Implementation of a Software-Defined GPS Receiver

Anthony J. Corbin

Advisor:

Dr. In Soo Ahn

Department of Electrical and Computer Engineering

Bradley University

Peoria, IL

Wednesday, May 14, 2008

Abstract

At present, most commercial GPS receivers utilize hardware designs to perform signal processing. While hardware designs may have many advantages, software designs have several added benefits including the potential for cost reduction, fast time-to-market, and the ability for after-market upgrades to support presently undefined navigation signals. Processing GPS signals in real-time is computationally complex and stretches the capabilities of even the newest processors. The goal of this project was to investigate this complexity and implement a software-GPS receiver in real-time. The software was written in C++ and utilizes a platform-independent and parallel architecture. In order to obtain a platform-independent and parallel architecture, an open-source library from Intel called Threading Building Blocks, which supports Windows and Linux, is used, which overcomes some of the problems with re-coding for a platform-specific application programmer's interface (API). While this software stretches the capabilities of present-day hardware, advances in processor design will lead to smaller, cheaper, and upgradeable GPS receivers.

Acknowledgements

I would like to thank the ECE department faculty, especially my advisor Dr. Ahn, for the endless help given to me over the past years at Bradley. Their mentoring and guidance has helped to give me a start towards a lifetime of achievement. I will always be in their debt.

Table of Contents

1. Introduction

Rationale

Advances in technology have provided society with numerous benefits not available to previous generations. However, a frequently overlooked fact is that the availability of technology is not entirely driven by capability, but, instead, by cost. While GPS has been available to the public since 1983, the technology did begin to see widespread use until this decade slowed down by cost. Advances in technology have finally lowered the cost to such a level that it is practical for a consumer to purchase a limited number of medium-cost devices for applications such as driving. However, the cost is still too high for the technology to practically be available in low-cost applications.

A possible solution to lower the cost of a GPS receiver lies in eliminating much of the hardware used by the receiver and replacing it with software. In executing this approach, the functionality of the hardware is shifted to the primary microcontroller eliminating the need for costly application-specific circuitry.

In addition to lowering cost, a software-model allows the receiver to be updateable. When the launch of the European Galileo System and the GPS Block III System is completed, legacy hardware devices will be incompatible with the new signal schemes. A software-defined system could merely be updated to work with the new system.

While the choice between using a hardware and software approach seems simple, a software system is still largely difficult to implement due to the extreme number of calculations required to process the signals.

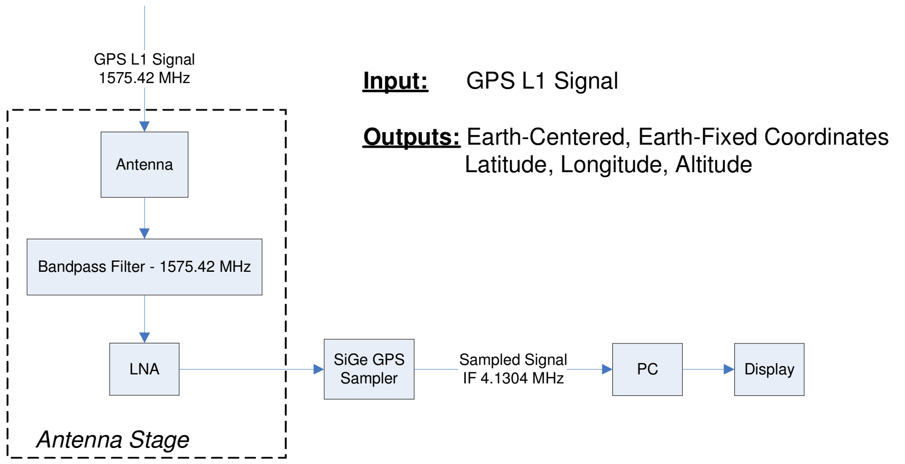
Project Goal

A team at the University of Colorado-Boulder successfully developed hardware to sample GPS signals and software in MATLAB to process it. However, the software implemented is excruciatingly slow taking tens of minutes to obtain an initial fix in certain instances. The limitation of the software prevents it from being practically applied in any way. The primary goal of this project was to redesign the software in C++ and implement it on a DSP. However, the difficulty of the application was largely underestimated leaving implementation on the available DSP impractical causing a shift in the project goal to a purely PC-based approach.

2. System Description

Figure 1, below, shows the high-level block diagram for the project. The GPS L1 signal is received through an active GPS antenna. After the antenna stage, the SiGe GPS Sampler chipset is used to sample and down-convert the signal. The PC then stores and processes the sampled signal.

Figure 1 – High-Level System Block Diagram



Antenna Stage

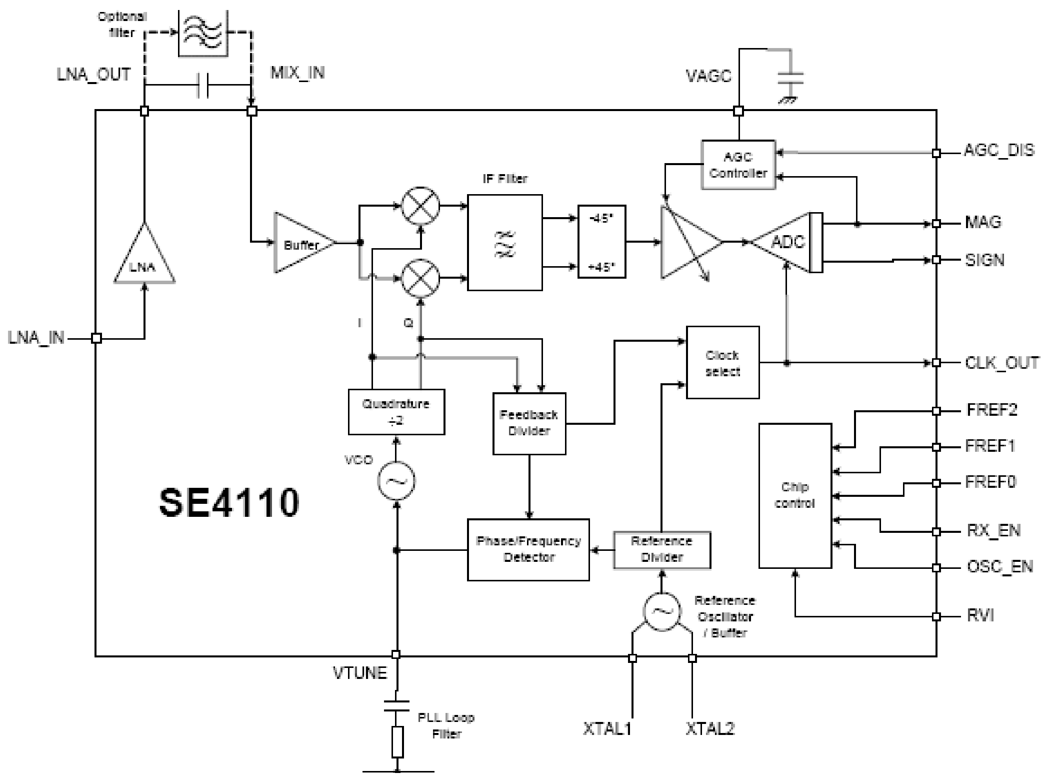
Active GPS antennas are available from a variety of manufacturers and include at a minimum a low-noise amplifier and a bandpass filter. An active antenna is necessary due to the low transmit power of GPS satellites and the degradation and attenuation of the signal as it travels through the atmosphere.

SiGe GPS Sampler

The SiGe GPS Sampler is a USB device developed by the University of Colorado and SiGe. The device contains a USB chipset and an SE4110L GPS chipset.

The SE4110L is a complicated chipset consisting of a several sub-stages. The subsystems are very similar to those of a superheterodyne receiver with the addition of a high-frequency A/D converter. The device outputs the magnitude and sign of the received signal at 4.1304 MHz. [2] Figure 2, below, shows the functional block diagram from the SE4110L datasheet. Functionally, the SE4110L provides the digitally sampled GPS L1 signal to the USB chipset.

Figure 2 - SE4110L Functional Block Diagram



PC

The software is designed in such a way that it could be run on any PC. However, the software contains optimizations for multi-core Intel PCs.

3. Subsystem Requirements

The subsystem requirements are shown in Figure 3, below. The individual requirements are discussed in the following sections.

Figure 3 - Table of Subsystem Requirements

Error	Specification(s)
Position Error	100 m
Sampling Rate	4.1304 MHz
Time to First Fix	Cold Start : 2 minutes Warm Start : 12 seconds
Display	Earth-centered, Earth-fixed Coordinates Latitude, Longitude, Altitude UTC Time Update

Position Error

The position error specification is set at 100 meters largely as a result of a limitation imposed by the sampling device. The sampling device obtains four samples per chip which results in a minimum error of 73.31 meters as shown in Figure 4, below. The specification is set at 100 meters in order to account for other possible error sources such as ionospheric or multipath effects.

Figure 4 - Position Error Calculation

$$\lambda_{Chip} = \frac{c}{f} = \frac{3\times10^{8}\,\frac{m}{s}}{1.023\times10^{6}\,\frac{1}{s}} = 293.26m$$

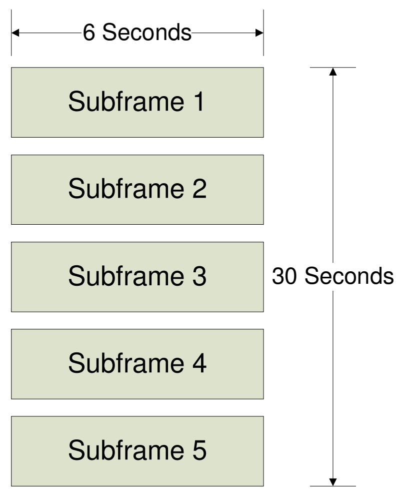
$$\text{Precision} = \frac{\lambda_{Chip}}{4} = \frac{293.26m}{4} = 73.31m$$

Time to First Fix

The time to first fix depends on whether or not initial information is available. In order to calculate a position fix, at least subframes 1-3 are required for processing. The subframes last 6 seconds each and are broadcast in the order 1-5 with four and five cycling through different sets of data. As it is initially unknown when subframe 1 will be sent, 36 seconds of data must be collected. Of this, thirty seconds accounts for collecting subframes 1-5 and the other six seconds accounts for starting the collection in the middle of a subframe.

Figure 5, below, illustrates the subframe transmission process.

Figure 5 - Subframe Transmission



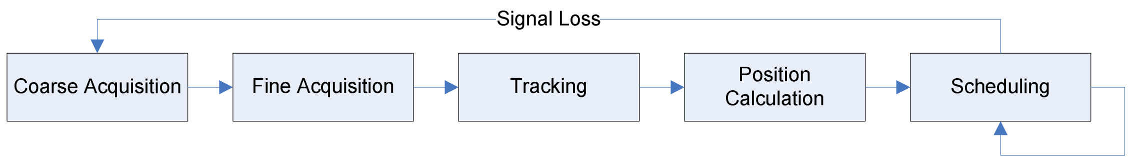
In addition to collecting 36 seconds of data, the data must be processed. The software cannot process the data in real-time due to hardware limitations so additional time must be added to account for this limitation. On a multi-core pc, approximately 20 seconds per satellite is required resulting in 80 seconds of processing time for 4 satellites. This sets the cold-start value at approximately 2 minutes.

When the subframe data is already available, the Time-of-Week (TOW) value must be updated which requires processing a minimal 1.2 seconds of data resulting in a warm start time of approximately 12 seconds.

4. Software Architecture

The high-level software architecture is shown in Figure 6, below. As the block diagram shows, the software first acquires the visible satellites. Next, enough information is extracted to determine the first position fix. Finally, using the initial estimate and ranging information, the position is estimated.

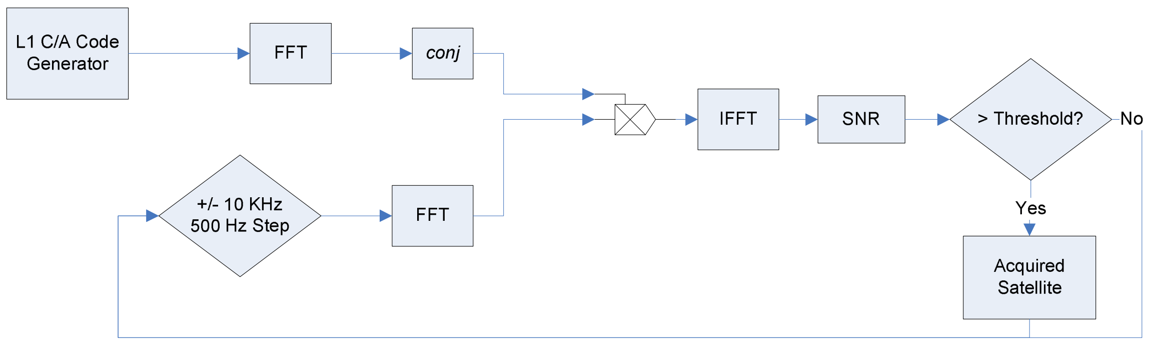
Figure 6 - High-level Software Architecture



Coarse Acquisition

The first step in the software process is determining which satellites are visible. Ideally, the intermediate frequency would be removed and the C/A code would be correlated for each satellite. However, several additional factors must be considered. First, since the satellites are moving very fast relative to the receiver position, the carrier frequency will be shifted due to the Doppler Effect. Due to this, the exact intermediate frequency is not known exactly making it necessary to check a range of frequencies. Second, it is very unlikely the C/A code was received beginning at the first chip requiring that every possible orientation be tested. Rather than checking each pattern one by one, correlation is performed in the frequency domain and the best correlation alignment is detected. Figure 7, below, shows the block diagram for the process.

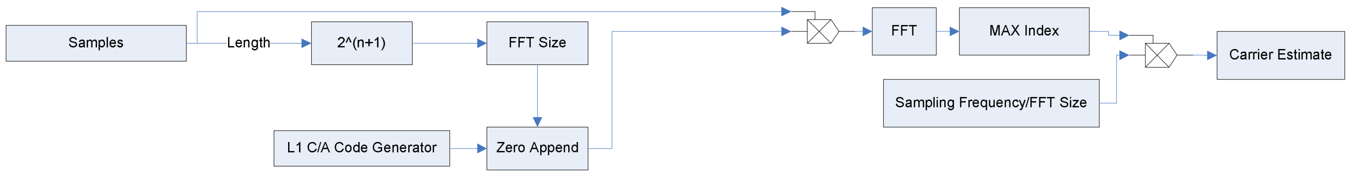
Figure 7 - Coarse Acquisition Block Diagram



Fine Acquisition

Once the satellites have been acquired, a rough estimate of the carrier frequency is known. However, this estimate is not accurate enough for the tracking loop to lock onto the carrier frequency. In order to find a better estimate, the current code phase estimate is used to remove the C/A code from a large set of data and an FFT is performed. If the C/A code was removed correctly, a peak will exist in the spectrum indicating an accurate value for the carrier frequency. Figure 8, below, shows a detailed block diagram describing the process.

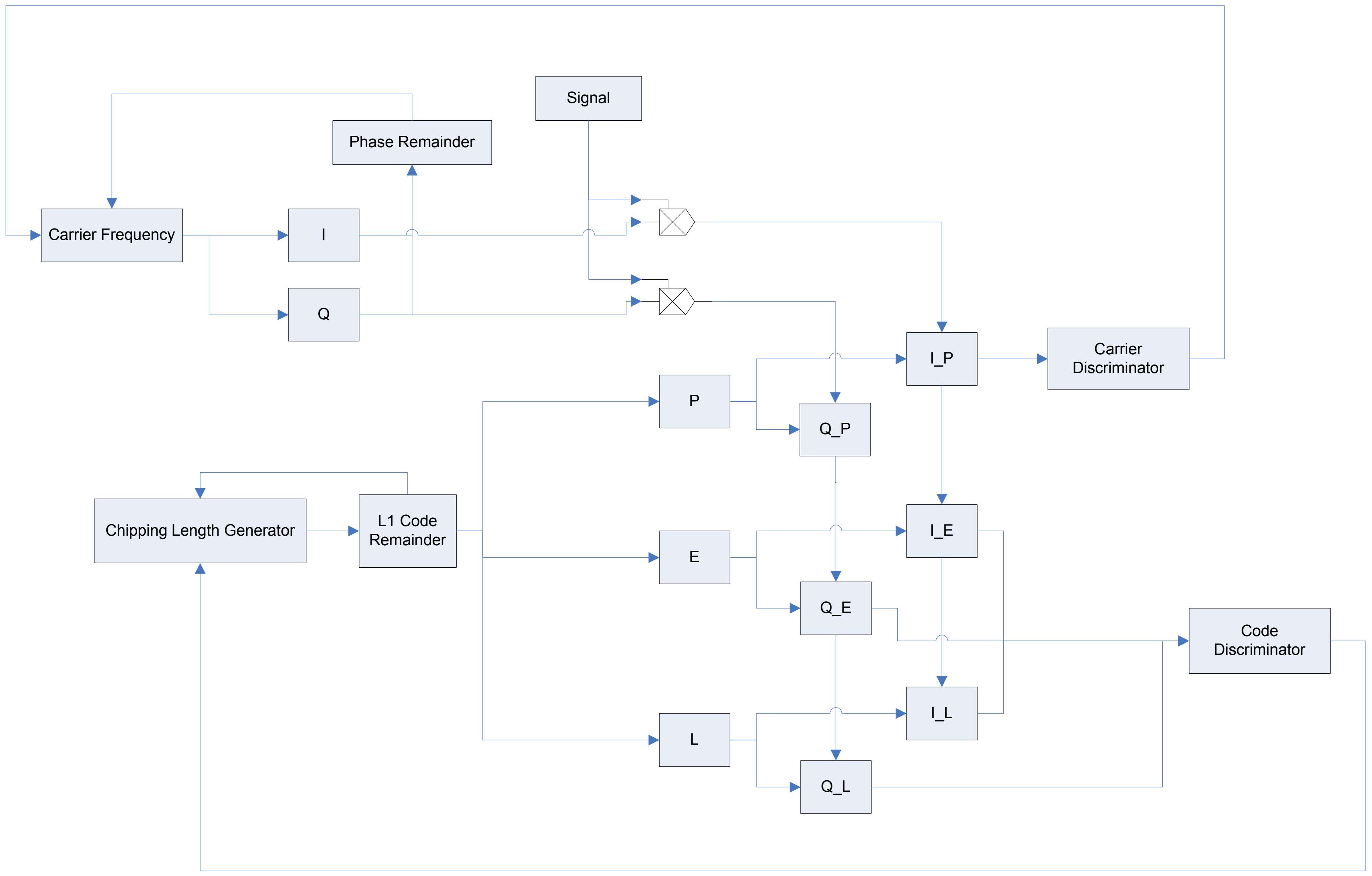
Figure 8 - Fine Acquisition Block Diagram



Tracking

After Fine Acquisition has completed, accurate estimates for the code phase and carrier frequency are available. These estimates can be used to demodulate the carrier, remove the C/A code, and extract the navigation data. However, the estimates for the code phase and carrier frequency are only valid over a short window of time. The Tracking loop updates the estimates for these two values by using signals shifted early and late. The computed correlation values for the early, prompt, and late signals are then used to generate a discriminator value which is used to update the two estimates. In addition to the in-phase signal, a quadrature signal must be generated to account for 180 degree phase shift in the incoming signal. Figure 9, on the next page, shows a detailed block diagram for the Tracking loop.

Figure 9 - Tracking Block Diagram [1]

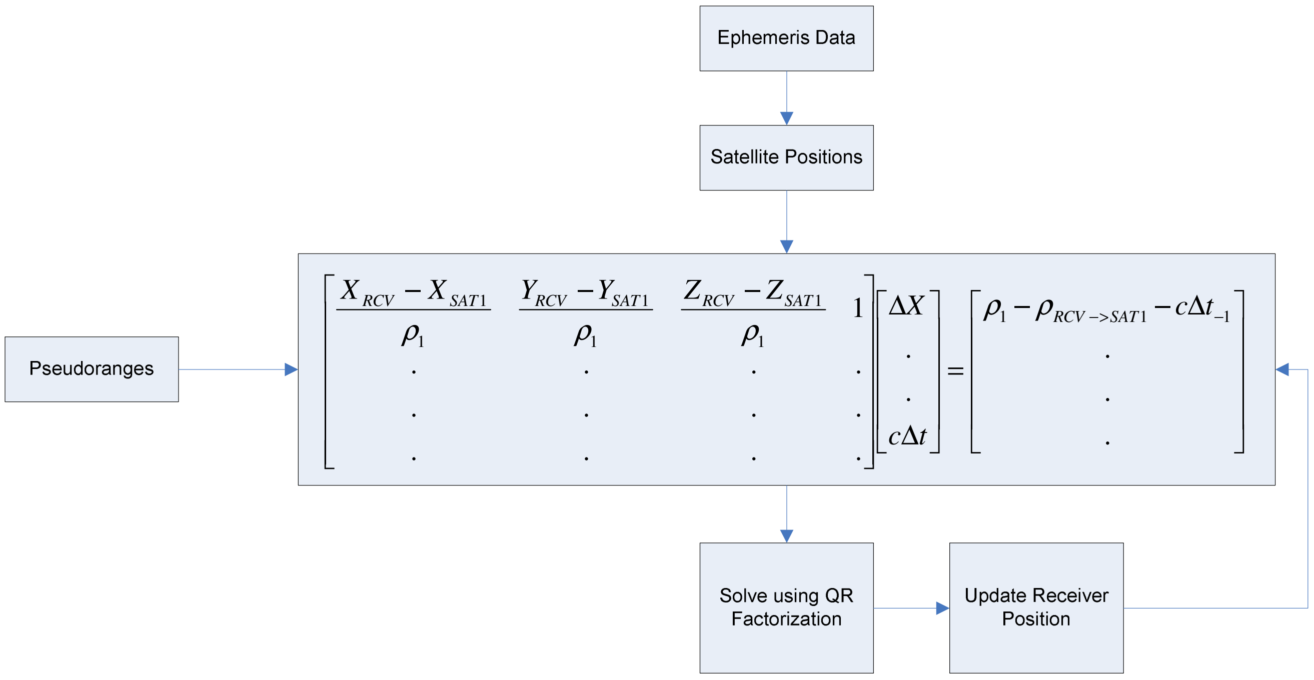


Position Calculation

Once Tracking has completed, the Ephemeris data can be extracted. The Ephemeris data provides information necessary to calculate the position and orbits of the satellites as well as to compute the time at which the data was transmitted. Furthermore, a value called Time of Week (TOW) is extracted and related to a sample index. The differences in the sample index values for the TOW is used to compute the pseudoranges.

Once the pseudoranges have been computed, the Least Mean Squares method is used to estimate the position of the receiver as well as any time bias present. Notably, at least four satellites must be available for the position fix to be computed. The matrix shown in Figure 10 is setup using the available information and decomposed using QR factorization. The use of QR factorization allows the software to solve the problem efficiently.

Figure 10 - Least Mean Squares Matrix



$$\begin{bmatrix} \frac{X_{RCV} - X_{SAT1}}{\rho_1} & \frac{Y_{RCV} - Y_{SAT1}}{\rho_1} & \frac{Z_{RCV} - Z_{SAT1}}{\rho_1} & 1 \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \end{bmatrix} \begin{bmatrix} \Delta X \\ . \\ . \\ c\Delta t \end{bmatrix} = \begin{bmatrix} \rho_1 - \rho_{RCV->SAT1} - c\Delta t_{-1} \\ . \\ . \\ . \end{bmatrix}$$

Scheduling

After the initial position fix, orbital data is still available for the processed satellites. This information is typically valid for at least 4 hours according to the Navstar specification [1]. During this time, only the Time-of-Week number must be synchronized for all satellites to determine the user's position.

Using this information, the position can be updated using a minimal amount of data and processing time. Figure 11, below, shows the timing involved.

Figure 11 - Scheduling Interval

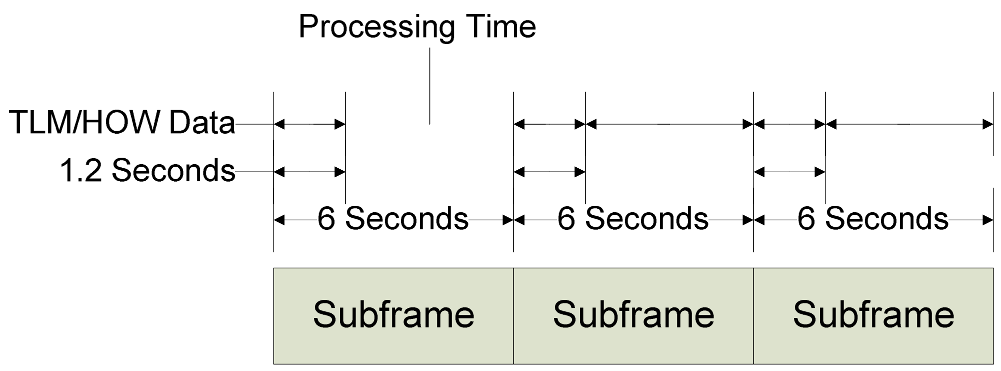
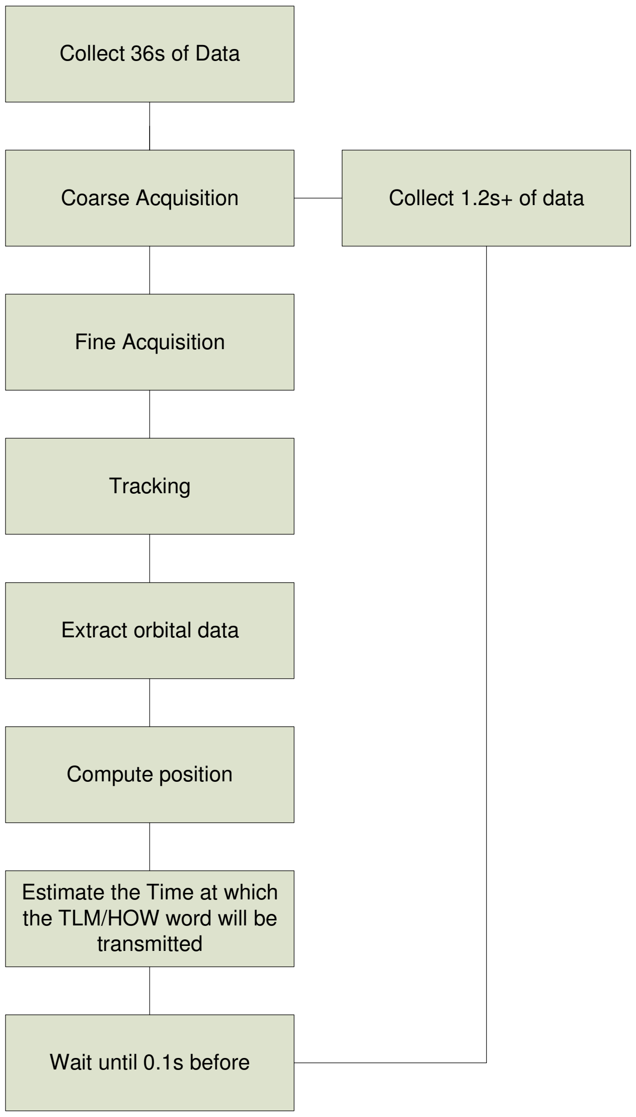


Figure 12, on the next page, indicates how the overall process will work in software. First, the initial data must be collected and processed. Then, using the previous time a subframe was received, the next subframe time is estimated. The scheduler waits until this time and collects a sample which will contain the telemetry (TLM) and handover (HOW) words for all the satellites that had been previously acquired. While the scheduler is waiting for the next subframe, the new data is processed and a position estimate is computed.

Theoretically, this process would yield a 1/6 Hz update rate or once every six seconds.

Figure 12 - Scheduler Flowchart



5. Software Display

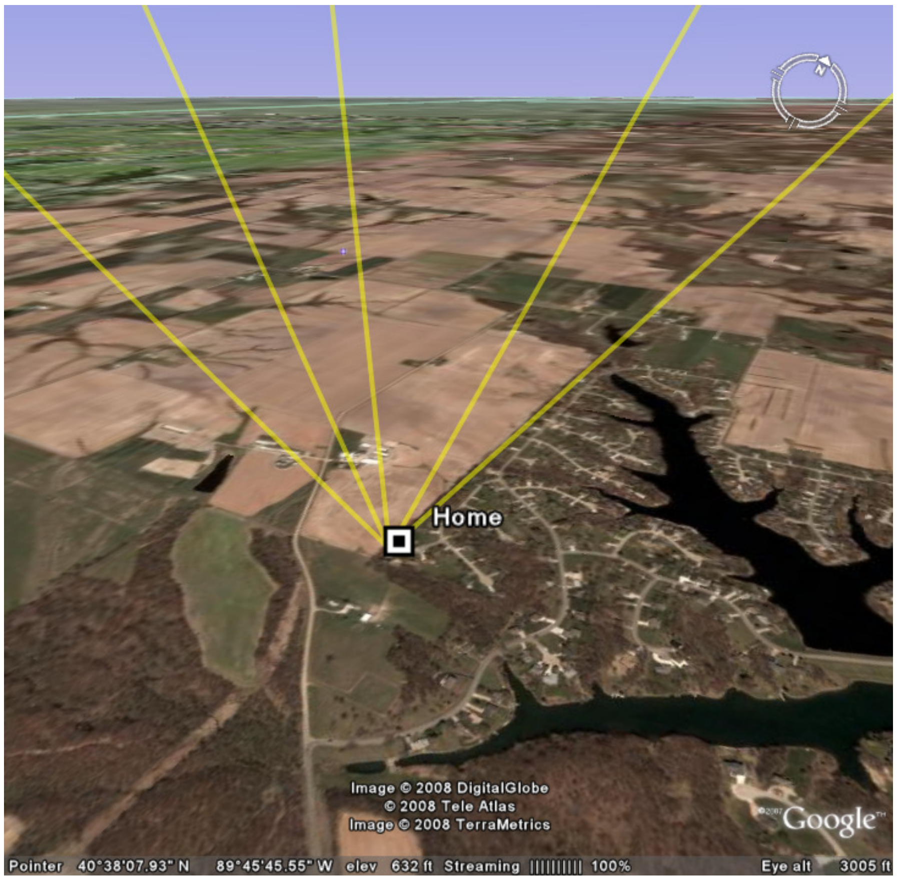
Google Earth

The program uses Google Earth to display the determined position. Google Earth is a free application available from Google which displays satellite imagery on a three-dimensional (3d) globe. The globe can be freely rotated and the view angle and elevation can be adjusted. The software interfaces directly with the application via COM (Component Object Model). A software wrapper for the interface provides functionality for drawing lines and points as well as positioning the view angle.

Drawing Lines

Figure 13, below, shows one of the benefits of being able to draw lines in three-dimensional space. In this picture, the lines begin at the individual satellite positions and converge on the receiver position.

Figure 13 - Line Drawing Functionality


Home
Image © 2008 DigitalGlobe
© 2008 Tele Atlas
Image © 2008 TerraMetrics
Google
Pointer 40°38'07.93" N 89°45'45.55" W elev 632 ft Streaming |||||||||| 100%
Eye alt 3005 ft

Drawing Points

Drawing points is essential to showing the position determined by the software. A text-only output indicating the latitude, longitude, and altitude means little to a user since slight variations in the numbers can represent miles. Furthermore, the continuous placement of points can help to indicate the motion of the receiver. Figure 14, below, demonstrates this functionality.

Figure 14 – Point Drawing Functionality

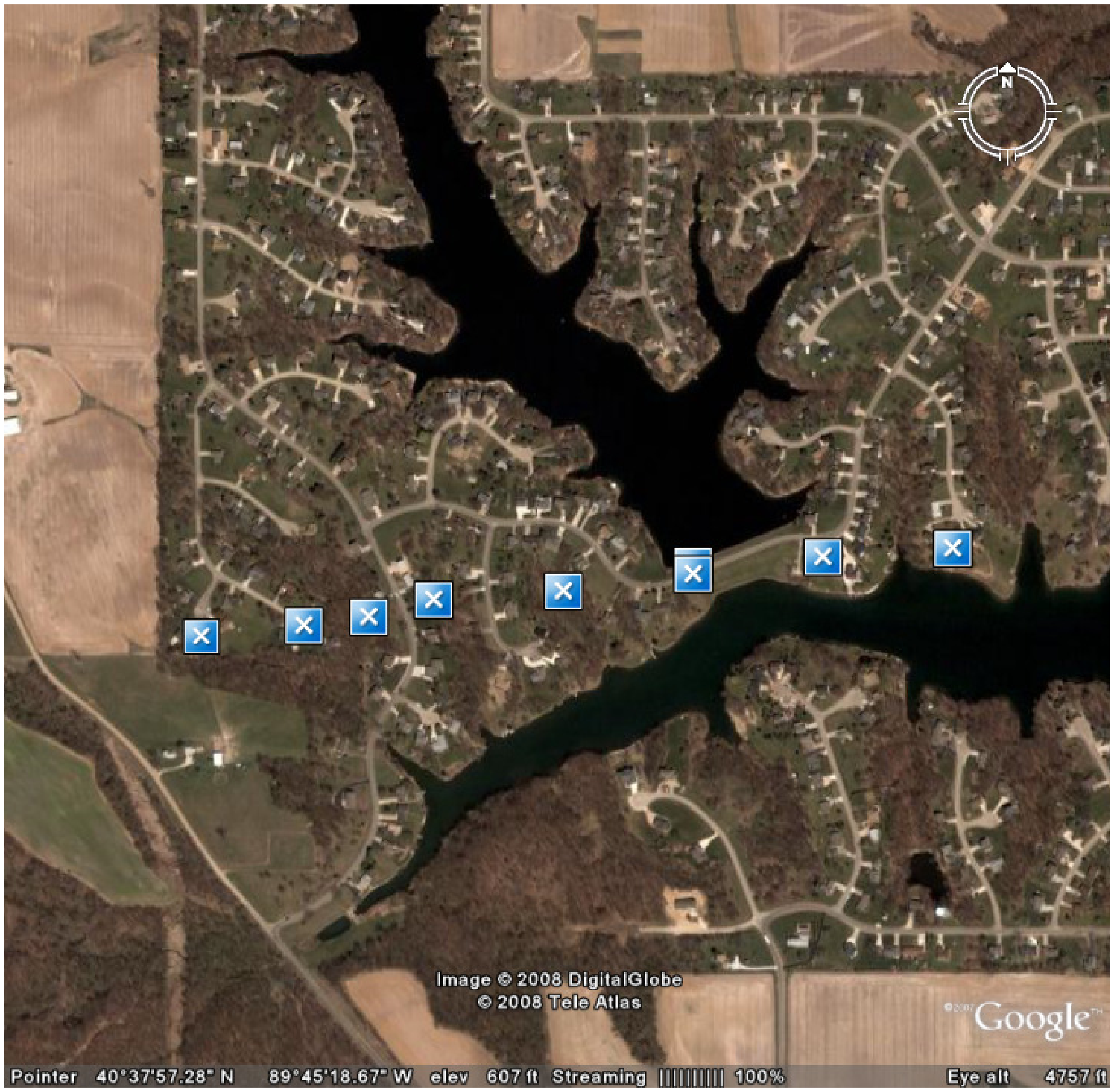


Image © 2008 DigitalGlobe
© 2008 Tele Atlas
Google
Pointer 40°37'57.28" N 89°45'18.67" W elev 607 ft Streaming |||||||||| 100%
Eye alt 4757 ft

3d Skymaps

Combining the line and point drawing functionality makes it possible to generate a 3d Skymap indicating the position of the satellites in the sky. This capability provides not only an aesthetic benefit but also functionality for verifying appropriate satellite geometry. Furthermore, this functionality could be used to verify that all visible satellites are acquired.

Figure 15 - 3d Skymap Functionality

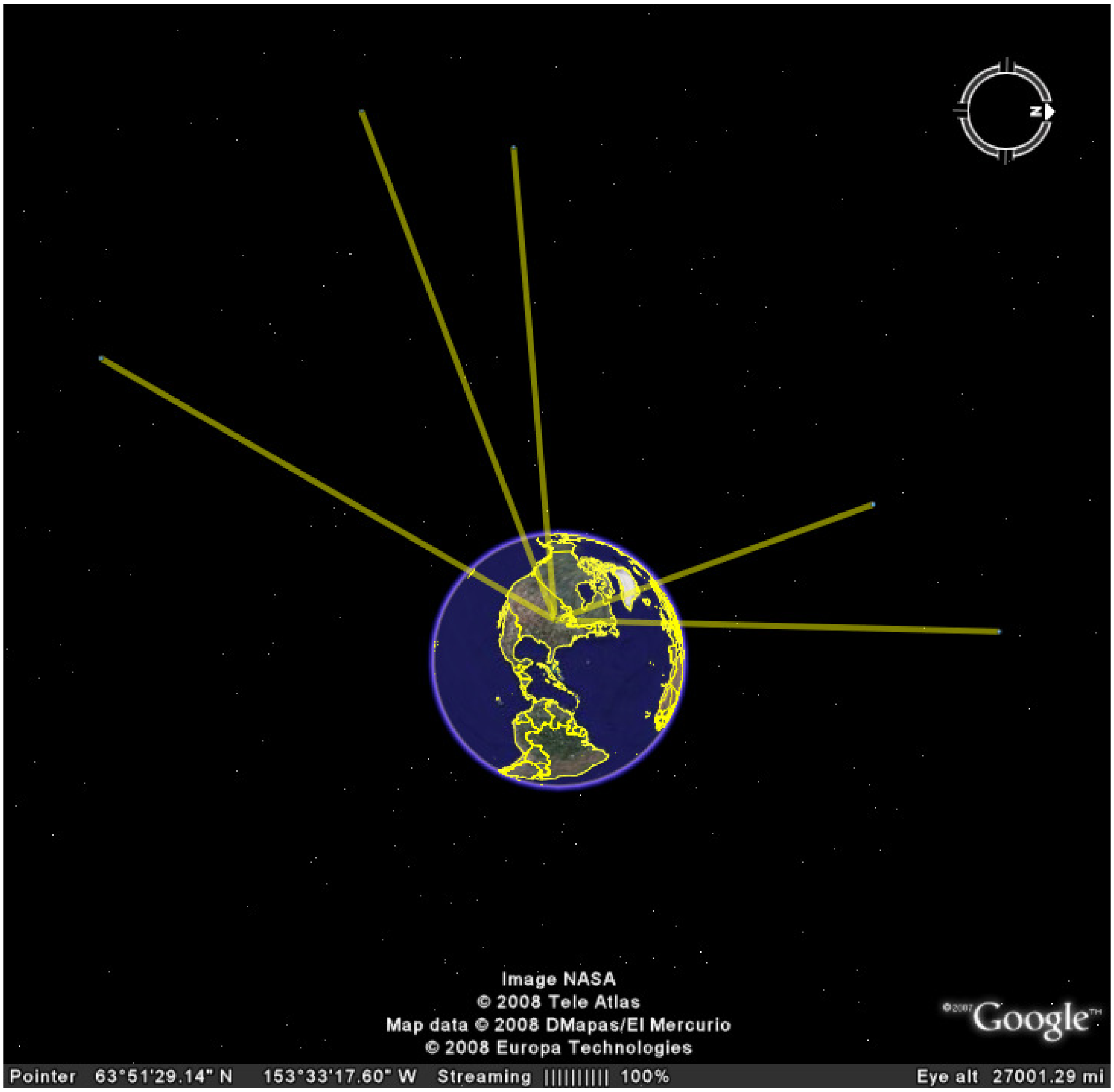


Image NASA
© 2008 Tele Atlas
Map data © 2008 DMapas/El Mercurio
© 2008 Europa Technologies
©2007 Google™
Pointer 63°51'29.14" N 153°33'17.60" W Streaming ||||||||| 100%
Eye alt 27001.29 mi

6. Results

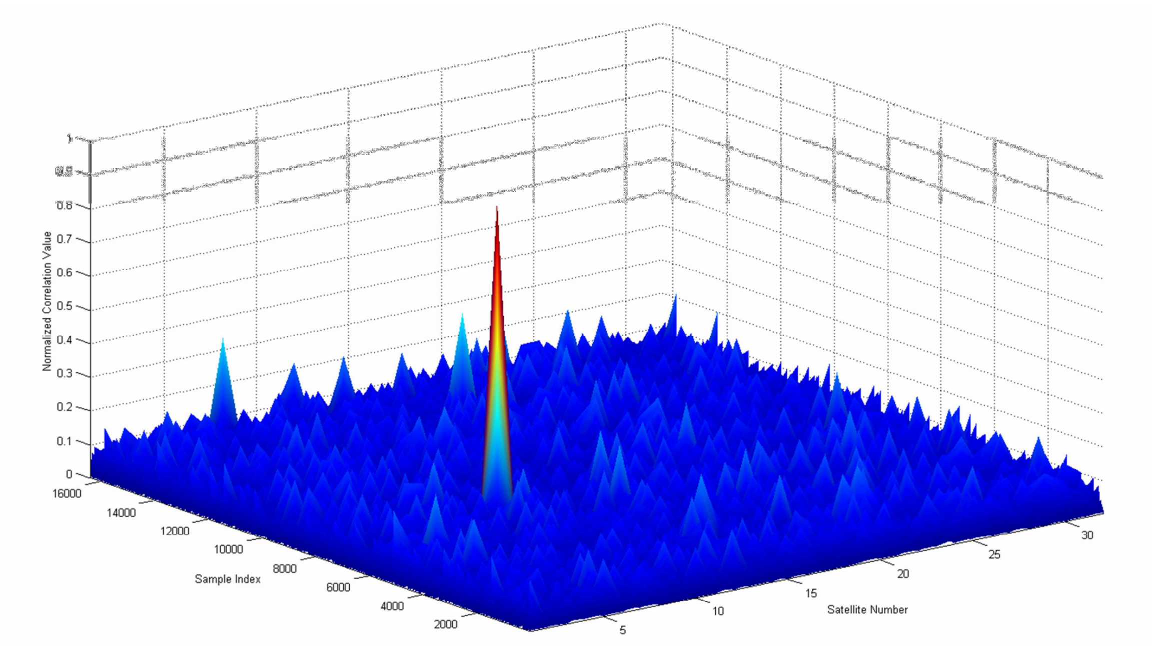
Coarse Acquisition Results

Both the satellite number and its carrier frequency must be determined. The results for a satellite search and carrier frequency search are shown.

Satellite Search

Figure 16, below, shows the results from a satellite search. The Sample Index refers to the sample at which the correlation began, and the Satellite Number refers to which satellite's C/A code was used. The large peak indicates acquisition. Notably, the width of the peak with respect to the sample index is very small. This indicates that even a slight shift in the C/A code results in a dramatic change in the correlation value. Most importantly, the values in the plane excluding the peak are very small limiting the possibility for false acquisition.

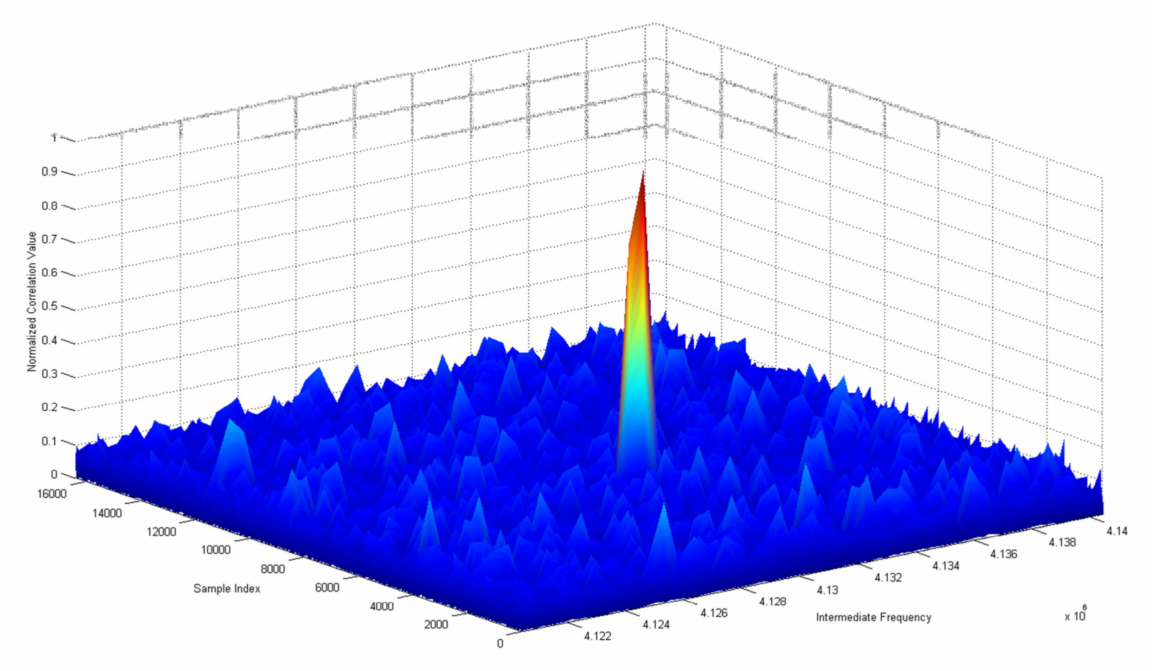
Figure 16 - Satellite Search



Carrier Frequency Search

Figure 17, below, shows the results of a correlation of a satellite's C/A code varying the carrier frequency. The large peak indicates the best carrier frequency estimate. Notably, the peak does have a bit of width with respect to the frequency range. This indicates that the frequency estimate may be off slightly without loss of signal. However, the width is not so large as to ignore the effect altogether.

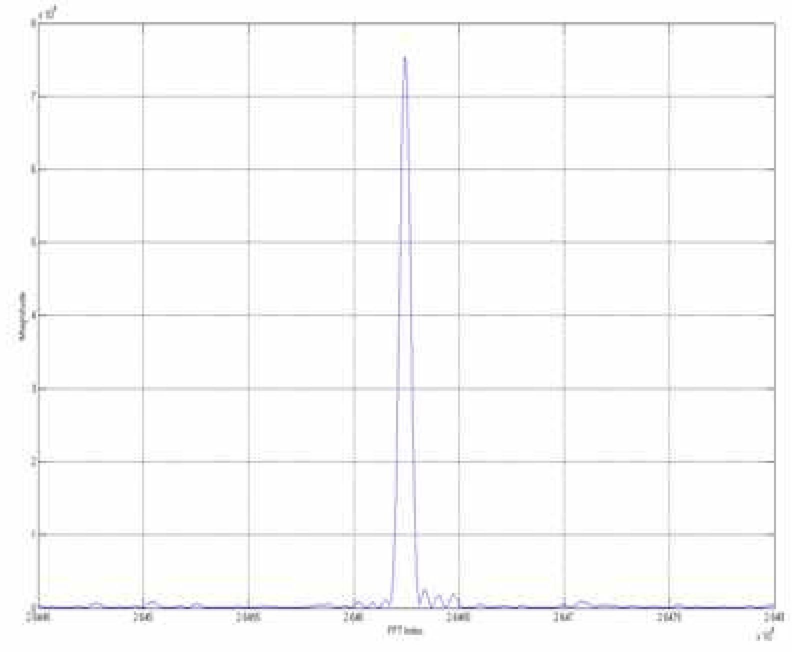
Figure 17 - Carrier Frequency Search



Fine Acquisition Results

Figure 18, shows the results from Fine Acquisition. The large peak at the center of the graph indicates the sample index corresponding to the carrier frequency. The sample index is then used with the nominal sampling frequency to compute a good carrier estimate.

Figure 18 - Fine Acquisition Result



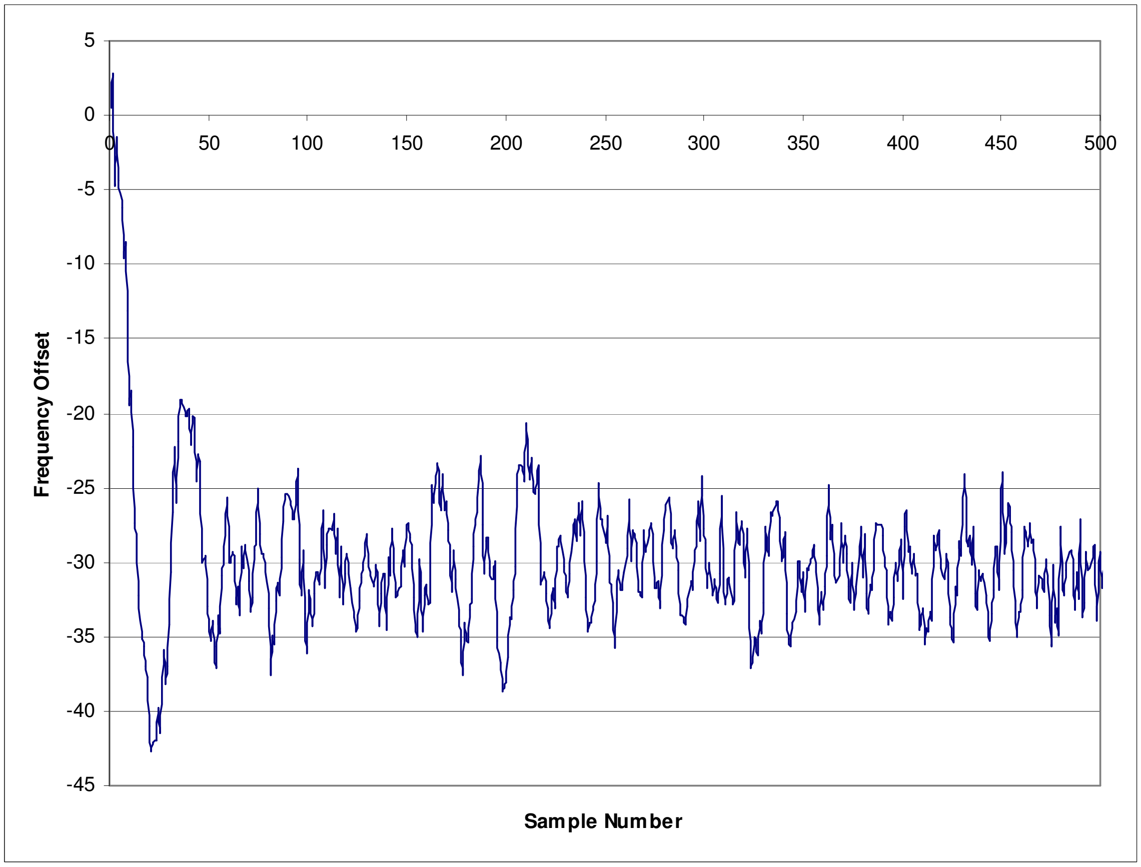
Tracking Results

The Tracking Loop simultaneously tracks both the carrier and code frequency of the incoming signal. Tracking results for both internal loops are shown below.

Code Tracking Loop

Figure 19 shows results from the code tracking loop. The offset is set to zero initially, but immediately moves in the negative direction as each millisecond of data is processed. After approximately 50 milliseconds (Samples) of data are processed, the loop offsets the code frequency around approximately -30 Hz.

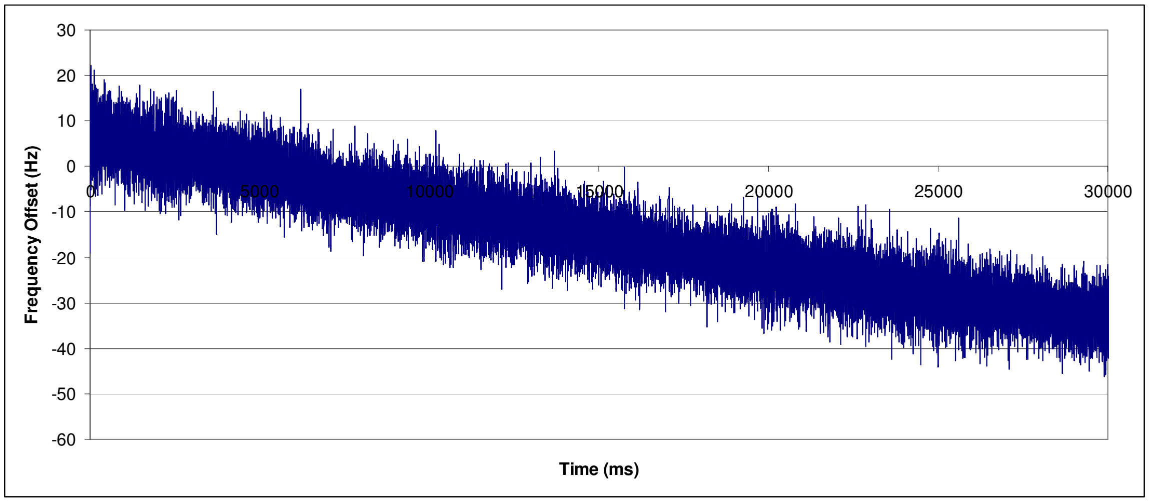
Figure 19 - Code Tracking Offset



Carrier Tracking Loop

The graph shown in Figure 20, below, shows the carrier frequency offset versus time. The offset initially remains close to zero indicating that the initial estimate from Fine Acquisition was, in fact, acceptable. The drift indicates that the actual carrier frequency is varying with respect to time due to the motion of the satellite.

Figure 20 - Carrier Frequency Offset



Position Estimates

The satellite image in Figure 21, on the next page, shows a position estimate determined by the software. The 51.81 m accuracy is acceptable, but not great. The position is calculated with the truncated TOW value likely accounting for some of the error in the pseudorange measurements.

Figure 22, also on the next page, shows a second position result. The accuracy in this case is worse than before. Like the estimate in Figure 21, on the preceding page, the truncated TOW is used. Additionally in this case, however, the estimate is worsened by poor satellite geometry. The antenna was held out the window on a stick on a lower level of the north side of the building indicated by the actual position marker. This, in effect, blocked or severely weakened many of the signals from the satellites in the Southern sky.

Figure 21 - Position Result

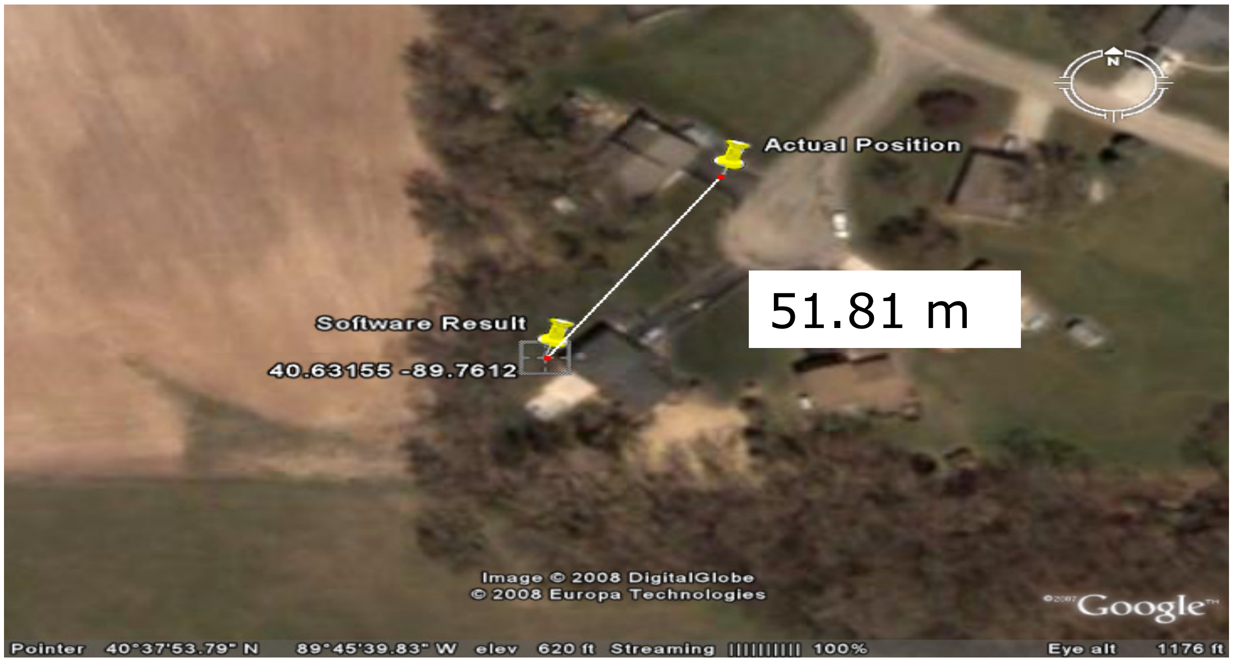
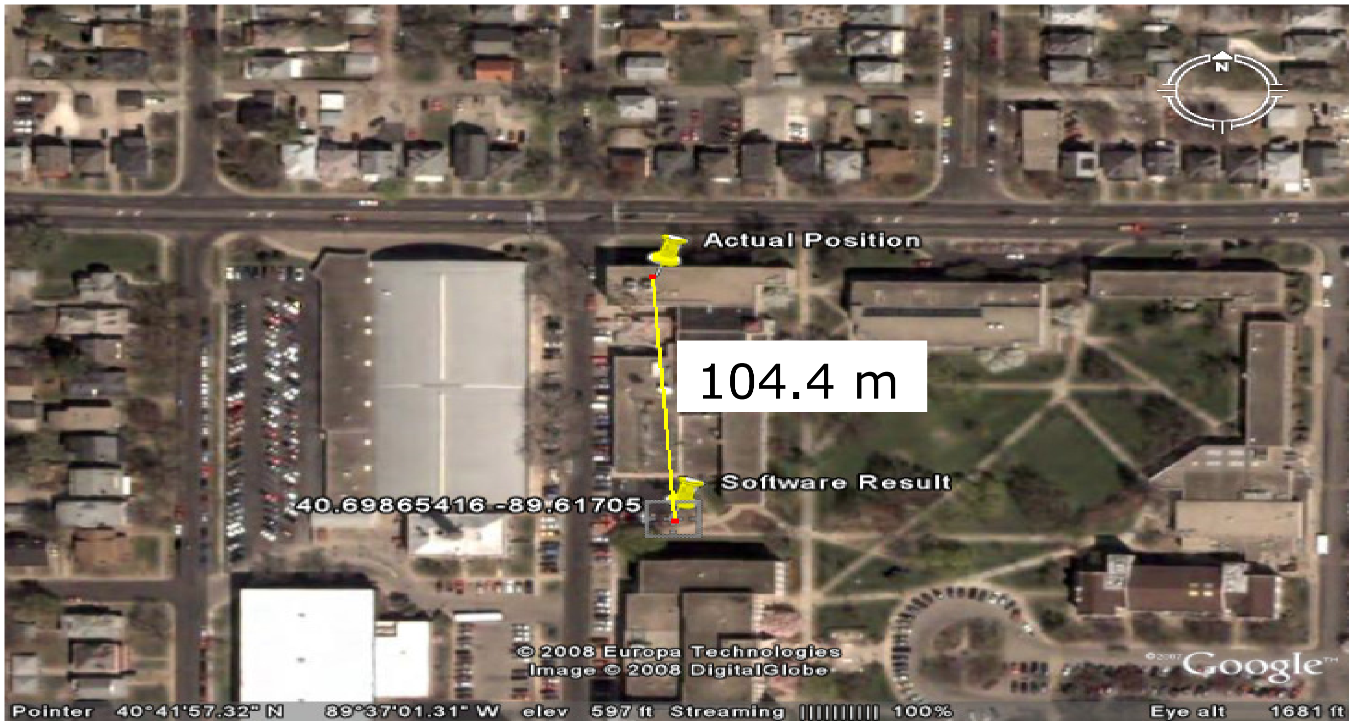

Actual Position
51.81 m
Software Result
40.63155 -89.7612
Image © 2008 DigitalGlobe
© 2008 Europa Technologies
Google
Pointer 40°37'53.79" N 89°45'39.83" W elev 620 ft Streaming |||||||||| 100%
Eye alt 1176 ft

Figure 22 - Position Result


Actual Position
104.4 m
Software Result
40.69865416 -89.61705
© 2008 Europa Technologies
Image © 2008 DigitalGlobe
Google
Pointer 40°41'57.32" N 89°37'01.31" W elev 597 ft Streaming |||||||||| 100%
Eye alt 1681 ft

7. Achievements

The project goals were largely met with the following achievements:

- Successful determination of position
  - An accurate position can be computed by the program.
- Real-time satellite availability determination
  - Program can determine which satellites are visible with a high update rate.
- Working C++ based receiver code
- Converted C driver code for the USB sampling device to C++
- Multi-threaded object-oriented design
  - Software can utilize multi-core processors on Windows based PCs.
- Google Earth C++ class wrapper
  - Program can mark the receiver position on satellite maps.

8. Conclusions

The project was a success meeting the revised project goals. The software is capable of determining the user's position; however, several problems faced during the project exposed the demanding baseband signal processing requirements of this approach. The raw hardware requirements make this approach impractical for commercial use.

Despite the challenging requirements, advances in future technology will eventually enable this approach. At present, a limited software approach utilizing a hardware-based carrier removal may be more feasible.

Software approaches certainly have merit given their potential for cost-reduction and upgradeability. The results of this project show that while it may not currently be possible to replace all hardware with software, a fully-functional software receiver design is not far away.

9. Recommendations for Future Work

Developing software is a continuous process with the ability for constant improvement. The first recommendation is to continue developing the software to improve its efficiency as well as its capabilities.

As a second recommendation, in order to make the process work in real-time, a hybrid approach where some of the processing is performed by hardware could be explored. Carrier removal is an expensive process in software that could be performed in hardware.

Finally, research into neural networks may provide a way to process the sampled signals in ways not otherwise considered. While the overall system requires high-accuracy not typically found with neural networks, the signal processing does not require extreme accuracy with parity checks providing verification of obtained data.

10. Appendix A – Software

11. Appendix B – Patent Search

Figure 23, below, lists relevant results from a patent search. While several patents exist for software-defined radio applications and hardware GPS applications, the number of patents specifically for software-defined GPS applications appears limited.

Figure 23 - Table of Relevant Patents

Patent Number	Description
20060074554	Software-defined GPS receivers and distributed positioning system
20067046193	Software GPS based integrated navigation
20067002515	GPS receiver using software correlation for acquisition and hardware correlation for tracking
2007213932	Computer Programmed with GPS Signal Processing Programs
2005162313	GPS Receiver (Software)

12. References

[1] Kai Borre, Dennis M. Akos, Nicolaj Bertelsen, Peter Rinder, and Soren Holdt Jensent, *Software-Defined GPS and Galileo Receiver : A Single-Frequency Approach.* Birkhauser: Boston, 2007, pp. 29, 83, 105.
[2] SiGe, SE4110L-EK1 Evaluation Board User Guide.
[3] SiGe, SE4110L Datasheet.
[4] U.S. DoD, *Navstar GPS Space Segment/Navigation User Interfaces.* IS-GPS-200 Rev. D.
[5] U.S. DoD, World Geodetic System 1984 : Its Definition and Relationships with Local Geodetic Systems
[6] Wikipedia, GLONASS. <http://en.wikipedia.org/wiki/GLONASS>
[7] Wikipedia, GALILEO. <http://en.wikipedia.org/wiki/Galileo_gps>
[8] EDACafe.com. *Atmel Introduces $5 GPS Baseband IC With 3 Meter Accuracy. <http://www10.edacafe.com/nbc/articles/view_article.php?articleid=177910&page_no=2>*