PROJECTE FINAL DE MÀSTER

TÍTOL DEL PFM : Tècniques de posicionament amb instantànies GPS

TITULACIÓ: Màster universitari en enginyeria i gestió de les telecomunicacions

AUTOR: Oriol Badia Solé

DIRECTOR: Dagoberto Salazar Hernàndez

DATA: 1 d'octubre de 2010

Títol : Tècniques de posicionament amb instantànies GPS

Autor: Oriol Badia Solé

Director: Dagoberto Salazar Hernàndez

Data: 1 d'octubre de 2010

Resum

En els darrers anys, l'augment de prestacions dels processadors està provocant una revolució en l'arquitectura dels receptors radio. La idea de reduir els components de maquinari, traslladant part de la càrrega computacional al programari, ha esdevingut una realitat molt atractiva. Permetent, a banda d'una miniaturització molt notable, major flexibilitat i estalvi d'energia.

En aquest sentit, dins del món dels sistemes de posicionament global per satèl·lit, l'aparició de tècniques de posicionament basats en captures instantànies de senyal GPS és un exemple clar de les noves possibilitats aportades per aquesta arquitectura.

Així doncs, la motivació d'aquest projecte es l'estudi i el desenvolupament d'algoritmes per al posicionament basat en captures instantànies GPS -"GPS snapshot techniques"-; un àmbit punter i que prometedor de nous productes comercials de posicionament. Servint aquest propòsit, ens hem centrat en una única aplicació, l'etiquetatge de fotografies en càmeres digitals.

El projecte aborda la qüestió, primerament, amb la implementació i comparació de diferents mètodes d'adquisició de senyal i, acte seguit, amb la programació i test dels algoritmes de navegació "coarse-time" necessaris.

Title : GPS Snapshot Techniques

Author: Oriol Badia Solé

Director: Dagoberto Salazar Hernàndez

Date: October 1, 2010

Overview

Software-defined radio receivers are becoming a more and more attractive technology in the GPS field. The idea of minimizing hardware components and handing over the signal processing load to programable devices which support modifiable software is very appealing.
The motivation of the present project is to study and develop GPS snapshot techniques. Serving this purpose, we focused on a single application, photograph geo-tagging for digital cameras.
This concept is approached by analyzing, implementing and comparing different acquisition techniques, followed by the implementation and testing of the coarse-time navigation algorithm.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

From ancient times, positioning and navigation represented a great challenge mainly for travelers, sailors, and military. Nowadays, almost all traditional navigation techniques have been replaced by the Global Positioning System (GPS) or other Global Navigation Satellite Systems (GNSS) such as the Russian GLONASS (based on the same principles) and other Satellite Based Augmentation Systems (SBAS), like Wide Area Augmentation System (WAAS) or European Geostationary Navigation Overlay Service (EGNOS), improving GPS performance within certain geographical areas.

Despite its military origins, a non-encrypted civilian signal was included in its design, over the main L1 carrier (at 1575.42 MHz). So, GPS is actually a U.S. government satellite navigation system that provides a civilian signal. Nevertheless, the precision of this signal has been degraded and controlled through Selective Availability (SA) by the US Department of Defence, the operator of the system. This was intended to deny enemies the use of civilian GPS receivers. In 2000, at midnight, on the first of May, SA was turned off following an announcement by U.S. President Bill Clinton. From that day, GPS accuracy was improved up to few meters for civil users and its use became more and more popular.

Currently, GPS is widely spread on mass market products such as car navigators and/or other hand-held devices like mobile phones, meaning that it is no longer exclusive to professional users. On the other hand, it is also used for serving a variety of different applications besides navigation and positioning:

- tropospheric and ionospheric sounding (radio occultation)
- attitude control
- timing and equipment synchronization.

Focusing on mass market applications, there is a clear trend along different product generations which includes miniaturization and reducing power consumption. In this sense, software-defined GPS receivers have a major role.

Synthesizing, the main idea behind the software-defined receiver concept is moving as much processing load as possible into a processor, removing most of the Application-Specific Integrated Circuits (ASIC) or any other dedicated hardware. Thus, thanks to a single front-end chip which is performing basic signal conditioning operations –pre-amplification, down-conversion, filtering and analog to digital conversion–, its integration has been feasible into mobile devices not specifically designed for that purpose, devices such as cell phones.

In addition, other important advantages in terms of manufacturing costs and software flexibility must also be considered. Hence, software-defined radio has meant a revolution in

the last decade, leading to a new way of implementing GPS applications for civilians. Further explanations and references about software architecture are provided in section 1.2.2..

In the last years, an inherit technique of the software-defined radio called “snapshot-technique”has been born. Its main strength leans on a dramatic reduction of power consumption and hardware requirements, since it is a post-processing technique that does not require accurate satellite tracking, which demands continuous operation of the receiver in order to precisely determine frequency and phase of the incoming signals. Even though such technique will deliver worse accuracies, given that GPS relies on time transfer principles, it could represent a solution for a variety of applications where accurate positioning in real time is not required.

Potential applications employing these techniques are geo-tagging of photos, vehicles (for tolling purposes) and marine fauna monitoring, among others. Due to recent invention and the increasing number of applications promised by the snapshot techniques, we decided to base this project around its study and implementation.

CHAPTER 1. ANALYSIS

This chapter defines a specific application that serves as an inspiration for our project and, it contents more in-depth study of the issues mentioned in the introduction leading to the final problem formulation.

1.1. Problem Identification

Can snapshot techniques provide a solution for digital photography geo-tagging?

1.2. Evolution on Receivers' Technology

Any GNSS needs to perform the following generic steps before obtaining positioning:

- Signal Conditioning: Manipulation of the analog signal in order to meet the requirements of the next stages. Processes such as filtering, amplification, frequency down-conversion and, eventually, digitalization.

- Acquisition and Tracking: Signal processing in order to monitor and demodulate the signal. Concerning GPS this means code correlation through a 2D search space formed by frequency and code-phase, for each satellite. Every satellite has a Coarse/Acquisition (CA) code (which is a Gold code) that is unique for each satellite and well known by the receivers.

- Navigation Data Recovery: Demodulation of the navigation message is only possible after tracking stage. In GPS, there is a bit shift every 20 ms –50 bps– and the whole content of the navigation message is broken down in 25 frames, which at the same time are divided into 5 subframes. Each subframe starts with a Telemetry label word (TLM) which contains a preamble used for timing and data synchronization.

- Solve Observation Equations: As soon as pseudo-range measurements and ephemeris are recovered, the observation equations can be posed allowing positioning estimation.

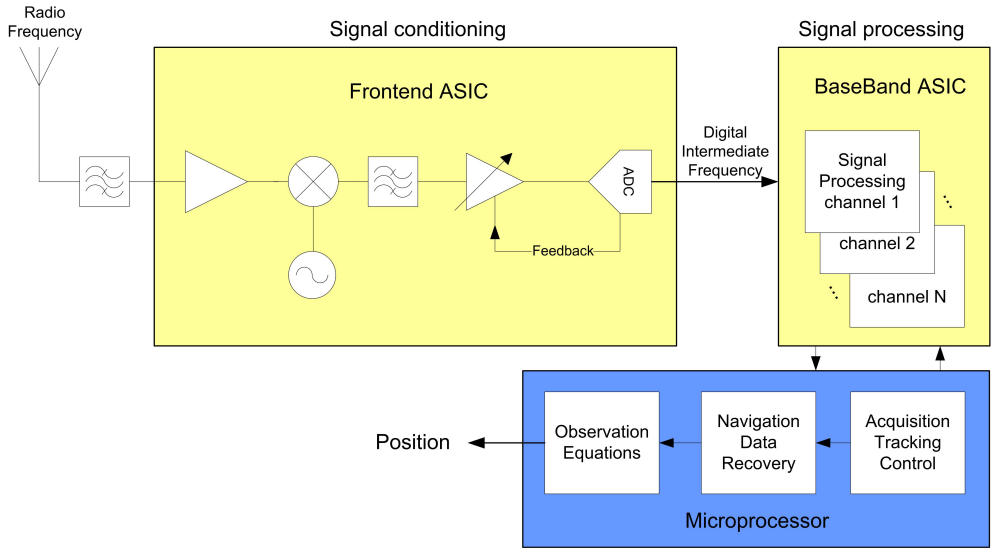

Figure 1.1: *Traditional GPS receiver architecture*

1.2.1. Traditional Hardware Receiver

Figure 1.2 represents a generic block diagram approaching the steps presented in 1.2., which serves as a basic design for any traditional solution, design used since the very beginning of the system operation:

Three physical parts are distinguished: front-end, baseband chip and a microprocessor. Each one of them performs different operations according to signal nature and their processing capacity.

The front-end is an integrated circuit in charge of signal conditioning and digitalization.

After digitalization at Intermediate Frequency, the heaviest signal processing operations take place: acquisition and tracking. Typically, the receiver has a number of ASICs, each one tracking a unique satellite. Every ASIC determines a channel, which is devoted to correlate the incoming signal and keep tracking on the Doppler frequency and code phase of a single space vehicle. Hence, the receiver will be able to follow as many satellites as channels (ASICs) –note that all specifications related to this process, including the number of channels, are determined by hardware so it is not possible to change any parameter unless the receiver is physical modified–.

Finally, the receiver synchronizes to the incoming signals, becoming able to decode the navigation messages and to form pseudo-range measurements. Whilst this is achieved, solving the observation equations is the only step missing. Both steps take place on a programable device –a microprocessor for instance–.

1.2.2. Software Defined Receiver

A decade ago, the technological improvements achieved on microprocessors and Digital Signal Processors (DSP) –in terms of processing rates–, as well as wider ADC bandwidths, permitted the development of a new receiver architecture called Software-Defined Radio (SDR).

The Wireless Innovation Forum in collaboration with the Institute of Electrical and Electronic Engineers (IEEE) P1900.1 group define a SDR as a *radio in which some or all of the physical layer functions are software defined*. In other words, it is a receiver architecture where some hardware stages are eliminated since their operations are moved into a software-defined processor instead.

The first complete GPS SDR implementation was described by Dennis Akos in 1997 [1]. Since then, several research groups have presented their contributions and some SDRs have been commercialized. Perhaps, one of the most successful cases was Nordnav-R30, a professional GNSS SDR produced by a Swedish start-up company, Nordnav, that was sold to CSR in 2007.

Nevertheless, besides receivers for professionals or researchers, the biggest market of SDR is found in a new trend of mass market devices called ”GPS data loggers”. Generally, they consist of a small GPS front-end connected to a computer through a USB cable and a specific software acting as a software-defined receiver. Some examples are the ones build by HOLUX, GiSTEQ, AmbiCom, GlobalSat or Deluo, among others. The majority of them are based on integrated front-ends manufactured by SiRF.

As we have seen, traditional hardware devices are built out of circuits dedicated to certain applications. If the user needs to use these devices in ways different than those for which they were purposely built for, physical modifications must be made. This could be a time consuming and expensive operation. With traditional hardware radios (any devices that transmit and/or receive via wireless media) it can be more cost effective to buy a new device rather than making hardware modifications. Therefore the idea of SDR has come up, allowing the user to easily and cheaply make changes and updates. Nevertheless, a 100% software radio is considered an utopia given that a certain level of signal conditioning is required –meaning that the front-end can not be eliminated from the architecture–. The more realistic case is that presented in Figure 1.2.

The basic idea of a SDR is that the signal processing is handed over to programable processing devices such as Field-Programmable Gate Arrays (FPGAs), Digital Signal Processors (DSPs) or General Purpose Processors (GPPs), devices which support the implementation of modifiable software [6].

The main advantages for using a SDR are:

- manufacturing on a common platform for a wide range of products

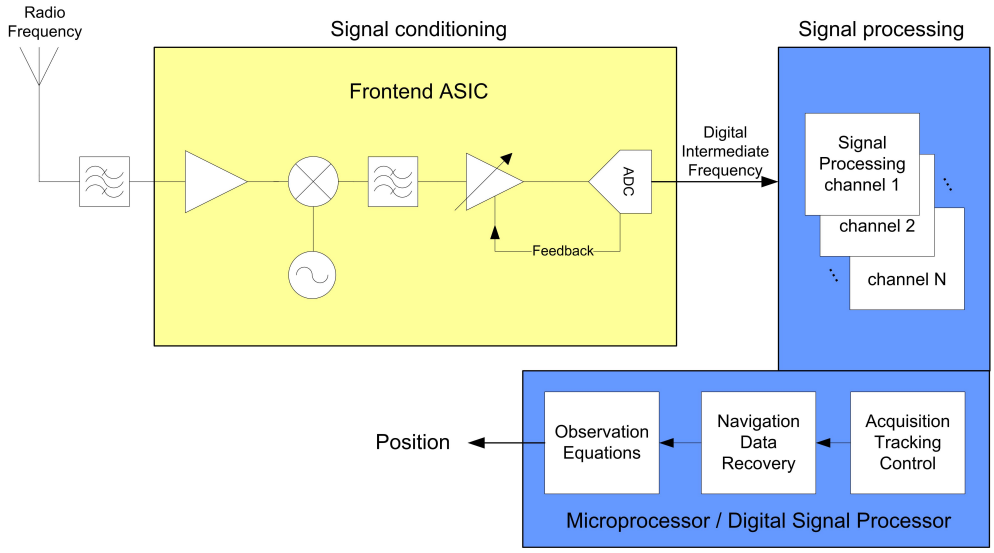

Figure 1.2: *Software defined receiver*

- cost effectiveness
- reusability
- integrability
- flexibility and reconfigurability
- ease of upgrading

Further explanations and details around this architecture can be found in [1].

1.3. Assisted GPS

AGPS, or Assisted GPS, is the technique of improving standard GPS performance by providing information, through an alternative communication channel, that the receiver would ordinarily have received from the satellites themselves [3].

These alternative communication channel could be any auxiliary data sources such as:

- Internet
- WLAN
- GPRS network

Assistance has permitted to integrate GPS on hand-held devices. Cell phones are the most common application. Its development was pushed by the Federal Communications

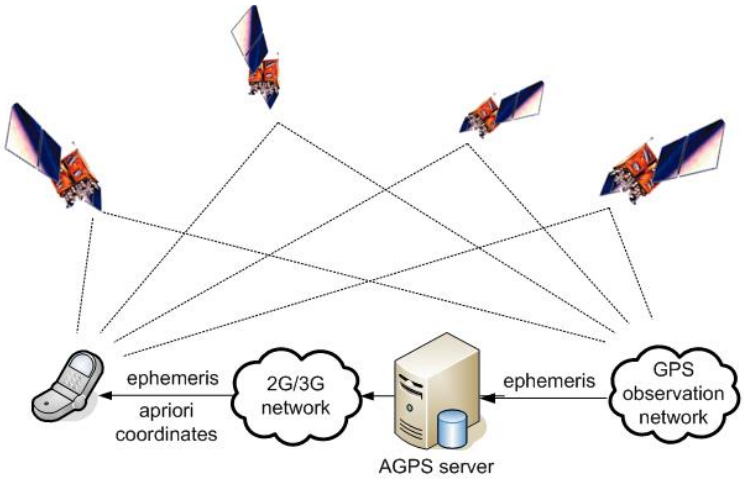

Figure 1.3: *AGPS schematic for cell phones*

Commission (FCC) in the USA, on a trial to improve the 911 emergency system –the American equivalent to 112 in Europe–.

Since the receiver is released from decoding the navigation messages, which is a costly processing operation, more resources can be invested on correlation, consequently enhancing sensitivity and speeding up the Time to First Fix (TTFF). Moreover, most of the cell towers provide their approximate coordinates, serving as a-priori guess for the receiver.

1.3.1. GPS Snapshot Techniques

GPS was designed with a start-up time in mind, approximately one minute, and after that it would operate continuously. However, AGPS allows to achieve TTFF below few seconds, which leads to a discontinuous use of the system, a completely new way of operation called snapshot techniques.

A snapshot is a short recording of the ”raw” data after digitalization of the IF signal in the front-end –short recording, meaning that it is in the order of seconds or milliseconds–. Typically, IF data implies high bit rates –in the order of several Mbps–, which would result in big storage capacities. Consequently, a limiting fact is that the snapshot time should be long enough to achieve acquisition –only few milliseconds required–.

When using snapshot techniques, the only real time operations such as amplification, down-conversion and digitization of the signal are the ones performed by the front-end. Other operations such as acquisition or positioning are performed in post-processing.

The concept of snapshot positioning is presented in figure 1.4:

Positioning using GPS snapshot techniques are expected to be within 20–50 meters accu-

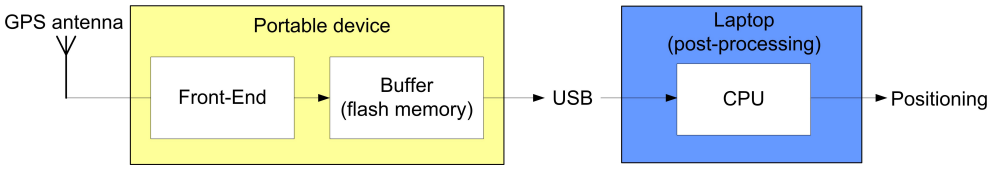

Figure 1.4: *Basic snapshot positioning principle*

racy range [7]. Hence, snapshot techniques provide a means to meet the needs of location based services by providing a low cost and low power consumption solution, by using simple front-ends.

Although the snapshot technique is a kind of AGPS in the sense that it also requires data from external sources, one could claim that its main distinctive aspects over AGPS are:

- It does not work in a continuous manner.
- Positioning is computed in post-processing.

1.4. Final Problem Formulation

Thanks to the evolution of GPS technology, we believe that nowadays snapshot techniques can represent a solution for digital photography geo-tagging due to the following reasons:

- It represents few hardware modifications over the original camera design –just enough room and power supply for a frontend chip and antenna–.
- Most of the processing load is moved into a laptop or desktop.
- It requires very small power consumption.
- It does not subject the user to any waiting time.
- It provides enough accuracy for the purpose of the application.

Consequently, we dedicated this project to the investigation of certain aspects of the snapshot techniques.

1.4.1. Previous Attempts to Solve the Problem

Various companies have and are still trying to address the issue of geo-tagging digital photographs. One of the earlier solutions was to build a separate piece of hardware, which is actually a GPS receiver able to achieve connectivity with both the digital camera and a PC. This is done either via a wireless media like Bluetooth or through a USB cable. These

devices are considered accessories and can be embedded or not in the camera mount. Examples of such devices may include the GiSTEQ PhotoTrackr series, Sony GPS-CS1, or the Jobo photoGPS. These devices have the disadvantage of being a separate piece of hardware with additional needs of power supply. Adding to this, their size is also an inconvenient. Therefore the need of an integrated solution arises.

Nevertheless, there are already digital cameras with GPS capabilities on the market. Examples of these may include: Ricoh 500SE, Nikon Coolpix P6000, Samsung ST1000 and Sony DSC-HX5V. All of them have been presented recently during the last year. However, we were not discouraged by the fact that these products already exist because we realized that all of them are based on SDR architecture instead of snapshot techniques, so they still require to operate continuously before and during the photo shoot, consuming considerable power and subjecting the user to waiting times. This, combined with the academical and educational purposes, is why we still have the desire to continue our research in GPS snapshot techniques.

1.4.2. Impact

Due to the fact that in this project we are trying to implement a technology that might result in the birth of a new product, we also took some time to analyze the economical impact that this might have. From an economical point of view, a digital photo camera falls in the so called *brown goods* category –consumer durables with low volume but high unit value–. We realized a brief quantitative market research, but because this is not our main field of expertise, we do not expect the results to be very exact.

During this research we found out that the world of digital cameras is ever evolving mostly due to the effervescent level of technological development and product feature innovations. According to International Data Corporation (IDC), these technical innovations are expected to propel the growth of digital cameras market in the year 2010 up to 122 million units. This makes the prospect of bringing up a new technical feature even more attractive.

After a low level market study we found out that the price of all the hardware elements that must be added to a digital camera rises up to a maximum of 10 dollars this including the front end, the antenna and the necessary connections to the memory which already exists in the camera. We did not take into account the software development costs. Just to get a rough idea of the magnitude of profit that can be achieved we did some simple calculations. We started with the premiss that 20 dollars must be added to the price of the camera. Due to the fact that production price is higher, as stated earlier, by 10 dollars, we get a profit of 10 dollars per unit. It is not to be expected that all the units on the market will deploy this technology. In this way we considered to take into our calculations just 1% of the market. This results in a profit of up to 12 million dollars. Again we want to stress that this calculations are very rough and a lot of economical details have not been taken into consideration. Even with some simple economical calculations performed on a nar-

row section of the market it is obvious that this technology may bring great profits to the companies that adopt it.

The social impact this product will have is also to be taken into account. One segment of the population that we think this product will be very handy to is the tourists. By having geo-referenced photographs, tourists and travelers will be able to keep much more exact diaries of their travels. Due to the high storage capacity of digital cameras, nowadays hundreds of photos can be stored. When watching these photos, this inevitably leads to some confusion about where a certain photo was taken. All this can be eliminated by using a digital camera with GPS snapshot capabilities. People used to mark places they have visited by pen, on a paper map. This can all be over. With this new technology people will be able to plot the visited points on applications such as Google Earth. This is much more attractive both from an esthetic and and from a practical point of view. Another social aspect that must be taken into account is the continuous growth in popularity of social networking web-sites such as: Facebook, Hi5, Flickr, MySpace, Yahoo!360, Zorpia and many others. These web-sites offer their users the possibility to share photographs amongst each other. Hence the desire of people to share more detailed experiences. This would be much easier to do with the help of a digital camera with GPS capabilities.

1.4.3. Project Scope

The scope of our project is to program post-processing software of a GPS snapshot system, given a specific front-end.

Behind any technological development or implementation there are always different compromises to be established. The project focuses on defining the performance of the snapshot techniques:

- Positioning accuracy versus snapshot length: Previous studies from other universities, research groups and companies show that an accuracy of 20–50 meters can be achieved using sub 20 millisecond snapshots [7]. As a general trend, accuracies come at the price of longer coherent integrations, meaning larger snapshots. On the other hand, reducing the amount of data is crucial in order to obtain longer autonomy of the device in terms of memory.

- Acquisition performance versus processing load: Given that snapshot techniques avoid a tracking stage, special attention must be payed to the acquisition algorithm, which should be as efficient as possible.

- Degree of assistance: A-priori position and time, as well as satellite ephemeris are required in order to achieve positioning. Without tracking stage, there is no way to synchronize with GPS time or decode any navigation data. Thus, such information must be assisted or provided from external sources. Studying the accuracy of

this assisted parameters is also required to guarantee successful postprocessing of each snapshot.

In this project we are dealing with acquisition and position computation in postprocessing. However, the communication link and protocols are not part of the project. Everything will be done locally on a computer connected to the front-end.

Therefore, this report will mainly focus on signal acquisition and Assisted-GPS (AGPS) algorithms presented in [1] and [3].

In case the reader is interested in a general explanation of the system, the following references might be useful [4], [8] and [9].

CHAPTER 2. DEVELOPMENT

2.1. Equipment

Our development was based on snapshots collected through a GPS RF front-end formed by an Integrated Circuit (IC) from SiGe semiconductors, the SE4110L, which includes an on-chip Low Noise Amplifier (LNA) and a low IF receiver with a linear Automatic Gain Control (AGC) and 2-bit analog-to-digital converter (ADC). All these are contained within a 4x4 mm low cost chip –further description of its specifications can be found in its datasheet [10]–.

Yet, an important issue is its frequency plan:

On paper, our device is configured with a sampling frequency of 16.368 MHz and an IF of 4.092 MHz but in practice, our unit has an IF of 4129945 Hz. These are key values limiting the pseudo-range resolution and, besides that, they must also be taken into account during acquisition.

$$Resolution = \frac{c}{SamplingFrequency} \simeq 18.32\,m$$

The election of the sampling frequency forces to establish a tradeoff between pseudo-range accuracy and amount of data. Let's assume for a moment that our snapshots have a length of 20ms, this would mean storing:

$$Size = SamplingFrequency \cdot 0.02 = 327.360\,kB$$

Nowadays, most of the hand-held devices use miniSD or microSD memory cards, whose capacities are often several gigabytes. Therefore, using 1 GB memory card would allow us to store more than a thousand snapshots. These rough calculations give us a good reason to think about its potential applications.

Figure 2.1: *SiGE front-end photography. It contains SE4110L and a USB interfacing chip.*

Reference frequency	Intermediate Frequency (SIGN/MAG pins)	Sample clock (CLK OUT pin)
13 MHz	-4.080 MHz	19.5 MHz
16.368 MHz	+4.092 MHz	16.368 MHz
19.2 MHz	-4.466 MHz	19.2 MHz
19.5 MHz	-4.080 MHz	19.5 MHz
26 MHz	-4.080 MHz	19.5 MHz

Figure 2.2: *Supported frequency plans*

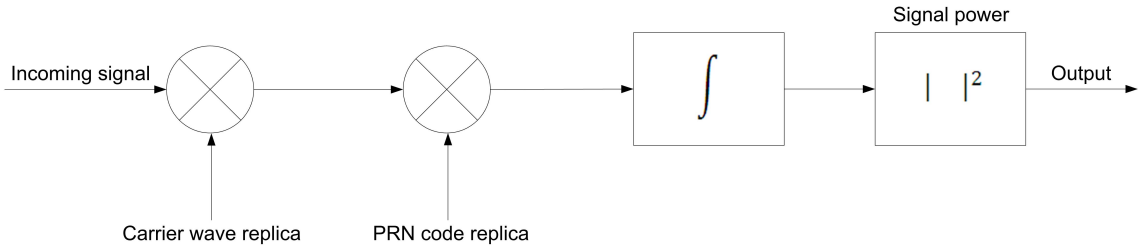

Figure 2.3: *Basic acquisition scheme, picture taken from [1]*

2.2. Acquisition Methods

The purpose of acquisition is to determine visible satellites and coarse values of carrier frequency and code phase. A basic acquisition scheme is presented in Figure 2.3.

A carrier wave replica must be generated locally in order to recover the original baseband signal. This is done by multiplying the two waves.

Next, the incoming code should be removed. This can be done only when the code phase of the signal is known. Due to the high autocorrelation with zero lag proprieties of the PRN code (see B.2.1.), it is possible to remove the incoming code by multiplying it with a locally generated PRN code replica.

It is clear that while performing acquisition there are two main search dimensions: frequency and code phase. The size of the search space can depend on whether the receiver has or does not have any kind of apriori knowledge. This can be observed in Figure 2.4:

While performing acquisition, the search space is an important issue because it determines the speed of the acquisition. Because this project employs snapshot techniques we should concentrate on the fastest method of acquisition. As stated before the search space is in two dimensions: carrier frequency and code phase. When it comes to code phase the search is done in 1023 combinations of code phase, corresponding to the number of chips. When searching carrier frequency combinations, the Doppler effect is an important issue. Due to the satellite and receiver motion a maximum Doppler shift of ~ 10kHz (this yields an approximate 20kHz search band for visible satellites) occurs. This frequency band is

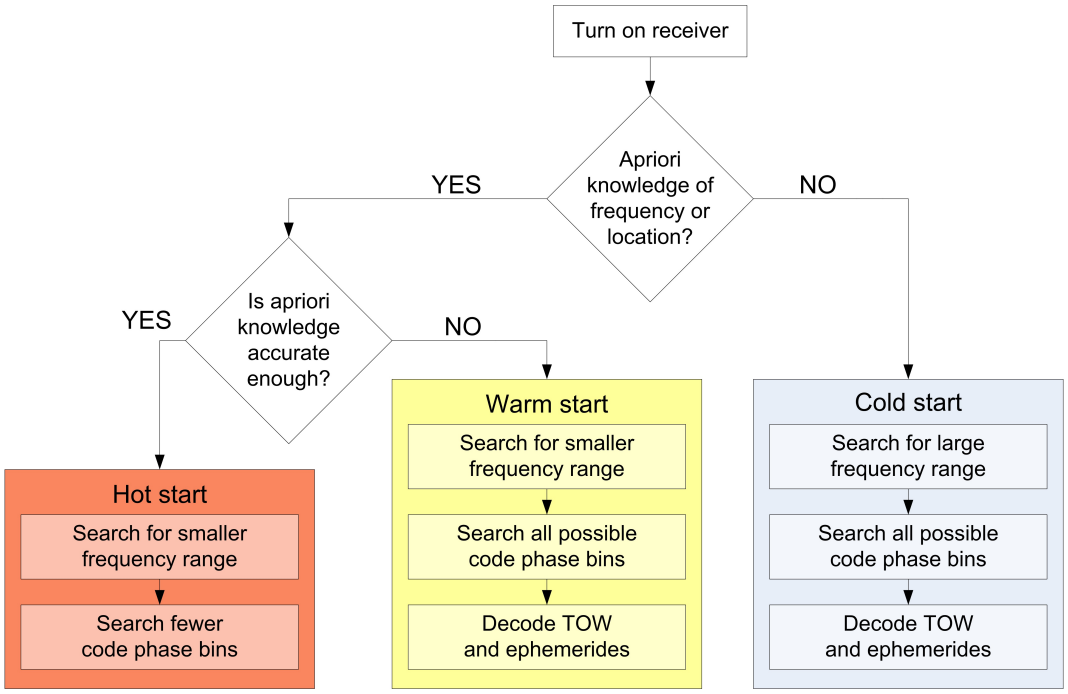

Figure 2.4: *Cold vs. warm vs. hot start [2]*

Table 2.1: *Acquisition methods, see [1]*

Type of acquisition	Advantages	Disadvantages
Serial	simple easy to implement	$41 \cdot 1023 = 41943$ search combinations very slow
Parallel Frequency space search	less search combinations 1023 code phases faster	more complicated due to FT dependant on FT implementation
Parallel Code Phase space search	least combinations 41 frequency bins the fastest	computationally demanding dependant on FT and IFT implementation the most complex

divided into frequency bins which give a certain number of possible code phase search combinations –note that the values are flexible on SDR–. By using a software receiver the search space can be exploited differently, according to the needs of the user, providing programming flexibility. Three types of acquisition methods are presented in [1]. A summary of their advantages and disadvantages are presented in Table 2.1 and the block diagrams in Figure 2.5.

The Serial acquisition is usually used in CDMA systems. The other two methods parallelize either frequency or code phase parameter in order to minimize the search space. They also use Fourier transform, adding to their complexity. The most efficient method, namely Parallel Code Phase search acquisition also uses circular correlation. All three methods use off-line code generation.

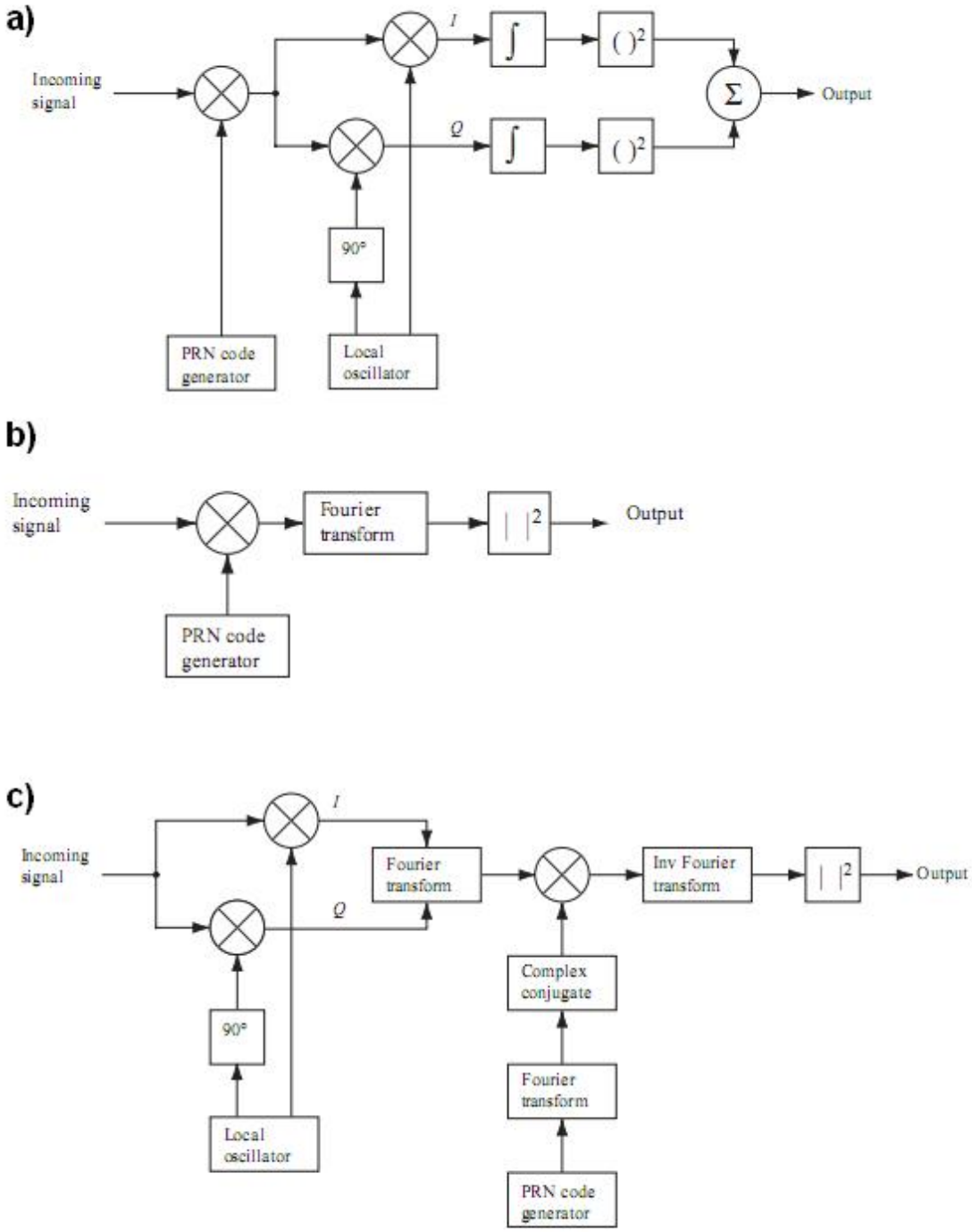

Figure 2.5: *a) Serial search acquisition; b) Parallel Frequency Space search acquisition; c)Parallel Code Phase search acquisition, see [1]*

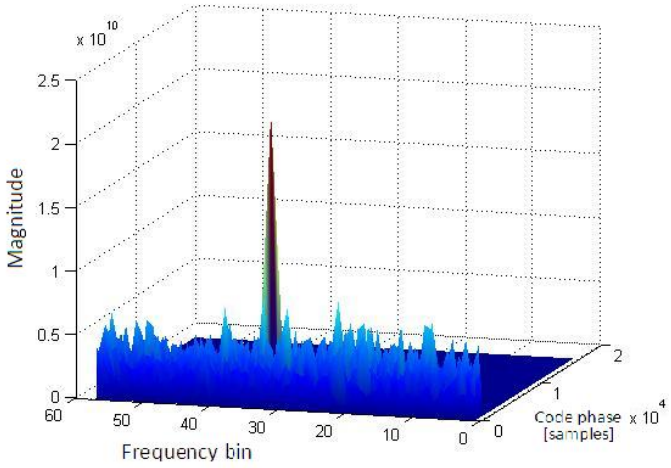

Figure 2.6: *Output of serial search acquisition performed on SV* 4

2.2.1. Serial Acquisition

The Serial acquisition is the simplest one to implement. Due to the fact that it does not use Fourier transform it is also the least computational demanding. As seen in figure 2.5a) the first step of this type of acquisition is to multiply the incoming signal with a locally generated PRN code. The in-phase and quadrature signals are obtained by multiplying the resulting signal with a locally generated carrier signal, and, respectively with a 90° shifted locally generated carrier signal. The last part of the algorithm involves obtaining signal power in both in-phase and quadrature arms. This algorithm is the slowest one because it sweeps all possible intermediate frequencies (± 10 kHZ corresponding to the Doppler shift) into 500 Hz steps, thus getting 41 frequency bins, plus 1023 different code phases. Due to the fact that 16 samples are used per each chip of the C/A code we actually get a number of 16368 samples per millisecond.

We proceeded to implement this algorithm and tested it with both simulated GPS signal and with real snapshots. Figure 2.6 shows the output of the Serial search algorithm with real snapshot. Due to the fact that this algorithm is very slow, the algorithm has been run only for five possible PRNs. Space vehicle (SV) 4 is visible and acquired.

2.2.2. Parallel Frequency Space Search Acquisition

This algorithm provides a faster acquisition method. In this case there exists a tradeoff between speed and how computational demanding it is, because, as shown in Figure 2.5b), the Parallel Frequency space search algorithm uses Fourier transform. The basic idea of this type of acquisition is to try to eliminate one of the two search spaces, namely the frequency dimension. Therefore the search only occurs in the code phase dimension, which means that only the 1023 different code phases are looked into.

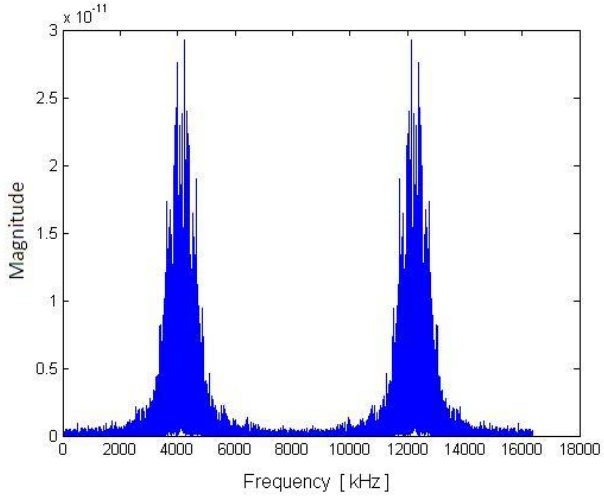

Figure 2.7: *Uncut spectrum*

The implementation of this algorithm is quite straightforward, following the basic block diagram presented in 2.5b). The incoming signal is multiplied with a locally generated PRN code. If the two signals perfectly align than the result will be a continuous wave. This takes advantage of the high autocorrelation and low cross-correlation proprieties of the PRN code (see B.2.1.). Therefore, if the two codes are perfectly aligned, the Fourier transform will result in a distinct peak.

While implementing this algorithm we observed that the spectrum of the signal power is symmetric. As stated in [11], when applying Fourier transform to a real input sequence, the first half of the resulting samples will be redundant with the second half (as shown in Figure 2.7) due to Nyquist theorem.

Therefore, in order not to get redundant peaks we cut the spectrum and used only the part around the intermediate frequency which we needed. The result is presented in Figure 2.8. One consequence of this is that in the plots presenting the obtained peaks, the frequency index where the peaks are situated are actually half of the real frequency index.

In order to verify the results of the Parallel Frequency space search acquisition algorithm we first used a simulated GPS signal without adding any noise. The results are presented in Figure 2.9, figure which shows the frequency and code phase of the obtained peak corresponding to the visible PRN. In this simulation we used the following settings:

- PRN2
- IF $= 4129945$ Hz
- sampling frequency $= 16.368$ MHz
- code length $= 1023$ chips
- Doppler frequency $= -6.3$ kHz

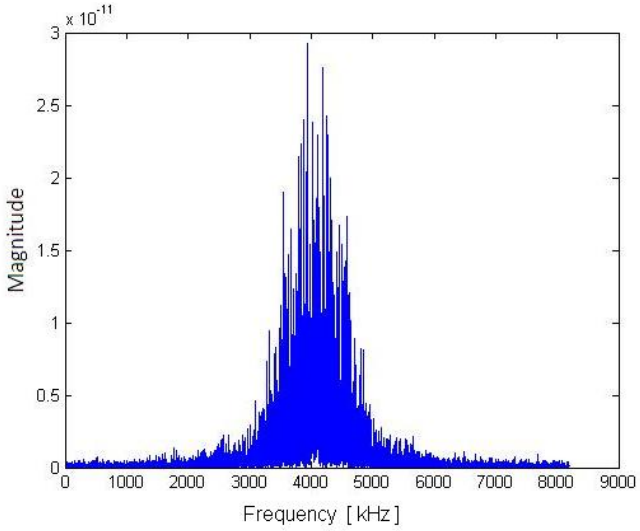

Figure 2.8: *Cut spectrum*

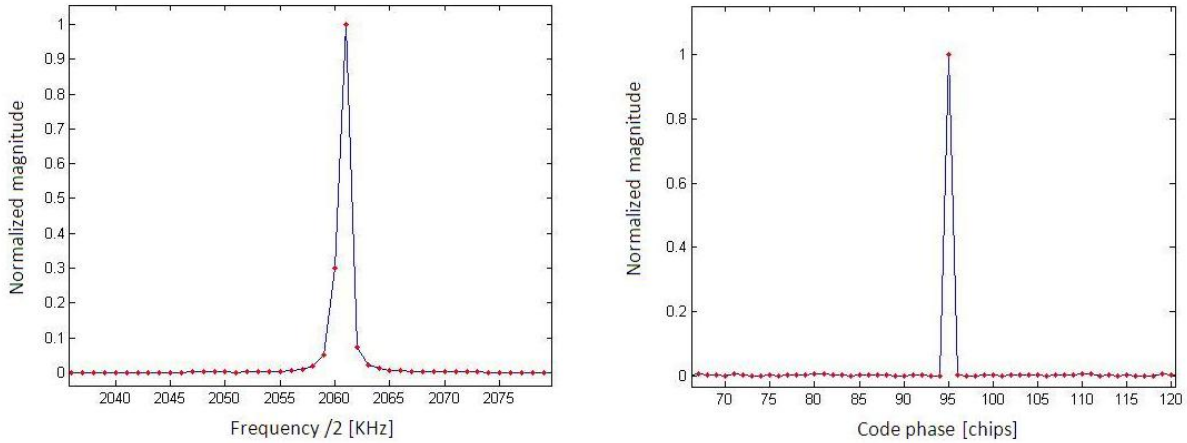

Figure 2.9: *Results from simulated Parallel Frequency search space acquisition*

- no noise

We then proceeded to test the algorithm on the real snapshot. Figure 2.10 shows the plot which correspond to visible satellites with PRN4 and PRN32.

2.2.3. Parallel Code Phase Space Search Acquisition

This is the fastest method of acquisition out of the three presented in Table 2.1, but the speed comes with the cost of being very computationally demanding. This method relies on parallelizing the code search space. Therefore we only need to search in the frequency bins. In our algorithm there are 57 different frequency bins due to our bandwidth and bin width settings. It is obvious that this is much more effective than searching all 1023 different code phases, or in both frequency bin and code phase search spaces. As shown in Figure 2.5c) the algorithm uses both Fourier transform and inverse Fourier transform (IFT). This

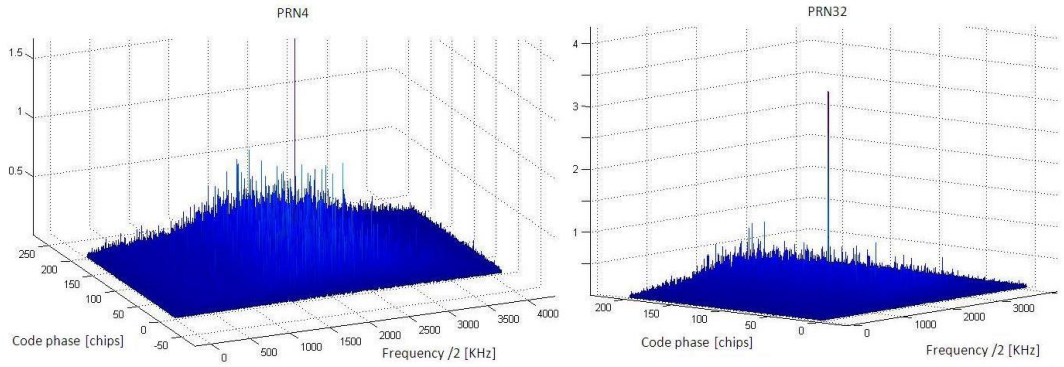

Figure 2.10: *Results from Parallel Frequency search space acquisition. PRN*4 *and PRN*32 *are visible*

is what makes the algorithm computational demanding. The incoming signal is multiplied by a locally generated carrier signal thus resulting in the in-phase (I) signal. By shifting the carrier signal with 90° and multiplying it with the incoming signal the quadrature (Q) signal is formed. These complex signals are combined into the sequence $x(n) = I(n) + jQ(n)$ and provided as input for the discrete Fourier transform function(DFT). The result of this function is multiplied with the complex conjugate of the locally generated PRN code in frequency domain. The next step is to transform the result back to time domain via inverse Fourier transform and get the absolute value of the resulting sequence. It must be taken into account when calculating the power of the signal that the output of the IFT is complex. Basically, at this stage we realized correlation between the input and the locally generated PRN code.

The code for this algorithm was taken from [1]. We proceeded in testing the algorithm as in the case of Parallel Frequency search space algorithm with both simulated GPS signal (with the same parameters) and with a real 1 second snapshot.

First we will show the results which were obtained while running acquisition with simulated GPS signal, both with and without noise in Figure 2.11. This test was made for visible PRN4. It is clearly visible that even when adding noise we can still obtain a peak which has a magnitude much higher than the noise, making acquisition possible.

We proceeded to test the algorithm with a real set of data. Figure 2.12 shows the plots for visible satellites with PRN4 and PRN32.

All the acquisition techniques presented above were performed on a single C/A period (1 ms). In order to achieve better acquisition, power integration can be implemented along several C/A periods.

Basically, there are two different types of power integrations: coherent and non-coherent. The difference lies on the fact that coherent acquisition integrates the signal separating between in-phase and quadrature branches, while non-coherent acquisition integrates the signal from its modulus. As shown in [12] and [13], implementing a coherent type of acquisition should have advantages over non-coherent acquisition, advantages as higher

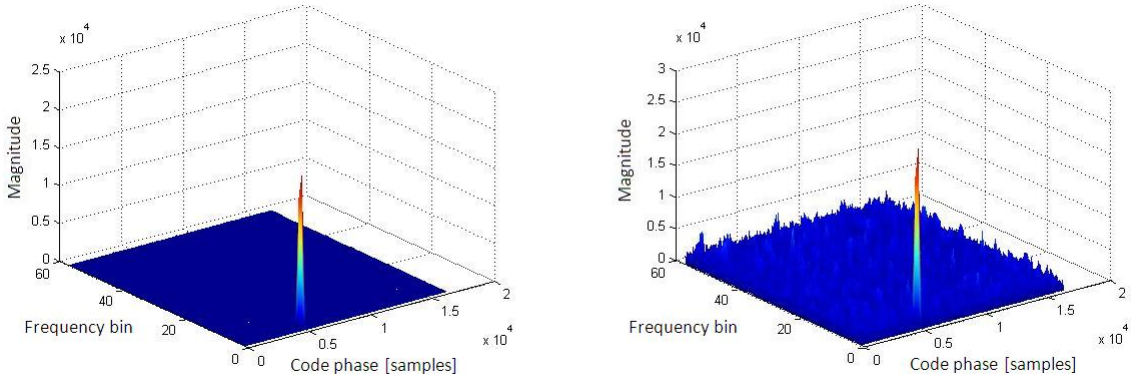

Figure 2.11: Simulated signal Parallel Code Phase space search acquisition results; a)without noise, b)with simulated noise

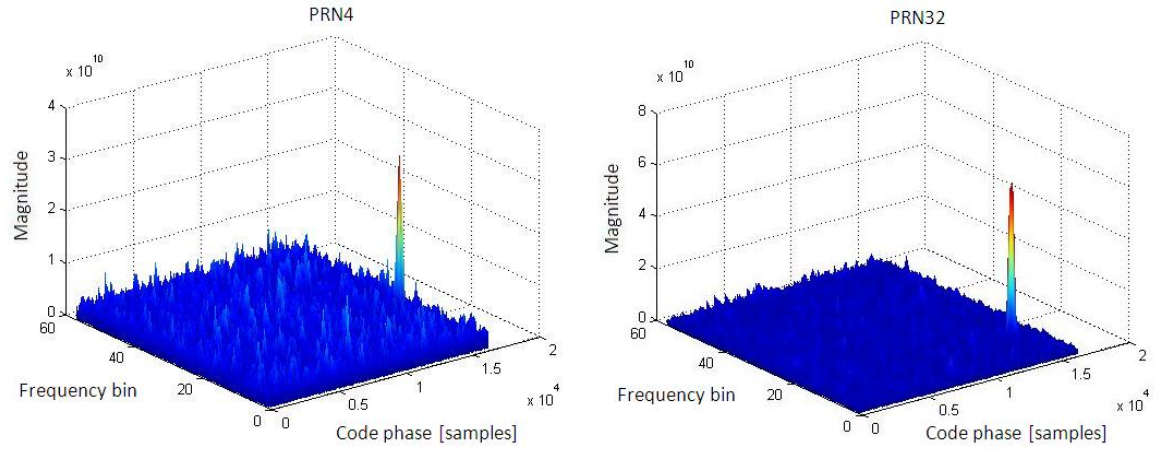

Figure 2.12: Results from Parallel Code Phase space search acquisition with real data

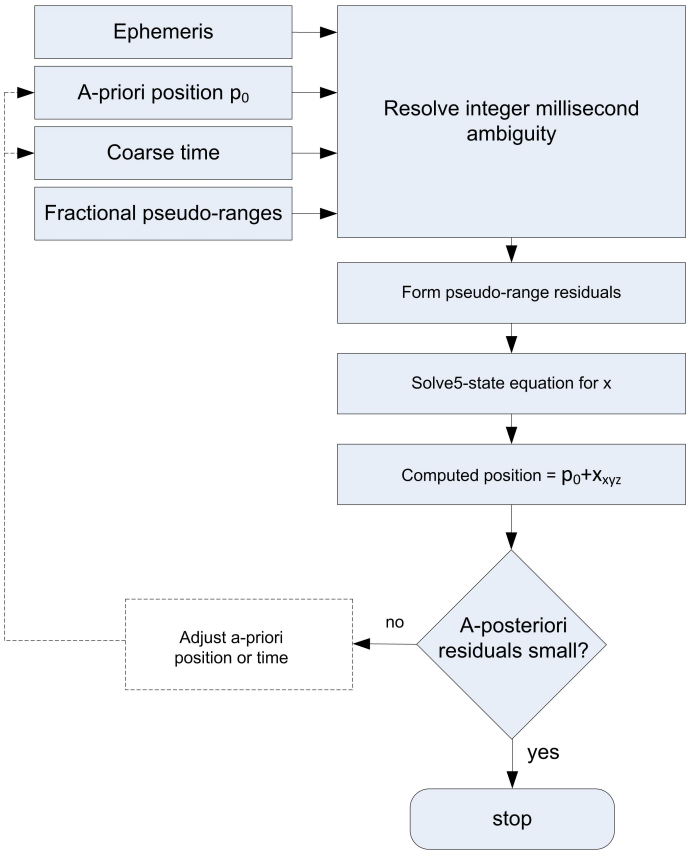

Figure 2.13: Flow chart of positioning algorithm [3]

acquisition success rate, more useful signal power can be recovered, false alarm reduction, higher detection probability and lower noise floors.

Even though we invested certain effort investigating this, we will not stress this aspect because it is beyond the purpose of this project.

2.3. Five State Position Computation Algorithm

This section contains an explanation of the algorithm used to achieve positioning without tracking. The algorithm used is described more in detail in [3]. The flow chart of this algorithm is presented in Figure 2.13.

[3] examines techniques to achieve sub-second first fix and to overcome situations where decoding time-of-week (TOW) from the handover word (HOW) is not possible due to signal weakness. This algorithm is also suitable while using snapshot techniques.

In navigation algorithms there are 4 main steps that need to be done:

- start with a-priori estimate of state

- predict the pseudo-ranges which would correspond to the above mentioned state
- take the actual pseudo-range measurements
- update the a-priori state by subtracting the predicted pseudo-range from the actual one

The term “state”is used to define quantities used in the algorithm such as position, time, velocity and others. The classic navigation algorithms use 4 components to define the state: three coordinates (x, y, z) and the receiver clock error (b), a common bias for all pseudo-ranges.

Time-keeping issues are crucial in GPS systems. An essential element of this algorithm is coarse time. By this we understand a time accuracy which is worse than $10\,\text{ms}$. This will yield a positioning error around or smaller than $8\,\text{m}$, an error which is considered acceptable. Therefore coarse time acts like the time accuracy threshold in order to achieve a position with acceptable accuracy.

The key of this algorithm is introducing a time bias over the coarse time, as a fifth state in the navigation equations. Thus allowing convergence towards a precise time starting from a coarse time. Further on we will present the mathematical description of this situation. We use superscript in order to relate certain values to different satellites, upper case for vectors, lower case for scalar values and bold format upper case for matrices. We start by defining the vector of a-priori state including coarse time as a fifth state:

$$\delta X = \begin{bmatrix} \delta_x \\ \delta_y \\ \delta_z \\ \delta_b \\ \delta_{tc} \end{bmatrix} \tag{2.1}$$

We continue by defining the vector of apriori measurement residuals (δZ) from satellite k as the difference between the vector of measured pseudo-ranges (Z) and the vector of predicted pseudo-ranges ($\hat{Z}$):

$$\delta Z^k = Z^k - \hat{Z}^k \tag{2.2}$$

$$\hat{Z}^k = |X^k(\hat{t}_{tx}) - X_0| - \delta_t{}^k(\hat{t}_{tx}) + b_0 \tag{2.3}$$

$\hat{t}_{tx} =$ estimated time of transmission of measured signal
$X^k(\hat{t}_{tx}) =$ calculated satellite position at $\hat{t}_{tx}$
$X_0 =$ a-priori receiver position
$\delta_t{}^k(\hat{t}_{tx}) =$ satellite clock error in unit of length at $\hat{t}_{tx}$

$b_0 =$ a-priori estimate of common bias

The only term affected by the coarse time in (2.2) is the vector of predicted pseudo-ranges, and this is described mathematically by (2.4). $-v^k\delta_{tc}$ represents the coarse time error.

$$\hat{Z}^k(\hat{t}_{tx}) - \hat{Z}^k(t_{tx}) = \hat{Z}^k(\hat{t}_{tx}) - \hat{Z}^k(\hat{t}_{tx} + \delta_{tc}) = -v^k\delta_{tc} \tag{2.4}$$

As shown in [3] the relationship between δZ^k and X is presented in (2.5). Notice that coarse time errors, common bias and all measurement errors are taken into account.

$$\delta Z^k = -e^k\delta X_{xyz} + \delta_b + v^k\delta_{tc} + \varepsilon^k \tag{2.5}$$

where:
$e^k =$ unit vector from receiver to satellite k
$\delta X_{xyz} =$ vector of spatial elements
$\varepsilon^k =$ measurement errors
$\delta_{tc} =$ update to a-priori coarse time state k
$v^k =$ pseudo-range rate

So far we are dealing with one satellite. It is a well known fact that in order to have a working GPS system you have to deal with a minimum of four satellites. Due to the fact that we introduce the coarse time as a fifth state, in the proposed algorithm for GPS snapshot techniques we have to deal with a minimum of 5 satellites. Continuing with this hypothesis we have to extend (2.5) for the number of visible satellites. It is very important to know that the common bias affects all measurements, all by the same amount, and the coarse time error affects only predicted measurements, each by different amount [3]. The next step is to introduce these equations in a matrix equation. This gives the opportunity to solve with the method of least squares. Therefore:

$$\delta Z = \mathbf{H}\delta x + \varepsilon \tag{2.6}$$

$$\mathbf{H} = \begin{bmatrix} -e^1 & 1 & v^1 \\ \vdots & \vdots & \vdots \\ -e^k & 1 & v^k \end{bmatrix} \tag{2.7}$$

We can solve for δx through a least squares solution:

$$\delta x = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\delta Z \tag{2.8}$$

2.3.1. Millisecond Ambiguity Resolution

As stated in section 2.3. we want to achieve positioning before decoding HOW. This implies that we are only dealing with fractional pseudo-ranges. Therefore arises the need to

reconstruct these pseudo-ranges. This need of reconstruction can also be observed from (2.2) because the vector of predicted pseudo-ranges $\hat{Z}^k$ has a value which is a multiple of milliseconds while the vector of measured pseudo-ranges Z^k has sub-millisecond values. Due to the fact that the common bias affects all measurements equally it must be taken into account that this common bias must be consistent for all measurements while reconstructing the pseudo-ranges.

In order to solve the millisecond ambiguity a reference satellite must be chosen. The reference satellite will be the one with the largest elevation angle. The next step will be to assign an integer value (N^0) to this satellite. The value chosen for N^0 was the value used in [3] which is obtained by rounding the difference between the multiple millisecond expected pseudo-range and the sub-millisecond measured ones 2.9. [3] also states that even if this value is not chosen in this way which makes pseudo-ranges as close as possible, it will not affect the outcome of the algorithm in a drastic way.

$$N^0 = round(\hat{Z}^0 - Z^0) \tag{2.9}$$

Next the full pseudo-range is reconstructed for the reference satellite:

$$N^0 + Z^0 = r^0 - {\delta_t}^0 + b + \varepsilon^0 = \hat{r}^0 + d^0 - {\delta_t}^0 + b + \varepsilon^0 \tag{2.10}$$

where:
$r^0 =$ actual geometric range
$\hat{r}^0 =$ geometric range from apriori position
${\delta_t}^0 =$ satellite clock error
$d^0 =$ error in $\hat{r}^0$
$b =$ common bias
$\varepsilon^0 =$ measurement errors

We rewrite (2.10) for the rest of the measurements:

$$N^k + Z^k = r^k - {\delta_t}^k + b + \varepsilon^k = \hat{r}^k + d^k - {\delta_t}^k + b + \varepsilon^k \tag{2.11}$$

After subtracting (2.10) from (2.11) we get:

$$(N^k + Z^k) - (N^0 + Z^0) = (\hat{r}^k + d^k - {\delta_t}^k + b + \varepsilon^k) - (\hat{r}^0 + d^0 - {\delta_t}^0 + b + \varepsilon^0) \tag{2.12}$$

$$N^k = N^0 + Z^0 - Z^k + (\hat{r}^k + d^k - {\delta_t}^k + b + \varepsilon^k) - (\hat{r}^0 + d^0 - {\delta_t}^0 + b + \varepsilon^0) \tag{2.13}$$

Because the common bias affects all measurements in exactly the same way it is plain to see that it cancels. Therefore we get:

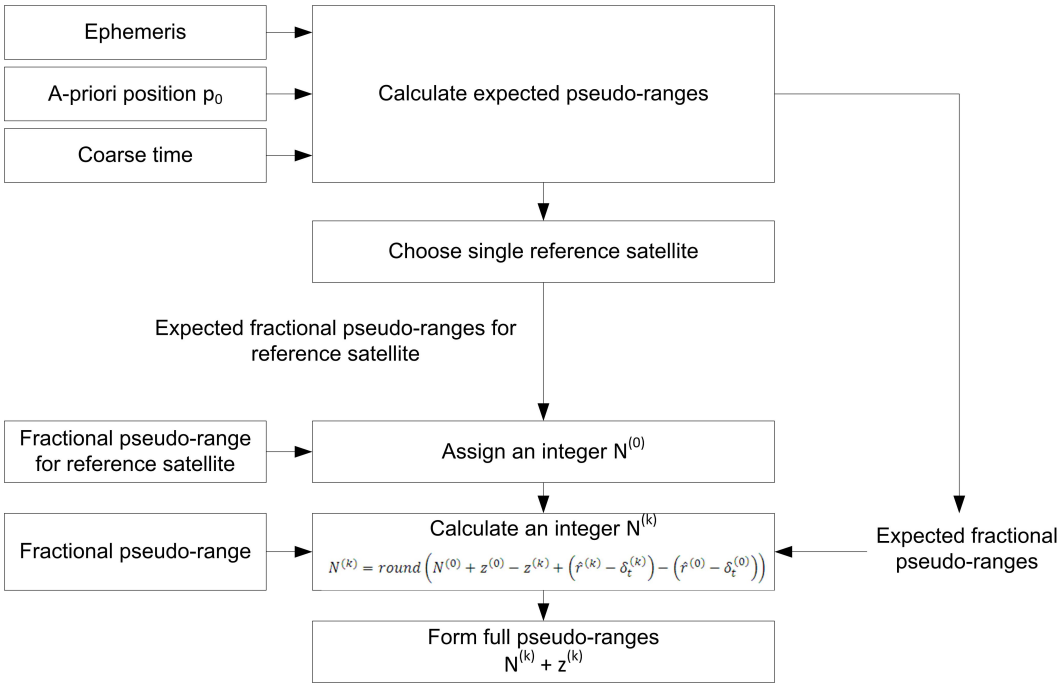

Figure 2.14: *Millisecond ambiguity resolution flow chart [3]*

$$N^k = N^0 + Z^0 - Z^k + (\hat{r}^k - {\delta_t}^k) - (\hat{r}^0 - {\delta_t}^0) + (d^k + \varepsilon^k - d^0 - \varepsilon^0) \tag{2.14}$$

The only term which is not known in (2.14) is $(d^k + \varepsilon^k - d^0 - \varepsilon^0)$ which represents the effects of measurement errors and of the errors in the predicted measurements. [3] describes in detail why this term can be excluded from our computations. Its value is less than half of millisecond so it can not be taken into account. This is how we get to the final form of this equation, equation which now can be solved because all variables are known or can be computed:

$$N^k = round[N^0 + Z^0 - Z^k + (\hat{r}^k - {\delta_t}^k) - (\hat{r}^0 - {\delta_t}^0)] \tag{2.15}$$

Finally we just have to form the full pseudo-ranges $N^k + Z^k$. Figure 2.14 shows the flow chart of the millisecond ambiguity resolution algorithm.

CHAPTER 3. EXPERIMENTAL RESULTS

This chapter describes the most interesting experiments realized during development. Basically, there are two main lines of experiments:

- acquisition methods performance
- sub-second positioning results

3.1. Acquisition Methods

Initially we focused our efforts on testing and studying acquisition algorithms, owing to the fact that they are a limiting factor on the performance of snapshot techniques, affecting both the pseudo-range accuracy and the satellite acquisition.

3.1.1. Execution Time Comparison

This section contains a comparison of the execution time for each of the three acquisition algorithms presented in 2.2.. All timing results where obtained using the Profiler tool integrated in Matlab and are based on a snapshot (taken on 02/03/2010 at 12:14:02 UTC) of 1 ms length, collected with the front-end presented in 2.1.. The search space was equally set for all algorithms, forming a 2D grid sweeping ± 14 kHz around the IF carrier with 500 Hz step and 1023 chip shifts of the C/A code. Obviously, if the search space and/or the resolution increases –this means reducing the frequency step, providing more frequency bins–, the execution time increases proportionally for all methods.

The results are presented in Table 3.1 in terms of elapsed time and number of loops to be executed by the routine.

Interpreting these results, one verifies the theoretical concepts described in 2.2..

Serial acquisition was ruled out for further testing in next subsections due to its massive execution time. In fact this was an optimized version of the algorithm. The execution time obtained for the first implemented version of the Serial acquisition algorithm was even

Table 3.1: *Acquisition methods performance comparison*

Acquisition method	Execution time [s]	Search space [loops]
Parallel Code Phase space search	27.465	57
Parallel Frequency space search	124.725	1023
Serial	1504.576	58311

Table 3.2: *Functions which influence execution time, and their contribution*

Acquisition method	Function	Contribution[%]
Parallel Code Phase	Fourier and Inverse Fourier Transforms	80.3
Parallel Frequency	Fourier Transforms	53.4
Serial	Local carrier wave generation	40

larger (3466.089 s). Even though providing correct solutions, it was clear that the code could be optimized. After debugging with Profiler, we noticed that there was high redundancy while generating copies of the local carrier waves, which significantly increased the code execution time.

In addition, we investigated the main factors conditioning the execution time for each algorithm. The results of this are presented in Table 3.2.

It turns out that the Fourier Transform (FT) and the Inverse Fourier Transform (IFT) operations, although resulting in faster algorithms, demand the greatest computational effort. On the other hand, in the Serial acquisition, most of the computational load was invested in generating the local carrier waves used to demodulate the signal from the carrier wave.

3.1.2. Acquisition Statistics

The following set of experiments is meant to illustrate the actual ability of the algorithms when it comes to detecting SVs. Basically there are three possible solutions resulting from the acquisition evaluation. A SV can be:

- detected – visible and acquired
- undetected – visible but not-acquired
- false alarm – not visible but acquired

Although theoretically there is a certain fixed amount of dBc between the main correlation peak and the nearest peaks, in real applications this value is unknown given the noise level and signal fading. Thus, in practice, all implemented algorithms decide upon acquisition using a threshold describing the ratio between the main and the secondary lobes –which is a way to measure carrier to noise ratio (C/N_0)–. If the actual ratio is bigger than the threshold, the algorithm interprets that a GPS signal is present.

Therefore, the selection of an appropriate threshold is a critical choice affecting the performance of the algorithm. In order to evaluate its impact and determine an optimal value, we have generated statistics about the three possible evaluation solutions presented above. The simulation was based on the same snapshot used in the previous subsection, and the acquisition results were collected along 100 consecutive milliseconds of the snapshot (the

Figure 3.1: Sky plot on 02/03/2010 at 12:14:05 UTC generated in Matlab from a RINEX file. A green star represents a healthy SV while a red star indicates unhealthy SV, the yellow line represents the elevation mask.

integration time being kept at 1 ms which produces 100 cases).

Thanks to the broadcasted 24h Receiver INdependent EXchange (RINEX) navigation files, one is able to recompute the position of any SV at a given time instant. This allows us to simulate the sky scenario by the time the snapshot was taken. The following figure shows a plot of the satellites in view on the 2nd of March of 2010 at 12:14:05 UTC in Aalborg University.

In order the check our SV positioning results, we used a trusted and widespread program among satellite observers, Orbitron.

Apart from a visual check, the numerical comparisons also showed the validity of our code. So, selecting an elevation angle of 0°, our scenario is composed by 11 SVs –PRN 1, 4, 11, 12, 13, 17, 20, 23, 30, 31 and 32– out of 32. The figure below shows the Parallel Code Phase acquisition results as a function of threshold.

It is quite interesting to observe the exponential behavior of the false alarms as soon as the threshold gets below the value of 2.5. Above it, the acquired signals decrease in favor of the non-acquired ones, so there is a clear compromise between the accepted false alarm level and the acquired and non-acquired SVs that sets an optimal threshold around 2.5.

Concerning the Parallel Frequency acquisition, the results showed in Figure 3.4 indicate that, regardless any threshold, its overall performance is inferior due to a higher level of false alarms and an earlier crossing point between acquired and non-acquired SVs.

The explanation of these results may rely on the GPS signal structure itself, since the code correlation presents better results than a frequency search in terms of absence of sec-

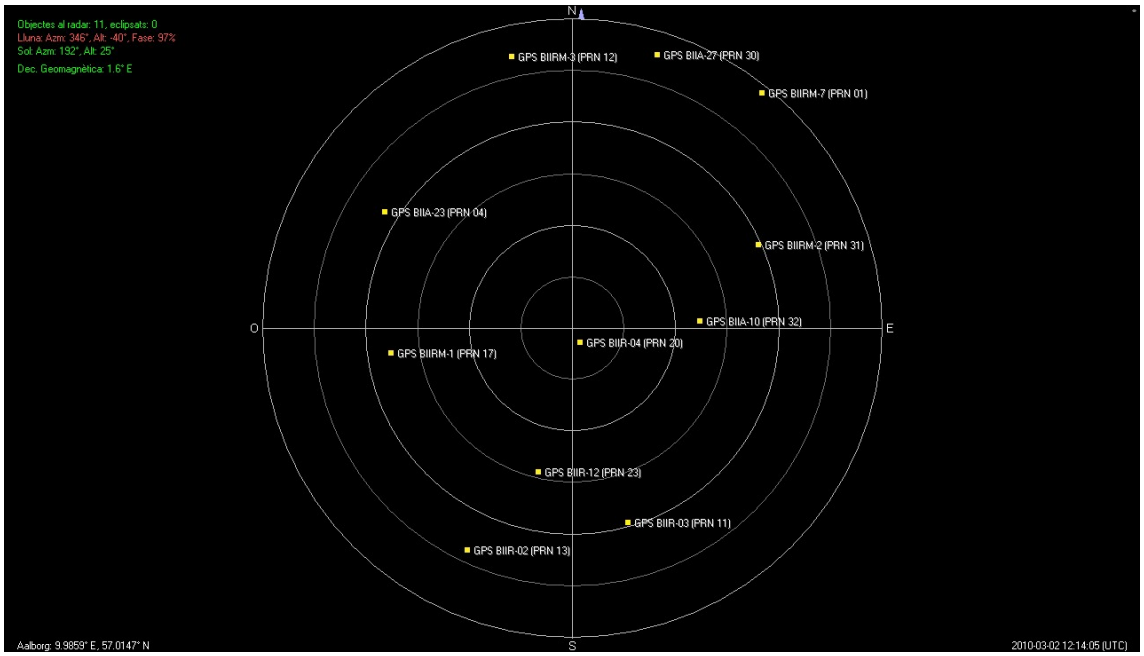

Figure 3.2: Sky plot on 02/03/2010 at 12:14:05 UTC simulated with Orbitron

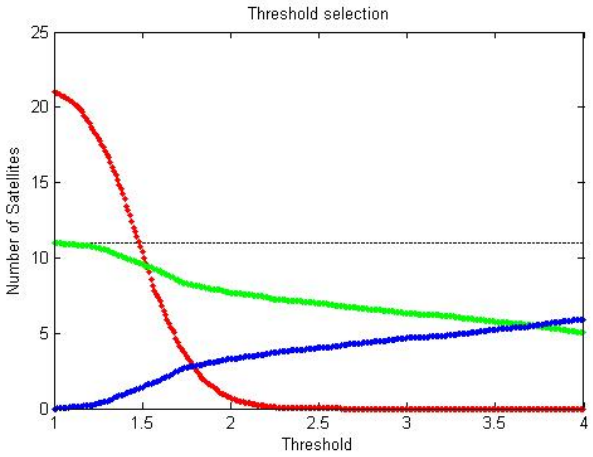

Figure 3.3: Parallel Code Phase acquisition results as function of the threshold value. Green represents acquired SV, blue indicates non-acquired SV and red means false alarms. The dashed line is the total number of visible SVs

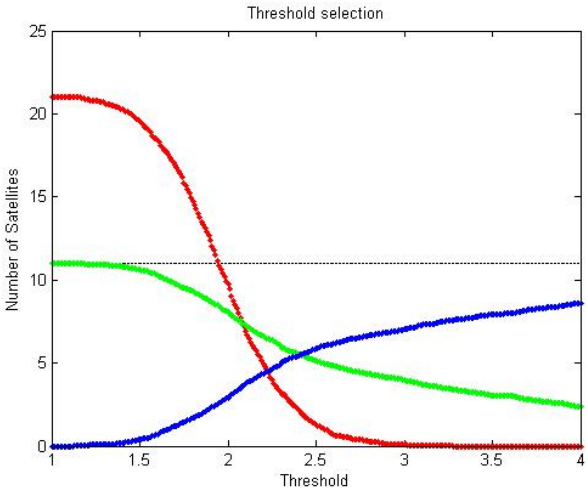

Figure 3.4: Parallel Frequency acquisition results as function of the threshold value. Green represents acquired SV, blue indicates non-acquired SV and red means false alarms. The dashed line is the total number of visible SVs

ondary lobes. As described in subsection B.2., the C/A code is a square wave that has triangular shape autocorrelation results. However, due to Fourier Transform properties, any time-frequency transformation computed from a carrier wave represented with a finite number of samples will result in a Sinc function –meaning the presence of signal lobes–. This fact implies that a lower threshold value shall be used in acquisition.

In order to illustrate this fact, we have simulated an ideal case of signal search without noise.

On the other hand, in a real situation there are some other inconveniences like the presence of white noise –quite visible in Figure 2.8– and the accidental generation of signal spurious during frequency down-conversion by the front-end. All these factors are contributing to a higher false alarm level.

Another interesting fact was to observe that the non-acquired SVs were exactly the ones with lower elevations, which fits with the fact that they are farther, and so, the signals are weaker, resulting in low carrier to noise ratio (C/N_0).

From here after, we decided to keep using only Parallel Code Phase acquisition since it is the fastest and the most reliable algorithm from the ones we have implemented and tested.

3.1.3. Enhancing Acquisition Performance

Basically, the acquisition performance is broken down into two different aspects:

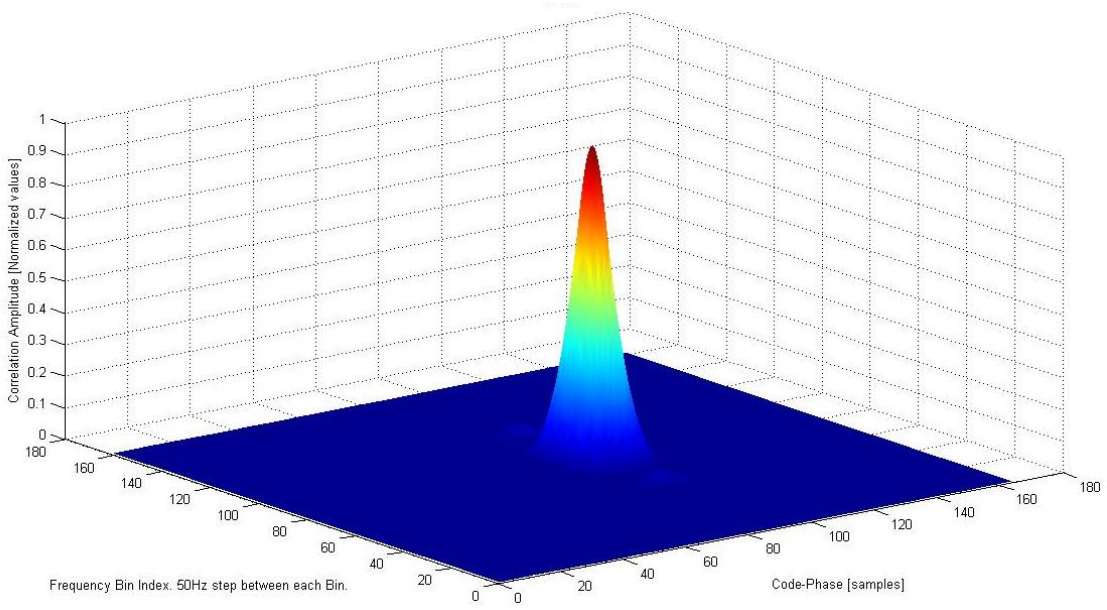

Figure 3.5: Frequency-Code phase search area. The code phase axis has been cut to around 163 samples out of 16368. The frequency axis is formed by 160 bins resulting from a ± 4 kHz bandwidth divided by 50 Hz steps.

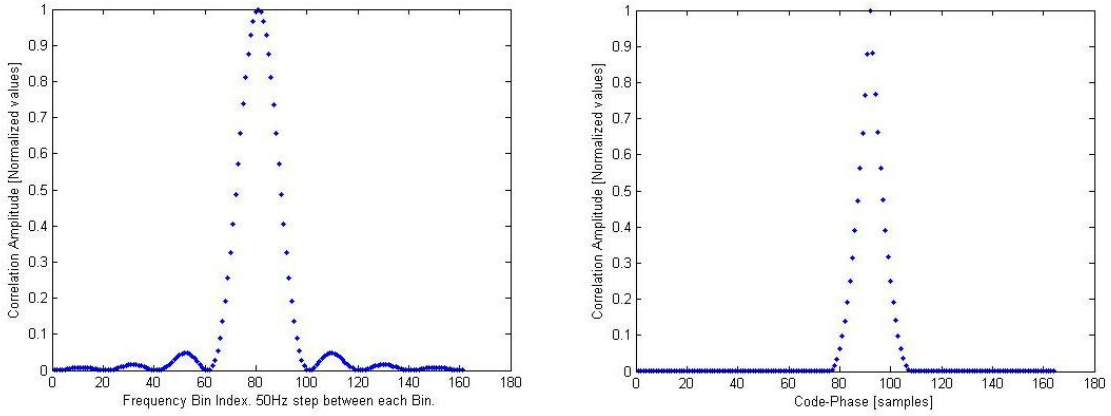

Figure 3.6: Both figures represent a cut containing the maximum peak but along different axis. Left: Frequency axis. Right: Code phase axis

- The ability of the algorithm to acquire line-of-sight satellites: The ideal scenario would be to detect all visible satellites without triggering any false alarms. This is the most important factor since the positioning accuracy is tightly related with the number of SVs (each one providing a pseudo-range measurement into the observations equation) and their geometrical dilution of precision (GDOP), which directly depends on this specific capability of the algorithm.

- Accuracy of code-phase and Doppler frequency values: Typically, acquisition is meant to compute rough values that would be used as "starters"for tracking stages, in order to refine them. However, in snapshot techniques tracking is omitted, thus the pseudo-range accuracy relies on acquisition. Nevertheless, given that our particular front-end offers a pseudo-range resolution $\approx 18\,\mathrm{m}$ (as stated in section 2.1.), we believe it will be good enough for our application.

Hence, we will focus on methods to improve acquisition performance in terms of the first aspect detailed above.

As we have described in 3.1.2., the threshold value is a way to measure C/N_0. So assuming that our threshold value is an optimal one –in the sense of detection to false alarm rate–, the only way to improve our acquisition algorithm is to, somehow, increase C/N_0 after correlation. A common technique for that purpose consists in power integration along several periods of the signal. This is possible assuming that code-phase and Doppler frequency values will not vary considerably from one period to the consecutive ones. Thanks to the short period of the C/A signal $-1\,\mathrm{ms}$– this conditions are ensured. On the one hand, the clock stability of a GPS receiver is required to be better than 10ppm [5] –this guarantees code-phase stability and reliable Doppler frequency measurements–, on the other hand, the Doppler drift rate, due to constellation dynamics, is way too small to create considerable differences from one period to the consecutive ones. Only high dynamics of the receiver could cause certain trouble but a snapshot taken from a photo camera cannot be considered to suffer from this issue. Therefore, we believe power integration to be a solid candidate to improve acquisition performance.

This subsection presents experimental results of non-coherent power integration implemented over Code-Phase Parallel acquisition. Their main benefits and drawbacks will be discussed and, finally, new settings of their parameters will be investigated for the purpose of our application on photo cameras. The non-coherent power integration was chosen for being the simplest form of power integration, given that is computed as the integral of the in-phase and quadrature power modulus, while a coherent acquisition shall be computed as a different power integral for each branch and further time-frequency transformations are to be considered.

All experimental results were computed using the same snapshot utilized in the previous subsection 3.1.2.. Due to the fact that we are dealing with discrete values, the power integration is in fact a summation of the signal powers obtained over several milliseconds.

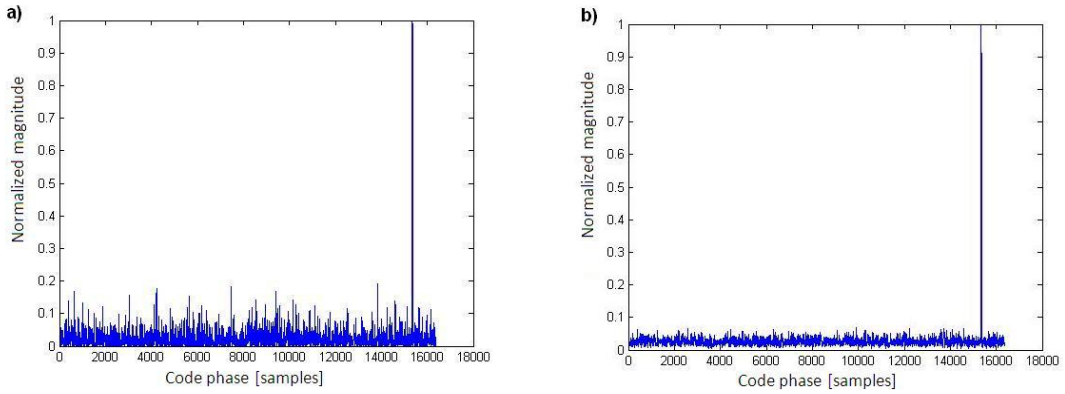

Figure 3.7: PRN4 code-phase axis after correlation ; a) Parallel Code Phase, b)Power Integration Parallel Code Phase

Table 3.3: *Visible satellites data*

PRN	Altitude [km]	Azimuth [°]	Elevation [°]	Minimum integration time [ms]
20	20084.680	151.8	85.2	1
32	20529.970	86.8	52.9	1
23	20320.940	192.9	47.0	1
17	20158.440	262.1	37.0	1
11	20326.390	163.9	31.2	1
31	20303.740	65.9	30.6	1
04	19995.160	301.8	26.1	1
13	20186.620	205.2	18.8	3
12	20255.390	347.5	9.2	4
30	19883.160	17.3	6.7	7
01	20081.220	38.9	2.1	–

Figure 3.7 clearly shows that the new acquisition algorithm improves C/N_0, which pretends to minimize the noise floor. In particular, its results were computed with an integration time of 20 ms –20 periods of the C/A code–. The plots show normalized values to avoid confusion induced by different absolute magnitudes resulting after each method.

In order to provide some numerical relation between integration time and C/N_0 improvement, Figure 3.8 shows the values of the peak metric (corresponding to PRN 32) obtained along different integration time intervals. On top of the experimental results, a logarithmic trend line showed to be the best fitting function. It is obvious that with the growth of the integration time, the peak metric value is increasing, tending a certain asymptotic value depending on the received power.

Consequently, by improving the acquisition sensitivity we were able to acquire three more satellites (PRN $12, 13$ and 30) than with the previous acquisition algorithm. During our experiments we observed that the apparition of these satellites is strongly dependant on the integration time. Table 3.3 shows the minimum integration time that was necessary in order to acquire certain satellites. Theoretically, as the elevation angle of the satellites decreases, the integration time should increase to compensate for its signal fading, fact which appears to be confirmed.

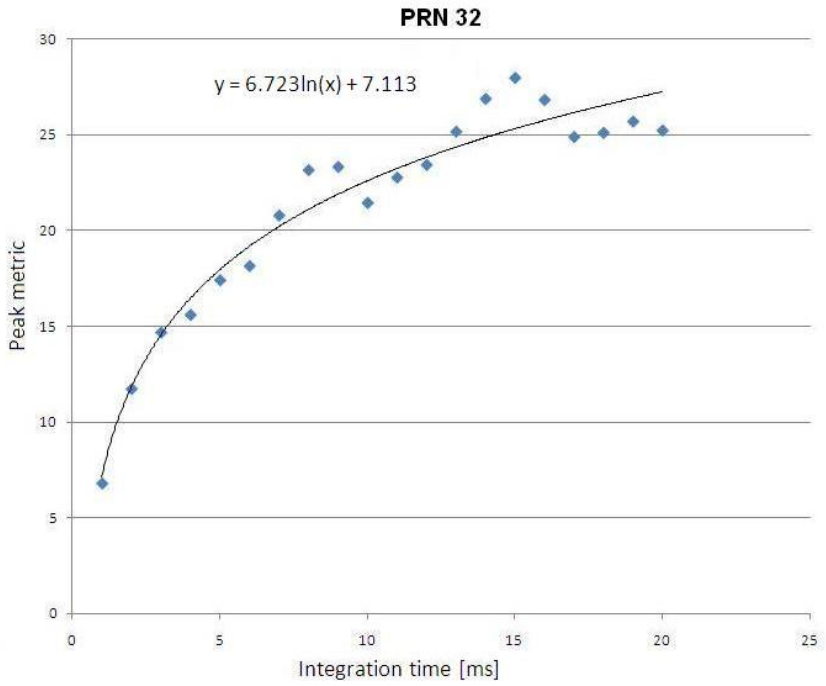

Figure 3.8: Peak metric vs. integration time

From these results we draw the conclusion that an integration time of 7 ms is enough in order to acquire 10 out of the 11 visible satellites shown in 3.2. And Figure 3.9 shows that 7 ms allows us to detect SVs with an elevation as low as $\approx 7^\circ$ over the local horizon, in this particular case, PRN30. Obviously, higher integration times will provide slightly better results but this will come to the price of higher execution times.

So after analyzing these results one concludes that 7 ms offers the best trade off between number of detected satellites and computation time. Due to its better performance we have used this method in our future experiments, and so we update Table 3.1 getting Table 3.4. The size of the search space in the case of the power integration algorithm comes from the multiplication of the original search space–the actual number of frequency bins– with the number of milliseconds to process.

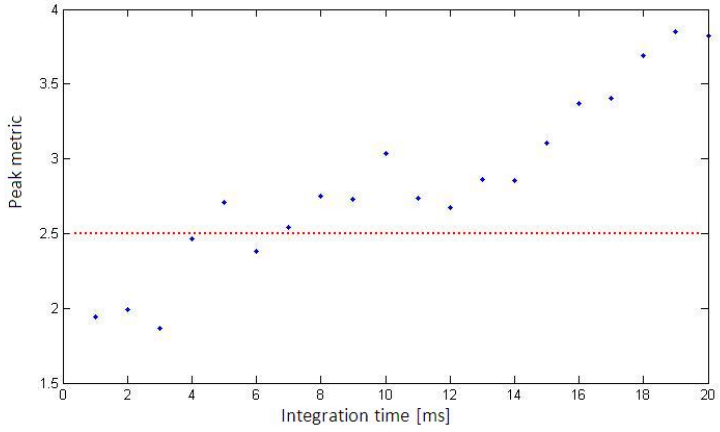

Figure 3.9: PRN30 peak metric versus integration time

Table 3.4: *Acquisition methods performance comparison–updated version*

Acquisition method	Execution time [s]	Search space [loops]
Parallel Code Phase	27.465	57
Power Integration Parallel Code Phase	136.355	399
Parallel Frequency	124.725	1023
Serial	1504.576	58311

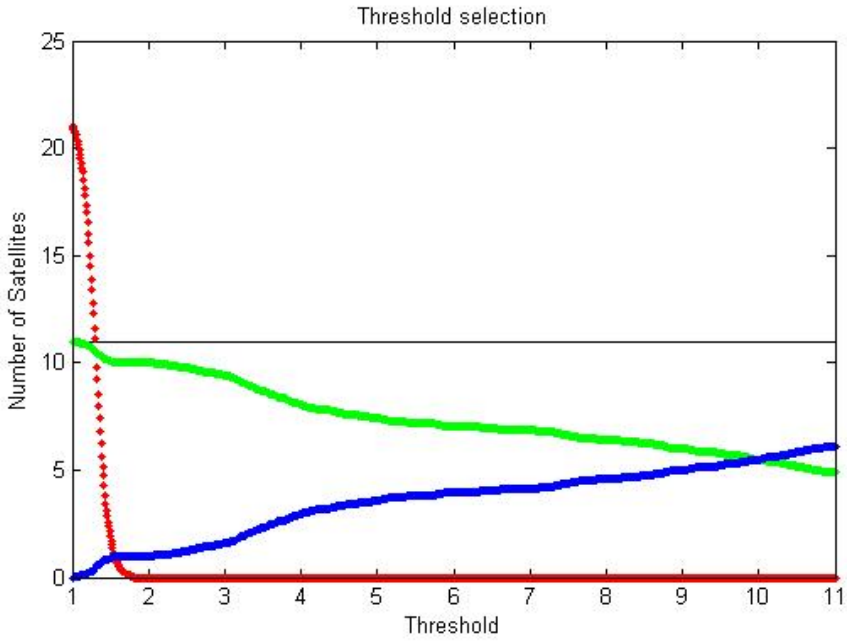

Figure 3.10: Power Integration Parallel Code Phase acquisition results as function of the threshold value. Green represents acquired SV, blue indicates non-acquired SV and red means false alarms. The black line is the total number of visible SVs.

After these results, we decided to repeat the statistical analysis presented in 3.1.2. in order to observe the number of acquired, non-acquired satellites and false alarms achieved by this algorithm as a function of the threshold value. Due to the longer execution time we collected results from 80 consecutive 7 ms samples of the snapshot, instead of 100 samples collected in subsection 3.1.2.. The obtained results are shown in Figure 3.10. It is interesting to notice the maximization of the "acquisition window"thanks to C/N_0 improvement after power integration –compare with Figure 3.3–, and that the false alarm exponential falls earlier, leading to a lower optimum threshold value than in the case of the Parallel Code-Phase or the Parallel Frequency. As it can be observed, in this case a more appropriate value for the threshold is 2. Statistically, this value provides ≈ 0 false alarms, and 10 acquired satellites versus 1 non-acquired satellite out of 11 –this satellite (PRN1) has a very low elevation angle as seen in 3.3–.

This proves that the power integration algorithm gives better performance than single period acquisition algorithms. So, despite its higher execution time, we decided to use the "Non-Coherent Power Integration Code-Phase Parallel acquisition"along all the positioning experiments presented below due to its reliability.

Table 3.5: *Snapshots specifications*

Tag	Date	Time (UTC)	GPS week	GPS second	Length [s]	Location
1-1s	02/03/2010	12h14m05s	1573	216860	1	DGC
1-50s	02/03/2010	16h35m02s	1573	225317	50	DGC
2-1s	25/03/2010	14h14m06s	1576	396801	1	CTAE
2-10s	25/03/2010	14h13m41s	1576	396836	10	CTAE
3-1s	27/04/2010	13h38m39s	1581	221934	1	DGC
4-1s	21/05/2010	08h35m39s	1584	462954	1	DGC

3.2. Positioning Results

Once an acquisition algorithm stable enough has been achieved, in the sense of results repeatability from one period of the snapshot to the following ones, it was time to prove the performance of the real core algorithm of the project, the coarse-time navigation presented in section 2.3.. This was the trickiest part of the project causing most of the implementation problems.

The experimental results were obtained from a small testbed formed by 6 different snapshots which are presented in Table 3.5:

Where,
DGC: 57.0147307° N 9.985904° E 59.998 m
CTAE: 41.314255° N 2.026309° E 68.7 m

Eventually, after overcoming several implementation problems, all snapshots except those taken in CTAE –2-1s and 1-10s– provided satisfactory results after running the coarse-time navigation algorithm. Initially, we believed that could be due to a mistake while accounting for beginning or end of week crossovers. Nevertheless, after a short debugging process, a second snapshot taken after half-week rollover, 4-1s, resulted in valid positioning results. Then, such programming error was ruled out, so we assumed that snapshots from CTAE were damaged or corrupted. Further details around this issue can be found in chapter 4 together with other implementation problems.

Thus, in order to obtain accuracy results, we have computed statistical results from different snapshots: 1-1s and 4-1s. For each snapshot, 80 fixes were computed from consecutive periods –the amount of samples was limited for the sake of simulation time–. Before representing the statistical results for each set of data, it is worth to pay attention to the sky plots in Figure 3.11.

The scenario on snapshot “1-1s”is formed by 11 SVs in a rather good distribution on the sky. Furthermore, we know from previous tests, that in such scenario, our acquisition routine is able to acquire 10 our of 11 of these satellites, which leads us to expect relatively good positioning accuracy caused by a favorable GDOP. On the other hand, the scenario on snapshot “4-1s”is also formed by 11 SVs but one of them was set as unhealthy, so it

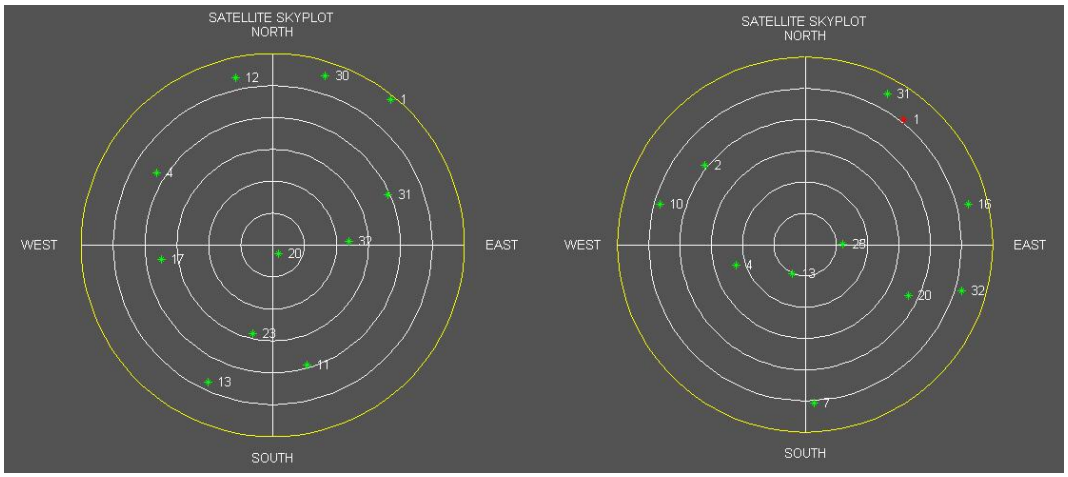

Figure 3.11: Left: sky plot for “1-1s”. Right, sky plot for “4-1s”. A green star represents a healthy SV while a red star indicates unhealthy SV.

Table 3.6: *Positioning statistics from snapshot “1-1s”. HPE stands for Horizontal Positioning Error. VTE means Vertical Positioning Error*

ID	μ HPE	σ HPE	2σ (95 %) HPE	μ VPE	σ VPE	2σ (95 %) VPE
1-1s	8.28	3.72	14.64	39.85	14.66	64.88
4-1s	29.47	11.38	47.48	11.81	15.93	38.14

was not used during coarse-navigation. Apart from that, the distribution of the satellites is not as uniform as in “1-1s”and we observed that acquisition function was detecting an average of 8 SVs, PRN1 among them, which leaves this scenario with a worst GDOP and 7 useful measurements. For these reasons, one would expect to find better accuracy using snapshot “1-1s”than “4-1s”.

The figures below show the accuracy results computed from both snapshots.

As predicted, the horizontal accuracy is much better for snapshot “1-1s”than “4-1s”results. However, the vertical accuracy of “1-1s”, although less dispersive, is worse than “4-1s”given to a height bias clearly visible in Figure 3.14 and Figure 3.16. Table 3.6 serves as a numerically summary of the accuracy results.

Besides these minor differences among snapshots, the overall accuracy fit with the requirements of our application and results from other studies [7] and theory [3].

3.2.1. Convergence Strength Solving for Millisecond Integer Ambiguity

In the previous paragraphs, some accuracy results have been presented after assisting the coarse-navigation algorithm with the proper ephemeris (from daily 30-second RINEX navigation file), and the actual time and coordinates of the taken snapshot. Nonetheless,

Figure 3.12: Positioning results plotted on Google Earth. Green dot: Actual position of the antenna. Red dots: fixes from snapshot “4-1s”. Blue dots: fixes from snapshot “1-1s”. All samples are grabbed to the ground level.

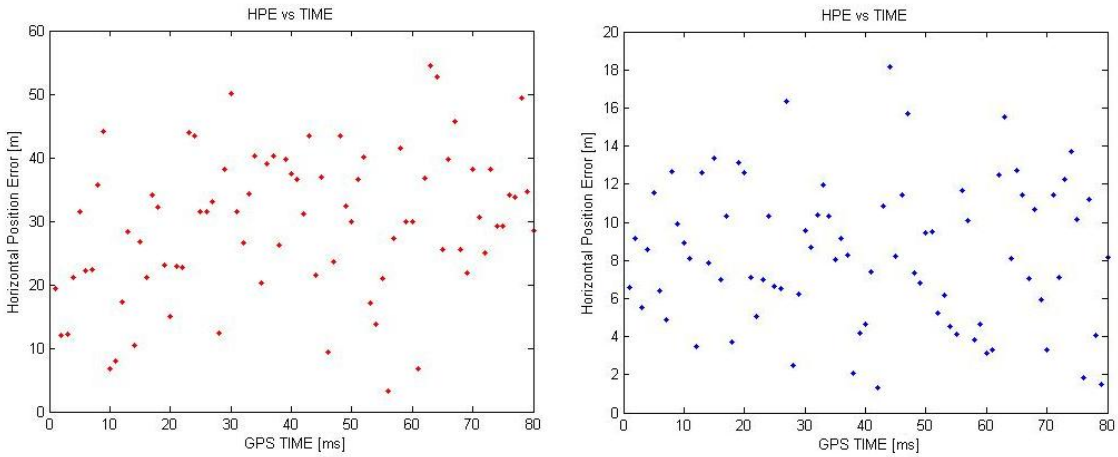

Figure 3.13: Horizontal Positioning Error. Red dots: fixes from snapshot “4-1s”. Blue dots: fixes from snapshot “1-1s”.

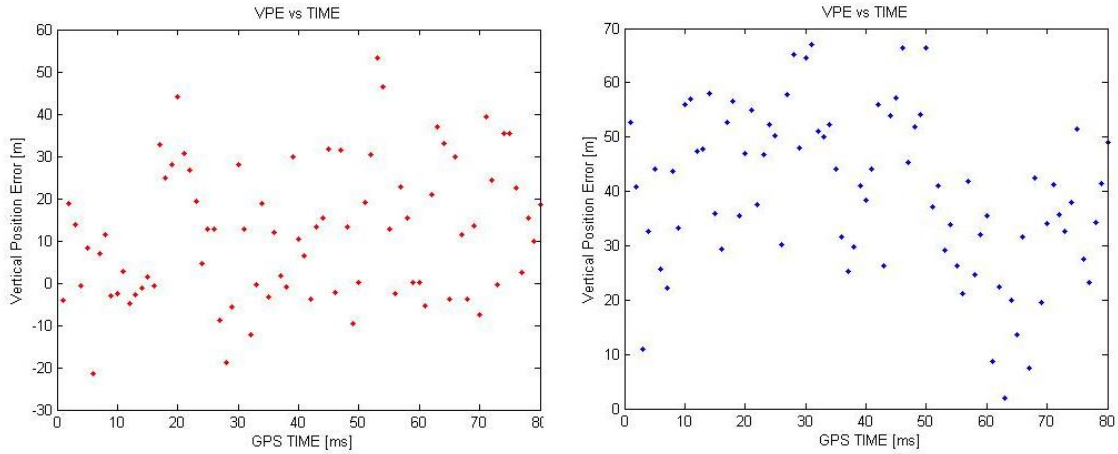

Figure 3.14: Vertical Positioning Error. Red dots: fixes from snapshot “4-1s”. Blue dots: fixes from snapshot “1-1s”.

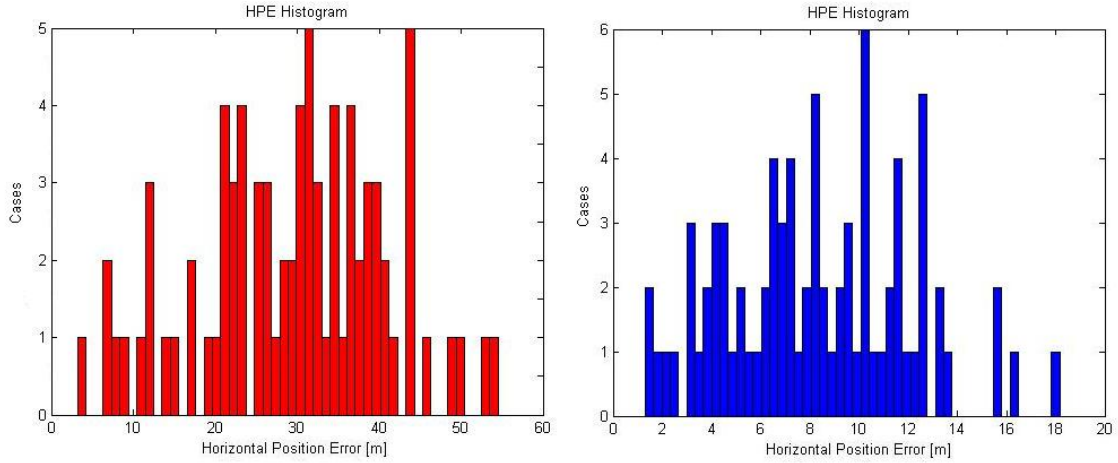

Figure 3.15: Histogram of Horizontal Positioning Error. Each bar represents the number of cases within the same error margin. Red: “4-1s”. Blue: “1-1s”.

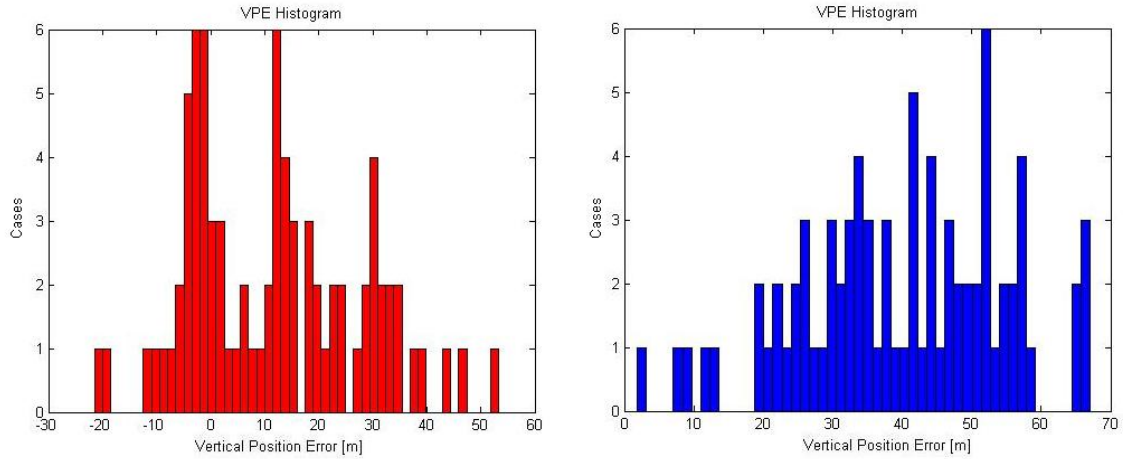

Figure 3.16: Histogram of Vertical Positioning Error. Each bar represents the number of cases within the same error margin. Red: “4-1s”. Blue: “1-1s”.

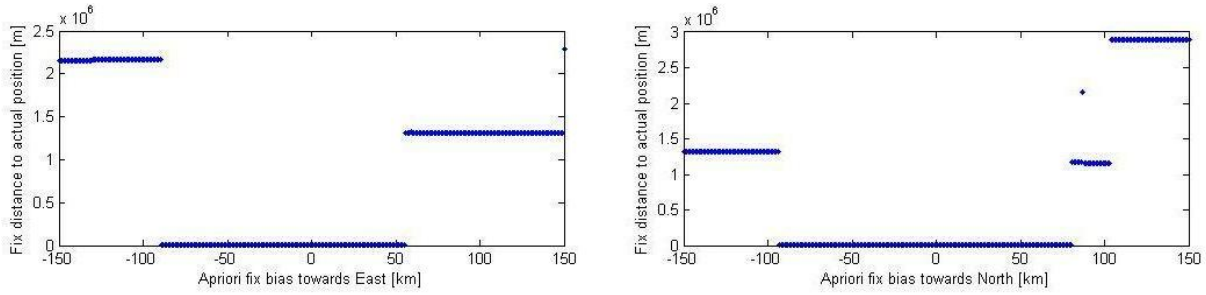

Figure 3.17: Distance between fix and real coordinates versus distance bias on apriori coordinates. 0s in the horizontal axis represents the actual coordinates where the snapshot was taken.

the greatest feature of the coarse-navigation algorithm is its convergence strength facing biased assisted time and coordinates. In other words, it is able to provide a fix with similar accuracy even if the assisted parameters are corrupted by a certain amount of error.

Theoretically, the boundaries of the assisted data that will cause incorrect millisecond integers are around 100km of bias on apriori coordinates guess and 1 minute of bias on the snapshot time [3]. The present subsection analyzes the flexibility allowed on these assisted parameters through experiments computed with real data.

3.2.1.1. Resistance Towards Wrong Apriori Coordinates

To test this aspect, the coarse-time navigation algorithm was feeded with what is believed to be the actual time when the snapshot was taken, plus an apriori component biased by an increasing distance. Owing to the fact that distance is just the modulus of a vector, we decided to run two simulations:

- the first one inducing a distance bias along North-South axis
- the second one through East-West axis, which would also provide us the means to evaluate whether the direction has any influence or not.

Figure 3.17 shows the fixes computed by the coarse-time algorithm after introducing a biased distance. It turns out that, for this specific snapshot, the algorithm is able to converge close to the actual position coordinates within a bias range compressed from 86km west until 55km east, and between 93km south and 80km north. Note that outside this ”convergence window“, the distance values diverge in discrete steps due to wrong integer millisecond ambiguity resolution.

Figure 3.18 gives a better understanding of the area within an apriori position which will drive the algorithm to a valid fix –we strongly believe anybody would be able to provide approximate coordinates of a photo within this distance range!–.

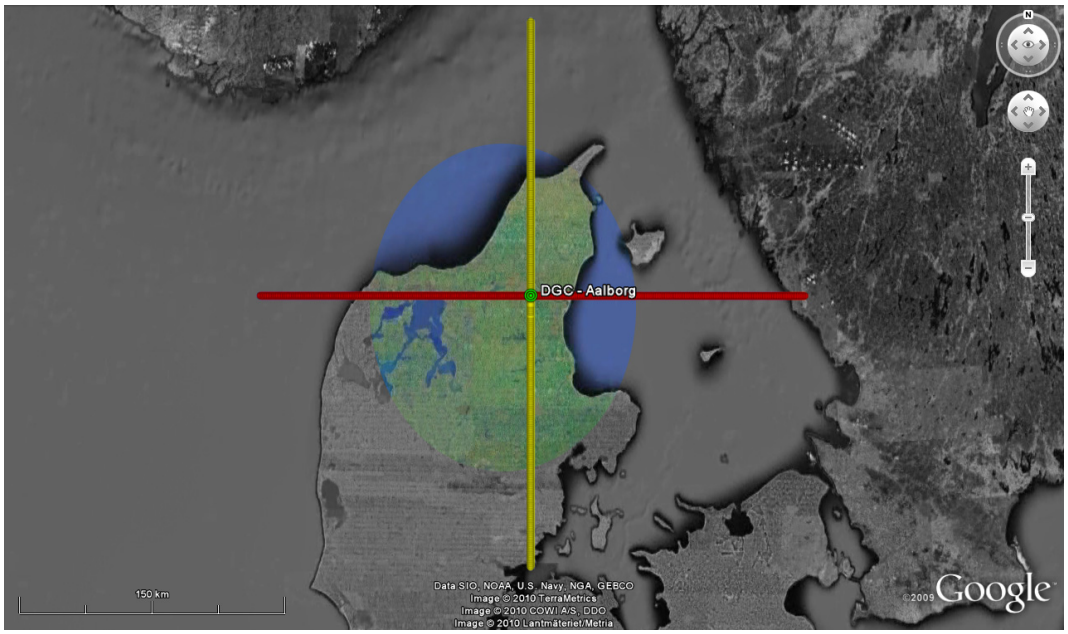

Figure 3.18: Google Earth plot of apriori coordinates used to test the algorithm. The green sample represents the actual position of the antenna. Yellow: samples with induced distance bias towards North. Red: samples with induced distance bias towards East. Distance biases take values within ± 150 km. Black and white area represents the apriori coordinates that would cause algorithm divergence.

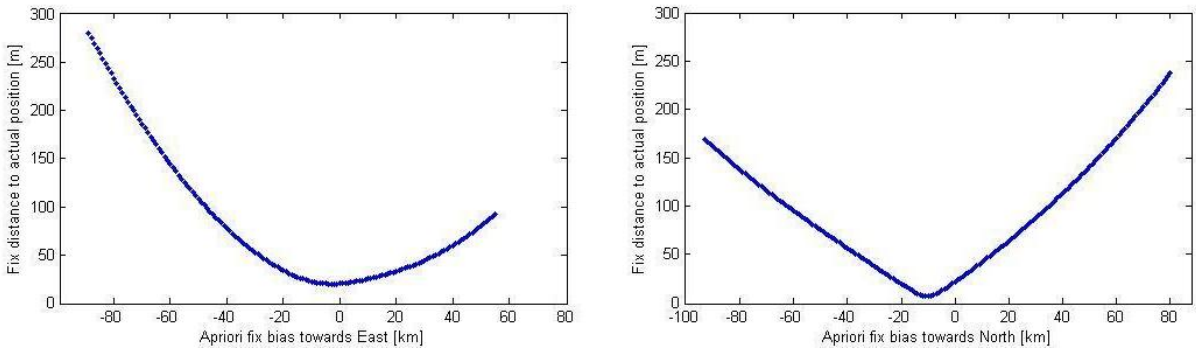

Figure 3.19: Zoom In to samples within the “convergence window”in Figure 3.17.

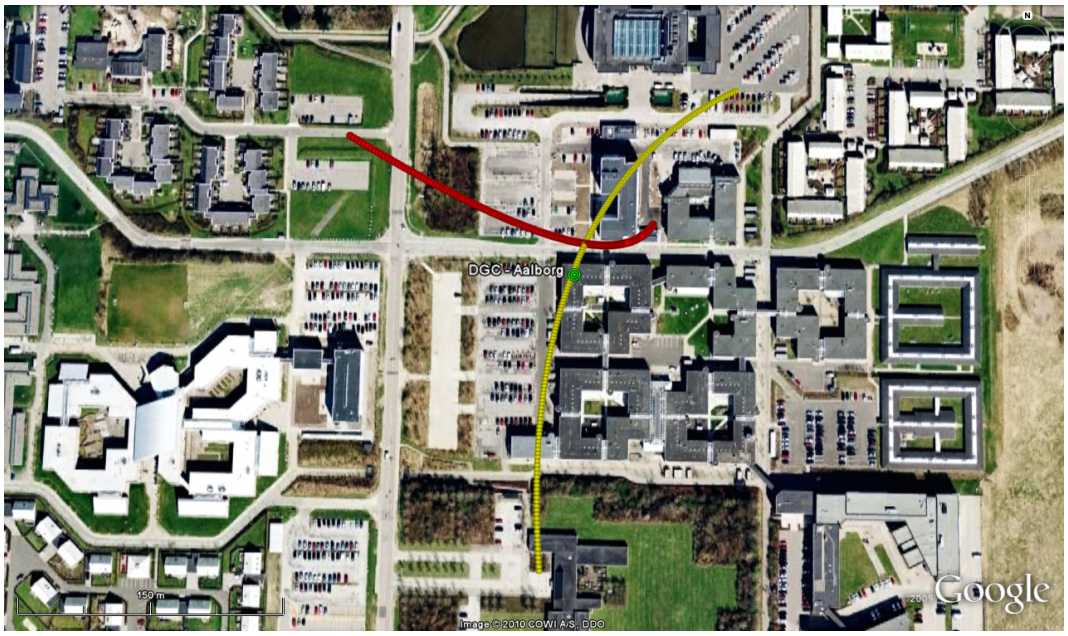

Figure 3.20: Google Earth plot of samples in Figure 3.19. The green sample represents the actual position of the antenna. Yellow: fixes from samples with induced distance bias towards North. Red: fixes from samples with induced distance bias towards East. All samples are grabbed to the ground.

Figure 3.19 zooms into Figure 3.17, and one is able to evaluate the accuracy of the positioning fixes, which in any case are better than $300\,\mathrm{m}$. Note that positions from a bias through north-south axis present an inflexion point close to the actual position. Unfortunately, this cannot be appreciated in Figure 3.20 since all samples are grabbed to the ground and such distance bias had a main contribution in height.

It is worth to say that other experiments for different snapshots showed similar behaviors and accuracies. So, we conclude that the algorithm responds to bias on apriori coordinates according to theoretical hypothesis.

3.2.1.2. Resistance Towards Bias on Assisted Time

The tests which will follow are based on the same concept used in the previous results but, in this case, the coarse-time algorithm was supplied with the actual coordinates of the antenna plus a time component increasing its bias.

Analogous to Figure 3.17, Figure 3.21 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 $\pm 100\,\mathrm{s}$, which is quite a long time, more than 3 minutes. The Figure below plots all fixes achieved from biased time guesses. In any case, its error exceeds $700\,\mathrm{m}$, but other snapshots returned slightly bigger errors, which demonstrates that providing a “precise”assisted time is much more demanding than an accurate apriori position fix.

Perhaps this would be one of the most problematic issues in the real application, since

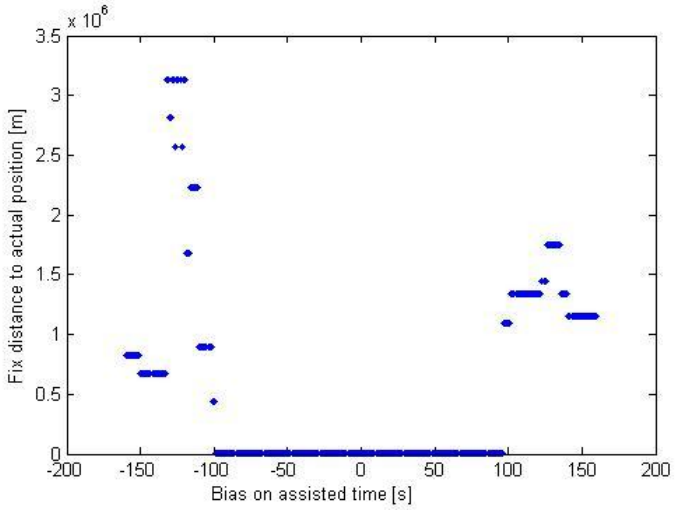

Figure 3.21: Distance between fix and real coordinates versus bias on assisted time. 0 s in the horizontal axis represents the actual time when the snapshot was taken.

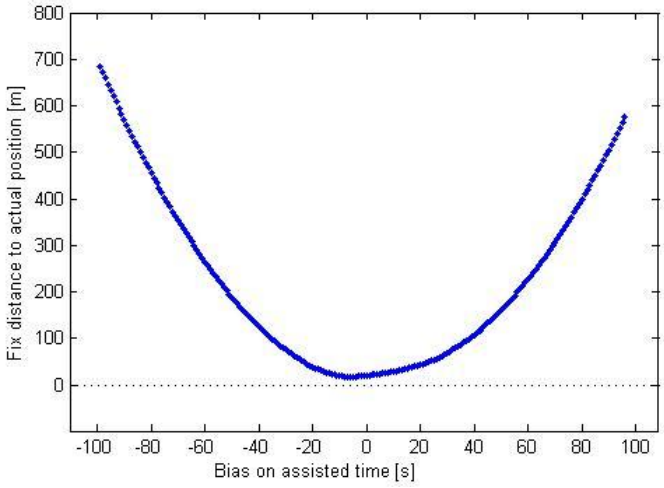

Figure 3.22: Zoom In to samples within the “convergence window”in Figure 3.21.

it means that the clock of the camera shall be coordinated with GPS time within this time span. Assuming than the clock of the camera is stable enough to acquire GPS signals, which required better than 10 ppm [5] time-keeping accuracy, in the worst case, after synchronization, the clock would exceed the time boundaries $-\pm 90$ seconds– after ≈ 100 days. Nevertheless, the internal clock could be re-synchronized each time the camera is connected to a laptop, computer or, alternatively, coordinated with the GPS signal itself by decoding the TOW from time to time.

CHAPTER 4. IMPLEMENTATION DISCUSSION

This chapter will present some of the issues and problems we encountered while working on this project. Some of these were successfully solved while some of them require further work and investigation.

Firstly we would like to talk about our time organization. Although we know that acquisition is an important issue in our project, we feel that we invested too much time on it. Therefore we believe that we should have been a little bit more organized or guided from this point of view, and we should have spent more time on the core of the problem which is the coarse-time navigation positioning algorithm [3].

While implementing the coarse-time algorithm certain problems came up. Further on we would like to describe some of these issues and the way in which they were solved.

One of the first problems we encountered was due to the fact that we made a programming error while computing satellite speed. In order to identify the problem we made a comparison with the trusted Orbitron software, which, among other data, also provides information about pseudo-range rate –what we actually aim to compute–. While programming this algorithm, GPS toolkit was used as a main reference, which is an open C library. A detailed description of this library can be found in [14]. After debugging the obtained results with the ones provided by Orbitron and after consulting the GPS toolkit we realized that our problem consisted in a trigonometric error on a rotation matrix.

After solving the satellite speed computation error we found that our algorithm was still not providing satisfactory results. Eventually we realized that while computing the estimated pseudo-ranges we committed a mistake applying the satellite clock corrections. The bug was not made in the clock correction algorithm itself, but it was an indexing error affecting the corrections which were applied to each satellite measurements. Summarizing, we were computing proper clock corrections for all satellites, but we were applying only one of them to all satellites.

These were some of the problems we had which were worth mentioning. Fortunately we succeeded to resolve them, resulting in a working algorithm. On the other hand we still have an unresolved issue. For testing purposes we took a snapshot in Viladecans, Spain (41.314255°N 2.026309°E). While the algorithm was working with snapshots taken in Aalborg from the DGC (57.014733°N 9.9859139°E) we obtained bad results with the one taken in Spain. At first we thought that we are dealing with an error related to time, due to the beginning or end of week crossover, because the snapshot was taken on a Thursday –after half week–. After applying the necessary corrections, the algorithm was still not providing satisfactory results. To verify that the corrections we made were correct we also took a snapshot on a Friday from the DGC. The obtained results were satisfactory which lead us to think that the problem must have occurred while getting the snapshot,

most probably due to a hardware failure. Unfortunately we could not take the snapshot from Spain again, therefore we can not have a precise answer about this issue.

CHAPTER 5. CONCLUSIONS

The present project aimed to study and implement algorithms for GPS snapshot techniques. To serve that purpose more specifically, we focused ourselves on a single application, digital photography geo-tagging.

Summarizing, our problem scope 1.4.3. stated three specific aspects to be implemented or studied along this project: positioning accuracy versus snapshot length, acquisition performance versus processing load and degree of assistance. Judging by the achieved experimental results, we believe that at least the two last aspects have been covered. On the other hand, positioning accuracy versus snapshot length was covered, somehow indirectly, by the statistical acquisition results. In other words, this was not achieved by refining code-phase and frequency values, but by improving its satellite detection capability through power integration.

Concerning acquisition results, we may conclude that the power integration –at least the non-coherent one– is the most effective technique in terms of statistical stability. Experiments showed that the threshold value shall be 2 and the integration period shall be 7 ms –which fits with real implementations like FastLoc^{TM}; 8 ms in that case–, in order to get optimal results. Furthermore, despite being slower than single period integration, it is much faster than Frequency Parallel and Serial acquisition.

Positioning results, on the other side, have shown that the coarse-time navigation algorithm provides accuracies within the expected range $-20-50$ m– and it is strong enough to converge under biased assisted data –about 100 km error in apriori guess and ± 90 s bias in snapshot record time–. However, time bias has a bigger repercussion than apriori guess in the algorithm's outcome. So it was shown that GPS snapshot techniques can be used in order to obtain positioning with acceptable accuracy in order to be used as photo-taggers on digital cameras.

From an academical point of view, working on this project lead us to study GPS related processing concepts that were not covered in depth in the last semester, therefore aspects which were new to us.

The reader should take into account that this is a student project, therefore not all technological aspects have been studied or applied. 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 6.

CHAPTER 6. FUTURE WORK

In this chapter we would like to present what future work should be done in order to get closer to the possibility of creating a new product, a product which would make geo-tagging on digital cameras possible in a cheap way, with a minimum of required hardware .

One of the things we still consider that it has to be done is a visual interface. Because the final product is addressed to a wide public this interface should be intuitive and easy to use, allowing even a non-trained user to be able to work with it. It would be interesting if the user is given the possibility by this interface to plot results directly on specialized software such as Google Earth.

As stated before, the final product is not addressed only to GPS experts and engineers. Therefore the users' work should be minimized. This is why some software should be developed that is able to detect the snapshot meta-data and automatically download all the necessary assisted data. This can be done in a transparent or visible way, according to the needs of the user.

Right now the smallest snapshot we are dealing with is 1 s long which needs 16 MB storage capacity. This brings up the need of large storage spaces if we think of the number of photograph which can be taken with a digital camera today. The length of the actual snapshots is controlled by the driver of the front-end. As shown we just need 11 ms of data in order to achieve positioning within our expected accuracy range $(20-50\,\text{m})$. Therefore the driver should be modified in order to be able to get just 11 ms long snapshots. This would greatly minimize the needed storage space.

These are just some of the possible improvements which we consider that are to be made in order to obtain a product which makes geo-tagging on digital photographs possible.

APÈNDIXS

APPENDIX A. LIST OF ACRONYMS

Table A.1: *Abbreviations used in the report in alphabetical order*

Abbreviation	Definition
AGPS	Assisted GPS
AGC	Automatic Gain Control
ASIC	Application-Specific Integrated Circuits
C/A	Coarse/Acquisition
C/N_0	Carrier to Noise Ratio
CDMA	Code Division Multiple Access
CPU	Central Processing Unit
DSP	Digital Signal Processors
EGNOS	European Geostationary Navigation Overlay Service
FPGA	Field-programmable gate array
FT	Fourier Transform
GDOP	Geometrical Dilution Of Precision
GPP	General Purpose Processors
GPRS	General Packet Radio Service
GPS	Global Positioning System
GPST	GPS Time
GNSS	Global Navigation Satellite
IC	Integrated Circuit
IDC	International Data Corporation
IFT	Inverse Fourier Transform
LNA	Low Noise Amplifier
PC	Personal Computer
PPS	Precise Positioning Service
PRN	Pseudo-Random Noise
RF	Radio Frequency
RINEX	Receiver INdependent EXchange Format
SA	Selective Availability
SBAS	Satellite Based Augmentation Systems
SDR	Software Defined Radio
SoC	Systems on a Chip
SPS	Standard Positioning Service
SNR	Signal to Noise Ratio
SV	Space Vehicle
TLM	Telemetry label word.
TTFF	Time To First Fix
USB	Universal Serial Bus
WAAS	Wide Area Augmentation System
WLAN	Wireless Local Area Network

APPENDIX B. BRIEF SIGNAL DESCRIPTION

GPS provides two levels of service a Standard Positioning Service (SPS) for general public use and a Precise Positioning Service (PPS) primarily intended for use by the Department of Defense. This leads to the use of channel access methods for blocking unauthorized users. Specifically, GPS signals are modulated using Code Division Multiple Access (CDMA) techniques. Thus, the original spectrum contained 3 different signals allocated over two carriers:

- L1 at 1575.42 MHz
- L2 at 1227.60 MHz

The PPS service is formed by two encoded signals: P1 and P2 located over L1 and, respectively, L2 –'P' stands for Precise code–. Dual-frequency signals allow PPS users to correct for the ionospheric delay, one of the main error sources of the system. Only authorized users know the code words, allowing them to recover signals from encryption. On the other hand, civil SPS consists on a unique signal modulated with public Coarse/Acquisition (C/A) codes over L1.

Even though DoD foresees the introduction of new civil and military signals in the forthcoming years –usually referred as GPS modernization–, our project focuses exclusively towards the use of the C/A code signal over L1, owing to the fact that this is the only operative civil signal and dual-frequency front-ends are currently not feasible for mass market applications.

Therefore, next subsections will provide further details of C/A signal structure along their generation steps. Understanding its generation turns out to be a key point to understand its demodulation in the receiver.

B.1. C/A Code Generation

CDMA techniques imply a set of pseudo-noise words or code words. Each word is formed by a certain number or chips –a chip represents a bit in the context of a pseudo-noise word–. These words serve the purposes of any spread-spectrum technique: spread the signal along the frequency domain increasing its bandwidth, thus increasing resistance to natural interference and jamming, and to limit power flux density (note that in figure B.1 the received signal power is below the noise floor).

Gold codes are used due to their low cross-correlation among the 32 different sequences assigned to GPS satellites and their relatively simple implementation. The C/A code has a chip rate of 1.023 Mcps and a code period of 1023 chips. The somewhat unusual code rates of 1.023 Mcps are selected so that the period of the C/A code corresponds exactly to

1 ms for time-keeping purposes [4].

The block diagrams below show C/A code generation step-by-step:

Besides the C/A code, the real content of the signal is the navigation message. So it is important to note that this is the only source of information, whereas the C/A code is just a manner to spread the signal and provide the time-transfer feature between the satellites and the receivers.

This report does not include an explanation of the navigation message content because snapshot techniques do not require their demodulation. However a detailed description of its contents can be found in [5].

B.2. C/A Signal Modulation

Any signal to be transmitted or radiated through air and space must be modulated into Radio Frequency (RF). This creates the need of modulating a digital signal over a carrier wave. For that purpose, GPS uses Binary Phase Shift Key (BPSK), which is a kind of angular modulation that consists on phase variation of the carrier wave by two discrete values.

The figures above summarize the modulation process. The first one is a representation of the mathematical/logical operations and the second one shows a graphic explanation of the signal shape and transformation on each step, especially useful to comprehend the BPSK modulation.

B.2.1. C/A Code Correlation Proprieties

The most important characteristics of the C/A codes are their correlation proprieties [1]. By correlation we can understand a parameter which describes the degree of independency between signals in the receiver. If the value of this parameter is high (in our case maximum 1023) that means that the signals have a high degree of dependency to one another. If the value reaches the maximum value it means that the signals are identical. It is obvious to state that if the value of the parameter is low, the degree of independency is high. In GPS, when dealing with correlation, high values represent a good alignment of an incoming signal with a locally generated code. As presented in [1], when talking about correlation we are dealing in fact with two types of coefficients:

- Cross-correlation
- Autocorrelation

Cross-correlation describes the measure of similarity between two different signals as a function of time-lag applied to one of them. In GPS the perfect scenario is that codes have a degree of cross-correlation of 0 with other codes. That is why C/A codes are used. They are nearly uncorrelated which each other [1]. Equation B.1 shows the cross-correlation function of two C/A codes coming from two different satellites. The value m is actually the time-lag introduced previously and it can have any value.

$$\sum_{l=0}^{1022} C^i(l)C^k(l+m) \approx 0 \tag{B.1}$$

Autocorrelation is in fact cross-correlation of a signal with itself. This will result in a peak only at 0 lag. This is essential in acquisition because it makes identification of satellites possible, indicating that the generated code is perfectly aligned with the incoming signal. Besides 0 lag the C/A codes are nearly uncorrelated with themselves. This can be described mathematically by equation B.2. This time we are deling with the same signal from the same satellite. As in equation B.1, m represents the time-lag. As we can see the expression is true only for values of the lag different from 0, which means that the codes are unaligned.

$$\sum_{l=0}^{1022} C^k(l)C^k(l+m) \approx 0, |m| \geq 1 \tag{B.2}$$

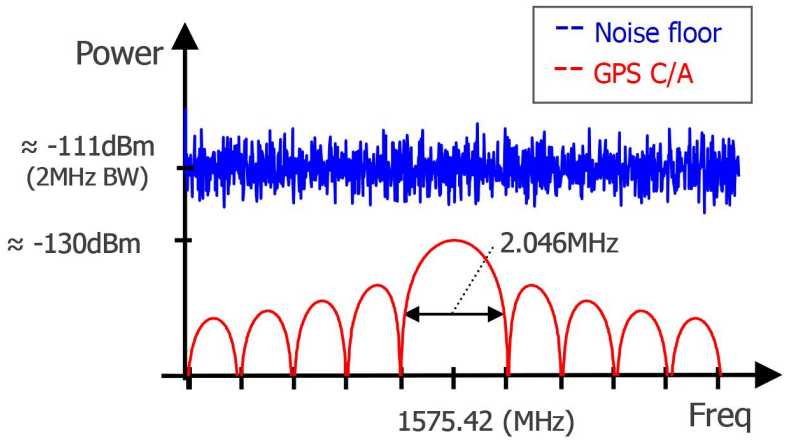

Figure B.1: *CA spectra as seen from the receiver's antenna.*

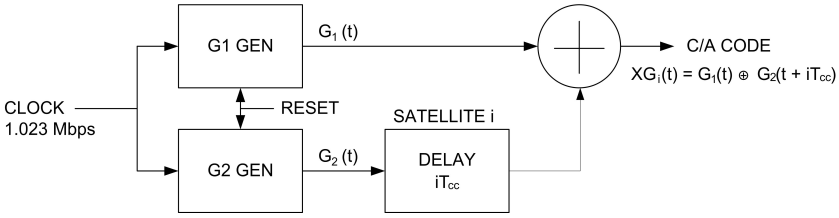

Figure B.2: *GPS code generators for satellite i [4]*

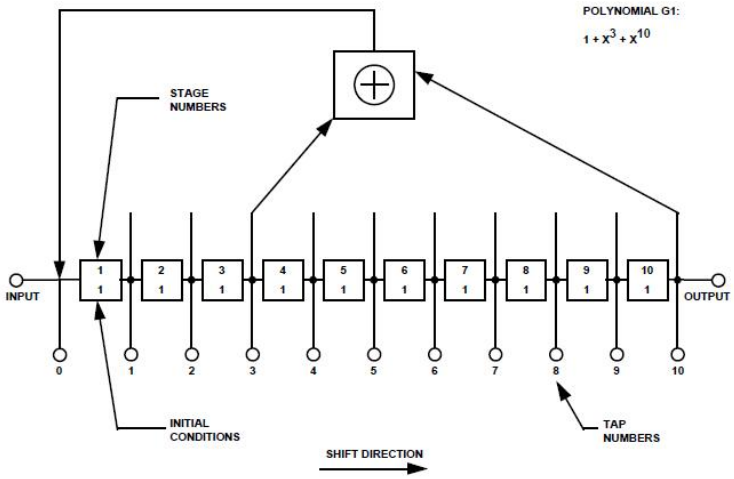

Figure B.3: *G1 shift register generator configuration [5]*

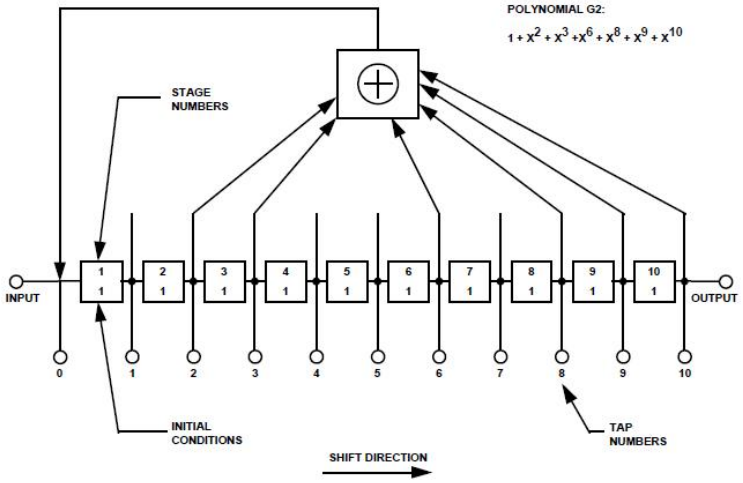

Figure B.4: *G1 shift register generator configuration [5]*

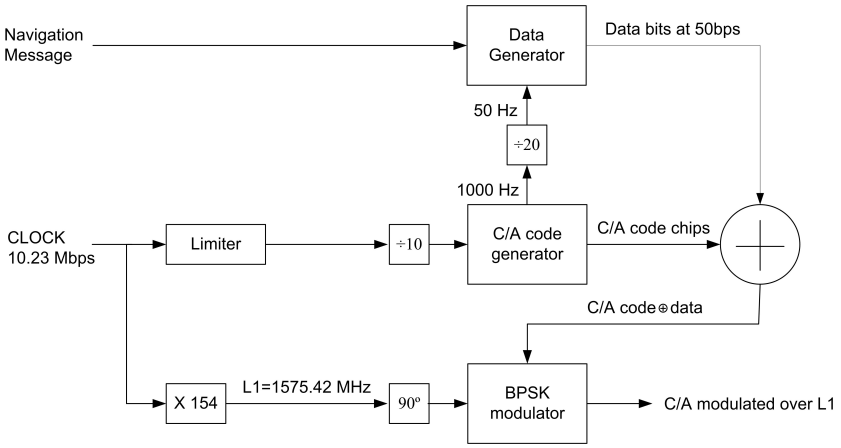

Figure B.5: *C/A signal modulation*

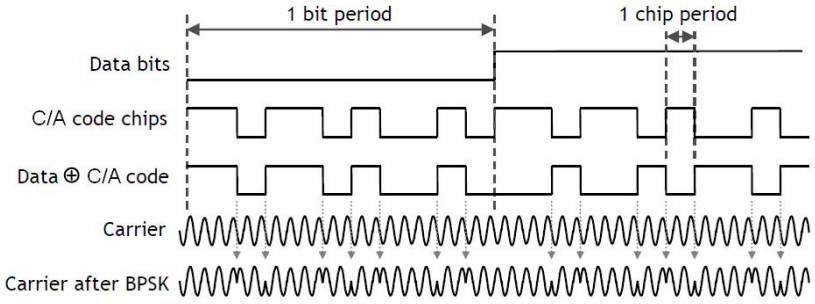

Figure B.6: *C/A spreading*

BIBLIOGRAPHY

[1] 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.

[2] Andrei Marinescu and Dragos Cătălin. *Assisted GPS System Design*. 2009.

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

[4] 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.

[5] GPS Joint Program Office. *IS-GPS-200 Revision D - Interface specification - Navstar GPS Space Segment/Navigation User Interfaces*. ARINC Engineering Services, 2004. http://www.navcen.uscg.gov/gps/geninfo/IS-GPS-200D.pdf.

[6] Wireless Innovation Forum. *Introduction to SDR*. 2003. http://www.wirelessinnovation.org.

[7] Roland Kaniuth, Bernd Eissfeller University FAF Munich Norbert Lemke, Günter Heinrichs, Jón Ó. Winkel IfEN GmbH Henning Ehm, Robert Weigel, Friedrich-Alexander University Erlangen-Nuremberg Andreas Schmid, André Neubauer, Infineon Technologies AG Peter Nagel, and Fraunhofer Institute for Integrated Circuits. *A Snap-Shot Positioning Approach for a High Integrated GPS/Galileo chipset*. ION GNSS 18th International Technical Meeting of the Satellite Division, 13-16 September 2005, Long Beach, CA, 2005.

[8] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins. *Global Positioning System: Theory and Practice*. Springer-Verlag Wien New York, 5th edition edition, 2001. ISBN-10: 3211835342.

[9] Elliott D. Kaplan and Christopher Hegarty. *Understanding GPS: Principles and Applications*. Artech House Publishers, 2th edition edition, 2005. ISBN-10 / ASIN: 1580538940.

[10] *SE4110L GPS Receiver IC*. SiGe semiconductor, 2006. http://ccar.colorado.edu/gnss/files/SE4110L_Datasheet_Rev3.pdf.

[11] Richard G. Lyons. *Understanding Digital Signal Processing*. Prentice Hall, 2nd edition edition, 2004. ISBN-10: 0-13-108989-7.

[12] Gerard Lacapelle, Daniele Borio, and Cillian O'Driscoll. *Coherent, Noncoherent and Differentially Coherent Acquisition of New Composite GNSS Signals*. IEEE, 2003.

[13] Sang Jeong Lee, Deuk Jae Cho, and Chansik Park. *An Assisted GPS Acquisition Method using L2 Civil Signal in Weak Signal Environment*. Journal of Global Positioning Systems Vol. 3, No. 1-2: 25-31, 2004.

[14] Brian Tolman, R. Benjamin Harris, Tom Gaussiran, David Munton, Jon Little, Richard Mach, Scot Nelsen, and Brent Renfro. The GPS Toolkit: Open Source GPS Software. In *Proceedings of the 16th International Technical Meeting of the Satellite Division of the Institute of Navigation*, Long Beach, California, September 2004.