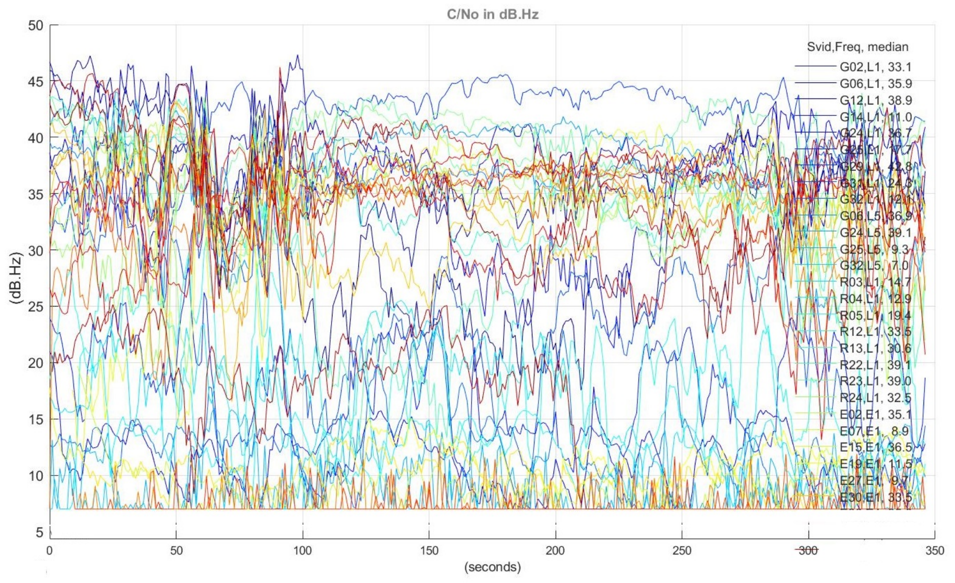


Figure 3.19: Signal C/N0 of Recorded Signals during the Scenario B in [dB.Hz]

3.3.4.a Positioning Performance

Smartphone positioning performance was expected to be greatly deteriorated in urban and constrained environment due to multiple architecture limitations, as exposed in A.2. However, preliminary urban positioning analysis demonstrated that Android smartphone devices were achieving a position solution accuracy of a couple meters. This accuracy performance can be directly confronted and compared to the level of performance expected from a COTS GNSS low-cost receiver. In our urban environment analysis, most tested smartphones were achieving the same level of positioning accuracy as the u-blox M8T receivers on-board both vehicles. This level of performance is due to the smart integration and hybridization of smartphone data within the Android FLP solution. FLP positioning solutions will represent the benchmark of global smartphone positioning performance, regardless of its ”black-box” nature. Figure 3.20 characterizes smartphone's GNSS-only positioning performance. This figure represents the positioning solutions output of two smartphones and the trajectory reference during the collaborative scenario C, in urban environment. The left graph, shows the vehicles trajectories with one being static in open sky conditions (green dots) and the second car was circling the area. The blue dots represents the GNSS-only solutions, computed with a WLS estimation technique, of the Xiaomi Mi 9. The red trajectory pictures the high-end COTS receiver (NovAtel SPAN: An IMU/GNSS coupled receiver) reference positioning solutions. As expected, consequently to smartphone's antenna specifications and local urban constraints, numerous position outliers are present in the final solution. The graph on the right, shows the smartphone positioning errors compared to the reference trajectory for the Xiaomi Mi 9. Mean positioning error for this smartphone is about 15 meters. This analysis will be considered as the performance baseline of GNSS-only solutions.

The nature and quality of recorded smartphone's measurements have also been studied for urban environment. Signal availability in city center was higher than expected. 36 signals were tracked in average from our fleet of tested smartphones (compared to 38 for Ublox F9P). This phenomena can be explained by the fact that the phones are now multi-constellation and multi-frequency which increases the number of tracked signals. Moreover, it appears that signals are acquired and tracked down to very low C/N0. Up to 10% of tracked signals have a C/N0 below 15 dBHz, observable on figure 3.19. A signal

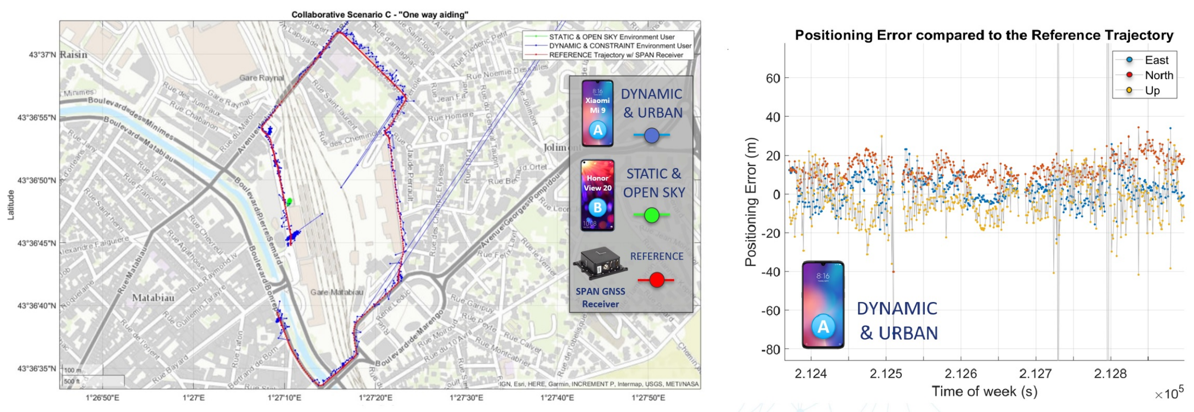


Figure 3.20: Smartphone *FLP* Positioning Solutions Compared to Reference Trajectory during Collaborative Scenario C and the Associated Smartphone Positioning Error Analysis in [m]

strength analysis has been made for the strongest observable satellite signal for the two Xiaomi Mi 8. Both tracked signals were observed during the entirety of the data collection campaign, allowing the analysis of signals strength evolution over time. Those results are exposed in figure 3.22. Moreover, it seems that E5a Galileo signals are more reliable over time than other tracked signals. This observation led to inspecting the nature of the most tracked signals per tested smartphones. This investigation is characterized by the bar diagram in figure 3.21. The percentages showed on the figure correspond to the ratio between the L1 and L5 frequency for GPS and Galileo. For some smartphones such as the Honor View 20, the proportion of L5 signals tracked is up to 92% for Galileo. this characteristic is observed to be only valid for Honor devices. Proportionally, on the L5 band, the most tracked signals are from the Galileo constellation. Certain signals and/or constellations appear to be favored by Android, through a weighting process within their positioning algorithm, according to signal reliability arguments allegedly.

3.3.4.b Impact of the Smartphone Antenna

As introduced during the presentation of the data collection campaign, one of the specificity of our analysis was to be able to characterize the smartphone antenna impact on the recorded measurements. One of our tested smartphone, the Xiaomi Mi 8 equipped on the Citroen Jumpy car had a specific feature. An antenna port was hacked onto our device in order to record GNSS signals from a geodetic antenna installed on the car's roof. A second Xiaomi Mi 8 smartphone with factory setting was used for benchmarking the observation and results obtained from the modified Android device. Several researchers attempted to characterize signal quality of smartphone's antenna [43] [70]. Recorded signals with low values of C/N0 (below 10 dBHz, observed in our analysis above) and poor multipath suppression have been imputed to the antenna performance by those cited research papers. Both Xiaomi Mi 8 devices, presented in this analysis, have similar hardware components and are running identical Android OS level (Android 10.0). The two smartphones are only differentiated by the antenna recording GNSS signals. Figure 3.23 shows the signal analysis of the Xiaomi Mi 8 connected to an external high-end antenna. This figure is intended to be compared with the graph printed above (in 3.22) showing the signal strength analysis for the second Xiaomi phone. It can be clearly identified that

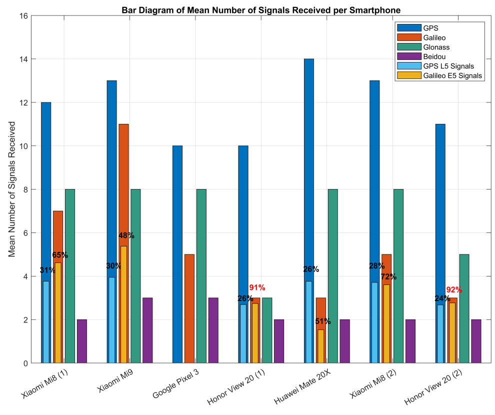


Figure 3.21: Bar Diagram of the Mean Number of Signals Received per Smartphones

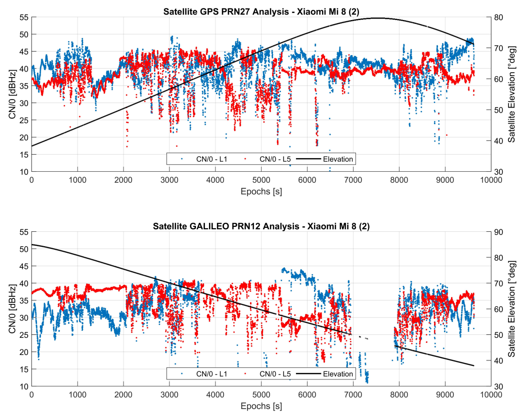


Figure 3.22: Signal Strength Analysis for the Xiaomi Mi 8

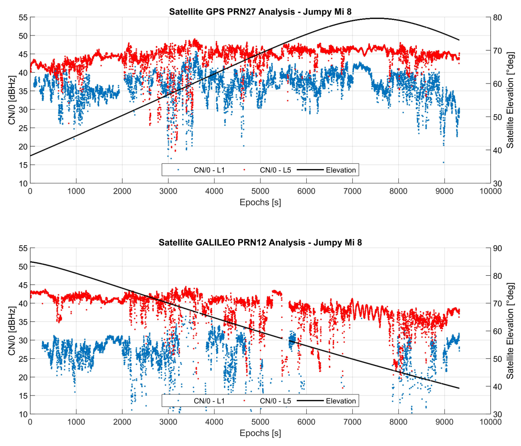


Figure 3.23: Signal Strength Analysis for a Xiaomi Mi 8 Connected to an External High-end Antenna

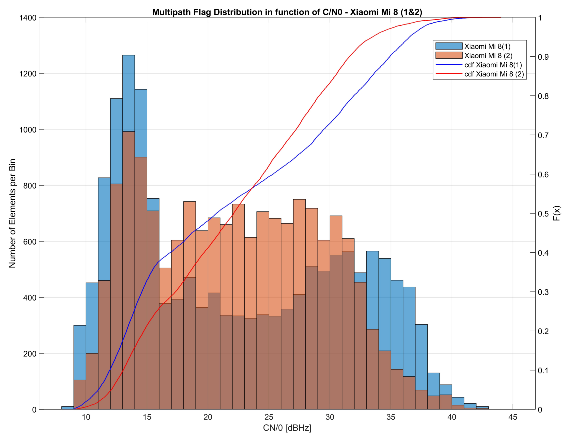


Figure 3.24: Multipath Android Flags Detection Histogram Distribution for the two Xiaomi Mi 8

L5 and E5a signals have been recorded with higher C/N0 values by the geodetic antenna, proving that smartphone's linearly polarized antenna is not optimized for recording GNSS signals.
Figure 3.24 arguments the hypothesis that the smartphone embedded antenna does not properly mitigate multipath interferences. This figure plots the histograms distribution of Android multipath flag events in function of C/N0. The red histogram represents the Xiaomi Mi 8 phone recording measurements with its original antenna, whereas the blue histogram represents the distribution of multipath events for the hacked phone. It is reminded, that regardless of the inconsistency of the Android multipath flag detection algorithms (see section 3.3.1.b), the comparison between both phones holds since both device use the exact same detection method. The distinction in multipath detection events between both Xiaomi Mi 8 phones is clearly visible. Multipath events have been detected more often on retrieved signals between 15 and 30 dB/Hz from the device running the standard linearly polarized patch antenna. This observation is backed up by a similar comparison between both tested Honor View 20, that did not show any differences in multipath events distribution. Those figures are exposed in annex C.

3.3.4.c Android Measurements Suitability for Collaboration

The presented throughout analysis allowed us to better understand smartphones' integrated chipset "black box" processes. The characterization methodology set during our data campaign analysis enlightens differences in positioning performance between different smartphones. The estimation of key positioning performance parameters allowed us to quantify those differences for both nominal and urban environments. Multiple innovative research studies conducted here allow to precisely characterize smartphone embedded chipset receiver. Android GNSS raw data measurements have been defined as suitable and reliable for collaborative positioning usage. The statistical analysis made on receiver-based error model will be used to simulate and implement innovative collaborative positioning techniques. An assessment methodology has been developed for creating a baseline method for characterizing any smartphone GNSS measurements. In urban environment, smartphone's embedded GNSS receivers exceeded positioning performance expectations. Indeed, it turns out that smartphone positioning solutions have similar accuracy level of low-cost GNSS receiver such as the u-blox M8T. However, the smartphone embedded antenna remains the main flaw in this hardware architecture design.

3.4 Chapter Conclusions

This chapter explored the past, current and future smartphone-based positioning techniques. After the release of Android GNSS raw data measurements, multiple research studies were launched for evaluating the implementation of advanced GNSS algorithms. Well-know advanced positioning algorithms, such as PPP and RTK methods, have been extensively studied in the literature and their implementation remain under-development for smartphone-based positioning. Therefore, it was ambitioned to develop a collaborative network positioning system between smartphone users. This technique has been selected for its pertinence and its adaptability to smartphones' hardware components. Thus, a data collection campaign has been successfully carried out for characterizing smartphone GNSS raw data measurements parameters in the interest of developing a smartphone-based collaborative network. Our analysis highlighted smartphones capabilities in nominal and urban environment. A characterization method was presented as a benchmark method for testing and analyzing Android GNSS measurements. Those innovative research studies contributed to the better understanding of Android embedded GNSS receiver operations within the scientific community.
Following the thorough review of smartphone-based positioning technique state of the art, we aimed at developing a smartphone-based collaborative network in order to enhance urban positioning. Our goal will be to take advantage of the increasing number of smartphones in today's city center for developing a safe and secure network of smartphone users exchanging GNSS data for increasing positioning performances. The following part of this research thesis will elaborate on the existing collaborative methods and how they can be transferable for smartphone purposes. Then, a proposed collaborative method will be implemented, tested and validated for smartphone-based urban positioning.

Part II

Collaborative Positioning

4

Definition & Methodology

Contents

The previous research part, on Android smartphone positioning, thoroughly introduced the theory behind smartphone-based positioning. After the presentation of smartphone positioning engine architecture, Android recorded GNSS measurements have been studied in open-sky and urban environments during a post-processing analysis of our data collection campaign. The literature reviewed revealed that the implementation of classical advanced positioning algorithm was unsuitable for smartphone positioning. Cooperative techniques aims at improving network's users positioning performance by processing additional shared GNSS measurements. Therefore, collaborative positioning was favored to be developed for improving smartphone urban positioning performances. This choice was motivated by two main factors. Firstly, the increasing number of connected Android devices in modern city centers is an undeniable advantage for developing a cooperative network. The abundance of sharable data is expected to mitigate constraints linked with smartphone positioning in urban conditions. Secondly, collaborative networking on Android devices will also benefit from the hyper-connectivity of smartphone devices. This implies that a safe and secure communication link will always be available to network's users for sharing data. Expectingly, the development of a smartphone-based collaborative network will significantly increase Android devices positioning performances and reliability.

This chapter is the preamble of the second part of this research thesis. Collaborative positioning techniques will be thoroughly reviewed before proposing an innovative smartphone-based collaborative algorithm. In the first place, collaborative positioning will be introduced and defined. Then, existing network architectures, the nature of exchanged data and communication links will be described. Afterward, classical implementation levels for cooperative positioning are exposed. Finally, our proposed smartphone collaborative network will be presented and argumented.

4.1 Introduction

Collaborative Positioning (CP) allows for the sharing of key information across a network of authorized users for improving positioning performance criteria. CP methods are often referred to as augmentation techniques, used for improving receiver performance [71] [72]. Fundamentally, the study of cooperative algorithms consists in exploiting aiding data from users within the network infrastructure in order to either complement or replace their own set of data allowing for faster computation and/or higher reliability and accuracy solutions. GNSS performance criteria has been defined by civil aviation regulation authorities as follows:

- **Accuracy**:
  This criterion describes the exactness of the estimated position compared to the true position. This quantity is often expressed as a statistical measure.
- **Availability**:
  This parameter represents, in percentage of time, the usability of the service by the user in the defined coverage area.
- **Continuity**:
  The continuity criterion of the GNSS consists of the ability of the system to perform its function for the intended period of time.
- **Integrity**:
  The integrity performance parameter symbolizes the trust parameter that we put into the obtained position. In Civil Aviation, this parameter measures the ability of the receiver to send timely warnings when the system is not reliable.
- **Time to First Fix**:
  This performance benchmark measures the time needed by the receiver to obtain the first estimated position from the moment the receiver is turned on. The parameter is used in order to compare receivers performance.

CP methods are defined and characterized by three main parameters: network architecture, nature of exchanged data and by the CP implementation level in the processing chain [71]. The sharing of information data is organized around the network architecture. It defines pathways between network's users for establishing a framework for exchanging useful data. The nature of exchanged data needs also to be accurately defined for characterizing CP structure. Subsequently, a safe and secure communication link is required to be identified and implemented for securing network transmissions. Our first approach for introducing CP methods will be to compare similarities with well-known GNSS aiding technique in the smartphone domain.

4.1.1 Comparison Against Traditional Methods

In this chapter, traditional augmentation technique are referred as a method that aims at improving GNSS receiver performance using various techniques. This aiding technique provides assistance and correction to a specific user. Typically, three categories of augmentation systems are defined: Satellite-based, ground-based and DGNSS. All those aiding methods share similarities with cooperative positioning algorithms, they aim at assisting a pre-defined GNSS receiver via computational or correction mechanisms for improving final positioning performance.
AGNSS is characterized as a receiver augmentation service, as defined in section 3.1.1.a. This assistance method is well implemented on smartphone device and constitute a perfect benchmark for comparing and analyzing collaborative positioning algorithms. CP and AGNSS methods shares common characteristics and attributes, such as:

- The AGNSS and CP methods share the same goal by aiding an user through performance criteria improvements (improving accuracy and TTFF).
- The two aiding methods provide a multi-constellation aiding service. Even though, the AGNSS service results from the Assisted GPS (AGPS) method [12], the standardization of this method to other GNSS constellations is not yet completely implemented.

However, cooperative positioning stands out in multiple ways. The differences with AGNSS are depicted below:

- AGNSS provides GNSS data only aiding. In contrary, CP method allows for the exchange of data from other sensors combined with GNSS data (e.g: IMU, Bluetooth, WiFi and more).
- The exchange of data within the CP protocol can be made while taking into account local environment. Indeed, method could be implemented for predicted and describing the user environment. Thus, making CP techniques interesting for urban positioning. On the opposite, AGNSS correction service are not specifically adapted for urban navigation and positioning.
- Specific CP structure can be implemented without additional hardware. Peer-to-peer CP does not require the use of an infrastructure outside the network, whereas AGNSS requires a global infrastructure to distribute information to the users.

4.2 Network Architecture

The first characteristic of CP to be explored is the network architecture. It exists two distinctive collaborative network architectures. The first one is called a Peer-to-Peer (P2P) network and is depicted on the left side (a) of figure 4.1. The theoretical principle of this network is based on the free sharing of data between authorized network's users. The second architecture, illustrated on the right side (b) in figure 4.1, is described as a centralized network. As opposed to the previous principle, a centralized server gathers all the information and data from the network back into a server. In this centralized process unit, the data and informations are processed and evaluated before being sent back into the network. The main difference between the two architectures is that users directly interact with each other in a P2P solution; whereas in a centralized network, users only communicate with a centralized unit. This processing unit could take the form of a cloud computing server, taking over the required computational power. The centralized network architecture is represented in figure 4.1 b) on the right.

4.2.1 Peer-to-Peer (P2P)

The P2P collaborative network is, in opposition to the other type of network, a decentralized communication web. The principle of this network, is that every users are able to share and communicate with every other users. The main advantages of this sharing technique is that it allows great scalability of the structure and the creation of an heterogeneous temporal network. On the other hand, the downside of this architecture is that shared data can be unreliable in certain cases and that a quality monitoring chain cannot be implemented. Moreover, all data processing power and computations are now dependent on individual user devices. In [73], a P2P network structure have been used to reduce the Mean Acquisition Time (MAT) of peers inside the network by sharing GNSS aiding data.

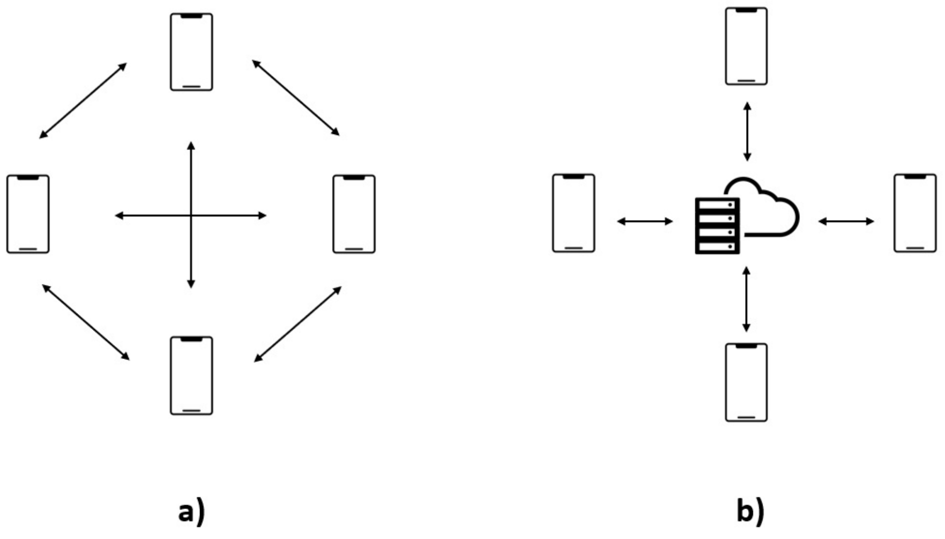


Figure 4.1: Collaborative Network Architecture Representations. a) CP P2P Network Architecture. b) CP Centralized Network Architecture

4.2.2 Centralized Network

A centralized collaborative network consists of peers communicating with a central processing unit gathering data and information. The main advantage of this exchange method is to outsource computational power to a dedicated server resulting in battery and computation power saving for every users of the network. Furthermore, the centralization process allows for integrity and quality checking of the data given by network's peers. The server unit will gather all the data information, check for redundancy, and make enhanced data solution available to peers in the network. Many well-known applications utilize server-based networks. Cloud-based snapshot GNSS receiver can be used as an example to illustrate centralized network techniques. The goal of this techniques is to outsource all the computation to a remote server. The snapshot receiver will simply operate only for a very short amount of time, collecting signal digital samples, before sending these data to the cloud for the signal processing and position estimation processes [74] [75]. Furthermore, a framework for centralized augmented system for smart cities has been developed and tested in [76] demonstrating the efficiency of a dedicated centralized network architecture.

4.3 Exchanged Data Packets

The nature of exchanged data within a CP algorithm can take various forms and are organized in data packets which are then exchanged within the collaborative network. In this study case, GNSS measurements are in the center of this exchange process. Other data and information types can also be exchanged and constitute added benefits for assisting the estimation process of collaborative positions. Two types of data packets configurations exist for characterizing the nature of exchanged data in any collaborative network. The first configuration, described as required, encapsulates GNSS-only observables. The second configuration regroups GNSS observables plus the addition of various measurements, called hybrid data packet.

4.3.1 Required Packet Configuration

GNSS-only data type can be transmitted via a CP network. The nature of this exchangeable data type is strictly limited to the transmission of GNSS observables between network's user. The following non-exhaustive list presents the main GNSS-only data exchanged in CP methods.

- Signal strength (Carrier-to-noise ratio C/N0)
- Pseudoranges
- Doppler measurements
- Carrier phase measurements
- Visible satellites (numbers and identification (PRN))
- Time tags
- Estimated user position

Analysis and tests have been performed in [77] for assessing CP P2P network sharing GNSS-only data. It has been confirmed that users in deep urban environment benefited from the implementation of such positioning technique. However, simulation results demonstrated that GNSS-only data exchange was not optimal for improving receivers positioning solutions in a small collaborative networks.

4.3.2 Hybrid Packet Configuration

An hybrid set of exchanged data is now considered as the most efficient data type for a given CP network. An hybrid dataset contains multiple data products outputted by network's receivers. This configuration is constructed around the transmission of GNSS measurements, similar as the one described in the subsection above. Additionally, extra receiver's data are included with the exchanged dataset such as: Terrestrial ranging measurements between peers, Bluetooth, WiFi, IMU data and more. Modern receivers, especially smartphone devices are now equipped with multi-sensors technology that can easily record and exchange this data, giving insight about their local environment and or peer's location.

4.4 Communication Link

The selection and creation of a communication link within a CP network represent the backbone for the successful implementation of this advanced positioning method. The communication method can be independent from the CP algorithm. Designated communication channel are proven to be safe, fast and reliable. Users information and data are required to be anonymously transmitted to other members of the network in order to ensure user data protection and privacy as stipulated by European Union laws. Communication between users are preferred to be fast and with low-latency thus not affecting the aging of transfered data. Finally, a reliable channel ensure the integrity of exchanged data packets and prevents data packets losses during data transmission.
Multiple communication channels are available and are suited for the exchange of data. Short range communication protocols have been demonstrated to be the most efficient link supporting low-latency data transmission [78]. Ultra-Wide Band (UWB) communication method are not taken into consideration in this study due to their characteristics incompatible with smartphone CP. Dedicated Short-Range Communication (DSRC) standard have been developed for vehicular cooperative network. The Institute of Electrical and Electronics Engineers (IEEE) is the official organization responsible for the standardization of communication procedures. The DSRC has been standardized as IEEE 802.11p. DSRC appears to support data rate from 3 to 27 Mb/s with a range of about 300 meters [79]. Another popular communication link that can be envisaged for CP techniques is WiFi protocol. It is defined by the IEEE 802.11b protocol in a similar way as for DSRC. Finally, cellular network have been also considered as pathway for sharing data among network's user and defined as Ultra-Reliable Low-Latency Communication (URLLC) [80].

Assessing the characteristics and performance level of a chosen communication link falls outside the scope of this research. During the remaining part of this thesis work, we will assume that a safe, fast and reliable communication link exists and can be used for collaborative exchange of data. This hypothesis is judged to be realistic considering the numerous communication channels available to smartphone users.

4.5 Implementation Strategies

CP methods can be implemented on different receiver levels. Two techniques for exploiting specific GNSS are reviewed: Physical and the range layer. Physical layer CP implementation refers to aiding techniques aiming at improving receiver's processing stages (DLL and PLL tracking procedures). On the other hand, the range layer covers aiding techniques and methods applied during PVT estimation via the sharing of GNSS observable data.

4.5.1 Physical Layer

This section will explore the different implementation opportunities to improve one or more users position estimation by applying methods at the physical layer (at the signal processing block) level of the GNSS receiver.
This method will be applied just before starting the acquisition phase, by receiving information about: Doppler frequency, Code delay and C/N0 from different peers in the network. The aided user is expected to significantly reduce its acquisition time [71]. However, time synchronization between individuals in the network is needed and can be achieved by following conventional approaches with limited complexity that can be used for low-cost GNSS receivers. The following list shows examples of *physical layer* CP techniques.

- **Aiding with Doppler Frequency**
  Peers among the network share estimated doppler frequency for each visible satellite seen by their receiver. This information would allow the aided receiver to decrease the size of the search space, thus decreasing the number of frequency bins needed for the computation of the cross-correlation function and finally leading to a decrease of the receiver's acquisition process time.

- **Aiding with Code Delay**
  Code delay data exchange could also allow a user to reduce its own acquisition time. Having information about neighbors receiver's code delay grants the aided peer to possibly remove data bit transitions while comparing received signal samples and locally generated replicas [71].

- **Aiding with C/N0**
  Aiding with C/N0 can be achieved by exchanging this parameter within the network. The aided receiver can decide to acquire the satellites with only a high value of C/N0. On the other hand, satellites with expected low C/N0 can be acquired by increasing the integration time and by choosing the appropriate number of coherent and non-coherent integrations in order to improve the detection ability. [71]

4.5.2 Range Layer

CP implementation methods at the range layer require aiding quantities that can be used for integration and complexity reduction of the PVT computation process (presented in B). It is assumed that an hybrid data exchange is possible between users inside the network and that each of them possess a terrestrial ranging capability. The importance of computing inter-user distances is further explained in chapter 5, as it is the first step toward the development of CP techniques. Examples of range layer CP methods from the literature review are given below.

- **Terrestrial Ranging used as N+1 Satellite**
  This technique uses an extra information, defined as a terrestrial ranging measurement between two users. The estimated distance between two receivers can be used as an extra pseudorange distance from a N+1 satellite. However, this method is limited by additional errors made during the estimation of terrestrial ranging and in the propagation channel. In order to implement this concept, an error estimation parameter must be transmitted as well as the estimated positions of peers in the network, as experimented in [81].

- **Virtual Satellites & Pseudoranges**
  Part of the hybrid shared data are the pseudoranges and the visible satellites of different users in the network. The virtual satellite method takes advantage of those extra information received by the aided peer. Satellite visibility and pseudoranges given by other peers can allow the aided user to obtain an expected satellite position that cannot be seen by its own receiver. From that estimated satellite position, the aided peer receiver can calculate a virtual pseudorange from this particular virtual satellite by estimating the distance between him and the aiding peer. Thanks to this method, users can obtain a better satellite coverage and decrease his geometrical dilution of precision therefore enhancing his receiver accuracy. This particular techniques is described in [82].

- **Collaborative Dilution of Precision**
  The Collaborative Dilution of Precision (CDOP) has been defined by [83] as a theoretical performance analysis parameter. This parameter takes into account the number and the repartition of all the users inside the network and the accuracy of the terrestrial ranging computations. As the name indicates, this quantity is similar to the Dilution of Precision (DOP) for a collaborative network.

4.6 Smartphone Collaborative Network Methodology

This section will present our proposed smartphone collaborative network. Our presented network have been thought out to take into consideration smartphone limitations and strength associated with those connected devices. The previous CP state of the art review revealed that any collaborative networks are characterized by three main parameters: Architecture, data exchanged and communication link. Our presented smartphone collaborative network will be articulated around those three main axes. Before the description of our smartphone cooperative algorithm structure, previous literature works will be studied for highlighting smartphone collaborative networking strengths.

4.6.1 Literature Review

CP algorithm started to spark interest during the last decade and was first implemented in the robotic domain. Recently, collaborative techniques were widely studied for autonomous driving applications. This type of application requires high-accuracy level that can be achieved by pairing a fleet of vehicles within a designated cooperative network [84]. CP methods are used for exploiting Vehicle-to-Vehicle (V2V) communication in order to improve overall targeted vehicle position. These approaches are similar to what we want to achieve with smartphone-based collaborative networking. However, very few papers and research projects explored cooperative approaches on Android smartphone devices. In [85], an analysis of three measurements sharing methods has been proposed in the smartphone positioning domain in future cooperative approaches. Exchanged data were simply used to increase the number of observations of one specific target smartphone. Cooperative algorithms have been studied by [86] via clustering techniques leveraging measurements based on fault detection mechanisms. Moreover, a project called *Flamingo* developed a high accuracy techniques applied to the smartphone domain using a centralized network. Preliminary results exposed in [50] showed promising results and comfort our idea to implement a centralized collaborative network for smartphone positioning.

4.6.2 Network Configuration

The main objective of our smartphone-based collaborative network will be to improve one or more user smartphone's positioning performances. Improving positioning performance can be subject to the enhancement of the four parameters used to describe GNSS receiver performance which are accuracy, availability, continuity and integrity. Limitations and strengths of smartphone-based positioning, showed in chapter 3, will be put in the center of our proposed network structure. The goal for our collaborative smartphone-based network is to establish a cooperative structure taking advantages of smartphone's capabilities and volume in today's dense city centers. We envision a low-cost structure built around Android mobile phones positioning capabilities, scalable to the size of a city center. Our network must be accessible easily to multiple users. The following describes CP parameters chosen for our smartphone-based cooperative network.

- **Network Architecture**
  The selected network architecture for smartphone-based CP technique is a centralized network implementation. There are two main advantages to choose this specific architecture. First, similar network architecture have been successfully employed in the smartphone domain. Secondly, the central processing unit will be in charge of the computational tasks associated with CP algorithms. Smartphones will transmit their data set to the central unit and receive back an updated collaborative position estimate.

- **Exchanged Data**
  The nature of exchanged data between the processing unit and the users will be defined as hybrid. The shared data set is expected to contain Android GNSS raw data measurements plus additional data components such as IMU or barometer measurements. The data frequency of any recorded data will be limited to 1 Hz due to the data recording frequency of Android GNSS raw data measurements. Figure 4.3 shows the proposition made for an hybrid data packet template for a smartphone-based collaborative network. This data format highlights all useful sharable smartphone data, including multi-constellation and multi-frequency GNSS measurements as described in this thesis.

- **Communication Link**
  The data exchange link will be provided by a DSRC communication channel. Indeed, smartphone offers multiple communication channels that can be used for transferring data packets to the remote centralized server. A client-server type of service can be

developed between multiple connected smartphones that are registered throughout a predefined access point. Similarly, cellular network technology (4th generation and the upcoming 5th generation i.e: 4G and 5G) can be exploited as low-latency communication channels using Ultra-Reliable Low-Latency Communication (URLLC).

Our CP technique will be implemented on the range layer, given that smartphone only allows to retrieve GNSS raw measurements. The selected collaborative positioning network will regroup a variety of smartphone users exchanging Android raw data measurements. An accurate ranging method needs to be put in place for estimating vectorial links between all network's users and thus creating a cooperative connection among them. Collaborative ranging methods is explained and argumented in chapter 5.

The block diagram in figure 4.2 highlights the proposed CP method structure. Our collaborative network is constructed around the users that exchange measurements. In the collaborative structure, each agent send the hybrid data packet to the central processing unit. This cloud-based processing regroups the software-based cooperative engine. Our collaborative algorithm can be described by three main steps. The first step is a measurements pre-processing block that aims preparing the hybrid data packets sent by each smartphone users. Afterward, the estimation of such vectorial links is expected to be computed by the central processing unit leveraging data shared by users within the network. This estimation method is referred as the Inter-Phone Ranging (IPR) technique in the present dissertation. Afterward, the final collaborative users positions are estimated based on an optimization process constrained by the aforementioned vectorial links. Finally, the newly estimated collaborative position are sent back to the participating users with the cooperative network. Collaborative smartphone users will be receiving their own updated positions without collecting any data from other agents. The main goal of our collaborative network is to improve users individual positions through data sharing and processing.

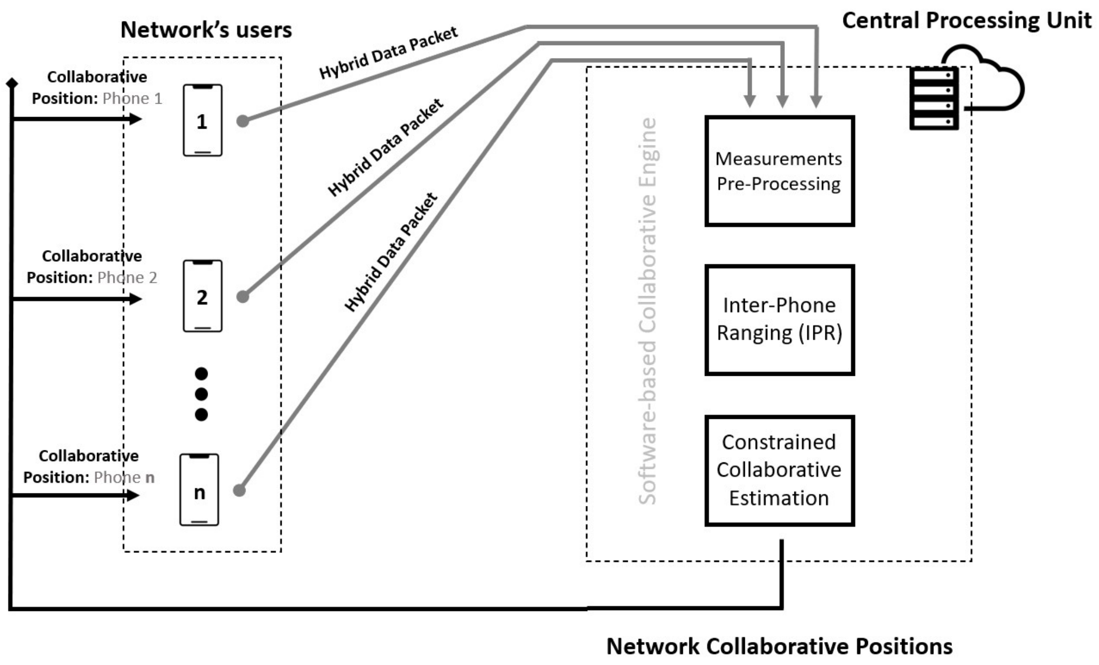

Network's users
Central Processing Unit
Collaborative Position: Phone 1
Collaborative Position: Phone 2
Collaborative Position: Phone n
1
2
n
Hybrid Data Packet
Hybrid Data Packet
Hybrid Data Packet
Software-based Collaborative Engine
Measurements Pre-Processing
Inter-Phone Ranging (IPR)
Constrained Collaborative Estimation
Network Collaborative Positions

Figure 4.2: Smartphone-based Collaborative Network Structure Block Diagram

4.6.3 Software-based Collaborative Engine

Our collaborative estimation engine is articulated around three main axes.

1. Measurements Pre-Processing
   Hybrid data packet are being prepared by the pre-processing block. This preparation includes the formation of Android raw data measurements as described in section 3.3.1.a.

2. Inter-Phone Ranging (IPR) Method
   The IPR method allows to compute vectorial links strictly based on Android raw data measurements. These measurements are estimated through the processing of code double difference. This method is presented in section 5.2. The obtained baseline vector estimates are employed to constraint the set of users position for estimating collaborative positioning solutions

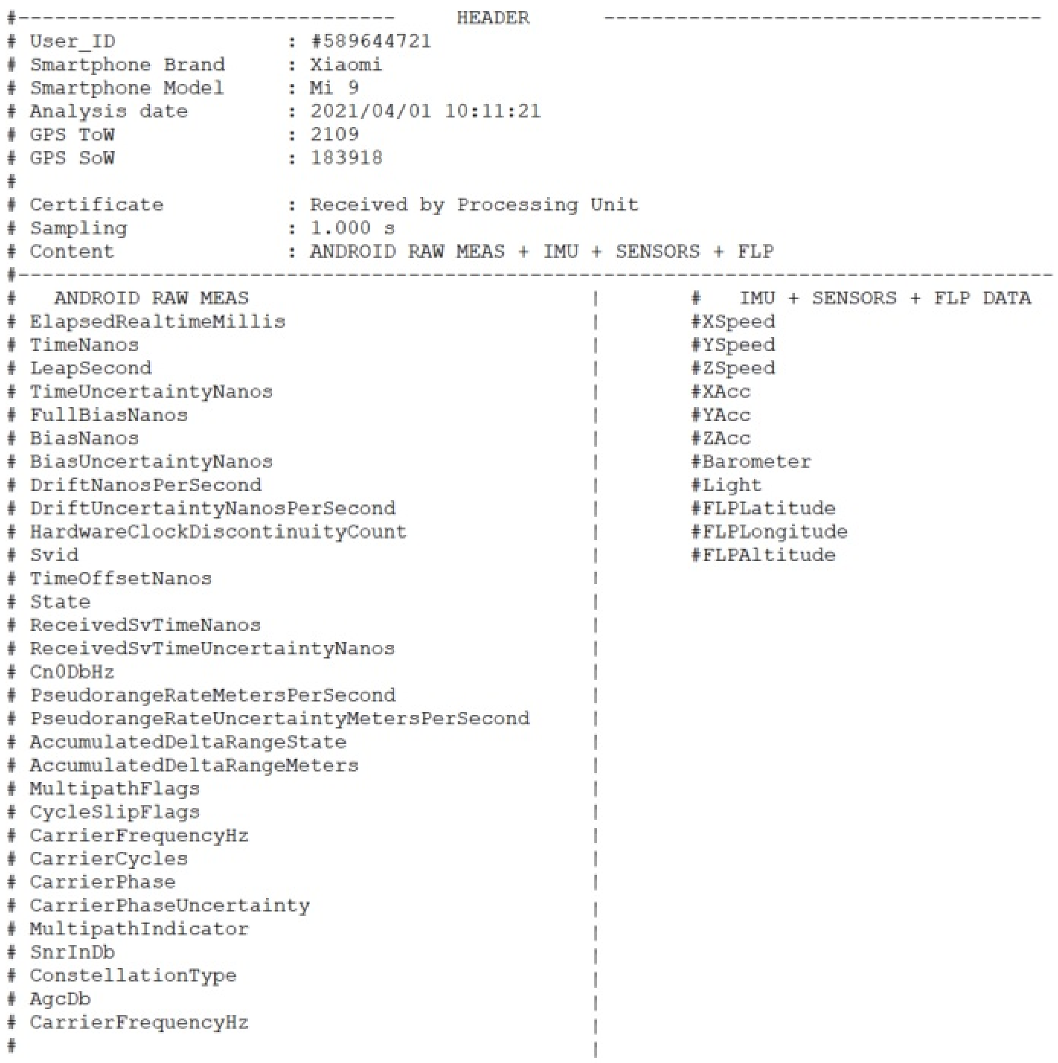
3. Constrained Collaborative Estimation Technique
   The collaborative engine is described as an optimization technique that minimize the positioning discrepancies between newly estimated collaborative positions and their original fix positions. This optimization process is constrained by the previously estimated vectorial links, IPR vectors. Our collaborative engine is defined in details in chapter 4.

4.6.3.a Discussion on the Selected Collaborative Engine

Our collaborative estimation technique is based on a Maximum Likelyhood Estimation (MLE) method. MLE are well suited for collaborative positioning as shown in [87] and [88], respectively. On the account of the innovative aspect of smartphone-based collaborative estimation technique, the selected estimation method has been derived from the automotive industry. Our collaborative estimation techniques is defined as a constrained non-linear optimization problem solved via an MLE estimation method. This technique allows for fast computation process without increasing computational complexity and additional hardware that can be run on a centralized server. One of the drawbacks of using MLE methods for CP is that due to the nature of multi-smartphone geometries, multiple local minima can be present in the final solution. Nevertheless, during simulation analysis of our collaborative solution, the minimization process was achieved 99% of the time.
Recent studies made in [14] explored other estimation methods applied to CP methods. The study of a statistics-adaptive sequential Bayesian estimation filter, a.k.a *Cognitive Particle Filter*, shows promising results compare to MLE estimation techniques. Indeed, the recursive integration of shared GNSS observations and inter-user distances enhanced positioning solution accuracy. However, Bayesian estimation filters require a-priori knowledge of measurements statistical distribution that cannot be guaranteed for smartphone-based measurement. In fact, this phenomenon has been highlighted by our data collection campaign analysis that demonstrated that Android measurements characteristics vary between smartphone brands and models. Measurements statistical error distribution used in Bayesian techniques would bias the final cooperative solution.

4.7 Chapter Conclusions

This chapter presented the state of the art review of collaborative positioning. A characterization method has been recovered and applied for designing an innovative smartphone-based collaborative network. Literature assessment showed cooperative work at the range layer (i.e: in the positioning domain) was possible. Moreover, range-based collaborative positioning is heavily supported by the estimation of inter-user distances within the network. From this first evaluation, our proposed smartphone-based collaborative network was revealed. This cooperative network has been thought out to take into consideration smartphones constraints and advantages. The hyper connectivity of smartphone devices allow for a reliable communication channel to be exploited for CP methods. The following chapter will introduce *Inter-Phone Ranging (IPR)*, a cooperative user ranging method dedicated to smartphone.



```
#--------------------------------     HEADER     ------------------------------------
# User_ID              : #589644721
# Smartphone Brand     : Xiaomi
# Smartphone Model     : Mi 9
# Analysis date        : 2021/04/01 10:11:21
# GPS ToW              : 2109
# GPS SoW              : 183918
#
# Certificate          : Received by Processing Unit
# Sampling             : 1.000 s
# Content              : ANDROID RAW MEAS + IMU + SENSORS + FLP
#-------------------------------------------------------------------------------------
#   ANDROID RAW MEAS                             |       #   IMU + SENSORS + FLP DATA
# ElapsedRealtimeMillis                          |       #XSpeed
# TimeNanos                                      |       #YSpeed
# LeapSecond                                     |       #ZSpeed
# TimeUncertaintyNanos                           |       #XAcc
# FullBiasNanos                                  |       #YAcc
# BiasNanos                                      |       #ZAcc
# BiasUncertaintyNanos                           |       #Barometer
# DriftNanosPerSecond                            |       #Light
# DriftUncertaintyNanosPerSecond                 |       #FLPLatitude
# HardwareClockDiscontinuityCount                |       #FLPLongitude
# Svid                                           |       #FLPAltitude
# TimeOffsetNanos                                |
# State                                          |
# ReceivedSvTimeNanos                            |
# ReceivedSvTimeUncertaintyNanos                 |
# Cn0DbHz                                        |
# PseudorangeRateMetersPerSecond                 |
# PseudorangeRateUncertaintyMetersPerSecond      |
# AccumulatedDeltaRangeState                     |
# AccumulatedDeltaRangeMeters                    |
# MultipathFlags                                 |
# CycleSlipFlags                                 |
# CarrierFrequencyHz                             |
# CarrierCycles                                  |
# CarrierPhase                                   |
# CarrierPhaseUncertainty                        |
# MultipathIndicator                             |
# SnrInDb                                        |
# ConstellationType                              |
# AgcDb                                          |
# CarrierFrequencyHz                             |
#                                                |
```

Figure 4.3: Proposition of an Hybrid Data Packet Template for Smartphone-based Collaborative Network

5 GNSS-based Collaborative Ranging

Contents

Our proposed collaborative network exposed in 4.6 is dependent on a ranging method that estimates vectorial links between smartphone's users within the network. In the literature, baseline estimation methods between users has been studied extensively. This problem is traditionally referred as *baseline estimation* [24] [89], alternatively in cooperative works it is also known as *inter-agent ranging* or *inter-user distance* estimation [90] [91]. The aforementioned *baseline* refers to a distance vector between users, whereas the *baseline length* is a scalar value representing the norm of that vector. The scope of this chapter will be to assess GNSS-based baseline estimation techniques to be applied for smartphone collaborative positioning.
Firstly, fundamentals of baseline estimation will be defined mathematically. Then, a state-of-the-art review on baseline computation methods will be displayed with a focus on GNSS-based solutions. Afterward, our proposed inter-agent estimation method called *Inter-Phone Ranging (IPR)* will be presented. This method generates 3D vectorial links that are required by our proposed smartphone-based collaborative positioning algorithm. Finally, the Inter-Phone Ranging (IPR) method will be evaluated in both open-sky and urban environment cases.

5.1 Collaborative Ranging: Fundamentals

Collaborative ranging is defined as a general method for obtaining range information of any form, either vector or scalar, using data exchanged by multiple sources within a network. This section provides an overall introduction to ranging methods and characterizes the differences between range vector and the range distance. A literature review including various methods for obtaining range information is firstly presented. As defined previously in section 4.6.3, our proposed collaborative network is built around the estimation of a vectorial quantity for linking mobile users. Therefore, the focus is put on a double difference technique that is commonly applied to DGNSS methods.

5.1.1 Ranging Definition

Classical GNSS literature refers to the term baseline as a vector between two receivers [24]. Historically, this notion is introduced in advanced positioning algorithms (i.e: DGNSS, PPP and RTK) for estimating the vector from a GNSS base station to a roving receiver. In this case, a ranging method is employed to estimate the baseline vector. The standard mathematical expression for baseline computation is given below.

5.1.1.a Mathematical Definition

Two independent users, denoted as A and B, are defined here. Their true position coordinates in ECEF reference frame is represented in equation 5.1. Each position is expressed at a given time t.

$$P_A(t) = \begin{bmatrix} x_A(t) \\ y_A(t) \\ z_A(t) \end{bmatrix} \text{ and } P_B(t) = \begin{bmatrix} x_B(t) \\ y_B(t) \\ z_B(t) \end{bmatrix} \tag{5.1}$$

The baseline vector between receiver A and B is noted as in equation 5.2.

$$\mathbf{d}_{AB}(t) = \begin{bmatrix} x_B(t) - x_A(t) \\ y_B(t) - y_A(t) \\ z_B(t) - z_A(t) \end{bmatrix} = \begin{bmatrix} \Delta x_{AB}(t) \\ \Delta y_{AB}(t) \\ \Delta z_{AB}(t) \end{bmatrix} \tag{5.2}$$

By definition, the baseline length between the two receivers, $d_{AB}(t)$, is computed by taking the norm of the vector $\mathbf{d}_{AB}(t)$. As a note, in the following mathematical models the bold characters in equations represent vectors. The obtained Euclidean distance is shown in equation 5.3.

$$d_{AB}(t) = \|\mathbf{d}_{AB}(t)\| = \sqrt{\Delta x_{AB}^2(t) + \Delta y_{AB}^2(t) + \Delta z_{AB}^2(t)} \tag{5.3}$$

5.1.1.b Alternative Ranging Techniques

Multiple baseline estimation techniques have been developed over the years. Most of them utilize external hardware components capable of computing accurately distances. Sensor-based ranging has been extensively implemented in vehicular and robotic application domains. UWB, Light Detection and Ranging System (LIDAR) and ultrasonic Sound Navigation and Ranging (SONAR) equipments have been used for providing enhanced performances to CP methods [92]. These methods are essentially based on TOA

approaches and output highly reliable estimate solutions. However, sensor-based baseline estimation techniques present multiple flaws that makes them impractical for smartphone-based CP algorithms [93]. Indeed, the implementation of such methods requires additional hardware equipments that greatly complexify the computational process chain. Furthermore, sensor-based ranging proves to be effective mostly for close-object localization and thus could become inoperative for NLOS distance determination in urban environment.
GNSS-based baseline estimation technique might represent a better alternative for answering and mitigating constraints generated with smartphone CP. Specific software algorithms have been developed for estimating baseline length between two independent GNSS receivers. The main advantage of this technique is to allow the computation of inter-user ranging between multiple users using Android raw measurements data stream. GNSS-based inter-user distance computation is detailed and argumented below.

5.1.2 Absolute Position Differencing (APD)

In the wake of developing collaborative positioning solutions for vehicular application, GNSS-based user distance estimation has been considered. In [94], a state-of-the-art approach allowed to set a typical case study for estimating baseline length between two GNSS receivers. This approach is called Absolute Position Differencing (APD) and is defined as a straightforward application of basic Euclidean distance definition shown in 5.1.1.a. The process is based on differentiating two position solutions derived from two independent estimation procedures of the two selected GNSS receivers. Based on previous definitions, we can set receiver A with an estimated position $\hat{P}_A(t)$ and receiver B outputting $\hat{P}_B(t)$. The remaining individual errors characteristic are represented by ε_A and ε_B, whereas ε_{AB} shows the error characteristics of the difference. Equation 5.3 can be rewritten as in 5.4.

$$\|\hat{\mathbf{d}}_{AB}(t)\| = \sqrt{\Delta\hat{x}_{AB}(t)^2 + \Delta\hat{y}_{AB}(t)^2 + \Delta\hat{z}_{AB}(t)^2} \tag{5.4}$$

with:

$$\begin{cases} \Delta\hat{x}_{AB}(t) = & \hat{x}_B(t) - \hat{x}_A(t) + \varepsilon_{AB_x} \\ \Delta\hat{y}_{AB}(t) = & \hat{y}_B(t) - \hat{y}_A(t) + \varepsilon_{AB_y} \\ \Delta\hat{z}_{AB}(t) = & \hat{z}_B(t) - \hat{z}_A(t) + \varepsilon_{AB_z} \end{cases} \tag{5.5}$$

The estimated baseline length $\|\hat{d}_{AB}(t)\|$ is impacted by the overall uncertainties resulting from the two estimation processes of both GNSS receivers. This error bias can be caused by different events and is recognized to be greater in urban environment. Specific urban environment errors are exposed and discussed separately in appendix B. True inter-user distance can be retrieved if receiver and signal processing errors can be mitigated.

Furthermore, the retrieval of asynchronous measurements could lead to biased inter-user ranging solutions. Time misalignment problematic is represented by equation 5.6. Where t_i and t_j correspond to two different discrete times at which the positions are computed and are defined in the same time frame. In this case, $\delta t = t_i - t_j$ characterizes the time misalignment between the two users in the GPS time frame. In our case, this is validated through a pre-processing step that converts local receiver time into GPS time. In dynamic scenarios, differentiating asynchronous and delayed measurements is directly impacting the quality of the estimated baseline length. Considering user dynamic, measurements synchronization is conceded to be the most important parameter to be considered in inter-user ranging method.

$$\hat{d}_{AB}(t_i) \simeq \|\hat{P}_B(t_i + \delta t) - \hat{P}_A(t_i)\| \tag{5.6}$$

As demonstrated above, this direct GNSS-based inter-user ranging technique is limited by two main constraining factors (i.e: overall positioning errors and asynchronous measurements). Those difficulties are mostly mitigated while applying DGNSS methods.

5.1.3 DGNSS Baseline Computation

In opposition to the previously exposed APD method, DGNSS ranging techniques are based on differencing of raw data measurements instead of differencing receiver positioning solutions. Multiple works investigated the implementation of DGNSS inter-user ranging [95] [96].
This differential technique mainly rely on shared common received signals between two agents. Indeed, the main advantage of DGNSS methodology is to mitigate common propagation errors and satellite clock biases between two users receiving the same signals simultaneously. In practice, the processing of asynchronous measurements is critical in a complex multiple-agent cooperative network. Therefore a Doppler-based time compensation solution is explored in this section.
Inspired by RTK and advanced differential positioning algorithms, various methods have been studied for estimating inter-user baseline vector. Such techniques extend from Raw Pseudorange Ranging (RPR) [95] to Single Difference Ranging (SD) [97] and Double Difference (DD) [24]. An emphasis will be made on DD ranging methods and will be exposed in this section.

5.1.3.a Measurements Data Exchange

As discussed in chapter 4.3, the nature of exchanged data contributes to the definition of any CP technique. The following description of DGNSS baseline computation is based on the exchange and processing of pseudorange measurements of multiple user's receivers. GNSS-only data is thus necessary to be exchanged within the cooperative network. We set our collaborative framework by defining the aiding and the aided agent. In a two user cooperative network, the agent that tries to improve its position through CP techniques will be referred as the aided agent. The second user providing assistance data to any network members will thus be named the aiding agent. The following example will expose a two users network for simplification and illustration purposes. A simple extrapolation can be made for applying similar technique to a bigger multi-user collaborative network.

5.1.3.b Doppler-based Measurements Synchronization

The most crucial processing step in any CP methods is the time synchronization of exchanged measurements. In the context of a multiple users network, the Doppler-based time compensation technique is a prerogative to any collaborative algorithms. An offset between two retrieved pseudorange measurements from two different receivers induces an error that cannot be mitigated by DGNSS techniques. The developed time-synchronization technique was built around Doppler-based compensation algorithms, well known from the GNSS community. This method relies on the assumption that relative movement between the satellite and the receiver remains constant over a short interval of time. This hypothesis holds in the case of smartphone use in low-dynamic (pedestrian and urban navigation) and static scenarios. In high dynamic scenarios, the resulting error from this time-compensation method has been shown to be negligible (e.g: vehicular applications) [98]. Equation 5.7 shows the time-synchronization of a pseudorange ρ at a time t expressed in a common GPS time frame. Time synchronization is achieved when $\rho_A(t)$ has been extrapolated at a time $t+\Delta t$ corresponding to the time of $\rho_B(t+\Delta t)$. The following equation considers only the first order approximation of pseudorange variation. The prevailing assumption is that measurements from A and B are taken in the same time frame.

$$\rho(t+\Delta t) = \rho(t) + (\Delta t \; . \; \lambda \; . \; \dot{\phi}(t)) \tag{5.7}$$

where λ represents the wavelength of the carrier frequency of the retrieved signals. Signal frequency and wavelength are linked according to the following formula $\lambda = \frac{c}{f}$. The term $\dot{\phi}(t)$ characterizes the Doppler frequency shift observed at a time t, estimated by the GNSS receiver. Finally, the associated expression $\lambda \; . \; \dot{\phi}(t)$ stands for the pseudorange rate estimation and is expressed in meter per second [m/s].

5.1.3.c Double Difference Ranging (DD)

After measurements synchronization has been achieved, double differences on code measurements are being processed. Double Difference ranging (DD) operation allows to mitigate common errors shared between users while estimating accurate inter-user distance [22]. The integral computational process of DD ranging is defined as follows and is illustrated by figure 5.1.

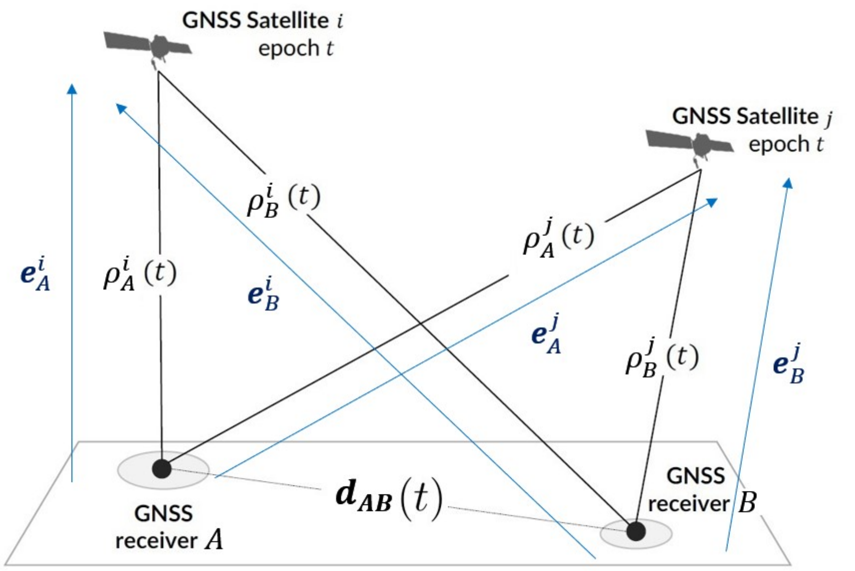


Figure 5.1: Double Difference Ranging Principle Illustration. Picture modified from [13].

At first, a single difference denoted D^i_{AB} is being computed between two agents and one common GNSS satellite signal at a common time. Equation 5.8 shows a single pseudorange (ρ) difference between users A and B with respect to satellite i at the same epoch. The used pseudorange model is defined in equation 3.3.

$$\begin{aligned} D^i_{AB} &= \rho^i_B - \rho^i_A \\ &= \mathbf{e}^{\,i}_B \;.\; \mathbf{r}^i_B - \mathbf{e}^{\,i}_A \;.\; \mathbf{r}^i_A + c \;.\; (t_{rx,B} - t_{rx,A}) + (\epsilon_B - \epsilon_A) \end{aligned} \tag{5.8}$$

where $\mathbf{e}$ are the steering vectors from the receivers to the satellites. We consider that baseline length d_{AB} between both receivers (as defined in equation 5.3) is negligible compared to ρ^i_B and ρ^i_A, thus $\mathbf{e}^{\,i}_A \simeq \mathbf{e}^{\,i}_B = \mathbf{e}^{\,i}$. The single difference D^i_{AB} can be simplified as shown in equation 5.9.

$$D^i_{AB} = \mathbf{e}^{\,i} \;.\; \mathbf{d}_{AB} + c \;.\; \Delta b_{AB} + \Delta\epsilon_{AB} \tag{5.9}$$

At this stage, satellite-based errors are removed including satellite clock bias and satellite position offset. Tropospheric and ionospheric delays are also mitigated in the case that both receivers are located within a short distance. Other non-correlated errors such as multipath and hardware biases remain. The remaining terms in the equation are the difference in geometric range ($\mathbf{r}^i_B - \mathbf{r}^i_A = \mathbf{d}_{AB}$) multiplied by the steering vector, receivers clock offset (Δb_{AB}), and residual errors ($\Delta\epsilon_{AB}$) such as thermal noise and receiver-dependent environmental errors. The single difference reveals the parameter to be estimated: the baseline vector $\mathbf{d}_{AB}$ between two network's users as defined in equation 5.2.

Afterward, a second differentiation operation is made between two single differences of two satellites' signals shared by both users. This phase is referred as a double difference and is denoted as dD. Equation 5.10 presents a double difference between two satellites i and j in view of users A and B. At this step, the double difference process further mitigates receiver dependent error terms including receiver clock offset and hardware bias.

$$\begin{aligned} \mathrm{dD}^{ij}_{AB} &= D^j_{AB} - D^i_{AB} \\ &= (\mathbf{e}^{\,j} - \mathbf{e}^{\,i}) \;.\; \mathbf{d}_{AB} + \Delta\epsilon^{ij}_{AB} \end{aligned} \tag{5.10}$$

The term $\Delta\epsilon^{ij}_{AB} = \Delta\epsilon^j_{AB} - \Delta\epsilon^i_{AB}$ represents the remaining unmitigated errors. A list of m common satellites signals between receivers A and B is established. Each received signal is classified according to signal performance characteristics (i.e: Satellite elevation, signal strength). For illustration purposes, we will define satellite j described in equation 5.10 as our reference signal and will be denoted as signal 1.
Thus, equation 5.10 generalizes as:

$$\mathbf{dD}_{AB} = \mathbf{H}\ \mathbf{d}_{AB} + \boldsymbol{\epsilon} \tag{5.11}$$

where:

- $\mathbf{dD}_{AB} = [dD^{21}_{AB}, dD^{31}_{AB}, \dots, dD^{m1}_{AB}]^T$
  $[(m-1) \times 1]$ vector of double differences for all common signals between both users.

- $\mathbf{H} = [(\mathbf{e}^2 - \mathbf{e}^1), (\mathbf{e}^3 - \mathbf{e}^1), \dots, (\mathbf{e}^m - \mathbf{e}^1)]^T$
  $[(m-1) \times 3]$ matrix of steering vector differences in the ECEF coordinate frame.
- $\mathbf{d}_{AB}$ vector defines the 3D ranging vector between two users.
- $\boldsymbol{\epsilon} = [\Delta\epsilon_{AB}^{21}, \Delta\epsilon_{AB}^{31}, \dots, \Delta\epsilon_{AB}^{m1}]$
  $[(m-1) \times 1]$ vector of the double difference measurement residual error terms including those caused by receivers thermal noise and multipath effect. Note that each of these residual errors are composed of errors originated from measurements coming from two satellites and two receivers and are expected to be greater than the original undifferentiated measurements

Finally, the 3D baseline vector $\hat{\mathbf{d}}_{AB}$ can be estimated via a weighted least square solution, described by equation 5.12.

$$\hat{\mathbf{d}}_{AB} = (\mathbf{H}^T \mathbf{W}\ \mathbf{H})^{-1}\ \mathbf{H}^T \mathbf{W} (\mathbf{dD}_{AB}) \tag{5.12}$$

with $\mathbf{W}$ being a diagonal weight matrix constructed based on the inverse of the previously defined $\boldsymbol{\epsilon}$ vector. This matrix is populated by weights coming from various techniques or via apriori knowledge. At least 3 sets of satellite pairs (or 4 common signals) are required to estimate our range.

5.2 Inter-Phone Ranging (IPR) Estimation Method

In the context of smartphone-based urban collaborative positioning, an innovative ranging estimation technique dedicated to smartphone raw measurements processing was developed, this algorithm has been baptized Inter-Phone Ranging (IPR). This method allows to estimate 3D inter-user range vectors and has been inspired by the studies made in [87] [91]. Inter-user range vector is defined as a mathematical vector composed by three algebraic components between two distinctive users that are part of the cooperative network. The IPR method has been developed explicitly around the exchange of Android raw data measurements. At this stage, we assume that a communication link exists and is available to every network's users for a safe and reliable exchange of data. This section will focus on the definition and methodology description of the IPR algorithm, while emphasizing on the specificities for dealing with smartphone-based measurements.

5.2.1 Definition & Terminology

The presented IPR method is a DGNSS based method. This implementation offers accurate estimation solution for processing GNSS raw data measurements. In [23], a first approach on cooperative ranging was established between Android smartphone devices. A double difference method allowed the authors to estimate inter-user baseline vector. Their work was the first stepping stone for establishing a range estimation process between Android devices. Figure 5.2 represents the implementation of IPR technique between two smartphones. The outputted 3D range vector between smartphone A and smartphone B is denoted $\mathbf{IPR}_{3D}^{AB}$.

As detailed previously, the IPR method is defined as the estimation process computing a 3D range vector between two users. The algorithm's resulting vector will be referred as

Inter-Phone 3D Range Vector ($\mathbf{IPR}_{3D}$). The quantity is mathematically defined in the ECEF reference frame as in equation 5.13 and is illustrated in figure 5.2.

$$\mathbf{IPR}_{3D}^{AB} = \begin{bmatrix} \text{IPR}_{3D}^{AB}(x) \\ \text{IPR}_{3D}^{AB}(y) \\ \text{IPR}_{3D}^{AB}(z) \end{bmatrix} \tag{5.13}$$

As a result to the previous definition, the distance between two users can be computed by taking the norm of $\mathbf{IPR}_{3D}$. This distance will be referred as Inter-Phone Range Distance (IPRd) and will be used for assessing a specific aspect of the IPR estimation technique performance.

Finally, the proposed IPR denotes from previous works by the following criteria. Those differences outline the innovation behind the presented method.

- The IPR method was specifically designed to accurately process Android raw data measurements.
- Design of a smoothing technique through the implementation of an Hatch Filter for smartphone-based pseudorange measurements.
- Multi-constellation and multi-frequency signal processing.
- Covariance matrix of WLS estimation technique populated by Android-based measurements.

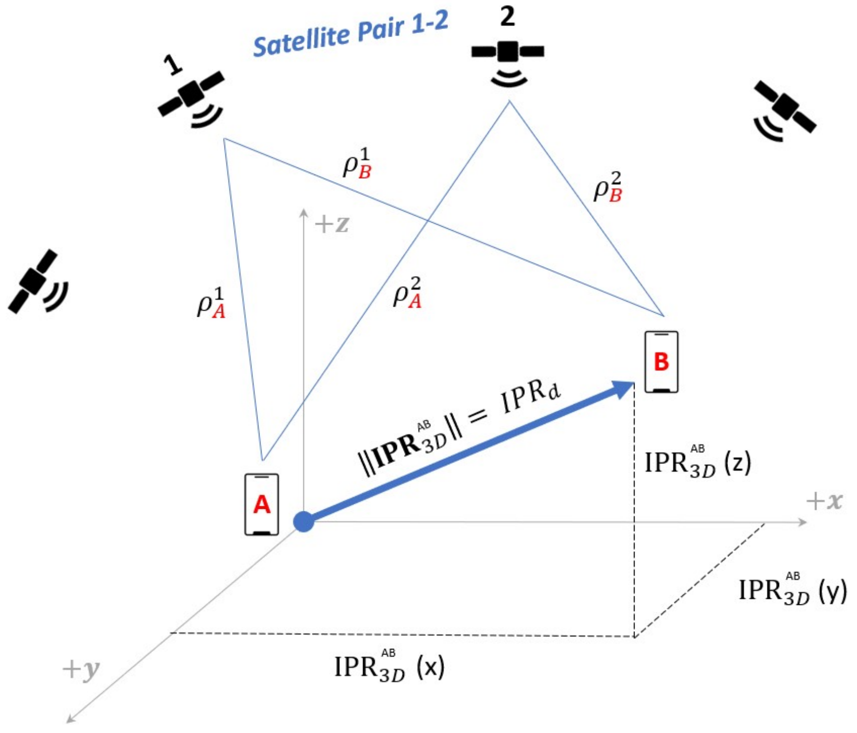

Satellite Pair 1-2
1
2
ρ_B^1
ρ_B^2
ρ_A^1
ρ_A^2
+z
+x
+y
A
B
$\|\mathbf{IPR}_{3D}^{AB}\| = IPR_d$
IPR_{3D}^{AB} (z)
IPR_{3D}^{AB} (y)
IPR_{3D}^{AB} (x)

Figure 5.2: Inter-Phone Ranging (IPR) Method Applied to Two Smartphone Users

- Estimation of a 3D Inter-User Range Vector between two collaborating smartphones, called $\mathbf{IPR}_{3D}$.
- Multi-smartphone brands and models analysis
- A detailed analysis of the IPR method, through testing and studying of $\mathbf{IPR}_{3D}$ and IPRd against other baseline estimation methods for both nominal and urban environments

5.2.2 Methodology

IPR methodology details the procedure followed for implementing smartphone-based ranging technique. The smartphone application domain brings limitation and constraints that needs to be handled by our algorithm. The following guidelines acts as recommendation for processing Android GNSS raw data measurements accurately for ranging computations. For simplification purposes, our methodology presents n satellites to be used as if only one signal was received per satellite. However, today's smartphone embedded positioning chipset are multi-constellation and multi-frequency. Then, multiple signals from one individual satellite can be received at the same epoch. Moreover, the multiplication of signals availability on smartphone favors the constructions of common received signals pairing.

The following subsections will define smartphone-specific implementation steps necessary for developing IPR algorithm. The presented process is divided into the next main stages:

- Formating data
- Establishing a list of common received signals
- Hatch filter
- Doppler synchronization of measurements
- Double difference
- WLS Estimation

This process is a recurrent mechanism incremented for every epochs and for all available smartphone's user pairs. Indeed, IPR estimation needs to be computed for every couple of agents within the collaborative network. The number of pairs to be computed is directly related to the number of network's users n and is given by this expression: $\frac{n(n-1)}{2}$. For example, a collaborative network populated by 5 independent users will generate 10 unique inter-phone ranging vectors, $\mathbf{IPR}_{3D}$.

5.2.2.a Data Exchange

As described above, IPR algorithm is dependent on the data exchanged between users. IPR method is a DGNSS-based algorithm using agent's code pseudorange measurements. The nature of exchanged data within the cooperative network must thus include raw GNSS measurements. As it turns out, Android raw data measurements can be processed, retrieved and exchanged within a collaborative network.
Android raw data measurements are paramount for the computation of $\mathbf{IPR}_{3D}$. Figure 2.10 and 2.11 in chapter 2, remind the denomination for each recorded Android GNSS measurement. Then, the first computation process consists of rendering Android measurements into GNSS observables (i.e: Pseudorange, Doppler Frequency, Pseudorange Rate

and more). Those measurements are processed and stored inside a matrix and constitute the input of our range estimation system.

5.2.2.b Smoothing Measurements

Our analysis made during the data collection campaign exposed the inconsistent behavior of retrieved Android GNSS raw data measurements (see section 3.3.4). This inconsistency is contingent to multiple factors and can vary depending on the smartphone brand or model. This problem is accentuated by urban positioning constraints and smartphone's hardware architecture. Therefore, we implemented an Hatch filter capable of smoothing smartphone's estimated code pseudorange measurements [99]. The characterization of this pseudorange smoothing method is established in equation 5.14 at an epoch t in GPS time. In adequation with Android raw GNSS measurements rate (1 hz), the term t has a time step of 1 second for continuous time segments.

$$\tilde{\rho}(t) = \frac{1}{h}\,\rho(t) + \frac{h-1}{h}\,[\,\tilde{\rho}(t-1) + \lambda\ .\ (\phi(t) - \phi(t-1)\,)\,] \tag{5.14}$$

where h is defined as a moving time window that represents the length of a continuous time segment (in epochs) when a specific signal has been received and correctly retrieved by our embedded smartphone GNSS receiver. Each time a signal loss of track is detected, a reinitialization of the index h is made. This method also allows to account for non-continuous signal segments when recorded signals are not visible for a short period of time or during cycle slip events. The characterization of continuous time segment is derived from the study made on Android flag detection mechanisms in 3.3.1.b. This method mitigates the impact of outliers pseudorange data on the final IPR estimation results.

5.2.2.c Synchronizing Android Measurements

The time-synchronization of pseudorange measurements is a mandatory step in any CP algorithms. The developed time-synchronization technique was built around Doppler-based compensation algorithms, well known from the GNSS community. This method relies on the assumption that relative movement between the satellite and the receiver is constant over a short interval of time. This hypothesis holds in the case of smartphone use in low-dynamic and static scenarios. Equation 5.7 can be rewritten, following the notation of Figure 5.2. Let smartphone user A being the aiding user assisting user B, the aided agent. Time synchronization is achieved when $\rho_B(t_B)$ has been extrapolated at a time t_A from the measurements recorded from the aiding user $\rho_A(t_A)$ at time t_A. The relation between asynchronous time measurements is simply derived as $t_A - t_B = \Delta t$

$$\rho_B(t_A) = \rho_B(t_B) + (\Delta t\ .\ \lambda\ .\ \dot{\phi}(t_B)) \tag{5.15}$$

The term $\lambda\ .\ \dot{\phi}(t)$ representing the Doppler shift can be approximated by the pseudorange rate measurements. This measurement can be easily retrieved from recorded Android data as `PseudorangeRateMetersPerSecond`. Secondly, the generation of the term Δt is straightforward since the Android raw measurement, `FullBiasNanos`, is already synchronized to the true GPS time and is expressed in nanoseconds [7]. Δt is approximated by taking the difference between the two Android measurements cited above.

5.2.2.d IPR Estimation Algorithm

The IPR code differentiation estimation method starts by computing the single and double differences between common received signals between the aided and aiding agents. Once the list of common signals is established, a signal of reference is identified. Usually said signals exhibits high C/N0 values coming from a satellite with a high elevation. Then, single and double differences are taken according to the description made in 5.1.3.c. Equation 5.12 shows the parametrization of a WLS estimation method where the outputted solution $\mathbf{IPR}_{3D}$ was identified as $\mathbf{d}_{AB}$. The construction of the denominated, $\mathbf{W}$, weighting matrix is elaborated via specific Android-based data measurements. Most of the time, in GNSS literature review, the computation of this matrix is made with respect to elevation and C/N0 parameters. However, in the smartphone domain case, the Android API gives to the user a parameter estimating the uncertainty error for each received signal. This data measurement is found in the Android class *GnssMeasurements* and is called `ReceivedSvTimeUncertaintyNanos`, expressed in nanosecond.

$$\sigma_{\epsilon}^{i,f} = \texttt{ReceivedSvTimeUncertaintyNanos}(t)\ .\ 10^{-9}\ .\ c \tag{5.16}$$

Equation 5.16 shows the implementation of the Android data measurements for estimating signal individual error standard deviation for a particular receiver, where i is the satellite index and f indicates the frequency band.
However, for simplification purpose, the following mathematical models consider only signals collected from one frequency band (f = L1).
A variance matrix of single-differenced measurements $\mathbf{R}_{sD}$ is formed and populated by retrieved sigma values, $\sigma_{\epsilon}^{i,f}$, from the Android API. Equation 5.17 is constructed using agent A and B as an example. Note that this matrix has a diagonal form since we assume that the measurements from different satellites are uncorrelated. This matrix will be of the size $(m \times m)$ with m being the number of satellites.

$$\underset{m\times m}{\mathbf{R}_{sD}} = \begin{bmatrix} (\sigma_{\epsilon}^{1}{}_{A})^2 + (\sigma_{\epsilon}^{1}{}_{B})^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & (\sigma_{\epsilon}^{m}{}_{A})^2 + (\sigma_{\epsilon}^{m}{}_{B})^2 \end{bmatrix} \tag{5.17}$$

Afterward, the double differences are taken. This procedure is performed by introducing the following D matrix that subtracts the measurement of the reference satellite from other tracked satellites. As exposed in equation 5.18, the first column is filled with "-1" as it represents the reference satellite [100]. This operation allows to account for the correlation between double differentiated measurements. The number of double difference measurements after this operation is equal to $m - 1$.

$$\underset{(m-1)\times m}{\mathbf{D}} = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 \\ -1 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & 0 & 0 & \cdots & 1 \end{bmatrix} \tag{5.18}$$

The matrix $\mathbf{H}$ contains the steering vectors and is defined as in 5.11. The final covariance matrix of double differenced measurements, written as $\mathbf{R}_{dD}$, is computed by equation

5.19. The weight matrix $\mathbf{W}$ is computed by taking the inverse of the covariance matrix: $\mathbf{W} = \mathbf{R}_{dD}^{-1}$. Finally as defined in the previous equation 5.12, the $\mathbf{IPR}_{3D}$ vector between smartphone A and B is outputted by the updated equation 5.20 in the ECEF reference frame.

$$\underset{(m-1)\times(m-1)}{\mathbf{R}_{dD}} = \mathbf{D}\ \mathbf{R}_{sD}\ \mathbf{D}^T \tag{5.19}$$

$$\mathbf{IPR}_{3D}^{AB} = (\mathbf{H}^T\,\mathbf{R}_{dD}^{-1}\ \mathbf{H})^{-1}\ \mathbf{H}^T\,\mathbf{R}_{dD}^{-1}\,(\mathbf{dD}_{AB}) \tag{5.20}$$

5.3 IPR Performance Analysis

The performance analysis of our proposed IPR ranging technique is exposed in this section. In order to draw a fair comparison analysis of our estimated inter-users ranges, we take into consideration other means of range computation available to smartphone users. As described in section 5.2, $\mathbf{IPR}_{3D}$ is a vector estimated by a WLS based on double differences. In order to compare their performance against the other methods, the norm of the $\mathbf{IPR}_{3D}$ vector is estimated and will be denoted in this section as IPRd. The first process is straightforward, it consists of estimating smartphones position by a WLS algorithm and using the obtained positions to compute a range between two users. The second technique is taking advantages of a unique feature available to Android mobiles' users, the Google FLP position solutions. Those retrieved positions will allow us to evaluate ranges between two smartphone peers. FLP positions are intended to be the ultimate positioning solution

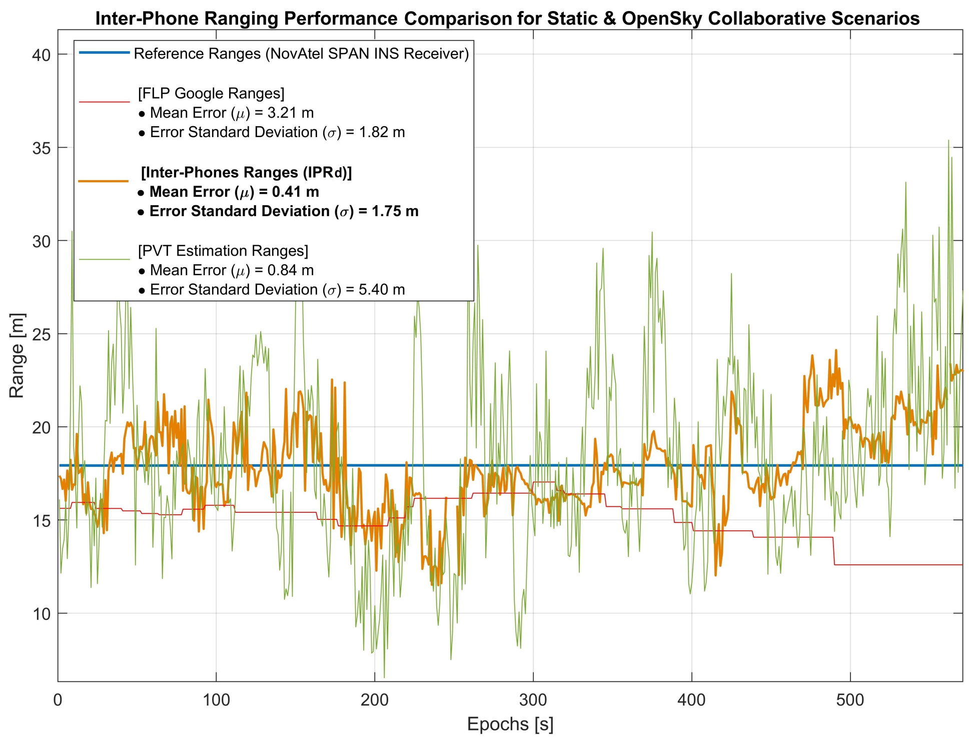


Figure 5.3: Estimated Ranging Error for Static & Open-Sky Environment Scenario

obtainable by a given Android device. However, the black-box processes used by FLP measurements while combining GNSS, cellular network and sensors informations, make them ambiguous and unreliable for scientific analysis. On the other hand, they provide a good reference for intrinsic quality of smartphones positioning capabilities. This reason justifies their use in our performance analysis for ranging purposes. Detailed description of Android's FLP position is given in 3.1.1.b.

Performance analysis provided below has been drawn using the data measurements collected during the data collection campaign presented in section 3.2.1. Collaborative scenarios created during this data collection (exposed in section 3.2.2) will be used to characterize the performance of our estimation technique in various environment, namely in open-sky and urban conditions.

5.3.1 Nominal Case

Firstly, IPRd performance analysis will be based on a static scenario in open-sky conditions. Measurements captured during the data collection campaign will be exploited for this study. Collaborative scenario A, presented in 3.2.2, recreates an open-sky static scenario between two users. Both users were simulated by two static vehicles. On the roof of those vehicles were placed smartphone devices, as depicted by picture 3.4. This scenario lasted for 10 minutes. The two vehicles were spaced by 17.65m and the reference position of both agents was recorded by an high-end NovAtel SPAN INS GNSS receiver. $\mathbf{IPR}_{3D}$ vectors were computed and analyzed between multiple pairs of smartphones in order to estimate the baseline length IPRd.
Preliminary results show that on average, 19 pairs of common received signals were seen by our tested Android devices in static open-sky scenario with a minimum of 15 pairs. Modern Android smartphones have multi-frequency and multi-constellation capabilities, which facilitate the creation of pairs of common received signals and thus aids at the computation of code pseudorange DD. The accessibility to GNSS measurements on Android device eases computational processes of IPR ranging. Geometric Dilution of Precision (GDOP) value was equal to 1.3 with an average of 15 satellites in view for all tested devices.

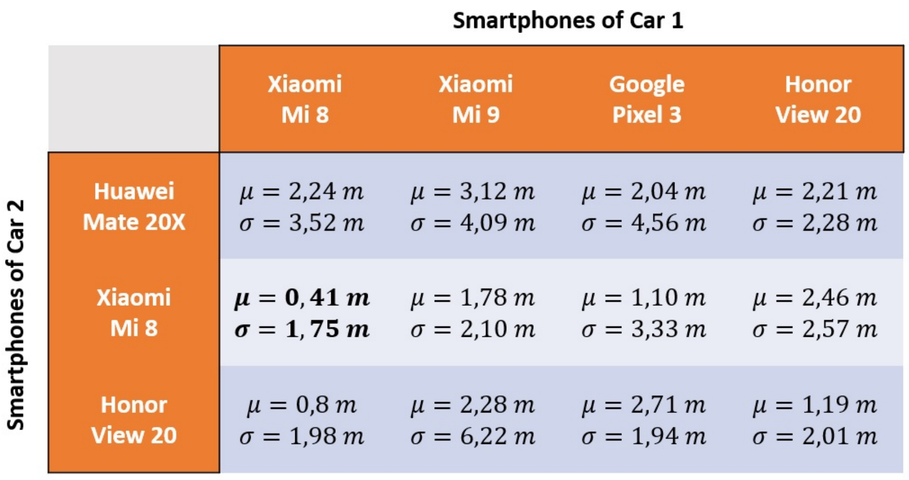
Figure 5.3 exhibits the ranging performance of our proposed IPRd method compared to other ranging techniques available to smartphone users. The displayed results have been computed between users 1 and 2, exploiting two Xiaomi Mi 8. Three ranging methods are tested and exposed in this figure: PVT, FLP and IPRd ranging methods. On the figure, norms of the computed range vectors (i.e: Baseline length) are plotted. The blue line represents the reference baseline length computed with the high-end reference receiver. Afterward, the red line shows the ranging solution obtained via Google's FLP positions solution. The green line stands for the estimated inter-user range computed with positions estimated via a GNSS-only PVT algorithm. And finally, the thick orange line shows the estimation results from the IPRd ranging method.

	Static & Open-Sky [600 s]		
Ranging techniques	mean[m]	std[m]	95%[m] (2σ)
IPRd	0.41	1.75	4.07
FLP	3.21	1.82	5.33
PVT	0.84	5.40	7.01

Table 5.1: Ranges Estimation Performance

Initial analysis depicts the excellent behavior displayed by our double-differenced ranging technique (IPRd) compared to the estimated ranges computed with GNSS PVT solutions. IPRd solution appears to be impacted by a residual noise component that does not affect the readability of the overall solution. On the other hand, the IMU-based FLP ranging solutions are less impacted by residual noise but shows a constant bias over time. This bias can be explain by the FLP positioning process that seems to rely heavily on tight coupling integration between GNSS and IMU data, hidden in Android "black-box" algorithms.
Statistical characteristics of tested ranging solutions are exposed in table 5.1. The statistical error characterization confirms the observations made on figure 5.3. We observe a standard deviation of 1.75 m and a mean error of 0.41 m for the IPRd estimated ranges. In addition, the bias and noise characteristics of FLP and PVT based ranging solutions are shown in table 5.1. IPRd statistical error analysis demonstrated that the ranging error tends to follow a Gaussian distribution. Those results comfort our idea that Inter-Phone Ranging (IPR) estimation technique is suitable and reliable for smartphone-based collaborative positioning. In [23], authors tested various ranging techniques in similar conditions also using Xiaomi Mi 8 smartphones. Both methods exposed validated this analysis. Relying on their findings, our proposed IPRd ranging method appears to be the most efficient technique for estimating inter-agent distances within a collaborative network of modern Android smartphones.

Furthermore, a complementary study has been carried out for comparing error characteristics of IPRd estimation between multiple smartphones brands and models. The goal was to study the theoretical correlation between smartphones brands and models. This study takes advantage of the variability of tested Android devices and echoes to the demonstration made in 3.3.3.a. Table 5.4 regroups the error characteristics obtained by estimating IPRd ranges with all presented smartphones. We consider a cooperative network of 7 smartphones with two distinct and equidistant groups of devices (i.e: 4 phones were placed on car 1 and 3 phones were stored in car 2). However, as shown on the table, no clear correlation can be established between either similar smartphones brand (Xiaomi Mi 8 and Xiaomi Mi 9), or between chipset brands (Honor View 20 and Huawei Mate 20X). Nevertheless, identical devices (Both Xiaomi Mi 8 and both Honor View 20) obtained the most accurate results in this controlled environment and scenario.



		Smartphones of Car 1			
		Xiaomi Mi 8	Xiaomi Mi 9	Google Pixel 3	Honor View 20
Smartphones of Car 2	Huawei Mate 20X	$\mu = 2{,}24\,m$ $\sigma = 3{,}52\,m$	$\mu = 3{,}12\,m$ $\sigma = 4{,}09\,m$	$\mu = 2{,}04\,m$ $\sigma = 4{,}56\,m$	$\mu = 2{,}21\,m$ $\sigma = 2{,}28\,m$
	Xiaomi Mi 8	$\boldsymbol{\mu = 0,41\,m}$ $\boldsymbol{\sigma = 1,75\,m}$	$\mu = 1{,}78\,m$ $\sigma = 2{,}10\,m$	$\mu = 1{,}10\,m$ $\sigma = 3{,}33\,m$	$\mu = 2{,}46\,m$ $\sigma = 2{,}57\,m$
	Honor View 20	$\mu = 0{,}8\,m$ $\sigma = 1{,}98\,m$	$\mu = 2{,}28\,m$ $\sigma = 6{,}22\,m$	$\mu = 2{,}71\,m$ $\sigma = 1{,}94\,m$	$\mu = 1{,}19\,m$ $\sigma = 2{,}01\,m$

Figure 5.4: IPRd Statistics with respect to Smartphone Brand & Model for Nominal Scenario

5.3.2 Urban Environment

Similar analysis have been made for urban environment. During the data collection campaign, collaborative scenario C was created to represent a low dynamic framework in urban canyon. Collaborative scenario C, detailed in 3.2.2, depicts an aiding user in static open sky condition assisting an aided second agent navigating in deep urban environment. Figure 5.5 pictures the ranging performance analysis between all tested ranging methods for urban environment. The tested ranging techniques are identical to the ones used in the previous nominal analysis for comparison purposes. Plus, these methods are plotted on the graph also using identical color codes. Left picture, noted a) on figure 5.5 shows the dynamic evolution of the estimated ranging solutions compared to a reference

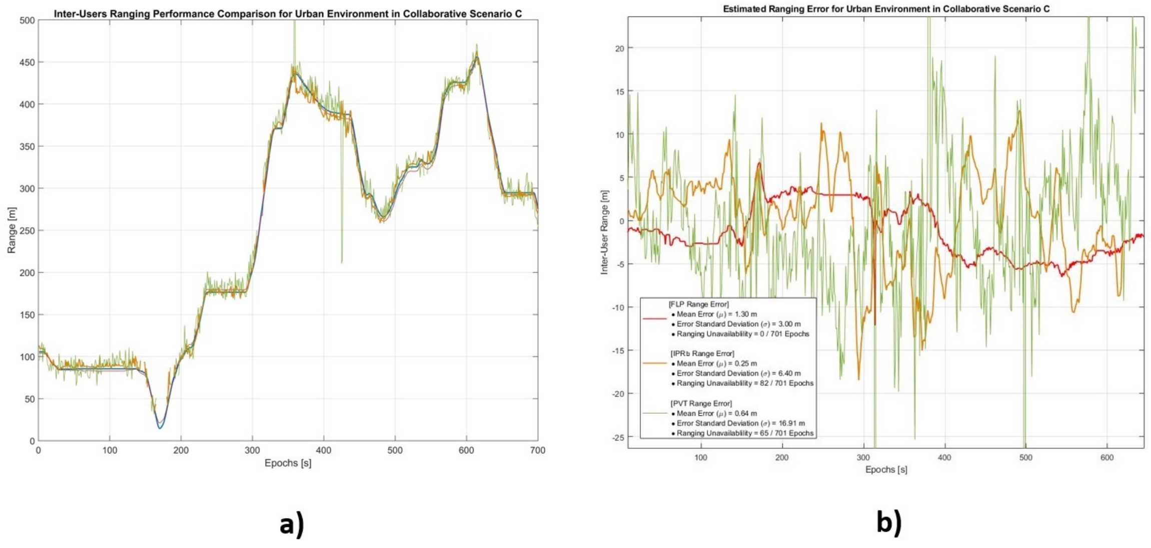


Figure 5.5: Inter-Phone Ranging Methods Performance for Urban Environment Scenario. a) Method Comparison against the Reference Trajectory. b) Ranging Error Comparison for all Tested Methods

trajectory. The right picture b), directly exposes the estimated ranging error for the three tested methods. Firstly, IPRd ranging solutions appear to be consistent with previously exposed results. However, as highlighted by the error characteristic statistics in Table 5.2, the error standard deviation significantly increased for this scenario. The performance of IPRd can be explained by the level of user dynamic experienced in this scenario that does not favor the user of DGNSS algorithm for baseline computation. Secondly, FLP based ranging demonstrates the most accurate performance. Nevertheless, FLP ranging bias appear during changes of direction in user's navigation. As an example, between epochs 510 and 525, the outputted FLP-based range bias increases in time with the succession of direction changes. This behavior can also be accredited directly to the computational processes of Android FLP positions.
An other element should be taken into consideration for choosing the most optimal ranging solution which is the solution availability. Indeed, during this urban analysis, IPRd range solutions were not computed in 36 occurrences out of 701 epochs. This can be explained by a reduced number of common signals pairs between the two agents in critical constrained environment. On average, 10 received signals were common to both users (thus generating 9 differential pseudorange pairs). During 36 occurrences, less than 4 common satellites were seen by both users, thus prohibiting the estimation of $\mathbf{IPR}_{3D}$ vectors. Overall, the GDOP value was approximately equal to 2.1 during the entirety of the scenario.

	Urban Environment [600 s]		
Ranging techniques	mean[m]	std[m]	95%[m] (2σ)
IPRd	0.25	6.40	8.83
FLP	1.30	3.00	3.35
PVT	0.64	16.91	21.97

Table 5.2: Ranges Estimation Performance (cont'd)

In urban conditions, FLP-based ranging method is the most efficient technique to be used for estimating inter-agent distance vector. However, Google's "black-box" process does not allow to control the outputted positioning solutions. Thus, this method cannot be kept and might become entirely dependent on Google's future decisions on the handling of self described ultimate positioning solution estimates. Therefore, we recommend to implement IPR differential ranging techniques for estimating inter-phone distances. In the case where FLP positions are available to the user then our emerging ranging technique can be used as a complementary method in order to increase the robustness of IPR ranging solutions.

5.4 Chapter Conclusions

This chapter presented GNSS-based collaborative ranging methods applied in the smartphone domain. Baseline estimation framework has been established and defined early on. DGNSS ranging method is favored for accurately depicting inter-user ranging and the technique is mathematically described. Moreover, our proposed smartphone-based IPR innovative method was introduced. This technique reviewed the particularities and advantages obtained by using Android GNSS raw data measurements for computing inter-user ranges. Then, the performance of IPRd ranging estimation was established through a nominal and urban study. The controlled conditions of our experiment setup, and the repetition of previously shown results with other Android devices, demonstrate the reliability of our analysis in nominal conditions. This analysis revealed that IPR ranging technique is one of the most optimal algorithms to be implemented for smartphone-based collaborative network. Statistical error analysis allow to generalize our results and to exploit IPRd solution for future simulation studies.
Inter-phone ranging is considered the first stepping stone towards smartphone-based collaborative networking. In the next chapter, $\mathbf{IPR}_{3D}$ estimated vectors play a crucial role in our proposed smartphone collaborative engine, *SmartCoop*. Inter-phone ranges will be used to constrain a set of agent positions within a cooperative network and thus setting a constrained optimization problem. Outputted smartphone collaborative positioning solutions will be proven to significantly increase agent's position accuracy and reliability.

6 Smartphone Collaborative Positioning

Contents

This chapter aims at discussing the selection of a collaborative positioning filter adapted to smartphone positioning for hybridizing GNSS raw measurements and $\mathbf{IPR}_{3D}$ estimated ranges. An estimation filter is defined as an algorithm that takes time dependent measurements and statistical distributions for producing estimates of unknown variables. In addition, this chapter is a complete synthesis of previously exposed research parts that combines smartphone positioning (part I) and collaborative positioning techniques (part II). This final chapter is divided as follows. First, Bayesian and non-Bayesian hybridization filters will be presented, exposing their strengths and constraints for implementing smartphone CP techniques. Then, our proposed smartphone-based collaborative algorithm, named *SmartCoop*, will be introduced and its implementation method will be fully described. Afterward, results analysis from simulations in open-sky and urban environments will be displayed, highlighting positioning performance enhancement derived from the implementation of our CP method. Finally, our method will be evaluated against other collaborative techniques. A discussion will be opened regarding the impact of a system *Anchor* within the developed CP method and likewise, the effect of cooperative network user's geometry on newly computed collaborative mobile positions.

6.1 Collaborative Hybridization Filters

In the early years of the twenty first century, multiple algorithms and estimation techniques have been used for solving positioning and navigation solutions of GNSS receivers. Historically, Least Square (LS) estimation techniques have been favored for estimating positioning solutions. Later on, advanced techniques have been developed for integrating additional available data with GNSS measurements [101]. Recursive Bayesian algorithms, whose canonical implementation for a Gaussian state space system model is the KF, became the best known method for estimating GNSS receiver positions. The approach proposed by this chapter aims at selecting a collaborative navigation filter adapted to smartphones' measurements by exposing existing CP methods.

In order to identify a suitable candidate for a collaborative filter, requirements for smartphone-based collaborative system are laid out. A collaborative estimation engine is defined as an algorithm capable of fusing multiple measurements and parameters from different sources. In our case, GNSS measurements will be complemented by previously computed cooperative ranges, IPR. Due to complex environment and compact smartphone's hardware constraints, cooperation filter must be pursued at range level. However, smartphone computing capabilities might limit the implementation of such complex filters. This is the reason why, a centralized cooperative network approach as been chosen (see section 4.6.2) allowing the transfer of computational loads to a hub server empowering the implementation of complex algorithms. In [102], centralized and distributed channel cooperative techniques have been studied through direct and indirect multi-agent systems. Cooperative positioning engines have been studied in different research fields and constituted a first approach to our problem.

6.1.1 State-of-the-Art Review

The integration of additional measurements and information with GNSS data has been developed at the end of the twentieth century, mainly targeting precise positioning applications and solution continuity. Historically, early contributions explored the integration of GNSS techniques within simple estimation processes [103]. Hybridization opportunities rose with the democratization of small IMU units that were hybridized with GNSS measurements. Multiple estimation techniques have been employed to fuse various types of measurements. The selection of these estimation tools represents the preliminary phase for establishing a collaborative network.
In the GNSS literature, estimation techniques are typically defined as state estimators which are based on linear steady state system model, applied to linearized measurement models. Methods differ based on complexity, conditions and recurrence. These methods can be categorized into two groups: *Bayesian* and *Non-Bayesian* estimation techniques. The main difference between these two classes lies in the assumptions made on the estimated state that inputs the two techniques. Comparisons between estimation techniques have been widely studied when applied to the majority of distribution models [104]. In GNSS, Bayesian processes have been thoroughly tested and analyzed for obtaining an

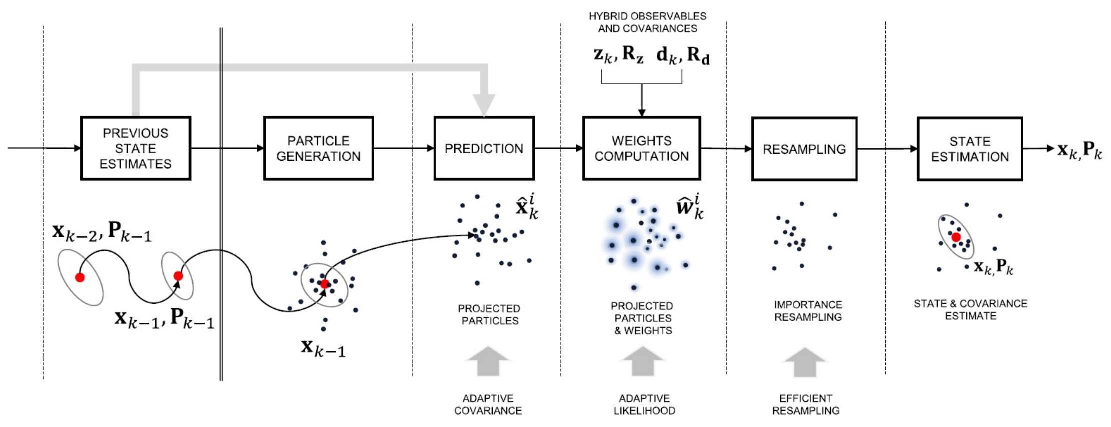


Figure 6.1: Block Diagram of a PF used for a Tight Integration Scheme in a Cooperative Engine. Picture extracted from [14].

efficient estimation of receiver position [105]. Popular Bayesian filters range from Kalman filters to Particle Filter (PF). Collaborative capabilities of such estimation techniques have been demonstrated in [94]. Intuitively, the non-Bayesian counterparts to KF and PF estimators are the Least Squares (LS) and Maximum Likelihood (ML) estimators, respectively. Estimation tools are presented below according to their respective class.

6.1.1.a Bayesian Approaches

Bayesian approaches account for prior information considering a state-space model. This representation stands on a twofold model; The first covers the evolution of states in time, whereas the measurements equation accounts for the dependency of given measurements on unknown states. Multiple techniques utilize this method, we are focusing here on the KF and the PF tools.

Kalman Filter (KF)

Kalman filters (KF, Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF)) are widely used estimation algorithms in the GNSS positioning domain. the former algorithm provides an optimal Bayesian technique for linear state and measurements equations. On the other hand, EKF and UKF provide a good approximation when the given measurements are non-linear. The particularity of these techniques resides in the recursiveness of their processes, allowing the estimation of linear or linearizable dynamic models. In many cases (including GNSS positioning), the assumption needed for optimal Bayesian estimation cannot be applied. Sub-optimal algorithms deal with measurements that are neither considered white noise nor Gaussian distributed. The most used variation of Kalman technique, the EKF, has been thoroughly reviewed in [105]. In [94], Kalman filter estimation process has been tested for integrating DGNSS inter-agent measurements with GNSS data in a collaborative manner. Promising results were exposed for integrating various sensors while decreasing the overall estimation errors.

Particle Filter (PF)

The PF is based on a Monte Carlo approximation of optimal sequential Bayesian state called particle processing. PF characterizes a posteriori distribution of the system. Figure

6.1 describes the processing stages of PF for tight integration of inter-agent distances. This method has been extensively studied in [14] and [106]. The main advantage of this solution is the ability to process non-linear measurements that follow unknown distributions. On the other hand, this technique is known to significantly increase the computational power and algorithm complexity compared to other estimation methods. In [106], the two previously exposed Bayesian methods have been applied to GNSS collaborative scenarios. The presented results show that despite the assumption that PF method would overcome in performance Kalman technique, in dynamic simulated scenario, the increased complexity brought by PF does not provide extensive advantages compared to the implementation of an EKF method.

6.1.1.b Non-Bayesian Techniques

Non-Bayesian filters are often referred to as classical estimation techniques referring to the earliest contributions of such method for position estimation purposes [24]. As opposed to Kalman filter, Maximum Likelyhood (ML) and LS principles are the most common non-Bayesian estimation methods in the positioning and navigation domain. ML filters target approaches where measurements models are known and where a likelihood function can be obtained. Most common to the GNSS domain, the LS technique is used to estimate variable in a linear regression model that minimizes the sum of squared measurement residuals. A LS estimation solution of a parameter denoted $\hat{x}$ is shown in equation 6.1.

$$\hat{x} = (\mathbf{H}^T\ \mathbf{H})^{-1}\ \mathbf{H}^T\ \mathbf{y} \tag{6.1}$$

with $\mathbf{y}$ being a set of noisy observations linearly related to the vector parameter $\mathbf{x}$.
$\mathbf{H}$ represents a design matrix constituted by a combination of linear coefficients with respect to each parameter components. In GNSS the computation estimation of PNT solutions are often performed using LS algorithms. This estimation technique is limited by the epoch by epoch estimation characteristic while not supporting apriori information. A WLS technique is exposed and detailed in appendix chapter B.

6.1.2 Selected Collaborative Engine

The previous section revealed the major estimation tools available for creating an adequate collaborative engine that would fit the requirements for implementing a smartphone-based cooperative network (such as: the network size, measurements versatility and integration). The investigation previously made on existing GNSS data fusing and cooperative positioning solutions highlighted various methods strengths and weaknesses. In the automotive and robotic industry, complex algorithms have been put in place for hybridizing rover GNSS position with additional sensors data (including additional hardware equipments such as LIDAR and SONAR). However, those implemented solutions do not correspond to smartphone-based collaborative network characteristics and measurements constraints. A compromise between performance and agility needs to be found to best fit smartphone-based collaborative engine.

Therefore, we argument that the most relevant estimation technique to be used for smartphone collaborative network is based on a non-Bayesian LS principle. Classical estimation processes allow for developing, analyzing and expanding flexible algorithmic mechanisms.

The following list records the advantages that supported our decision for implementing LS estimation in place of other estimation tools.

- Reduces the computational complexity of the collaborative engine, decreasing computing loads on the centralized processing unit.
- Better suited for hybridizing multiple smartphone measurements gathered from various brands and models. Fewer hypotheses are made on the evolution of the state vector allowing for the implementation of an agile technique.
- A priori knowledge on measurements statistical distribution, required for Bayesian techniques, cannot be guaranteed for smartphone-based measurements due to the differences found in hardware components on past and future devices.
- Optimized scalability capabilities granting an easy implementation and analysis of the smartphone cooperative network size.
- A well-established estimation tool shift focus to prioritize collaborative performance. The estimated collaborative results will be less dependent on measurements modeling and parameter initialization (e.g: filter tuning).
- Provides a fair comparison with LS driven PVT solutions for GNSS standalone positioning results.

Various studies, such as in [107] and [22], demonstrated the implementation of CP methods based on LS estimation tool with promising results.

6.2 *SmartCoop* Algorithm

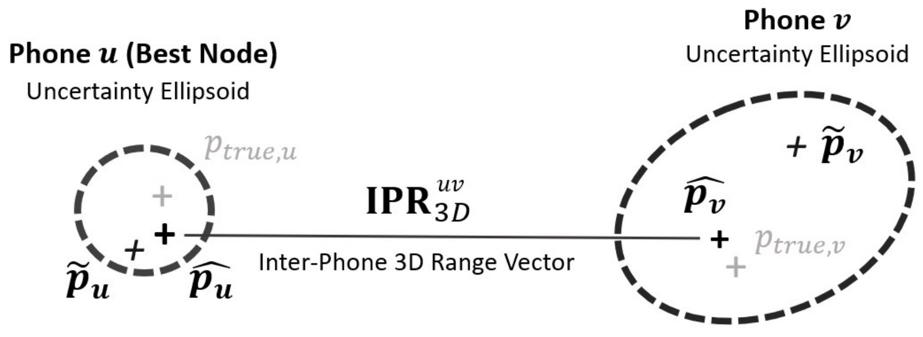
This section will present our proposed smartphone collaborative algorithm. This **SMART**phone **COOP**erative positioning technique has been baptized *SmartCoop*. Our innovative cooperative engine aims at improving positioning performances of smartphones within a defined network of users. Collaborative positions from network's members will be computed based on a first raw position estimate (computed or extracted from Android GNSS data measurements) while being restrained by previously introduced inter-phones vectors ($\mathbf{IPR}_{3D}$) between all smartphones users. The following methodology will characterize the constitution of a *constrained optimization problem* to be solved for estimating collaborative positions. This method has been inspired by works developed in the automotive industry released in [22] and [87].
This section is providing a complete overview of our proposed collaborative engine. First and foremost, the mathematical problem will be defined and initialized, explained in a step by step procedure. Then, the following subsection will detail the assumptions made by our cooperative engine. A focus will be given to the role played by $\mathbf{IPR}_{3D}$ as constraining parameters in this optimization problem. Afterwards, the algorithm implementation on the Matlab environment is presented. Finally, the innovative characteristics of the developed *SmartCoop* algorithm are exhibited.
Figure 6.2 illustrates the principle of the *SmartCoop* algorithm. The general concept and philosophy of the presented collaborative method is displayed by a simple two users network.

6.2.1 Mathematical Definition

The main goal of this subsection is to characterize our nonlinear constrained optimization problem built for estimating new cooperative positions. The objective is to minimize the norm of the 3D position errors between the newly estimated users cooperative positions $\hat{\mathbf{p}}$ and their reciprocal true positions $\mathbf{p}_{true}$. This minimization process is leveraged by Inter-Phones vectors ($\mathbf{IPR}_{3D}$) computed between each individual forming the collaborative network. The mathematical definition of the problem is detailed below, after setting up parameters notation and initializing the problem.

6.2.1.a Problem Initialization & Notation

A set of connected smartphone devices true positions is denoted
$\mathbf{P} = [p_1, p_2, \cdots, p_u, \cdots, p_n]$ with n representing the total number of users and u represents the user index within the network. This position vector regroups all smartphones positions in time. The absolute true location of Android mobiles are represented by a 3D coordinate vector $\mathbf{p}_{true,u}$. We consider that each device connected to the cooperative network is able to communicate with the central processing unit. Android GNSS raw data measurements, as discussed in section 4.6.2, are exchanged by mobile users. Thus, a preliminary estimated position is computed or retrieved for each user using their transmitted GNSS-only measurements. The following estimation process is characterized as snapshot, meaning that each estimated collaborative positioning solution are epoch independent. This first estimate will be referred to as an original mobile GNSS fix and is noted $\tilde{\mathbf{p}}_u$. The positioning error vector between the original fix and the true position is set as $\boldsymbol{\delta}_u = \tilde{\mathbf{p}}_u - \mathbf{p}_{true,u}$. The goal of this cooperative engine is to estimate collaborative position defined as $\hat{\mathbf{p}}_u$.


Phone $\boldsymbol{u}$ (Best Node)
Uncertainty Ellipsoid
Phone $\boldsymbol{v}$
Uncertainty Ellipsoid
$p_{true,u}$
$\mathbf{IPR}_{3D}^{uv}$
Inter-Phone 3D Range Vector
$\tilde{\boldsymbol{p}}_{\boldsymbol{u}}$
$\widehat{\boldsymbol{p}_{\boldsymbol{u}}}$
$\tilde{\boldsymbol{p}}_{\boldsymbol{v}}$
$\widehat{\boldsymbol{p}_{\boldsymbol{v}}}$
$p_{true,v}$

Satisfying the constraining 3D equations given by:

$$\begin{cases} (\hat{p}_{v,x} - \hat{p}_{u,x}) = \mathbf{IPR}_{3D,x}^{uv} \\ (\hat{p}_{v,y} - \hat{p}_{u,y}) = \mathbf{IPR}_{3D,y}^{uv} \\ (\hat{p}_{v,z} - \hat{p}_{u,z}) = \mathbf{IPR}_{3D,z}^{uv} \end{cases}$$

Figure 6.2: Illustration of *SmartCoop* Constrained Non-Linear Optimization Problem between two Smartphones u and v.

3D inter-distance range vector are also a pre-requisite to the presented *SmartCoop* algorithm. $\mathbf{IPR}_{3D}$ range estimates are computed on the remote server for every user couple present in the network. This GNSS-based collaborative ranging vectorial solution is denoted as $\mathbf{IPR}_{3D}^{uv}$ between mobile users u and v.

6.2.1.b IPR-based Constraints Notation

The constraining equations play a key role in this optimization problem. The following mathematical expression in equation 6.2 will show the notation that will be used for describing the constraint equations

$$\begin{cases} (\,\hat{p}_{v,x} - \hat{p}_{u,x}\,) &= \mathrm{IPR}_{3D,x}^{uv} \\ (\,\hat{p}_{v,y} - \hat{p}_{u,y}\,) &= \mathrm{IPR}_{3D,y}^{uv} \\ (\,\hat{p}_{v,z} - \hat{p}_{u,z}\,) &= \mathrm{IPR}_{3D,z}^{uv} \end{cases} \tag{6.2}$$

The right side of the equation represents the $\mathbf{IPR}_{3D}$ vector components and are known during the optimization process. The IPR computation method is exposed in section 5.2. Those pre-established estimates are compared to the collaborative distance computed by taking difference of tentative collaborative position between two phone candidates on x, y and z coordinates. Those tentative collaborative norms are represented on the left side of equation 6.2. The objective of these constraining equations is to match the tentative position differences with the established IPR results.

6.2.1.c Collaborative Algorithm

Our SmartCoop algorithm is articulated around an optimization problem. The objective here is to minimize the sum of 3D position discrepancies between the newly estimated collaborative positions ($\hat{\mathbf{P}}$) and their original positioning solutions ($\tilde{\mathbf{P}}$) as defined in section 6.2.1.a. The minimization process will be constrained by nonlinear equations satisfying the equality between our estimated $\mathbf{IPR}_{3D}$ and the corresponding baseline vector of two tentative positions, as expressed in equation 6.2. This collaborative process is described by the following mathematical model.

Algorithm Hypothesis
The mathematical derivation is supported by the following two main hypotheses. The impact of those hypotheses is further discussed in section 6.4.3.

- Independence between smartphones positioning error
  In the following mathematical derivation, we will assume that the positioning errors between different smartphone users within the collaborative network are independent.

- Independence of GNSS positioning error on x, y and z
  For the sake of simplification, we assume that the GNSS positioning errors on x, y and z are independent in the ECEF reference frame. Even though we know that this hypothesis does not hold, as demonstrated in this paper [24], it has been established this way in the following mathematical derivation for creating a preliminary proof of concept for our collaborative engine.

We introduce the probability density function of the discrepancy between the original positioning solution ($\tilde{\mathbf{p}}_u$) and the true position solution for a specific user indexed by u. The position solution difference is denoted as $\boldsymbol{\delta}_u = \tilde{\mathbf{p}}_u - \mathbf{p}_{true,u}$. Based on our hypothesis, this quantity is assumed to follow a multivariate Gaussian distribution $\boldsymbol{\delta}_u \sim \mathcal{N}(\boldsymbol{\mu}_u, \Sigma_u)$ characterized by $\boldsymbol{\mu}_u$ and Σ_u, respectively describing the mean and variance of this distribution expressed in equation 6.3

$$\boldsymbol{\mu}_u = \begin{bmatrix} \mu_{u,x} \\ \mu_{u,y} \\ \mu_{u,z} \end{bmatrix} \quad \text{and} \quad \underset{3\times3}{\Sigma_u} = \begin{bmatrix} \sigma^2_{u,x} & 0 & 0 \\ 0 & \sigma^2_{u,y} & 0 \\ 0 & 0 & \sigma^2_{u,z} \end{bmatrix} \tag{6.3}$$

In this case, the covariance matrix Σ_u is diagonal due to the assumptions stated previously about the independence of GNSS positioning error on x, y and z.
The probability density function for a multivariate normal distribution is expressed as in equation 6.4. In this equation, the determinant of the covariance matrix Σ_u is denoted $Det(\Sigma_u)$.

$$\varphi(\boldsymbol{\delta}_u) = \frac{1}{\sqrt{2\pi[Det(\Sigma_u)]}} \,.\, e^{-\frac{(\boldsymbol{\delta}_u-\boldsymbol{\mu}_u)^T \Sigma_u^{-1} (\boldsymbol{\delta}_u-\boldsymbol{\mu}_u)}{2}} \tag{6.4}$$

Taking into account our previously stated assumptions, the above equation can be simplified as shown in equation 6.5, to express the probability density function of each individual axis on x, y and z. Equation 6.5 exposes the probability density function on the x axis.

$$\varphi(\delta_{u,x}) = \frac{1}{\sqrt{2\pi}\sigma_{u,x}} \,.\, e^{-\frac{(\delta_{u,x}-\mu_{u,x})^2}{2\sigma^2_{u,x}}} \tag{6.5}$$

The minimization of the sum of the squared 3D position solution differences is equivalent to maximizing the probability density function of the established $\boldsymbol{\delta}_u$. Equation 6.6 provides an overview of this optimization problem with vectors $\hat{\mathbf{P}}$ and $\mathbf{P}$ as defined in section 6.2.1.a.

$$\hat{\mathbf{P}} = \underset{\mathbf{P}}{\arg\max} \prod_{u=1}^{n} \varphi(\hat{\mathbf{p}}_u - \tilde{\mathbf{p}}_u) \tag{6.6}$$

By taking into consideration the three independent axes, equation 6.6 can be further rewritten as in equation 6.7.

$$\hat{\mathbf{P}} = \underset{\mathbf{P}}{\arg\max} \prod_{u=1}^{n} \varphi(\hat{p}_{u,x} - \tilde{p}_{u,x}) \,.\, \varphi(\hat{p}_{u,y} - \tilde{p}_{u,y}) \,.\, \varphi(\hat{p}_{u,z} - \tilde{p}_{u,z}) \tag{6.7}$$

The product of probability density functions, described in equation 6.7 can be isolated and will be denominated U. Equation 6.8 describes the newly appointed quantity U. Moreover, this equation is expanded by applying the definition of probability density function given previously by equation 6.5.

$$\begin{aligned} W &= \varphi(\hat{p}_{u,x} - \tilde{p}_{u,x}) \;.\; \varphi(\hat{p}_{u,y} - \tilde{p}_{u,y}) \;.\; \varphi(\hat{p}_{u,z} - \tilde{p}_{u,z}) \\ &= \frac{1}{\sqrt{2\pi}\sigma_{u,x}} \,.\, e^{-\frac{(\hat{p}_{u,x}-\tilde{p}_{u,x}-\mu_{u,x})^2}{2\sigma_{u,x}^2}} . \frac{1}{\sqrt{2\pi}\sigma_{u,y}} \,.\, e^{-\frac{(\hat{p}_{u,y}-\tilde{p}_{u,y}-\mu_{u,y})^2}{2\sigma_{u,y}^2}} . \frac{1}{\sqrt{2\pi}\sigma_{u,z}} \,.\, e^{-\frac{(\hat{p}_{u,z}-\tilde{p}_{u,z}-\mu_{u,z})^2}{2\sigma_{u,z}^2}} \end{aligned} \tag{6.8}$$

The previous equation can then be simplified by executing the logarithmic expansion of the term W giving $log(W) = -U + V$. The logarithmic function W can be divided into two distinctive terms, denoted by the arbitrary terms U and V. These two terms are described by equations 6.9 and 6.10.

$$U = \frac{(\hat{p}_{u,x} - \tilde{p}_{u,x} - \mu_{u,x})^2}{2\sigma_{u,x}^2} + \frac{(\hat{p}_{u,y} - \tilde{p}_{u,y} - \mu_{u,y})^2}{2\sigma_{u,y}^2} + \frac{(\hat{p}_{u,z} - \tilde{p}_{u,z} - \mu_{u,z})^2}{2\sigma_{u,z}^2} \tag{6.9}$$

$$V = \log \frac{1}{\sqrt{2\pi\sigma_{u,x}}} + \log \frac{1}{\sqrt{2\pi\sigma_{u,y}}} + \log \frac{1}{\sqrt{2\pi\sigma_{u,z}}} \tag{6.10}$$

According to the mathematical properties of the logarithmic function, we can characterize each individual term. It appears that the term V is a constant with regards to our minimization problem. Therefore, in order to maximize the product of W (showed in equation 6.7), the quantity defined by V needs to be minimized. Finally, equation 6.7 can be approximated as shown in equation 6.11 and represents our collaborative engine. This final minimization process is completed by the constraining equations constructed as the difference of tentative collaborative positions compared against estimated $\mathbf{IPR}_{3D}$ vector orthogonal components.

$$\begin{gathered} \hat{\mathbf{P}} = \arg \min_{\mathbf{P}} \sum_{u=1}^{n} \frac{(\hat{p}_{u,x} - \tilde{p}_{u,x} - \mu_{u,x})^2}{2\sigma_{u,x}^2} + \frac{(\hat{p}_{u,y} - \tilde{p}_{u,y} - \mu_{u,y})^2}{2\sigma_{u,y}^2} + \frac{(\hat{p}_{u,z} - \tilde{p}_{u,z} - \mu_{u,z})^2}{2\sigma_{u,z}^2} \\ \text{while satisfying:} \begin{cases} (\,\hat{p}_{v,x} - \hat{p}_{u,x}\,) &= \mathrm{IPR}^{uv}_{3D,x} \\ (\,\hat{p}_{v,y} - \hat{p}_{u,y}\,) &= \mathrm{IPR}^{uv}_{3D,y} \\ (\,\hat{p}_{v,z} - \hat{p}_{u,z}\,) &= \mathrm{IPR}^{uv}_{3D,z} \end{cases} \end{gathered} \tag{6.11}$$

The main processes used in our collaborative smartphone-based engine *SmartCoop* are summarized as follows:

- Network's smartphone users send, via the collaborative network, data packet containing at minima Android GNSS raw data measurements.
- Central processing unit, computational head of the network, receive and analyze data packet for all network's mobiles.
- $\mathbf{IPR}_{3D}$ vectors are computed for each network's receiver couple.

- Network's user collaborative data is gathered into an objective function for defining our optimization problem.
- Estimated collaborative positions are being computed iteratively throughout a series of tentative guesses aiming at minimizing the objective function **and** to satisfy an $\mathbf{IPR}_{3D}$ set of constraints.
- *SmartCoop* outputs newly collaborative solutions for each network's member.

6.2.2 Constraints & Hypotheses

A successful outcome of collaborative estimation via our *SmartCoop* algorithm is bounded by three main hypotheses. Those characteristics are detailed below. Following that, the constraints system is explained in details, showing the impact of $\mathbf{IPR}_{3D}$ on the estimation process.

6.2.2.a Network Size

The size of the cooperative network limits the ability to communicate and/or exchange effective measurements. In this study case, we are focusing on low-dynamic urban scenario, corresponding to a nominal smartphone's user navigation behavior through urban canyons. A relevant communication link needs to be provided for connecting network's users. The total number of users on the cooperative network is anticipated to be large due to the high density of smartphone devices in today's city centers. This problematic is answered with the development of a collaborative algorithm capable to be scaled to a large number of users due to the flexibility nature of our chosen estimation method (e.g: network size variation, difference in measurements quality, etc).

Secondly, network spatial size has also been determined specifically for our study case. The limiting factors for determining such size are directly related to the computation of $\mathbf{IPR}_{3D}$ vectors between users. Indeed, the IPR estimation uses double difference computational techniques for mitigating common error effects. In order to hold assumptions made in the IPR estimation method, the network size needs to be small enough for users to observe similar atmospheric and ionospheric biases. Our collaborative network is thus assumed to be the size of cities hyper centers where users can be spread by an order of magnitude of tens kilometers.

6.2.2.b Communication Link

We assume that a communication link exists between each individual constituting the collaborative network and the central processing unit. Our hypothesis presumes that this existing link is: Reliable, Secure and Efficient. Loss of data packets or bits during the data transmission is assumed to be negligible. This hypothesis is supported by the hyper connectivity of modern smartphones as stipulated in section 4.4.

6.2.2.c User's System Anchor

A system anchor will be put in place in our collaborative engine. This role will be taken by one of the network's mobile. A user that shows outstanding standalone positioning capabilities will be selected as the collaborative system Anchor. In this research, the designated smartphone will be identified as the *Best Node* of the network. Originally, this idea was inspired by differential GNSS techniques (such as RTK) that employ reference stations for computing differential corrections. In a CP technique, the role of the *Best Node* will be to correctly initialize the constraints structure of the collaborative system. Indeed, the *Best Node* smartphone shows great positioning solutions translating to a smaller positioning error. In the CP literature, multiple papers demonstrated the theoretical and practical use of defining system anchor in a collaborative system [108] [109]. In the real world, this hypothesis is supported by different scenarios. A variety of environments can be found in modern city centers. Also, it can be ambitioned that one of the smartphone user possess a newer device compare to other or is positioned in an open area. Those scenarios could justify the presence of a *Best Node* smartphone within a collaborative network.

6.2.2.d Constraining a Non-Linear Optimization Problem

The objective function that we are minimizing is bounded by nonlinear constraints, as seen in equation 6.11. A constraint is defined by the difference between two estimated cooperative positions compared in opposition to their estimated $\mathbf{IPR}_{3D}$. It is then derived on all the position and on all the $\mathbf{IPR}_{3D}$ components: x, y and z axes, in order to bind our system in the 3D positioning domain. For our collaborative engine, two independent users, u and v, are constrained by a set of equations, shown in 6.12.

$$\text{Set of Constraints} = \begin{cases} (\,\hat{p}_{v,x} - \hat{p}_{u,x}\,) & = \text{IPR}^{uv}_{3D,x} \\ (\,\hat{p}_{v,y} - \hat{p}_{u,y}\,) & = \text{IPR}^{uv}_{3D,y} \\ (\,\hat{p}_{v,z} - \hat{p}_{u,z}\,) & = \text{IPR}^{uv}_{3D,z} \end{cases} \tag{6.12}$$

The number of constraints sets needed for binding all smartphone users in the network varies. The number of sets needed can be easily computed by equation 6.13. As an example, if a network is made of 10 users ($n = 10$) then our collaborative system will be bounded by 45 sets of constraints representing a total of 135 equality constraint equations.

$$\text{Number of constraints' sets} = \frac{n(n-1)}{2} \tag{6.13}$$

Example and Constraints Representation

Let us draw an example for demonstrating the role of constraints within the *SmartCoop* engine. A network of four cooperative smartphone users is created, with user 2 being selected to be the *Best Node* of the network. Figure 6.3 depicts a 3D representation of constraints bidding an array of positions. This example involves a network of four smartphones, thus generating six sets of constraints totaling 18 equations restraining our minimization process. For analysis purposes, the true positions (represented by black crosses) and true ranges (black lines) are pictured on this figure. Obviously, none of those parameters were used in the collaborative engine and are defined here only for evaluation purposes. The red lines and light colored spheres represent initial GNSS fix positions, obtained via Android GNSS raw data measurements, and their resulting unsatisfied constraints. On the other hand, the red lines represent the newly estimated inter-user distances (evaluated by $\mathbf{IPR}_{3D}$ estimates, after all constraining equations are satisfied). Newly estimated collaborative positions are drawn in darker color and are

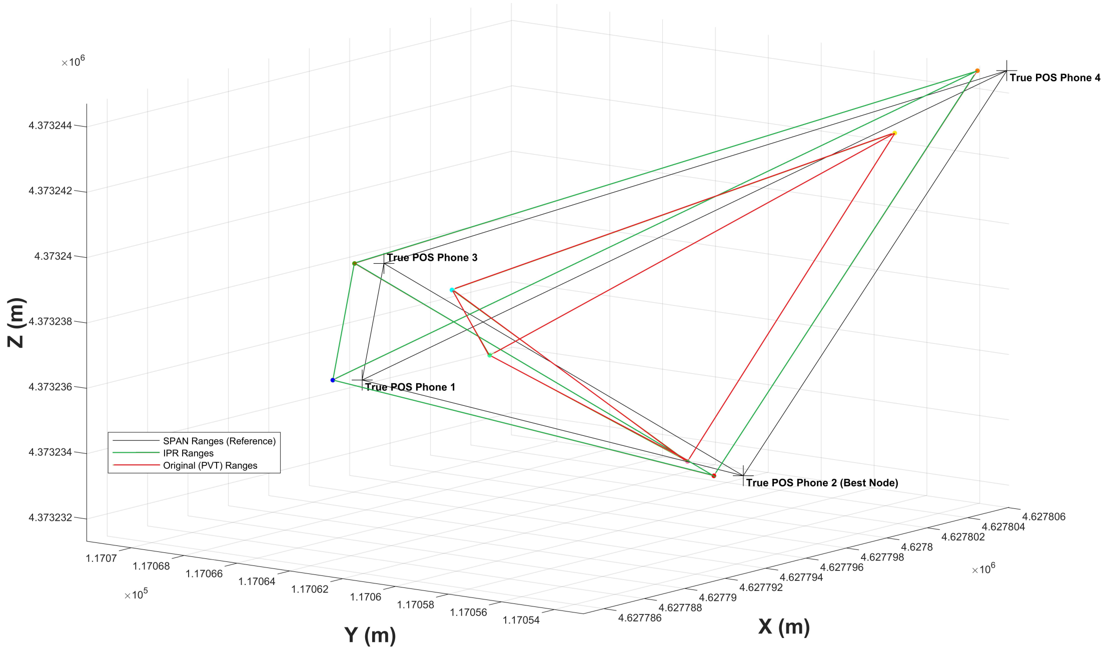


Figure 6.3: Inter-Phone Range Vector ($\mathbf{IPR}_{3D}$) Constraint Visualization in 3D - ECEF Coordinates in [m]

linked by the green lines. This figure demonstrates the importance of satisfying the previously defined constraints and shows how they properly impact the new estimation of collaborative positions. As shown on this figure, the *Best Node* user is used as a system Anchor for initializing the orientation for the polygon of constraints. Finally, this short (single epoch) example illustrates how the newly collaborative positions have been improved through the *SmartCoop* algorithm.

6.2.3 Algorithm Implementation

SmartCoop algorithm has been developed in a Matlab® environment. This programming environment offers multiple ways to solve various types of optimization problems. Two main approaches exist on Matlab for defining and solving optimization problems. The first one is a "solver-based" approach, the second is called "problem-based" approach. They both differentiate on how the optimization problem is set within Matlab environment and do not impact the estimation process. In general, an optimization problem is represented in Matlab by following the next steps.

- Choosing an approach method
- Choosing an optimization solver (or solver function)
- Setting up the optimization problem (initialization, creating the objective function and the corresponding constraints)
- Tune up the options of the solver method
- Launch the estimation process

MathWorks offers a complete database and a complete user manual for selecting the most optimized method for a specific process. For developing a Non-linear constrained optimization problem, the best solver function is *fmincon*. This function can be interpreted as a constrained minimization process outputting a vectorial value x, that is a local minima to an objective function $f(x)$ subjected to invariable constraints. Equation 6.14 gives the general definition for the *fmincon* function.

$$\min_x f(x) \quad \text{given: } \begin{cases} c(x) \leq 0 \\ ceq(x) = 0 \\ A.x \leq b \\ Aeq.x = beq \\ lb \leq x \leq ub \end{cases} \tag{6.14}$$

The expression on the right side of the bracket in equation 6.14 are constraining functions that can be set up for defining any optimization problem. b and beq are vectors. $c(x)$ and $ceq(x)$ are functions returning a vector solution. A and Aeq are in matrix form. It is to be noted that $ceq(x)$, $c(x)$ and $f(x)$ can be setup as non-linear functions. Moreover, this minimization process holds only if one or more (defined by the user) constraining function is satisfied, in equation 6.14.

The optimization problem is then initialized by setting the objective function (function to be minimized) and the associated constraints. In our case, the objective function is defined as in equation 6.11. Initial conditions (x_0) are set by the initial GNSS fixes computed or retrieved from Android GNSS raw data measurements for all network's mobiles. At last, the constraints are established in a matrix form arranged in nonlinear equality constraints that the solver function will attempt to satisfy. Thus, the set of constraints are written in $ceq(x)$ in function of the iterative tentative collaborative position estimates x.

Moreover, the algorithm associated with the solver function, *fmincon*, needs to be selected and adapted to the situation. Five algorithm options are available on Matlab. After careful review and multiple comparison tries, the Sequential Quadratic Programming (SQP) algorithm showed the highest efficiency performance for our *SmartCoop* method compared to the other four techniques. This technique estimates the Hessian of the Lagrangian function employing a quasi-Newton updating method for every iteration steps. This procedure will allow to generate a searching direction for a line search procedure in order to find the local minima of the objective function. Further details and testing methodology can be found on SQP algorithm in [110] and [111]. The implementation of the *fmincon* function is further discussed in section 6.4.3.

A complete and commented "live" Matlab script is available in appendix D. This code shows the exact steps taken in our *SmartCoop* algorithm for estimating smartphone collaborative positions.

6.3 Simulation and Real Data Analyses

After a thorough description of our innovative *SmartCoop* algorithm, the results of our simulations will be displayed, demonstrating the intrinsic performance of this collaborative positioning method. This result analysis has been divided into two parts. The first one exposes the results of a nominal case scenario, in static open-sky environment, for a given number of cooperative network's users. The simulation parametrization will be detailed before discussing *SmartCoop* positioning enhancement. In a second time, a static urban scenario will be presented. For this simulation, smartphone urban positions impairments have been generated using a powerful 3D model-based GNSS software simulator leveraging ray-tracing capabilities, named *SPRING*. This simulator has been developed by Centre National d'Études Spatiales (CNES) and its access has been granted for the purpose of this research in the Centre Spatial de Toulouse (CST) technological center, under the supervision of CNES engineers. The following results analysis shows the outstanding performance of our CP method for static Android mobiles in degraded conditions.

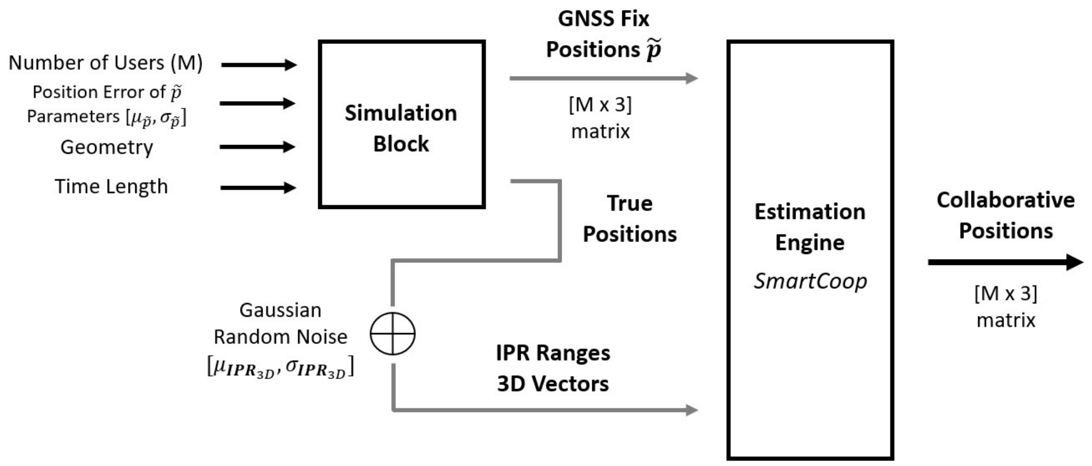
6.3.1 Nominal Scenario

First and foremost, the implemented CP method has to be tested in optimal conditions in order to assess the benefits of collaborative positioning for Android smartphone in controlled environment. The nominal scenario generated in this simulation is based on the statistical analysis made during the data collection campaign (see section 3.2.1). Nominal environment depicts an open-sky and static conditions where multipath and NLOS occurrences are mostly absent. Simulation results analysis will be considered as a preliminary framework for testing and investigating the implementation of *SmartCoop*. The simulation initialization and parametrization is firstly discussed. Thereafter, smartphone-based collaborative results will be revealed, exposing the performance of the CP method.

6.3.1.a Simulation Parametrization

The implemented algorithm is divided into two distinctive blocks. The first block, referred as the simulation block, generates a random collaborative network of smartphone users. The second block, called the estimation engine, defines the optimization problem and iteratively solves for newly smartphone collaborative positions. The Block diagram depicted by figure 6.4 shows the procedural steps for implementing this CP method.

The first block of this collaborative algorithm is in charge of randomly simulating network's users positions. The inputs to this block are: the desired number of users populating the network (n), the position error distribution parameters of $\tilde{p}$ (μ and σ) from section



Number of Users (M)
Position Error of $\tilde{p}$ Parameters $[\mu_{\tilde{p}}, \sigma_{\tilde{p}}]$
Geometry
Time Length
Simulation Block
GNSS Fix Positions $\tilde{p}$
[M x 3] matrix
True Positions
Gaussian Random Noise $[\mu_{IPR_{3D}}, \sigma_{IPR_{3D}}]$
IPR Ranges 3D Vectors
Estimation Engine
SmartCoop
Collaborative Positions
[M x 3] matrix

Figure 6.4: Block Diagram of *SmartCoop* Collaborative Engine Algorithm

3.3, the geometry of the network and the length of the simulated scenario (in epochs). This block will then output each smartphone user position that would be considered as their initial fix GNSS position by the *SmartCoop* algorithm. The last outputs of this block are the unbiased "true" users' positions, that will be used for simulating $\mathbf{IPR}_{3D}$. In practice, the simulation block randomly generates user positions at 1 Hz over a specified geographical area. Initial fix smartphones positions are randomly generated following a centered Gaussian distribution. Standard error deviation and a zero-mean Gaussian distribution used to generate those positions have been selected based on previous observations and analysis made on smartphones devices for static open-sky scenarios (see section 3.3.3). The error position standard deviation has been selected as: $\sigma_{\tilde{\mathbf{p}}} = [2.5, 2.5, 3.8]$ in meters on x, y and z in a ECEF reference frame for all simulated smartphones except for the *Best Node* user. Indeed, *Best Node* mobile is randomly selected within network's users. The error distribution for this phone has been set to $\sigma_{\tilde{\mathbf{p}}_{BestNode}} = [1, 1, 2]$ on x, y and z in meters, representing an higher positioning performance compared to the rest of the field.

In a second time, $\mathbf{IPR}_{3D}$ are generated from the unbiased "true" positions fed by the simulation block. Inter-user distances are computed by taking the norm of the difference between two independent user positions. Then, $\mathbf{IPR}_{3D}$ vectors are generated by adding a randomly generated Gaussian noise. The open-sky analysis made earlier (see section 5.3.1) on $\mathbf{IPR}_{3D}$ behavior in nominal cases, suggests that inter-phone distances can be approximated by a Gaussian distribution with $\sigma_{IPR_{3D}} = [1.75, 1.75, 2.5]$ in meters on x, y

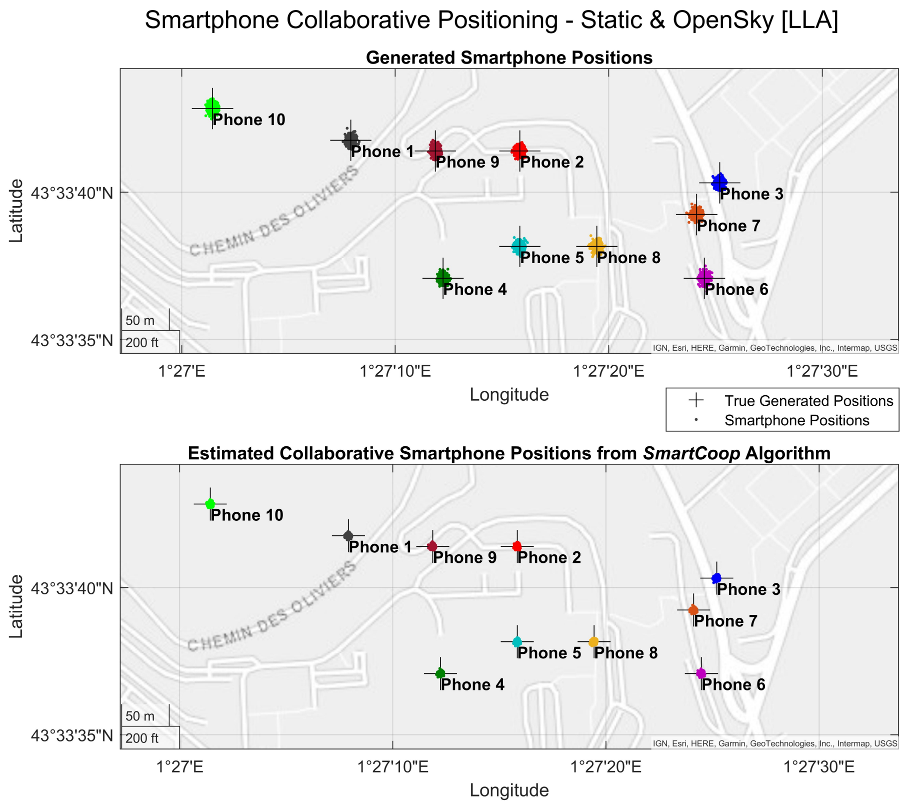

Smartphone Collaborative Positioning - Static & OpenSky [LLA]
Generated Smartphone Positions
Phone 10
Phone 1
Phone 9
Phone 2
Phone 3
Phone 7
Phone 5
Phone 8
Phone 4
Phone 6
CHEMIN DES OLIVIERS
Latitude
43°33'40"N
43°33'35"N
50 m
200 ft
IGN, Esri, HERE, Garmin, GeoTechnologies, Inc., Intermap, USGS
1°27'E
1°27'10"E
1°27'20"E
1°27'30"E
Longitude
True Generated Positions
Smartphone Positions
Estimated Collaborative Smartphone Positions from SmartCoop Algorithm

Figure 6.5: Simulated Smartphone-based Collaborative Network for a Static Open-sky Scenario - Positioning Performance Comparison in a LLA Coordinate System

and z in a North, East, Up (NEU) reference frame. Simulating smartphones positions and $\mathbf{IPR}_{3D}$ independently ensure that the added Gaussian noises do not correlate. Therefore, $\mathbf{IPR}_{3D}$ can be fed into the next estimation engine block.

Finally, GNSS fix positions and the simulated $\mathbf{IPR}_{3D}$ are inputted into the estimation engine. The estimation process of the *SmartCoop* algorithm is detailed in 6.2.1.c and in appendix D. Iteratively, collaborative positions are estimated on an epoch-by-epoch basis. This approach allows us to experiment and test the constraining capabilities of the algorithms with $\mathbf{IPR}_{3D}$. This simulation process outputs newly estimated smartphone collaborative positions.
The implementation of this algorithm on the Matlab environment allows to compute random cooperative networks with any number of users constituting the network. The following presented simulations represents a collaborative network of 10 users over a 1 hour time period, generating positions at 1 Hz (3600 epochs). During the computation process, the estimation procedure ran by the *fmincon* function takes on average 5 iterations per epochs for finding a local minimum that entirely satisfies all constraining equations.

6.3.1.b Results Analysis

This section will present the results analysis obtained for a nominal simulated scenario for our *SmartCoop* algorithm. In the presented simulation scenario, a network of 10 collaborative smartphone users was generated. Their initially generated positions are shown on figure 6.5, on the top graph. A random user geometry was used to position each individual user. Each dot represents an estimated position at 1 Hz frequency and the associated cross describes their reference position. Smartphone data and $\mathbf{IPR}_{3D}$ vectors have been simulated for 3600 epochs. In this scenario, smartphone user 2 has been randomly selected to be the *Best Node* of the network, pictured on the right hand side on Figure 6.5.

The above mentioned figure demonstrates the proof of concept for the *SmartCoop* algorithm. The top graph shows the smartphone positions before collaboration. The comparison can be directly made with the plot below, displaying the newly estimated collaborative positions. Collaborative positioning has improved all users position accuracy. Moreover, the positions dispersion have been reduced, thus increasing smartphone position precision.

Table 6.1 presents a statistical characterization of positioning error of the newly estimated cooperative positions. The first two columns show the statistical distribution of the newly estimated collaborative position error. Those parameters are to be compared with the simulated standalone position errors exposed in the simulation parametrization in 6.3.1.a as $\sigma_{\tilde{\mathbf{p}}}$. Figure 6.6 supports the observation made in table 6.1 by showing the position error distribution for a specific network user (phone 3) before and after collaboration. Thus, the last two columns of the table display measures variation between initial positions ($\tilde{\mathbf{p}}$) and collaborative positions from SmartCoop ($\hat{\mathbf{p}}$). Negative values indicate an improvement observed with our collaborative method. Essentially, this table validates the improved positioning performance induced by *SmartCoop*. All smartphones (except *Phone 2*) positions performances improve. In particular, the drastic improvement of the 95^{th} percentile ($\approxeq 2\sigma$) parameter confirms the positioning performance enhancement.

The *Best Node* phone represents an exception to the previous analysis, undeniably no improvement has been made on the global estimation of this user's position on its already (assumed) best positioning performance.

A second analysis can be made on an epoch by epoch basis. It has been established along multiple simulation runs, that our algorithm increases position accuracy more than 92% of the time for all smartphones within the network (except for the *Best Node* user where position accuracy was increased in 68% of the time). The average accuracy gain has been shown to be of ≈ 3.3 meters. In the few cases where newly estimated positions have not been improved, the investigation carried out pointed at a failure of the estimation process. Indeed, it appears that in rare cases, the minimization operation converged to an infeasible point where not all constraints could be satisfied and thus leading to erroneous position estimation.

Overall, the proficiency of the *SmartCoop* algorithm has been illustrated for static open-sky case scenario. This first analysis represents the framework for future work and acts as a proof of concept for the use of LS-based CP positioning technique. Furthermore, the constraining capabilities of the defined optimization process have been found to rely on the accuracy of cooperative double-difference GNSS-based computation of $\mathbf{IPR}_{3D}$ estimated vectors. Indeed, the estimation of precise inter-phone vectors characterize the tight implementation of the constraining equations. One could implement the *SmartCoop* algorithm using a more complex ranging method (i.e: LIDAR) for refining the collaborative position performance.

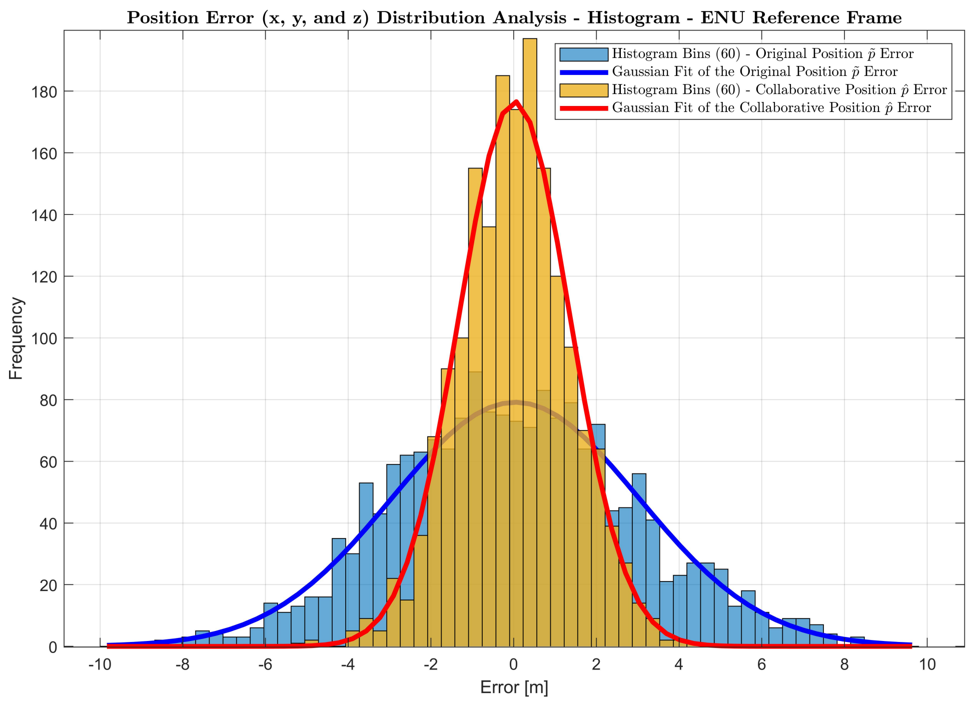


Figure 6.6: Position Error Distribution Analysis (combined error on x, y and z coordinates) for a Static Open-sky Scenario - Before and After Collaboration - Phone 3 - ENU Reference Frame

	SmartCoop Positions			
Network's Smartphones	$\mu_{\hat{p}}[m]$	$\sigma_{\hat{p}}[m]$	$\Delta\sigma[m]$	$\Delta 95\%[m]\,(2\sigma)$
Phone 1	0.06	1.61	-3.27	-4.60
Phone 2 (*Best Node*)	0.03	1.34	+0.11	+0.16
Phone 3	0.06	1.63	-3.26	-4.67
Phone 4	0.05	1.63	-3.23	-4.62
Phone 5	0.03	1.63	-3.31	-4.52
Phone 6	0.04	1.61	-3.22	-4.53
Phone 7	0.01	1.75	-3.27	-4.51
Phone 8	0.02	1.82	-3.29	-4.68
Phone 9	0.08	1.60	-3.21	-4.40
Phone 10	0.02	1.65	-3.22	-4.63

Table 6.1: SmartCoop Positioning Performance in Static Open-sky Conditions - Position Error Statistics Comparison (combining errors on x, y and z) between Standalone Position and Collaborative Position - ENU Reference Frame

6.3.2 Urban Canyon Scenario

The following analysis depicts a collaborative scenario in a dense urban environment. The positions of network's users have been selected within the core downtown area of Toulouse in south-west France. The goal of this simulation analysis is to test our *SmartCoop* algorithm in urban conditions where signals are affected by various disruptions. In opposition to the previous analysis for the nominal case scenario, smartphones' positions will be generated via a simulator tool called *SPRING* that takes into consideration local environment (using a georeferenced 3D model of buildings) for generating realistic masking and pseudorange impairments, and thus simulating life-like users' positions. Firstly, the simulation parametrization will be exposed for this urban analysis. Then, the generation of smartphone urban positions will be detailed with the introduction of the *SPRING* simulator. Finally, simulation results will be discussed, highlighting *SmartCoop* collaborative algorithm performance in deep urban environment.

6.3.2.a Simulation Parametrization

The creation of an urban scenario started by pre-determining smartphone's positions within a geographical area. In order to better represent life-like situations, network's users have been placed in various environments found in deep urban areas. Those local environments range from deep urban canyon (narrow streets) to dense urban (avenues), to semi-open areas (city squares) to finally open-sky areas (a park or on top of a bridge). For this scenario, 10 smartphones network users have been placed in the city center of Toulouse, France for a total duration of 30 minutes. Their pre-defined positions represent a wide variety of local environments. Figure 6.7 shows the created collaborative user network over a satellite image of Toulouse city center. The selected geographical area

represents the heart of Toulouse city center, where the two farthest cooperating users are at a distance of 1.41 kilometers.
As previously exposed in section 6.3.1.a, the *SmartCoop* algorithm outputs the newly estimated collaborative positions and remains identical compared to the previously presented engine in the nominal case scenario. The main difference comes from the way users positions are generated. In this scenario, smartphone's positions and associated pseudoranges are simulated by a 3D model-based GNSS software simulator. Pseudoranges are then used to compute $\mathbf{IPR}_{3D}$, following the algorithm presented in 5.2. Users network positions and estimated $\mathbf{IPR}_{3D}$ are used as inputs for setting up the non-linear constrained optimization problem in *SmartCoop*.

6.3.2.b Generating Smartphone Urban Positions

Simulating GNSS urban positions has always been a challenge for the research community. Urban environment navigation and positioning are heavily impacted by low satellite availability, NLOS signals and multipath occurrences. In order to generate realistic urban position, the complete propagation channel needs to be simulated including propagation effects due to buildings around the desired location. An advanced GNSS software simulator has been used for generating urban smartphone positions. In cooperation with CNES, *SPRING* 3D GNSS software was adopted for characterizing mobiles' urban positions.

SPRING is a GNSS software algorithm that aims at estimating the error due to multipath taking into account adjoining buildings and surfaces [112]. Thanks to a powerful implementation of a local 3D model, information about buildings geometry and surfaces can be retrieved. This model is provided by precise geo-referenced measurements. The detailed description of local geometry allows to individually simulate every signal of interest. GNSS signal propagation computation in *SPRING* is based on ray tracing principle.

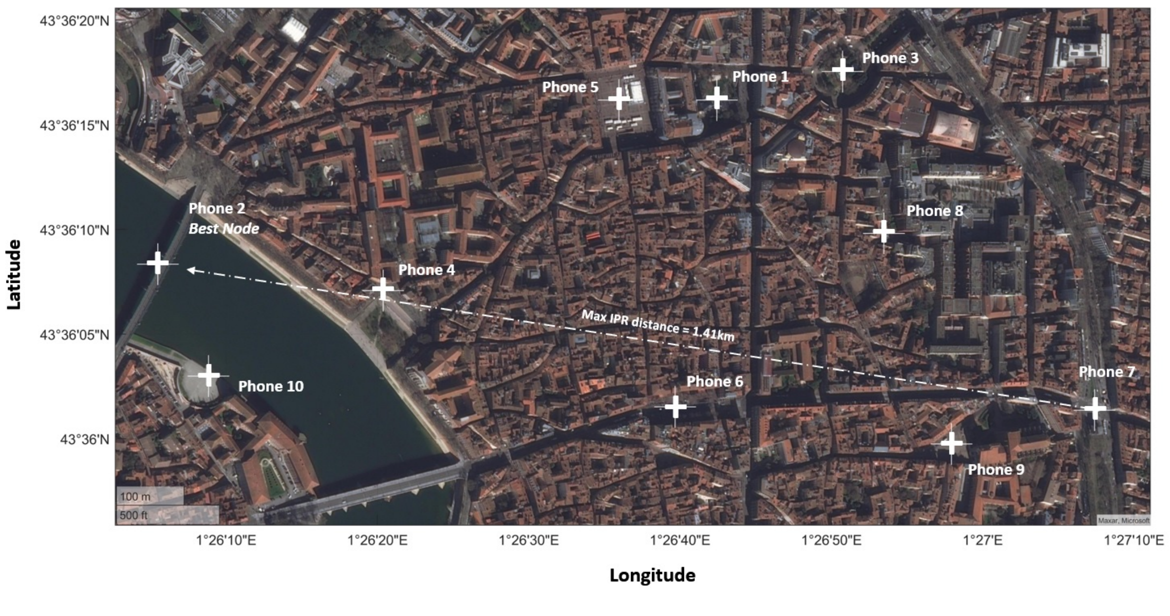
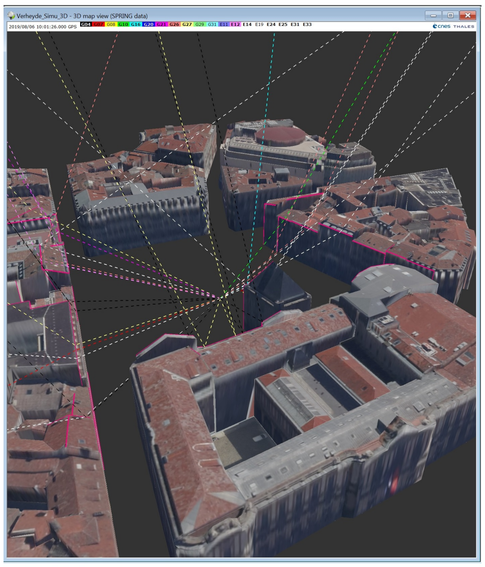


Figure 6.7: Simulated Collaborative Smartphone Network in an Urban Scenario. Satellite image from Maxar, Microsoft.


a)

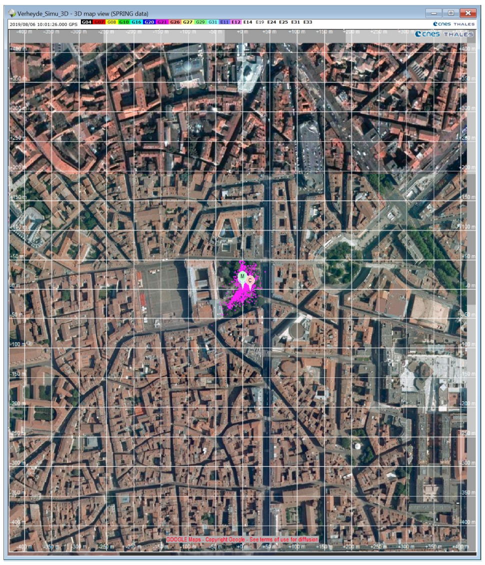

b)

Figure 6.8: *SPRING* Urban Positioning Simulation for Smartphone 1. a) Ray Tracing of Incoming GNSS Signals in the Generated 3D Environment. b) Plot of Simulated Positions [LLA].

This method is well-known for modeling various effects during the propagation channel: free-space propagation, diffraction, shadowing, reflection and refraction. Each ray is propagated in the scene until it intersects an object contained in the 3D modeled site where the generated receiver is located. The implementation of such algorithm allows for an accurate estimation of all available GNSS signals. Afterwards, realistic pseudoranges (biased by uncorrelated multipath events) are computed. Finally, PVT computations are made for estimating "life-like" smartphone's urban positions. Figure 6.8 captures the simulating processes used for estimating smartphone user 1. The figure on the left shows the different rays computed for the given urban position taking into account local architecture at a given epoch. The second picture on the right (b), shows the cloud of point positions computed for the entire scenario duration.

Reasons for Simulating Urban Positions with *SPRING*

Android-based positioning is currently being studied by multiple entities and advanced positioning algorithms taking advantage of this recent smartphone-based measurement are still under development. Our data collection campaign focused on understanding and analyzing urban positioning performances of Android devices. However, the collected data hardly represent a global representation and measurements models for urban positioning cannot be extracted from the samples of retrieved data.
One possible way forward consists of simulating urban positions from well-known propagation models in order to accurately represent Android measurements. The use of the *SPRING* simulator gives two main advantages to our study. First is the possibility to control and monitor the entire chain of simulation processes, thus creating with confidence a reliable collaborative network in constrained environment. And secondly, it offers the opportunity to assess the theoretical performance of our collaborative algorithm *Smart-Coop* in urban environment. The model used in the *SPRING* simulator is directly derived

from low-cost processes (hardware + antenna) and reflect well the model tuning and error model characteristics for smartphone-based positioning. The simulation was done for GPS L1 C/A observations at 1 Hz frequency.

6.3.2.c Results Analysis

The simulation has been performed applying the same method described in the section above. The goal of this analysis is to assess the performance of the newly estimated cooperative solutions compared to the initially generated urban positions. This study will also highlight the strengths and weaknesses of our collaborative positioning engine in degraded urban conditions.

The inputs to the *SmartCoop* algorithm remain similar to the ones exposed in 6.3.1.a. First, a network of 10 users is constituted in Toulouse city center. For this scenario, the *SPRING* GNSS software simulator was used for generating realistic GNSS pseudoranges for all users in a 30 minutes scenario. Network's users positions have been directly derived from those generated raw GNSS measurements. Furthermore, we used the code pseudorange from *SPRING* to estimate realistic IPR ranges. Both users' positions and $\mathbf{IPR}_{3D}$ vectors are then inputted in the collaborative engine. Figure 6.9 demonstrates the estimated inter-user distance between users 5 and 3. The obtained IPRd, figure *a*) on 6.9, correlates with our previous analysis made on static scenario. The plot on the right side of figure 6.9 *b*) characterizes the error distribution of our estimated IPRd between phones 5 and 3. The histogram of this error distribution (1800 samples per PRN) is characterized by a Gaussian distribution.

Figure 6.10 presents the results of this urban simulated scenario. The top plot depicts the initial positions of all network users. The bottom plot shows the newly estimated collaborative positions for each simulated individual. The first observation that can be drawn from this figure is that our collaborative engine significantly improved overall positioning

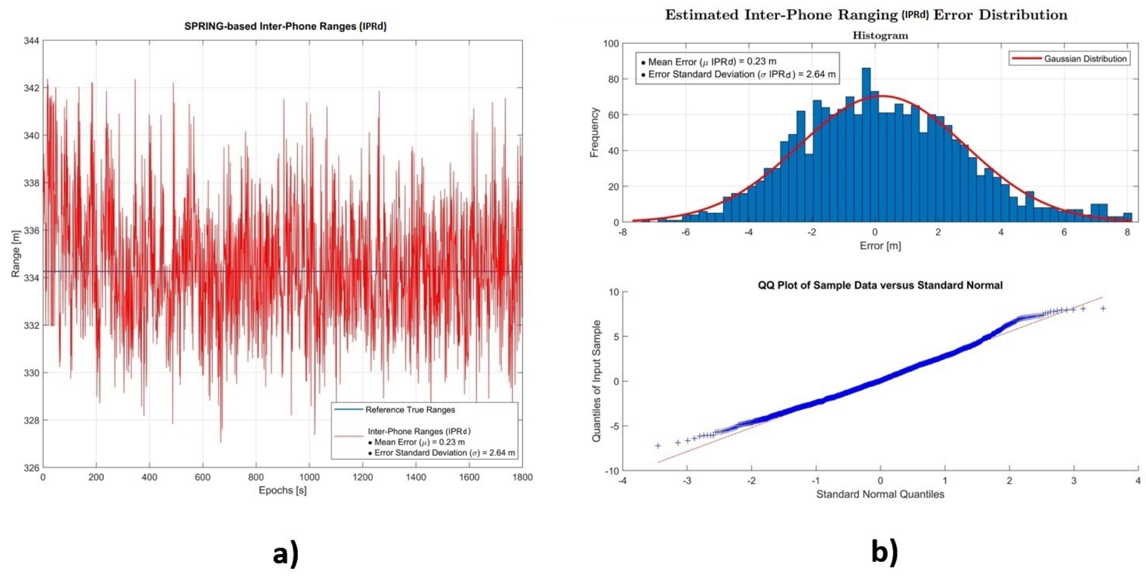


Figure 6.9: *SPRING*-based Inter-Phone Range Baseline (IPRd) Estimation between User 5 and 3.

solutions. Secondly, the position dispersion decreased and thus allows users to obtain a reliable positioning solution over time. All smartphones solutions have been improved by the collaborative solution, except for phone 2 the *Best Node* of the network.
Furthermore, it appears that all newly estimated positions share similar dispersion patterns. Figure 6.11 shows a zoomed view of the estimated collaborative positions of users 5 and 1. Both estimated positions dispersions are oriented in a north/south axis. This behavior is directly correlated to the constrained minimization process. This result demonstrates that all positions solutions are correctly bounded by $\mathbf{IPR}_{3D}$ vectors at every epochs. The restriction coming from the processing of constraints creates a grid structure (illustrated for a 4 user network in Figure 6.3) between network's users and thus generates a "network signature" on the resulting position dispersion. On the other hand, the orientation of the position dispersions are mostly due to the network geometry of users. This phenomenon is further discussed in section 6.4.2.

Moreover, this study compared the position solutions on a epoch by epoch basis. It has been found that for all smartphones the collaborative process improved the horizontal positioning solution in more than 75% of the time (except the *Best Node*). Plus, on average a gain of more than 10 meters has been seen compared to their original position. Table 6.2 describes individual smartphone positioning performance in function of their reference position. Error statistical characteristics are here described by the mean μ,

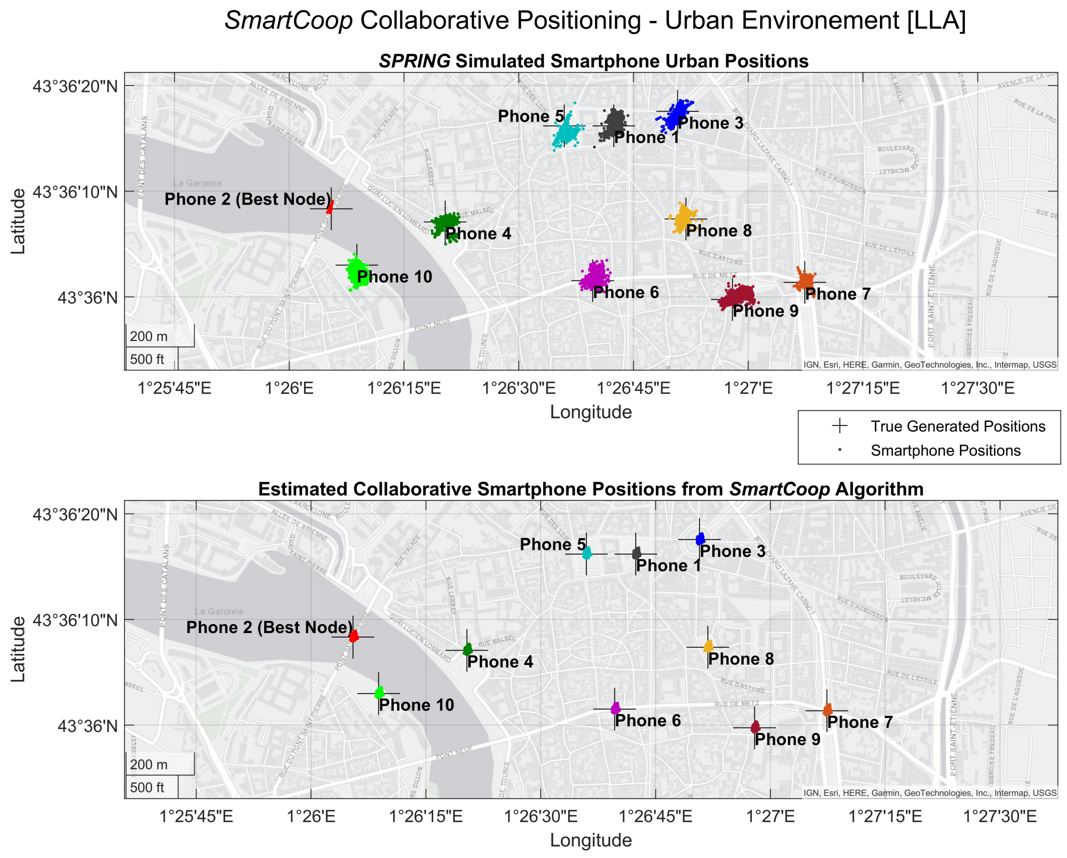


Figure 6.10: Smartphones Collaborative Positioning Solutions from *SmartCoop* Algorithm in Urban Environment

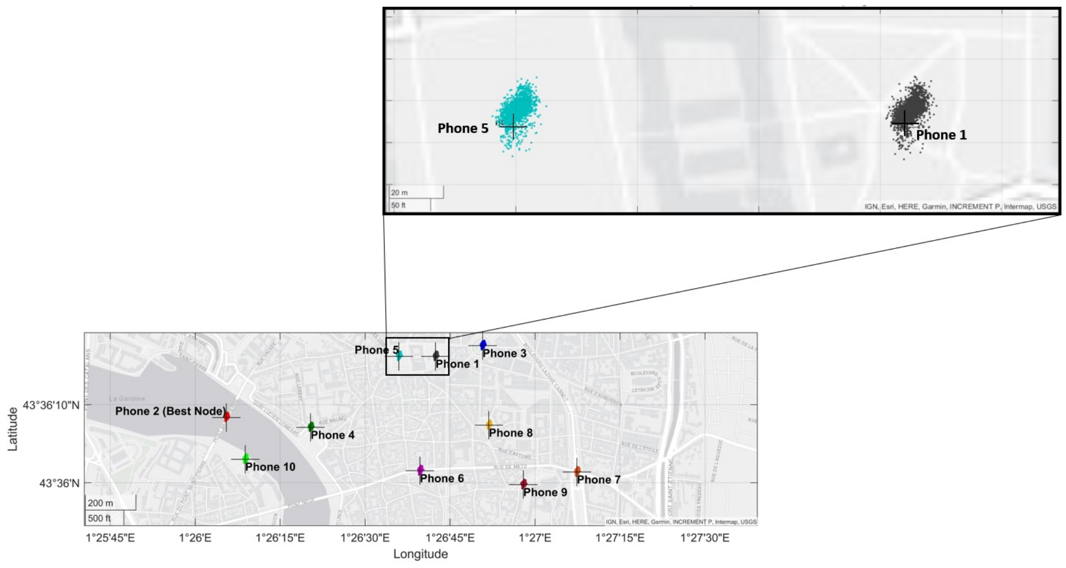


Figure 6.11: Smartphone Collaborative Solutions Zoom on Users 5 and 1

the standard deviation σ and the 95^{th} percentile value in meters. In this table, negative values represent a gain compared to the error characteristics generated from the initial position coming from *SPRING*. The presented results reveal that globally positioning error has been enhanced by tens of meters compared to the generated urban positions. This claim is supported by figure 6.12 comparing the positioning error of both original and collaborative solutions in the NEU reference frame. In this figure, smartphone user 6 is used as an example and similar results are found for all remaining network's users.

	SmartCoop Positions		
Network's Smartphones	$\Delta\mu[m]$	$\Delta\sigma[m]$	$\Delta 95\%[m]\ (2\sigma)$
Phone 1	-11.16	-8.21	-13.80
Phone 2 (*Best Node*)	+2.62	+1.10	+2.86
Phone 3	-7.30	-6.97	-9.93
Phone 4	-8.74	-9.22	-12.46
Phone 5	-8.88	-10.77	-13.57
Phone 6	-9.04	-8.32	-12.13
Phone 7	-4.54	-2.47	-5.17
Phone 8	-7.25	-2.19	-7.51
Phone 9	-21.94	-14.65	-26.35
Phone 10	-23.58	-11.64	-26.29

Table 6.2: SmartCoop Horizontal 2D Positioning Performance in Urban Environment Conditions - Position Error Statistics Comparison between Standalone Position and Collaborative Position - ENU Reference Frame

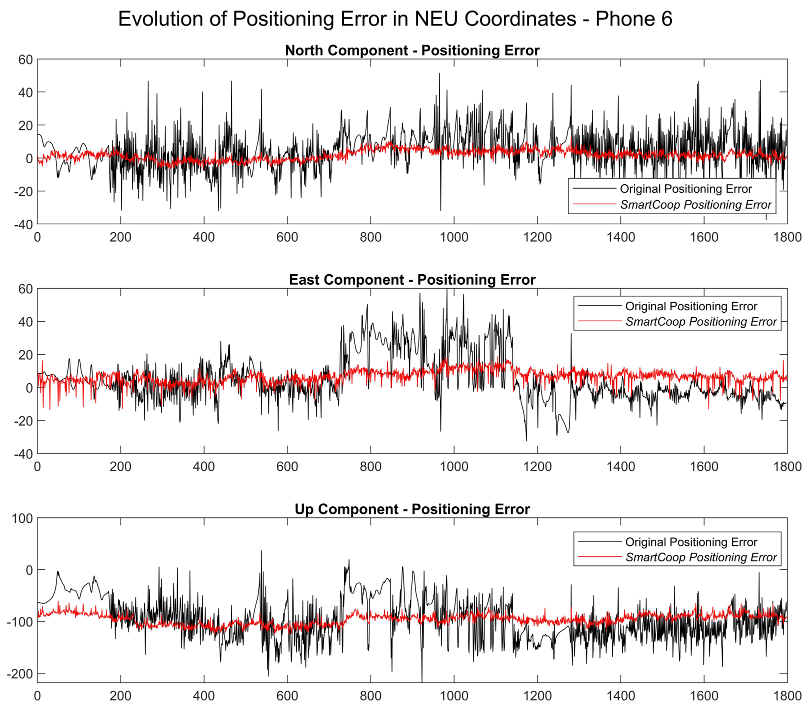


Figure 6.12: Generated Versus Collaborative Positioning Error Comparison in the NEU Coordinate Frame [m].

The positioning error is significantly reduced on the North and East components. The improvement on the standard deviation presented above is clearly highlighted by this figure. Furthermore, occurrences where the original position solution have been heavily impacted by characteristic urban impairments have been smoothed and mitigated by the collaborative estimation process. This event can be clearly identified on Figure 6.12, on the East component plot between epochs 750 and 1150.

On the other hand, a large error remains on the Up component of the positioning error. Similarly, the representation of the 3D error on adjacent figure 6.13 shows the impact of this error on the resulting 3D error plot. The origin of this bias can be linked to two distinct phenomena. Firstly, the generation of positions heavily depends on the 3D models implemented in *SPRING*. A wrongful setting of this model could have led to generate our initial position on sea level (elevation = 0 meter). This hypothesis is suggested by the error value on the vertical axis being equal to ≈120 to 130 meters, corresponding to the actual elevation of the city of Toulouse.

A second hypothesis emerges from the creation of the previously mentioned grid structures. In this scenario, no major elevation difference can be reported between smartphone position. Thus, naturally the created network of users sits on a common horizontal plane. We can consider that the constraining algorithm with our optimization problem does not account for the small variation on the vertical axis. The minimization process could potentially only restrict the initial positions on the horizontal axis while neglecting the change on the vertical component. This would explain why our *SmartCoop* algorithm did not correct the positioning errors on the Up component.

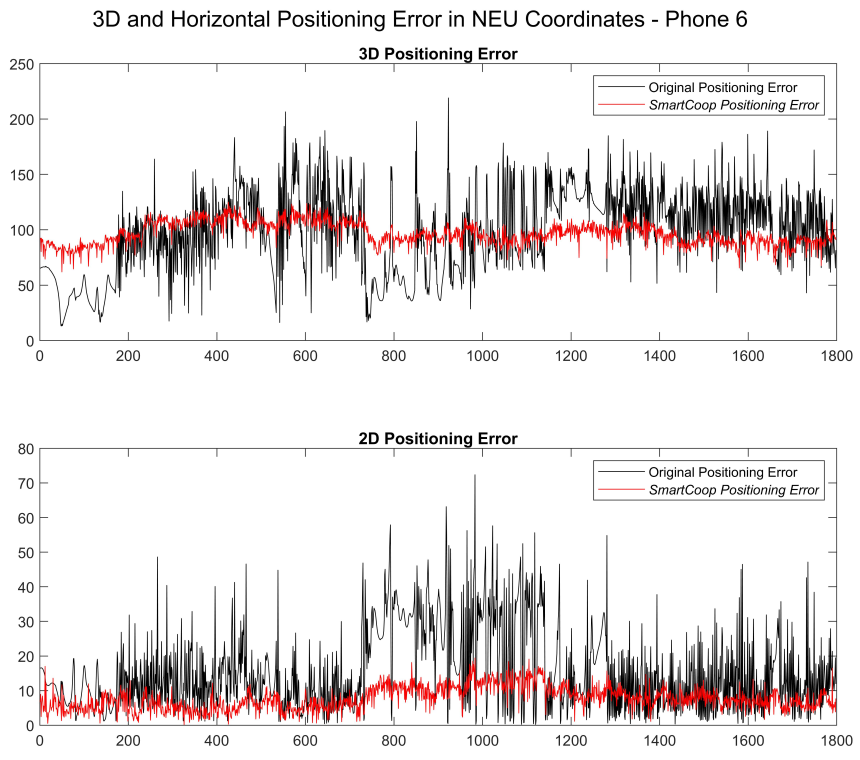


Figure 6.13: Horizontal and 3D Positioning Error Comparison in the NEU Coordinate Frame [m].

However, this issue can be tempered with the usage made by smartphone users. Indeed, smartphone-based positioning and navigation are mostly conducted in urban and sub-urban areas where elevation changes are minimal. Finally, smartphone-based positioning and navigation are mostly conducted in urban and suburban areas for users on the ground so the altitude is often forced to be the assumed ground level (e.g. from a Digital Elevation Model). For users that may not be on the ground, such as UAVs, altitude is often supplied by means other than GNSS and/or the GNSS-based altitude is augmented by additional sensors (e.g. barometer) [40].

Nevertheless, this problem can be mitigated via the implementation of additional measurements for restraining the vertical axis. Smartphone device offers sensors capable of constraining vertical user movements. It can be imagined that barometric measurements variation could be used as a constraining solution in our optimization process, fixing the vertical component on the cooperative grid structure.

In conclusion, urban smartphone-based positioning was successfully simulated and our *SmartCoop* algorithm have been studied. Results analysis confirmed the assessment made for nominal open-sky scenario. It was shown that the proposed collaborative algorithm can be used for improving smartphone network's users positions, in difficult environment for most of the cooperative individuals. Horizontal positioning performances were enhanced in more than 75% of the time, increasing position accuracy by a about 4.8 meters on average.

These results validate a proof of concept for a collaborative smartphone network. However, the presented conclusions rely on decisions and hypothesis made during the construction of the presented smartphone collaborative algorithm. Therefore, the following section will evaluated the collaborative algorithm outputs against the decision made and a detailed discussion studies the impact of the different hypotheses previously presented in section 6.2.

6.4 Algorithm Design Discussion

After unraveling the innovative concept of the *SmartCoop* algorithm and the performance through a simulation campaign, this section will evaluate our method against previously stated hypothesis and other method results available in the literature. Firstly, the concept of network Anchor or *Best Node* is revisited. This evaluation aims at assessing the benefits associated with the selection of an Anchor within the collaborative system. Then, the impact of network's user geometry will be provided for explaining the positions "signature" observed in previous analysis. Finally, a discussion will assert the impact of hypotheses made in this dissertation.

6.4.1 Impact of the Anchor

Initially, the role of the network's anchor was to mimic the reference position used in DGNSS advanced algorithms. In this study, the anchor user was denominated *Best Node*. The implementation of this anchor user was justified by real-life occurrences. A static user in an open-area (e.g: a park) could show better positioning performance compared to other users in adjacent urban environment. For this research work, the *Best Node* phone was used for correctly setting our constrained optimization problem. Its role was to be the anchor point for building around a network grid based on other satisfied constraints. This process is illustrated in figure 6.3.

The impact of the *Best Node* feature can be assessed by coming back to the results analysis in 6.3.1.b. The selected method for evaluating the effect of the network anchor can be made by simulating a similar network in similar conditions without picking a *Best Node* phone. Since positions are randomly selected within a certain area, the following simulation results will not be identical to the one presented on figure 6.5.

Figure 6.14 presents the collaborative positions of a 10 users network in open-sky conditions. Here, no *Best Node* phone has been selected and all smartphones have been simulated using identical distribution characteristics. As demonstrated by the plot on figure 6.14, newly estimated smartphone positions have all been improved by the *SmartCoop* algorithm. Thus, the selection of a system does not impact the final collaborative solution.
However, it appears that the computational time and complexity have been increased. In comparison to the analysis made above, the algorithm was found to be 35% slower. In fact, the *fmincon* function responsible for the resolution of the optimization problem took on average more iteration steps in order to both find a local minima and satisfying the given constraints.

Finally, we can conclude that the selection of an *anchor* user is not a requirement in the

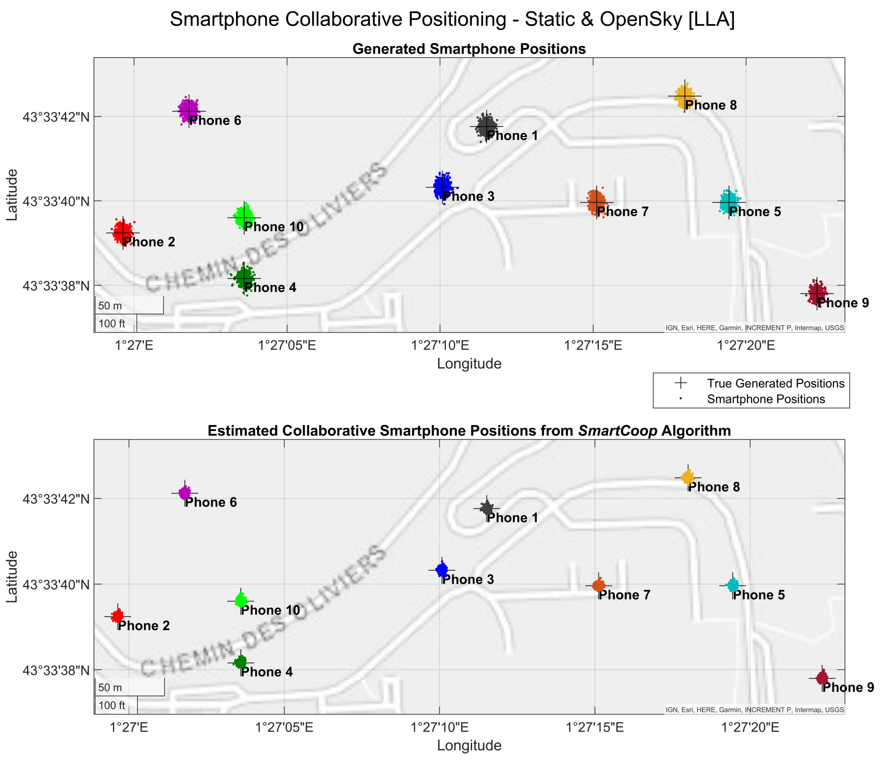


Figure 6.14: Simulated Smartphone-based Collaborative Network - Positioning Performance Comparison in a LLA Coordinate System - without a *Best Node* Phone.

implementation process of the *SmartCoop* algorithm but is recommended, if the situation allows, in order to reduce computational loads.

6.4.2 Impact of Users Geometry

In previous results analysis, we observed that the outputted collaborative positions could have similar ground dispersion that shared an identical signature. All the represented point clouds were oriented on the North/south axis in figure 6.12. This behavior could be correlated to the network geometry, meaning the geographical repartition of users within the collaborative network. This concept directly echoes to the DOP parameters computed in GNSS for assessing a sigma ratio between the precision of measurements and the position. This ratio is computed for evaluating the geometric distribution of satellites.
The goal of this analysis will be to evaluate the impact of users' geometry within the network. Two independent simulation analyses will be performed.

Firstly, a collaborative network of 6 mobiles is created for nominal test case scenario. All users are placed along an imaginary line, thus depicting the worst network geometry (on a single plane). In this case we expect the collaborative positions to be distributed along this axis. Figure 6.15 illustrates that scenario. The bottom plot reveals the position ellipses characterized by a semi-major axis along the East/West axis, as anticipated. This behavior might originate from the constraining characteristics of the collaborative engine.

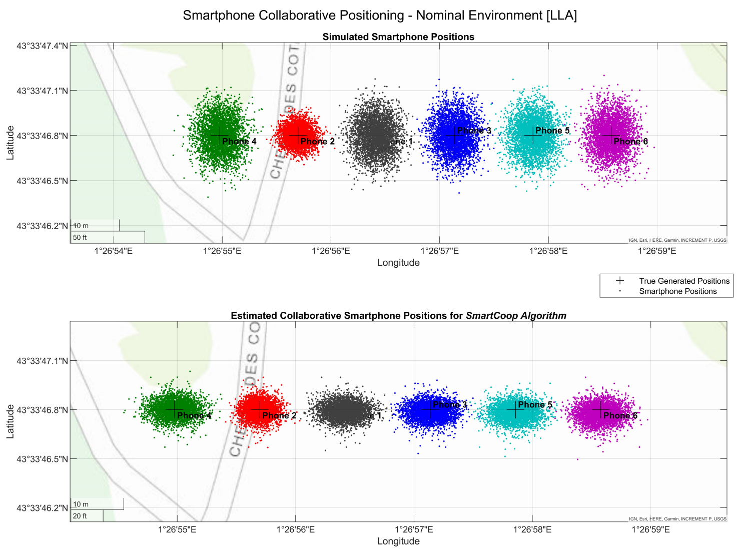

Smartphone Collaborative Positioning - Nominal Environment [LLA]
Simulated Smartphone Positions
Estimated Collaborative Smartphone Positions for SmartCoop Algorithm
Latitude
Longitude
Phone 1
Phone 2
Phone 3
Phone 4
Phone 5
Phone 6
True Generated Positions
Smartphone Positions

Figure 6.15: Simulated Smartphone-based Collaborative Network - Positioning Performance Comparison in a LLA Coordinate System - without a *Best Node* Phone.

All positions seem to be locked onto the horizontal plane formed by the initial positions. Afterward, the minimization process easily finds a local minima within the same plane. In this situation, multiple local minima cannot be characterized as a point peak, but rather as a line. The minimization process picks a local minima along the line and thus creates this position East/West dispersion signature.

Figure 6.16 displays a visual 3D representation of the constraints role for this scenario. At this epoch, a shift of all estimated collaborative positions took place. This bias is represented by the red arrows on the figure. This plot visually supports the discussion above.

Afterwards, a collaborative network with good user geometry was created in order to

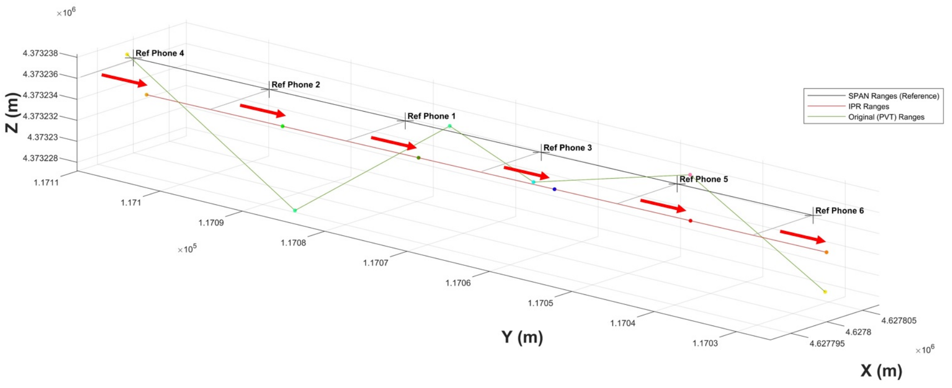


Figure 6.16: Simulated Smartphone-based Collaborative Network - Positioning Performance Comparison in a ECEF Coordinate System - without a *Best Node* Phone.

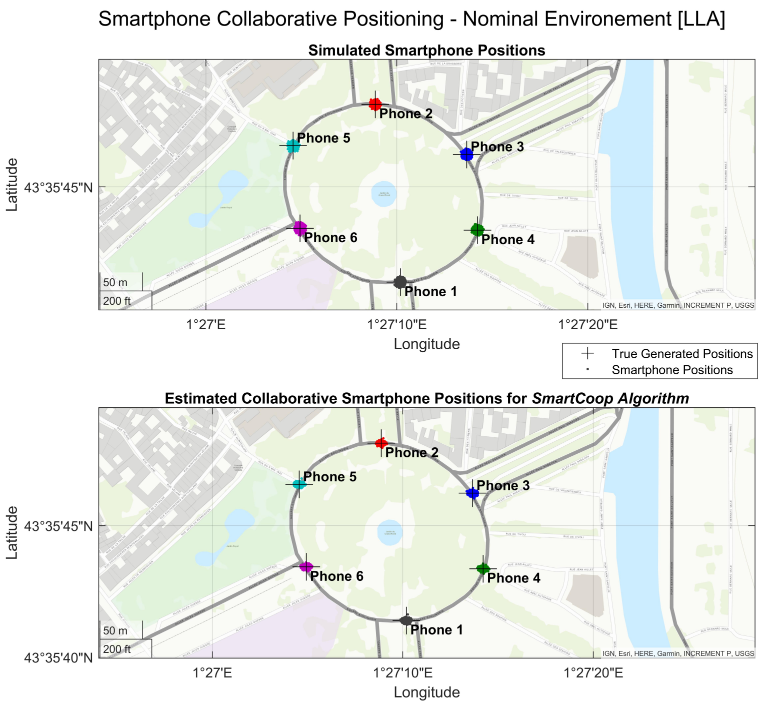

Smartphone Collaborative Positioning - Nominal Environement [LLA]
Simulated Smartphone Positions
Estimated Collaborative Smartphone Positions for SmartCoop Algorithm
Phone 1
Phone 2
Phone 3
Phone 4
Phone 5
Phone 6
Latitude
Longitude
43°35'45"N
43°35'40"N
1°27'E
1°27'10"E
1°27'20"E
50 m
200 ft
True Generated Positions
Smartphone Positions
IGN, Esri, HERE, Garmin, INCREMENT P, USGS

Figure 6.17: Simulated Smartphone-based Collaborative Network - Positioning Performance Comparison in a LLA Coordinate System - without a *Best Node* Phone.

confirm our hypothesis on the impact of user ground repartition. In this simulation, a scenario of 6 users were simulated around a circle. This disposition corresponds to a good geometrical distribution of network's users. Figure 6.17 demonstrates that the outputted collaborative positions point clouds have round shapes. The good user geometry allowed the *SmartCoop* algorithm to efficiently solve our optimization problem. Outputted positions shows now centered Gaussian distribution, round behavior.

6.4.3 Hypotheses Discussion

This section is discussing the impact of all hypotheses formulated in this dissertation for the creation of the presented collaborative algorithm. Moreover, this evaluation will lay the groundwork for improvement in the development of a collaborative smartphone network.

Independence of Smartphone Positioning Errors

During the mathematical derivation of the collaborative algorithm, it was assume that users positioning error are independent between all members of the collaborative network. This hypothesis is supported by the fact that users are mostly to be using different GNSS receivers (i.e different smartphone) for receiving and processing GNSS measurements. Moreover, for simplification purposes, we assumed that the user positioning error follow a Gaussian distribution and that their statistical parameters (μ and σ) were selected based on the characterization of smartphone measurements.

However, it can be argued that measurement errors between users are dependent. One axis for future improvement would be to characterized smartphone measurements with a more complex statistical distribution that would take into account correlation between collaborative users.

Independence of GNSS Positioning Errors
For simplification purposes, we assume that GNSS positioning errors on all components (x, y and z) were independent. However, as demonstrated by [24], this assumptions does not reflect real life applications. This hypothesis was formulated in order to validate a preliminary proof of concept for the presented collaborative engine. In future study, the error correlation between coordinates could be introduced by transforming the covariance matrix Σ_u introduced in section 6.2.1.c.

Independent Positioning Solution
The presented collaborative solutions outputted by the *SmartCoop* algorithm are independent in time. This process has been developed as snapshot technique, computing positioning solutions without apriori knowledge. A future implementation of collaborative positioning engine could be made recursively. A new collaborative position at an epoch t could be estimated with an apriori knowledge such as: its own previous positioning solution at epoch $t-1$ or the estimated IPR linking this agent to other network users.

Equality Constraints of IPR Vectors in Matlab
As discussed in section 6.2.3, the presented collaborative engine is characterized as a constrained optimization problem. This optimization highly depend on the estimated IPR vector constraining the tentative collaborative positions. In the current approach, the IPR vectors are estimated independently from the optimization problem. These IPR vectors are strictly constraining tentative positions without apriori knowledge of the previous estimation method. In future works we could consider implementing a covariance matrix of IPR vector estimates for weighting these constraints into the optimization process.

In Matlab, the *fmincon* function is used as the solver to our constrained optimization problem. In our collaborative network model, the IPR estimated vectors are used as strictly equality constraints. In theory, the Matlab solver tries to minimize the objective functions while strictly applying the equality constraints. However in practice, the *fmincon* solver rarely manage to find a local minima to the objective functions while strictly satisfying the equality equations set by the IPR vectors. A constraint tolerance (equal to 1^{-6}) is enforce by the solver preventing unsolvable processes. Furthermore, other tolerances and stopping criteria prevents the *fmincon* function to solve for strict constrained equations. The documentation concerning tolerance details for specific solver and algorithm can be found on the MathWorks website [113]. Thus, a fine tuning of Matlab solver could be studied for further improving the estimation of collaborative positions.

6.5 Chapter Conclusions

This chapter was introduced with a complete review of the state-of-the-art concerning collaborative hybridization filters. It was then followed by an argumentation for selecting the most adapted CP method to be implemented with Android devices. This led to the presentation of an innovative smartphone-based collaborative engine, called *SmartCoop*. A detailed description of this collaborative estimation engine was given. Android GNSS raw data measurements were put in the center of the attention with the interpretation of

constraints role within our non-linear constrained optimization problem.
In the wake of this presentation, simulations results analysis were displayed for two test case scenarios: nominal and urban environment. Both simulations results showed that newly estimated collaborative positions have reduced original horizontal position errors. In difficult environment, a gain of a couple of meters was established on user's positioning solutions. *SmartCoop* collaborative engine showed reliable performance enhancing global network positioning capabilities.
Future works will aim at mitigating resulting positioning error on the vertical axis. It is ambitioned a second layer of restriction on the vertical axis can be applied to the characterization of our collaborative engine. Additional smartphone's sensors provide entrancing measurements for constraining the future implementation of our algorithm. Furthermore, *SmartCoop* remains to be tested with real Android GNSS raw data measurements for validating the simulated results. After regrouping a fleet of tens of smartphones, a data collection campaign could retrieve measurements from those devices placed on geo-referenced points. This step is essential for characterizing *SmartCoop* algorithm in real conditions before implementing a life-like collaborative network.

7 Conclusions

This chapter is concluding and synthesizing the presented research work of this PhD thesis. A brief summary will display the progress and the scientific contributions made in this research study. The conclusion highlights the completion of initially set objectives. Firstly, a conclusion will epitomize the main steps and successes leading to the development of *SmartCoop* positioning engine. Furthermore, implementation solutions and commissioning strategies are exposed for exploiting our collaborative system. Finally, future software implementation and future works are discussed for improving collaborative estimation processes.

Conclusions

Overall, this research work successfully developed a smartphone collaborative positioning engine enhancing users positioning performance in urban environments. In the first part of this thesis, smartphone positioning characteristics were studied for better apprehending Android GNSS measurements. An assessment methodology, derived from our data collection campaign, statistically described positioning performance in constrained environments for a broad spectrum of smartphone brands and models.
In the second part of this research work, the selection of a smartphone-based collaborative network has been introduced. A GNSS double difference technique was implemented and optimized for efficiently estimating inter-phone 3D distance vectors. Finally, an innovative smartphone cooperative positioning engine has been developed, named *SmartCoop*. This algorithm is based on the previously estimated Inter-Phone vectors ($\mathbf{IPR}_{3D}$) used as dynamic constraints in an non-linear optimization problem.

A detailed simulation result analysis revealed a significant positioning accuracy increase in both open-sky and in urban conditions. Simulated network's smartphone users reported a positioning solution improvement more than 75% of the time. Moreover, our smartphone collaborative positioning engine decreased position error standard deviation by approximately a factor of 1.5, thus producing a reliable solution over time.

Applications for *SmartCoop* Positioning Engine

The following section debates the application opportunities that could fit the use of a smartphone-based cooperative engine. The authors believe that this positioning algorithm could be used within a pre-defined system.
A first suggestion would be to integrate the *SmartCoop* algorithm within Google services. This implementation would profit from the computational capabilities of the American firm, plus an unprecedented access to an abundance of measurements for creating local cooperative networks. It can be envisaged that our collaborative solution could be directly added to the Google localization services for enhancing FLP positions in urban environments.
A second suggestion could be the commission of our collaborative algorithm by a smartphone or a chipset manufacturer. The natively execution of our solution on branded devices could assist the development of certain brands and would participate in the technological race engaged between manufacturers. These two propositions allow for better understanding the role of a collaborative algorithm in the smartphone positioning domain.

Perspectives

This section explores the future studies to be conducted following up the presented research work. The perspectives exposed below are direct contributions to be implemented with the *SmartCoop* cooperative positioning engine.

- Extend the Data Collection Campaign for Refining Measurement Models
  Additional data measurements would be needed for validating the results drawn by the different analyses presented in this thesis. A future data collection campaign would be necessary for assessing Android GNSS measurements characteristics of new smartphone brands, chipset models and firmware updates. The constitution of a dense and robust dataset would allow for the implementation of a regression system dedicated to the improvement of measurement model and collaborative algorithm performance.

- Analyzing Low Dynamic Scenario
  The presented research focused on static scenarios in urban environment. Future works should study the impact of dynamic scenarios on the navigation solution outputted by *SmartCoop*. In urban conditions, low dynamic navigation scenarios are mostly conducted with smartphone devices. Higher dynamics type of events should be thoroughly analyzed. In this situation, we expect smartphones' embedded receivers to struggle due to untimely cycle slip events and frequent losses of lock.

- Real Condition Testing for *SmartCoop*
  After a successful simulation result analysis made on the cooperative positioning filter *SmartCoop*, these results should be validated in real condition testing. The processing of Android GNSS recorded raw data measurements are to be implemented within our given cooperative engine. It is ambitioned that a reduced collaborative smartphone network can be created. Each smartphone should be placed on geo-referenced points within an urban area for enabling a post-processing reference point. Recorded data could then be inputted into our cooperative engine for analyzing smartphone positioning performance.

- Adding New System Constraints
  The presented cooperative algorithm takes advantages of the Inter-Phone vectors ($\mathbf{IPR}_{3D}$) for constraining a non-linear optimization problem. As shown in 6.3.2.c, vertical position error remains high. A solution to that issue could be the introduction of additional constraints to our system. Smartphone are equipped with different environmental sensors that could be used in this interest. As an example, the derivation of barometric measurements could be a source of information concerning the vertical dynamic of the mobile. This measurements could be implemented as a constraint, tightening our vertical position estimate on the correct vertical plane.

- Implementation of a Collaborative Network for Direct Application
  After validating the experimental results with real data measurements, the implementation of the *SmartCoop* engine in real-time could be the next achievement. This implementation could be of different form. The creation of a dedicated Android API for collaborative positioning could potentially lead to the release of the presented algorithm on mass-market devices. The example of the European project, Flamingo [114], could be taken as an initial release plan.

Publications

International Journals

[**2020a**] Verheyde. T, Blais. A, Macabiau. C, Marmet. François-Xavier, *"Analyzing Android GNSS Raw Measurements Flags Detection Mechanisms for Collaborative Positioning in Urban Environment"* IEEEXplore, vol.2020, International Conference on Localization and GNSS(ICL-GNSS 2020), pp. pp. 1–6, doi: 10.1109/ICL–GNSS49 876.2020.9 115 564, 2020.

[**2021**] Verheyde. T, Blais. A, Macabiau. C, Marmet. François-Xavier, *"SmartCoop Algorithm: Improving Smartphones Position Accuracy and Reliability through Collaborative Positioning"* IEEEXplore, vol.2021, International Conference on Localization and GNSS(ICL-GNSS 2021), pp. pp. 1–6, doi: 10.1109/ICL–GNSS49 876.2020.9 115 564, 2021.

Conference Proceedings

[**2020b**] Verheyde. T, Blais. A, Macabiau. C, Marmet. François-Xavier, *"An Assessment Methodology of Smartphones Positioning Performance for Collaborative Scenarios in Urban Environment"* Proceedings of the 33rd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2020), pp. pp. 1893–1901., Sep. 2020.

Presentation in International Conference

[**2019**] Verheyde. T, Blais. A, Macabiau. C, Marmet. François-Xavier, *"Statictical Analysis of Android GNSS Raw Data Measurements in an Urban Environment for Smartphone Collaborative Positioning Methods"* International Navigation Conference (INC 2019), Edinburgh, UK. 2019

A

Global Navigation Satellite Systems

Contents

Global Navigation Satellite System (GNSS) defines a satellite-based navigation and positioning system with global terrestrial coverage. This type of system provides users a three-dimensional positioning and timing solution in a georeferenced frame. Precise processing of transmitted radio signals, continuously broadcasted by orbiting satellites, allows for user positioning solutions. Positioning solutions are estimated by specialized receivers using trilateration techniques.

This appendix chapter will present the different segments that define those navigation and positioning systems. Satellite constellations and signals descriptions used in this research will be described. Afterward, a detailed overview of GNSS receiver architecture will be provided. Constellations and signals, used in this research work, will be explicitly detailed in this chapter.

A.1 GNSS Systems

Global constellation refers to the composition of orbiting satellites family. Most constellation are constituted of a core 27 or more Medium Earth Orbit (MEO) satellites. This satellite-based positioning and navigation system provides pin-point location and timing services. This systems is widely used worldwide for various daily operations. Multiple

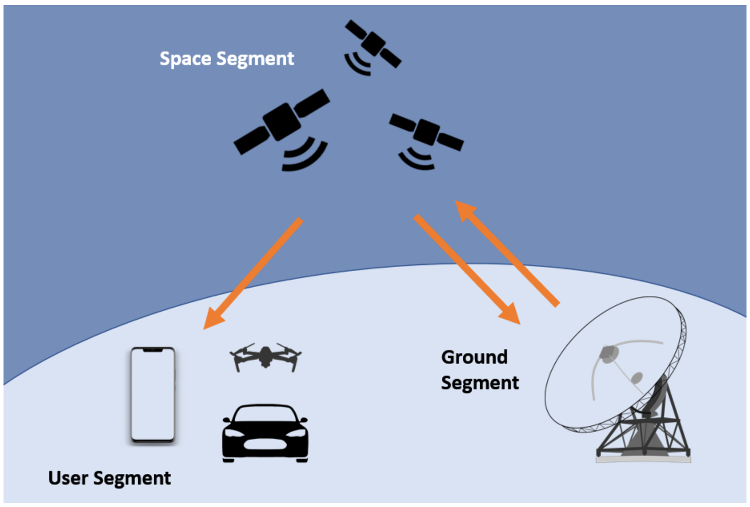


Figure A.1: GNSS Segmentation Structure

global positioning systems are currently in service. However, the structure of all GNSS system is common. GNSS systems are composed of three distinct segments: the user segment, the space segment and the control segment. GNSS system architecture is described in figure A.1. This system of satellites is often referenced as a passive structure, enabling users to receive signals without interacting directly with the emitting spacecraft. This particularity makes the system scalable resulting in a unlimited user capacity benefiting from GNSS services.

A.1.1 Space Segment

The space segment is characterized by satellite constellation orbiting the Earth. A constellation is made of multiple satellites broadcasting radio signals towards the planet's surface. Multiple GNSS constellations are already active and are presented below.

1.1.1.a GPS

The first GNSS developed is the American system called Global Positioning System (GPS) in the end of the 20th century. It was originally created by the U.S Department of Defense for military purposes. Nowadays, the system is owned by the United States government and operated by the United States Space Force. Currently the GPS constellation is constituted by 31 operational satellites. Satellites have been developed over the last fifty years and are segmented in specific blocks. As of today, GPS satellites come from 4 blocks ranging from legacy to modernized systems. All existing blocks are listed below:

- **Block IIR**: 8 Operational Satellites
  Launched between 1997 and 2004
  2nd generation of legacy GPS satellites.
- **Block IIR-M**: 7 Operational Satellites
  Launched between 2005 and 2009
  Introduced 2nd civil signal on L2 band and a new military code signal.
- **Block IIF**: 12 Operational Satellites

Launched between 2010 and 2016
Introduced 3rd civil signal on L5 band and an improved on-board atomic clock.

- **GPS III/IIIF**: 4 Operational Satellites
  Current satellite generation. First Launch in 2018
  Introduced 4th civil signal on L1 band (L1C).

Figure A.2 shows the GPS satellites orbital plane positions and is divided into their respective blocks. Three frequency bands are available on GPS. The L1 band, centered at a carrier frequency of 1575.42 MHz, is used to broadcast four signals (including code M). The most commonly used is the L1 C/A signal. The second one is the encrypted precision military signal called L1 P(Y). Most recently, the L1C signal has been implemented on the newest generation of GPS satellites within the Block III. The second frequency band, L2 band centered at 1227.60 MHz, emits two signals: the L2C for civilian purposes and the L2P(Y). Finally, L5 signals are transmitted on the 5^{th} frequency band, centered around 1176.45 MHz.
GPS utilizes their numerous frequency bands to provide two main services to their users. The first service is referred as Standard Positioning Service (SPS). This service is free of charge and provide position, velocity and timing to a plethora of civil and commercial users. The second service, called Precise Positioning Service (PPS), is an encrypted service mainly for military and government purposes. Table A.1 regroups the characteristics of the GPS space segment.

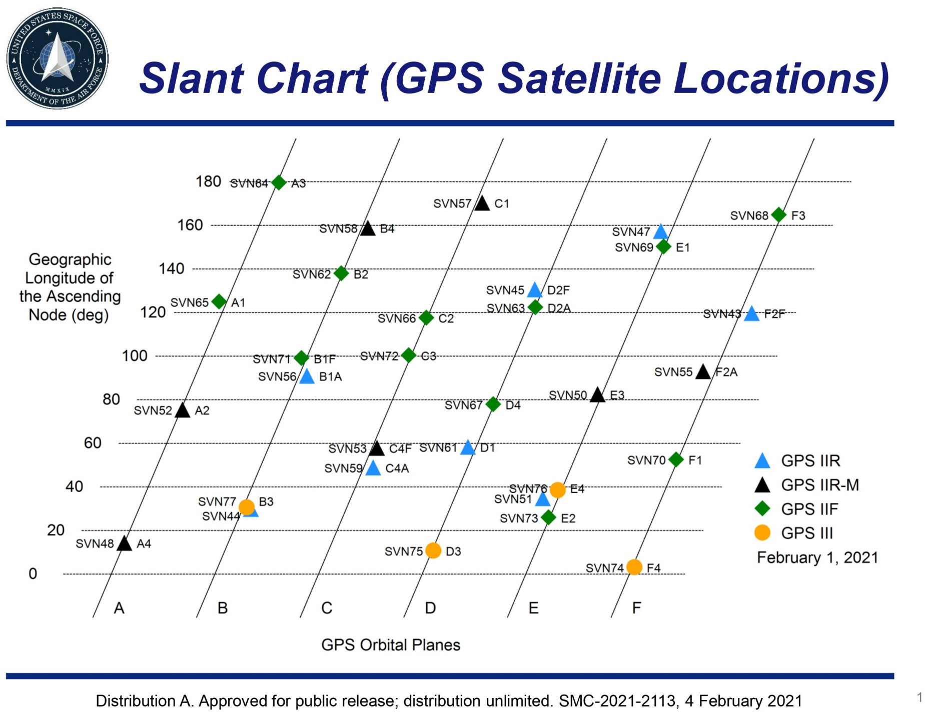


Figure A.2: GPS Satellites Orbital Locations [15]

1.1.1.b Galileo

At the beginning of the 20th century, the European Union wanted to pursue their independence toward navigation solutions and developed their own satellite navigation system called Galileo. The European system is currently operated by the European

Union Agency for the Space Program (EUSPA). The main differentiating factor of the Galileo system is the plurality of services that are offered by the navigation system. The European GNSS system offers 4 services as listed below:

- **Open Service (OS)**
  The service provides Position, Navigation and Timing (PNT) solutions free of charge mainly targeting civil applications.
- **High Accuracy Service (HAS)**
  This service aims at market application and can be used as a complement to the OS Galileo service for professionals requiring high accuracy solutions.
- **Public Regulated Service (PRS)**
  This service is intended to be used by authorized entities. It has been designed as a robust and encrypted service that provides high accuracy PNT solutions to European institution under all circumstances.
- **Search and Rescue (SAR)**
  This unique service contributes to the international cooperative effort for search and rescue activities. The particularity of this COSPAS-SARSAT service is that it offers a return link service for distressed users directly built in within the satellites' payload.

Navigation signals are broadcasted on four frequency bands by Galileo. These four frequency bands are called E1, E5a, E5b and E6. The frequency plan of Galileo offers wider bandwidth for signal transmission than GPS. Table A.1 regroups the characteristics of the Galileo space segment.

1.1.1.c GLONASS

The Globalnaya Navigatsionnaya Sputnikovaya Sistema (GLONASS) system has been developed by the Soviet Union has a counterpart to the GPS during the historic cold war. This constellation's space segment is populated by 27 satellites in orbit. Satellites deployed in the GLONASS constellation have been segmented into 3 specifications. Currently, only the two most recent satellite blocks are in service. GLONASS-M satellites are the second generation that modernized the characteristics of the orbiting payloads. The current satellite generation is called GLONASS-K and was firstly launched in 2011. The particularity of GLONASS is that the system was implemented via a legacy Frequency Division Multiple Access (FDMA) technique for differentiating incoming satellite signals. Table A.1 regroups the characteristics of the GLONASS space segment.

1.1.1.d BeiDou

The BeiDou Navigation Satellite System (BDS) architecture has been developed by China. The original Chinese regional navigation system became a global navigation system in 2011. Nowadays, the BDS space segment is constituted of 44 operational satellites. This system offers multiple services that differ from other GNSS constellations, due to its pedigree. BeiDou services can be divided into three categories:

- **Radio Navigation Satellite Service (RNSS)**
  This service provides navigation solutions that are comparable to other GNSS systems. Similar to the Galileo constellation, an open service is available free of charge

for civilian purposes. Encrypted positioning and navigation services are also implemented via an authorized signals service.

- **Radio Determination Satellite Service (RDSS)**
  This unique regional service gives users a positioning service computed by ground stations via geostationary satellites. RDSS also provides short message radio communication and other message communication capabilities.

- **Wide-area Differential Service**
  This service is supported by BeiDou embedded augmentation system currently activated on the constellation.

BeiDou RNSS service can be considered a global service, whereas RDSS and Wide-area Differential services are referred as regional services and are directly embedded inside the BeiDou global constellation. The development of satellites has been implemented in three distinctive phases. Currently the BeiDou constellation has reached Full Operational Capability (FOC) in the phase 3 of its development. Table A.1 regroups the characteristics of the BDS space segment.

1.1.1.e Regional & Augmentation Systems

At first, the position accuracy delivered by GNSS was suitable for positioning and navigation in open-sky environment. Then, complex requirements were established by the civil aviation for aircraft precision landing that could not be met by standalone GNSS. Structure and system were put in place in order to address those issues. Two types of system were put in place for assisting already existing GNSS structures: Regional and Augmentation systems. Regional satellite systems are small satellites constellation that focus their operation on a specific geographical area and therefore cannot be considered global. They are mainly used in addition to global satellite constellations. Augmentation systems are small independent satellite constellation whose role is to enhance GNSS performance. Civil aviation system called Satellite-based Augmentation System (SBAS) provides regional augmentation through the use of geostationary satellites that transmit correction data.

- **Quasi-Zenith Satellite System (QZSS)** The Japanese regional system was developed by the Japan Aerospace Exploration Agency (JAXA) in order to answer urban canyons problematic. This system is composed of four satellites (geosynchronous and geostationary) that supportively allow users to permanently have a satellite in view in deep Japanese urban environments.

- **Navigation Indian Constellation (NAVIC)** The Indian regional system, previously named Indian Regional Navigational Satellite System (IRNSS), possesses seven orbiting satellites and is described as an independent and autonomous regional navigation entity. Future developments suggest that NAVIC might be the preliminary step toward an Indian global navigation system called Global Indian Navigation System (GINS).

- **European Geostationary Navigation Overlay System (EGNOS)** The European augmentation system, EGNOS, was developed to complement GPS. This system service a wide continental area ranging from the north of Africa to the northern part of Europe. EGNOS is being monitored by European Satellite Services Provider (ESSP) and has been developed by European Space Agency (ESA).

This European service allows to increase integrity and accuracy performances of system users. Future evolution of EGNOS, labeled V3, is planed to augment multi-constellation (GPS and Galileo) and multi-frequency data.

- **Wide Area Augmentation System (WAAS)** The American augmentation system was the first one to be implemented and is operated by the Federal Aviation Administration (FAA). The service area covers the continental United States plus Canada and Mexico. Currently three geostationary satellites compose its space segment. Augmentation service is designed to provide augmentation solutions to the American GPS system [115].
- **Multi-functional Satellite Augmentation System (MSAS)** The Japanese SBAS is composed of two geostationary satellites. This system is intended to augment GPS L1 signals. The system coverage range through most Asia and Oceania. The system was declared by the Japan Meteorological Agency (JMA) operational in September 2007.
- **GPS Aided Geo Augmented Navigation System (GAGAN)** This augmentation system has been created by the Indian government in early 2008. Three geostationary satellites have been launched allowing the system to cover the Indian country in totality.

	GPS	**Galileo**	**GLONASS**	**BeiDou**
Coverage	Worldwide	Worldwide	Worldwide	Worldwide + Regional
Operator	United States	European Union (EUSPA)	Russia	China
Status	Full Operational Capability (FOC)	Full Operational Capability (FOC)	Full Operational Capability (FOC)	Full Operational Capability (FOC)
Services	Precise Positioning Service (PPS) Standard Positioning Service (SPS)	Open Service (OS) High Accuracy Service (HAS) Public Regulated Service (PRS) Search and Rescue Service (SAR)	Standard Precision Service (SP) High Precision Service (HP)	Open Service Authorized Service Wide Area Differential Service Short Message Service
Satellites	MEO: 31 satellites in orbit 31 satellites in operation	MEO: 28 satellites in orbit 24 satellites in operation	MEO: 27 satellites in orbit 22 satellites in operation	MEO: 49 satellites in orbit 44 satellites in operation
Orbital Height	20,180 km	23,222 km	19,130 km	21,150 km
Period	11 h 58min	14 h 05min	11 h 15min	12h 53min
Signal Plan	L1 band: 1575.42 MHz L2 band: 1227.60 MHz L5 band: 1176.45 MHz	E1 band: 1575.42 MHz E5a band: 1176.45 MHz E5b band: 1207.14 MHz E6 band: 1278.75 MHz	L1 band: 1598.06 - 1605.37 MHz L2 band: 1242.93 - 1248.62 MHz L3 band: 1207.14 MHz	B1I band: 1561.09 MHz B1C band: 1575.42 MHz B2a band: 1176.45 MHz B3 band: 1268.52 MHz
Modulation (L1)	BPSK	CBOC	BPSK	BOC
Coding (L1)	CDMA	CDMA	FDMA and CDMA	CDMA

Table A.1: Current GNSS Characterization and Description

A.1.2 Control Segment

GNSS control segment role is to monitor the services offered by their global navigation systems. Control centers of a specific satellite constellation are dispatched all over the world in order to always be able to track the desired satellites. Their activity ranges from monitoring signals characteristics, satellite orbital parameters, check health parameters, update navigation messages and solve potential payload anomalies.
In order to illustrate the infrastructure necessary to perform all ground segment task the Galileo control segment will be presented. Other GNSS constellations use similar type of infrastructure for monitoring their global navigation systems.
The Galileo control segment is composed of two control centers: Galileo Control Centers (GCS) and Galileo Mission Segment (GMS). A ground station network, pictured on figure A.3, has been implemented for executing monitoring tasks.

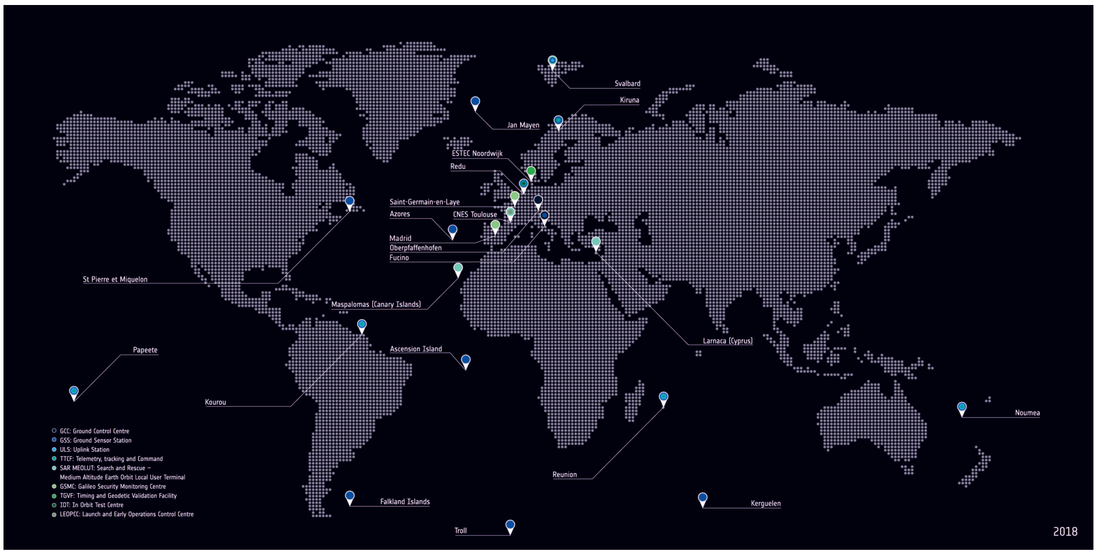

Svalbard
Kiruna
Jan Mayen
ESTEC Noordwijk
Redu
Saint-Germain-en-Laye
Azores
CNES Toulouse
Madrid
Oberpfaffenhofen
Fucino
St Pierre et Miquelon
Maspalomas (Canary Islands)
Larnaca (Cyprus)
Papeete
Ascension Island
Kourou
Noumea
Reunion
Falkland Islands
Kerguelen
Troll
GCC: Ground Control Centre
GSS: Ground Sensor Station
ULS: Uplink Station
TTCF: Telemetry, tracking and Command
SAR MEOLUT: Search and Rescue – Medium Altitude Earth Orbit Local User Terminal
GSMC: Galileo Security Monitoring Centre
TGVF: Timing and Geodetic Validation Facility
IOT: In Orbit Test Centre
LEOPCC: Launch and Early Operations Control Centre
2018

Figure A.3: Galileo Ground Segment: Ground Station Network [16]

A.1.3 User Segment

The user segment is made of every GNSS's user receivers that are able to process GNSS transmitted signals. User's receivers are devices that are capable of processing radio signals broadcasted by orbiting satellites in order to estimate PNT solutions. Such receivers are used by everyone on a daily basis. They can be found in vehicle navigation devices, wearables and most importantly for this research thesis in all smartphones.
A GNSS receiver can be modeled into three blocks:

- **Radio Frequency (RF) Front-end Block**
  Receive and digitize GNSS RF signals
- **Signal Processing Block**
  Signal acquisition and tracking producing observables
- **Data Processing Block**
  PNT computations are made in order to extract user's position.

The principle of a GNSS user receiver is detailed below and illustrated by the block diagram on figure A.4. First of all, the signal is captured by the receiver's antenna; then, it

is processed by the Front-end block. This receiver's function performs down-conversion, filtering, amplification, and digitalization of the signal. Analog signals is inputed at the entrance of function and digital signal samples typically, In-phase and Quadrature components are outputted.
Afterward digital samples are processed by the signal processing (a.k.a baseband processing) block. This part of the receiver is responsible for the acquisition and tracking of the signal. This process produces observables (or GNSS raw data measurements) like pseudoranges, Doppler frequency, carrier phase measurements ...
Finally, raw measurements are exploited by the data processing block of the receiver. Navigation and observation data are gathered by the navigation solution filter in order to estimate the user PNT solutions.

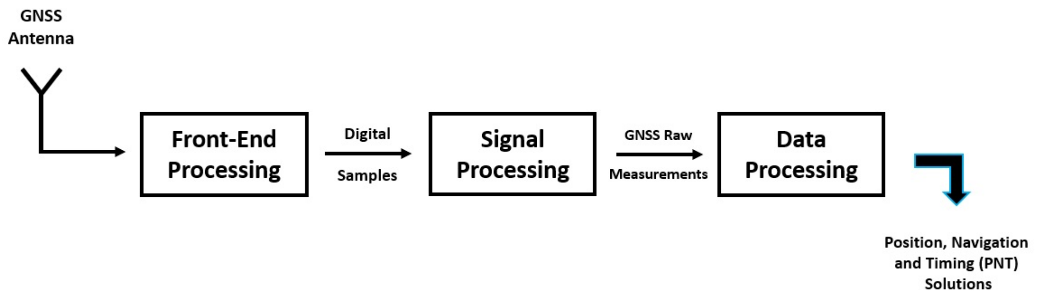


Figure A.4: Block diagram of a GNSS Receiver

A.2 GNSS Receiver Architecture

The core functionalities of a GNSS systems has been detailed in the section A.1. In order to better understand smartphones' embedded GNSS receiver, the detailed panorama of a conventional GNSS receiver architecture will be given in this section. As mentioned in A.1.3, a GNSS receiver is composed of three blocks: RF Front-end Block, the Signal Processing Block and finally the Data Processing block. In this section, the specific tasks dedicated to those blocks, such as *Acquisition* or *Tracking*, will be detailed.

A.2.1 RF Front-end Block

The RF Front-end goal is to provide digital samples of the received signal captured by the antenna. Its role also involves the selection of the useful part of the signal, trying to mitigate the impact of exterior interferences malicious or not. The classical architecture of a RF front-end block is provided in A.5. The main steps taken by the RF front-end are:

- Amplification of the incoming signal captured by the antenna.
- Down conversion of the incoming signal by mixing with local oscillators.
- Band-pass filter focus on the desired part of the signal.
- Analog to Digital Converter (ADC) device that convert the signal from analog to digital coupled with an ADC device.

- Output signal digital samples

The digital samples outputted by the RF front-end can be mathematically modeled as $r_{out}(t)$ by equation A.1. For simplification purposes, only one ray is being derived in the equation below. Real life signal can be interpreted as a linear combination of rays.

$$r_{out}(t) = h_{RF}(t) * [h_c(u,t) * S_T(t) + n(t)] \tag{A.1}$$

where:

- h_c is the impulse response of the propagation medium. This parameter can by modeled a delay varying in time $\delta(t - \tau(t))$
- h_{RF} is the impulse response of the RF front-end filter.
- n is an additional signal parameter taking into account additional noise. It is refereed to as White Gaussian Noise.

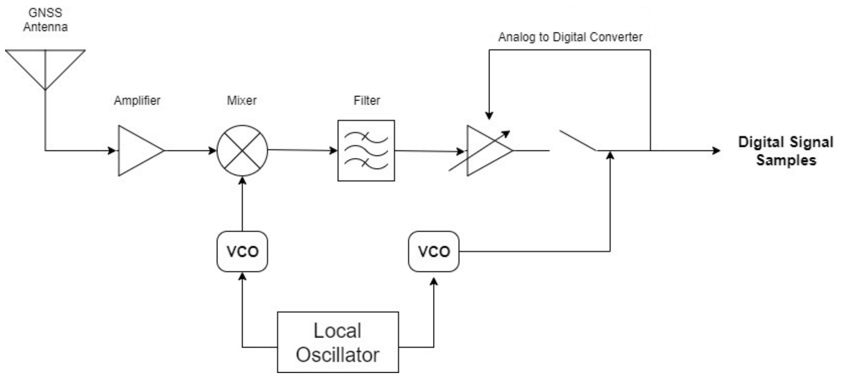


Figure A.5: Block diagram of the RF Front-end of a GNSS Receiver

A.2.2 Signal Processing Block

The signal processing block is in charge of estimating signals parameters that will be used to produce GNSS raw data measurements. The basic principle of baseband processing rely on correlation processes. Indeed, a local signal replica is generated at the receiver level and is then synchronized and compared to the captured incoming GNSS radio signal. Correlator outputs are then used in the three main stages of the signal processing block: Acquisition, Tracking and data demodulation.
The signal acquisition phase targets at identifying the incoming signal and to obtain a first rough estimate of time and frequency synchronization. Then, the tracking phase generates a local replica that aims at synchronizing with the incoming signal. Such processes allow for a refinement of previously roughly estimated signal parameters.

1.2.2.a Acquisition

The main goal of the receiver's acquisition process is to detect and identify the incoming signal while estimating the code delay and the Doppler frequency. Due to the unicity nature of the Pseudo-Random Noise (PRN) sequences, each received signal is processed independently. A common method is to create an acquisition search matrix. The code delay denoted $\hat{\tau}$ and the Doppler frequency denoted $\hat{f_D}$ are being roughly estimated throughout an acquisition search matrix [24]. This matrix is defined in size by the uncertainties of the code delay and of the Doppler frequency. An example of an acquisition matrix is shown by figure A.6. In this case, the receiver is receiving a navigation signal from the GPS PRN 2 satellite. The graph on the left shows that a correlation peak as been found for PRN 2. When similar test is performed for PRN 14, no correlation has been found (right figure on A.6). Thus, once a correlation has been established, the detection output process provide a rough estimate of Doppler frequency and code delay at the location of the correlation peak.

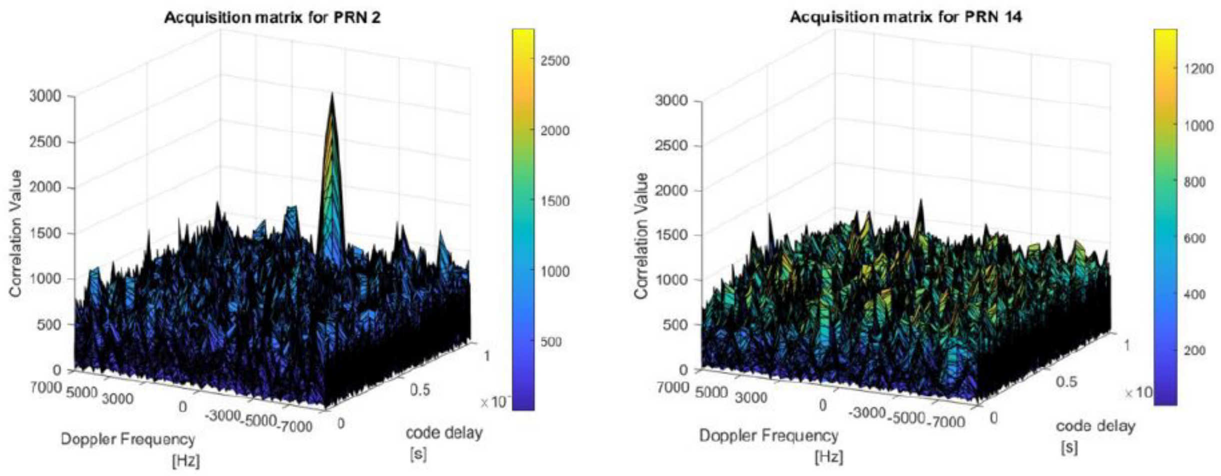


Figure A.6: Acquisition Matrix with a Signal containing Data from GPS PRN 2, unnormalized.

1.2.2.b Tracking

The tracking process is responsible for estimating accurately the Doppler frequency $\hat{f_D}$ and the code delay $\hat{\tau}$. The tracking process is achieved when the received signal is synchronized with a local replica in a closed loop process. The closed loop operation continuously tracks those parameters. They are referred as: Phase Lock Loop (PLL) and as Delay Lock Loop (DLL) . The DLL is in charge of tracking the code delay of the incoming signal $\hat{\tau}$ and the PLL aims at tracking the phase. The generic tracking process of GNSS receivers is represented by figure A.7.

The correlator output models are reminded in equation A.2 [24]. Those equations represents the In-Phase $I(k)$ and the Quadrature-Phase $Q(k)$ outputs. The phase is considered tracked when $\epsilon_\theta \approx 0$ and $\epsilon_f \approx 0$. On the other side, the code delay estimation is precise when $\epsilon_\tau \approx 0$. The measurements noise are $n_I(k)$ and $n_Q(k)$ are represented for both In-Phase and Quadrature-Phase outputs. The term $d(k)$ represents the data bits of the navigation message

$$\begin{aligned} I(k) &= \frac{A}{2}\, d(k)\, cos(\pi\epsilon_f T_I + \epsilon_\theta)\, R_c(\epsilon_\tau)\, \mathrm{sinc}(\pi\epsilon_f T_I) + n_I(k) \\ Q(k) &= \frac{A}{2}\, d(k)\, sin(\pi\epsilon_f T_I + \epsilon_\theta)\, R_c(\epsilon_\tau)\, \mathrm{sinc}(\pi\epsilon_f T_I) + n_Q(k) \end{aligned} \tag{A.2}$$

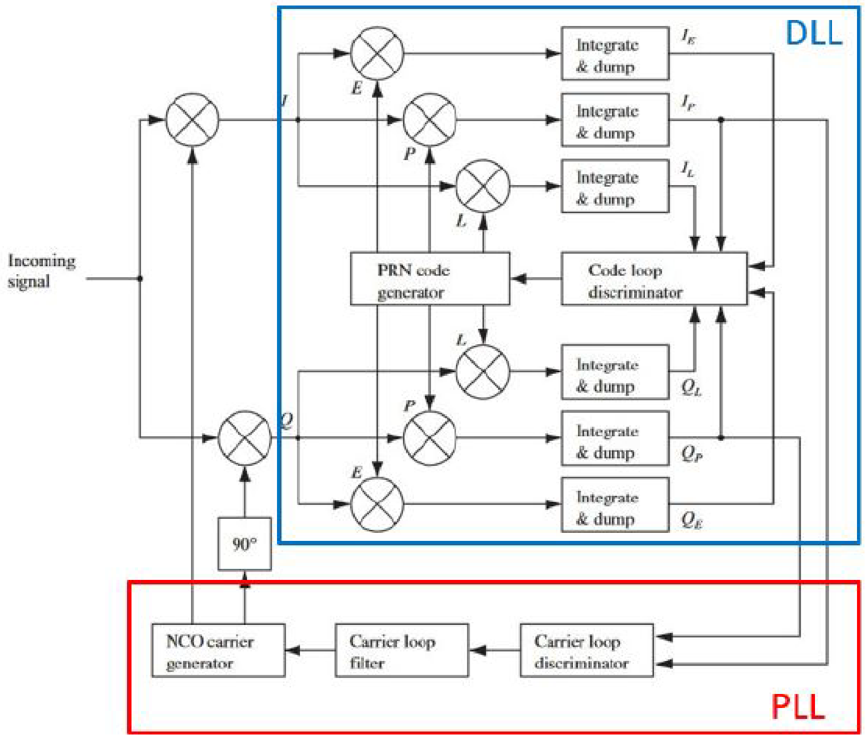


Figure A.7: Block diagram of a Tracking Process in a GNSS Receiver. Modified diagram from [17].

The tracking loops processes and goals are detailed below:

- **Code Lock Loop (DLL)**

  The goal of the code tracking loop is to synchronize the incoming signal with a locally generated PRN replica. The DLL block diagram can be seen on figure A.7 depicted by the blue square. Synchronization error between both signals is estimated by code delay discriminator. Three correlator outputs are used to assess code delay: the early, prompt and late. The early discriminator correlates with an early version of the local PRN code replica with a delay of $\frac{d}{2}$. The late discriminator correlates with a late version of the local PRN code replica with a delay of $-\frac{d}{2}$. Most of the time, the DLL uses the output of the prompt correlator to normalize the discriminator. The parameter d represents the Early-Late spacing, as shown on the horizontal axis on figure A.8.

- **Phase Lock Loop (PLL)** The goal of a PLL is to synchronize the phase of the incoming signal with a locally generated signal carrier. Similar to what was presented above, a phase discriminator is used to evaluate the phase error between the incoming signal carrier and the local replica. After the error has been found and filtered, a Digitally Controlled Oscillator (DCO) generates a new locally generated signal carrier. Modern PLL discriminators are developed in order to be insensitive to the data component of the signal. Discriminator algorithm and definitions are detailed in [24].

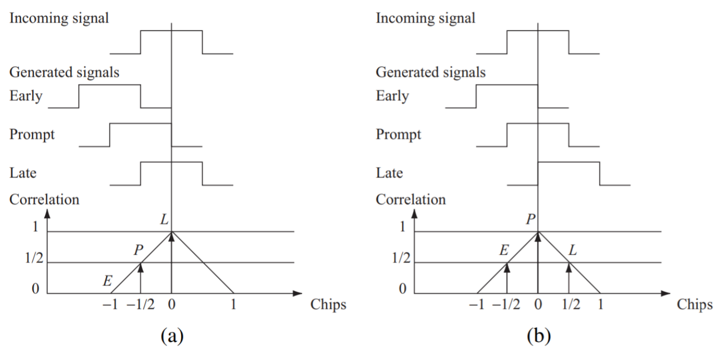


Figure A.8: Code Delay Tracking. (a) Signal Synchronization not Achieved. (b) Perfect Code Tracking. [17]

A.2.3 Data Processing Block

In parallel of the tracking processes displayed above, the data demodulation is put in place during last processing block of a generic GNSS receiver. This process will allow users to retrieve the navigation information contained within the received signal. The data processing block aims at extracting and processing navigation messages and observables.

1.2.3.a Navigation Messages

Data components of the incoming signal can be retrieved once the received signal is correctly tracked by the receiver. By looking back at equation A.2, the In-Phase and Quadrature signal models can be simplified as in equation A.3 since $\epsilon_\theta \approx 0$, $\epsilon_\tau \approx 0$ and $\epsilon_f \approx 0$ when the signal is correctly tracked.

$$\begin{aligned} I(k) &= \frac{A}{2}\, d(k) + n_I(k) \\ Q(k) &= n_Q(k) \end{aligned} \tag{A.3}$$

Bit synchronization techniques are used to estimate the navigation message embedded within the In-Phase correlator output. Such techniques aims at determining the start of the navigation message data bits in order to observe the prompt correlation value over 20 ms (for GPS L1C/A) and therefore, estimate the navigation message.

1.2.3.b Observables

The generation of observable data is the last retrieval process made by the receiver. Observables can also be referred as GNSS raw data measurements. Main observable parameters are the Doppler frequency, pseudoranges, phase measurements and C/N0. Doppler and phase measurements are directly derived from PLL or Frequency Lock Loop (FLL) loops. On the other hand, pseudoranges are derived from the previously estimated code delay. Pseudoranges are estimated by taking the difference between the Time of transmission and the time of reception multiply by the speed of light.

A.3 Chapter Conclusion

This chapter presented a global review of the GNSS structure. All existing satellite constellation have been presented though the display of their proposed services. A classical system structure used by most of the global systems has been exposed by showing the three main segments. Finally, a review was offered displaying a classical receiver architecture.

The following appendix chapter will detail the positioning and navigation method offered by GNSS. Measurements models and positioning principles will be laid out before characterizing positioning constraints linked with urban environments.

B GNSS Positioning

Contents

After defining GNSS system as a whole in previous chapter, this appendix will focus on presenting the method used for estimating user's position with a GNSS system. Firstly, the measurements model will be presented. The objective is to characterize the outputted observation measurements of a classic GNSS receiver. Then, the User Equivalent Range Error (UERE) total error budget will be displayed, explaining the sources of biases affecting pseudorange measurements. In a second time, basic positioning principles are exposed. This section will highlight the algorithmic processes used for computing a PVT solution using GNSS signals. Finally, a section will discuss the challenges associated with urban positioning. The characteristics of urban positioning are exposed.

B.1 Measurements Model

This section will present the measurement models for the GNSS observable outputted by the signal processing block of the receiver. Measurement models are then used to compute the residuals between the measures and the predicted phase and code pseudoranges. These residuals will contain the measurements errors including the position and clock biases. Thereafter, a set of equations containing all the residual measurements will form a linear system which solutions the estimator filters. This estimation process allows to decompose the error components and thus determining the receiver position. Therefore, it is of the utter importance that the measurements modeling accurately depicts the error causes for then obtaining an accurate position estimation.

B.1.1 GNSS Observable Models

A GNSS receiver computes observation measurements from the retrieved GNSS signals. These measurements can be characterized by their respective mathematical models. Equations B.1 and B.2 show the measurements model for the code measurement $\rho^i(k)$ for a satellite i at an epoch k and the carrier phase measurement model $\varphi^i(k)$.

$$\rho^i(k) = r^i(k) + c(dt_{rx} - dt^i_{sat}) + \epsilon^i_{Iono} + \epsilon^i_{Tropo} + \epsilon^i_{Multipath} + n^i_\rho + b^i_\rho \tag{B.1}$$

$$\varphi^i(k) = r^i(k) + c(dt_{rx} - dt^i_{sat}) - \epsilon^i_{Iono} + \epsilon^i_{Tropo} + \epsilon^i_{Multipath} + N^i\lambda + n^i_\varphi + b^i_\varphi \tag{B.2}$$

where:

- $r^i = \sqrt{((x - x^i)^2 + (y - y^i)^2 + (z - z^i)^2)}$ represents the geometric range with
  - (x, y, z) are the user position coordinates to be estimated.
  - (x^i, y^i, z^i) represents the given satellite position.
- $(dt_{rx} - dt^i_{sat})$ is the receiver-satellite clock offset multiplied by c the speed of light in vacuum.
- ϵ_{Iono} and ϵ_{Tropo} are the ionospheric and tropospheric delays.
- $\epsilon_{Multipath}$ is the multipath generated bias.
- n^i_ρ and n^i_φ are the noise components for the code and phase measurements respectively.
- b^i_ρ and b^i_φ are the RF code and phase biases.
- $N^i\lambda$ represents an integer number of Ambiguity present on the phase measurement.

The exposed errors present in the above measurement models are corrected before PVT computation. Indeed, those errors are either predicted via established models (e.g: Klobuchar for the ionospheric component and the UNB3 model for the tropospheric delay) or by applying a correction from the ephemeris data for the clock drift.
A classical error budget for the presented errors has been standardized taking in example the GPS L1 C/A signal model. The magnitude of these errors can be found in table B.1.

Error Sources:	Satellite Clock	Ephemeris Data	Ionosphere	Troposphere	Multipath	Thermal Noise
$1-\sigma$ **Error [m]**	1.1 to 2.1	1.1 to 2.1	4.0 to 7.0	0.2 to 1.4	0.2 to 1.4	0.1 to 0.5

Table B.1: GNSS Error Budget for GPS L1 C/A in Nominal Conditions.
Table derived from [19].

B.1.2 User Equivalent Range Error (UERE)

Pseudorange quality can be monitored by analyzing the UERE. The parameter is the user equivalent sum of all residual errors that affects the receiver GNSS measurements.

Equation B.3 presents the UERE quality parameter. The accuracy of a receiver measurements directly correlates to 3 factors: measurement accuracy, satellite elevation and the user local environment.

$$\sigma^2_{UERE} = \sigma^2_{Tropo} + \sigma^2_{Iono} + \sigma^2_{Multipath} + \sigma^2_{SatPos} + \sigma^2_{Noise} \tag{B.3}$$

B.2 Positioning Principles

Global Navigation Satellite System (GNSS) offers a positioning and timing service provided by a fleet of orbiting satellites. The main objective of this system is to estimate a user position anywhere around the globe. The promise is based on a trilateration principle illustrated by figure B.1. Satellites signal processing is used to determine the user position intersecting a set of four or more spheres characterized by the user-to-satellite distance. A GNSS receiver measures this distance by assessing the transmission time from the transmitter (satellite) to the receiver (user) multiplied by the speed of light. This principle relies on the time-synchronization between the two communicating devices.
However, in real scenarios, the time-synchronization condition is not met. It is of the utmost importance that both measurements need to be expressed in the same GNSS reference time. A clock bias exists between the receiver and the GNSS time whereas the clock shift between the satellite and the reference time is revised by ephemeris data. Thus, the corrected measured signal propagation delay is always biased by a clock error. In this context, GNSS distance measurements are referred as pseudo-range measurements. A positioning estimate results from the solution given to a set of equations bounding our system. 4 parameters needs to be estimated in order to retrieved the user position: $[\hat{x}, \hat{y}, \hat{z}, \Delta t]$. The first three parameters represent the estimated user's position in a Cartesian coordinate frame. Δt represents the receiver clock offset compared to the GNSS reference time.

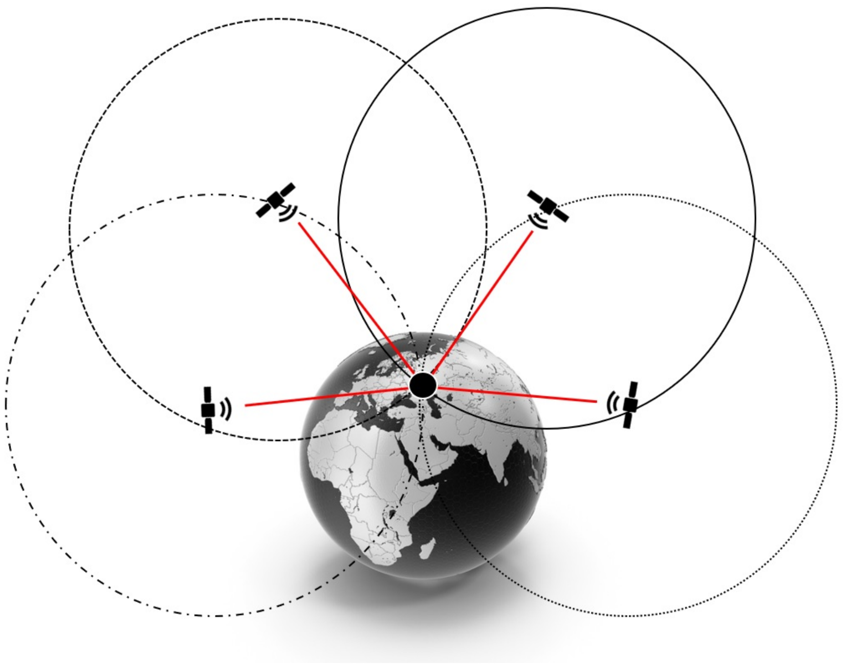


Figure B.1: Satellite Positioning Principle by Triangulation

A PVT computation aims at estimating the previously exposed parameters in order to estimate a position. Multiple estimation techniques exist for computing this solution. This section will present the two most used estimation algorithms used in GNSS: Least Square (LS) and Kalman Filter (KF). The former will be detailed in the following subsection for complementing its use within this research work.

B.2.1 Positioning Estimation Techniques (PVT)

GNSS positioning computation is supported by the estimation process of PVT. A GNSS receiver will output GNSS raw measurements from the signal processing block. Those measurements will then be used to create a system of equations that solution the user position. The two most widely used estimation technique in the GNSS domain are described here: Least Square (LS) and Kalman Filter (KF). A focus will be made on the WLS estimator, supporting previously exposed PVT results.

First we set a range length measurement P^i between the receiver antenna and the i-th satellite. Its model has been derived above in B.1.1. The aim is to obtain the estimate of the receiver position defined by its coordinates (x, y, z) and the clock offset Δt. In the meantime, the coordinates of the satellite have been transmitted through the navigation message and are expressed as (x^i, y^i, z^i). The pseudorange can then be written as in equation B.4.

$$P^i = \sqrt{(x^i - x)^2 + (y^i - y)^2 + (z^i - z)^2} + \Delta t \tag{B.4}$$

As described above, a set of 4 equations represents the minimal requirements for solving our 4 unknowns. As the presented equation in B.4 defines a non-linear system, the set of equations is solved by iteratively linearizing this system starting from an initial position of the receiver defined as (x_0, y_0, z_0). The linearization process is made by Taylor series approximation to the first term and is presented in equation B.5.

$$P^i(x,y,z,\tau) \approxeq P^i(x_0,y_0,z_0,\tau_0)+(x-x_0)\frac{\partial P}{\partial x}+(y-y_0)\frac{\partial P}{\partial y}+(z-z_0)\frac{\partial P}{\partial z}+(\tau-\tau_0)\frac{\partial P}{\partial \tau} \tag{B.5}$$

The residual measurements are obtained by taking the difference between the retrieved measurements and the predicted measurements computed with initial position values. Therefore, for a single satellite the residual measurements are expressed as in equation B.6.

$$\Delta P = \begin{pmatrix} \frac{\partial P}{\partial x} & \frac{\partial P}{\partial y} & \frac{\partial P}{\partial z} & \frac{\partial P}{\partial \tau} \end{pmatrix} \begin{pmatrix} \Delta x = (x - x_0) \\ \Delta y = (y - y_0) \\ \Delta z = (z - z_0) \\ \Delta \tau = (\tau - \tau_0) \end{pmatrix} + \epsilon \tag{B.6}$$

We can now generalized this expression for an n number of satellites corresponding to a system of n equations. The following equation B.7 is expressed in matrix form.

$$\underbrace{\begin{pmatrix} \Delta P^1 \\ \Delta P^2 \\ \Delta P^3 \\ \vdots \\ \Delta P^n \end{pmatrix}}_{\delta x} = \underbrace{\begin{pmatrix} \frac{\partial P^1}{\partial x} & \frac{\partial P^1}{\partial y} & \frac{\partial P^1}{\partial z} & \frac{\partial P^1}{\partial \tau} \\ \frac{\partial P^2}{\partial x} & \frac{\partial P^2}{\partial y} & \frac{\partial P^2}{\partial z} & \frac{\partial P^2}{\partial \tau} \\ \frac{\partial P^3}{\partial x} & \frac{\partial P^3}{\partial y} & \frac{\partial P^3}{\partial z} & \frac{\partial P^3}{\partial \tau} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial P^n}{\partial x} & \frac{\partial P^n}{\partial y} & \frac{\partial P^n}{\partial z} & \frac{\partial P^n}{\partial \tau} \end{pmatrix}}_{H} \underbrace{\begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \\ \Delta \tau \end{pmatrix}}_{x} + \underbrace{\begin{pmatrix} \epsilon^1 \\ \epsilon^2 \\ \epsilon^3 \\ \vdots \\ \epsilon^n \end{pmatrix}}_{e} \tag{B.7}$$

The linear relationship can then be rewritten in matrix form as $\delta x = Hx + e$ where δx represents the residual observation (observed minus predicted measurements), x are the equation unknowns that we are looking to solve for and the matrix e contains the remaining noise terms. Moreover, the "design matrix" H can be re-written by deriving the partial differentiation of the observation equation and is presented in equation B.8.

$$H = \begin{pmatrix} \frac{x_0 - x^1}{\rho} & \frac{y_0 - y^1}{\rho} & \frac{y_0 - y^1}{\rho} & c \\ \frac{x_0 - x^2}{\rho} & \frac{y_0 - y^2}{\rho} & \frac{y_0 - y^2}{\rho} & c \\ \frac{x_0 - x^3}{\rho} & \frac{y_0 - y^3}{\rho} & \frac{y_0 - y^3}{\rho} & c \\ \vdots & \vdots & \vdots & \vdots \\ \frac{x_0 - x^n}{\rho} & \frac{y_0 - y^n}{\rho} & \frac{y_0 - y^n}{\rho} & c \end{pmatrix} \tag{B.8}$$

2.2.1.a Weighted Least Square (WLS)

The LS estimation method is based on a MLE process. The goal of this method is to minimize the residual positioning error from our previously presented system of equations. A variant method called WLS is described below for estimating user position from GNSS measurements.

Let set $\hat{x}$ defined as the solution to the system $\delta x = H\hat{x} + e$. The least square estimator iteratively compute a value of x that minimizes the sum of squares of the estimated residuals.

$$f(x) = \arg\min_{\hat{x}} \sum_{i=1}^{n} e_i^2 = e^T e = (\delta x - Hx)^T (\delta x - Hx) \tag{B.9}$$

Therefore, the weighted least square solution results from equation B.10.

$$\hat{x} = (H^T\ W\ H)^{-1}\ H^T\ W\ D \tag{B.10}$$

where:

- $\hat{x}$ represents the set of solutions for each epoch corresponding to the positioning coordinates in a Cartesian reference frame
- H is the design matrix
- W is the weighting matrix
- D representing the residual measurements

The particularity of the WLS algorithm resides in the formation of a weighted matrix denoted W. The role of this matrix is to qualitatively assess the measurements used by the estimation algorithm. In the case where measurements errors are uncorrelated, the weighting matrix is built as the inverse of the covariance matrix. Those variances can be obtained either via apriori knowledge or by a derivation of stochastic models. Traditionally, the properties of GNSS signals have been provided by geometrical and signal quality parameters (satellite elevation and signal carrier to noise ratio). A weight will then be assigned to a specific measurement. The final W matrix will be a diagonal matrix expressed as in equation B.11 with w_i representing the weights associated to each received signals.

$$D = \begin{bmatrix} w_1 & & \\ & \ddots & \\ & & w_n \end{bmatrix} \tag{B.11}$$

2.2.1.b Kalman Filter

Kalman Filter (KF) is a Bayesian recursive estimation method integrated in a dynamic model of equations. The KF has been firstly described in [116]. Since then, the method is one of the most popular data fusion algorithm in the GNSS domain. Its use has been derived for integrating GNSS measurements with IMU sensor data. KF technique is considered a recursive estimator of an initial state in a dynamic system where apriori knowledge comes from the measurements models and uses previous state estimates to refine the current estimates.

2.2.1.c Assessing Positioning Performance

The expected accuracy resulting from GNSS positioning algorithm is dependent of two factors: Measurement quality (assessed by UERE) and by the satellite geometry. For a given user, satellite geometry is qualified by the DOP value. A low DOP value indicates a well distributed grid of satellites in the sky over the receiver, enhancing positioning accuracy performance. On the contrary, a high DOP value shows a bad repartition of satellites. Recently, with the increase of GNSS satellites orbiting the Earth, a DOP is considered good when its value does not exceed 2. The DOP parameter translates how the error in the measurements propagates to errors in the position and time domain. Multiple DOP values exist and are listed below.

- The Geometric DOP quantifies the 4D error standard deviation
- The (3D) Position DOP assesses the 3D position standard deviation of the error
- The Horizontal DOP assesses the horizontal positioning error
- The Vertical DOP quantifies the vertical error standard deviation
- The Time DOP shows the receiver time error standard deviation

All the presented DOP parameters can be retrieved from the weighted least square estimation technique, their computation processes are presented in equation B.12

$$\mathrm{DOP} = [H^T H]^{-1} = \begin{bmatrix} \mathrm{DOP}_E^2 & & & \\ & \mathrm{DOP}_N^2 & & \\ & & \mathrm{DOP}_U^2 & \\ & & & \mathrm{DOP}_t^2 \end{bmatrix}$$

$$\begin{aligned} \mathrm{GDOP} =& \sqrt{\mathrm{DOP}_E^2 + \mathrm{DOP}_N^2 + \mathrm{DOP}_U^2 + \mathrm{DOP}_t^2} \\ \mathrm{PDOP} =& \sqrt{\mathrm{DOP}_E^2 + \mathrm{DOP}_N^2 + \mathrm{DOP}_U^2} \\ \mathrm{HDOP} =& \sqrt{\mathrm{DOP}_E^2 + \mathrm{DOP}_N^2} \\ \mathrm{VDOP} =& \sqrt{\mathrm{DOP}_U^2} \\ \mathrm{TDOP} =& \sqrt{\mathrm{DOP}_t^2} \end{aligned} \tag{B.12}$$

B.3 Urban Positioning

In the world of GNSS, positioning in an urban area represents a challenge. In this constraint environments made of narrow streets and tall buildings, signals quality often deteriorates. Recently, the development of new GNSS constellations improved globally urban positioning. During this overview, the first part will describe in details the challenges associated to GNSS urban positioning. Then, existing techniques will be explored in order to mitigate the previously described errors. Finally, the advantages brought by the arrival of new GNSS constellation will be studied.

B.3.1 Characteristics of Urban Positioning

GNSS receivers have been designed for operating in open-sky conditions where satellite signals can be retrieved directly by the receiver in Line of Sight (LOS) conditions. However due to environmental constraints in urban conditions, GNSS signals are heavily impacted by interferences causing signals delays and errors. The local surrounding environment of the receiver defines its reception condition. Urban environment are characterized by various type of reception conditions. Sub-urban areas are assimilated to difficult situations where signals can be blocked by small constructions, dense canopy and houses. Deep urban areas, usually referred as urban canyon, present high blocks of buildings forming narrow streets and usually navigating through dense traffic. For a GNSS receiver, positioning and navigation capabilities are significantly reduced in those environments since GNSS satellites signals are mostly blocked by the above mentioned blockage.
Furthermore, giant buildings are also acting as reflectors on which signals can bounce multiple times. Those physical phenomenon are referred as signal refraction and diffraction. Those impacted signals replicas are called multipath and highly disrupt the tracking processes of a GNSS receiver. This section will review the impact of the two most disruptive scenarios that can handicap urban positioning: NLOS and multipath events.

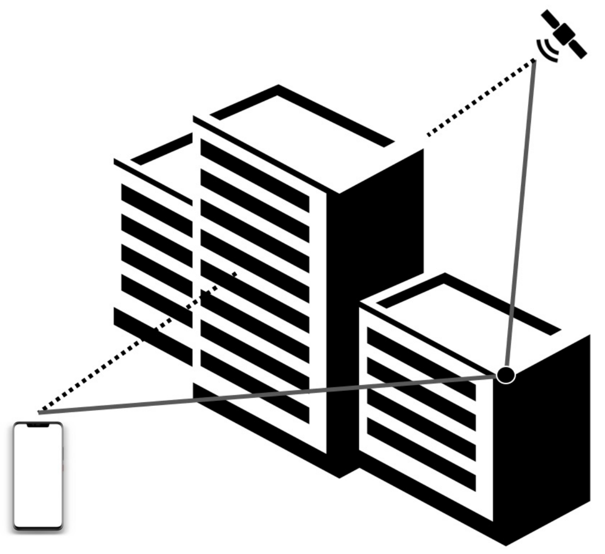


Figure B.2: Principle of a NLOS event caused by Building Blockage

2.3.1.a Impact of Urban Buildings Blockage

Historically, the main issue associated with urban positioning and navigation comes from the insufficient number of satellites in sight for the receiver. As reminded in the section above, a minimum of 4 satellites are needed for performing the user PVT. Satellite availability restriction can be explained by the masking occurrences induced by buildings in the receiver vicinity. Satellite signal blockage can become disruptive for the receiver acquisition process, especially when the number of tracked signals drops. This phenomenon requires the receiver to re-acquired lost signals and thus penalizing its positioning performance. In some cases, this non visible signal can be diffracted by adjacent buildings and still be acquired by the receiver. This principle is referred as NLOS. In this case, the received signals path is modified and induces an error on the computed pseudorange measurements. This principle is depicted by figure B.2.
In order to overcome signal availability in LOS conditions, satellites constellation needs to be optimized. Increasing the number of GNSS satellites will provide a chance to the user to obtain measurements from high elevation satellites and thus reducing the occurrences of NLOS signal acquisition. Moreover, since the development of new GNSS constellations such as BDS, GLONASS, Galileo and QZSS, the satellites visibility significantly increased in urban environment. A study realized by [117] shows that by using a multi-constellation receiver, satellite geometry GDOP was improved and thus increased the position accuracy of the tested system. In this context, the QZSS constellation has been developed so that high elevation Geosynchroneous Orbits (GSO) satellites should always be visible in dense Japanese urban canyons.

2.3.1.b Impact of Multipath Disruption

Multipath occurrences are defined as an accumulated signal sum between direct line of sight and a single or multiple reflected counterpart signals. These additional signal paths are created by reflected or diffracted replicas of the GNSS signal. The multipaths are originated from multiple objects in the local environment of the receiver such as: buildings, canopy, vehicles, ... Figure B.3 is a schematic representation of a multipath

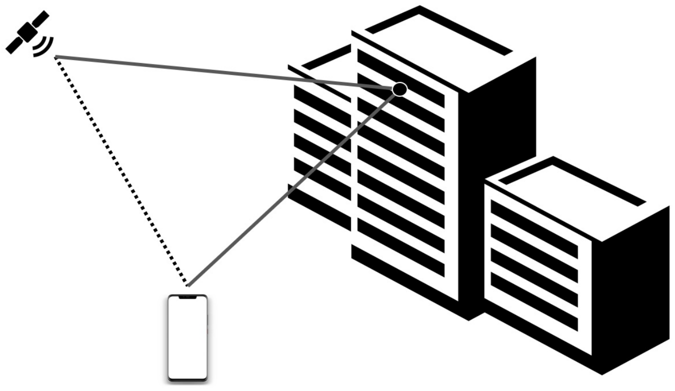


Figure B.3: Principle of Multipath Event

scenario.
The unpredictable nature of multipath events makes their prediction and detection difficult for the receivers. Indeed, the added bias paths are exact copies of the original signal that cannot be differentiated by a GNSS receiver due to their coherent behaviors. The main impact of multipath interference is seen in the correlation function in the signal acquisition process. Figure B.4 illustrates the effect of multipath events on the correlation function. The blue plot represents the correlation output of an unbiased signal whereas the red plot shows its reflected counterpart correlation function. As described by the red and orange plots, the effects of multipath interferences can either be a constructive or disruptive effect on the code and carrier tracking processes.

B.3.2 Detection & Mitigation Techniques

After reviewing the main challenges of urban positioning, the consequences of those phenomenons will be discussed and mitigation techniques will be described. Due to the complexity of detecting those phenomenons in urban areas, multiple mitigations techniques are combined to mitigate the resulting errors.

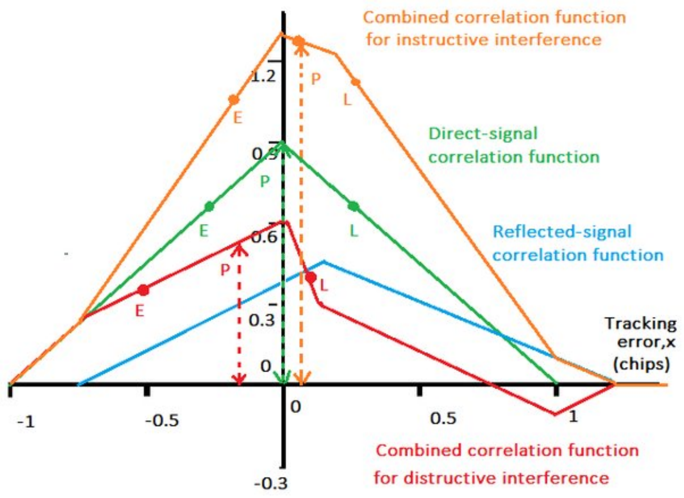


Figure B.4: Effect of Multipath Interferences on the Correlation Function for both Constructive and Disruptive Effects. Pictured extracted from [18]

2.3.2.a Detecting & Mitigating Multipath

Mitigation techniques for multipath rely mainly on the separation of the line of sight signal and its reflexion. In the literature, multiple works highlight the main mitigation techniques used for multipath events [118]. These methods have been classified according to the specification characteristics of Multipath error.

- **Changes in signal polarization**
  GNSS signals are characterized as Right-Hand Circular Polarized (RHCP). When a signal is reflected, its polarization attribute can change and become Left-Hand Circular Polarization (LHCP). However if the signal is reflected twice, the polarization of the signal could remain unchanged. The detection of polarization can be detected at the antenna level [119] [120]. However this would require a specific antenna design that, most of the time, is not available for low-cost receivers. If a different polarization could be detected in an urban environment that could designate that multipath signals have been potentially received.

- **Distorts the correlation function**
  At the signal processing level, the correlation function role is to synchronizes the received signal Pseudo-Random Noise (PRN) code with a local replica. By receiving a multipath infected signal, the correlation function will be distorted and thus the tracking process (code and phase) of the receiver will be degraded resulting in major accuracy errors. Figure B.4 shows the distortions that can be observed when multipath phenomenon is present. Multiple techniques exist in order to detect and mitigate multipath at the correlation function level [121]. The implementation of multiple early and late correlators could detect the distortion of the correlation function and prevent tracking errors.

- **Affects the power to noise ratio (C/N0)**
  The power-to-noise ratio (C/N0) represents the ratio between the received power of a specific signal compared to noise. Multipath induces a change of this C/N0 value (either an increase or a decrease). The variation of C/N0 impacts the performance of the receiver's tracking process. The most common method for multipath detection would be to compare the C/N0 measure from two different frequencies for the expected satellite elevation angle [122].

- **Impacts pseudoranges measurements**
  Due to reflection, multipath will impact the pseudoranges measurements by adding travel time between the satellite and the receiver for the reflected signal. Pseudoranges measurements of one or more satellites could vary from epoch to epoch because of the presence of that phenomenon. A consistency check between different pseudoranges of one particular satellite could lead to the detection of the error. Then by applying a weighting process and by excluding faulty signals, multipath errors could be reduced. In [123], a Random Sample Consensus (RANSAC) consistency based algorithm has been developed for mitigating multipath events.

2.3.2.b Detecting & Mitigating Non-Line-of-Sight

Detecting and mitigating NLOS is a complex task. A simple detection process would be to simply set a C/N0 threshold in order to reduce the received reflected signals; this method is however very limited since the received C/N0 could be affected by numerous

factors. Most of known techniques require external hardware devices. One of those techniques consists on using an external camera in order to map the environment around the receiver/user. By doing so, satellites with low elevation angle, supposedly not seen by the receiver, are excluded.
However, alternative methods inspired by Receiver Autonomous Integrity Monitoring (RAIM) checking algorithm have been developed to mitigate NLOS. A method of consistency checking of C/N0 weighting have been developed in [118]. By revising consecutive measurements, NLOS signals can be excluded. The downside of this method is that with a multi-constellation receiver faulty exclusion could be frequent [124].
In his paper [125] states that NLOS detection/mitigation can be either achieved by a statistical approach. By knowing the initial position of the user (within a street) the probability of being able to see a specific satellite and not its reflexion is computed. From there, the algorithm decides or not to exclude faulty signals.

B.3.3 Alternative Methods

As characterized above, urban environment conditions impact the resulting user positioning capabilities. GNSS signals quality are highly dependent on the receiver position in the street or of its local environment. This section will expose and discuss existing alternative methods and techniques used for enhancing receiver' positioning performance. The aim is to accurately determine the receiver position within the street in order to better detect multipath and NLOS signals before correcting their effects.

2.3.3.a 3D City Modeling

3D cities model are now being integrated with GNSS observable in order to mitigate multipath and NLOS events [54] [126]. Those models allow the user to estimate satellites visibility and level of signal degradation due to multipath. Models include geometry information, location and dimensions of the physical objects (building) around the user [127]. The receiver approximative position allows the algorithm to locate its position within the 3D city model. This process will give the user a priori characteristics of buildings surrounding his receiver. It will allow the user to estimate if the received signal from a satellite is actually visible or if it should be mask by the adjacent structure. Additional informations such as edifices surface orientation, can be added to the model. If the surface orientation is known, then given the satellite trajectory, a multipath detection techniques could be executed. However, the use of 3D city modeling method is limited by the model accuracy and location.

2.3.3.b Shadow Matching

The shadow matching technique answers a common pedestrian positioning problem; *On which side of the street am I located ?*. Most of the received signals are defined as along-track (along a street) in a urban canyon, and the cross-track signal are most of the time masked by adjacent natural or man-made obstacles. The arrival of new GNSS constellations would only increase satellite availability on the along-track segment.
The shadow-matching positioning method reverses 3D city models positioning as exposed in [54] [55]. The 3D city model computes the expected visible signals within the receiver's street. The shadow-matching tool will determine whether or not the signal is in line of sight of the user. Then, the user can localize his position on the street. This algorithm

is a simple comparison between received signals and signals that should be seen by the receiver. Although, this method requires an accurate 3D city model where the receiver stands. Nowadays, only a very few cities have the needed decimeter precision city model.

C

Android Flags Detection Mechanisms Analysis

Contents

This chapter is dedicated to provide additional resources complementing the analysis presented in section 3.3.1.b. This appendix is a collection of results derived from the presented analysis for different smartphone models and brands. This analysis archive is organized as a library for each directed investigation. The aim is to record and classify smartphone's measurement behaviors for quality monitoring checking and regression analysis that will be conducted in future works. All the work and analysis presented in this annex have been derived from the data collection campaign exposed in 3.2.

First and foremost, the complementary signal analysis of the remaining tested smartphone is shown. This study highlight the retrieved signal strength in function of time completed by the flags activation frequency over time. Then, a correlation analysis is performed for characterizing flags events on different smartphones and embedded chipsets. Finally, the continuation of the CMC study will be presented for the main tested smartphone on various constellation signals.

C.1 Smartphone Signal Analysis & Flags Detection Contingency

This section complements the preliminary signal analysis detailed in 3.3.2.a. The aim for this study was to correlate flag detection events with typical signal parameters. Firstly, the flag detection events are compared with received signal strength. Then, flag activities are compared against signal frequencies and satellite constellation in order to retro-engineer the algorithmic mechanisms of Android flag detection system. The analysis in 3.3.2.a presented the Huawei Mate 20X results for illustrating purpose. In this appendix, the processed results from the other smartphones are exposed.

C.1.1 Signal Analysis

This subsection complements the analysis made on received signal power against flag detection mechanisms. Here, the analysis results are exposed for all tested smartphones during the data collection campaign. Figure C.3 through figure C.6 display the study results for a specific smartphone model.

All figures are organized as follow: The top plot represents the C/N0 fast fluctuations observed over time. Three parameters are shown on this plot: minimum, median and maximum C/N0 values has been computed as a function of each individual received signal (considering all frequencies and all constellations simultaneously) for every epoch.
The bottom plot shows the Android flag activation frequency as a function of received signals for both multipath and cycle slip events. The activation process is characterized in percentage of flag activation per received signals. As an example: Let a total of four signals being received by a smartphone receivers, if a multipath flag has been activated on three of those signals then our frequency parameter would show a 75% activation for the Android multipath mechanism.

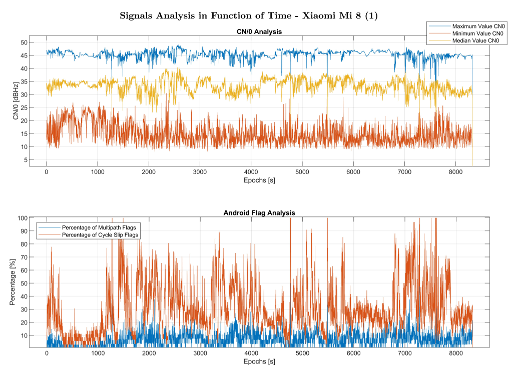


Figure C.1: Signal Analysis of the Xiaomi Mi8 (1). a) Upper Figure: C/N0 Analysis. b) Lower Figure: Percentage of Android Flags Activation

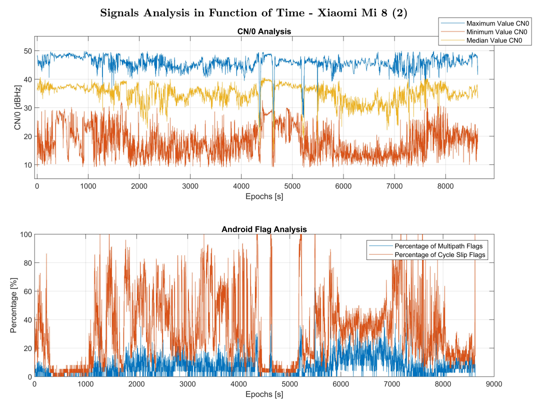


Figure C.2: Signal Analysis of the Xiaomi Mi8 (2). a) Upper Figure: C/N0 Analysis. b) Lower Figure: Percentage of Android Flags Activation

Figures C.1 and C.2 represent the signal analysis for both Xiaomi Mi 8. Signal analysis characteristics have similar patterns to the one exposed in section 3.3.2.a. The bottom plot depicts a high frequency of cycle slip flag detection mechanisms. The observed variation in activation frequency can be easily explained by the local constraining environment. Furthermore, despite a low activation frequency, multipath flags detection seems to have similar characteristics as the one of cycle slip. Both activation events seem to be correlated with each other.

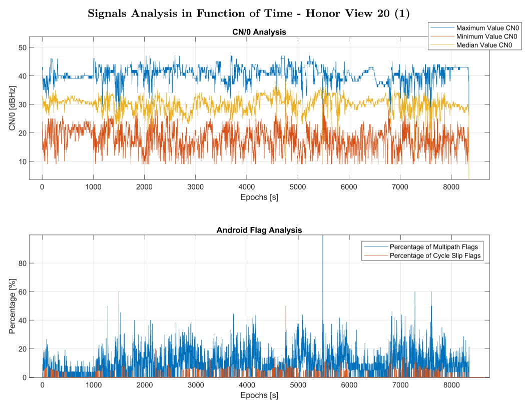


Figure C.3: Signal Analysis of the Honor View 20 (1). a) Upper Figure: C/N0 Analysis. b) Lower Figure: Percentage of Android Flags Activation

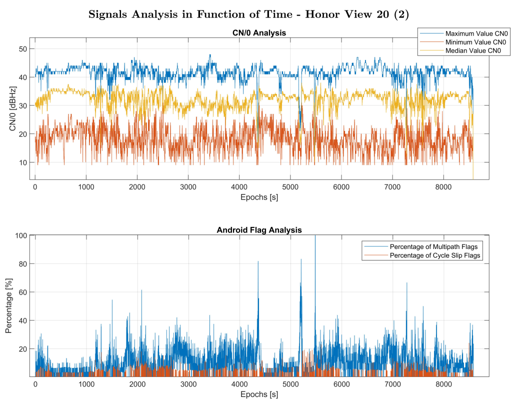


Figure C.4: Signal Analysis of the Honor View 20 (2). a) Upper Figure: C/N0 Analysis. b) Lower Figure: Percentage of Android Flags Activation

Figures C.3 and C.4 depict the signal analysis performed on both Honor View 20. As a reminder, Honor devices are equipped with the same chipset as the Huawei Mate 20X (HiSilicon Kirin 980). However, the cycle slip mechanism exhibits a drastically different behavior. Cycle slip activations are nearly inexistent, while the multipath detection frequency remains on the same order of magnitude to previously exposed analysis. This demonstrates that Android flags mechanisms algorithms are implemented at a lower level in the Android API and can drastically change in function of the smartphone model and is independent of the operation of the chipset itself. This observation needs to be confirmed by future analysis work.

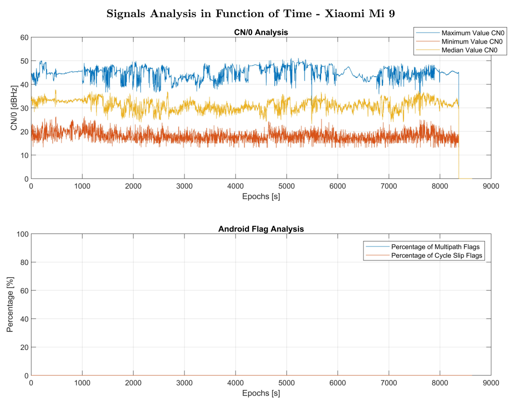


Figure C.5: Signal Analysis of the Xiaomi Mi9. a) Upper Figure: C/N0 Analysis. b) Lower Figure: Percentage of Android Flags Activation

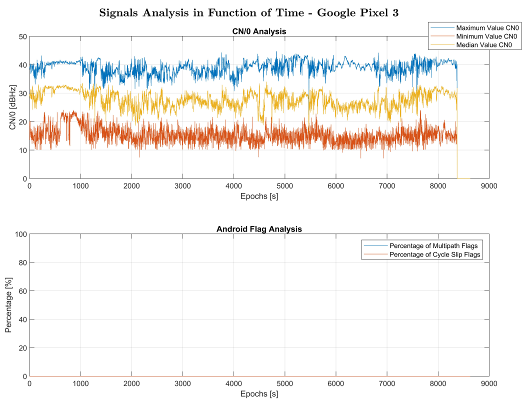


Figure C.6: Signal Analysis of the Google Pixel 3. a) Upper Figure: C/N0 Analysis. b) Lower Figure: Percentage of Android Flags Activation

Finally, figures C.5 and C.6 displays the signal analysis on the Xiaomi Mi 9 and the Google Pixel 3. The similarity between those two devices is that they do not allow the collection and processing of phase measurements. The bottom plot demonstrates the inactivity of flag detection mechanisms on both devices. Therefore, it is assumed that flag detection mechanism are directly dependent on phase measurements extracted from the smartphone embedded GNSS receiver.

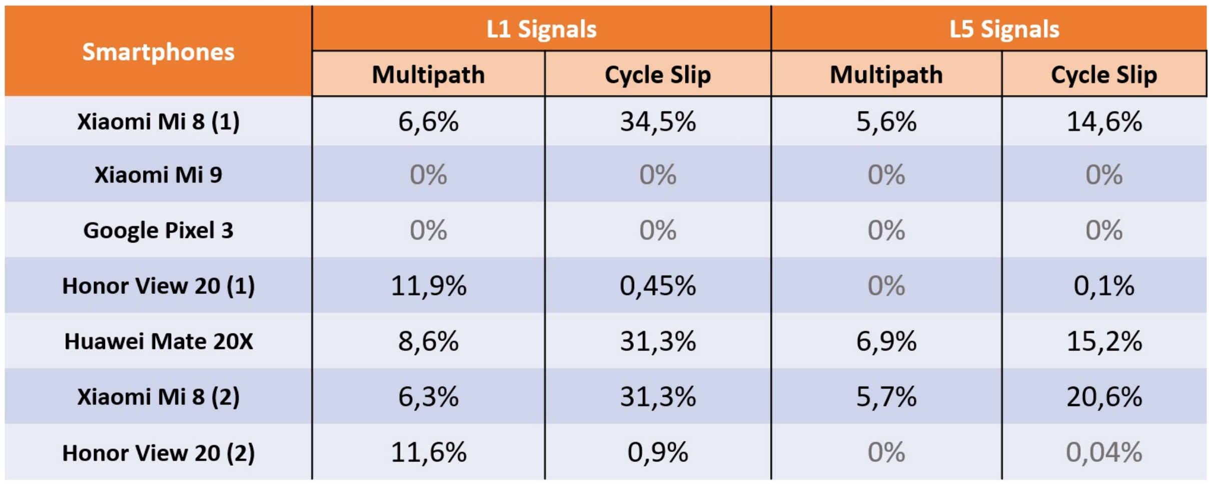
C.1.2 Flags Mechanism Detection Frequency

The following study presents the flags detection frequency as a function of signal frequency and satellite constellation. This analysis aims at defining is some particular signals is being deliberately targeted by the system. The presented tables show the activation frequency [in % of the time] of both multipath and cycle slip flagging events for all tested smartphones.

Figure C.7 shows the cycle slip and multipath activation frequency in function of two signal frequencies: L1/E1 and L5/E5a. The first observation that transpire from this analysis is that the results presented above are being confirmed. Moreover this table highlights that cycle slip activation are reduced by 2 on the L5 signals and that multipath events are non existent on L5 signals for both Honor devices.
Those characteristics can be associated with the results presented in figure C.8. This table shows the cycle slip and multipath activation frequency in function of satellite constellation: GPS and Galileo. However, no major correlation characteristics can be extracted from this table. No major differences exist between the two constellation systems in terms of flags activities.

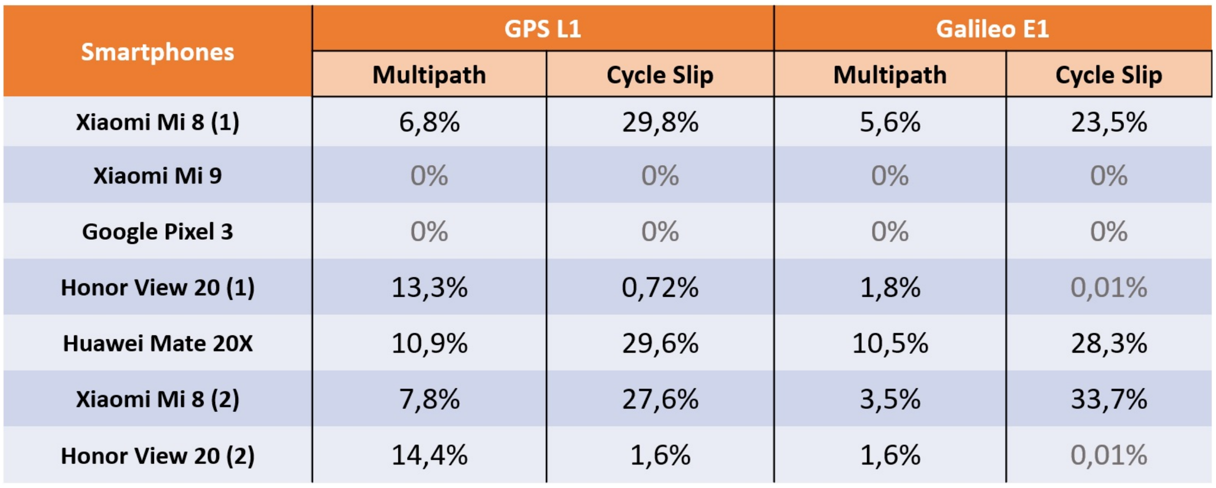
In conclusion, multipath and cycle slip flag detection mechanisms behaviors have been analyzed. The following list highlights the main characteristics of Android flag detection mechanisms.

- Both parameters appear to be co-dependent
- The implementation of those measurements are made in a lower level of the Android API. This detection technique seems to be an option that could be activated by smartphone manufacturer.



Smartphones	L1 Signals		L5 Signals	
	Multipath	Cycle Slip	Multipath	Cycle Slip
Xiaomi Mi 8 (1)	6,6%	34,5%	5,6%	14,6%
Xiaomi Mi 9	0%	0%	0%	0%
Google Pixel 3	0%	0%	0%	0%
Honor View 20 (1)	11,9%	0,45%	0%	0,1%
Huawei Mate 20X	8,6%	31,3%	6,9%	15,2%
Xiaomi Mi 8 (2)	6,3%	31,3%	5,7%	20,6%
Honor View 20 (2)	11,6%	0,9%	0%	0,04%

Figure C.7: Multipath and Cycle Slip Flag Detection Events in function of Signal Frequency for all Tested Smartphones



Smartphones	GPS L1		Galileo E1	
	Multipath	Cycle Slip	Multipath	Cycle Slip
Xiaomi Mi 8 (1)	6,8%	29,8%	5,6%	23,5%
Xiaomi Mi 9	0%	0%	0%	0%
Google Pixel 3	0%	0%	0%	0%
Honor View 20 (1)	13,3%	0,72%	1,8%	0,01%
Huawei Mate 20X	10,9%	29,6%	10,5%	28,3%
Xiaomi Mi 8 (2)	7,8%	27,6%	3,5%	33,7%
Honor View 20 (2)	14,4%	1,6%	1,6%	0,01%

Figure C.8: Multipath and Cycle Slip Flag Detection Events in function of Signal Constellation for all Tested Smartphones

- The similarities observed during the analysis makes us think that the implementation of flags detection algorithms has been implemented by Google services.
- Flag detection algorithms do not only account for the C/N0 values but are rather integrated in a complex method.

C.2 Correlating Flag Detection Mechanisms

Following our preliminary analysis, multiple basic GNSS measurements have been tested through a series of correlation events. We made the hypothesis that multipath and cycle flag detection algorithms were not solely linearly correlated to C/N0. To validate this hypothesis, flags distributions in function of C/N0 and elevation were analyzed. Section 3.3.2.a presented the cycle slip flags in function of C/N0. This study exposes the activation of both multipath and cycle slip flags in function of satellite elevation.

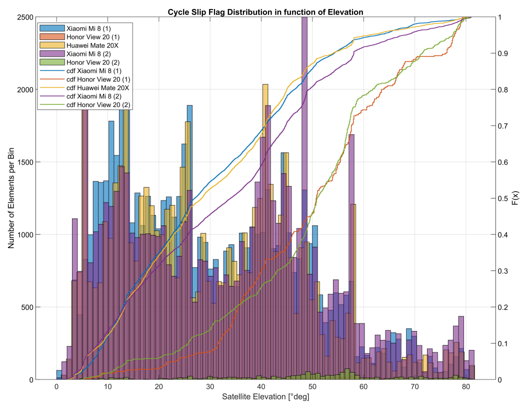


Figure C.9: Cycle Slip Flag Detection Events in function of Satellite Elevation

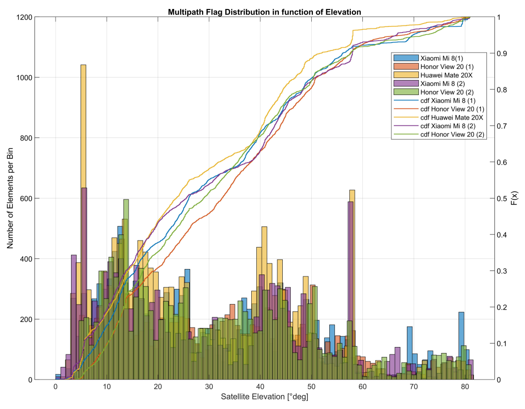


Figure C.10: Multipath Flag Detection Events in function of Satellite Elevation

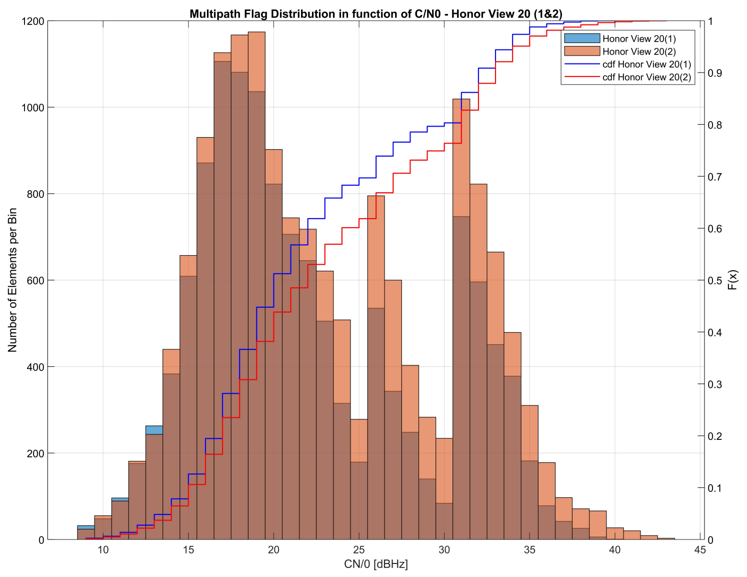


Figure C.11: Multipath Flag Events Distribution in function of C/N0 for the Honor view 20 (1&2)

Figures C.9 and C.10 respectively show the distribution of cycle slip and multipath flagged events in function of satellite elevation. It is expected that cycle slip and multipath occurrences are less detected when a satellite is at high elevation ($> 65°$) since direct line of sight between the receiver and satellite can be anticipated. These figures confirms our expectation with a decreasing detection rate when satellites reach an elevation angle above 60°. Moreover, no definite correlation can be established between flag detection systems and satellite elevation.

Figure C.11 depicts the multipath flag events distribution in function of C/N0 for both Honor View 20. It appears that both devices show similar properties signature in the detection of multipath events. This observation confirms the hypothesis made above, that flag detection algorithms are dependent of the smartphone brand

In conclusion, this study validates the hypothesis that the generation of multipath and cycle slip flag detection mechanisms is not linearly correlated with either signal strength C/N0 or satellite elevation parameters.

C.3 Code-Minus-Carrier Analysis

This analysis complements the study made in section 3.3.2.b. This study aims at assessing the global efficiency of flag detection mechanisms. Figures C.12, C.14 and C.16 exhibit the CMC computation for both Xiaomi Mi 8 devices with the two strongest satellite signals present during the data collection campaign: GPS PRN 27 and Galileo PRN 12. On the above mentioned figures, the top plot represents the CMC evolution in time, while applying the sliding mean fixing method on segments where the satellite was physically visible by the receiver. This implies that cycle slip events should be visible on those plots. The red dots show where a cycle slip flag activation has been reported by Android. Thereafter, the bottom plot illustrates the computed CMC values still corrected

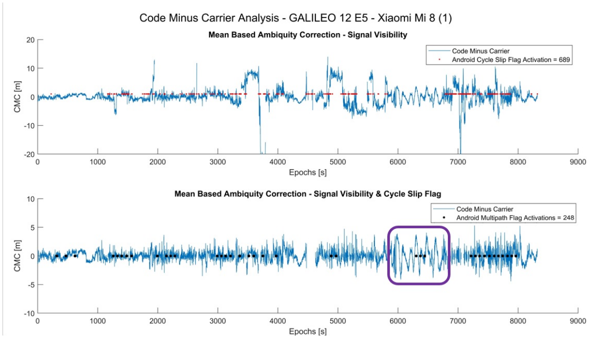


Figure C.12: Code-Minus-Carrier Correlation Analysis for the Xiaomi Mi8 (1) - Galileo PRN 12 E5a

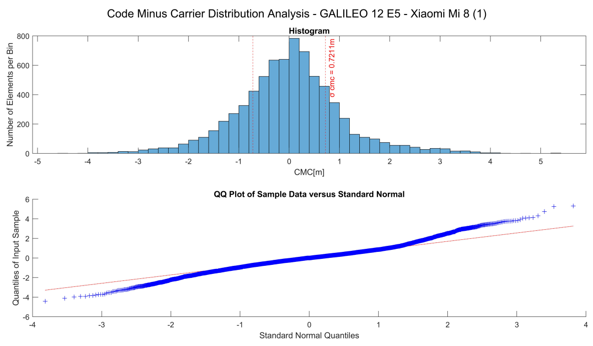


Figure C.13: Code-Minus-Carrier Distribution Analysis for the Xiaomi Mi8 (1) - Galileo PRN 12 E5a

by the sliding mean fixing method per segments. However this time, segments were said to be continuous if the satellite was physically visible *and* if the Android flag algorithm did not detect any cycle slip. In this case, visible cycle slips remaining on the figure would mean that Android failed to correctly detect cycle slips. Theoretically at this stage, cycle slips should have been removed, leaving multipath and noise characteristic behaviors on the CMC plot. Black dots indicates Android multipath flag detection events.
Figure C.13, C.15 and C.17 shows the CMC distribution errors histograms characterizing the DLL jitter.

The overall flag activation seems to be over proportionate and too strict to detect real occurrences. However, the few cycle slips that happened during our data collection seem to have been successfully detected by Android. Firstly, multipath flag algorithm have been activated only under 250 occurrences for all tested smartphones. This number is supposedly underestimating the reality of our deep urban environment data collection. Moreover, a significant multipath event is visible on both top and bottom graph. A typical multipath oscillation can be seen (depicted by purple boxes on all figures) and not being detected at any moment by the Android algorithm. This implies that the multipath indicator is not triggered by a simple threshold on CMC.

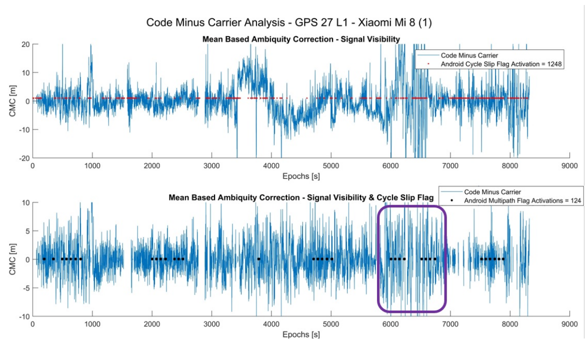


Figure C.14: Code-Minus-Carrier Correlation Analysis for the Xiaomi Mi8 (1) - GPS PRN 27 L1

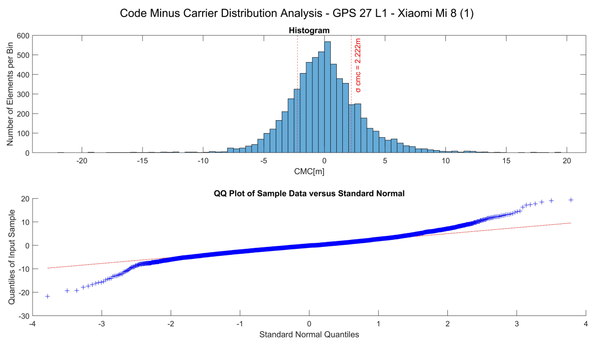


Figure C.15: Code-Minus-Carrier Distribution Analysis for the Xiaomi Mi8 (1) - GPS PRN 27 L1

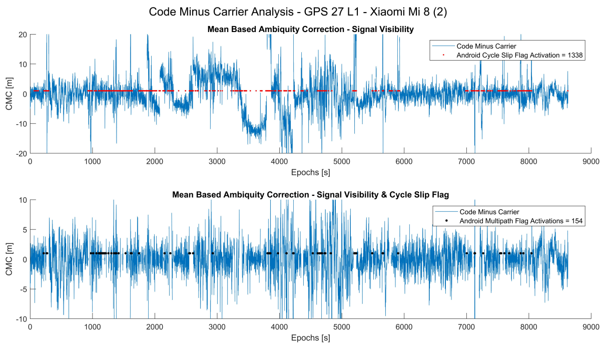


Figure C.16: Code-Minus-Carrier Correlation Analysis for the Xiaomi Mi8 (2) - GPS PRN 27 L1

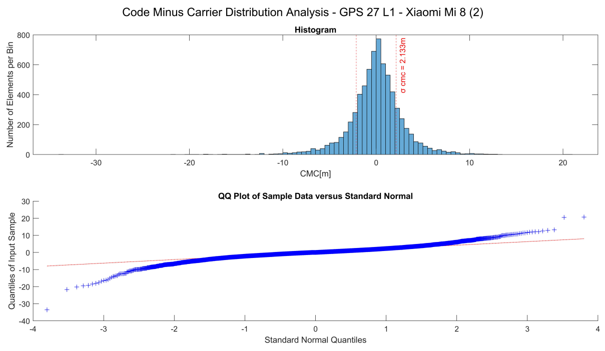


Figure C.17: Code-Minus-Carrier Distribution Analysis for the Xiaomi Mi8 (2) - GPS PRN 27 L1

In conclusion, the computation processes of Android flags mechanisms seem to not be exclusively based on a naive interpolation of C/N0 or satellite elevation parameters. Moreover, the similarities observed between smartphone brands make us believe that detection algorithms might be computed at a low-level Android layer. Multipath flags tend to be inconsistent whereas cycle slip flags were proven to be coherent despite their high false alarm activation frequency. However, Android multipath and cycle slip indicators might not be used as reference parameters to qualitatively assess smartphone positioning performance.

D

SmartCoop Algorithm: A live Matlab® Script

The following document is a live script from the Matlab® environment. Live scripts document are been conceived to be interactive documents. This scripts carefully explain the choices and the implementation procedure used for creating the *SmartCoop* smartphone-based collaborative technique. This document represents a well commented version of our functioning algorithm. This is included in this research report for its pedagogical character.

Constrained Nonlinear Optimization - Problem-Based

Collaborative Positioning (VERHEYDE 2021) - $SmartCoop^{TM}$ *Algorithm*

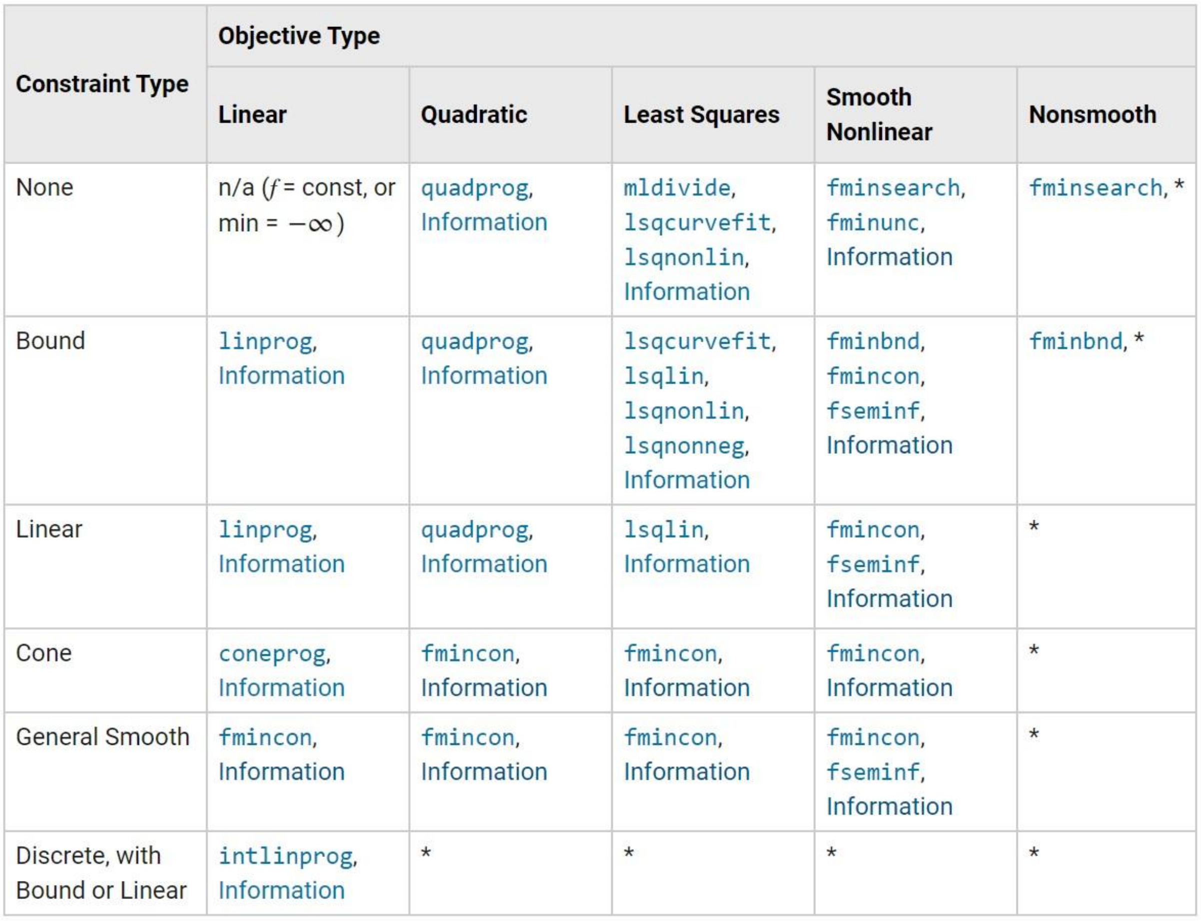
I - INTRODUCTION

Matlab offers multiple ways to solve optimization problems (ranging from Least-square, to minimization algorithms).

2 types of approaches exists in Matlab:

- **Solver-based** (see solver-based nonlinear problem)
- **Problem-based** (see problem-based nonlinear problem)

Characteristics of the two approaches are listed in the table below or following this link.



Constraint Type	**Objective Type**				
	Linear	**Quadratic**	**Least Squares**	**Smooth Nonlinear**	**Nonsmooth**
None	n/a (f = const, or min = $-\infty$)	`quadprog`, Information	`mldivide`, `lsqcurvefit`, `lsqnonlin`, Information	`fminsearch`, `fminunc`, Information	`fminsearch`, *
Bound	`linprog`, Information	`quadprog`, Information	`lsqcurvefit`, `lsqlin`, `lsqnonlin`, `lsqnonneg`, Information	`fminbnd`, `fmincon`, `fseminf`, Information	`fminbnd`, *
Linear	`linprog`, Information	`quadprog`, Information	`lsqlin`, Information	`fmincon`, `fseminf`, Information	*
Cone	`coneprog`, Information	`fmincon`, Information	`fmincon`, Information	`fmincon`, Information	*
General Smooth	`fmincon`, Information	`fmincon`, Information	`fmincon`, Information	`fmincon`, `fseminf`, Information	*
Discrete, with Bound or Linear	`intlinprog`, Information	*	*	*	*

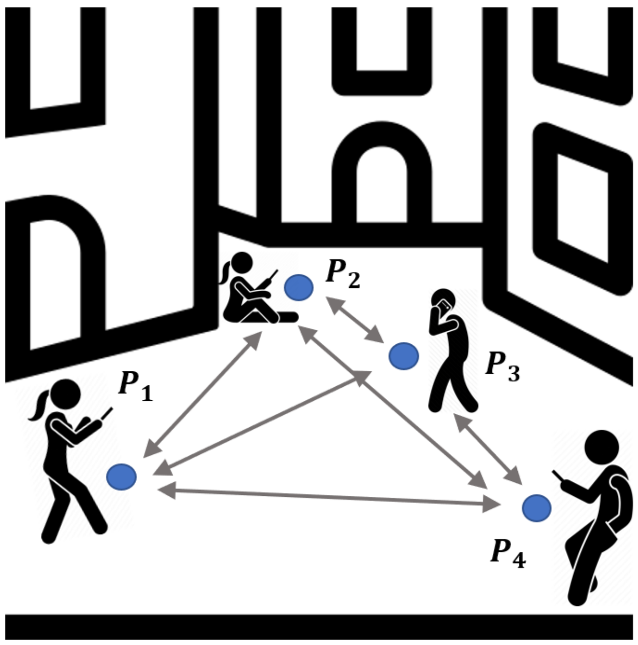
After choosing your approach, you need to select a solver function that depends on the definition of you problem. Solver function could be used for multiple problems such as: Quadratic functions, multiobjectives optimization problems and more.

Follow this table, in order to choose the Matlab solver that best fit the definition of your problem.

Link to MATLAB Optimization Videos: Master Class with Loren Shure & Example of Problem-based Nonlinear Programming.

I - DEFINITION OF OUR COLLABORATIVE PROBLEM

Create a Collaborative Algorithm for Smartphone Positioning in Urban Environement.



Given a set of estimated position $\hat{P}_n$ (rough estimate made using Android Raw Data Measurements), with n being the number of users in the network, and the distance between smartphones (defined by $d_{12} = \sqrt{(P_{2_x} - P_{1_x})^2 + (P_{2_y} - P_{1_y})^2 + (P_{2_z} - P_{1_z})^2}$).

The distance between smartphone users has been **Previously Estimated** by performing double differences of each GNSS raw data measurements retrieved for each users. Those distance will be refered as **Inter-Phone Ranges (IPR).**

We now set our optimization. The goal is to minimize the 3D position error between the collaborative *newly estimated* positions and the initial calculated positions (Position Fix).

Our minimization process will be constrained by nonlinear equations satisfying the equality between IPR and positions distances (ex: d_{12}) of the *newly estimated* collaborative positions.

Objective Function to Minimize is:

$$\widehat{P}_n = \arg\min_{P} \sum_{u=1}^{n} \frac{(\widehat{p}_{u,x} - \widetilde{p}_{u,x} - \mu_{u,x})^2}{2\sigma_{u,x}^2} + \frac{(\widehat{p}_{u,y} - \widetilde{p}_{u,y} - \mu_{u,y})^2}{2\sigma_{u,y}^2} + \frac{(\widehat{p}_{u,z} - \widetilde{p}_{u,z} - \mu_{u,z})^2}{2\sigma_{u,z}^2}$$

while satisfying: $\begin{cases} (\widehat{p}_{v,x} - \widehat{p}_{u,x}) = \mathrm{IPR}^{uv}_{3D,x} \\ (\widehat{p}_{v,y} - \widehat{p}_{u,y}) = \mathrm{IPR}^{uv}_{3D,y} \\ (\widehat{p}_{v,z} - \widehat{p}_{u,z}) = \mathrm{IPR}^{uv}_{3D,z} \end{cases}$

Number of Constraints = $\frac{n(n-1)}{2}$

with σ_x^2 and μ_x being the variance and mean of position error in 3D.

Inputs & Outputs

System Inputs:

- *Smartphones' Positions (coming from the first estimation process obtained by a PVT algorithm using Android Raw Data Measurements)*
- *Inter-Phone Ranges (IPR)*
- *Number of smartphone users*

System Outputs:

- *Collaborative Smartphones' Positions*

Optimization Algorithm (Simulated Data)

Hypothesis:

- *Data are simulated following smartphone STD and mean from real data*
- *Distances between smartphones are assumed have been previously estimated via **IPR** Algorithm*

Generating Simulated Data

```
global Range
global RangeVector
global ECEF

NumPhones = 4;
Scenario = 'OpenSky';
Geometry = 'Square';
RangeType = 'SPAN';
RangeVector = 'ON';
Length = 60; %Length of the scenario in Epochs(sec)
```

```
[LLA, ECEF, Range, NEU] = DATA_GENERATOR(NumPhones, Scenario ...
                    , Geometry, RangeType, Length, RangeVector);

[ErrorStats, Global] = PositionStats(ECEF);
```

```
Mean = [0 0 0 ; 0 0 0 ; 0 0 0 ; 0 0 0];
Var = [(0.5^2) (0.5^2) (1^2) ; (2.5^2) (2.5^2) (3.8^2) ...
    ; (2.5^2) (2.5^2) (3.8^2) ; (2.5^2) (2.5^2) (3.8^2)];
```

Problem-based Approach - NonLinear Optimization

Create optimization problem

```
prob = optimproblem
```

Create optimization variable (in this case: 1 matrix of 3xN representing N users' position in 3D)

```
x = optimvar('x');
```

To find the minimum value of a nonlinear objective function using the problem-based approach, first write the objective function as a file or anonymous function. The objective function for this example is

$$f(x) = \sum \frac{\left(\hat{P}_i^x - \tilde{P}_i^x - \mu_x\right)^2}{2\sigma_x^2} + \frac{\left(\hat{P}_i^y - \tilde{P}_i^y - \mu_y\right)^2}{2\sigma_y^2} + \frac{\left(\hat{P}_i^z - \tilde{P}_i^z - \mu_z\right)^2}{2\sigma_z^2}$$

```
type objfunx
```

START a loop for computing collaborative solutions Epoch by Epoch (MLE method)

```
for epoch = 1:Length
```

Create the objective function as an expression in the optimization variables.

```
fun = @(x) ( (x(1,1) - x0(1,1) - Mean(1,1))^2 ) / (2 * Var(1,1)) + ...  % X_Phone1 +
           ( (x(1,2) - x0(1,2) - Mean(1,2))^2 ) / (2 * Var(1,2)) + ...  % Y_Phone1 +
           ( (x(1,3) - x0(1,3) - Mean(1,3))^2 ) / (2 * Var(1,3)) + ...  % Z_Phone1 +
           ( (x(2,1) - x0(2,1) - Mean(2,1))^2 ) / (2 * Var(2,1)) + ...  % X_Phone2 +
           ( (x(2,2) - x0(2,2) - Mean(2,2))^2 ) / (2 * Var(2,2)) + ...  % Y_Phone2 +
           ( (x(2,3) - x0(2,3) - Mean(2,3))^2 ) / (2 * Var(2,3)) + ...  % Z_Phone2 +
           ( (x(3,1) - x0(3,1) - Mean(3,1))^2 ) / (2 * Var(3,1)) + ...  % X_Phone3 +
           ( (x(3,2) - x0(3,2) - Mean(3,2))^2 ) / (2 * Var(3,2)) + ...  % Y_Phone3 +
           ( (x(3,3) - x0(3,3) - Mean(3,3))^2 ) / (2 * Var(3,3)) + ...  % Z_Phone3 +
           ( (x(4,1) - x0(4,1) - Mean(4,1))^2 ) / (2 * Var(4,1)) + ...  % X_Phone4 +
           ( (x(4,2) - x0(4,2) - Mean(4,2))^2 ) / (2 * Var(4,2)) + ...  % Y_Phone4 +
           ( (x(4,3) - x0(4,3) - Mean(4,3))^2 ) / (2 * Var(4,3));       % Z_Phone4
```

The unused `@fmincon` parameters (`lb, ub, A, b, Aeq` and `beq)` are initialized to empty vectors

Defining our Non-linear equalitites constraints

```
nonlcon = @nlcon_NEW;
```

while satisfying: $\begin{cases} (\widehat{p}_{v,x} - \widehat{p}_{u,x}) &= \mathrm{IPR}^{uv}_{3D,x} \\ (\widehat{p}_{v,y} - \widehat{p}_{u,y}) &= \mathrm{IPR}^{uv}_{3D,y} \\ (\widehat{p}_{v,z} - \widehat{p}_{u,z}) &= \mathrm{IPR}^{uv}_{3D,z} \end{cases}$

The function @nlcon_NEW is shown at the bottom of this code.

This function can be tested with our initial guesses

Ouputs: **`c`** and **`ceq`** represents respectively the inequality and equality constraints

```
[c , ceq] = nlcon_NEW(x0);
```

Solving our Constrained Non-Linear Optimization Problem using *fmincon*

We set optimization options for @fmincon. It has been determined that the most adapted algorithm for smartphone collaborative

positioning is: "sqp". Further information can be found here, for choosing the @fmincon algorithm

```
options = optimoptions(@fmincon,'Algorithm','sqp','Display','iter-detailed');
```

Solving for x (Collaborative Positions in a N x 3 matrix) with N being the total number of users within the network.

```
[x,fval,exitflag,output] = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options);
```

The process is then iteratively conducted, each solutions are saved in an output matrix for each computed epochs.

END of the Epoch loop

```
end
```

Analysing scripts and functions can be implemented showing the performance improvement of our algorithm.

Creating a function for Setting Up Constraints

In a separate file, the constraints function is defined as follow:

```
function [c, ceq] = nlcon_NEW(x)

global epoch
global Range

    for a = 1:3
    ceq(:,q) =  ( x(1,a) - x(2,a) ) - Range.R12(epoch,a);
    q = q+1;
    end
    for a = 1:3
    ceq(:,q) =  ( x(1,a) - x(3,a) ) - Range.R13(epoch,a);
    q = q+1;
    end
    for a = 1:3
    ceq(:,q) =  ( x(1,a) - x(4,a) ) - Range.R14(epoch,a);
    q = q+1;
    end
    for a = 1:3
    ceq(:,q) =  ( x(2,a) - x(3,a) ) - Range.R23(epoch,a);
    q = q+1;
    end
    for a = 1:3
    ceq(:,q) =  ( x(2,a) - x(4,a) ) - Range.R24(epoch,a);
    q = q+1;
    end
    for a = 1:3
    ceq(:,q) =  ( x(3,a) - x(4,a) ) - Range.R34(epoch,a);
    q = q+1;
    end

c = []; % No inequality constraints
end
```

with Range vector containing the previously computed **IPR** 3D vector ranges and x representing the tentative of collaborative positions.

The newly estimated collaborative positions (**x** from `fmincon`) are outputted iteratively epoch by epoch at same Android raw data measurements rate (1Hz)

Acronyms

IPR$_{3D}$ Inter-Phone 3D Range Vector. XII, 84–89, 92, 93, 97–103, 105–108, 111, 113, 115, 123, 125

3DMA 3D Mapping Aided. 36

ADC Analog to Digital Converter. 139

AGNSS Assisted GNSS. XI, 24, 32, 33, 67

AOA Angle-of-Arrival. 17

APD Absolute Position Differencing. 79, 80

API Application Programming Interface. XI, 18, 22–26, 33, 44, 45, 86, 125, 159, 160

BDS BeiDou Navigation Satellite System. 16, 134, 135, 152

CDOP Collaborative Dilution of Precision. 72

CMC Code-Minus-Carrier. 47–49, 53, 54, 157, 162, 163

CNES Centre National d'Études Spatiales. 104, 105, 111

COTS Commercial Off-the-Shelf. 2, 3, 11, 37, 39, 53, 55, 57

CP Collaborative Positioning. 66–75, 78–80, 86, 93, 94, 97, 101, 105, 108, 117, 121

CPU Central Processing Unit. 13, 19

CST Centre Spatial de Toulouse. 104

CSV Comma-Separated Values. 43

DCO Digitally Controlled Oscillator. 142

DD Double Difference. 80, 81, 89

DGNSS Differential GNSS. 34, 67, 78, 80, 83, 85, 91, 92, 95, 117

DLL Delay Lock Loop. 50, 53–55, 71, 141, 142, 163

DOP Dilution of Precision. 72, 119, 150

DSP Digital Signal Processor. 13, 20

DSRC Dedicated Short-Range Communication. 70, 73

EGNOS European Geostationary Navigation Overlay System. 135

EKF Extended Kalman Filter. 95, 96

ESA European Space Agency. 135

ESSP European Satellite Services Provider. 135

EUSPA European Union Agency for the Space Program. 12, 133

FAA Federal Aviation Administration. 135

FDMA Frequency Division Multiple Access. 16, 134

FDOA Frequency Difference of Arrival. 17

FLL Frequency Lock Loop. 50, 143

FLP Fused Location Provider. 33, 36, 57, 88–92, 124

FOC Full Operational Capability. 135

GAGAN GPS Aided Geo Augmented Navigation System. 136

GCS Galileo Control Centers. 138

GDOP Geometric Dilution of Precision. 89, 92, 152

GINS Global Indian Navigation System. 135

GLONASS Globalnaya Navigatsionnaya Sputnikovaya Sistema. 16, 134, 152

GLP GPS Location Provider. 33

GMS Galileo Mission Segment. 138

GNSS Global Navigation Satellite System. I, III, VII, IX, XI, XIII, 1–4, 9–13, 16, 17, 19–27, 29, 31, 32, 34–40, 42–44, 50, 51, 53, 55–61, 65–71, 73–75, 77–81, 83–86, 89, 90, 92–102, 104–106, 108, 111–113, 119, 121, 123, 124, 131–135, 138–155, 160

GPS Global Positioning System. XI, XIII, 10, 13, 14, 16, 22, 25, 44, 67, 132–136, 141, 146, 160, 162

GPU Graphics Processing Unit. 13, 19

GSO Geosynchroneous Orbits. 152

HAS High Accuracy Service. 134

IEEE Institute of Electrical and Electronics Engineers. 70

IGS International GNSS Service. 34

IMU Inertial Measurement Unit. XI, 16–18, 26, 90, 94, 150

IOT Internet of Things. 11

IPR Inter-Phone Ranging. 2, 4, 74, 77, 83–86, 88, 89, 92, 94, 99, 101, 113

IPRd Inter-Phone Range Distance. 84, 88–92, 113

IRNSS Indian Regional Navigational Satellite System. 135

ISP Image Processing Unit. 20

JAXA Japan Aerospace Exploration Agency. 135

JMA Japan Meteorological Agency. 136

KF Kalman Filter. 51, 94, 95, 148, 150

LBS Location Based Services. 1, 10, 11

LE Low Energy Consumption. 17

LHCP Left-Hand Circular Polarization. 154

LIDAR Light Detection and Ranging System. 78, 96, 109

LOS Line of Sight. 151, 152

LS Least Square. 94, 96, 97, 108, 148, 149

MCU Microcontroller Unit. 20

MEMS Micro-Electro Mechanical Systems. 16–18

MEO Medium Earth Orbit. 131

ML Maximum Likelyhood. 96

MLE Maximum Likelyhood Estimation. 74, 75, 149

MSAS Multi-functional Satellite Augmentation System. 136

NAVIC Navigation Indian Constellation. 16, 135

NCO Numerically-Controller Oscillators. 50

NEU North, East, Up. XIII, 106, 115, 116

NLOS Non-Line of Sight. XIII, 1, 35–37, 57, 79, 105, 111, 151, 152, 154, 155

NLP Network Location Provider. 33

OS Open Service. 17, 22, 23, 38, 55, 59, 134

P2P Peer-to-Peer. 68, 69

PCB Printed Circuit Board. 19, 20

PF Particle Filter. XII, 95, 96

PLL Phase Lock Loop. 50, 55, 56, 71, 141–143

PNT Position, Navigation and Timing. 10, 11, 96, 134, 138, 139

PPP Precise Point Positioning. 11, 34, 35, 60, 78

PPS Precise Positioning Service. 133

PRN Pseudo-Random Noise. XIII, 22, 44, 47, 69, 140–142, 162

PRS Public Regulated Service. 134

PVT Position, Velocity and Timing. 24, 71, 89, 90, 97, 112, 145, 146, 148, 152

QZSS Quasi-Zenith Satellite System. 16, 135, 152

RANSAC Random Sample Consensus. 154

RDSS Radio Determination Satellite Service. 134, 135

RF Radio Frequency. IX, XIII, 131, 138–140

RHCP Right-Hand Circular Polarization. 20, 21, 154

RINEX Receiver Independent Exchange Format. 25, 26

RNSS Radio Navigation Satellite Service. 134, 135

RPR Raw Pseudorange Ranging. 80

RSS Received Signal Strength. 17

RSSI Received Signal Strength Indicator. 18

RTK Real-Time Kinematic. 11, 17, 34, 35, 60, 78, 80, 101

RTT Round-Trip Time. 18, 33

SAR Search and Rescue. 134

SBAS Satellite-based Augmentation System. 135, 136

SD Single Difference Ranging. 80

SoC System-on-a-Chip. 13, 19, 20, 31, 38

SONAR Sound Navigation and Ranging. 78, 96

SoOP Signals of Opportunity. 1, 12, 17

SPS Standard Positioning Service. 133

SQP Sequential Quadratic Programming. 104

SUPL Secure User Plane Location. 32

TDOA Time Difference of Arrival. 17

TOA Time-of-Arrival. 17, 18, 78

ToW Time-of-Week. 43

TTFF Time To First Fix. 32, 67

UERE User Equivalent Range Error. 145–147, 150

UKF Unscented Kalman Filter. 95

URLLC Ultra-Reliable Low-Latency Communication. 70

UWB Ultra-Wide Band. 70, 78

V2V Vehicle-to-Vehicle. 72

VPS Visual Positioning System. 35

VRW Velocity Random Walk. 17

WAAS Wide Area Augmentation System. 135

WLAN Wireless Local Area Networking. 18

WLS Weighted Least Square. 51, 58, 84–86, 88, 96, 148–150

Bibliography

[1] Robert Odolinski and Peter J.G. Teunissen. Innovation: Low-cost single-frequency positioning approach. *GPS World*, 2017.

[2] European Space Agency (ESA). GNSS Users Technology Report. Technical report, GSA ESA, 2018.

[3] Fabio Dovis, Paolo Mulassano, and Fabrizio Dominici. *Handbook of Position Location: Theory, Practice, and Advances, Second Edition*, chapter Overview of Global Navigation Satellite Systems, pages 923–974. John Wiley, 09 2011.

[4] Himanshu Sharma and Mohamed Bochkati. RTK with Smartphone and Need of Sensor Fusion. In *Third Annual GNSS Raw Measurements Task Force Workshop - Universität der Bundeswehr München*, 2019.

[5] Chih-Ming Su, Chih-Hsien Wu, Kin-Lu Wong, Shih-Huang Yeh, and Chia-Lun Tang. User's hand effects on EMC internal GSM/DCS mobile phone antenna. In *2006 IEEE Antennas and Propagation Society International Symposium*, pages 2097–2100, 2006.

[6] Todd E. Humphreys, Matthew Murrian, Frank van Diggelen, Sergei Podshivalov, and Kenneth M. Pesyna. On the feasibility of cm-accurate positioning via a smartphone's antenna and GNSS chip. In *2016 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pages 232–242, 2016.

[7] GNSS.Raw.Measurements.Task.Force. Using gnss raw measurements on android device. *White Paper - European GNSS Agency (GSA)*, 2017.

[8] Laetitia Kessas. GitHub Source Code: SmartLogger Android Application. *URL: https://github.com/laetks/SmartLoggerGnss-Java*, 2021.

[9] Google/Android. Android GNSS Raw Data Measurements - android.location.GnssClock. Website, last visit 15/08/2021, Updated: 2021-04-21. https://developer.android.com/reference/android/location/GnssClock.

[10] Google/Android. Android GNSS Raw Data Measurements - android.location.GnssAntennaInfo. Website, last visit 15/08/2021, Updated: 2021-04-21. https://developer.android.com/reference/android/location/GnssAntennaInfo.

[11] Google/Android. Android GNSS Raw Data Measurements - android.location.GnssMeasurement. Website, last visit 15/08/2021, Updated: 2021-04-21. https://developer.android.com/reference/android/location/GnssMeasurement.

[12] Frank Van Diggelen. *A-GPS: Assisted GPS, GNSS, and SBAS (Edition 1st)*. Artech House, 03 2009.

[13] Jan Van Sickle. GPS and GNSS for Geospatial Professionals: Differencing. *College of Earth and Mineral Sciences - Department of Geography - Penn State*, 2020.

[14] Alex Minetto, Alessandro Gurrieri, and Fabio Dovis. A Cognitive Particle Filter for Collaborative DGNSS Positioning. *IEEE Access*, Volume 8:pp: 194765–194779, 2020.

[15] U.S. Coast Guard Navigation Center. Slant chart (gps satellite locations) - smc-2021-2113. *Website: https://www.navcen.uscg.gov/pdf/gps/current.pdf*, 2021.

[16] European Space Agency (ESA). Galileo's global ground segment. *Website: http://www.esa.int/ESAMultimedia/Images/2018/03/Galileosglobalgroundsegment*, 2018.

[17] Kai Borre, Dennis M Akos, Nicolaj Bertelsen, Peter Rinder, and Soren H Jensen. *A Software-Defined GPS and Galileo Receiver*. SpringerLink, 2007.

[18] Jasmine Zidan, Elijah Adegoke, Erik Kampert, Stewart Birrell, Col Ford, and Matthew Higgins. GNSS Vulnerabilities and Existing Solutions: A Review of the Literature. *IEEE Access*, 2020.

[19] Ni Zhu, Juliette Marais, David Bétaille, and Marion Berbineau. GNSS Position Integrity in Urban Environments: A Review of Literature. *IEEE Transactions on Intelligent Transportation Systems*, PP:1–17, 01 2018.

[20] Anne-Marie Tobie. *GNSS/5G Hybridization for Urban Navigation*. PhD thesis, INP Toulouse, 2021.

[21] Google. Guides to Raw GNSS Measurements - GNSSMeasurement. *https://developer.android.com/reference/android/location/GnssMeasurement.*

[22] Kai Liu, Hock B Lim, Emilio Frazzoli, Houling Ji, and Victor C S Lee. Improving Positioning Accuracy Using GPS Pseudorange Measurements for Cooperative Vehicular Localization. *IEEE Transactions on Vehicular Technology*, 63(6):2544–2556, 2014.

[23] Neil Gogoi, Alex Minetto, and Fabio Dovis. On the Cooperative Ranging between Android Smartphones Sharing Raw GNSS Measurements. In *2019 IEEE 90th Vehicular Technology Conference (VTC2019)*, pages 1–5, 2019.

[24] Elliott Kaplan and Christopher Hegarty. *Understanding GPS/GNSS: Principles and Applications, Third Edition*. Artech House, 2017.

[25] u-blox AG. Product Description: u-blox F9 high precision GNSS module. Website: https://www.u-blox.com/sites/default/files/ZED-F9PProductSummaryUBX-17005151.pdf, Copyright 2021.

[26] Natalia Wielgocka, Tomasz Hadas, Adrian Kaczmarek, and Grzegorz Marut. Feasibility of Using Low-Cost Dual-Frequency GNSS Receivers for Land Surveying. *Sensors*, 21(6), 2021.

[27] Demoz Gebre-Egziabher. Evaluation of Low-Cost, Centimeter-Level Accuracy OEM GN. Technical Report 6, Department of Aerospace Engineering and Mechanics, University of Minnesota, 2018.

[28] Daniel Janos and Przemyslaw Kuras. Evaluation of Low-Cost GNSS Receiver under Demanding Conditions in RTK Network Mode. *Sensors*, 21(16), 2021.

[29] European Union Agency for the Space Programme (EUSPA). GNSS User Technology Report - Issue 3, 2020 - Editor's Special on Space Data for Europe - Website: https://www.gsa.europa.eu/european-gnss/gnss-market/gnss-user-technology-report, 2020.

[30] Inc Qualcomm Technologies. Snapdragon 888+ 5G Mobile Platform. *URL: https://www.qualcomm.com/media/documents/files/qualcomm-snapdragon-888-mobile-platform-product-brief.pdf*, 2021.

[31] Wenlin Yan, Luísa Bastos, and Américo Magalhães. Performance Assessment of the Android Smartphone's IMU in a GNSS/INS Coupled Navigation Model. *IEEE Access*, 7:171073–171083, 2019.

[32] Mohamed Bochkati, Himanshu Sharma, Christian A. Lichtenberger, and Thomas Pany. Demonstration of Fused RTK (Fixed) + Inertial Positioning Using Android Smartphone Sensors Only. In *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pages 1140–1154, 2020.

[33] Zun Niu, Ping Nie, Lin Tao, Junren Sun, and Bocheng Zhu. RTK with the Assistance of an IMU-Based Pedestrian Navigation Algorithm for Smartphones. *Sensors*, pages 19, 3228. https://doi.org/10.3390/s19143228, 2019.

[34] Qiong Wu, Mengfei Sun, Changjie Zhou, and Peng Zhang. Precise Point Positioning Using Dual-Frequency GNSS Observations on Smartphone. *Sensors*, 19(9), 2019.

[35] Guangcai Li and Jianghui Geng. Characteristics of raw multi-GNSS measurement error from Google Android smart devices. *GPS Solutions*, 23:1–16, 2019.

[36] Liang Chen, Heidi Kuusniemi, Yuwei Chen, Ling Pei, Tuomo Kröger, and Ruizhi Chen. Information-Filter-Assisted-Indoor-Bluetooth-Positioning. 05 2015.

[37] Ta, Viet-Cuong. *Smartphone-based indoor positioning using Wi-Fi ,inertial sensors and Bluetooth.* PhD thesis, Hanoi University of Science, 2017.

[38] Santosh Subedi and Jae-Young Pyun. A Survey of Smartphone-Based Indoor Positioning System Using RF-Based Wireless Technologies. *Sensors*, 20(24), 2020.

[39] Franck VanDiggelen, Roy Want, and Wei Wang. How to achieve 1-meter accuracy in android. *GPS World*, 2018.

[40] Chao Ruan, Min Yu, Xiaona He, and Binbin Song. An Indoor Floor Positioning Method Based on Smartphone's Barometer. In *2018 Ubiquitous Positioning, Indoor Navigation and Location-Based Services (UPINLBS)*, pages 1–9, 2018.

[41] Eduardo Sánchez, Michael Botsch, Bertold Huber, and Andres García. High precision indoor positioning by means of LiDAR. In *2019 DGON Inertial Sensors and Systems (ISS)*, pages 1–20, 2019.

[42] Samsung Electronics Co., Ltd - 2017.04Rev1.0. Samsung SM-SMG950U Service Manual - Website: https://www.ifixit.com/Document/AZhlwZlMT2DipqSY/SM-SMG950U.pdf.

[43] Kenneth M. Pesyna, Robert W. Heath, and Todd E. Humphreys. Centimeter Positioning with a Smartphone-Quality GNSS Antenna. *GPS World*, 2015.

[44] Simon Banville and Frank Van Diggelen. Precise GNSS for Everyone: Precise Positioning Using Raw GPS Measurements from Android Smartphones. *GPS World*, 27:43–48, 11 2016.

[45] Franck VanDigglen and Mohammed Khinder. GNSS Analysis Tools from Google. *Inside GNSS*, pages Website, last visited 03/03/2021, 2018. https://insidegnss.com/gnss-analysis-tools-from-google/.

[46] Jimmy LaMance, Javier DeSalas, and Jani Jarvinen. Assisted GPS: A low-Infrastructure Approach. *GPS World, Volume 13, ISNN: 1048-5104*, 2003.

[47] Damian Miralles, Dennis M. Akos, Dong-Kyeong Lee, Andriy Konovaltsev, Lothar Kurz, and Sherman Lo. Robust Satellite Navigation in the Android Operating System using the Android Raw GNSS Measurements Engine and Location Providers. In *2020 European Navigation Conference (ENC)*, pages 1–12, 2020.

[48] Filip Nedelkov, Dong-Kyeong Lee, Damian Miralles, and Dennis Akos. Accuracy and Performance of the Network Location Provider in Android Devices. pages 2152–2165, 10 2020.

[49] Todd E Humphreys, Matthew Murrian, and Lakshay Narula. Low-cost Precise Vehicular Positioning in Urban Environments. *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pages 456–471, April 2018.

[50] Marco Fortunato, Joshua Critchley-Marrows, Malgorzata Siutkowska, Maria L. Ivanovici, Eric Benedetti, and William Roberts. Enabling High Accuracy Dynamic Applications in Urban Environments Using PPP and RTK on Android Multi-Frequency and Multi-GNSS Smartphones. In *2019 European Navigation Conference (ENC)*, pages 1–9, April 2019.

[51] Marco Fortunato, Michela Ravanelli, and Augusto Mazzoni. Real-Time Geophysical Applications with Android GNSS Raw Measurements. *Remote Sensing*, 11:2113, 2019.

[52] Tilman Reinhardt. Using Global Localization to Improve Navigation. *Google AI Blog - Website: https://ai.googleblog.com/2019/02/using-global-localization-to-improve.html*, February 2019.

[53] Ming Li, Ruizhi Chen, Xuan Liao, Bingxuan Guo, Weilong Zhang, and Ge Guo. A Precise Indoor Visual Positioning Approach Using a Built Image Feature Database and Single User Image from Smartphone Cameras. *Remote Sensing*, 12(5), 2020.

[54] Paul Groves. Shadow matching: A new gnss positioning technique for urban canyons. *Journal of Navigation*, 64:417 – 430, 07 2011.

[55] Lei Wang, Paul Groves, and Marek Ziebart. Urban Positioning on a Smartphone: Real-time Shadow Matching Using GNSS and 3D City Models. *Inside GNSS*, 8:44–56, 11 2013.

[56] Shunsuke Hsu, Li Ta; Mirua and Shunsuke Kamijo. Street Smart: 3D City Mapping and Modeling for Positioning with Multi-GNSS. *GPS World*, 2015.

[57] Franck Van Diggelen and Jennifer Wang. Improving Urban GPS Accuracy for your APP. *Android Developers Blog - Website https://android-developers.googleblog.com/2020/12/improving-urban-gps-accuracy-for-your.html*, 2020.

[58] Google. GnssLogger App - Play Store . *URL for .apk download: https://play.google.com/store/apps/*, 2021.

[59] Umberto Robustelli, Valerio Baiocchi, and Giovanni Pugliano. Assessment of Dual Frequency GNSS Observations from a Xiaomi Mi 8 Android Smartphone and Positioning Performance Analysis. *Communications Smart City*, 2019.

[60] Francesco Darugna. *Improving Smartphone-Based GNSS PositioningUsing State Space Augmentation Techniques*. PhD thesis, Veröffentlichungen der DGKAusschuss Geodäsie der Bayerischen Akademie der Wissenschaften, 2021.

[61] Thomas Verheyde, Antoine Blais, Christophe Macabiau, and François Xavier Marmet. Analyzing Android GNSS Raw Measurements Flags Detection Mechanisms for Collaborative Positioning in Urban Environment. *IEEEXplore*, 2020 International Conference on Localization and GNSS (ICL-GNSS):pp. 1–6, doi: 10.1109/ICL–GNSS49876.2020.9115564, 2020.

[62] Daniele Borio, Nadezda Sokolova, and Gerard Lachapelle. Doppler Measurements and Velocity Estimation: A Theoretical Framework with Software Receiver Implementation. In *Proceedings of the 22nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2009), Savannah, GA*, pages pp. 304–316, September 2009.

[63] Nadezda Sokolova, Daniele Borio, and Borje Forssell. Loop Filters with Controllable Doppler Jitter for Standard and HighSensitivity GNSS Receivers. In *In Proceedings of the International Conference on Indoor Positioning andIndoor Navigation (IPIN), Guimarpes, Portugal*, 2011.

[64] Marc Petovello and Salvatore Gaglione. How Does a GNSS Receiver Estimate Velocity. *InsideGNSS - Edition March April 2015*, pages pp 38–41., 2015.

[65] Lehtola Vile V., Stefan Söderholm, Michelle Koivisto, and Leslie Montloin. Exploring GNSS Crowdsourcing Feasibility: Combinations of Measurements for Modeling Smartphone and Higher End GNSS Receiver Performance. *Sensors*, 19(13):1–17, 7 2019.

[66] Fabrice Legrand, Christophe Macabiau, Jean-Luc Issler, Laurent Lestarquit, and Christian Mehlen. Improvement of pseudorange measurements accuracy by using fast adaptive bandwidth lock loops. *Proceedings of the 13th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GPS 2000)*, 09 2000.

[67] Peter F. De Bakker. Effects of Radio Frequency Interference on GNSS Receiver Output. Master's thesis, Delft university of Technology - Faculty of Aerospace Engineering, 2006.

[68] Thomas Verheyde, Antoine Blais, Christophe Macabiau, and François-Xavier Marmet. Statictical Analysis of Android GNSS Raw Data Measurements in an Urban Environment for Smartphone Collaborative Positioning Methods. *International Navigation Conference (INC)*, 2019.

[69] Jacek Paziewski, Marco Fortunato, Augusto Mazzoni, and Robert Odolinski. An analysis of multi-GNSS observations tracked by recent Android smartphones and smartphone-only relative positioning results. *Measurements*, 175:109162, 2021.

[70] Lotfi Massarweh, Francesco Darugna, Dimitrios Psychas, and Jon Bruno. Statistical Investigation of Android GNSS Data: Case Study Using Xiaomi Mi 8 Dual-Frequency Raw Measurements. *Proceedings of the 32nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2019)*, pages pp. 3847–3861., September 2019.

[71] Garello, Letizia LoPresti, E. Giovanni, and C. Samson. Peer-to-peer cooperative positioning, part i: Gnss-aided acquisition. *Inside GNSS*, 2012.

[72] Garello, Jaron Samson, Maurizio A. Spirito, and Henk Wymeersch. Peer-to-peer cooperative positioning, part ii: Hybrid devices with gnss & terrestrial ranging capability. *Inside GNSS*, 2012.

[73] Nicolas Kassabian and Laetizia LoPresti. Mean acquisition time of GNSS peer-to-peer networks. In *International Conference on Localization and GNSS*, pages 1–6, June 2012.

[74] Enrique L. Aguado, Ben Wales, Davide Guerrini, and Jose A. Garcia-Molina. Cloud/Cooperative Navigation - The ICON Concept Demonstrator. *Navitec 2018*, 2018.

[75] Xiao Liu, Miguel Ribot, Adrià Gusi-Amigó, Pau Closas, Adrià Garcia, and Jaume Subirana. RTK Feasibility Analysis for GNSS Snapshot Positioning. In *Proceedings of the 33rd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2020)*, pages 2911 – 2921, 11 2020.

[76] Alex Minetto, Fabio Dovis, Andrea Vesco, Miquel Garcia-Fernandez, Àlex López-Cruces, José Luis Trigo, Marc Molina, Alejandro Pérez-Conesa, Joaquín Gáñez-Fernández, Gonzalo Seco-Granados, and José A. López-Salcedo. A Testbed for GNSS-Based Positioning and Navigation Technologies in Smart Cities: The HANSEL Project. *Smart Cities*, 3(4):1219–1241, 2020.

[77] Lina Deambrogio, Claudio Palestini, Francesco Bastia, Guilio Gabelli, Giovanni E. Corazza, and Jaron Samson. Impact of High-End Receivers in a Peer-to-Peer Cooperative Localization System. *Ubiquitous Positioning Indoor Navigation and Location Based Service*, pages 1–7, Oct 2010.

[78] Wenyang Guan, Jianhua He, Lin Bai, and Zuoyin Tang. Adaptive congestion control of DSRC vehicle networks for collaborative road safety applications. In *2011 IEEE 36th Conference on Local Computer Networks*, pages 913–917, 2011.

[79] Khadige Abboud, Hassan Aboubakr Omar, and Weihua Zhuang. Interworking of DSRC and Cellular Network Technologies for V2X Communications: A Survey. *IEEE Transactions on Vehicular Technology*, 65(12):9457–9470, 2016.

[80] Petar Popovski, Kasper Fløe Trillingsgaard, Osvaldo Simeone, and Giuseppe Durisi. 5G Wireless Network Slicing for eMBB, URLLC, and mMTC: A Communication-Theoretic View. *IEEE Access*, 6:55765–55779, 2018.

[81] Frederik Berefelt, Bengt Boberg, Jonas Nygårds, Peter Strömbäck, and Son L. Wirkander. Collaborative GPS / INS Navigation in Urban Environment. *Proceedings of the 2004 National Technical Meeting of The Institute of Navigation*, pages 1114 – 1125, 2004.

[82] Ilsun Kim, Chansik Park, Gyu-In Jee, and Jang Gyu Lee. Gps positioning using virtual pseudorange. *Control Engineering Practice*, 6(1):25 – 35, 1998.

[83] Bin Huang, Yao Zheng, Xiaomei Cui, Minguan Lu, and Jing Guo. GNSS Collaborative Positioning and Performance Analysis. In *Proceedings of the 27th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2014)*, volume 3, pages 1920–1930, 09 2014.

[84] Ahmed Hussein, Pablo Marín-Plaza, Fernando Garcia, and Jose-Maria Armingol. Autonomous cooperative driving using V2X communications in off-road environment. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, 10 2017.

[85] Ciro Gioia and Daniele Borio. GNSS-based Cooperative Positioning: Approaches and Demonstration. In *Applied Geomatics 13*, pages 195–216, 10 2019.

[86] Huiguang Liang, Hyong S. Kim, Hwee-Pink Tan, and Wai-Leong Yeow. Where am I? Characterizing and improving the localization performance of off-the-shelf mobile devices through cooperation. In *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, pages 375–382, 2016.

[87] Kai Liu and Hock Beng Lim. Positioning Accuracy Improvement via Distributed Location Estimate in Cooperative Vehicular Networks. In *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pages 1549–1554, 2012.

[88] Van Der Laan M and Susan Gruber. Collaborative Targeted Maximum Likelihood Estimation. *Collection of Biostatistics Research Archive*, 2009.

[89] Hector Garcia De Marina, Bayu Jayawardhana, and Ming Cao. Taming Mismatches in Inter-agent Distances for the Formation-Motion Control of Second-Order Agents. *IEEE Transactions on Automatic Control*, 63(2):449–462, 2018.

[90] Chuan-Chin Pu and Wan-Young Chung. A New Approach to Collaborative Ranging Using Received Signal Strength Indicator in Wireless Sensor Network. *Sensor Letters - American Scientific Publishers*, 6(1):pp. 237–241, February 2008.

[91] Alex Minetto, Calogero Cristodaro, and Fabio Dovis. A Collaborative Method for Positioning based on GNSS Inter Agent Range Estimation. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 2714–2718, 2017.

[92] Fabian De Ponte Müller. Survey on Ranging Sensors and Cooperative Techniques for Relative Positioning of Vehicles. *Sensors*, 17(2), 2017.

[93] Benoit Denis, Jean-Benoit. Pierrot, and Chadi. Abou-Rjeily. Joint distributed synchronization and positioning in UWB ad hoc networks using TOA. *IEEE Transactions on Microwave Theory and Techniques*, 54(4):1896–1911, 2006.

[94] Fabian de Ponte Müller, Estefania Munoz Diaz, Bernhard Kloiber, and Thomas Strang. Bayesian Cooperative Relative Vehicle Positioning using Pseudorange Differences. In *2014 IEEE/ION Position, Location and Navigation Symposium - PLANS 2014*, pages 434–444, 2014.

[95] Muhammad Tahir, Sayed Saad Afzal, Muhammad Saad Chughtai, and Khurram Ali. On the Accuracy of Inter-Vehicular Range Measurements Using GNSS Observables in a Cooperative Framework. *IEEE Transactions on Intelligent Transportation Systems*, 20(2):682–691, 2019.

[96] Wang Gao, Chengfa Gao, Shuguo Pan, Xiaolin Meng, and Yan Xia. Inter-System Differencing between GPS and BDS for Medium-Baseline RTK Positioning. *Remote Sensing*, 9(9), 2017.

[97] Bernhard Hofmann-Wellenhof, Herbert Lichtenegger, and James Collins. *Global Positioning System. Theory and Practice.* Springer-Verlag Wien, 02 2001.

[98] Fabian De Ponte Müller, Alexander Steingass, and Thomas Strang. Zero-Baseline Measurements for Relative Positioning in Vehicular Environments. *6th European Workshop on GNSS Signals and Signal Processing*, 05-06 Dec 2013.

[99] Padma Bolla and Eduard Lohan. Dual-frequency signal processing architecture for robust and precise positioning applications. In *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pages 72–80, April 2018.

[100] Dominik Prochniewicz, Kinga Wezka, and Joanna Kozuchowska. Empirical Stochastic Model of Multi-GNSS Measurements. *Sensors*, 21(13), 2021.

[101] D. Gebre-Egziabher and S. Gleason. *GNSS Applications and Methods.* GNSS technology and applications series. Artech House, 2009.

[102] Maurizio A. Caceres, Antonio A. Damico, Dardari Davide, Mats Rydstrom, Francesco Sottile, Erik G. Strom, and Lorenzo Taponecco. *Satellite and Terrestrial Radio Positioning Techniques: A signal processing perspective*, chapter Chapter 3: Terrestrial Network-Based Positioning and Navigation, pages 75–153. Elsevier, Oxford, 2012.

[103] Mohinder S. Grewal, Angus P. Andrews, and Chris G. Bartone. *Global Navigation Satellite Systems, Inertial Navigation, and Integration*, chapter Differential GNSS, pages 293–330. Wiley Telecom, 2020.

[104] Richard L. Smith and John C. Naylor. A Comparison of Maximum Likelihood and Bayesian Estimators for the Three- Parameter Weibull Distribution. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 36(3):358–369, 1987.

[105] Pau Closas-Gomez. *Bayesian Signal Processing Techniques for GNSS Receivers: from Multipath Mitigation to Positioning.* PhD thesis, Universtitat Politecnica de Catalunya (UPC), 2009.

[106] Alex Minetto, Gianluca Falco, and Fabio Dovis. On the Trade-Off between Computational Complexity and Collaborative GNSS Hybridization. In *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, pages 1–5, 2019.

[107] Donghwan Yoon, Changdon Kee, Jiwon Seo, and Byungwoon Park. Position Accuracy Improvement by Implementing the DGNSS-CP Algorithm in Smartphones. *Sensors*, 16(6), 2016.

[108] Jihua Huang and Han-Shue. Tan. A Low-Order DGPS-Based Vehicle Positioning System Under Urban Environment. *IEEE/ASME Transactions on Mechatronics*, 11(5):567–575, 2006.

[109] Mohsen Rohani, Denis Gingras, and Dominique Gruyer. A Novel Approach for Improved Vehicular Positioning Using Cooperative Map Matching and Dynamic Base Station DGPS Concept. *IEEE Transactions on Intelligent Transportation Systems*, 17(1):230–239, 2016.

[110] Mike J. D. Powell. *Mathematical Programming: The State of the Art*, chapter Variable Metric Methods for Constrained Optimization, pages 288–311. Springer Berlin Heidelberg, Berlin, Heidelberg, 1983.

[111] Klaus Schittkowski and William Hock. A Comparative Performance Evaluation of 27 Nonlinear Programming Codes. *Computing 30*, pages 335–358, 1983.

[112] Thierry Chapuis, Bernard Bonhoure, Sébastien Rougerie, Frédéric Lacoste, Thomas Grelier, Didier Lapeyre, and Pierre Noirat. SPRING: A powerful 3D GNSS Simulator for Constraint Environment. In *Proceedings of the 27th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2014)*, pages pp. 454–465, 2014.

[113] MathWorks. Tolerance details for the optimization toolbox. Website, last visit 02/07/2022, 2022. https://de.mathworks.com/help/optim/ug/tolerance-details.html.

[114] European Union Agency for the Space Programme (EUSPA). FLAMINGO: Fulfilling Enhanced Location Accuracy in the Mass-market Through Initial Galileo Services. Website: https://www.euspa.europa.eu/fulfilling-enhanced-location-accuracy-mass-market-through-initial-galileo-services, Website: https://www.euspa.europa.eu/fulfilling-enhanced-location-accuracy-mass-market-through-initial-galileo-services.

[115] Loh, Robert and Wullschleger, Victor. and Elrod, Beatrice and Lage, Mike and Haas, Francis. The U.S. Wide-Area Augmentation System (WAAS). *Computer Science: Annual Navigation*, 42(3):435–465, 1995.

[116] Rudolf E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *ASME Journal of Basic Engineering*, pages pp 35–45, 1960.

[117] Antonio Angrisano, Salvatore Gaglione, and Ciro Gioia. Performance Assessment of GPS / GLONASS Single Point Positioning in an Urban Environment. *Acta Geodaetica et Geophysica*, 48:149–161, 2013.

[118] Paul D. Groves, Ziyi Jiang, Lei Wang, and Marek Ziebart. Intelligent Urban Positioning, Shadow Matching and Non-Line-of-Sight Signal Detection. In *6th ESA*

Workshop on Satellite Navigation Technologies (Navitec 2012) European Workshop on GNSS Signals and Signal Processing, pages 1–8, Dec 2012.

[119] Oliver Leisten and Viktor Knobe. Optimizing Small Antennas for Body-Loading Applications. *GPS World*, 23:40–44, 09 2012.

[120] Paul D. Groves, Zi Jiang, Barbel Skelton, Paul A. Cross, Lawrence Lau, Yacine Adane, and Izzet Kale. Novel Multipath Mitigation Methods using a Dual-polarization Antenna. In *23rd International Technical Meeting of the Satellite Division of the Institute of Navigation 2010, ION GNSS*, volume 1, pages 140–151, 09 2010.

[121] Jin Lee, Deok Won Lim, Jae Noh, Gwang Jo, Chansik Park, Jae-Min Ahn, and Sang Lee. A GPS Multipath Mitigation Technique using Correlators with Variable Chip Spacing. *E3S Web of Conferences*, 94:03006, 01 2019.

[122] Paul Groves, Ziyi Jiang, Morten Rudi, and Philip Strode. A Portfolio Approach to NLOS and Multipath Mitigation in Dense Urban Areas. volume 4, 09 2013.

[123] Gaetano Castaldo, Antonio Angrisano, Salvatore Gaglione, and Salvatore Troisi. P-RANSAC: an Integrity monitoring approach for GNSS signal degraded scenario. *International Journal of Navigation and Observation*, 2014, 09 2014.

[124] Sébastien Carcanague. *Low-cost GPS/GLONASS Precise Positioning Algorithm in Constraint Environment.* PhD thesis, ENAC, 2013.

[125] David Bétaille, Stéphan Miquel, Fred Godan, and François Peyret. 3D-City-Model-Aided GNSS Accurate Positioning with Integrity Provision using Simplified Geometry of Buildings. In *European Navigation Conference*, 04 2015.

[126] Lei Wang, Paul D. Groves, and Marek K. Ziebart. Multi-Constellation GNSS Performance Evaluation for Urban Canyons Using Large Virtual Reality City Models. *Journal of Navigation*, 65(3):459–476, 2012.

[127] Jonathan Bradbury, Marek Ziebart, Paul A. Cross, Peter. Boulton, and Alex. Read. Code Multipath Modelling in the Urban Environment Using Large Virtual Reality City Models: Determining the Local Environment. *Journal of Navigation*, 60:95 – 105, 01 2007.

Abstract:

After Google announced the release of Android GNSS raw data measurements on mobile devices, the enthusiasm around those low-cost positioning devices quickly grew in the scientific community. The increasing need of Location Based Services (LBS) provoked the rapid evolution of smartphones embedded low-cost Global Navigation Satellite System (GNSS) chipsets within the last few years. However, various drawbacks prevent the realization of advanced positioning techniques on hand-held mobiles. Smartphones positioning capabilities are limited by the tight-integration of hardware components within the device. This research work ambitions to develop a collaborative network positioning system between smartphones. The implementation of a cooperative smartphone network takes advantage of the tremendous number of connected Android devices present in today's city centers for refining and improving users position accuracy and integrity in urban environment. This research thesis presents a thorough analysis of Android GNSS raw data measurements aiming at lifting the ambiguity generated by receivers' "black-box" processes on a wide variety of Android smartphone brand and models. After grasping the receivers' mechanisms, the implementation of Android GNSS raw data measurements in collaborative positioning algorithm has been investigated. An innovative smartphone-based double code difference method has been employed to compute the inter-phone distance between network's users, named Inter-Phone Ranging (IPR). This technique was tested for nominal and urban scenario cases and has demonstrated its reliability for collaborative positioning implementation. Finally, a smartphone-based cooperative engine, called *SmartCoop*, was developed and evaluated. This collaborative estimation technique exploits the previously computed IPR ranges in a non-linear constrained optimization problem. An experimental protocol has been put in place in order to determine the estimation method efficiency through a series of simulation runs for both nominal and urban scenarios. The presented results analysis supports our hypothesis that smartphone-based collaborative engine enhances Android positioning performance in urban canyon.

Keywords:

Smartphone, Collaborative Positioning, Inter-Phone Ranging (IPR), SmartCoop Engine, Android GNSS Raw Data Measurements

Résumé:

Après l'annonce faite par Google, concernant la mise en service d'une mise à jour Android permettant de récupérer les mesures GNSS brutes, les smartphones sont rapidement devenus attrayant pour la communauté scientifique, en tant que récepteur GNSS bas-coût grand public. Cependant, la performance du positionnement sur mobile est rapidement impactée par différent biais d'erreur. Ce phénomène s'accentue en milieu urbain, notamment à cause de l'antenne interne du téléphone dont les spécifications sont inadaptées au traitement de signaux multifréquences et au positionnement en environnement contraint. Afin de surmonter ces difficultés, ce projet de recherche ambitionne le développement d'un algorithme collaboratif dédié aux smartphones. La création de ce réseau coopératif permettrait de tirer avantage du nombreux croissant de téléphone mobile connecté aggloméré en centre urbain. Après avoir analysé les mécanismes de positionnement interne aux smartphones, la mise en place d'un algorithme collaboratif a été instaurée utilisant les données GNSS d'Android. Une méthode de double différence sur les mesures de code a été proposée afin de permettre l'estimation des distances entre utilisateurs d'un réseau coopératif. Cette technique innovante a été baptisée "Inter-Phone Ranging" (IPR). La fiabilité et la précision de cette méthode d'estimation sont démontrées par plusieurs études couvrant plusieurs environnements. Enfin, après avoir méthodiquement caractérisé la mise en place d'un réseau collaboratif de téléphone mobile, un algorithme de positionnement collaboratif appelé "SmartCoop" est présenté. Ce dispositif permet d'exploiter les mesures d'inter-distances entre utilisateurs du réseau afin de résoudre un problème d'optimisation non-linéaire à contraintes. Cette méthode d'estimation a pour but d'améliorer la précision et la dispersion de la position de tous les utilisateurs du réseau. L'analyse des résultats obtenus nous permet de penser que cet algorithme coopératif innovant participe à l'amélioration globale de la performance du positionnement sur téléphone mobile en milieu urbain.

Mots-Clé:

Smartphone, Positionement Collaboratif, Inter-Phone Ranging (IPR), SmartCoop Engine, Android GNSS Raw Data Measurements