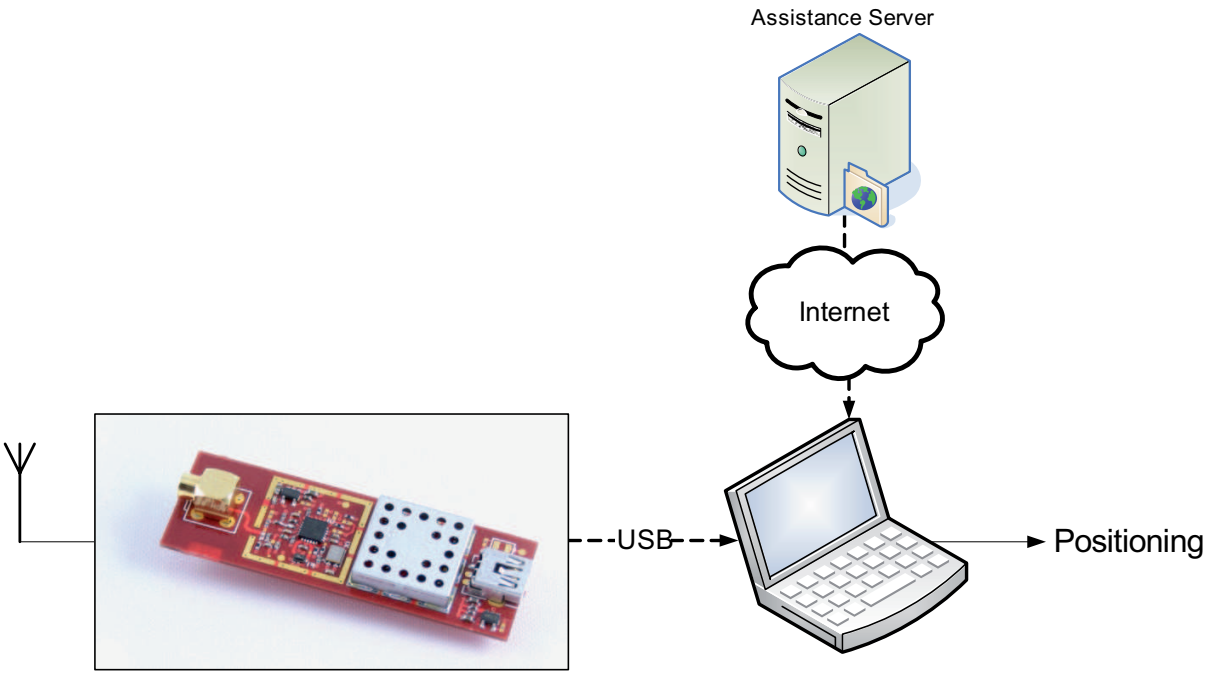
MASTER THESIS



TITLE: Enhancement Study of GPS Snapshot Techniques

AUTHORS: Oriol Badia Solé
Tudor Iacobescu Ioan

SUPERVISOR: Kai Borre

DATE: January 5, 2011

AALBORG UNIVERSITY

Studyboard for Danish GPS Center
Fredrik Bajers Vej 7 B4-207, 9220 Aalborg Ø
Telephone +45 9940 8714
www.esn.aau.dk

Title
Enhancement Study of
GPS Snapshot Techniques

Type
Master Thesis for GPS technology

Period
1st September 2010 to
5th January 2011

Semester and group
10th semester, GPS technology,
group 1230

Participants

Tudor Iacobescu Ioan

Oriol Badia Solé

Supervisor
Kai Borre

Copies: 4

Pages: 63 (appendix: 5)

CD included

Webpage: `kom.aau.dk/group/10gr1230/`

e-mail: `10gr1230@es.aau.dk`

Synopsis

GPS snapshot techniques are becoming more and more popular in the scientific community mostly due to the fact that they involve minimizing hardware components and reducing power consumption, both aspects leading to dramatic cost reductions.

The motivation of the present project is to provide new functionality to GPS snapshot techniques. We focused on two main directions: improving position accuracy and making the application available for non-experienced users by automatically providing necessary assistance data.

These concepts are approached by automating the process of providing assisted data, by analyzing, implementing, and comparing different tracking algorithms and pseudo-range corrections.

Our results demonstrate improvements in accuracy levels and energy consumption.

Acknowledgements

We would like to take this opportunity to express our gratitude and appreciation to all the people who assisted us in preparation of this thesis and who have made our overall experience at DGC both fun and challenging.

We would like to thank Prof. Dr. Kai Borre and to Dr. Darius Plaušinaitis for the theoretical and technical insights they have shared, and in general for all the help they have provided. We would also like to thank the entire DGC for providing us with all the necessary equipment and tools that made the realization of this project possible.

Last, but not least, we would also like to thank our families for all the support and encouragement they have provided all along our stay in Aalborg University.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

The GPS system was created in order to satisfy the need of several U.S. governmental institutions and organizations such as Department of Defence (DoD), the National Aeronautics and Space Administration (NASA) agency and the Department of Transportation, for a three-dimensional positioning system which would be able to provide [4]:

- Global coverage
- Continuity
- Availability
- Capability to serve high dynamical platforms

Despite its military origin, a civilian signal was also included in the original design. Initially, the accuracy of this signal was degraded and controlled through Selective Availability (SA). However SA was removed in the year 2000, leading to an explosion of civilian applications. Nowadays GPS is used for a great variety of civil and scientific applications which include: position and navigation, land surveying (from cartography to monitoring of tectonic plates movement), animal tracking, troposphere and ionosphere monitoring and timing synchronization of network equipments.

The first mainstream application that really spiked the interest of the general public was the portable car navigator, which enabled its user to visualize his/her position on a map, plan a route and get driving directions to the destination. Since then, this field witnessed a continuous growth, bringing GPS uses into everyday life. Perhaps, the last important boost in mass-market applications is the creation of A-GPS (Assisted-GPS), pushed by the U.S. Federal Communications Commission, which was aiming to provide position information for the 911 emergency service, thus facilitating search and rescue operations after emergency calls from cell phones.

More recently, another positioning technique called snapshot technique was born and it is believed to bring new and promising applications for hand-held devices. The "GPS Snapshot Techniques" project [5], done in the last semester, focused on the feasibility of snapshot techniques in the framework of geo-tagging photographs in digital cameras. The present project attempts to push forward, taking over some of the future work that concluded our last project, mainly focusing on providing a greater degree of assistance and improving positioning accuracy. Therefore, this report forgets a little bit about the application and concentrates more on the limits of the snapshot techniques.

As a result, for this project we expect a more finished system in a sense of self-sufficiency and performance.

CHAPTER 1. PROBLEM FORMULATION

This chapter will define the main problem which was approached in this project, along with the work plan and our aims.

1.1 Previous Results

As stated in the introduction, this project is a continuation of project [5]. Therefore, a quick review of the achieved results may put the reader into context —read appendix A for further explanation of these results.

Firstly, the overall conclusion was the proof of concept of snapshot techniques itself: The use of a RF (Radio Frequency) front-end in order to take an IF (Intermediate Frequency) snapshot, which later will be post-processed into a computer by performing acquisition, and then computing the coarse time navigation algorithm.

Concerning acquisition it was shown that non-coherent power integration provided the best results as far as the detection of space vehicles (SVs) and false alarms are concerned. Our experiments provided us with information about the optimal acquisition threshold (our previous experiments have shown that 2 is the optimal value for the threshold) and integration period (7 ms) values. A tradeoff between algorithm speed and performance was also observed due to signal power integration.

The most interesting results obtained concerned position accuracy. A position error between 20 to 50 m was achieved without using tracking, meeting the expected theoretical accuracy. We have also shown that the algorithms used are robust, being able to converge under biased assisted data:

- About 100 km error in apriori guess
- ± 90 s bias in snapshot record time

After the completion of our previous project we found that other companies were as well interested in geo-tagging digital photographs and came up with solutions in order to answer the market's demand for such a product. As an example uBlox has developed a "Capture and Process" system and created the YUMA software which works on the same basic principles. The fact that important companies are manifesting interest in this subject, strengthened our opinion that using GPS Snapshot Techniques for digital photography geo-tagging is a subject worth pursuing. Therefore we are trying to bring improvements to our initial idea, hoping to get closer to a product that would become an alternative to the already existing technologies.

1.2 Future Tendencies

Nowadays a new trend is emerging in social networks: sharing your location with friends and family. This trend is closely related to the ever increasing popularity of social network web-sites and to the explosive growth in smart-phone and digital camera technology. Several applications which allow sharing position have already been launched on the market. Google Inc. has developed "Google Latitude" which allows mobile phone users to share their location with close contacts. Another application that is quickly gaining popularity is "Sport Tracker", created by the Nokia Corporation, which allows its users to share location and details of their workouts. The largest social network web-site, Facebook®, has created "Places" which according to the company CEO, Mark Zuckerberg, would enable the users to share where they are in a fun and social way, see who is around them and discover new places. The company believes the location services will be a powerful new addition to the social network. Other companies like Glympse Inc. and Loopt® are focused on Location-Based Services (LBS) online. The mobile phone and smart-phone market

has also started to show a growing interest in this field. Applications such as LOCiMOBILE, GPS Tracking Lite, iLOCi2 Lite, Friends Around Me, Find My Friend, FriendMapper and many others are already available.

Nowadays, basically all LBS obtain positioning information either through the nearest cell tower(s), Wi-Fi router(s) pinpoint or an embedded GPS receiver. These are traditional hardware receivers that require tracking of the GPS signal in order to obtain a position before being able to share it. This tracking operation consumes a lot of power considering a hand-held device and their continuous operation for LBS is a discharge drawback for batteries. The motto "The software is the instrument" launched by National Instruments seems to perfectly describe the future tendencies in technology.

With the growing popularity of LBS, one cannot help but to observe that there is a domain which is growing in parallel with the growth of social networks and smart phone technology. This domain is digital photography. Besides simply sharing photographs, which is very common in the present, people will more eagerly want to show exactly where and when they were there. This is why we think that we are just the tip of the iceberg when it comes to digital photography geo-tagging, and we expect this technology to become very popular in the near future.

1.3 Problem Identification

Due to the fact that it was previously shown that GPS snapshot techniques can be used in order to geo-tag digital photographs, the next questions arise:

- Is it possible to provide all the necessary assistance data automatically, in a way which is transparent for the user?
- Can the accuracy of the positioning obtained by using GPS snapshot techniques be improved?

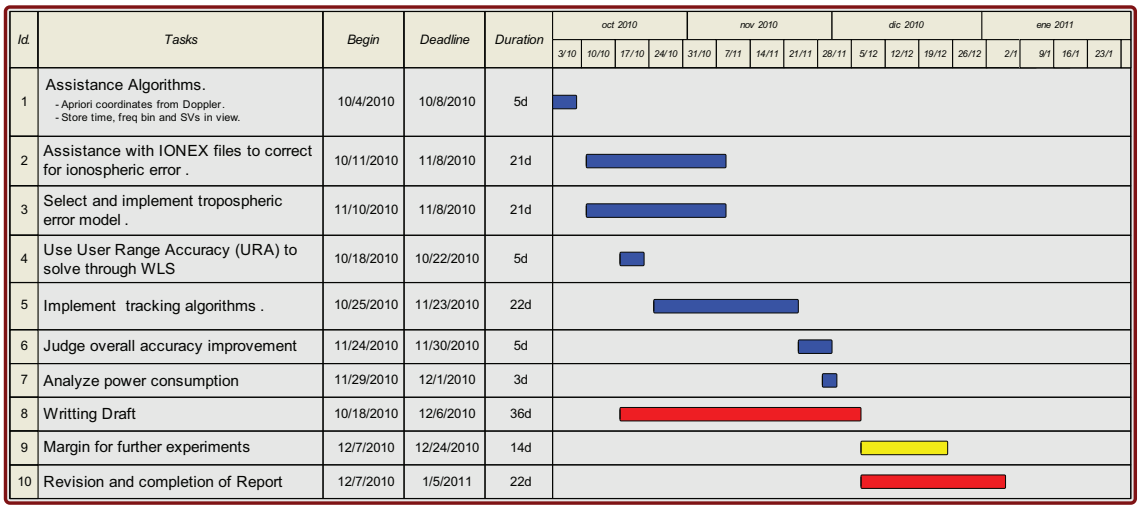
In order to answer these questions an analysis of GPS signal tracking, GPS errors, and assistance data should be performed.

1.4 Project Scope

The scope of the project is to bring new functionality to the already existing post-processing software of a GPS snapshot system. The main focii of the project are:

- Provide assisted time and automatically download ephemerides
- Implement assistance algorithms: compute initial position based on Doppler measurements, store initial position, satellites in view and acquisition results for ulterior snapshots
- Correct for ionosphere delays —by using IONEX files— and troposphere delays —by using an appropriate tropospheric model
- Correct for relativistic errors and total group delay (TGD)
- Implement different tracking algorithms and compare results
- Analyze overall accuracy improvement
- Analyze power consumption

The focii stated above as well as their projected development times are shown in the Gantt diagram presented in Figure 1.1.



Id.	Tasks	Begin	Deadline	Duration
1	Assistance Algorithms. - Apriori coordinates from Doppler. - Store time, freq bin and SVs in view.	10/4/2010	10/8/2010	5d
2	Assistance with IONEX files to correct for ionospheric error .	10/11/2010	11/8/2010	21d
3	Select and implement tropospheric error model .	11/10/2010	11/8/2010	21d
4	Use User Range Accuracy (URA) to solve through WLS	10/18/2010	10/22/2010	5d
5	Implement tracking algorithms .	10/25/2010	11/23/2010	22d
6	Judge overall accuracy improvement	11/24/2010	11/30/2010	5d
7	Analyze power consumption	11/29/2010	12/1/2010	3d
8	Writting Draft	10/18/2010	12/6/2010	36d
9	Margin for further experiments	12/7/2010	12/24/2010	14d
10	Revision and completion of Report	12/7/2010	1/5/2011	22d

Figure 1.1: Project Gantt diagram. Blue symbolizes development time, red means the time spent on the report writing and yellow suggests a time margin we have allowed for further experiments.

1.5 Final Problem Formulation

This project focuses on providing a solution for day-to-day life applications, more explicitly geo-tagging digital photographs. This is actually a continuation of our previous project [5] which has shown that using GPS snapshot techniques in order to geo-tag digital photographs is a feasible solution due to the few hardware modifications which should be made in the digital camera, the expected low power consumption and the accuracy levels which can be achieved. The purpose of our present work and research is to bring additional features and functionality to the previous design.

The project will focus on two main directions:

- Providing more assistance data such as assisted time and ephemerides, and calculating first fix, therefore making the application available for non-experienced users
- Improving the position results by applying tracking algorithms, and by analyzing and correcting different GPS error sources

While building a GPS-based product for non-professional users, it should be kept in mind that the final result should be user-friendly and should provide added value for money, it should consume as less power as possible and it should integrate today's miniaturization tendencies. In this sense, miniaturization and power reduction have played a major role facilitating the embedding of GPS systems into hand-held devices.

CHAPTER 2. ANALYSIS

Prior to focusing ourselves on specific problems, an initial analysis is important in order to organize them according to their potential benefits. Hence, different priorities will be assigned to each task, ending up with a sorted list of tasks to be addressed within the time period of the project. This chapter is devoted to provide a brief theoretical description of each problem. This should allow the reader to have a better comprehension about each problem and its priority.

Given that our problem formulation states two fairly different action fronts, the content of this chapter has also been split into the same two sections. In the end, the chapter summary contains a table summarizing each task and its priority.

2.1 Overall Positioning Accuracy

Firstly, the signal tracking theory is presented, followed by a brief description of the main pseudo-range errors. Note that pseudo-range errors group all the remaining effects to be corrected even after an ideal tracking of the signal.

2.1.1 Signal Tracking

The main purpose of GPS signal tracking is to refine the rough estimates of the frequency and code phase parameters obtained during acquisition, keep track of the signal and demodulate navigation data [1]. The acquisition resolution is mainly limited by the sampling frequency at the front-end. Code phase measurements, for instance, can be achieved down to sample precision, which means:

$$\text{Resolution} = \frac{c}{\text{Sampling Frequency}} \simeq 18.32\,\text{m}$$

Performing signal tracking is a method to overcome this limitation by solving the time transfer issue and achieve sub-sample resolution of the signal code phase. Therefore, implementing tracking is the most important task in order to get down to the accuracy level where pseudo-range errors, see section 2.1.2, become the next accuracy boundary.

Figure 2.1 is copied from reference [1] and represents the basic demodulation scheme. Tracking systems usually consist of two major tasks: code tracking and carrier tracking.

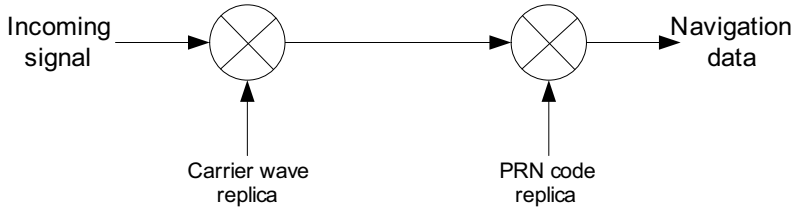


Figure 2.1: Basic demodulation scheme

2.1.1.1 Carrier Tracking

As shown in Figure 2.1, in order to be able to demodulate navigation data the first task that should be done is to generate an exact carrier wave replica and multiply this replica with the incoming signal in order to eliminate the carrier wave. Typically, carrier wave tracking is done with a Phase Lock Loop (PLL) or Frequency Lock Loop (FLL). Due to the fact that navigation data bits may cause phase transitions, GPS receivers usually use Costas Loops (Figure 2.3). Unlike PLL, Costas Loops are insensitive to 180° phase shifts as demonstrated in Figure 2.2.

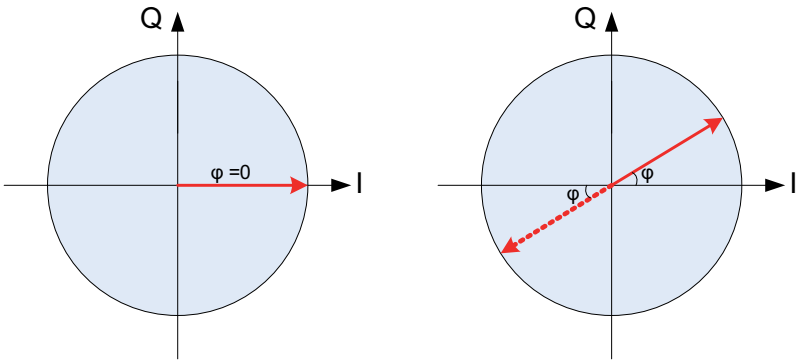


Figure 2.2: Phasor diagram. Left : Input signal is in-phase with the locally generated carrier wave. Right: The locally generated carrier wave is offset in respect to the input signal, therefore, φ will represent the phase error. Even for a phaseshift of 180° the phase error will remain the same.

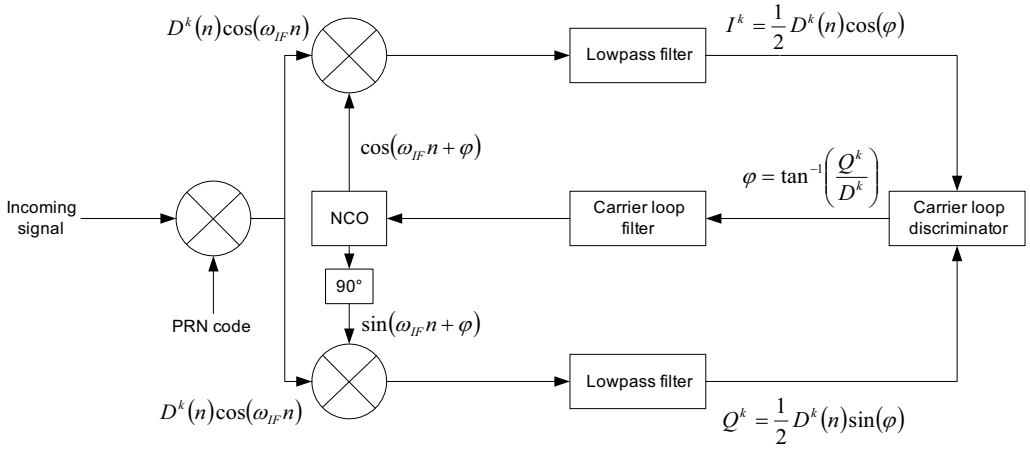


Figure 2.3: Costas Loop block diagram. $D^k(n)$ is the navigation data sequence after analog-to-digital conversion and φ is the phase error

The basic purpose of the Costas loop is to keep all the energy in the in-phase arm [1]. In order to be able to produce an exact replica of the carrier wave, the feedback controlled by the carrier loop discriminator is used. The discriminator is an error detector, and it is expressed as a function of phase. In Figure 2.3 the discriminator function is $\tan^{-1}$ which will output phase error. The carrier loop filter is used to filter the phase discriminator in order to compensate for the motion of the satellite and for estimating the Doppler frequency.

2.1.1.2 Code Tracking

Code tracking seeks to synchronize a specific code in the incoming signal with the one generated locally. Code tracking is typically achieved through a Delay Lock Loop (DLL) which implements an early-late correlator (see Figure 2.4 taken from [1]). This loop should provide a perfectly aligned code replica. After conversion of the incoming signal to baseband –this is done by multiplying the incoming signal with a carrier wave replica that has been aligned through carrier tracking– the local code generator outputs three code replicas E (Early), P (Prompt) and L (Late). These replicas are delayed in respect to each other by half of chip which are multiplied with the baseband signal previously obtained. Basically three different correlations are performed, and based on the highest correlation result, the code must be delayed, advanced or left untouched. The code is perfectly aligned if the correlation with the P replica provides the highest result. The concept is described more in detail in [1] and [6] and it is depicted in Figure 2.5.

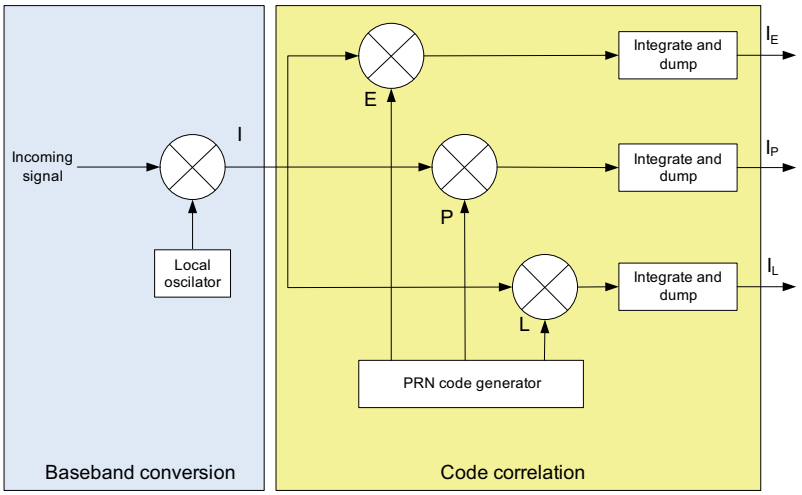


Figure 2.4: Early-Late DLL

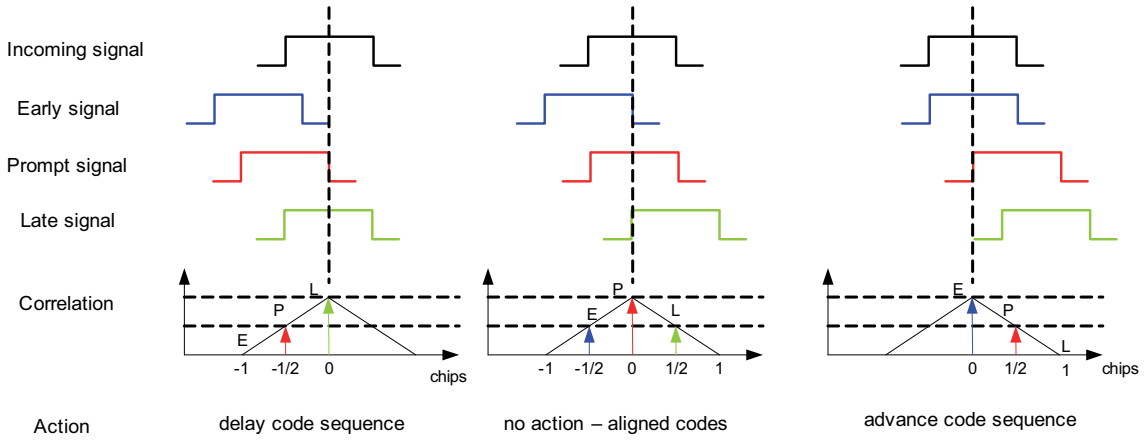


Figure 2.5: Early-Prompt-Late correlation

2.1.2 Pseudo-Range Error Budget

GPS measurements are affected by several error sources. We will provide a short description of the main contributions and some mitigation techniques. A more detailed survey of these errors can be found in references [7], [4],[8] and [9].

In some of the references, the space vehicle (SV) clock offset, the bias between each onboard network of clocks in the satellites and the official GPS time, is also classified as a pseudo-range error because, like all the timing errors in the system, they certainly have a critical impact on the measurement. Hence, correcting for them is not an option but a mandatory step.

For this reason, in addition to the fact that they are not originated during signal transmission, like all other pseudo-range errors described here after, we have decided not to treat them in this section.

2.1.2.1 Ionospheric Delay

The ionosphere is the upper layer of the atmosphere, expanding in altitude from 70 km up to more than 1000 km. It has a high content of free electrons due to photo-ionization which is produced by the ultraviolet solar radiation. These electrons, electrically charged atoms and molecules surrounding the Earth create an electromagnetic shell made out of ionized gas. Thus, GPS signals do

not travel at vacuum speed of light while traveling through this region, therefore producing delays (errors) in the satellite-user time of arrival.

The ionosphere delay depends on the signal's frequency f (dispersive medium) and elevation angles.

$$\delta_{ion} = F_{pp} \frac{40.3}{f^2} \text{ TEC [s]} \tag{2.1}$$

where:

TEC	is the Total Electron Content, normally expressed in electrons per surface unit
F_{pp}	is an obliquity factor or mapping function that accounts for the increased path length that the signal will travel within the ionosphere as a function of its angle

Because the ionosphere depends on many different factors: solar cycle, geographical location and time of day, it is difficult to accurately estimate its value. In order to provide some information on the state of the ionosphere, eight parameters based on the model developed by Klobuchar are transmitted in the navigation message, allowing single frequency users to correct about 60 % of the ionospheric delay.

Dual frequency receivers, on the other hand, are able to correct about 99.9 % of the ionosphere delay, taking advantage of its frequency dispersion. TEC is computed from the difference of arrival times at different carrier waves:

$$\text{STEC} = \frac{(\Delta t_2 - \Delta t_1) f_1^2 f_2^2}{f_1^2 - f_2^2} \tag{2.2}$$

STEC means Slant TEC and the subindeces represent the carrier waves (L_1 and L_2 in this case).

Currently, GPS is an important tool for the study of ionosphere and troposphere and the International GNSS Service, as well as other agencies and research institutes dedicated to astronomy and physics, serve TEC data stored within a standardized file format called IONosphere map EXchange format (IONEX), see section 3.1.2.1 . This format is based on a first proposal by Stefan Schaer in 1996, which strongly followed the Receiver Independent Exchange format (RINEX). The latest IONEX version supports the exchange of 2- and 3-dimensional TEC maps and their root mean square (RMS) values given in a geographic grid.

The use of global ionospheric maps gives us higher accuracy in computing the ionosphere delay and also a very detailed description of the ionosphere's nature. So IONEX files could be used as assistance data to correct GPS solutions which are taken by a single frequency receiver, correcting for the ionosphere delay over the pseudo-range measurements to achieve more accurate position in post-processing.

Obviously, their use demands more processing resources in our computer (not in the front-end) but it is a price to pay in return for better accuracy. Another drawback is that the IONEX files are daily updated, as well as the 24h RINEX navigation files already in use.

2.1.2.2 Tropospheric Delay

The troposphere is the lowest layer of the atmosphere. It can rise between $7 - -20$ km, depending on the time of year and latitude. The troposphere contains almost 75% of the atmosphere's mass and 99% of its water vapor and aerosols. Therefore most of the weather phenomena occur in the troposphere [6].

The troposphere affects radio waves differently than the ionosphere in the way that it is a *nondispersive* medium, meaning that it is independent of radio frequency. The radio waves are nevertheless refracted by the changing refractive index causing scintillations, depolarization or delays.

Tropospheric delays are commonly expressed as a combination of two components:

- the "*wet*" component, water vapors, is usually around 10% of the total effect
- the "*dry*" component, oxygen and nitrogen in hydrostatic equilibrium, is the rest of 90%

Due to its continuous variation and unpredictability the "*wet*" component is the most difficult to mitigate.

Techniques of troposphere error mitigation include using Differential GPS (DGPS), solving for tropospheric delay based on elevation angle values and tropospheric modeling. The latter one will be described in more detail in Section 3.1.2.2 .

2.1.2.3 Relativistic Errors

The word relativity evokes two kind of different errors in GPS systems: on one side, the "Sagnac Effect" and, on the other side, the "General Relativity" encompassing the theories of Albert Einstein.

Concerning "Sagnac Effect", physical laws hold in all inertial frames. One has to take into account that the GPS system uses two different coordinates systems: Earth-Centered, Earth-Fixed (ECEF) and ECI (Earth Center Inertial). In this sense, the user shall take into account the Earth rotation corrections during time of flight of the signal.

In respect to "General Relativity", one has to consider that the GPS system relies on clocks in orbit, flying at high speeds and in different gravitational planes relative to Earth's surface. The space-time curvature due to the Earth's mass is less than it is at the Earth's surface. Therefore, clocks closer to a massive object will seem to tick slower than those located further away. Luckily, scientists counteracted this effect by slowing down the frequency of the onboard atomic clocks so that, once they were in orbit, the clocks would appear to tick at the correct rate as compared to the reference clocks in the ground stations. However, the user still needs to correct for a smaller part of the problem, and that is a component depending on the high speeds of the satellites. The mathematical equation for this correction can be found in [3]:

$$\delta_{\text{rel}} = Fe\sqrt{a}\sin(E_k) \tag{2.3}$$

e is the eccentricity.
a is the semi-major axis.
E_k is the eccentric Anomaly.
and the constant $F = \frac{-2\sqrt{\mu}}{c^2} = -4.442807633 \cdot 10^{-10}$.

This last left-over correction is optional and is the only one we will refer to while below talking about relativistic error.

2.1.2.4 Total Group Delay

The Total Group Delay (TGD) is a term provided by the ground segment that mainly accounts for the instrumental delays of the satellites. Specifically, it is the delay suffered by the signal in the circuitry of the satellite: between the creation instant and its transmission through the antenna. Each satellite has a different one and it is transmitted to the user in the navigation message.

This would lead to think that a similar parameter has to be taken into account for the delay suffered on the receiver. Nevertheless, such delay is constant for all signals so it is canceled out or absorbed in the navigation equations by the offset of the receiver clock.

2.1.2.5 Multipath Errors

Usually, the receiver also acquires signals which are not only the direct line-of-sight but some replicas that have been bounced by one or more obstacles. Therefore they are delayed in time, phase shifted, and attenuated. However, their contribution may distort the spectrum of the incoming signal, biasing the measurements from the correlators.

Multipath is the ultimate accuracy limiting phenomenon in Differential GPS techniques. Nowadays there are several techniques used to mitigate or minimize the multipath effects. These techniques may include: using multiple antenna groundplanes, using phase observations, averaging measurements over 24 hours, or implementing a receiver design with multiple correlators, or very narrow correlation functions.

All these techniques require hardware modifications that will only pay off for expensive professional equipment but not for a mass market front-end. Thus, the only remaining choice is modeling which is rather complicated in a dynamic scenario like our case-study. For this reason we have rejected to apply any multipath corrections.

2.2 Assistance Data

By assisted data we understand all data provided through an external source rather than the GPS signal itself. In this context, we would like to formally break down assistance data into two main categories:

2.2.1 Mandatory Assistance Data

This refers to all required data that must be provided to the algorithm in order to obtain positioning.

- Time: Coarse-time algorithm requires, as its name suggests, a time guess. As demonstrated in [5], this variable shall not be biased by more than 90 s. It shall be provided as Time of Week, second within the current GPS week, format. Such value was obtained from the Operating System metadata of the snapshot file and manually input to the system.
- Apriori user coordinates: A rough fix of the user location is also required by the coarse-time algorithm. The coordinates shall not be biased by more than 150 km to the actual origin, as demonstrated in [5]. Initially, these were manually introduced by the user. An algorithm based on Doppler measurements after acquisition stage may be a solution allowing the receiver to auto-assist itself with such information.
- Ephemeris: An accurate knowledge about satellite trajectories is necessary in order to compute its position at any given time, this being the basis of the trilateration concept. Ephemeris are formed by the almanac with the fundamental Keplerian parameters plus further temporal corrections. Because the navigation message is not decoded by the receiver, a RINEX file containing them was manually downloaded and imported into the system by using our own parsing function, called "*loadrinexn.m*".

2.2.2 Supplementary Assistance Data

While addressing the issue of improving position accuracy we may need to access additional data. IONEX files presented in the previous section fall inside this category.

2.3 Summary

By applying tracking algorithms we should be able to obtain more accurate values for code phase and carrier frequency. This should lead us to get more accurate positioning results. While implementing tracking algorithms for GPS snapshots we have to keep in mind the short length of the snapshot because this may cause convergence failure, therefore we have to find alternative algorithms.

Due to the fact that GPS signals are affected by several different errors which make different contributions, in order to obtain positions as precise as possible, these errors should be mitigated or at least minimized as much as possible. Table 2.1 contains a short description of the main error sources considered for correction. These errors are presented in the order of their priority: from large to smaller size.

Table 2.1: Error budget summary. All values presented here are rough estimations of their average contribution.

Error Source	Behavior	Magnitude
Ionosphere Delay	Dispersive	$< 25\,\mathrm{m}$
Troposphere Delay	Weather dependent	$< 5\,\mathrm{m}$
Relativistic Error	Predictable	$\sim \mathrm{m}$
Total Group Delay	Constant	$\sim \mathrm{dm}$

The algorithms used should be automatically provided with as much assistance data as possible, making the whole process transparent to the user.

Concerning data assistance, all mandatory variables are required:

- Time: Obtained from snapshot metadata
- Apriori location: Obtained from Doppler navigation algorithm
- Ephemeris: RINEX navigation files

Furthermore, IONEX files interfacing will be implemented as an additional assistance method to correct for ionosphere delays, as exposed in Section 2.1.2.1 .

Considering this, our overall priority list is:

1. Improve accuracy using tracking like techniques
2. Auto-assist snapshot time tag
3. Auto-assist ephemeris
4. Auto-assist a priori user coordinates
5. Auto-assist IONEX files and correct ionosphere delay
6. Correct troposphere delay
7. Correct relativistic error
8. Correct for total group delay

CHAPTER 3. DEVELOPMENT

This chapter goes deeper into the concepts identified in Section 2. A summary of the main highlights will be, again, presented right before the experimental results.

3.1 Improving Accuracy

In this first section of the Development chapter we will detail how signal tracking is performed. After, a description of the pseudo-range corrections applied will follow. This will include the correction of the ionosphere delay using TEC information from IONEX, troposphere delay correction through modeling and, lastly, performing weighted least-squares solutions using User Range Accuracy (URA) and other weighting criteria.

3.1.1 Signal Tracking

This subsection extends the reasoning behind our decision to apply tracking algorithms in order to improve position accuracy. The different ideas and concepts which have been implemented will be presented.

We decided to apply tracking algorithms in order to improve the position results by refining the acquisition results. Due to the fact that we are dealing with short snapshot cuts (11 ms) we are not dealing with traditional tracking. In order to generate pseudo-range measurements, traditional tracking implies continuously generating a replica code in the receiver, perform correlation with the incoming signal and, afterwards, decoding the navigation message from the GPS satellites. In the present project, due to short integration time, tracking convergence seemed difficult to obtain. Therefore we just want to correct the values obtained from acquisition, concentrating on the code phase. As a reminder, code phase is an independent time parameter of the PRN code, expressed in chips —portion of PRN waveform over the minimal time interval between code transitions [4]— which actually indicates where in the incoming signal the code corresponding to a certain PRN starts. As previously stated our main purpose was to correct the code phase values. The starting point for our algorithms was the tracking function (*tracking.m*) from the softGPS project, which is an attachment of the reference book [10].

We started with the idea that the carrier frequency obtained through acquisition was good enough for this application. Therefore, firstly we implemented just a simple DLL as depicted in Figure 2.4, aiming to retrieve the discriminator output and apply it as a correction to the code phase. The discriminator used is the normalized early minus late discriminator and it is characterized by the following expression:

$$D = \frac{1}{2}\frac{E - L}{E + L} \tag{3.1}$$

where:
D is the discriminator output
E is the early component, see equation (3.2)
L is the late component, see equation (3.3)

$$E = \sqrt{I_E^2 + Q_E^2} \tag{3.2}$$

$$L = \sqrt{I_L^2 + Q_L^2} \tag{3.3}$$

where:
I_E is the in-phase early component

Q_E is the quadrature early component
I_L is the in-phase late component
Q_L is the quadrature late component

As presented in [4] and [1] this discriminator has a performance which is independent of the PLL, using both in-phase and quadrature correlator outputs. This normalized non-coherent discriminator will produce in the ideal case, for a ± 0.5 correlator spacing, a true tracking error within ± 0.5 chips as seen in Figure 3.1.

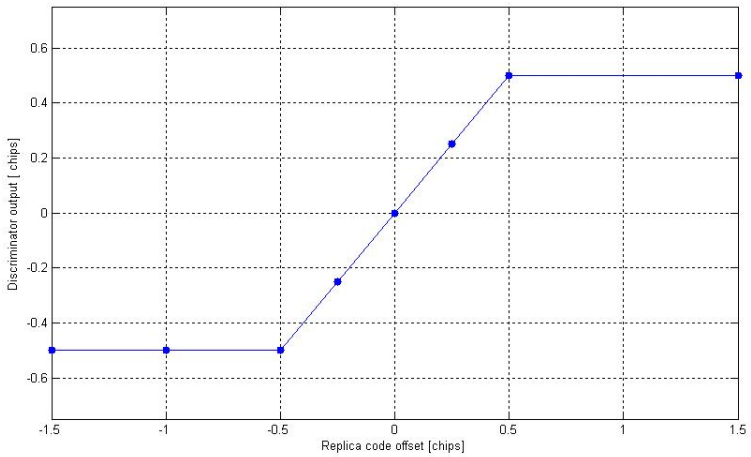


Figure 3.1: Discriminator output versus code error

In our first attempts to implement tracking algorithms we considered just 1 ms of the incoming signal. The main idea was to read the first millisecond of the incoming signal, generate the early, prompt and late local copies, perform correlation and, as long as the discriminator output was outside a defined threshold interval, restart the algorithm and repeat the process in the next millisecond. Generating the local copies involved generating a local C/A code, oversampling it with at a rate equal to the code frequency divided by the sampling frequency. The idea was that until convergence was achieved the discriminator outputs should be used to correct the code phase obtained through acquisition. While choosing the threshold value, we had to relate the discriminator output to a distance measurement. Therefore, knowing that 1 ms causes 300 km of bias and that in the same time interval 1024 chips are sent, it was clear to see that one chip of error causes approximately 300 m of error. After experimenting with this algorithm we realized that convergence was never achieved, so we had to find a way to fix this problem.

Our next approach was to use an algorithm very similar to the one described in [1]. This meant combining DLL and PLL tracking loops as shown in Figure 3.2.

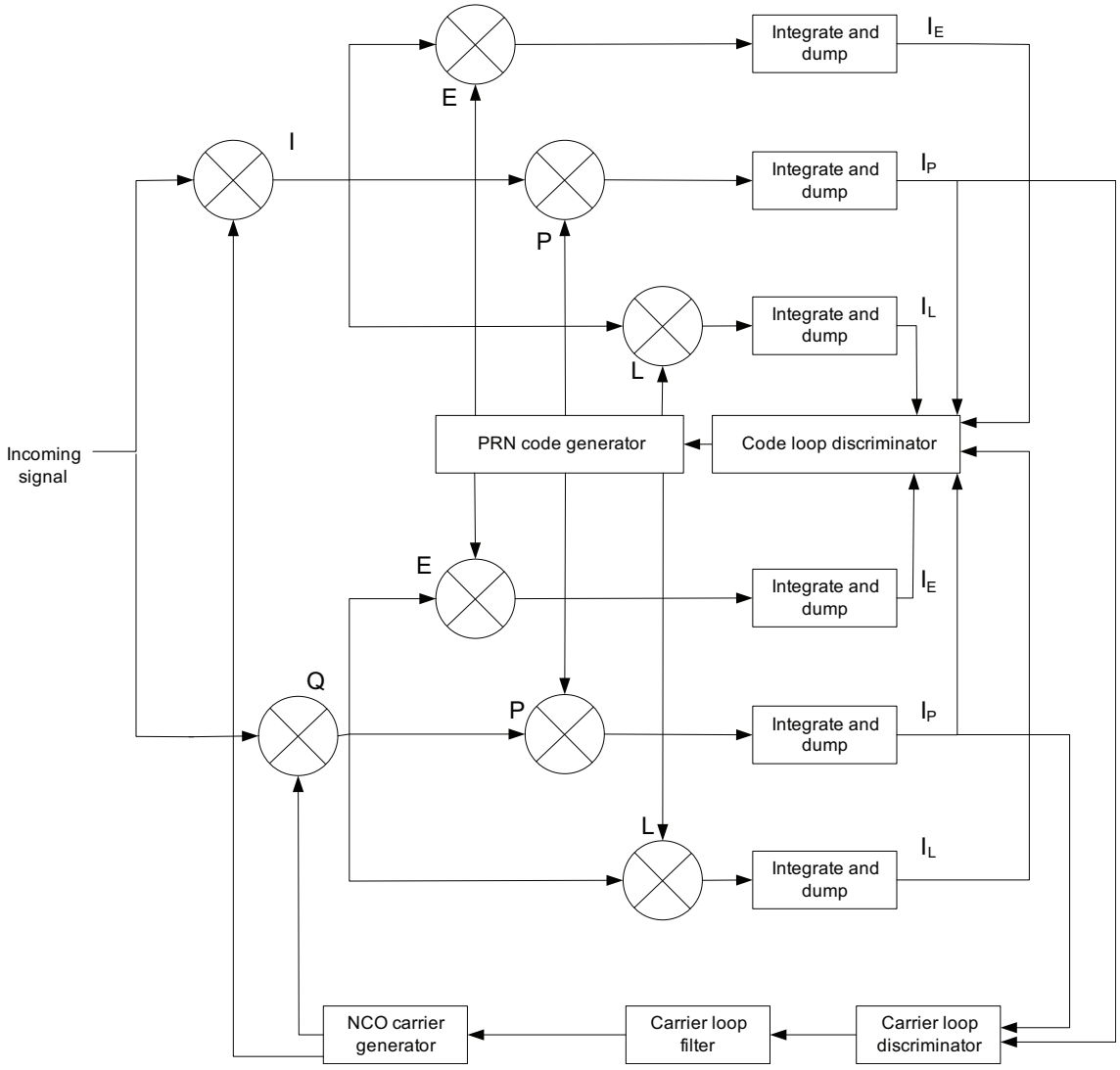


Figure 3.2: Block diagram of combined DLL and PLL tracking loops [1]

Using this method we also applied corrections to the carrier phase with the help of the PLL. In order to assure convergence of the algorithm, we decided to proceed in the same manner as in the softGPS and modify the code frequency by a value proportional to the filtered discriminator output. We were expecting that this would ensure better alignment of the incoming code with the local copy. By testing the algorithm we concluded that the obtained results were unsatisfactory. The algorithm yet again failed convergence. Trying to solve this we soon realized that while repeating the algorithm in only one millisecond, the discriminator value does not change significantly from one iteration to the other, therefore remaining fairly constant. At this point it was decided to form a residual from the difference between code errors in consecutive iterations, and compare this residual with the tracking threshold, but this proved to be unsuccessful as well.

Due to the fact that the discriminator output remains fairly constant while repeating the tracking algorithm for the same part of the signal, we decided to take into consideration the entire signal, treating each millisecond independently. The main idea was that we should define a common measuring point for all the channels which were to be tracked. This common point was defined relative to the channel with the minimum code phase. Therefore, we repeated the tracking algorithm for a number of times which is one millisecond shorter that the integration time (in our case the integration time is 7 ms [5]). In this way by the end of the tracking algorithm we wanted to obtain the code phase corresponding to the channel with the minimum code phase. All other measurements were done relative to this common point. Figure 3.3 shows a graphic interpretation of this algorithm. Due to very short integration time, the corrections applied to the code phase

were not significant. The tracking algorithm converges after a longer period of time —in some cases around 200 ms as shown in Figure 3.4—. We are dealing with only 7 ms, that is roughly 28 times less. The conclusion is that applying this algorithm is not suitable for our case.

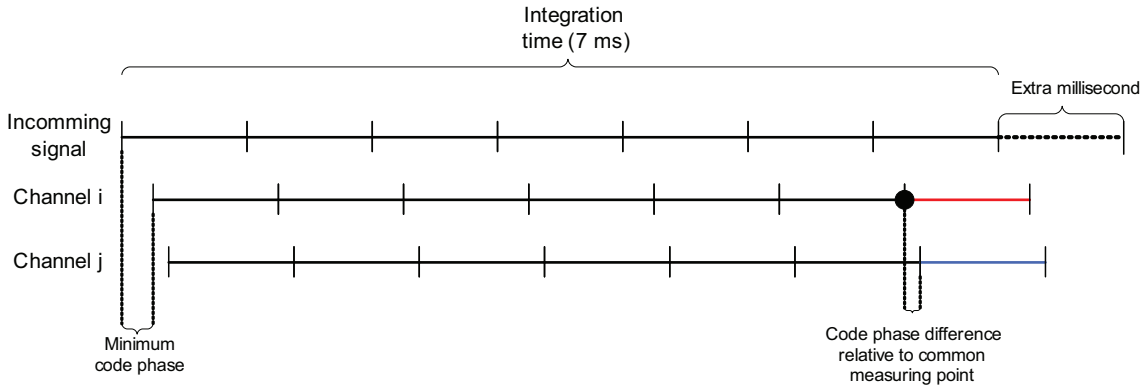


Figure 3.3: Code phase measurements relative to common measuring point. Channel i is considered the channel corresponding to the PRN with the minimal code phase

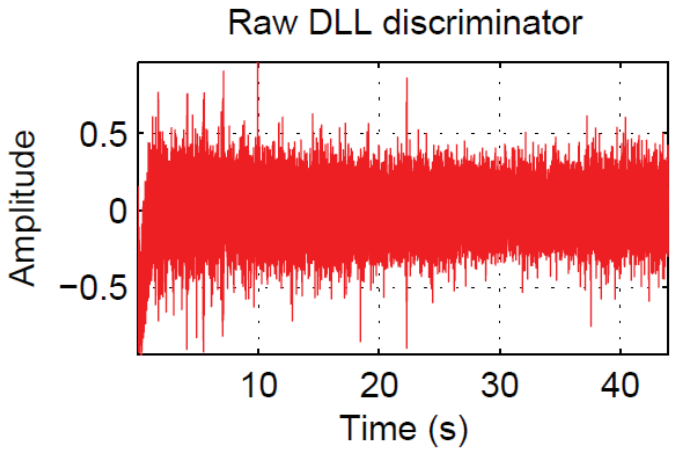


Figure 3.4: Raw DLL discriminator output

Figure 3.5 shows the relationship between the discriminator output and the true error it produces. As can be seen, unlike the ideal case which is shown in Figure 3.1, there is no direct correspondence between these two values. For example a discriminator value of -0.5 chips would not guarantee a true error of -0.5 chips. Analyzing the discriminator output we realized that the sign of the obtained value is more consistent, telling us if the replica code should be moved forward or backward. We continued by setting up a nominal value of 0.5 which represents the initial amount of how much the code should be moved, corresponding to the discriminator sign. By each iteration we divide the previously mentioned value by two. This should provide a fast way of aligning the codes. By this method we aimed to align the replica code with the incoming one, and after this is realized, use the corresponding code phase in order to obtain an improved position. Unfortunately no noticeable results were obtained.

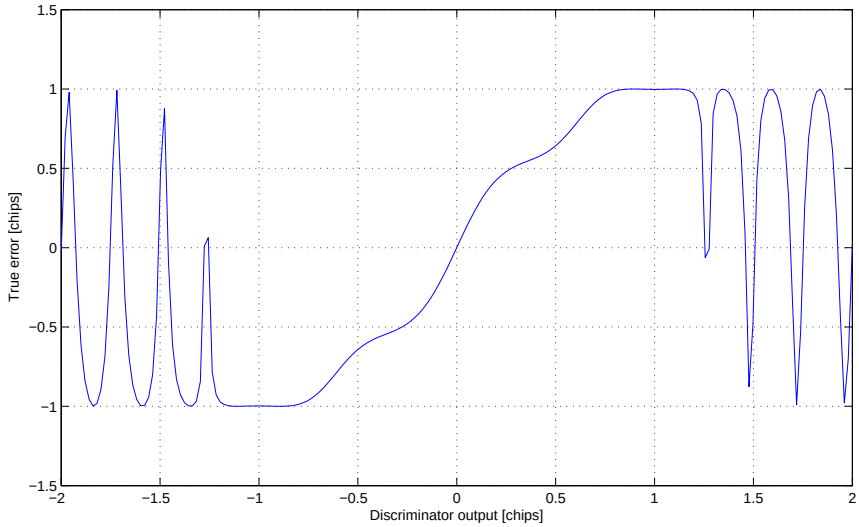


Figure 3.5: Discriminator output versus true error

In order to be able to correct the acquired code phase, we had to better understand the behavior of the discriminator output. As seen in Figure 3.4, this output is very noisy. Figure 3.6 shows the behavior of the raw discriminator output, observed for a time corresponding to the integration time (7 ms). As clearly seen the outputs vary in value as well as in sign. Therefore we tried to average the code error in order to obtain a more precise value. The results of this will be detailed in Section 4.2.1.

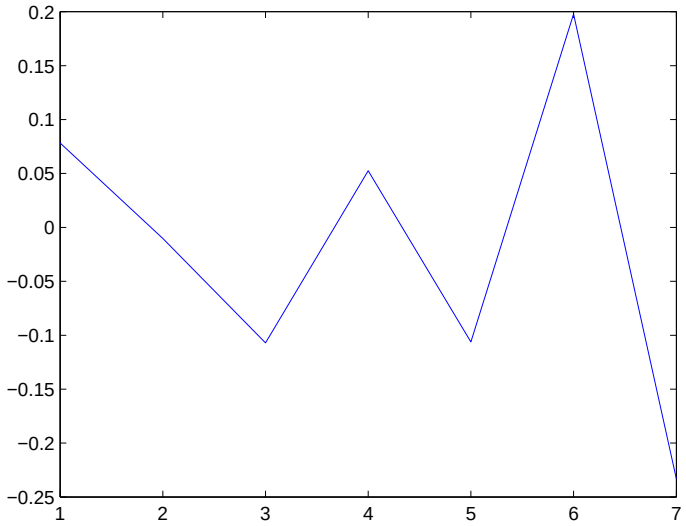


Figure 3.6: Raw DLL discriminator output observed for 7 ms

3.1.2 Pseudo-range Corrections

3.1.2.1 Ionosphere Delay - IONEX

In order to obtain a better precision of our position measurements we decided to apply ionospheric corrections while using the IONosphere Map Exchange Format (IONEX) Version 1. IONEX has

been created in order to satisfy the GPS world's need for a common data format which enables the exchange, comparison, or combination of Total Electron Content (TEC) maps [11]. This format, which is described more in detail in [11], supports 2-dimensional and 3-dimensional TEC maps and TEC Root Mean Square (RMS) values given in a geographic grid formed by constant increments in latitude and longitude, for a specific epoch . However, the IONEX file is also able to support different maps which correspond to different epochs. A 24 hours IONEX file can support a number of 13 TEC maps, therefore describing the ionosphere at an approximate interval of 1 hour and 51 minutes. The IONEX data files are created by combining measurements from a large number of dual-frequency receivers which are spread around the world in specific locations, combining measurements obtained from the International GPS Service for Geodynamics (IGS) as well as other external processing centers.

In order to correct for the ionospheric errors we first need to download the IONEX file, read it and then compute the ionospheric delay as depicted in Figure 3.7.

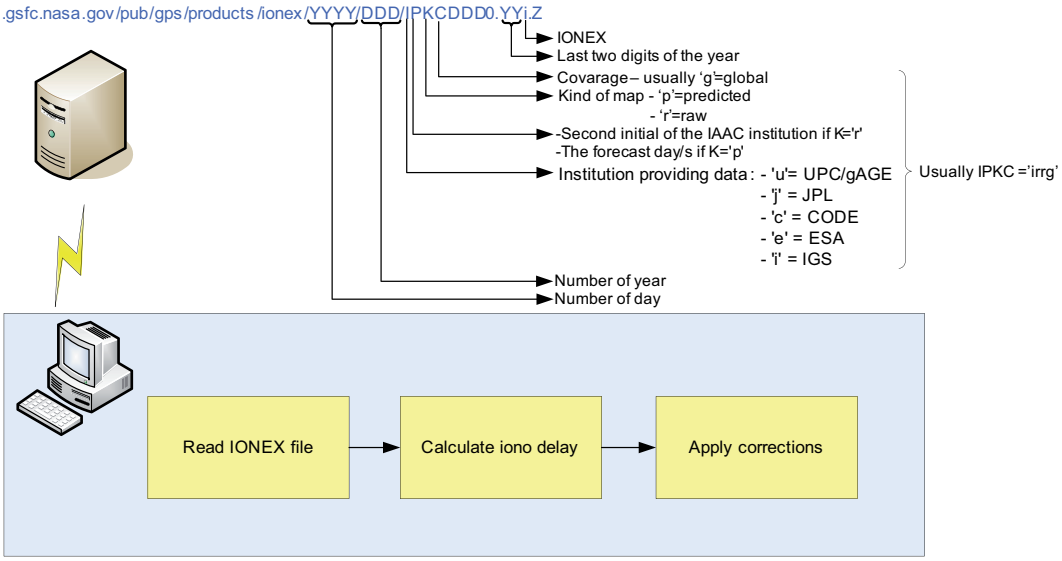


Figure 3.7: Basic tasks needed to solve for ionospheric delay

In order to calculate the correct TEC values we have to know the Ionospheric Piercing Point (IPP). The IPP is represented by the latitude and longitude of the point in which the line segment between the receiver and the satellite intersects with a parallel surface to WGS84 ellipsoid at an altitude of 350 km [9]. The method of calculating the IPP is the one used in the Klobuchar model and it is detailed in [3]. Figure 3.8 shows the basic principle where:

- O - Earth Center
- U - User location (ϕ_u, λ_u)
- IPP - Ionospheric piercing point $(\phi_{ipp}, \lambda_{ipp})$
- E - Elevation angle between user and satellite
- Ψ - Earth's central angle between user position and earth projection of IPP
- R - Earth radius
- H - IPP altitude = 350 km

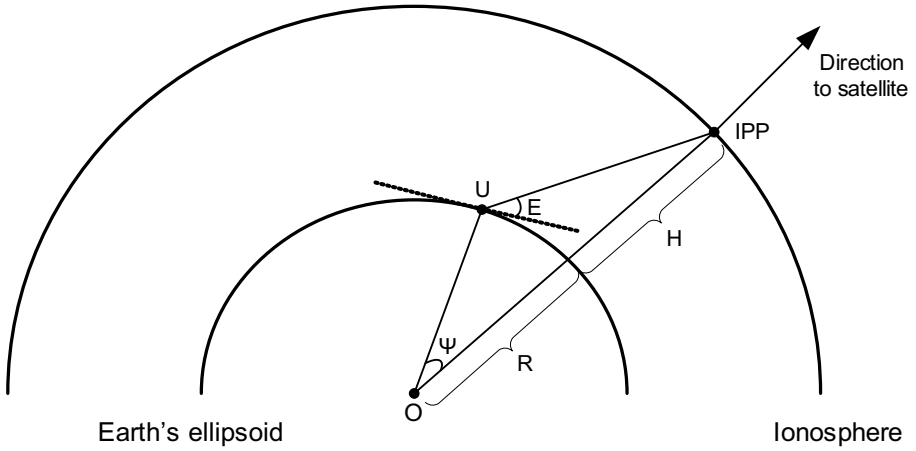


Figure 3.8: IPP computation [2]

From [3] we get:

$$\Psi = \frac{\pi}{2} - E - \sin^{-1}\left(\frac{R}{R+H}\cos E\right)_{\text{radians}} \tag{3.4}$$

$$\phi_{ipp} = \sin^{-1}\left(\sin\phi_u \cos\Psi_{ipp} + \cos\phi_u \sin\Psi_{ipp}\cos A\right)_{\text{radians}} \tag{3.5}$$

If the following set of conditions are fulfilled:

$$\begin{array}{llll} \phi_u > 70^\circ & and & tan\Psi_{ipp} cosA > tan(\pi/2 - \phi_u) & \text{or} \\ \phi_u > -70^\circ & and & -tan\Psi_{ipp} cosA > tan(\pi/2 - \phi_u) & \text{then :} \end{array}$$

$$\lambda_{ipp} = \lambda_u + \pi + \sin^{-1}\left(\frac{\sin\Phi_{ipp}\sin A}{\cos\phi_{ipp}}\right)_{\text{radians}} \tag{3.6}$$

else:

$$\lambda_{ipp} = \lambda_u + \sin^{-1}\left(\frac{\sin\Phi_{ipp}\sin A}{\cos\phi_{ipp}}\right)_{\text{radians}} \tag{3.7}$$

where:

ϕ_u, λ_u	is the user's latitude and longitude
$\phi_{ipp}, \lambda_{ipp}$	is the IPP latitude and longitude
A	is the azimuth angle between user and satellite measured clockwise positive from the true North

After determining the coordinates of the IPP we must determine the TEC in the place the satellite signal crosses the ionosphere. It is obvious that the signal will not always cross exactly over the grid points of the ionosphere map, therefore we have to interpolate between the nearest 4 coordinates with the known TEC values and the IPP coordinates previously determined. The interpolation algorithm is detailed in [11].

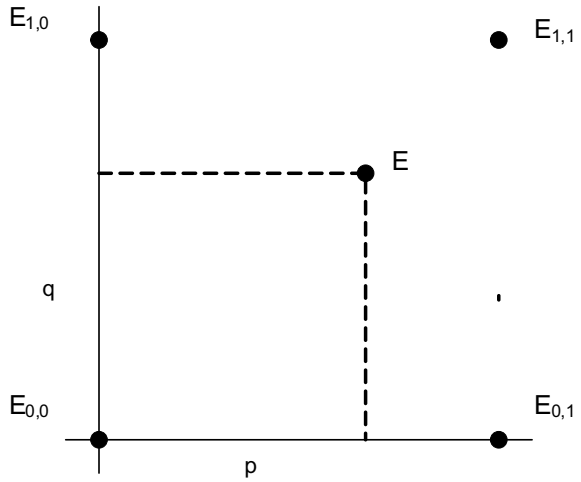


Figure 3.9: Bivariate interpolation using the 4 nearest TEC points [3]

$$E\left(\lambda_0 + p\Delta\lambda, \beta_0 + q\Delta\beta\right) = (1-p)\,(1-q)\,E_{0,0} + p\,(1-q)\,E_{0,1} + p\,(1-q)\,E_{1,0} + pqE_{1,1} \tag{3.8}$$

The TEC values are usually expressed as vertical components. Due to the fact that in most of the cases we are dealing with elevation angles, we must use an obliquity factor F_{pp} or mapping function. This factor will take into account the increase of the length of the signal path through the ionosphere [4].

$$F_{pp} = \left[1 - \left(\frac{R\cos E}{R+H}\right)^2\right]^{\frac{1}{2}} \tag{3.9}$$

3.1.2.2 Troposphere Delay - MOPS Model

Our desire to further improve accuracy made us pursue the idea of applying a tropospheric model to our measurements. This section will provide more details about the model and algorithms used. The main source was [2].

While dealing with tropospheric corrections we have to keep in mind that we are dealing with local phenomena, therefore the corrections are strongly based on the location of the user. Tropospheric models are based also on seasonal variations of some meteorological characteristics:

- **Pressure** [P(mbar)]= the force per unit area applied in a direction perpendicular to the surface of an object
- **Temperature** [T(K)]= a measurement of the average kinetic energy of the molecules in an object or system and can be measured with a thermometer or a calorimeter. It is a means of determining the internal energy contained within the system
- **Water vapor pressure** [e(mbar)]
- **Temperature lapse rate** [β(k/m)]= the rate of change of temperature with altitude
- **Water vapor lapse rate** [λ(dimensionless)]= the rate of change of water vapor with altitude

The values of these parameters are seasonal variation values as a function of latitude(ϕ) and day-of-year (D). Therefore, by using the general term Υ we will describe how these parameters are

obtained, as detailed in [2]:

$$\Upsilon(\phi, D) = \Upsilon_0(\phi) - \Delta\Upsilon(\phi)\cos\left(\frac{2\pi(D - D_{min})}{365.25}\right) \tag{3.10}$$

$D_{min} = 28$ for northern latitudes and $D_{min} = 211$ for southern latitudes. Υ_0 and $\Delta\Upsilon$ are the seasonal variation values as functions of latitude. If $|\phi| \leq 15°$ and $|\phi| \geq 75°$ than Υ_0 and $\Delta\Upsilon$ are taken directly from Tables 3.1 and 3.2. Else, their values are obtained as a result of linear interpolation between the values of the nearest latitudes $[\phi_i, \phi_{i+1}]$:

$$\Upsilon_0(\phi) = \Upsilon_0(\phi_i) + [\Upsilon_0(\phi_{i+1}) - \Upsilon_0(\phi_i)]\,\frac{\phi - \phi_i}{\phi_{i+1} - \phi_i} \tag{3.11}$$

$$\Delta\Upsilon(\phi) = \Delta\Upsilon(\phi_i) + [\Delta\Upsilon(\phi_{i+1}) - \Delta\Upsilon(\phi_i)]\,\frac{\phi - \phi_i}{\phi_{i+1} - \phi_i} \tag{3.12}$$

Table 3.1: Average meteorological parameters for tropospheric delay

Latitude (°)	P_0 (mbar)	T_0 (K)	e_0 (mbar)	β_0 (K/m)	λ_0
15° or less	1013.25	299.65	26.31	6.30e-3	2.77
30°	1017.25	294.15	21.79	6.05e-3	3.15
45°	1015.75	283.15	11.66	5.58e-3	2.57
60°	1011.75	272.15	6,78	5.39e-3	1.81
75° or more	1013.00	263.65	4.11	4.53e-3	1.55

Table 3.2: Seasonal variation of meteorological parameters for tropospheric delay

Latitude (°)	ΔP (mbar)	ΔT (K)	Δe (mbar)	Db (K/m)	Dl
15° or less	0.00	0.00	0.00	0.00e-3	0.00
30°	-3.75	7.00	8.85	0.25e-3	0.33
45°	-2,25	11.00	7.24	0.32e-3	0.46
60°	-1.75	15.00	5.36	0.81e-3	0.74
75° or more	-0.50	14.50	3.39	0.62e-3	0.30

We have applied the Minimum Operational Performance Standard (MOPS) developed by Radio Technical Commission for Aeronautics (RTCA). This model needs to receive as input day-of-year, satellite elevation angle, receiver height and latitude. The day-of-year and receiver latitude are used in order to choose the parameters from Tables 3.1 and 3.2. Elevation angle and receiver height are needed in order to compute the tropospheric delay estimate (TC_i). This delay, as presented in 2.1.2.2 is composed of a dry component and a wet component and it results from mapping these components by a mapping function $m(El_i)$ in order to obtain a processed slant delay according to signal path:

$$TC_i = -\left(d_{\text{dry}} + d_{\text{wet}}\right) m(El_i) \tag{3.13}$$

The two components $d_{\text{dry}}, d_{\text{wet}}$ are the troposphere's dry and wet components according to signal path and they are computed using the user height and estimates of meteorological parameters:

$$d_{\text{dry}} = z_{\text{dry}} \left(1 - \frac{\beta H}{T}\right)^{\frac{g}{R_d \beta}} \tag{3.14}$$

$$d_{\text{wet}} = z_{\text{wet}} \left(1 - \frac{\beta H}{T}\right)^{\frac{(\lambda+1)g}{R_d \beta} - 1} \tag{3.15}$$

where g= $9.80665\, m/s^2$, H is the receivers height above the mean sea level, $R_d = 287.054\,\text{J/kg/K}$ and z_{dry} and z_{wet} are the dry and wet components at zenith:

$$z_{\text{dry}} = \frac{10^{-6} k_1 R_d P}{g_m} \tag{3.16}$$

$$z_{\text{wet}} = \frac{10^{-6} k_2 R_d e}{[g_m(\lambda + 1) - \beta R_d]T} \tag{3.17}$$

where $k_1 = 77.604\,\text{K/mbar}$, $k_2 = 382000\,\text{K}^2\text{/mbar}$ and $g_m = 9.784\text{m/s}^2$.

The only thing that is still needed in order to calculate the tropospheric delay is the mapping function, which is a function of satellite elevation angle, being valid only for elevation angles greater that 5°:

$$m(El_i) = \frac{1.001}{\sqrt{0.002001 + \sin^2(El_i)}} \tag{3.18}$$

3.1.3 Weighted Least Squares

Once all pseudo-range measurements have been obtained and corrected, it is time to formulate the observation equations that will result in a fix. Summarizing, the observation equations form a system based on a linearization of the distance vectors (pseudo-range measurement) between the satellites and the user. Such a linear system can be summarized like:

$$\delta \mathbf{z} = \mathbf{H} \delta \mathbf{x} + \varepsilon \tag{3.19}$$

where:

$\mathbf{H}$	is a matrix of normalized geometric distances between user and satellites
$\delta \mathbf{z} = \mathbf{z} - \hat{\mathbf{z}}$	is the vector of a-priori pseudo-range measurement residuals
$\mathbf{z}$	is the vector of measured pseudo-ranges
$\hat{\mathbf{z}}$	is the vector of predicted geometric distances
ε	is the vector of measurement and linearization errors
$\delta \mathbf{x} = \begin{bmatrix} \delta_x \\ \delta_y \\ \delta_z \\ \delta_b \end{bmatrix}$	is the vector of updates to the a-priori state: x, y, z and b

Least Squares (LS) method is the traditional way to solve an over-determined system. The solution is:

$$\delta \mathbf{x} = \left(\mathbf{H}^T \mathbf{H}\right)^{-1} \mathbf{H}^T \delta \mathbf{z} \tag{3.20}$$

However, Weighted Least Squares (WLS), is an algebraic function that extends LS assigning a relative degree of "trust" to each one of the equations improving our estimation according to them. Thus, it allows us to refine the results taking into account additional information. The weight matrix W will be used in computing least-squares solution for the position:

$$\delta \mathbf{x} = \left(\mathbf{H}^T W \mathbf{H} \right)^{-1} \mathbf{H}^T W \delta \mathbf{z} \tag{3.21}$$

where W is a diagonal matrix:

$$W = \begin{bmatrix} w_1 & 0 & \dots & 0 \\ 0 & w_2 & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \dots & 0 & w_i \end{bmatrix} \tag{3.22}$$

w_i is the weight corresponding to satellite i.

It is up to the user to introduce the weights. A possible option is to select a weight based on the elevation of the satellite in the sky, as described in PhD Thesis [12], the reasoning behind this relies in the fact that code observations may deteriorate with decreasing satellite elevations, so their root mean square errors are higher. In that study, it was found that such dependance could be easily modeled through the following expression:

$$y = a_0 + a_1 exp^{(-el_i/el_0)} \tag{3.23}$$

y is the root mean square error in the code observations
a_0 and a_1 are constants defining the offset and exponential gain
el_0 is an reference elevation angle
el_i is the elevation of satellite i in the sky
All elevation angles are expressed in degrees

The value of y could be used to define the weight:

$$w_i = \frac{1}{y^2} \tag{3.24}$$

Some other common weighting criteria are described in the next paragraphs.

3.1.3.1 User Range Accuracy - URA

A usual option is to use the User Range Accuracy (URA) indicator as weight, described in [3]. As defined in [3] the User Range Accuracy (URA) is in fact a statistical indicator of the ranging errors which can be obtained with a specific satellite. URA can be obtained from the navigation message. URA indicators provide only information about the errors for which the Space and Control Segments are responsible. Therefore URA indexes do not contain any error information corresponding to the User Segment or to the transmission medium.

URA indeces therefore give information about space vehicle accuracy. They are contained in bits 13 to 16 of the third word in the first subframe in the navigation message. As it is described more in detail in [1], the navigation data contains a frame of 1500 bits. This frame is divided into 5 subframes, each 300 bits in length. Each of these subframes contain 10 30 bit words.

The URA index is an integer N ranging from 0 to 15 which is related with the range accuracy of the space vehicle as it is presented in Table 3.3.

Table 3.3: Relationship between URA index and URA of the satellite (taken from [3])

URA INDEX N	URA (meters)
0	0.00 $<$ URA $\leq$ 2.40
1	2.40 $<$ URA $\leq$ 3.40
2	3.40 $<$ URA $\leq$ 4.85
3	4.85 $<$ URA $\leq$ 6.85
4	6.85 $<$ URA $\leq$ 9.65
5	9.65 $<$ URA $\leq$ 13.65
6	13.65 $<$ URA $\leq$ 24.00
7	24.00 $<$ URA $\leq$ 48.00
8	48.00 $<$ URA $\leq$ 96.00
9	96.00 $<$ URA $\leq$ 192.00
10	192.00 $<$ URA $\leq$ 384.00
11	384.00 $<$ URA $\leq$ 768.00
12	768.00 $<$ URA $\leq$ 1536.00
13	1536.00 $<$ URA $\leq$ 3072.00
14	3072.00 $<$ URA $\leq$ 6144.00
15	6144.00 $<$ URA not available

The weights are computed in relationship with the nominal URA values, URA_{nom} as follows:

$$URA_{nom} = \begin{cases} 2^{(1+N/2)} & \text{for} \quad N \leq 6 \\ 2^{(N-2)} & \text{for} \quad 6 < N \leq 15 \end{cases} \tag{3.25}$$

If the value of the URA index N equals 15 it means that no accuracy prediction is available and users are advised to use the satellite data at their own risk. For $N = 1, 3$ and 5, URA_{nom} values should be rounded to $2.8, 5.7$ and 11.3 meters respectively.

3.1.3.2 SBAS Protection Levels

In Satellite Based Augmentation Systems (SBAS) different important protection levels are computed as a quality control for each satellite. Appendix J of the document [2], so-called DO-229C, defines multi-criteria algorithms for the computation of these protection levels, or weights. Such procedures are used by Wide Area Augmentation Systems (WAAS) airborne equipment for en route through precision approach mode (landing). Although the computation of most of the components of the weights relies on the additional information supplied by the SBAS there are several components which could be applied to any standalone GPS receiver. These components are:

- $\sigma^2_{i,\text{tropo}} = (\sigma_{TVE} m(E))^2$. A variance model of the residual error for the tropospheric delay formed by $m(E)$ in equation (3.18), and σ_{TVE}, the tropospheric vertical error. This value is set to a constant value of 0.12.
- $\sigma^2_{i,\text{multipath}} = 0.13 + 0.53 \exp^{(-el_i/10)}$. A variance model of the multipath error, clearly based on the previously presented equation.

Considering these components together with URA, we have created our own "custom" weighting system:

$$\sigma_i^2 = \sigma_{i,URA}^2 + \sigma_{i,\text{tropo}}^2 + \sigma_{i,\text{multipath}}^2 \tag{3.26}$$

3.2 Auto-Assistance

This section continues to describe the way assisted data is obtained, either from the Internet or through algebraic methods such as the Doppler positioning algorithm (assisting a priori coordinates) or the Time of Week algorithm (assisting a snapshot time to those snapshots not properly time-tagged by the operating system) we have invented.

In order to have an application which can be used by any kind of user (not only dedicated to professionals) we want the user to have a minimum number of tasks to do in order to get positions of his digital photographs. This is the reason why we wanted to provide our algorithms with more assisted data.

3.2.1 Assisted Time

Our first report [5] about snapshot techniques, stated that the convergence strength towards a wrong assisted time allows a bias of $\pm 100\,\text{s}$. Therefore we assumed that the internal camera clock could be re-synchronized with the Universal Time Coordinate (UTC) each time the camera is connected to a laptop, computer or, alternatively, coordinated with GPS Time (GPST) by decoding the Time of Week (TOW) from time to time. Therefore, the camera shall be able to attach a time tag to each snapshot, which would be accurate enough to provide an initial time for the positioning algorithms to converge. Ignoring the assumption that the camera has the means to assist time information, the front-end itself, in fact, does not provide any other data rather than IF samples.

In our case, these samples are stored into a file under Windows OS (Operating System). In general, any file within an OS has different metadata (information about the information contained into the file) or associated attributes describing its content. In Windows OS, for instance, regardless which type of file systems — New Technology File System (NTFS) or any File Allocation Table (FAT) version— is used, there are a minimum of metadata fields or details of a file: Name (string), Size (in Bytes), Type (typically indicated through the extension of the file and it associates the content with a program or application able to process it), Creation Time, Modification Time and Access Time. All these time attributes are set using the internal CPU clock of the laptop/computer. At the same time, this clock uses to be automatically synchronized with a time server of its domain (network of computers) or another time server on the Internet. Therefore, unless the computer is isolated, all files are automatically time tagged during their creation at UTC.

Until now, time assistance has been manually performed by introducing creation time of the snapshot file into the positioning algorithms. Along this project, we have programmed the automatization of this process as a new feature. To do so, a piece of C code, which uses standard Windows libraries for this purpose, has been pre-compiled into a MEX-file (Matlab EXternal file) so that the function can be invoked from MATLAB's workspace and retrieve the creation time from OS metadata. As a result, this approximation relies on the correct time synchronization of the CPU clock.

However, in the event of a non-synchronized CPU clock, such procedure will not work. In the previous project [5], there was a snapshot taken in Spain, identified as "S2C", (a table with all snapshots and their IDs is provided in appendix A. Many references to this table will be done during Section 3.2.1.1) that never provided satisfactory results. Although we suspected it could be due to a wrong time-tagging of the file, we had no means to demonstrate that, so we concluded claiming that it might be due to some kind of hardware malfunction. This problem led us to the creation of an algorithm able to estimate TW (Time of Week) of a snapshot. As a result, it was possible to obtain satisfactory results using the estimated time, hence demonstrating that divergence was caused by wrong time tagging due to CPU clock isolation instead of an error in our

code. Next section describes the algorithm used to estimate TW of a snapshot.

3.2.1.1 TOW Estimation

This section describes an algorithm for estimating the time instant at which a given snapshot was taken. Its inputs are date, ephemeris and location of the snapshot, as well as the snapshot itself. The output is a time estimate expressed as TW with an accuracy of few seconds.

Firstly, it is important to remark that this algorithm only works if the date and the user coordinates are known. It might seem a nonsense to program an algorithm which requires user coordinates as an input, whereas this is typically our unknown, but, in our case, it was useful to recover TOW of snapshot "S2C". This snapshot was taken in a computer running with a non-synchronized clock because, for safety reasons, it was isolated from any network. However, this problem lead to the creation of this algorithm, which is, at least, interesting from a theoretical point of view.

Basically the algorithm performs three steps:

1. Find the time interval where the acquired satellites best fit with line-of-sight interval.
2. Compute coarse-time algorithm within that time interval, ruling out those time instances which are resulting in divergence or "big" distance residuals.
3. Find the residuals' time trend-line and differentiate it in order to compute TOW where distance residual is minimum.

In step 2, "big" means a distance residual above 1 km between the position computed through coarse-time algorithm and the actual receiver position. In practice, step 1 and 2 are broken down into two search routines: coarse and fine. Their only difference is the time step they use: whereas fine searches always use a 1 s time step, coarse time steps are set to optimize execution time of the function.

If our aim is to find the time interval where line-of-sight satellites best fit with acquired ones, we shall set up a coarse time step according to constellation architecture and dynamics. Thus, at this point, it is worth to refresh general characteristics of GPS orbits.

Currently, the GPS constellation is formed by 32 active satellites in 6 equally spaced orbital planes, each rotated 60 ° relative to the preceding plane . Satellites are not equally spaced within the orbital planes, and there are phase offsets between planes to achieve improved Geometric Dilution of Precision (GDOP) characteristics of the constellation [4]. All orbital planes are inclined by 55 °. Concerning dynamics, the same satellites repeat to be at line-of-sight every sideral day for a static observer, so the period is half a sideral day. For further details on constellation design aiming for continuous whole-Earth coverage refer to [13]. As a result, in order to ensure at least 4 satellites in the sky, the practical constraint for coverage of the GPS constellation was set to a minimum of sixfold coverage above 5° minimum elevation angle [4].

Seeking for an optimum coarse time step, one shall consider that even if the algorithm is not fine enough to hit the exact satellite combination after acquisition, at least, it shall ensure to match most of them. Furthermore, note that acquisition might provide false alarms, acquire satellites which are not in view, or miss some satellite(s) in line-of-sight. Thus, while maximizing the number of coincident satellites is important to drop the probability of repetition of any satellite combination.

Roughly, the duration of any combination can take between 0 s and half of a sidereal day. Common sense tells that the duration of a given satellite combination decreases inversely to the amount of satellites conforming it. In case we decided to set the coarse time step just considering this fact, in the event of 15 acquired satellites, the time step would have been so close to the fine one that its purpose would have been ruined, giving rise to large execution times. Therefore, the logical decision is to select a time step according to the duration of most of the cases.

An initial approximation to this value was thought using combinatorics. According to this constellation settings, one expects that less than half of the constellation is visible from the Earth surface. The maximum number of potential combinations in a given time instant can be calculated by the binomial coefficient:

$$\text{Number of satellite combinations} = C_{n,k} = \binom{n}{k} = \frac{n!}{(n-k)!k!} \tag{3.27}$$

where:

n is the number of line-of-sight satellites

k is the grouping number of satellites forming a combination.

From the binomial function, it arises that the maximum number of satellite combinations take place when satellites are grouped in half of the visible satellites ($k = n/2$). Thus, theoretically, the maximum number of combinations appears when $n = 16$ and $k = 8$, even though it is, in practice, impossible to acquire 16 SVs using a reasonable elevation mask. After this, it was computed from simulations during a complete day, that the average duration of any given $k = 8$ combination is around 25 minutes. However, using a mean value we are not covering all $k = 8$ combinations and it is fortunate that $k > 8$ will result in smaller duration of combination.

Eventually, a better solution came up through simulations of the visible satellites along a whole week. Results of these simulations for different user locations are presented in Table 3.4 providing practical information on the duration of combinations.

At the Equator and low latitudes, we observe that there are between 7 and 15 satellites in view. The mean number of visible satellites around the globe is close to 9, which is another argument to discard the first approximation.

In conclusion, coarse-time step was set to 1 minute in order to ensure getting at least 95% of the possible combinations.

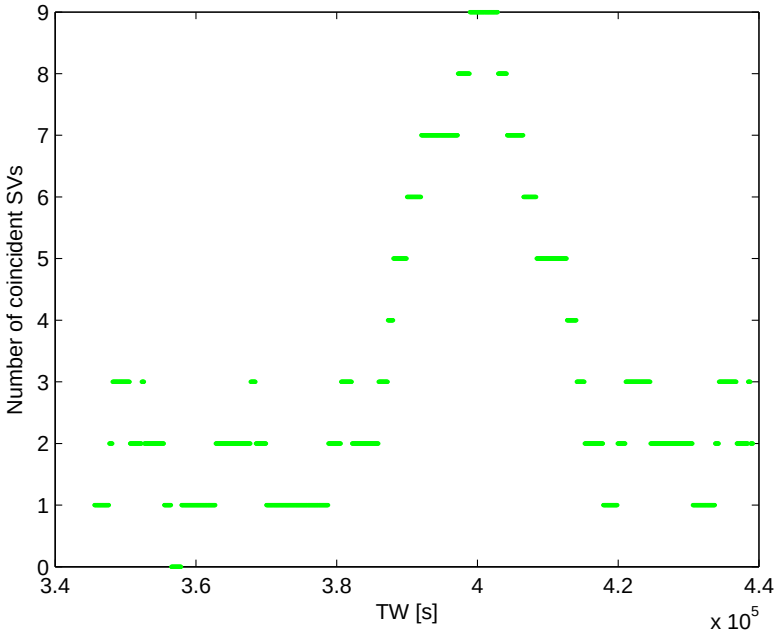


Figure 3.10: Number of coincident satellites along day. Results of coarse search for snapshot "2C"

After the time interval in step 1 has been refined, a second search within it starts in step 2. This time we are taking advantage of the convergence properties of the coarse-time algorithm described

Table 3.4: Simulation of GPS satellite combinations along one week at different user locations. Time resolution set to 1 s, producing 604, 800 sky computations for each user origin. 10° elevation mask angle. λ is latitude, φ is longitude, μ is mean, σ is standard deviation and 2σ represents 95th percentile, the value of the variable **above** which 95% of observations fall .

User Origin		Combination Duration			Number of SVs		
λ	φ	μ	σ	2σ (95 %)	min	max	μ
0°	0°	2774.3 s	1840.9 s	165.4 s	7	15	9.5
	60°	2774.3 s	2047.7 s	326.6 s	8	13	10.3
	120°	2774.3 s	2279.9 s	428.2 s	7	14	9.9
	180°	2749.1 s	1853.1 s	158.0 s	7	14	9.5
	240°	2774.3 s	2032.0 s	325.2 s	8	13	10.3
	300°	2774.3 s	2279.6 s	421.6 s	7	14	9.9
15°	0°	2371.8 s	1485.0 s	477.0 s	7	13	9.0
	60°	2129.6 s	1717.4 s	146.8 s	6	13	9.4
	120°	2256.7 s	1568.8 s	81.2 s	7	13	9.0
	180°	2371.8 s	1483.6 s	471.0 s	7	12	9.0
	240°	2137.1 s	1718.6 s	139.1 s	6	13	9.4
	300°	2265.2 s	1567.6 s	76.8 s	7	13	9.1
. 30°	0°	2273.7 s	1798.1 s	298.6 s	6	12	8.4
	60°	2009.3 s	1670.1 s	276.5 s	6	13	8.8
	120°	2183.3 s	1679.2 s	312.3 s	6	12	8.5
	180°	2273.7 s	1794.6 s	306.2 s	6	12	8.4
	240°	2002.6 s	1676.2 s	276.0 s	6	13	8.8
	300°	2167.7 s	1683.8 s	311.7 s	6	12	8.5
45°	0°	1957.3 s	1714.4 s	76.6 s	6	11	8.3
	60°	1872.4 s	1320.2 s	186.6 s	6	12	8.7
	120°	1901.9 s	1520.3 s	96.6 s	6	11	8.4
	180°	1957.3 s	1707.6 s	69.4 s	6	11	8.3
	240°	1872.4 s	1322.2 s	189.9 s	6	12	8.7
	300°	1913.9 s	1522.0 s	**66.5 s**	6	11	8.4
60°	0°	1460.9 s	1465.8 s	128.2 s	5	12	9.1
	60°	1390.3 s	1167.3 s	173.2 s	6	13	9.1
	120°	1413.1 s	1168.2 s	214.9 s	6	14	9.1
	180°	1460.9 s	1465.8 s	121.2 s	5	12	9.0
	240°	1387.2 s	1168.7 s	171.6 s	6	13	9.1
	300°	1409.8 s	1166.6 s	215.7 s	6	14	9.1
75°	0°	1377.7 s	1173.5 s	70.7 s	6	13	10.2
	60°	1384.0 s	1108.9 s	84.1 s	8	13	10.0
	120°	1387.2 s	1077.3 s	161.4 s	8	13	10.0
	180°	1384.0 s	1174.6 s	73.9 s	6	13	10.1
	240°	1387.2 s	1111.2 s	85.8 s	8	13	10.0
	300°	1384.0 s	1078.5 s	166.4 s	8	13	10.1
90°	-	1390.3 s	1044.0 s	112.2 s	9	12	10.2

in appendix A. Experiments showed that the algorithm could converge even with a time bias of ±100 s, which is consistent with theoretical results described in [14]; about ±90 s. Hence, 90 s coarse-time step was selected in step 2 to cope with the most restrictive of both cases.

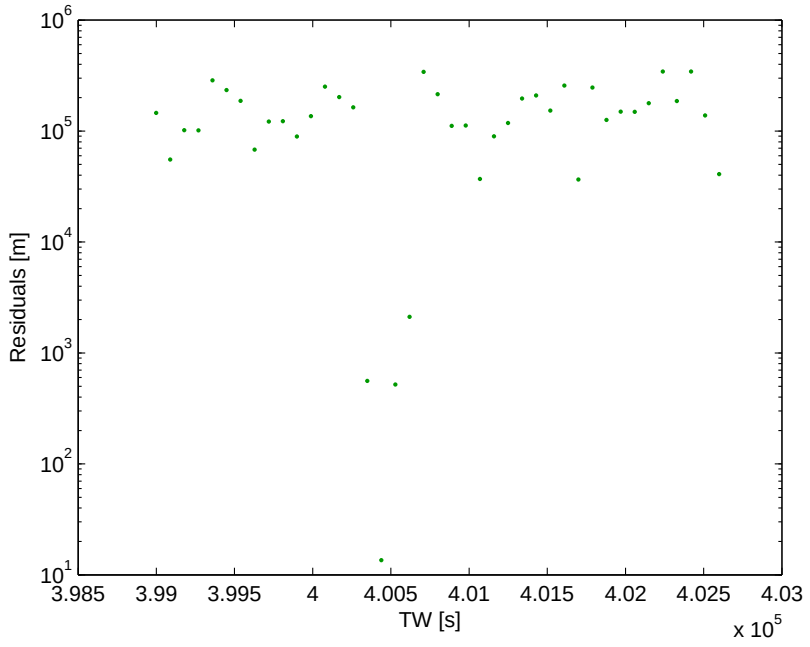


Figure 3.11: Distance residuals along day. Results of coarse time step for snapshot "2C".

Once again, the time interval is refined ending up with a set of time-distance samples like the ones in Figure 3.11 or Figure 3.12, creating a second order like function.

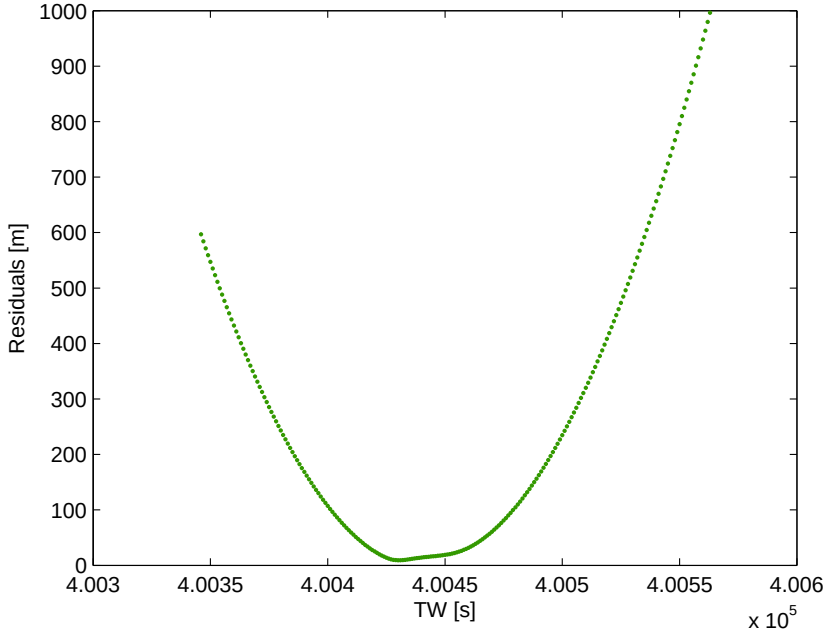


Figure 3.12: Distance residuals within fine time interval, again, for snapshot "S2C"

Using these samples, one is able to compute the coefficients of the second order polynomial obtain-

ing the trendline that best fits in terms of least square error. Last step is to derive such expression and compute the time instant where distance residuals are minimum.

The result for snapshot "S2C" was $400,435\,\mathrm{s}$. All results for "S2C" were computed using this estimation as TW-tag. In addition, accuracy of the algorithm has been tested with the rest of the snapshots, see Table 3.5. Theoretically, the error of the algorithm depends on the coarse-time algorithm itself and the accuracy of the user coordinates.

Table 3.5: TW estimation results. "S3D" did not converge due to lack of line-of-sight satellites

ID	TW-tag	Estimated TW	Error
S1D	216912s	216918s	+6s
S2C	NaN	400435 s	NaN
S3D	315999 s	NaN	NaN
S4G	391423 s	391389 s	-34s
S5G	122591 s	122582 s	-9s

The accuracy of the algorithm depends on the coarse-time algorithm itself and the accuracy of the user coordinates used for the computation. Figure 3.13 illustrated the complete flow diagram of the algorithm.

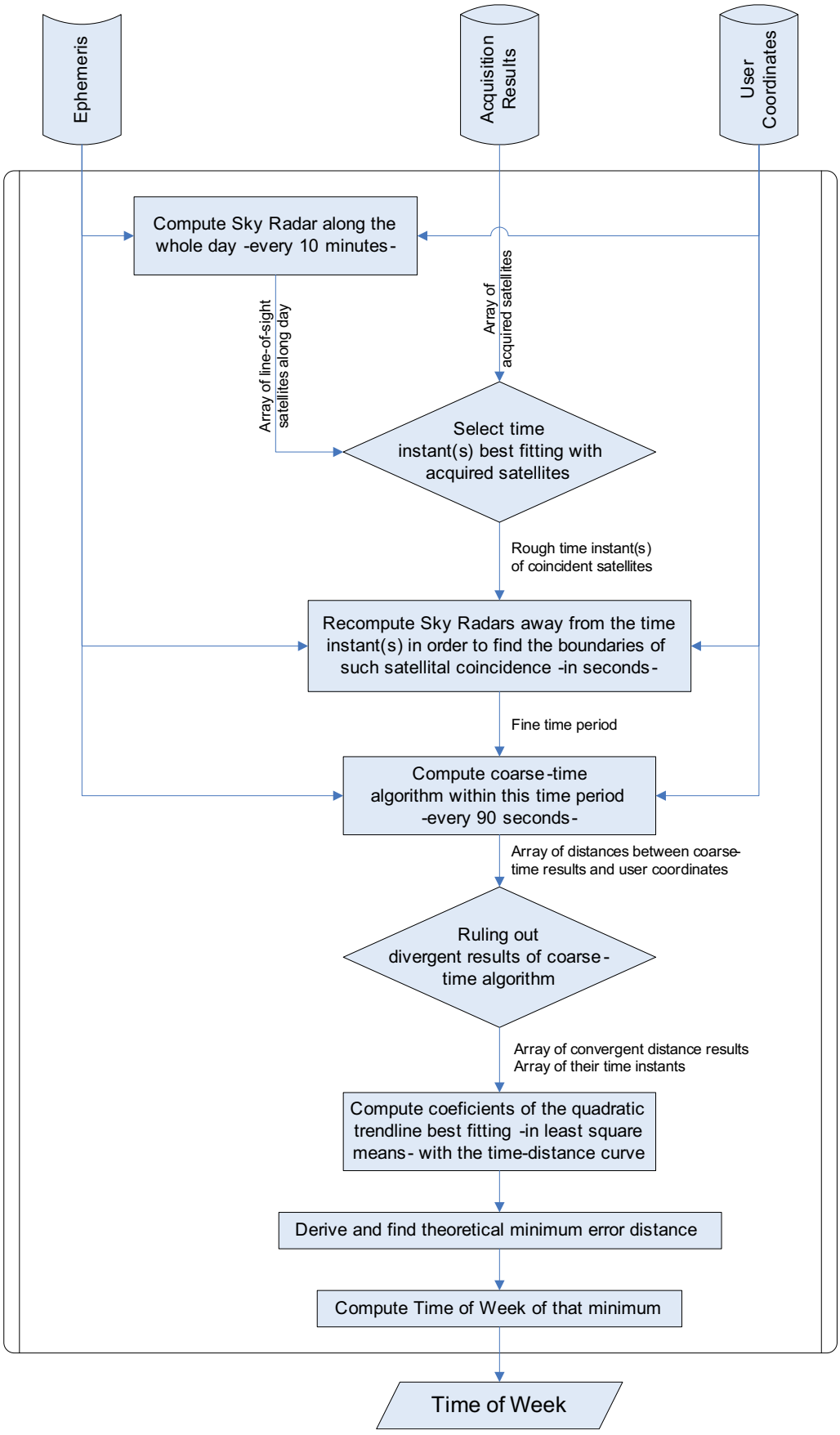


Figure 3.13: Flow diagram of TW computation using ephemeris and user coordinates

3.2.2 Ephemerides

Ephemerides contain information about Keplerian elements plus corrections for the Earth's gravitational field and other forces [9], providing time dependent, accurate satellite orbit values. In a classic GPS hardware receiver, obtaining ephemerides is only possible after tracking. Due to the fact that snapshot techniques do not perform tracking, ephemeris have to be supplied by other means.

In the code presented with [5], ephemeris were manually assisted as well as time. In practice, there are several file formats designed to broadcast information about GPS satellites orbits. Perhaps, the most popular ones are:

- YUMA almanac: Contains the six basic Keplerian elements describing orbits in the long-term. It is not accurate.
- RINEX navigation files: Collection of ephemeris broadcasted by the constellation within a given period of time. For 24 h RINEX navigation files, for instance, ephemerides for the different satellites are updated, at least, every 4 h. Inside this 2 to 4 h period, ephemerides provide best fit with the real satellite trajectory thanks to some additional correction parameters applied on top of the almanac. Ephemerides provide satellite positioning with sub-meter accuracy.

From the beginning, 24 h broadcast RINEX navigation files where selected for being the most accurate and widely used standard files, although suffering from very low update cadence (daily). They are available from public databases such as the Interntional GNSS Service (IGS) or other public entities offering GPS data services. Typically, the contents of these databases are automatically sorted by date and all files are named following a known pattern. Taking advantage of this fact, and assuming that the snapshot date is known, it was relatively easy to implement an automatic assistance of ephemeris, releasing the user of this task.

In our particular application, all navigation files are retrieved from CDDIS (Crustal Dynamics Data Information System) database. There files are sorted using a tree like structure of folders according to: `ftp://cddis.gsfc.nasa.gov/pub/gps/data/daily/YYYY/brdc/brdcDDD0.YYn.Z`, where "YYYY" is 4 digit number of year, "DDD" is 3 digit day number (0 padding in the left hand side might be required) "YY" is 2 last digits of the year and "n" indicates navigation file. One limitation of this solution might be that the user has to wait until the files are published (after 24 h in the worst case) before being able to compute positioning of the snapshot. Nevertheless, it is possible to find hourly data files also using RINEX format and it has been considered a good academical exercise to use them and get used to their structure.

Eventually, only a priori position is remaining to be manually introduced as an input to the algorithm.

3.2.3 Computing First Fix from Doppler Measurements

This algorithm is intended to solve for a position first fix using instant Doppler measurements. In [5] the user should have introduced an initial position manually in order to solve for its position with the coarse time navigation algorithm. As stated in 1 one of our purposes is to make the application as transparent to the user as possible. This is why we have chosen to apply Doppler measurements for computing an initial position.

Firstly we had to investigate if the Doppler positioning is able to provide us with a position within the accuracy range that is needed by the other algorithms. In [5] it was also shown that this initial position can vary in a range of ± 150 km without affecting the capacity of the coarse time algorithm to converge and provide an position accurate within $20 - 50$ m. Reference [14] states that the purpose of the Doppler navigation is to provide a first fix better than 150 km. Therefore we decided that this method is suitable for our application.

We pursued to implement the algorithm presented in [14] by first starting from the linear navigation equation which relates pseudo-ranges to the state update, already presented in (3.19). In order to have Doppler measurements we must differentiate it with respect to time. Therefore we get (bold characters represent vectors):

$$\frac{\partial \delta \mathbf{z}}{\partial t} = \frac{\partial (\mathbf{H} \delta \mathbf{x})}{\partial t} + \varepsilon' \tag{3.28}$$

$$\frac{\partial (\mathbf{z} - \hat{\mathbf{z}})}{\partial t} = \frac{\partial (\mathbf{H} \delta \mathbf{x})}{\partial t} + \varepsilon' \tag{3.29}$$

$$\frac{\partial \mathbf{z}}{\partial t} - \frac{\partial \hat{\mathbf{z}}}{\partial t} = \mathbf{H} \frac{\partial (\delta \mathbf{x})}{\partial t} + \frac{\partial (\mathbf{H})}{\partial t} \delta \mathbf{x} + \varepsilon' \tag{3.30}$$

The left side is identified as the vector of a priori Doppler measurement residuals $\delta \mathbf{y} = \mathbf{y} - \hat{\mathbf{y}}$. We continue with the next notations:

$\mathbf{y} = \partial \mathbf{z} / \partial t$	is the vector of measured Doppler
$\hat{\mathbf{y}} = \partial \hat{\mathbf{z}} / \partial t$	is the vector of predicted Doppler based on satellite-user motion

In order to be able to follow the equations more easily we consider the two terms in the right side separately. Therefore the first term will be:

$$\mathbf{H} \frac{\partial (\delta \mathbf{x})}{\partial t} = \mathbf{H} \frac{\partial}{\partial t} \begin{bmatrix} \delta_x \\ \delta_y \\ \delta_z \\ \delta_b \end{bmatrix} = \mathbf{H} \begin{bmatrix} \delta_{x'} \\ \delta_{y'} \\ \delta_{z'} \\ \delta_{b'} \end{bmatrix} = \begin{bmatrix} -\mathbf{e}^1 & 1 \\ \vdots & \vdots \\ -\mathbf{e}^i & 1 \end{bmatrix} \begin{bmatrix} \delta_{x'} \\ \delta_{y'} \\ \delta_{z'} \\ \delta_{b'} \end{bmatrix} \tag{3.31}$$

where:

$\delta_{x'}, \delta_{y'}$ and $\delta_{z'}$	represent the updates to the a priori receiver velocity states.
$\delta_{b'}$	is the update to the a priori frequency state.

The second right hand term of equation 3.30 will provide us with the necessary relationship between position and Doppler measurements:

$$\frac{\partial \mathbf{H}}{\partial t} \delta \mathbf{x} = \frac{\partial}{\partial t} \begin{bmatrix} -\mathbf{e}^1 & 1 \\ \vdots & \vdots \\ -\mathbf{e}^i & 1 \end{bmatrix} \begin{bmatrix} \delta_x \\ \delta_y \\ \delta_z \\ \delta_b \end{bmatrix} = \begin{bmatrix} \frac{-\partial \mathbf{e}^1}{\partial t} & 0 \\ \vdots & \vdots \\ \frac{-\partial \mathbf{e}^i}{\partial t} & 0 \end{bmatrix} \begin{bmatrix} \delta_x \\ \delta_y \\ \delta_z \\ \delta_b \end{bmatrix} = \begin{bmatrix} \frac{-\partial \mathbf{e}^1}{\partial t} \\ \vdots \\ \frac{-\partial \mathbf{e}^i}{\partial t} \end{bmatrix} \begin{bmatrix} \delta_x \\ \delta_y \\ \delta_z \end{bmatrix} \tag{3.32}$$

$\mathbf{e}^i = [{e_x}^i; {e_y}^i; {e_z}^i]$ is the unit vector in the direction from the receiver to satellite i. Therefore we must find an explicit expression for the vector $-\partial \mathbf{e}^i / \partial t$:

$$\frac{\partial \mathbf{e}^i}{\partial t} = \frac{\partial}{\partial t} \left(\frac{\mathbf{x}^i - \mathbf{x}_0}{|\mathbf{x}^i - \mathbf{x}_0|} \right) \tag{3.33}$$

where:

$\mathbf{x}^i$	is the position of satellite i
$\mathbf{x}_0$	is the apriori position of the receiver

Solving (3.33) becomes:

$$\begin{aligned}\frac{\partial \mathbf{e}^i}{\partial t} &= \left(\frac{\partial \mathbf{x}^i - \mathbf{x}_0}{\partial t}.|\mathbf{x}^i - \mathbf{x}_0| - \left(\mathbf{x}^i - \mathbf{x}_0\right).\frac{\partial |\mathbf{x}^i - \mathbf{x}_0|}{\partial t}\right)\frac{1}{|\mathbf{x}^i - \mathbf{x}_0|^2} \\ &= \left(\frac{\partial\left(\mathbf{x}^i\right)}{\partial t}.|\mathbf{x}^i - \mathbf{x}_0| - \left(\mathbf{x}^i - \mathbf{x}_0\right).\left(\mathbf{e}^i \bullet \mathbf{v}^i\right)\right)\frac{1}{|\mathbf{x}^i - \mathbf{x}_0|^2} \\ &= \left(\frac{\partial\left(\mathbf{x}^i\right)}{\partial t}.|\mathbf{x}^i - \mathbf{x}_0|.\mathbf{e}^i.\left(\mathbf{e}^i \bullet \mathbf{v}^i\right)\right)\frac{1}{|\mathbf{x}^i - \mathbf{x}_0|^2} \\ &= \left(\frac{\partial\left(\mathbf{x}^i\right)}{\partial t} - \mathbf{e}^i.\left(\mathbf{e}^i \bullet \mathbf{v}^i\right)\right)\frac{1}{|\mathbf{x}^i - \mathbf{x}_0|}\end{aligned}$$

This means that the derivation in time of the unit vector equals to the difference between the relative satellite velocity and the satellite velocity component in direction of line-of-sigh, divided by the range.

Therefore we have all the components we need in order to solve (3.30) which becomes:

$$\delta\mathbf{y} = \begin{bmatrix} -\mathbf{e}^1 & 1 \\ \vdots & \vdots \\ -\mathbf{e}^i & 1 \end{bmatrix}\begin{bmatrix} \delta_{x'} \\ \delta_{y'} \\ \delta_{z'} \\ \delta_{b'} \end{bmatrix} + \begin{bmatrix} \frac{-\partial \mathbf{e}^1}{\partial t} \\ \vdots \\ \frac{-\partial \mathbf{e}^i}{\partial t} \end{bmatrix}\begin{bmatrix} \delta_x \\ \delta_y \\ \delta_z \end{bmatrix} + \varepsilon' \tag{3.34}$$

Eventually, we obtain a linear equation with seven unknowns and, in order to solve through least squares, we desire to have all the unknowns grouped in the same term:

$$\delta\mathbf{y} = \begin{bmatrix} -\mathbf{e}^1 & 1 & \frac{-\partial \mathbf{e}^1}{\partial t} \\ \vdots & \vdots & \vdots \\ -\mathbf{e}^i & 1 & \frac{-\partial \mathbf{e}^i}{\partial t} \end{bmatrix}\begin{bmatrix} \delta_{x'} \\ \delta_{y'} \\ \delta_{z'} \\ \delta_{b'} \\ \delta_x \\ \delta_y \\ \delta_z \end{bmatrix} + \varepsilon' \tag{3.35}$$

Theoretically, if the receiver is stationary, $\delta_{x'}$, $\delta_{y'}$ and $\delta_{z'}$ equal to zero, then the system reduces to four unknowns:

$$\delta\mathbf{y} = \begin{bmatrix} \frac{-\partial \mathbf{e}^1}{\partial t} & 1 \\ \vdots & \vdots \\ \frac{-\partial \mathbf{e}^i}{\partial t} & 1 \end{bmatrix}\begin{bmatrix} \delta_x \\ \delta_y \\ \delta_z \\ \delta_{b'} \end{bmatrix} + \varepsilon' \tag{3.36}$$

3.3 Summary

This chapter was devoted to describing the theory and implementation methods for the two main directions of the present project:

- Improving accuracy
- Providing auto-assistance

Concerning accuracy improvement this chapter contained a description of the tracking algorithms we have implemented, it showed the method of solving for ionospheric delay,by using IONEX, and

for tropospheric delay, and it described how to get URA indeces and how they are used as weights in a weighted least squares solution.

The auto-assistance section covered the description of how we successfully provided assisted time to the positioning algorithms and described a new method to estimate TOW. Furthermore this section described how ephemerides were automatically provided. The last subject we approached was a description of the Doppler navigation algorithm implementation.

CHAPTER 4. EXPERIMENTAL RESULTS

This chapter will provide the reader with experimental results that prove that by using GPS snapshot techniques power consumption is greatly reduced. Furthermore, this chapter also presents the accuracy improvements obtained by applying tracking algorithms, pseudo-range corrections and weighted least squares. Therefore this chapter shows the results obtained after the analysis, development and implementation of those issues which were pointed out in our project scope in Section 1.4.

4.1 Energy Saving

Aiming to demonstrate the great benefits provided by snapshot techniques in comparison to software-defined radio or other continuous receivers, power tests were carried out, achieving significant numerical values.

Basically, the hardware block used in this project is the same one used in [5]: Front-end SiGe SE4110L and a USB chip allowing interfacing with any computer. Power consumptions presented below include the contribution of both devices. All measurements were taken by a multitester using a bypassed USB cable.

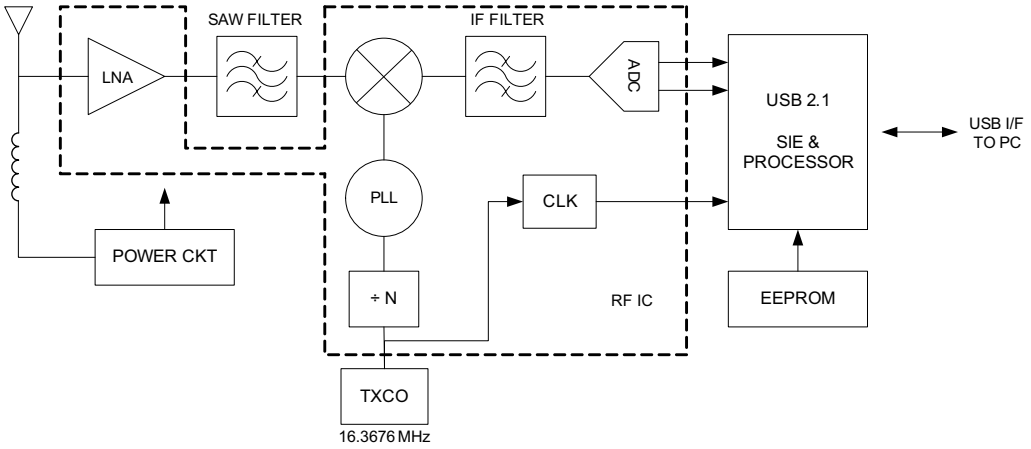


Figure 4.1: GPS1A hardware block

According to the GPS1A brochure [15], power consumption at 5 V shall be 45 mA. In practice, the power consumption was slightly bigger. The USB port of the laptop was constantly supplied 5.01 V during the current measurements.

Table 4.1: GPS1A power consumption

Status	Current consumption	Power
Active	72.5 mA	363.2 mW
Transferring	87.5 mA	438.4 mW

"Active" means that the front-end chip is supplied and operating, but the USB chip is not sending any data. On the other hand, "Transferring" represents the state when data is being transferred to the laptop. From these values one is able to estimate which is the power increment required in order to transmit data, although it has to be taken into account that the USB chip is also consuming power in the "Active" state because it has to be listening for potential commands sent by the laptop. The datasheet of the USB chip [16] specifies that the power dissipation is 300 mW.

Subtracting this value from the power consumption in the "Active" state results in an operating current of the front-end of 12.64 mA, which fits with the 10 mA value shown in [17] (the value might be a little bit higher due to the step-down voltage converter used to supply 3.3 V to the front-end and/or other residual power consumptions of the circuitry).

Perhaps, the comparison between snapshot technique and other kinds of GPS receivers may not be fair considering their radical different operational approaches: Snapshot techniques do not allow on-the-fly positioning. On the other side, one needs a reference to evaluate its benefits and drawbacks. For that purpose, typical Time To First Fix (TTFF) values, minimum operating time of a receiver prior to converging to the first fix, have been used to compute orientating energetic expenditures.

Table 4.2: Typical TTFF for different types of starts. Further details on each type of start and the origin of these values is found in [14]

Type of Start	TTFF
Autonomous Cold	1 min
Autonomous Warm	30 s
Autonomous Hot	< 6 s
Assisted Cold, Coarse Time	< 6 s
Assisted Cold, Fine Time	1 s

In the most optimistic case, "Assisted Cold, Fine Time" (only possible on a hand-held device with Internet access or an alternative media for data assistance), the energy consumed by the front-end in 1 second would be around 63 mJ (363.2 mW – 300 mW of power dissipation), whereas, while using snapshot techniques, 11 ms of "Transferring" state would be enough. This means that just 5 mJ are consumed in the GPS1A hardware block (including required energy for the USB chip to transfer the snapshot). This proves a dramatic decrease in used energy levels.

4.2 Accuracy Improvement

4.2.1 Tracking results

As described in Section 3.1.1 by averaging the discriminator output we were hoping to get a more precise correction value. For the present example, the mean value of the discriminator was -0.128 which means that the late code replica had the best correlation results, therefore the acquired code phase should be moved back.

Table 4.3 shows statistical positioning error results obtained by applying the method previously described in comparison to statistical positioning error results obtained by just performing acquisition. As we can observe, by applying the tracking algorithm we obtain an improvement in positioning.

Table 4.3: Error statistics for acquisition and/or tracking based positioning. Detailed specifications of all snapshots is found in appendix A. Unit: meters.

		HPE			VPE		
Snapshot	Method	μ	σ	2σ (95 %)	μ	σ	2σ (95 %)
S4G	Acquisition	12.256	6.519	23.202	53.386	16.143	79.513
	Tracking	10.678	5.600	20.005	52.076	14.449	75.874
S1D	Acquisition	10.092	5.552	19.559	20.300	17.356	47.874
	Tracking	8.341	4.543	16.707	21.845	13.993	45.209

We tested the present algorithm for more than one snapshot. As it can be observed, the degree of correction in positioning is similar for both snapshots. This proves that the algorithm is consistent. While implementing several different versions of the tracking algorithm, we obtained the best results by applying the averaged discriminator value as a correction to the code phase obtained by acquisition. Needless to say, this might not seem as the best solution. We expect tracking algorithms to have a greater impact on positioning results. Nonetheless, we must keep in mind that we are not dealing with traditional continuous tracking.

4.2.2 Pseudo-range Corrections

After tracking, removing pseudo-range delays becomes the ultimate "bottle-neck". Unfortunately, the performance achieved by our tracking codes was not significantly better than those from acquisition, therefore, the benefits of pseudo-range corrections cannot be appreciated on top of any snapshot. This is due to the coarse acquisition resolution shadowing the improvement obtained by pseudo-range corrections which is of an inferior magnitude. In other words, using pseudo-range corrections does not pay off while applying them on measurements which have accuracy levels much worst than the error they would correct.

Hence, taking this fact into account, and accepting failure at computing tracking, we cannot demonstrate their benefits on top of snapshot techniques. Nevertheless, the research and didactic purpose of the present project drove us to find an alternative method to demonstrate the theoretical concepts presented in Section 2.1.2. So, in this section we are going to proceed anyway by presenting results for pseudo-range corrections obtained with a traditional GPS receiver. In this way, one is able to figure out which would be the accuracy level in case tracking is achieved in the future.

Serving that purpose, a test-bed based on observations taken by traditional GPS receivers tracking the civil signal has been specifically created. Given that the fundamental measurements are full pseudo-ranges contained into RINEX observation files, instead of fractional pseudo-ranges after acquisition stages, a parallel set of code has been developed implementing a GPS navigator with the necessary RINEX interfacing and pseudo-range correction functions. Although this was a time consuming job, we believe the results will satisfy our curiosity and, somehow, bypass the tracking problems.

Table 4.4 shows the observation set selected:

Table 4.4: Observations set

ID	Date	Day of Year	GPS week	Characteristic	Location
O1D	06/04/2000	97	32	SA still active	DGC
O2D	16/05/2000	137	38	Solar cycle maximum	DGC
O3E	30/01/2002	30	1151	Second solar cycle peak	EBRE
O4D	24/11/2010	328	1611	Solar cycle minimum	DGC

Where,

DGC: 57.014731° N, 9.985904° E, 59.998 m Aalborg, Denmark.
EBRE: 40.820885° N, 0.492359° E, 107.797 m Ebre, Catalonia.

"O1D" has been selected owing to the fact that Selective Availability (SA) was still enabled by that time. The accuracy degradation resulting from SA is comparable to the effect of computing positioning from coarse acquisition values, so it will be an equivalent case study illustrating the shadowing effect described in the previous paragraph. On the other hand, the main reason for selecting observations from year 2000 and 2002 is that during this time period, a solar cycle maximum was taking place (see Figure 4.2), so they may be useful in order to compare the improvements provided by ionosphere corrections from IONEX files with standard single frequency

models (Klobuchar), as well as, a fair evaluation of the overall improvement in different conditions.

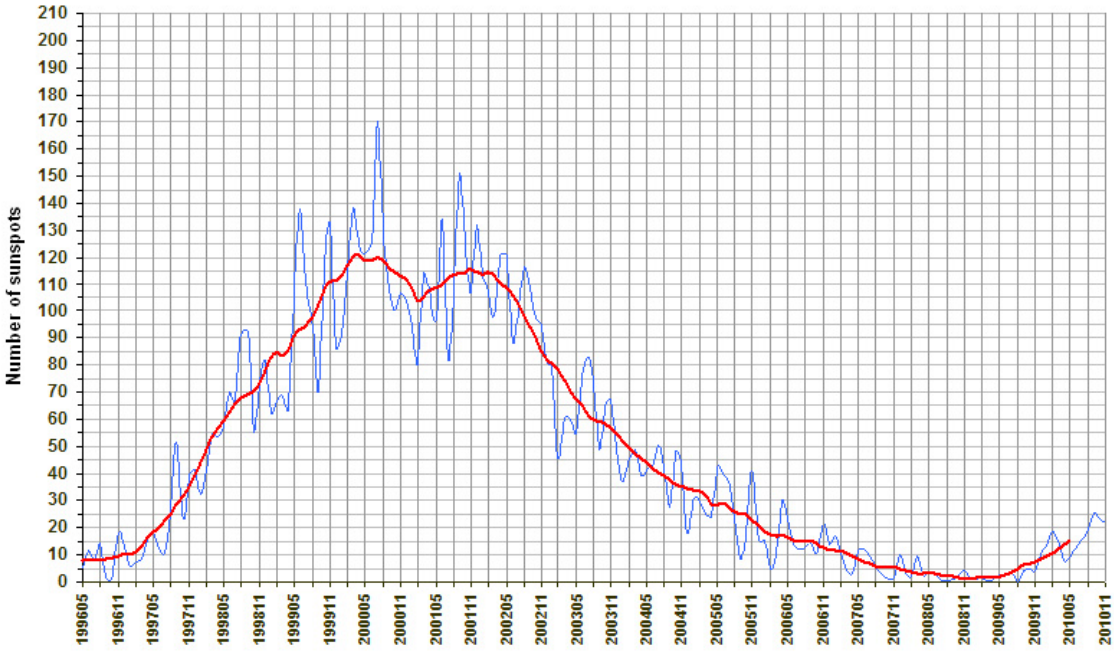


Figure 4.2: Solar Cycles 23-24. The current prediction is for a smoothed sunspot number maximum somewhere in the range 50-70. Figure taken from [18].

"O2D" in particular, has been selected matching with a "Grear Spot" recorded over 14/19 May 2000, which shall disturb the state of the ionosphere. All information related to sunspot events during the last solar maximum peak is found in reference [19].

"O3E" is found on a secondary peak of the solar maximum but, in this case, no special events were described and observations were collected by a permanent reference station located in Ebre, south of Catalonia.

Finally, "O4D" observations will provide the contrast with those observations corresponding to the maximum solar cycle.

Because there are many different corrections to be applied, different correction levels have been created to allow us to realize a fair comparison through the different sets of data. 6 consecutive levels have been defined, accumulating a new correction on each one of them. Table 4.5 describes which corrections are used:

Table 4.5: Correction levels. "Rel." stands for relativistic correction, "TGD" means Total Group Delay, "Trop." represents Tropospheric delay and "Klob." is an abbreviation for Klobuchar model.

	SV Clock Offset	Rel.	TGD	Trop.	Ionosphere	
					Klob.	IONEX
Level 0	X					
Level 1	X	X				
Level 2	X	X	X			
Level 3	X	X	X	X		
Level 4	X	X	X	X	X	
Level 5	X	X	X	X		X

SV Clock Offset is necessary for all of them and, as well as the relativistic correction, the total group delay and the ionosphere delay obtained with the Klobuchar model, is corrected according

to [3]. On the other side, the tropospheric correction and the ionosphere delay obtained with IONEX files are computed as described in Sections 3.1.2.2 and 3.7, respectively, following [2] recommendations.

Firstly, let us confirm the premise that pseudo-range corrections might be neglected in case an error of bigger magnitude is already present in the measurements: such as SA or coarse acquisitions values. Figure 4.3 displays the results after computing all C/A measurements contained in "AAUC0970.00o", a RINEX observation file which results in 2880 fixes for each level due to a 30 s data rate. Elevation mask was set to 10° for all computations in this section.

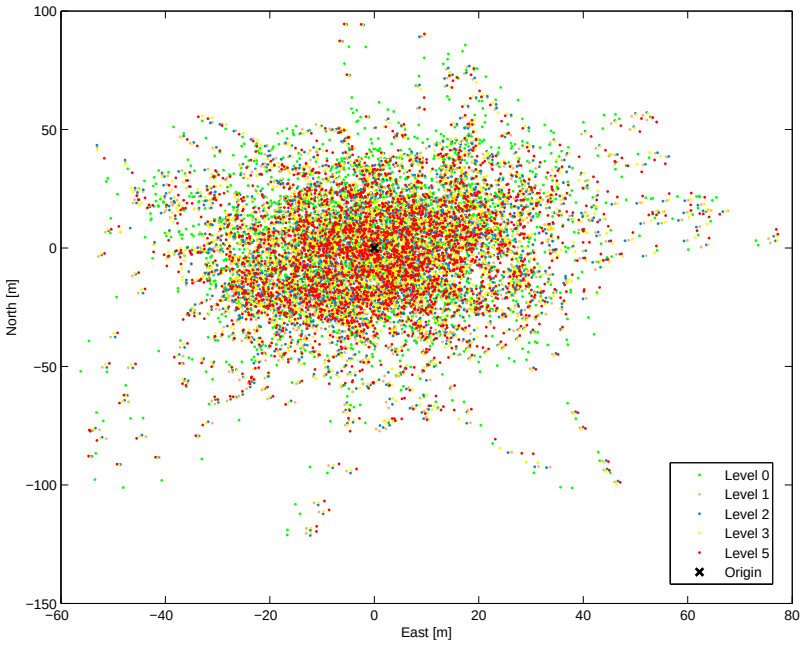


Figure 4.3: Pseudo-range corrections effects for "O1D": SA enabled. Correction levels are defined in Table 4.5

Note that almost no differences are appreciated, regardless which is the corrections level. Although not graphically presented, the same behavior is observed for the vertical component as showed by the statistics collected in Table 4.6. These results claim the need of tracking-like accuracy before any other further pseudo-range corrections can be applied.

From here on after, the matter is to verify the theoretical predictions of positioning accuracy summarized in Table 2.1. Repeating the experiment for "O4D"and using measurements from "AAUC3280.00o" we obtained results which are much more revealing.

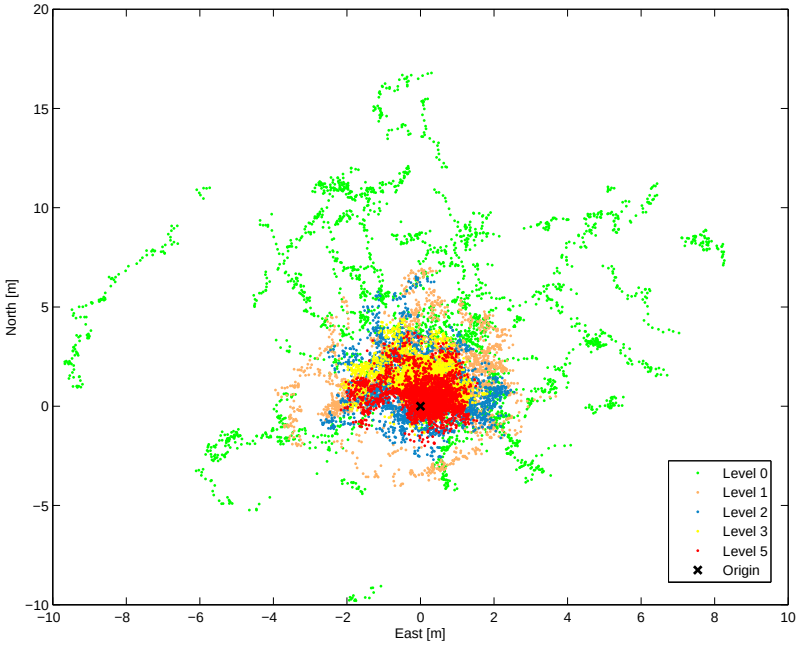


Figure 4.4: Horizontal accuracy improvement for "O4D". Correction levels defined in Table 4.5

It was drawn to our attention to realize the big impact of the relativistic correction, which appears to be the main source of error: more than 1 m gain in the horizontal plane and nearly 5 m in the vertical axis. From statistical results, troposphere and TGD corrections present similar contributions, although TGD has a bigger contribution in the vertical plane (> 1 m), because it is more sensitive to pseudo-range corrections due to the geometry of the system. Eventually, ionosphere delay corrections have an insignificant impact. In fact, in this case, applying Klobuchar corrections (Level 4 not shown in the figure above) had a slightly negative effect in the standard deviation of both the vertical and the horizontal plane. The feasible explanation might be that these observations are taken during a solar cycle minimum.

In order to fairly check the overall accuracy improvement of the ionosphere delay correction, "O2D" and "O3E" demonstrate its contribution during a solar cycle maximum. See Figure below:

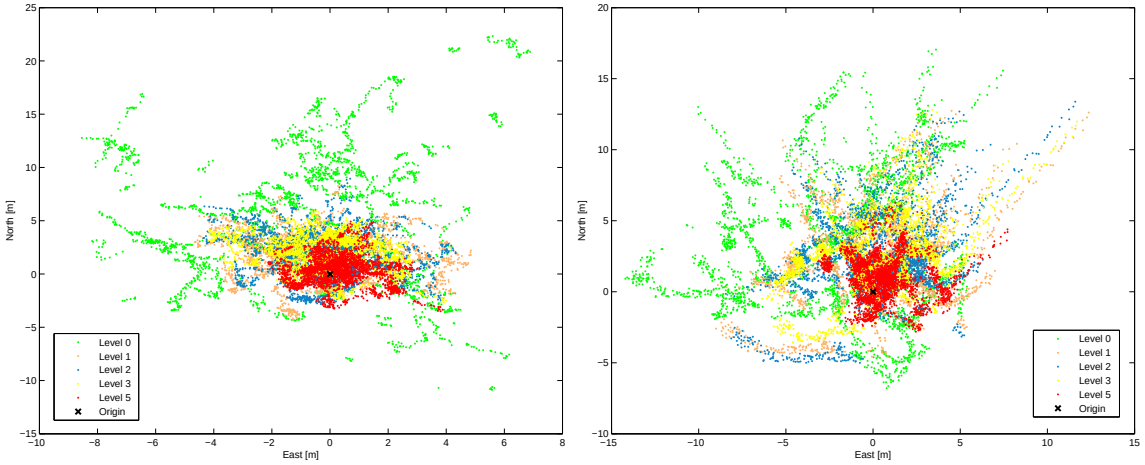


Figure 4.5: Horizontal accuracy improvement for "O2D" (left) and "O3E" (right). Both observables are found in a maximum solar cycle. Correction levels are defined in Table 4.5

In both cases, a visual check allows to notice a good concentration of all fixes after level 3. According to statistical results, to be more specific, the standard deviation σ in the Horizontal Positioning

Error (HPE) and the Vertical Positioning Error (VPE) are reduced by, at least, 0.5 m. Even higher values are found by the 2σ (95 %) indicator. However, the major benefit of the ionosphere corrections is not the standard deviation but the bias removal for all fixes. So, even though, the distribution of the fixes does not appear to decrease much, they are all dragged much closer to the origin, drastically reducing their offset (μ serves as an bias indicator). Thus, the ionosphere delay correction turns out to be a key one.

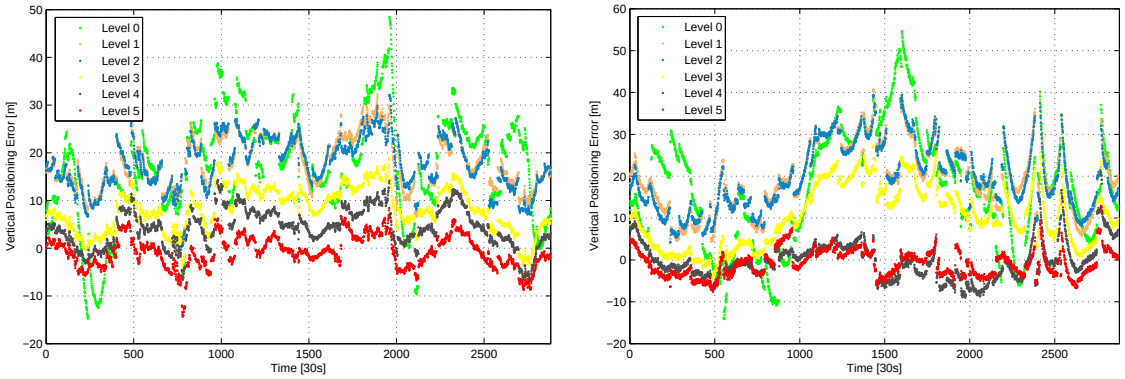


Figure 4.6: Vertical Positioning Error vs. Time for "O2D" (left) and "O3E" (right). Zero represents the actual height of the origin.

Figure 4.6 shows the evolution of the VPE during the day. At first, it confirms the offset reduction deduced from statistics. In addition, it also proves that, as expected, IONEX corrections perform better than the Klobuchar model. Lastly, as a curiosity, it depicts the behavior of the ionosphere delay itself during the day. Thus one is able to observe that its effect is bigger at higher latitudes. "O2D" is taken in Aalborg, at 57° of latitude whereas "O3E" is from Ebre at 41° of latitude (during night time). Instant 1500 (during day time) corresponds to time of day which equals to 1500·30 s= 45000 s, which actually is 12:30 h p.m.

All in all, after pseudo-range corrections, most of the systematic errors are removed from the measurements, leaving few errors to be corrected such as ephemeris errors, multipath and thermal noise, errors indeed difficult to correct by the user without using Differential GPS (DGPS) techniques, due to their random nature.

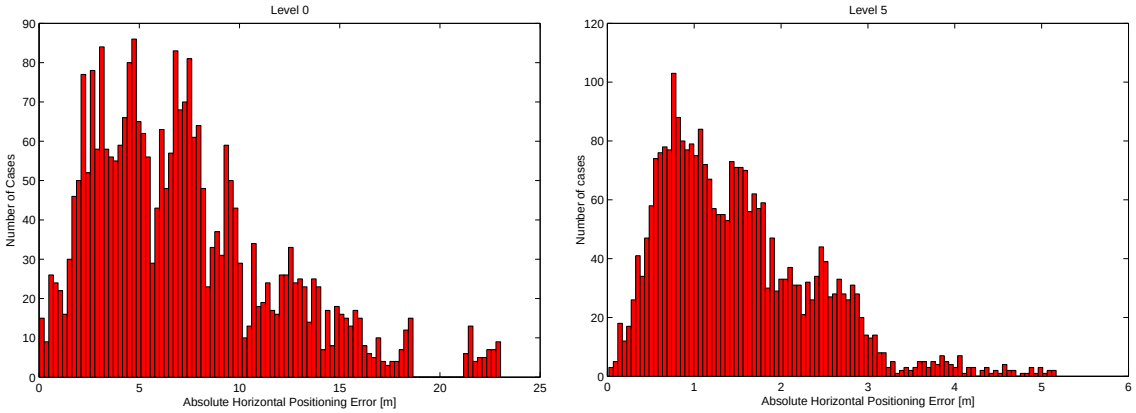


Figure 4.7: HPE histograms before (left) and after (right) corrections for "O2D"

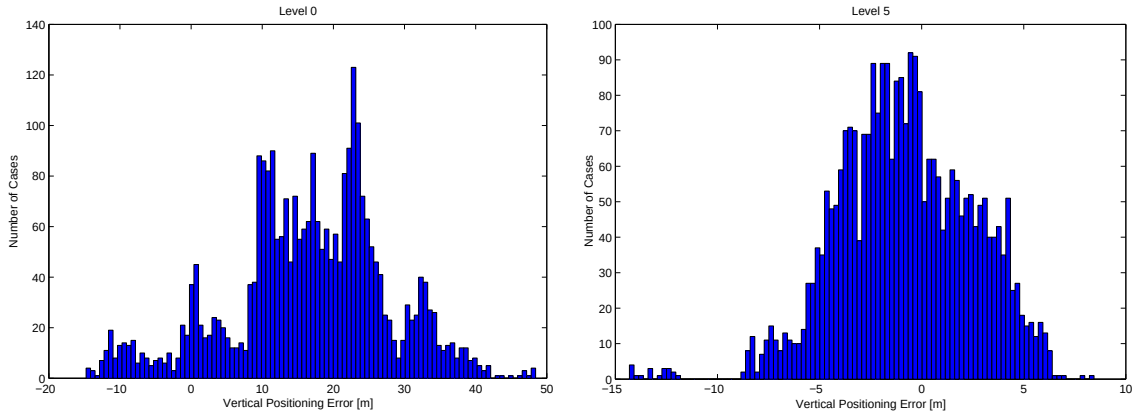


Figure 4.8: VPE histograms before (left) and after (right) corrections for "O2D"

The most remarkable point about Figures 4.7 and 4.8 besides showing one more time the accuracy enhancement, is the shape of the distribution before and after corrections. After corrections, the shape is undoubtedly a gaussian or normal distribution for the VPE and a raised normal for HPE. This demonstrates that the error leftovers behave like white noise which is equivalent to claim that they are random.

Table 4.6 gathers all statistical results of the experiments.

Table 4.6: Positioning statistics. HPE stands for Horizontal Positioning Error. VPE means Vertical Positioning Error

		HPE			VPE		
ID	Corrections	μ	σ	2σ (95 %)	μ	σ	2σ (95 %)
O1D	Level 0	26.309 m	16.050 m	55.276 m	15.075 m	54.759 m	97.544 m
	Level 1	25.619 m	15.828 m	54.640 m	15.842 m	54.347 m	95.157 m
	Level 2	25.559 m	15.807 m	54.549 m	15.549 m	54.211 m	94.574 m
	Level 3	25.540 m	15.826 m	55.194 m	6.064 m	54.125 m	85.292 m
	Level 4	25.540 m	15.830 m	55.249 m	3.078 m	54.123 m	82.390 m
	Level 5	25.526 m	15.800 m	55.456 m	−4.665 m	53.828 m	75.349 m
O2D	Level 0	7.284 m	4.605 m	15.951 m	16.621 m	11.137 m	34.036 m
	Level 1	2.930 m	1.408 m	5.526 m	17.954 m	5.286 m	26.683 m
	Level 2	2.846 m	1.466 m	5.598 m	17.654 m	5.300 m	25.839 m
	Level 3	2.616 m	1.234 m	4.844 m	8.125 m	4.559 m	15.004 m
	Level 4	2.456 m	1.238 m	4.712 m	4.270 m	4.022 m	10.500 m
	Level 5	1.502 m	0.882 m	3.025 m	−0.807 m	3.283 m	4.472 m
O3E	Level 0	6.909 m	3.463 m	13.127 m	17.059 m	13.410 m	39.062 m
	Level 1	4.554 m	2.435 m	9.276 m	19.492 m	7.880 m	32.188 m
	Level 2	4.282 m	2.434 m	8.756 m	19.293 m	7.605 m	32.098 m
	Level 3	3.956 m	2.275 m	8.131 m	10.129 m	7.201 m	22.232 m
	Level 4	2.901 m	1.879 m	6.291 m	−0.377 m	4.363 m	7.706 m
	Level 5	2.102 m	1.281 m	4.375 m	−0.868 m	3.110 m	4.434 m
O4D	Level 0	5.972 m	3.519 m	11.689 m	7.698 m	9.099 m	21.738 m
	Level 1	2.394 m	1.204 m	4.745 m	10.480 m	4.071 m	17.002 m
	Level 2	1.694 m	1.060 m	3.637 m	11.097 m	2.385 m	15.755 m
	Level 3	1.573 m	0.765 m	3.045 m	1.594 m	1.623 m	4.928 m
	Level 4	1.589 m	0.786 m	3.125 m	−2.019 m	2.015 m	1.930 m
	Level 5	0.975 m	0.648 m	2.321 m	−0.814 m	1.779 m	2.738 m

4.2.3 Weighted Least Squares

So far all results have been computed as Least Squares Solutions. In the following paragraphs, we will proceed describing the results for the WLS criteria detailed in Section 3.1.3.

Table 4.7: Weighted least squares positioning statistics. LS stands for Least Squares. WLS means Weighted Least Squares

		HPE			VPE		
ID	Corrections	μ	σ	2σ (95 %)	μ	σ	2σ (95 %)
O2D	LS	2.102 m	1.281 m	4.375 m	−0.868 m	3.110 m	4.434 m
	WLS_{IURA}	2.193 m	1.344 m	4.362 m	−1.052 m	3.165 m	4.289 m
	$\text{WLS}_{\text{custom}}$	2.177 m	1.303 m	4.335 m	−1.056 m	3.132 m	4.252 m
O3E	LS	1.502 m	0.882 m	3.025 m	−0.807 m	3.283 m	4.472 m
	WLS_{IURA}	1.481 m	0.883 m	3.056 m	−1.086 m	3.135 m	4.095 m
	$\text{WLS}_{\text{custom}}$	1.472 m	0.879 m	3.031 m	−1.081 m	3.141 m	4.080 m
O4D	LS	0.975 m	0.648 m	2.321 m	−0.814 m	1.779 m	2.738 m
	WLS_{IURA}	0.926 m	0.622 m	2.221 m	−0.818 m	1.725 m	2.728 m
	$\text{WLS}_{\text{custom}}$	0.928 m	0.623 m	2.203 m	−0.790 m	1.725 m	2.828 m

Eventually, refined values for the multipath variance model were tuned according to results in [12] and through simulations. The best results were achieved for $\sigma^2_{i,multipath} = 0.12 + 0.8exp^{(-el_i/10)}$. However, results do not show a significant accuracy improvement for WLS_{custom} relative to WLS_{IURA}.

4.3 Summary

In the beginning of this chapter, numerical values about power consumption demonstrated that snapshot techniques permit an energy saving of at least 11 times the minimum energy required by the fastest assisted GPS receiver.

After, the experimental results achieved through tracking which, although improving around 3 m the ones computed after acquisition, are believed not to be successful enough according to our initial expectations. Admitting this, the next section assumes tracking is obtained with an accuracy similar to a traditional GPS receiver and demonstrates the potential of the pseudo-range error corrections discussed in the Development chapter. Numerical results prove their significant accuracy boost, for both, bias and standard deviation. The relativistic correction shows to be the one providing a better standard deviation enhancement, while the ionospheric delay correction is a key improving bias. Ionosphere delay computed using TEC from IONEX files deliver better corrections.

Finally, the different WLS solutions were evaluated. WLS using IURA slightly contribute to a better accuracy.

CHAPTER 5. IMPLEMENTATION DISCUSSION

This chapter contains a brief discussion about the implementation of the algorithms used in order to improve snapshot positioning accuracy and automatically provide assistance data, concentrating on the unresolved issues and on the problems we had while implementation. The chapter will end with a brief discussion concerning our group's time and priorities planning.

5.0.0.1 Improving Accuracy Algorithms Discussion

Although at a first glance applying tracking algorithms seems pretty simple, we found it particulary difficult to obtain noticeable improvements in positioning. One of our ideas was to set up a threshold value and apply tracking in the same millisecond of the signal until this threshold is reached, showing that the incoming code and the replica code are well aligned. Due to the fact that the value which was compared to the previously mentioned threshold, the discriminator output, does not change radically while analyzing the same millisecond. Convergence of the algorithm was impossible to achieve. Another major issue related to tracking was that we are dealing with really short signals –7 ms–, therefore, even when we applied tracking for all the milliseconds iteratively, we had convergence problems. Therefore, even if we obtained at some point an improvement in accuracy, we consider that the overall tracking implementation was a failure.

Concerning solving the ionospheric and tropospheric delay and calculating URA weights, there were no major implementation issues. After analyzing and understanding the theory and methods, the whole process was pretty smooth, but at the same time very time consuming. The only thing that remained unsolved is that, due to the fact that our tracking algorithms did not provide significantly better results than acquisition, we could not judge the effects of the applied corrections for our snapshots. The reason why is explained in Section 4.2.2. Therefore we tested the pseudo-range corrections on a traditional GPS receiver by programming a different GPS navigator.

5.0.0.2 Auto-Assistance Methods Discussion

We did not have any major problems while implementing automatization of the time and ephemerides assistance. The only small obstacle was to find a way to download and de-compress a ".Z" file (file format of the ephemerides on the *ftp* server) in Matlab, due to the fact that normal Matlab de-compression functions do not work for this particular format.

The Doppler navigation algorithm is the only part of the project that remained unsolved, due to the fact that we could not make it to converge to a position within 150 km accuracy range. The algorithm needs to be provided with an initial position guess. We started by guessing this position at $(x, y, z) = (0, 0, 0)$. With this we could not obtain any good results, therefore we tried to get an initial guess calculating the initial linearization position using the average latitude and longitude of the visible satellites. This would provide us with an initial guess within the correct hemisphere at least, and theory says this should be enough [14]. Unfortunately this did not help. When we first applied the Doppler navigation algorithm we started from the premises that our receiver is stationary. This had the direct implication that, as seen in (3.35), the vector of unknowns would contain only the position updates and the frequency update. We thought that by eliminating updates to the receiver velocity states from the unknown matrix we would have to deal with numerical problems, and that could be the reason of the convergence failure. After programming the full Doppler navigation equation system we found that our problem was not yet solved. After an ulterior and more in depth research we discovered that the height component can prevent the algorithm to converge. Theory stated that the height should be obtained from knowledge of the area, fact that was not good for us due to the fact that we are trying to achieve a portable application, or it should be limited to zero. In order to do this we had to program the Doppler navigation algorithm in a local coordinate system (East,North,Up). Unfortunately this also ended up in a failure.

5.0.0.3 Time and Priorities Planning Discussion

We started this semester project having a different direction in our plans. We were planning to build a prototype as it is described in Chapter 7. When we finally decided that we might not have the necessary knowledge and time to complete this task we had already lost quite some time. Therefore, in order to make our work efficient we had to carefully plan an prioritize working tasks. We feel that we have done that successfully, even if some issues remained unfinished.

CHAPTER 6. CONCLUSIONS

The present project aimed to bring new functionality while using GPS snapshot techniques. Summarizing, our problem formulation, see Section 1.5, stated two specific aspects to be implemented or studied along this project: Providing more assistance data and improving position accuracy.

As stated in Chapter 5 we consider that we have failed to implement a stable, robust tracking algorithm, even though we obtained an improvement in the positioning errors. Another failure which we accepted was the implementation of the Doppler navigation algorithm. Nevertheless we would like to stress the fact that we consider these as failures only from a practical point of view, in the sense that we were not able to obtain results. Nevertheless, we still consider that, due to all the hard work that was put into solving these issues, we were successful from an academical point of view. Even though we failed to deliver results, we believe that all the research and time that were invested in applying proper tracking algorithms and achieving a first position fix through Doppler navigation algorithm were not in vain.

A success however, is the fact that we succeeded to simplify the work of a potential user of our software, in the sense that all the assistance data which is needed, except for the initial position, is automatically and transparently supplied. Therefore, the user must just record the snapshot and run the created software. Without any manual input the snapshot time tag is read and fed to the coarse time algorithm, and the ephemerides, in the format of RINEX files, are automatically downloaded from a *ftp* server, de-compressed and saved on the post processing calculating unit. Even more, in the same time, the same procedure takes place for the IONEX files that give information needed to solve for the ionospheric delay. Therefore, a potential user just has to assure that Internet connectivity is assured. We also consider that our TOW estimation algorithm was a success. Although this might not be obvious from a practical point of view, we consider that this was a good exercise and it was appealing from a theoretical point of view. Moreover, we have shown that by using this algorithm it was possible to recover the TOW of a snapshot taken in a computer which was isolated from any network, therefore not having a synchronized clock.

In our previous project about using GPS snapshot techniques we repeatedly stated that this technology should provide power consumption levels well below other technologies. This statement was not backed up by any tests or experiments. This project, however, provides a palpable measure of power consumption. We have shown that roughly 13 times less energy is consumed while using snapshot techniques in comparison to using a hand-held device with internet access that performs "Assisted Cold Fine Start", shortest time to first fix, therefore most energy efficient (see Section 4.1).

Concerning pseudo-range improvements we have only conducted a series of experiments on a traditional GPS receiver, due to the fact that we could not obtain significantly better results from the tracking algorithm, therefore we couldn't have shown the true effect of the pseudo-range corrections we have applied. Different experimental scenarios have been created aiming to show the way the ionosphere affects measurements in solar maximum cycles and that without a proper tracking algorithm we can not emphasize the true effects of the corrections. It was confirmed that if you have a major error source (SA, bad tracking algorithm) the pseudo-range corrections are not noticeable. Furthermore, different correction levels have been implemented, analyzing each of their effects. The main conclusion was that the best results are obtained while using IONEX files in order to correct for the ionospheric delay, and that by applying all correction levels we succeeded to remove most of the systematic errors from the measurements. All experiments have been done both by using LS and WLS using two different weighting criteria.

The reader should take into account that this is a student project, therefore, even if some aspects weren't solved, we consider that, all in all, from an academical point of view, the project was a success. Further work and analysis should be done in order to get closer to the actual possibility of creating a new product, some of these being presented in Chapter 7.

CHAPTER 7. FUTURE WORK

One of our ideas while choosing the main direction of the present project was to actually create a prototype that would prove the functionality of using GPS snapshot techniques in order to geo-tag digital photographs. After investigating what would this imply, we decided that we do not have sufficient time to put our idea into practice. Therefore we concluded that this should be left for future work. Nevertheless, we thought about a basic system architecture. Presently we are using a RF front-end in order to get variable length snapshots. From the obtained snaphots we are just processing blocks of 11 ms of data. This data is transmitted through a USB interface to the PC at a data rate between 4 and 16 Mbps. All the computations and algorithms are post-processed. The basic scheme is presented in the Figure 7.1.

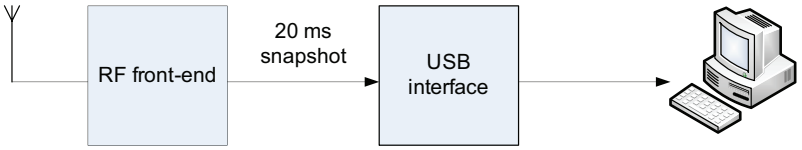


Figure 7.1: Current basic system architecture

One of the first things that should be done, in order to minimize the storage capacity requirements is to modify the front-end driver so that it can take snapshots shorter than 11 ms. Next we think it would be feasible to develop a portable prototype, eliminating the necessity of a PC connection. We think that this prototype could be implemented on a development platform which would satisfy a series of requirements:

- It has to be as small as possible
- It has to have a module (a clock) that can attach a time tag to the snapshot in the moment when it is taken (this should be controlled with a push of a button)
- It has to have some kind of memory large enough in order to store the snapshot and it's time tag — at least 1MB
- It should be able to write the data in the memory at a rate of 4–16 Mbps
- It has to have an interface through which the data from the memory can be later downloaded to a PC for post-processing

Figure 7.2 is meant to provide the reader with a basic idea of the architecture that could be created.

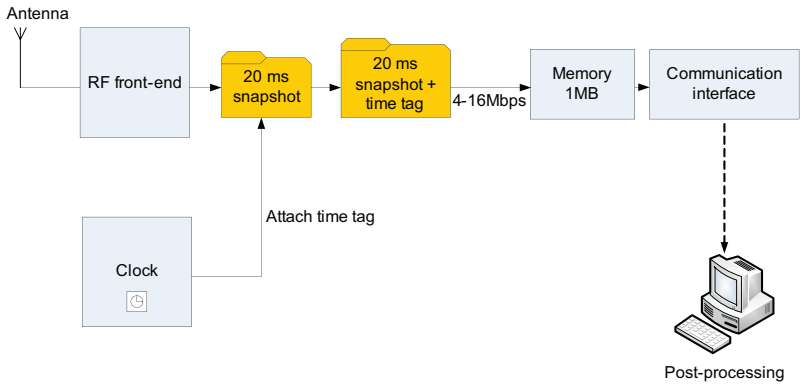


Figure 7.2: Prototype basic system architecture

It is clear that future work should implement a tracking algorithm that would lead to a more drastic improvement of the positioning results than the ones obtained after performing acquisition.

If such an algorithm would be implemented, the next step would be to check the effects of the pseudo-range corrections while using snapshot techniques. One other issue that has to be dealt with in future work is implementing a proper working Doppler positioning algorithm. The major part of these two algorithms are already done, their bodies have been programmed, therefore, we feel that we did not succeed to properly deal with this two issues also due to lack of time.

This being a student project, its main purposes are academical. We just wanted to provide a proof-of-concept. Consequently, we did not deal that much with the optimization of our software. We consider this also to be a significant future work. One way to optimize the code and shorten the execution time is to program the algorithms in another programming language such as C++ instead of using Matlab. Matlab is a very good didactic tool but it can not compete with C++ when it comes to building a commercial mass-market application. Another idea that we had that might optimize the execution time of the real-life application is that, due to the convergence capabilities of the positioning algorithms, we might be able to eliminate redundant snapshots. For example if two or more snapshots are taken in a very short interval of time, 20 s, say, we can store and use only one of them, taking into account that the user could not have moved that far away from the initial position. We imagine an application meant for digital photography, therefore most of the snapshots will be taken while standing still.

APPENDICES

APPENDIX A. PREVIOUS RESULTS

This Appendix summarizes some of the results obtained in our previous semester project [5], results which are relevant for the present report.

In order to validate our ideas we used several snapshots taken at different places. These are described in Table A.1.

Table A.1: Snapshots specifications

ID	Date	Time (UTC)	GPS week	GPS second	Location
S1D	02/03/2010	12h14m57s	1573	216912	DGC
S2C	25/03/2010	15h13m42s	1576	400437	CTAE
S3B	26/05/2010	15h46m24s	1584	315999	Blegkilde Allé
S4G	17/06/2010	12h43m28s	1588	391423	Gallecs
S5D	11/10/2010	10h02m55s	1605	122590	DGC

Where,

Blegkilde Allé:	57.030350° N, 9.941244° E, 51.00 m.	Aalborg, Denmark.
DGC:	57.014731° N, 9.985904° E, 59.998 m.	Aalborg, Denmark.
CTAE:	41.314255° N, 2.026309° E, 68.700 m.	Viladecans, Catalonia.
Gallecs:	41.549733° N, 2.212255° E, 117.922 m.	Mollet, Catalonia.

The optimal settings for the used acquisition algorithm are:

- Acquisition method = non-Coherent Power Integration Parallel Code Phase.
- Integration Period = 7 ms.
- Threshold = 2.
- Acquisition Search Band = ±14 kHz.
- Frequency Step = 500 Hz.
- Elevation mask = 5°.

Next we would like to present the position error statistics obtained through coarse-time position algorithm. Table A.2 serves as a numerically summary of the accuracy results.

Table A.2: Positioning statistics from snapshot "S1D". HPE stands for Horizontal Positioning Error. VTE means Vertical Positioning Error. All numbers are in units of meters

ID	μ HPE	σ HPE	2σ (95 %) HPE	μ VPE	σ VPE	2σ (95 %) VPE
S1D	8.28	3.72	14.64	39.85	14.66	64.88
S4D	29.47	11.38	47.48	11.81	15.93	38.14

One of the most important experimental results was the convergence strength of our algorithms towards biased assistance data. These results were important to consider for the present project, giving us some important information that made us:

- choose Doppler navigation as a solution which would enable us to get a position first fix
- estimate TOW

A.0.0.4 Resistance Towards Wrong Apriori Coordinates

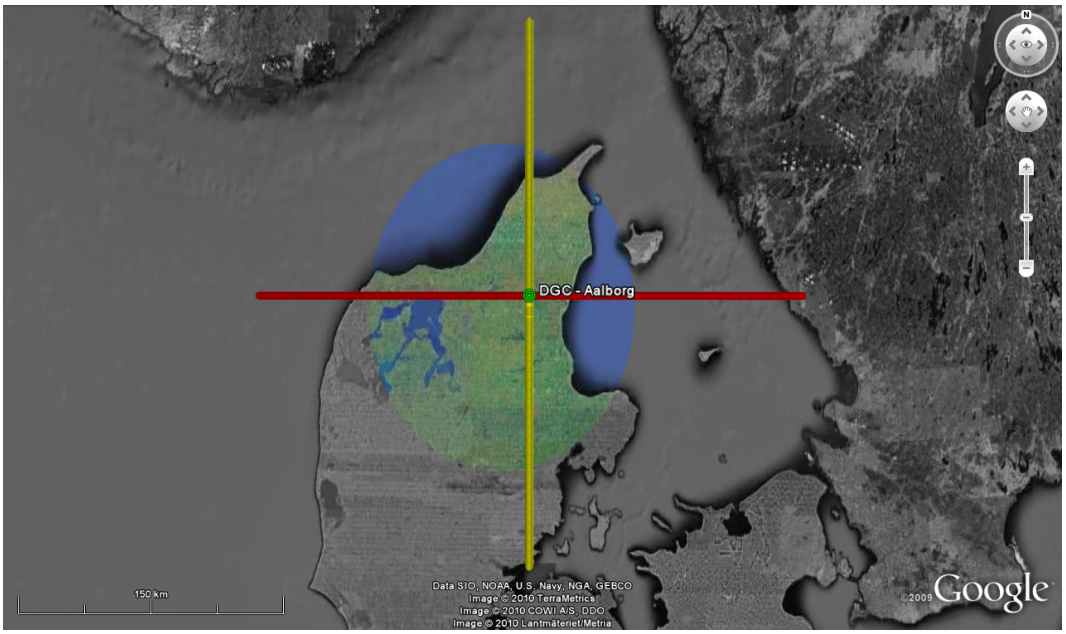


Figure A.1: Google Earth plot of a priori coordinates used to test the algorithm. The green sample represents the actual position of the antenna. Yellow: Samples with induced distance bias towards North. Red: Samples with induced distance bias towards East. Distance biases take values within ±150 km. Black and white areas represent the a priori coordinates that would cause algorithm divergence

Figure A.1 gives a better understanding of the area within an a priori position which will drive the algorithm to a valid fix.

A.0.0.5 Resistance Towards Bias on Assisted Time

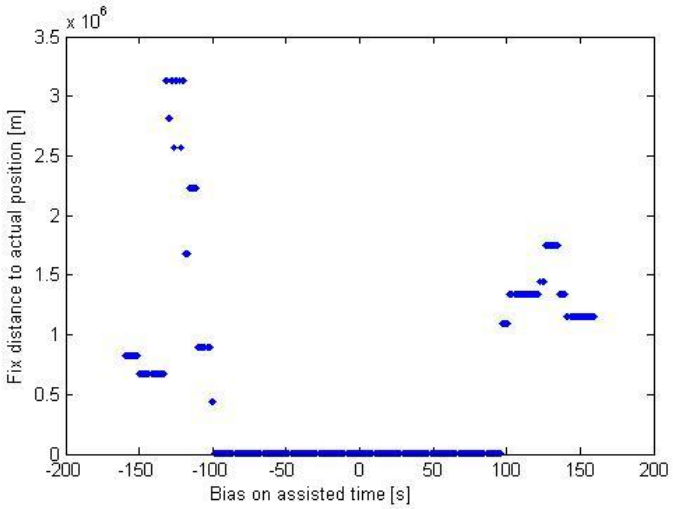


Figure A.2: Distance between fix and real coordinates versus bias on assisted time. 0 s in the horizontal axis represents the actual time when the snapshot was taken.

Figure A.2 shows the "convergence window" resulting from a time bias. For this particular snapshot, the algorithm was able to cope with an approximate time bias of ±100 s.

APPENDIX B. GLOSSARY

Table B.1: Abbreviations used in the report in alphabetical order

Acronym	Definition
A-GPS	Assisted GPS
CDDIS	Crustal Dynamics Date Information System
CPU	Central Processing Unit
DGC	Danish GPS Center
DLL	Delay lock loop
DoD	Department of Defence
DOP	Dilution of Precision
ECEF	Earth-Centered, Earth-Fixed
ECI	Earth-Centered Inertial
ENU	East North Up
FAT	File Allocation Table
FLL	Frequency Lock Loop
FTP	File Transfer Protocol
GDOP	Geometrical Dilution of Precision
GPS	Global Positioning System
GPST	GPS Time
GNSS	Global Navigation Satellite System
IF	Intermediate Frequency
IGS	International GNSS Service
IONEX	IONosphere map EXchange
IPP	Ionospheric Piercing Point
LBS	Location-Based Services
MEX	Matlab EXchange File
MOPS	Minimum Operational Performance Standard
NASA	National Aeronautics and Space Administration
NTFS	New Technology File System
OS	Operating System
PC	Personal Computer
PLL	Phase Lock Loop
PRN	Pseudo-Random Noise
RF	Radio Frequency
RINEX	Receiver Independent Exchange Format
RMS	Root Mean Square
RTCA	Radio Technical Commission for Aeronautics
SA	Selective Availability
SBAS	Satellite Based Augmentation Systems
SV	Space Vehicle
STEC	Slant TEC
TEC	Total Electron Content
TGD	Total Group Delay
TW	Time of Week
TTFF	Time to First Fix
URA	User Range Accuracy
USB	Universal Serial Bus
UTC	Universal Time Coordinate
Wi-Fi	Wireless Fidelity
WGS84	World Geodetic System 1984

APPENDIX C. GROUP DYNAMICS

For this semester project our group was made up just from two students: Oriol Badia Sole from Barcelona and Tudor Iacobescu from Romania. Both of us have similar education background, both having a bachelor degree in telecommunication engineering.

Because of the fact that we knew each other and worked together in the previous semesters as well, it was easier to get along from a personal point of view. Both of us have more or less the same mentality when work and study is concerned, so work cooperation was good this semester. Since the beginning of the semester we decided that we should have daily meetings. With some exceptions we respected this. Working together daily made communication between us easier. Another reason why we decided to work together every day is that in this way we could avoid redundant work. Being in the same room, it was very easy for each of us to communicate his work plan of the day. Basically we tried every morning to explain to each other what each of us wanted to do. After we agreed on the work plan of the day, we actually started work. Of course objections came up, but we succeeded to discuss in a calm way and work all our problems out. Besides these daily meetings we also tried to have a reflection and planning meeting each week. The purpose of this was to see what was accomplished each week, explain certain aspects to one another and plan for the next stage of the project work.

From a more technical point of view, we shared our work and data through Smart SVN version control. This software also acts like a means of data backup. But still after each modification in the code or the report we still made verbal notifications to each other, in order to avoid conflict errors which are very common while using version control.

As it is expected in every group of people, however small it may be, small conflicts and quarrels also took place. But in the end we think these were solved in a calm and mature way, leaving more time to concentrate on the study aspect of group work instead of the personal one.

With this project we will conclude our work together. As a reflection on group dynamics, not only in this semester, but throughout our entire studies, we feel we worked very good together.

BIBLIOGRAPHY

[1] Wireless Innovation Forum. Introduction to sdr. "`http://www.wirelessinnovation.org`", "unpublished manuscript".

[2] Minimum operational performance standards for global positioning system/wide area augmentation system airborne equipment. Technical report, 2006. DO-229C.

[3] GPS Joint Program Office. Is-gps-200 revision d - interface specification - navstar gps space segment/navigation user interfaces. Technical report, 2004. "`http://www.navcen.uscg.gov/gps/geninfo/IS-GPS-200D.pdf`".

[4] Elliott D. Kaplan and Christopher Hegarty. *Understanding GPS: Principles and Applications*. Royal Aircraft Establishment.

[5] Oriol Badia and Tudor Iacobescu. Gps snapshot techniques. Technical report, Danish GPS Center, AAU.

[6] Kai Borre and Darius Plaušinaitis. Reconfigurable gnss receivers. "`http://kom.aau.dk/project/softgps/GNSS_SummerSchool_DGC.pdf`", 2007.

[7] Bradford W. Parkinson and James J. Spilker Jr. *Global Positioning System: Theory and Applications, volume I*. American Institute of Aeronautics and Astronautics, Inc., 1996. ISBN: 1-56347-106-X.

[8] M. Hernández-Pajares, J.M. Juan Zornoza, and J. Sanz Subirana. *GPS data processing: code and phase: Algorithms, Techniques and Recipes*. Edicions UPC, 1st edition (english) edition, 2005. ISBN 84-932230-5-0.

[9] Gilbert Strang and Kai Borre. *Linear Algebra, Geodesy and GPS*. Wellesley-Cambridge Press, 1997.

[10] Kai Borre, Dennis M. Akos, Nicolaj Bertelsen, Peter Rinder, and Søren Holdt Jensen. *A Software-defined GPS and Galileo receiver: A single-frequency approach*. Birkhäuser, 2007. ISBN-10 0-8176-4390-7.

[11] Stefan Schaer, Werner Gurtner, and Joachim Feltens. Ionex: The ionosphere map exchange format. Technical report, 1998. Version 1.0.

[12] Xin-Xiang JIN. *Theory of Carrier Adjusted DGPS Psitioning Approach and Some Experimental Results*. PhD thesis, Delft University, 1996.

[13] J.G. Walker. *Continuos Whole-Earth Coverage by Circular-Orbit Satellite Patterns*. Artech House Publishers, 2th edition edition, 2005. ISBN-10 / ASIN: 1580538940.

[14] Frank van Diggelen. *A-GPS: Assisted GPS, GNSS, and SBAS*. Boston: Artech House Publishers, 2009. ISBN 1596933747.

[15] Inc. John Wiley & Sons. Gps1a: A software-defined gnss receiver. Technical report, 2004. ISBN 0-471-38154-3.

[16] Cypress Semiconductor Corporation. *EZ-USB FX2LP™ USB Microcontroller*, 200104. Document: 38-08032 Revised September 23, 2010.

[17] SiGE. *SE4110L GPS Receiver IC*.

[18] Jan Alvestad. Solar terrestrial activity report. "`http://www.solen.info/solar/solcycle.html`".

[19] A. D. Andrews. Solar maximmum events 22 march to 31 may 2000. *Irish Astronomical Journal, 27(2), 160-232. Provided by the NASA Astrophysics Data System*.