Precise Real-Time Relative Localization Using Single-Frequency GPS

By

Ronald William Hedgecock II

Dissertation
Submitted to the Faculty of the
Graduate School of Vanderbilt University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY
in

Electrical Engineering

May, 2014

Nashville, Tennessee

Approved:
Akos Ledeczi, Ph.D.
Aniruddha Gokhale, Ph.D.
Xenofon Koutsoukos, Ph.D.
Yuan Xue, Ph.D.
Miklos Maroti, Ph.D.

Copyright © 2014 by Ronald William Hedgecock II
All Rights Reserved

ACKNOWLEDGMENTS

The work in this dissertation could never have been made possible without the financial support of Vanderbilt University, a Google Research Award, and the National Science Foundation, and I am extremely grateful to NSF Project Manager Thyagarajan Nandagopal for taking a personal interest in this research. First and foremost, however, I would like to express my deepest and most heartfelt gratitude to my advisor and mentor, Akos Ledeczi, for his incredible support throughout the entire research and development process, as well as for his academic and professional guidance, without which I would never have been able to navigate this crazy maze known as grad school.

Of course, every project like this is a team effort, and I have to both acknowledge and thank two team members who worked closely with me on this endeavor: Janos Sallai and Peter Volgyesi. They participated in innumerable meetings that often involved large amounts of head-scratching and inevitably led to everyone leaving more confused about GPS than when we started, but it was during these brainstorming sessions that the largest amount of progress was made. I learned an immense amount from both of them and could never have seen this project to fruition without both their input and their insights. Finally, I would like to extend a huge level of gratitude to Miklos Maroti for always questioning EVERY assumption I made, as well as every last scrap of my math he could get his hands on. His desire to know, understand, and internalize every last detail of this work invariably helped me to understand it better myself, and his intuitive and unique way of approaching problems from a variety of different angles led to many of the most significant breakthroughs we encountered.

I would also like to thank each and every member who served on my Ph.D. committee for expending both the time and effort to understand this research and for taking an active interest in my journey through the graduate school process. I know that it takes a significant amount of time away from your own research interests to aid a student in his or her pursuit of a doctoral degree, but I truly appreciate the investment you made in seeing me through to the end of this milestone.

Finally, I have to extend my deepest gratitude to my parents, Ron and Emily Hedgecock, for their constant, unwavering support over the years. I don't think a day has gone by that they ever doubted I could achieve anything I set my mind to, and their unconditional belief in my abilities undoubtedly instilled in me the confidence I needed to embark on and achieve not only this degree, but any future undertaking I decide to throw my heart and mind into. Thank you both for all that you do, and I dedicate this dissertation to you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

A/D	Analog-to-Digital
AFM	Ambiguity Function Method
AFV	Ambiguity Function Value
ARP	Antenna Reference Point
BPSK	Binary Phase Shift Keying
C/A	Course-Acquisition
CDMA	Code Division Multiple Access
DGPS	Differential GPS
DSP	Digital Signal Processor
DSSS	Direct Sequence Spread Spectrum
ECEF	Earth-Centered, Earth-Fixed
ECI	Earth-Centered Inertial
GPS	Global Positioning System
IF	Intermediate Frequency
MCS	Master Control Station
MEO	Medium-Earth Orbiting
P	Precision
PCV	Phase Center Variation
PPS	Precise Positioning Service
PRN	Pseudo-Random Noise
RF	Radio Frequency
RHCP	Right-Hand Circularly Polarized
RTK	Real-Time Kinematic
SA	Selective Availability
SBAS	Satellite Based Augmentation System

SPS	Standard Positioning Service
SVN	Space Vehicle Number
TOA	Time of Arrival
TOF	Time of Flight
UAV	Unmanned Aerial Vehicle
UERE	User-Equivalent Range Error
WAAS	Wide Area Augmentation System
WSN	Wireless Sensor Network

CHAPTER I

INTRODUCTION

Many of today's wireless mobile applications rely heavily on highly accurate node location information. For outdoor applications, GPS is regarded as the clear choice for location determination, especially as its price and power requirements steadily decline; however, a significant tradeoff exists between cost and accuracy. Low-cost receivers, such as those found in smartphones and other handheld devices, can exhibit large errors on the order of tens of meters, especially in challenging RF environments such as wooded areas or urban centers. Higher quality devices exist that can provide much better accuracy, but they come at the expense of increased price, with commercial-grade surveying equipment costing thousands of dollars while requiring extensive setup and calibration before becoming usable.

GPS enjoys widespread applicability in the positioning domain due to its unique ability to simultaneously service an infinite number of users, providing absolute coordinates anywhere on Earth. The caveat is that many applications, including typical wireless sensor network (WSN) deployments, do not need precise *absolute* locations. In fact, a large percentage of WSN localization methods rely on estimating the range between nodes to derive their locations in a *relative* coordinate system. There also exist numerous applications beyond WSNs that require accurate relative locations, including autonomous driving, collision avoidance, land surveying, precision agriculture, and formation flying of unmanned aerial vehicles (UAVs), among others. Today's solutions to these types of problems involve post-hoc manipulation of absolute GPS coordinates (e.g. Differential GPS) or the use of additional sensors, such as LIDAR, accelerometers, or gyroscopes, which tend to be very costly.

This dissertation aims to bridge the gap between cost and accuracy by presenting a novel relative localization system based on the ubiquitous U.S.-based Global Positioning System and only low-cost, single-frequency GPS receivers. The goal is to enable commercial receivers to achieve an unprecedented centimeter-scale level of accuracy for applications that require node locations in a relative coordinate frame. Such an increase in GPS accuracy (in the relative sense) without a corresponding increase in cost should enable the production of novel applications that are either not economical today or have proven to be out of reach given the current state of the art.

Historical Background

The NAVSTAR Global Positioning System, or simply GPS for short, is a satellite-based radio navigation system designed and operated by the United States Department of Defense that provides accurate localization services anywhere on Earth for military and civilian applications. Its original design criteria consisted of [24]:

- Global coverage,
- Continuous, all-weather operation,
- The ability to operate in high-dynamic conditions, and
- High accuracy

The result is a system that allows individual receivers to calculate their approximate positions in three dimensions using signals from a closely monitored network of satellites, as well as the determination of a global atomic time to nanosecond accuracy without requiring Internet connectivity or external inputs.

Although the original idea behind a GPS-like radio navigation system was conceived as early as the 1940s [3, 29], the system in its current state was not commissioned until 1973 for use primarily in military applications [38]. It was realized, however, that the implications of such a system were immensely far-reaching and could include many non-military and commercial applications, especially in the realms of navigation and land surveying. As such, it evolved into a dual-signal system with one very precise restricted signal to be used in military-only applications and a second civilian signal to extend localization services to commercial applications. In an effort to protect the United States' interests by ensuring that its military was able to calculate receiver positions much more precisely than foreign militaries or civilian users, a mathematical degradation process known as Selective Availability (SA) was developed and applied to the civilian signal to decrease the maximum precision achievable by the system [53].

GPS reached a fully operational status on April 27, 1995 and found immediate commercial interest and success. A significant amount of investigative resources were devoted to overcoming the degraded accuracy levels imposed by SA, and after several years of extremely rapid, widespread growth, along with research into differential techniques which effectively removed any advantage arising from the military's access to more accurate ranging signals, the use of SA on civilian signals quickly became superfluous. In 2000, the degradation service was turned off, and non-military users have since been able to enjoy new levels of increased accuracy due to un-degraded ranging signals [6].

Following the economies of scale, GPS has experienced exponential growth and wide commercial acceptance in the recent decade and permeates daily life in ways completely unforeseen at the time of its inception. It has become the world's most widely used means of localization and navigation in outdoor deployments, both for scientific and commercial purposes [41, 48], and its accuracy and level of utilization continue to grow to this day.

Novel Research Contributions

In addition to the immense amount of research carried out during the years when SA was still active, the increase in popularity, applicability, and ubiquity of GPS has inevitably led to an even more significant body of research concerning new and better methods of localization using GPS signals, as well as ways to increase the accuracy of resulting coordinates. This dissertation explores and extends the current state of the art in terms of the algorithms and methodologies used in relative receiver localization for a variety of different applications and conditions. Specifically, it makes the following contributions to the relevant body of GPS localization research:

- Investigation into dual-receiver sources of error caused by the relativistic effects arising from receiver clock synchronization biases ("Section: *Dual-Receiver Error Sources*" in Chapter IV),
- Various methods of correction to minimize or even negate the effects of these newly-investigated error sources ("Section: *Time Bias Data Extrapolation*" in Chapter IV),
- Creation of a temporal observation model representing the relative motions of a set of receivers through both time and space ("Section: *Temporal Double-Differencing Model*" in Chapter IV),
- Development of an algorithm using this new observation model that enables centimeter-scale relative node tracking without requiring a reference node, reference satellite, stationary setup or calibration phase, or any *a priori* knowledge of the locations of one or more of the participating receivers ("Section: *Relative GPS Tracking Algorithm*" in Chapter IV),
- Creation of a Peak Tracking Filter, which extends the so-called Ambiguity Function Method (AFM) to include tracking results in a dynamic, multi-epoch baseline estimator between pairs of receivers ("Section: *Relative GPS Localization Algorithm*" in Chapter IV), and
- Synthesis of the aforementioned contributions into a complete and cohesive localization platform, enabling real-time, centimeter-scale estimation of relative GPS node locations without requiring predefined reference nodes or stationary calibrations ("Section: *Relative Localization Architecture and Practical Considerations*" in Chapter V).

Dissertation Organization

The remainder of this dissertation is organized as follows. The next chapter provides a high-level overview of the entire Global Positioning System, including its component segments, the structure of its radio signals used for ranging, and a discussion of the measurements that can be observed from a satellite at any given moment. An understanding of these signals and observables is necessary to comprehend the various sources of error inherent in GPS ranging, what we can do to correct or mitigate these errors, and how the observables can be combined into useful models for localization.

The subsequent chapter, Chapter III, introduces the various sources of error inherent in all GPS ranging methodologies, and explains the use of several observation models that deal with the mitigation or minimization of these errors.

Chapter IV begins discussion of the theoretical research in this dissertation, including *dual-receiver* relativity errors, methods of dealing with such errors, and a new temporal model that can be created once such errors have been removed. This is followed by creation of a multi-receiver relative tracking algorithm, the results of which are used to extend an existing baseline localization technique known as the Ambiguity Function Method. From these results, a complete positioning algorithm is constructed to include multiple dynamically-moving receivers without requiring calibration, a reference node, or a well-known reference location.

Chapter V discusses the practical implications of the theoretical research described in Chapter IV and synthesizes the individual research components into a working centimeter-scale relative localization prototype. This prototype is then tested in various conditions, environments, and network configurations, with the results being analyzed and discussed in Chapter VI. We conclude the dissertation in Chapter VII with a recap of the work and contributions presented here, as well as suggestions for improvement in future work.

CHAPTER II

GLOBAL POSITIONING SYSTEM OVERVIEW

The Global Positioning System can be separated into three distinct components: a space segment, a user segment, and a control segment. These segments are respectively made up of the GPS satellite constellation, the user equipment which receives satellite ranging signals, and a ground monitoring network. This section discusses each of these segments in turn and examines how they relate and work together to comprise the ubiquitous localization service that is GPS.

Space Segment

GPS Satellite Constellation

The original GPS commission called for 24 satellites to make up the bulk of what is known as the "space segment." Nominally, these satellites would move in 6 orbital planes with 4 evenly-spaced satellites per plane, ensuring that a minimum of five satellites would be visible from any line-of-sight point on Earth at any given time [34]. This configuration was realized in 1995 when the GPS system became fully operational; however, additional satellites have been launched over the years, resulting in an asymmetrical configuration that expands upon the original principles. The current GPS satellite constellation consists of a set of 31 medium-Earth orbiting (MEO) satellites, which has substantially increased the number of satellites visible at any given time from the original specification [37].

The orbital period of each satellite is 11 hours, 58 minutes, with each plane evenly-spaced around the equator at 60° of separation and a nominal inclination of 55° relative to the equatorial plane. Each satellite maintains a nearly circular orbit approximately 26,600 kilometers from the center of the Earth. Assuming an average Earth radius of ~6,378 km, the satellites tend to orbit about 20,222 km above any given point on the surface of the Earth [35]. With the original 24 satellites, the relative angle between any two satellites in adjacent orbits was ~40°, but this number has since decreased with the seven additional satellites that were added to the current constellation [24].

GPS satellites are referred to formally by their official Space Vehicle Numbers (SVN) as assigned by the Department of Defense, but can also be referenced according to their orbital plane (a capital letter in the range A-F) and numerical slot in that plane. Most commonly, satellites are referenced by the unique individual ranging codes that they emit. There are currently 32 pre-defined digital sequences known as Pseudo-Random Noise (PRN) codes, with each satellite emitting exactly one of these PRN sequences at any given time. It is possible for a satellite to switch between different PRN sequences, but no two satellites will ever emit the same sequence simultaneously; thus, the PRN ranging signal can be used to uniquely identify any satellite at a given time [36]. For the rest of this dissertation, we will discuss satellites in terms of their assigned PRN codes ranging from 1 to 32 (e.g. if NAVSTAR SVN 60 is emitting PRN code #12, the satellite will simply be referred to as 'PRN 12').

GPS Signal Composition

GPS is a passive system in which each satellite transmits a unique one-way signal to a potentially infinite number of receivers whose job it is to decode and utilize that signal for navigation purposes. As such, receivers rely on a one-way form of Time-of-Arrival (TOA) ranging to determine their positions. Due to the one-way nature of these signals along with the fact that satellites cannot communicate with one another, it is essential that all satellite clocks be precisely aligned to some reference time. This is achieved via synchronization of highly accurate atomic clocks on board each satellite to a so-called absolute "GPS time" which is directly related to the Coordinated Universal Time (UTC) to which our Earth clocks are synchronized [35]. These atomic clocks generate a long-band (IEEE designation "L-band") sine wave with a frequency of 10.23 MHz, which is then separated into two additional signals via multiplication by 154 and 120. The resulting frequencies, denoted **L1** (1,575.42 MHz) and **L2** (1,227.6 MHz), are the carrier frequencies used by all GPS satellites on which to encode their ranging signals [12].

As mentioned in the introduction, GPS currently implements two positioning services. The first is called the Standard Positioning Service (SPS), and it is available for unrestricted civilian use worldwide. The signal associated with this service is called the Course-Acquisition (C/A) code, and it is encoded onto the L1 frequency. A separate Precision (P) code, associated with the restricted, military-only Precise Positioning Service (PPS), is encrypted and encoded onto both the L1 and L2 carrier frequencies [24]. The focus of this research lies strictly with SPS, so the signals of interest are the various C/A codes used to modulate the L1 frequency.

We mentioned that each satellite emits a unique, pre-defined pseudo-random noise (PRN) sequence. The C/A code for each satellite is a 1 kHz signal made up of continuously repeating 1 millisecond long PRN sequences over time. Each digital PRN sequence consists of 1,023 binary bits which modulate the L1 carrier wave at a rate of 1.023 MHz to form a ranging signal [12]. Note that the PRN code is considered a 1 kHz signal because each code *symbol* is 1 ms long, but the actual PRN bits modulate the carrier wave at 1.023 MHz. This becomes important when discussing frequency-dependent error sources in Chapter III. In essence, this distinction between carrier and code allows GPS to utilize Direct Sequence Spread Spectrum (DSSS) modulation to provide Code Division Multiple Access (CDMA) multiplexing, whereby each satellite signal has access to its own high-bandwidth (2.046 MHz) data channel by virtue of the unique PRN code used to encode its data [35].

In DSSS modulation, a carrier wave is phase-modulated by a string of pseudo-random noise symbols called "chips" which are made up of a deterministic (but seemingly random, hence **pseudo**-random) set of binary noise bits. In GPS, the 1,023 bits of the PRN code sequence are used as chips to modulate the L1 carrier signal using Binary Phase Shift Keying (BPSK). This results in a signal that looks like and has characteristics similar to white noise [24]. The benefits of such a modulation technique include resistance to jamming, reduced signal-to-background noise, and the ability to share a single data channel among multiple receivers, hampered only by the amount of cross-correlation between the various PRN codes used to "spread" the signal [43]. As such, the 32 PRN sequences currently in use were carefully chosen based on their low cross-correlation profiles with respect to one another.

Finally, each satellite continuously sends navigation messages encoded at 50 data bits per second on top of the C/A code signal [12]. These navigation messages contain satellite and clock parameters, enabling a receiver to compute the position of the satellites at the time of signal transmission, determine correction factors for satellite clock biases and atmospheric disturbances, and read status and health updates regarding the transmitting satellite. Together, these signals form a composite GPS L1 signal as shown in Figure 1 (adapted from [11]).

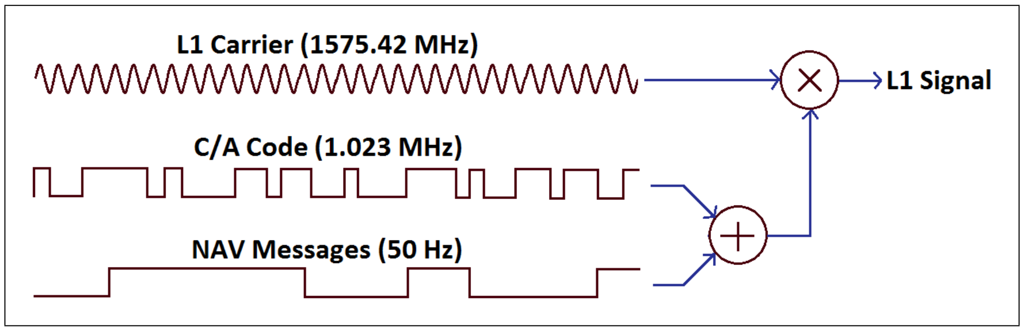


Figure 1: GPS satellite signal composition

User Segment

Receiver Hardware

In order to utilize the spread-spectrum L1 signal as described in the previous section, GPS receivers use a variety of specialized hardware to detect, demodulate, and separate the composite signal into its individual components. While the hardware specifications of a receiver depend heavily on its intended application, the necessary set of components needed to process a GPS signal remains the same: an antenna, filtering and downconversion components, an internal oscillator, digital signal processors, and a navigation processor.

Antenna

The antennas used in the majority of GPS receivers are right-hand circularly polarized (RHCP) with nearly complete hemispheric coverage. Most antennas have unity gain at elevation angles around 15°, increasing to about 2.5 dBic (decibels-isotropic circular) at the zenith [24]. The gain below an elevation of 15° is typically negative; thus, stronger and more reliable signals can be received from satellites at higher angles with respect to the receiver.

Depending on the use case, a GPS antenna should be able to pass the entire bandwidth of an L1 signal (2.046 MHz). Many low-cost, inaccurate receivers will only pass a portion of the bandwidth (for instance, many hiking receivers pass only 1.7 MHz [24]). This results in a loss of information and less accurate ranging signals. Additionally, antennas have electrical phase centers that do not usually coincide with the actual geometric center of the antenna. In fact, the phase center may be dependent not only on the receiver itself, but also on the elevation and azimuth of the incoming signal, with some antennas more sensitive to these effects than others. It is, therefore, imperative that the antenna design characteristics match the intended application's specifications.

Filtering and Downconversion

The received signals are immediately routed from the antenna through a passive band-pass filter to remove any out-of-band RF interference. This is followed by pre-amplification before a downconversion process in which the RF signals are converted to an intermediate frequency (IF) to be sampled by an analog-to-digital (A/D) converter. Generally, the A/D converter will sample at a rate anywhere between 2 and 20 times the PRN code chipping rate (1.023 MHz for L1 C/A) [24]. The minimum sampling rate that satisfies the Nyquist sampling theorem would be 2.046 MHz, but oversampling reduces the sensitivity of the receiver to quantization noise, thereby reducing the number of bits required by the A/D converter.

Internal Oscillator

The entire GPS receiver is controlled by a low-cost internal oscillator that can be implemented using any number of technologies. This oscillator is used to drive the downconversion and A/D conversion processes, as well as to synthesize local versions of the various PRN codes being emitted by the satellites. Since these local oscillators are much less accurate than the atomic clocks found on board a GPS satellite, their biases and drifts over time constitute large sources of error in the localization process, requiring them to be handled mathematically during position estimation.

Digital Signal Processors

A digital output stream from the A/D converter is fed simultaneously into a bank of Digital Signal Processors (DSPs) which are responsible for tracking the various PRN code streams being transmitted by the satellites. Each DSP comprises its own separate "channel," with each channel being used to track one and only one PRN sequence at a time. Since there are a limited number of satellites visible to a receiver at any given time, the number of parallel channels does not need to equal the total number of possible PRN sequences. In most receivers, there are typically 8 to 12 channels, which adequately allows the receiver to track all satellites that may be visible at any one time instant [24]. The output from these processors includes the raw navigation messages that were encoded on the L1 signal for that satellite, as well as the estimated time-of-flight measurements and any Doppler shifts noted in the tracked signal.

Navigation Processor

Finally, the output from the DSP channels are used by a Navigation Processor component to keep track of various signal acquisition properties and measurement observables, and in some receivers, to provide an instantaneous estimate of the current position, velocity, and time to the user.

Signal Decomposition

The L1 GPS signal captured by a receiver's antenna will be the same composite signal as shown in Figure 1. The individual signal components can be recovered by essentially reversing the modulation operations used to create the signal in the first place. In order to do this, however, the receiver must be able to uniquely identify the transmitting satellite and its current position in transmission of the PRN code sequence (known as the *codephase*). This operation is carried out by a process known as *digital autocorrelation* [46].

In autocorrelation, a signal is compared with a duplicate of itself shifted by some arbitrary amount of time. An integrator keeps track of the "sameness" of the digital signals by adding a +1 whenever the current chips or bits match and a -1 whenever they are different. If the signals being compared are of a finite length (like PRN codes), the resulting accumulated value can be divided by the total code length so that a perfectly matched code will result in an autocorrelation value of +1. Random DSSS binary code signals (including GPS PRN codes) are unique in that they produce an autocorrelation value of +1 only when they successfully correlate with themselves to within one chip offset. Correlation of the signal with any other random binary code or with itself shifted any other number of chips will result in an autocorrelation value close to 0 [5]. It is for this reason that multiple PRN codes can be transmitted on the same frequency without corrupting each others' data.

Since there are currently 32 possible GPS PRN codes, each only 1 ms long, a receiver is required to search through all available PRN sequences shifted arbitrarily through time until it finds a sequence that exactly matches one of the PRN codes present in the L1 signal [24]. (Note that in order to "match" code sequences, a receiver must know the PRN sequences *a priori*.) For each available DSP channel in the receiver, a different PRN code is generated locally and fed into an autocorrelation component along with the downconverted L1 GPS signal. This autocorrelation component consists of a feedback loop which attempts to maximize the amplitude of the autocorrelation function by shifting the currently-generated PRN sequence chip by chip through time over a period of 1 ms (the length of one full PRN cycle). Upon finding a PRN sequence and shift that results in an autocorrelation value near 1, a satellite "lock" is said to have been acquired. If no lock is acquired after attempting to autocorrelate over the entire PRN sequence, it is evident that the receiver is not receiving a signal from the satellite associated with that PRN, and it may move on to the next untested PRN sequence.

After a satellite lock has been acquired, the DSP will add its locally generated version of the shifted PRN sequence to the received signal, which effectively cancels out the PRN code leaving only the intermediate carrier frequency and the navigation message corresponding to the locked PRN [46]. By stripping the carrier frequency from the resulting signal, the receiver is left with the raw 50 Hz navigation messages which are then forwarded to the navigation processor, along with the amount of shift required to lock onto the PRN code and any Doppler shift required to maintain that lock after it was initially acquired. (These shifts are used in ranging and will be discussed in more detail in "Section: *Measurement Observables*" in Chapter II.)

Control Segment

In order to keep the entire GPS system running smoothly, the network of satellites is closely monitored by what is known as the "Control Segment." This segment consists of sixteen monitoring stations, four ground antennas, and a Master Control Station (MCS) located at Schriever Air Force Base in Colorado Springs, Colorado [32]. Its purpose is to monitor all satellite navigation signals for accuracy, coherence, and consistency, update the satellites' position data (known as the *ephemerides*), resolve satellite anomalies, correct satellite clock and orbital errors, and keep up with the general housekeeping tasks required for continuous, smooth satellite operation [24].

Satellite Augmentation Systems

An additional "helper" service called the Wide Area Augmentation System (WAAS) was developed to enhance the accuracy and availability of the standard GPS system and satellite constellation. The WAAS belongs to a group of Satellite Based Augmentation Systems (SBAS) whose purpose is to augment the features and applicability of various navigation systems from different countries. By analyzing received satellite ranging signals based on what the expected signals should be, WAAS monitoring stations are able to determine differential corrections with respect to atmospheric signal delays, satellite orbital and clock errors, and a number of other environmental conditions that can affect GPS signals uniformly over a wide region for all users within a given satellite's visibility range [1].

Unlike GPS satellites, WAAS satellites are geostationary, with the current constellation consisting of three equatorial satellites at varying longitudes, providing full-time coverage of the entire United States. While the primary purpose of WAAS is to improve the accuracy, availability, and integrity of GPS, WAAS satellites also transmit ranging signals of their own using PRN codes just like GPS, essentially serving as additional GPS satellites for receivers capable of decoding the WAAS signals [1].

Measurement Observables

Contrary to popular belief, the *natural* measurements of a GPS receiver are not the same as the so-called "raw measurements" that the receiver outputs. In fact, the natural measurements have already been alluded to in "Section: *Signal Decomposition*" in Chapter II when we discussed the replica carrier wave and PRN codes generated locally by the receiver. The only measurements a GPS receiver actually makes are the amount of time shift required to align the received and locally generated PRN codes (the codephase) and the amount of frequency shift required to keep the locally generated carrier wave synchronized to the frequency of the received wave (the Doppler shift) [58]. The measurement observables from a GPS receiver are formed directly from these two observations and consist of a set of three distinct values known as the pseudorange, carrier phase, and Doppler shift.

Pseudorange

The pseudorange observable is one of the most basic and fundamental concepts in GPS. In essence, it is the real physical range between a satellite and receiver, including possibly hundreds of kilometers of error due to receiver clock bias [24]. Fortunately, this bias manifests itself uniformly in the observed pseudoranges to all visible satellites, so it is a correlated form of error that can be accounted for. Thus, the “pseudo” in *pseudorange* comes from the fact that all range values may be tainted significantly by a uniform receiver-side error.

Pseudorange is calculated quite simply by multiplying the time-of-flight (TOF) of a transmitted signal from satellite to receiver by the propagation speed of the radio signal which, in this case, equals the speed of light. The transmit time is known exactly since it is included in the navigation message, but the receive time is based on an inaccurate local clock; thus, the complexity in pseudorange measurement arises from synchronizing the receiver clock to GPS time with such precision that an accurate TOF can be measured.

In general, a GPS receiver is able to utilize the structure of the navigation messages and bit transitions encoded on the various satellite signals to synchronize its own internal clock to an accuracy of better than 1 ms. In fact, since PRN codes repeat themselves every millisecond, most receivers are able to detect when their synchronization has drifted over that threshold, limiting the worst-case bias to no more than 1 ms. Determination of *sub*-millisecond clock bias directly from the radio signals, however, is not possible. As such, by multiplying the observed time of flight of each radio signal (with a maximum error of 1 ms) by the speed of light, the GPS receiver is able to return a pseudorange to each visible satellite [45], with observation errors ranging anywhere from -300 to 300 km, corresponding to a receiver clock synchronization error in the range from -1 to 1 ms.

Assume now that the pseudoranges are known to at least four satellites. The primary source of error in these observations is the local clock bias from GPS time due to an imprecise local oscillator. It can be seen in Figure 2, taken from [24], that this bias manifests itself equally in determination of the local time and the pseudorange calculation. Thus, by subtracting the pseudorange (divided by the speed of light) from the local clock's estimate of the receive time, we are left with the *actual* transmit time (i.e. the time when the current point in the L1 signal left the corresponding satellite).

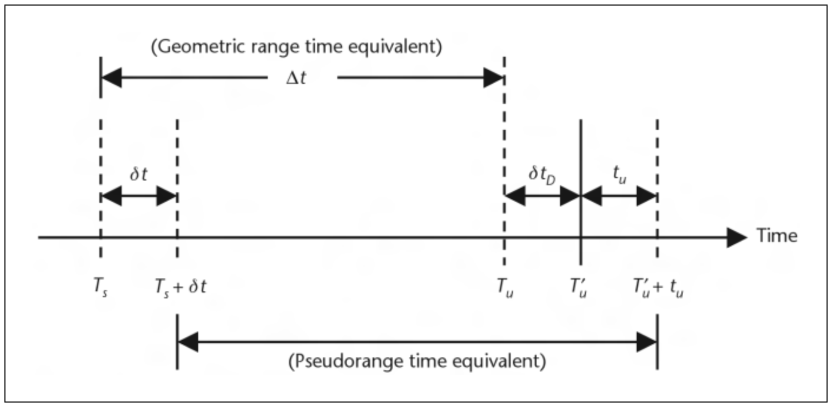

(Geometric range time equivalent)
Δt
δt
δt_D
t_u
Time
T_s
T_s + δt
T_u
T′_u
T′_u + t_u
(Pseudorange time equivalent)

Figure 2: Range measurement timing relationships

In this figure:

T_s = GPS system time when the signal left the satellite
T_u = GPS system time when the error-free signal would have reached the user
T'_u = GPS system time when the signal actually reached user with delay δt_D
δt = satellite clock offset from GPS time
t_u = receiver clock offset from GPS time

Since we know the actual satellite transmit times, the positions of the satellites (from their navigation messages), and the corresponding pseudoranges which include the receiver clock bias times the speed of light, the *actual* GPS system receive time can be found by estimating the receiver clock bias along with the X,Y,Z-coordinates of the receiver position [24].

In order to use the measured TOF and receiver clock bias to calculate pseudorange, the following generalized equation for receiver r and satellite s is used [5]:

$$P_r^s = (T_{rx} - T_{tx}) \cdot c \tag{1}$$

where T_{rx} is the local clock time of reception and T_{tx} is the satellite clock time of transmission. Since both times contain a clock bias, τ, from actual GPS time, t, (due to either satellite clock error or receiver clock error), the pseudorange equation can be rewritten:

$$\begin{aligned} P_r^s(t_{rx}) &= (t_{rx} + \tau_r) \cdot c - (t_{tx} + \tau^s) \cdot c \\ &= (t_{rx} - t_{tx})c + c\tau_r - c\tau^s \\ &= \rho_r^s(t_{rx}) + c\tau_r - c\tau^s \end{aligned} \tag{2}$$

where $\rho_r^s(t_{rx})$ is the physical distance the signal had to travel from time of transmission to time of reception; in other words, the geometric range. Additionally, the satellite clock bias is constantly measured by the Control Segment and transmitted in the navigation message, so most receivers will correct for this term ($c\tau^s$), making it negligible in Equation 2.

It should be noted that this formula represents a *generalized* model for pseudorange, neglecting relatively small errors such as atmospheric delay and multipath; however, these error sources must be taken into account to achieve a higher level of accuracy [26]. (See Chapter III for more detail about additional sources of error.)

Doppler Shift

While pseudorange is used in virtually all localization and navigation procedures, the method by which it is acquired makes it the least accurate of the measurement observables. Two other measurements known as *Doppler shift* and *carrier phase* are formed directly from the received L1 carrier signals, providing a means for much more accurate position estimation. Doppler shift is a well-known phenomenon that affects traveling wavefronts when two objects move at different speeds relative to one another [63]. In this case, a radio signal transmitted at a nominal frequency of f_0 from a satellite in motion will appear at the receiver with a slightly different frequency of f_R. This is significant because the amount of shift from the nominal frequency is mathematically related to the relative change in position of the satellite over time.

There are two methods by which GPS receivers typically measure this effect. The first and most common method arises simply from the way in which a GPS receiver maintains its satellite locks. It has been shown that satellite locks are achieved by maximizing an autocorrelation function between the received PRN code on the L1 signal and the same locally generated code shifted through time. It is apparent that the autocorrelation function must be maximized by aligning the received and generated code sequences over a significant amount of time (at least one full code cycle). This is only possible if the code being generated has the same frequency as the code being received.

When a receiver is carrying out its search-and-acquire procedure for the various PRN codes, it must try to align the codes not only in time, but also in frequency. As such, most receivers will scale the generated code according to a set of frequency bins centered around the nominal L1 frequency [42]. Upon finding a frequency value (and alignment) that results in a successful autocorrelation, the frequency can then be fine-tuned to maximize the autocorrelation result. The exact frequency that does so should be equal to the frequency of the received signal. As such, the instantaneous Doppler shift can be found by simply subtracting the nominal frequency from the received frequency: $f_D = f_R - f_0$.

It should be noted that the satellites and receivers are not moving relative to one another in a constant fashion; therefore, the Doppler shifts change over time. The receiver is constantly tracking this shift to maintain its satellite lock, with the result that every Doppler observation is an *instantaneous* observation that changes with each consecutive epoch.

The second method by which a receiver might calculate the Doppler shift to a satellite is by determining the *beat frequency* resulting from mixing the received GPS signal with the locally-generated one [5]. When two sine waves with slightly different frequencies are mixed together, a wave is produced with two frequency components equal to the sum and difference of the original waves' frequencies. Knowing that $f \equiv \frac{d\phi(t)}{dt}$, this can be written as:

$$\begin{aligned} S_1(t) \otimes S_2(t) &= A_1 sin2\pi\phi_1(t) \times A_2 sin2\pi\phi_2(t) \\ &= \frac{A_1 A_2}{2}[cos2\pi(\phi_2(t) - \phi_1(t)) - cos2\pi(\phi_2(t) + \phi_1(t))] \end{aligned} \tag{3}$$

By passing this resulting wave through a low-pass filter to remove the high-frequency component, a pure sine wave containing only the beat frequency (i.e. the Doppler shift) will remain:

$$S_B(t) = Bsin2\pi\phi_B(t) \tag{4}$$

where B is the amplitude of the resulting beat signal and $\phi_B(t)$ is the resulting phase difference which changes as a function of time (a.k.a. the beat frequency). Once the Doppler shift is known, it can be used to determine user velocity, signal anomalies, or even user position with the aid of additional observables.

Carrier Phase

An extremely valuable observable utilized in most high-precision applications is the carrier phase. Like Doppler shift, this type of observation is formed directly from the GPS carrier signal instead of TOF measurements; as such, accuracies in line with 1% of the wavelength of the L1 carrier wave (∼1.91 mm) are theoretically achievable using this type of observation [24].

The term "carrier phase" is confusing terminology which seems to indicate that the observation is some sort of phase angle. In reality, it is the number of whole and fractional L1 carrier cycles that exist between a satellite and receiver at a given time, or simply put, the geometric range between satellite and receiver in units of carrier cycles. The term *phase* is used because it is derived from an accumulation of phase over time.

It should be noted that phase is a property fundamentally related to pure sinusoidal waves. Since GPS uses a sine wave as a carrier with a well-known, constant L1 frequency, the carrier can be written in complex form as $y = Ae^{j2\pi(f_{L1}t+\theta)}$, where A is the signal amplitude, f_{L1} =1575.42 MHz, and θ is the phase in radians at time $t = 0$. For an L1 sine wave transmitted from satellite s at time t_{tx} and received by receiver r at time t_{rx}, the signal can be represented as [40]:

$$y_r^s(t_{rx}) = Ae^{j2\pi\phi_r^s(t_{rx})} \tag{5}$$

where $\phi_r^s(t_{rx})$ is the received phase in cycles. The following figure depicts how phase relates to frequency and signal amplitude in a sine wave over time (taken from [5]):

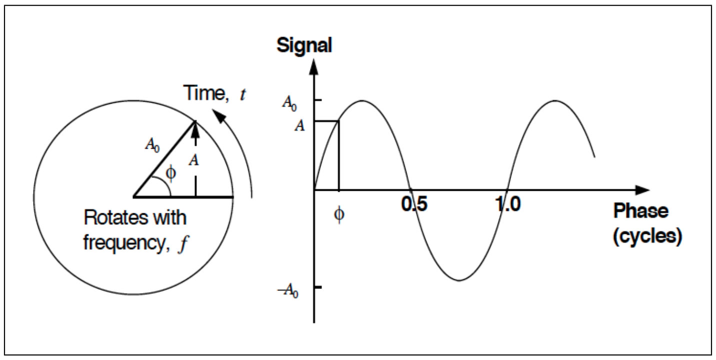


Figure 3: Signal amplitude as a function of phase

For ideal plane waves propagating in free space, the behavior of ϕ_r^s is well-known; therefore, assuming t_{tx} and t_{rx} are the transmit and receive times of the exact same point in the sine wave, the received phase in Equation 5 can be written,

$$\begin{aligned}\phi_r^s(t_{rx}) &\equiv (f_{L1}t_{tx} + \phi^s(0)) \mod 1 \\ &\equiv (f_{L1}t_{rx} - \frac{\rho_r^s(t_{rx})}{\lambda_{L1}} + \phi^s(0)) \mod 1\end{aligned} \tag{6}$$

where $\phi^s(0)$ is the initial transmit phase at the satellite, λ_{L1} is the L1 carrier wavelength, and $\rho_r^s(t_{rx})$ is the satellite-receiver range, with $\rho_r^s(t_{rx}) = (t_{rx} - t_{tx})c$ [40].

Geometrically, we have stated that we are trying to use the carrier phase measurement to determine a receiver-satellite range. In this case, Equation 6 can be rearranged to give the satellite range in units of carrier cycles as:

$$\frac{\rho_r^s(t_{rx})}{\lambda_{L1}} = (t_{rx} - t_{tx})f_{L1} = f_{L1}t_{rx} + \phi^s(0) - \phi_r^s(t_{rx}) + N_r^s(t_{rx}) \tag{7}$$

where N_r^s is the integer number of full cycles (wavelengths) between the satellite and receiver (arising from the modulo operation in Equation 6). In order to calculate the range, the receiver must therefore know or be able to estimate the receive time, t_{rx}, the initial satellite phase offset, $\phi^s(0)$, the received RF phase, ϕ^s_r, and the integer number of cycles between satellite and receiver, N^s_r.

Unfortunately, receivers cannot calculate this range value directly because the initial satellite phase offsets, integer ambiguities, and receiver clock biases are unknown, and Equation 7 represents the instantaneous phase at the exact time instant that the transmitted signal phase arrives at the receiver given a constant frequency. If there were no errors and the GPS satellites were geostationary, this equation would hold true. In reality, the satellites are in constant motion relative to us; thus, the received frequency is **not** constant nor equal to the ideal L1 frequency. In order for the phase value to be useful in terms of GPS positioning, it is necessary to know the difference between the received frequency and the ideal, transmitted frequency – in other words, the Doppler shift. Luckily, the previous section explained how this shift can be found.

Since Doppler shift can be measured to much greater precision than the codephase used in pseudorange determination, use of this value will greatly increase the potential accuracy in position estimation. We know that Doppler shift arises from the change in relative position (i.e. velocity) of a satellite to a receiver over time. Thus, by integrating the Doppler shift values at each epoch, a receiver is able to keep track of the line-of-sight changes in position of each satellite relative to where it was when the Doppler integration began. By initializing a simple integer counter that increases every time a full cycle of the beat signal has passed, the receiver can calculate how far the satellite has moved since its time of lock in units of L1 carrier cycles. Mathematically, this looks like [24]:

$$\phi^s_r(t_n) = \phi^s_r(t_{n-1}) + \int_{t_{n-1}}^{t_n} f_D(\tau)d\tau + \phi^s_{r,frac}(t_n) \tag{8}$$

where t_n is the time associated with epoch number n, ϕ^s_r is the accumulated phase at the specified time, f_D is the Doppler shift at the specified time, and $\phi^s_{r,frac}$ is the fractional remainder of the received phase. The reason the fractional portion is separated out from the accumulated Doppler shift is because receivers usually implement this function using simple integer counters [24]. The counter will increase by 1 whenever a full Doppler cycle has completed, which means that there will exist some fractional amount of phase remaining at the end of each epoch that a receiver can easily identify from its carrier-phase tracking loops. Due to this method of calculation, carrier phase is oftentimes referred to as *accumulated* or *integrated Doppler shift*.

It is apparent that Equation 8 constitutes a highly accurate phase accumulation function, but that it neglects the base case of the number of cycles already present between a satellite and receiver at the arbitrary startup time, t_0. From Equation 7, we can see that this omission includes the initial signal phase from the satellite at the time of transmission, the initial signal phase of the replica receiver wave at the time of satellite lock, and the integer number of wavelengths between satellite and receiver. Most receivers will simply initialize these unknowns to zero and return the carrier phase observable under the assumption that $\phi_r^s(t_0) = 0$.

In order to use this carrier phase observable, we must formulate a model which incorporates the measurement aspects just discussed. This is a simple enough task, given our knowledge that for an incoming L1 signal with phase $\phi_r^s(T_{rx})$ at receiver time T_{rx} that left satellite s at satellite time T_{tx}, the following equations are true [5]:

$$\begin{aligned} \phi^s(T_{tx}) &= (f_{L1}T_{tx} + \phi^s(t_0)) \mod 1 \\ \phi_r(T_{rx}) &= (f_{L1}T_{rx} + \phi_r(t_0)) \mod 1 \end{aligned} \tag{9}$$

The difference between these two equations will give the geometric satellite-receiver range in units of carrier cycles at time T_{rx}:

$$\Phi_r^s(T_{rx}) = f_{L1}(T_{rx} - T_{tx}) + \phi_r(t_0) - \phi^s(t_0) - N_r^s(T_{rx}) \tag{10}$$

As mentioned in the section on pseudorange formulation, both T terms include some amount of clock bias, τ, from GPS time, t. We have also shown in Equation 8 that the carrier phase observable integrates continuously once a satellite lock is established, with the result that the $N_r^s(T_{rx})$ term actually models the integer number of wavelengths between satellite and receiver at the reference time, t_0, and remains constant as long as a satellite lock is maintained. In other words, $N_r^s(T_{rx}) = N_r^s(t_0) = N_r^s$. Therefore, the formula can be re-written in terms of GPS time as:

$$\Phi_r^s(t_{rx}) = f_{L1}(t_{rx} - t_{tx}) + f_{L1}\tau_r - f_{L1}\tau^s + \phi_r(t_0) - \phi^s(t_0) - N_r^s \tag{11}$$

Again, the $f_{L1}\tau^s$ term can be virtually eliminated based on satellite clock correction data in the navigation messages, so it will fall out of this equation once compensated for.

We can note that the final three terms of this equation are constants which together indicate the total number of carrier cycles between satellite and receiver at the time of satellite lock. In many cases, these terms will be lumped together as $B_r^s \equiv (\phi_r(t_0) - \phi^s(t_0) - N_r^s)$ [5]. The reason to keep them separate is that several differencing techniques will result in the fractional ϕ terms completely canceling out, leaving only the integer ambiguities to deal with.

For the sake of symmetry and ease of combining carrier phase and pseudorange measurements, we will convert the carrier phase observable into a so-called *carrier range* by multiplying it by the wavelength of the L1 sine wave. This results in an observation in units of meters, like pseudorange, with a very similar-looking equation:

$$\begin{aligned} L_r^s(t_{rx}) &\equiv \lambda_{L1}\Phi_r^s(t_{rx}) \\ &= \lambda_{L1} f_{L1}(t_{rx} - t_{tx}) + \lambda_{L1} f_{L1}\tau_r - \lambda_{L1} f_{L1}\tau^s + \lambda_{L1} B_r^s \\ &= c(t_{rx} - t_{tx}) + c\tau_r - c\tau^s + \lambda B_r^s \\ &= \rho_r^s(t_{rx}) + c\tau_r - c\tau^s + \lambda B_r^s \qquad (12) \end{aligned}$$

In the rest of this dissertation, we will use the term *carrier phase* to refer to the observable in units of L1 carrier cycles and *carrier range* to refer to it in terms of meters, as shown above.

CHAPTER III

GPS ERROR SOURCES AND OBSERVATION MODELS

Thus far, discussions regarding the observables that a GPS receiver uses to calculate its position have been taken from the point of view of an idealized world. In reality, there are a variety of effects and conditions, in addition to clock synchronization errors, that alter the precision of the observations and make it much more difficult to achieve highly accurate position localization. This chapter gives an overview of these error sources and presents ways in which some of them can be mitigated, if not removed completely.

Known Sources of Error

Satellite Clock Bias

Each GPS satellite uses a highly stable atomic clock to ensure that it is precisely aligned to actual GPS time; however, the Control Segment (CS) allows for these clocks to drift up to 1 ms from GPS time before sending synchronization corrections [24]. Essentially any bias from actual GPS time means that the satellite is transmitting its PRN code at a slightly incorrect time. Since a GPS receiver calculates pseudorange based on TOF values times the speed of light, 1 ms of satellite clock error will correspond to approximately 300 km of error in the pseudorange. The CS is constantly monitoring these errors, however, and sends correction values in the navigation message of each satellite.

Specifically, a satellite's navigation message will include its clock bias (denoted a_{f0}) at some reference time, t_{oc} – usually the time when the clock correction parameters were estimated and uploaded to the satellite – its clock drift (a_{f1}), and its frequency drift (a_{f2}) [13]. These parameters can be used by the GPS receiver to correct the signal transmit time according to the following formula:

$$\delta t_{clk} = a_{f0} + a_{f1}(t - t_{oc}) + a_{f2}(t - t_{oc})^2 \tag{13}$$

where t represents the current receive time epoch.

It should be noted that these corrections are **estimates** of the clock parameters based on previous behavior and are not completely accurate. The atomic clocks on each satellite have stabilities of about 1 part in 10^{13} per day [39]. After one full day (~10^5 s), the clock error will have accumulated to about 10^{-8} s, or roughly 3.5 m. Empirically, residual errors on the order of 0.8-4 m usually remain, depending on the type of satellite and the age of the estimated parameters, with precision degrading over time. In most cases, clock parameters are re-estimated and uploaded to each satellite at least once per day such that, according to a 2004 study, the 1-sigma error due to satellite clock bias averaged over the entire age of data (up to 24 hours) was 1.1 m [24].

Receiver Clock Bias

As with GPS satellite clocks, the internal clock in each GPS receiver is subject to bias from GPS time. In fact, these biases are much larger and change more rapidly than satellite biases since these clocks are driven by low-cost internal oscillators instead of the atomic clocks used on board GPS satellites. The effect of receiver clock bias on each of the observables is identical to that of satellite clock bias; namely, the introduction of significant error based on the magnitude of the bias times the speed of light [24].

Since this bias is large and changes so rapidly, it is frequently handled by estimation in the localization procedures for each GPS receiver. This is possible because the bias is uniform in the observations to all satellites, enabling it to be modeled by a single error parameter for each receiver in addition to the X, Y, and Z coordinate parameters.

One further complication arises from the fact that receiver clocks drift out of synchronization from GPS time so quickly that they must be reset fairly often. We have discussed how PRN codes repeat every millisecond and how the amount of shift required to align the locally generated code with the received code is the basis for forming the pseudorange observation. Since each code is only 1 ms long, this shift is only meaningful (in that it is not a modulo operation) for clock biases less than 1 ms. As such, most GPS receivers will internally keep track of the estimated receiver clock bias and perform a hard reset to synchronize back to GPS time when the bias approaches or reaches ± 1 ms. These resets produce noticeable "jumps" in the observables, since all counters must be re-initialized, causing the measurement period for one observation, by necessity, to be longer or shorter than the rest. Luckily, observation jumps are easily detectable, and care must be taken to account for them in any GPS localization software.

Satellite Orbital Errors

The GPS satellite constellation has been very precisely designed, with each satellite following an extremely specific orbit around the Earth. Nonetheless, various forces such as solar radiation pressure, gravitational forces, tidal movements, infrared radiation, and deformations of the Earth can cause slight shifts and inaccuracies in the pre-defined orbital paths of the satellites [8]. These shifts cannot be anticipated and only become apparent during *post facto* analysis.

Luckily, orbital errors are usually small, and furthermore, only the portion of the error that is projected onto the line-of-sight vector from satellite to receiver has any effect on the ranging observables [39]. Typical magnitudes for such orbital errors range from 1 to 6 meters, with the average effective 1-sigma error on the observables approximately 0.8 m [24].

Atmospheric Effects

When discussing the formation of the various GPS observables, we made the assumption that all radio signals were traveling through a vacuum at the speed of light. This is not strictly true since all signals must travel through the various layers of the Earth's atmosphere. Two layers in particular cause deformations of the radio waves, namely the ionosphere and the troposphere. Since the index of refraction – and thus the radio wave propagation speed – is different for each of these layers, they will contribute some amount of error to each GPS observable formed under the pretense of speed-of-light propagation with no refraction between layers.

Ionospheric Delay

The ionosphere is a region of the atmosphere spanning from about 70 km to 1,000 km above the surface of the Earth. It is filled with free electrons that cause it to be a *dispersive* medium (a medium in which propagation speeds are a function of the frequency of the wave) [24]. It is beyond the scope of this dissertation to discuss material propagation properties, but it is important to note that the propagation velocity of the signal's carrier phase differs from the velocity of the PRN codes used to carry the signal information.

The velocity associated with the carrier phase of the signal is known as *phase velocity* while the velocity associated with the PRN codes is known as *group velocity.* It can be observed that the phase velocity always exceeds the group velocity in GPS signals. Also, the advance of the phase velocity with respect to vacuum-propagation speeds is equal to the retardation of the group velocity. In terms of GPS observables, this means that all signal information (PRN codes and navigation data) are delayed by exactly the same amount that the carrier phase is advanced. This phenomenon is known as *ionospheric divergence.* Therefore, when looking at the corresponding errors (in meters) of the pseudorange and carrier range observables, the magnitudes of the errors due to the ionosphere are identical, but with opposite signs.

Since the amount of delay a radio wave experiences in the ionosphere is proportional to the amount of time it spends there, ionospheric delay is a function of not only the location where the signal entered the ionosphere (known as the “pierce point”), but also the elevation of the satellite with respect to the receiver (as radio waves from lower-elevation satellites necessarily have more atmosphere to penetrate than waves that entered at a higher angle). In fact, the error incurred on waves from satellites at low elevations is almost three times greater than the error on signals coming in from the zenith.

Each satellite transmits ionospheric correction parameters for use by the receiver in a *Klobuchar model* which can reduce the amount of error due to the ionosphere by approximately 50%. Without correction, ionospheric delay can introduce errors ranging from 3-9 m at night and 15-45 m during the day. A typical 1-sigma error value for ionospheric delays averaged over all locations and all elevation angles is 7 m.

Tropospheric Delay

The troposphere is located directly below the ionosphere and is a *non-dispersive* medium, meaning that all phase and group velocities are equivalent in this layer. The delay arises from the index of refraction which, in this layer, is dependent upon the local temperature, pressure, and relative humidity. Uncompensated errors due to the troposphere can range from 2.4 m for satellites at the zenith to 25 m for satellites approximately 5° above the horizon. Using advanced modeling techniques, it is possible to minimize, but not completely eliminate the errors arising from this layer.

The following figure, taken from [25], shows how the various layers of the atmosphere contribute to the delay of the radio signal from satellite to receiver:

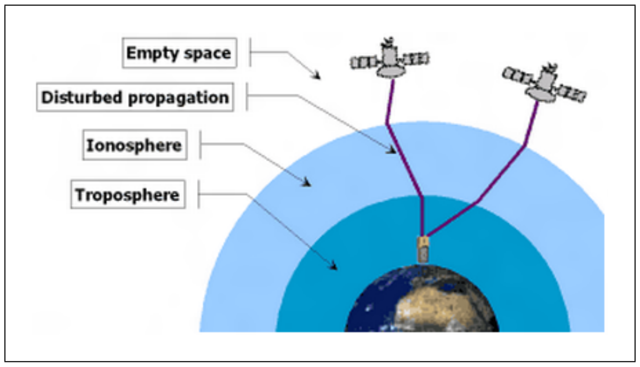


Figure 4: Propagation of radio waves through the Earth’s atmosphere

Relativistic Effects

Einstein's Theory of Relativity is an omnipresent, but often overlooked concept that becomes quite apparent in terms of GPS satellites and signals. In general, it states that time runs more slowly for objects experiencing fast motions when viewed from a slower-moving frame of reference (or conversely, time runs faster for slower-moving objects). GPS satellites move at a speed of 3,874 m/s relative to the Earth, causing their clocks to run more slowly than the clocks on Earth. The amount of time dilation experienced at these speeds causes clock inaccuracies of approximately 7.2 microseconds per day [25]. Additionally, the theory states that time runs more quickly for clocks experiencing lower gravitational potentials. The satellites orbiting $\tilde{}20,000$ km above the Earth experience four times less gravity than our Earth clocks, resulting in an apparent time speedup of about 45 microseconds per day. These two effects negate each other, resulting in the appearance that satellite clocks are running 38 μs a day faster than clocks on Earth [59].

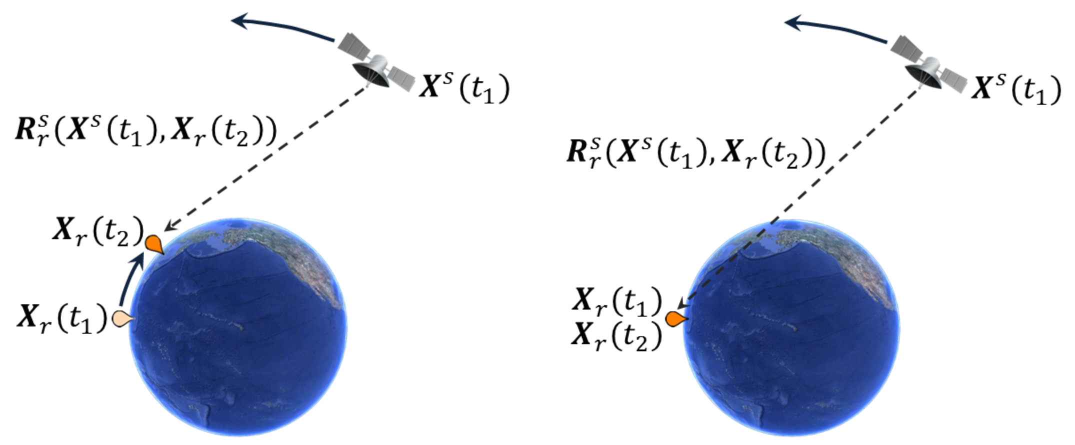
The architects of GPS decided to overcome these effects by adjusting the satellite clock frequencies from the nominal 10.23 MHz to 10.22999999543 MHz prior to launch. When radio signals are transmitted at this frequency, they appear, on average, to arrive on Earth at the nominal frequency of 10.23 MHz, and the satellite clocks stay synchronized to GPS time as viewed from an Earth frame of reference [24]. Although this adjustment allows satellite clocks to remain synchronized and removes the bulk of the relativistic error, there still exists a small clock deviation due to the fact that satellite orbits are slightly elliptical, meaning both their speeds relative to the Earth and their gravitational potentials change as a periodic function of their locations in orbit. Left uncorrected, these relativistic effects can contribute a maximum of 70 ns of clock error (21 m in range) to the observations. Luckily, these effects can be almost completely eliminated by compensating the received clock time according to:

$$\Delta t_r = Fe\sqrt{a}\sin E_k \tag{14}$$

where all parameter values are either constant or transmitted by the satellite with:

$F = -4.442807633 \times 10^{-10}\ s/m^{1/2}$
$e =$ satellite orbital eccentricity
$a =$ semimajor axis of the satellite orbit
$E_k =$ eccentric anomaly of the satellite orbit

There also exists another relativistic effect known as the *Sagnac effect* due to rotation of the Earth during the time of signal transmission [4]. Once a GPS signal leaves a satellite, it must travel a finite amount of time before it reaches a receiver. Since the Earth is rotating during this time, the propagation time (and hence the raw observations) will either grow or shrink as the receiver rotates away from or toward the incoming signal. If receiver and satellite positions were calculated in a non-rotating Earth-Centered Inertial (ECI) frame, this would not cause a problem since the calculated satellite positions at the time of transmission would not rotate with the Earth and therefore would be the correct distance from the receiver. In GPS, however, satellites transmit their ephemeris data to allow users to calculate their positions in an Earth-Centered, Earth-Fixed (ECEF) coordinate frame which rotates along with the Earth. As such, the calculated satellite position will be in terms of the rotated receiver position at the time of signal reception instead of at the time of signal transmission, regardless of the fact that the pseudorange observable will be based on the full propagation time. Figure 5 explains this phenomenon:



(a) ECI Frame: Since the coordinate system is fixed in space, the receiver position, $\mathbf{X}_r(t_1)$, at the time of signal transmission, t_1, is not equal to its position, $\mathbf{X}_r(t_2)$, at the time of signal reception.

(b) ECEF Frame: Since the coordinate system rotates with the Earth, the receiver positions at the time of signal transmission and reception appear identical, $\mathbf{X}_r(t_1) = \mathbf{X}_r(t_2)$

Figure 5: The Sagnac Effect

Since the ECEF coordinate system rotates along with the Earth, it is clear from Figure 5b that the signal propagation range from satellite to receiver, $\mathbf{R}_r^s = ||\mathbf{X}^s(t_1) - \mathbf{X}_r(t_2)||$, computed under the assumption that the coordinate system has remained stationary between t_1 and t_2 will be incorrect and will not coincide with the observed pseudorange and carrier phase measurements. In order to account for this problem, the user must correct the computed satellite positions for the amount of rotation the receiver experienced during the time of signal transmission. Corrections for this effect are commonly referred to as *Earth rotation corrections* and a number of algorithms exist for computing them. Left uncorrected, the Sagnac effect can contribute to satellite position errors on the order of 30 m, so this type of effect cannot be ignored.

Antenna Phase Center Offsets

Another problematic source of error arises from both the type of antenna used to receive the GPS signals, as well as the specific antenna itself, since small manufacturing anomalies can impact the characteristics of this error source. The problem arises from the fact that the electrical phase center (i.e. the location where the GPS signals are received and collected by the antenna) does not usually coincide with the geometric center of the antenna, also known as the Antenna Reference Point (ARP) [19].

Compounding this problem is the fact that the signal collection point varies based on the direction (azimuth and elevation) of the incoming signal. The average of all of the signal collection points for all possible signal directions is called the mean electrical phase center, or more commonly the phase center offset. This value can range from a few millimeters to several centimeters. The individual, instantaneous offsets used to calculate the mean phase center are called Phase Center Variations (PCV) — $\Delta\phi_{PCV}(\alpha, e)$ in Figure 6 below, taken from [61] — and are usually larger than the mean electrical phase center.

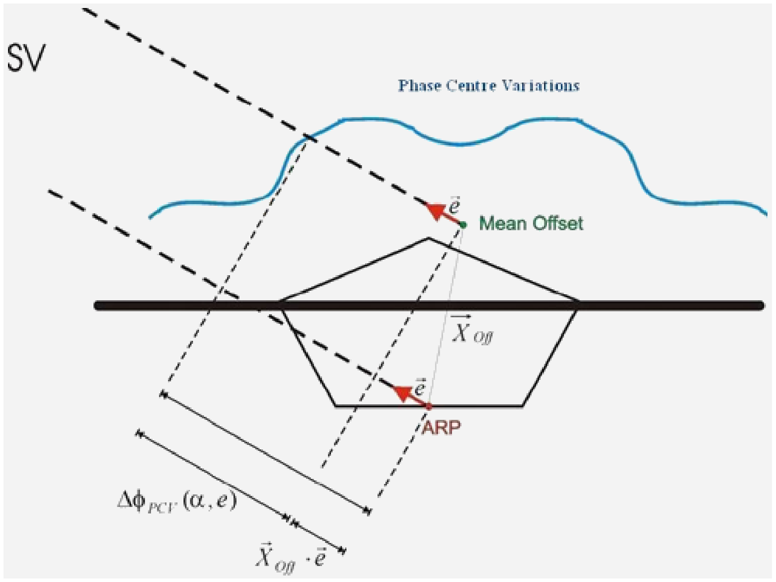


Figure 6: Antenna phase center offsets and variations

The carrier phase observable can be corrected for antenna phase center offset as a function of satellite azimuth, α, and elevation, e, according to the following formula:

$$\Delta\epsilon_{\phi}(\alpha, e) = \Delta\phi_{PCV}(\alpha, e) + \overrightarrow{X_{off}} \times \overrightarrow{e_{\alpha,e}} \tag{15}$$

Unfortunately, the only way to find the phase center offset and PCVs for an antenna is by experimentation. Most receiver manufacturers will report a tolerance range for phase center offset, and some receivers actually have pre-calculated offset tables based on empirical data from extremely large sample sets [54].

While phase center offsets and variations can potentially introduce measurable amounts of error into the raw observations, the majority of error manifests itself in the height component of the location estimate. Therefore, this type of error is much more important to consider when calculating 3D positions. Additionally, antennas of the same type from the same manufacturer tend to have similar mean phase center offsets, so using the same antenna for relative positioning will result in more accurate results than mixing a variety of antenna types [19].

Multipath

One of the most serious and difficult-to-remedy errors in GPS receiver measurements is due to a phenomenon called *multipath*. Multipath occurs when satellite signals (which are transmitted in all directions) reflect off of objects nearby to the receiver, causing identical radio signals to be picked up one or more time instants after the correct line-of-sight signal has already reached the receiver. The magnitude of these additional signals varies greatly depending on the specific environment, but they can arise from any reflective surface such as trees, buildings, people, or even the ground.

The reason these signals are so hard to deal with is because they are not only time-dependent, but also receiver-, azimuth-, and elevation-dependent. As long as the multipath delay is large and the reflections arrive a measurable length of time greater than the PRN code period, the receiver can detect and discard the erroneous signals (since it is already locked onto and tracking the correct, **direct** signal, and erroneous signals will appear inconsistent). If, however, the multipath delay is short and unresolvable, it will distort the autocorrelation function between the received composite signal and the locally generated one, as well as the composite phase of the incoming signal, causing unpredictable errors in both the pseudorange and carrier phase [24].

Typical 1-sigma multipath error values in a benign environment average about 2 cm for the carrier phase measurement and 20 cm for the pseudorange. These are reasonable values; however, as the environment degrades with the addition of reflective surfaces, the pseudorange error can reach a maximum of one full PRN code length (293 m). It is important to note that multipath errors affect the carrier phase **significantly** less than the pseudorange, with maximum errors topping out around 5 cm [13].

Carrier Cycle Slips

A GPS receiver will fail to return a pseudorange observable if the satellite signal is dropped for any amount of time causing the PRN autocorrelation function to diverge. This is known as a *loss of lock*, and it is easily detectable by any GPS receiver. Far more difficult to detect are minute losses of the satellite carrier signal which are regained quickly enough that the PRN code can still be detected and correlated. In this case, the autocorrelation function continues to produce a result, and the receiver does not know that any problem has occurred.

This problem manifests itself exclusively in the carrier phase observable (which is supposed to indicate the number of carrier wavelengths between a satellite and receiver). Since one full wavelength is equal to about 19 cm, a “cycle slip” of only five cycles will produce almost a full meter of error in the carrier range observable. For this reason, it is imperative that GPS localization software keep track of the carrier phase observable and try to detect any cycle slips. Numerous methods for doing this exist [15, 49], including monitoring the Doppler shift and predicting what the carrier phase observable should be at the next epoch assuming linearly smooth satellite motion between consecutive measurements.

If a cycle slip is detected, the GPS receiver can either attempt to fix the slip by inserting the correct number of cycles, or it can simply discard the observations for that satellite until the signal has regained a steady state.

Receiver Measurement Noise

The main sources of error inherent to the GPS receiver itself include thermal noise jitter and interference effects on the receiver tracking loops. Experiments have shown the 1-sigma error on pseudorange measurements due to receiver noise to be less than a decimeter, while the 1-sigma error on carrier phase measurements is approximately 1.2 mm [24].

Observation Models and Positioning Techniques

The various measurement observables that are output and used by GPS receivers in their position estimation procedures were introduced in “Section: *Measurement Observables*” in Chapter II, while this chapter has discussed the possible sources of error that corrupt these observables, as well as ways by which they may be mitigated. The synthesis of these two types of analysis results in the creation of observation models which allow us to see how manipulating and combining the observations can result in accurate methods of localization.

Single Point Positioning Model

The simplest and most basic set of observation models is used primarily in standalone, point-positioning algorithms, providing absolute coordinates for a single receiver anywhere on Earth. This is the default operating paradigm for GPS, and all other observation models stem from it. In general, these modeling equations attempt to assign geometric error contributions to each of the sources of error discussed in the previous section to determine their total impact on each of the pseudorange and carrier range observations, and by extension, the User-Equivalent Range Error (UERE) resulting from use of these observations in a point positioning algorithm.

Pseudorange

A generalized pseudorange equation was given in "Section: *Pseudorange*" in Chapter II, and the full pseudorange observation model is a direct extension of this, including all of the error sources discussed thus far:

$$P_r^s(t) = \rho_r^s(t) + c\tau_r(t) - c\tau^s(t) + d_{iono}^s + d_{tropo}^s + M_{r,P}^s(t) + \epsilon^s + \epsilon_P(t) \tag{16}$$

where:

$\rho_r^s(t)$ is the geometric range from satellite to receiver at time t
$c\tau_r(t)$ is the pseudorange error due to receiver clock bias at time t
$c\tau^s(t)$ is the pseudorange error due to satellite clock bias at time t
d_{iono}^s is the error due to ionospheric delay
d_{tropo}^s is the error due to tropospheric delay
$M_{r,P}^s(t)$ is the pseudorange error due to multipath at time t
ϵ^s is the error due to satellite orbital offsets
$\epsilon_P(t)$ is the pseudorange error due to receiver noise at time t

Note that the satellite orbital errors and atmospheric delay terms are modeled as constants in the equation above. Since satellite position is calculated according to a set of smooth Keplerian models, any instantaneous orbital errors will change extremely slowly over time such that they can be viewed as constants for each satellite over relatively long intervals.

The atmospheric delay terms also depend loosely on time; however, they change very slowly (with temporal correlations up to 50 minutes long), so that they can be modeled as constants from one epoch to the next [14]. This is significant because it allows for equation differencing through time to eliminate these types of errors. It must always be kept in mind that although epoch-to-epoch differencing virtually eliminates these errors, differencing over more significant amounts of time results in less and less of the error being removed. It should also be noted that the atmospheric delay terms are not dependent on a specific receiver as long as multiple receivers are located in a geographically similar region. Receivers with baselines shorter than 200 km under very disturbed atmospheric conditions or up to 1000 km under ideal conditions experience negligible atmospheric delay differences relative to one another and can be assumed to be identical [14].

Finally, any antenna phase center offsets are also neglected in this model since they are introduced solely by the manufacturing processes used to create the antenna and can be resolved or minimized by carefully matching the antenna to the application for which it is most suitable.

Carrier Range

Again, a generalized equation for carrier range was given in "Section: *Carrier Phase*" in Chapter II, but a better model will include the error sources discussed in this chapter. It is important to note the similarities of this model to that of the pseudorange:

$$L_r^s(t) = \rho_r^s(t) + c\tau_r(t) - c\tau^s(t) + \lambda B_r^s - d_{iono}^s + d_{tropo}^s + M_{r,L}^s(t) + \epsilon^s + \epsilon_L(t) \tag{17}$$

where:

$\rho_r^s(t)$ is the geometric range from satellite to receiver at time t
$c\tau_r(t)$ is the carrier range error due to receiver clock bias at time t
$c\tau^s(t)$ is the carrier range error due to satellite clock bias at time t
$\lambda B_r^s \equiv \lambda(\phi_r(t_0) - \phi^s(t_0) - N_r^s)$ is the range ambiguity at the time of satellite lock
d_{iono}^s is the error due to ionospheric delay
d_{tropo}^s is the error due to tropospheric delay
$M_{r,L}^s(t)$ is the carrier range error due to multipath at time t
ϵ^s is the error due to satellite orbital offsets
$\epsilon_L(t)$ is the carrier range error due to receiver noise at time t

The main differences in this model versus the pseudorange model are the additional satellite range bias term, λB_r^s, and the opposite sign for the ionospheric delay term, d_{iono}^s. The satellite range bias term was defined and explained in "Section: *Carrier Phase*" in Chapter II, and the reason for the ionospheric sign inversion is *ionospheric divergence* as discussed in "Section: *Ionospheric Delay*" in Chapter III. To reiterate, the magnitude of errors due to multipath and receiver noise are **much** lower for carrier range observations than for the pseudorange.

Absolute Localization

In general, the standalone carrier range model is rarely used by itself for localization of a single GPS receiver. Instead, the error terms described in the previous chapter are minimized in the pseudorange equation as much as possible, and localization is carried out using a mathematical concept called trilateration [23]. The idea behind trilateration is simple: it is possible to determine the coordinates of an arbitrary reference point in space given a set of position coordinates and their respective distances from the reference. Geometrically, this can be modeled by finding the single intersection point of a number of spheres with known center coordinates (satellite positions) and radii (ranges):

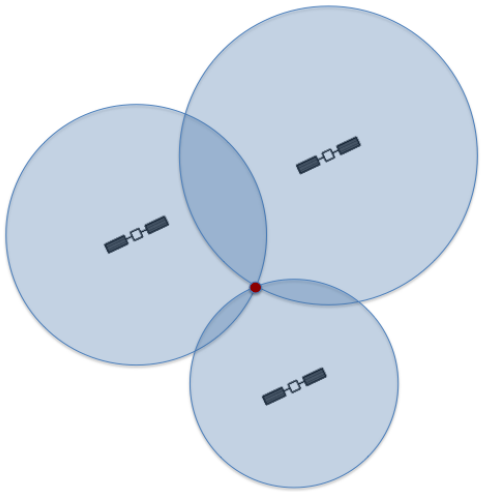


Figure 7: Localization using trilateration

As shown in "Section: *Pseudorange*" in Chapter III, however, all pseudorange observations for a given epoch will have a uniform error component due to the receiver's local clock offset from GPS time. As such, we cannot simply use the reported pseudorange values as actual satellite ranges in the trilateration procedure described above. Any amount of range error will result in a set of spheres centered around each of the satellites that do not intersect at exactly one and only one point. In fact, depending on the satellite geometry, the spheres may not intersect at all.

One further complication arises from the fact that pseudoranges contain errors not only due to receiver clock bias, but also all of the errors discussed in Chapter III, with the full model described by Equation 16. Even taking into account all other sources of error, the receiver clock bias dominates with the potential for up to ~300 km of error. As such, it is always modeled as a fourth localization parameter. The remaining errors usually account for anywhere between 10 to 20 meters of error per pseudorange observation [13]. For this reason, the spheres of potential receiver locations around each satellite will generally not intersect at only one point; however, there will tend to be a large number of intersections focused around a central area which define a *region* of possible receiver locations.

Assuming that we have already corrected the pseudorange observations for any additional sources of error excluding the uniform receiver clock bias, the equation representing pseudorange is:

$$P_r^s = \rho_r^s + c\tau_r = \sqrt{(X^s - x_r)^2 + (Y^s - y_r)^2 + (Z^s - z_r)^2} + c\tau_r \tag{18}$$

For n satellites, we will have n pseudorange observations (P_r^1, P_r^2, ..., P_r^n) with 4 unknowns (x_r, y_r, z_r, and $c\tau_r$). Therefore, we can see that by adding a fourth dimension to the intersection of spheres model, namely the time offset ($\epsilon = c\tau_r$ in Figure 8), it is once again possible to find a common intersection point.

Using this model and noting that we cannot fully account for all of the other error sources in the model, it becomes apparent that these spheres will still never intersect at exactly one point in real-world situations. As such, most GPS applications will use the simplified pseudorange model in Equation 18 in a least-squares solver to find a single set of coordinates that minimizes the error in a system of equations containing observations to the various satellites.

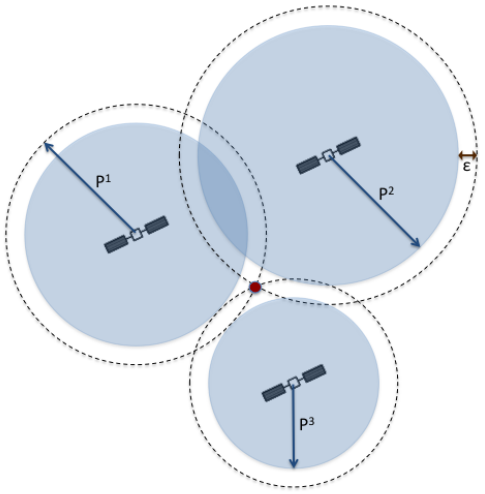


Figure 8: 2D localization using trilateration with receiver clock error

Dual-Receiver Baseline Models

While single point positioning is a necessary and well-studied area of GPS, recent research has led to advances in a much more accurate realm of GPS localization techniques known as relative positioning. As the name indicates, relative positioning consists of localizing one or more GPS nodes relative to either a reference node or to one another. In general, this equates to solving for the so-called *baselines* between pairs of receivers. Since we are now dealing with applications involving the relative locations between two or more nodes, additional sets of modeling equations have been developed to more accurately remove many of the errors that dictate the highest level of accuracy achievable by single point positioning techniques.

As in point positioning, the most basic form of relative positioning involves simple manipulation of the primary measurement observables: the pseudorange and carrier range. The primary way in which these observables are used in relative positioning is through a manipulation of the observations known as *differencing*. As the name implies, the differencing operation quite literally takes the difference between two observations (and their corresponding models) to eliminate some of the common error sources present in both observations. There are three commonly accepted differencing variations — single, double, and triple-differencing — corresponding to the number of subtraction operations used in the model. Each of these models can be used to eliminate a different set of error sources in the ranging signals while magnifying others.

Single-Differencing Model

The most simple differencing operation, called "single-differencing" or "between-receivers differencing," involves subtracting the observations (either carrier range or pseudorange) from two different receivers, j and k, to the same satellite, s, for a single time epoch [56]. Mathematically, this results in the following models (where we have combined the multipath and measurement noise terms into a single error, ϵ_r^s, since they are practically inseparable from one another):

$$\begin{aligned}
\Delta P_{jk}^s &= P_k^s - P_j^s \\
&= (\rho_k^s - \rho_j^s) + c(\tau_k - \tau_j) + (\epsilon_{P,k}^s - \epsilon_{P,j}^s) \\
&= \Delta\rho_{jk}^s + c\Delta\tau_{jk} + \Delta\epsilon_{P,jk}^s \\
\Delta L_{jk}^s &= L_k^s - L_j^s \\
&= (\rho_k^s - \rho_j^s) + c(\tau_k - \tau_j) + \lambda(B_k^s - B_j^s) + (\epsilon_{L,k}^s - \epsilon_{L,j}^s) \\
&= \Delta\rho_{jk}^s + c\Delta\tau_{jk} + \lambda\Delta B_{jk}^s + \Delta\epsilon_{L,jk}^s
\end{aligned} \tag{19}$$

You will note that many of the error sources present in observation model Equations 16 and 17 are not present in the single-differenced model, including satellite clock errors, orbital errors, ionospheric delay, and tropospheric delay. Since these errors are satellite and/or location-dependent, they completely cancel out when two observations to the same satellite in the same geographic region are differenced [18].

A set of single-differenced pseudorange observations can be used in a linear least squares solver in much the same way as the zero-differenced (raw) observations are used in point positioning. Assuming that one of the participating receivers, j, in the single-differenced model is static and its location is precisely known, the unknowns are the roving receiver's 3D coordinates and the relative clock bias between the two receivers (ignoring any unmodeled errors, ϵ, due to receiver noise or multipath); thus, a pseudorange solver breaks down identically to the one in Equation 18 with the exception that we are now solving for the pseudorange *difference* instead of the pseudorange outright.

It is important to note that although many observation errors have been accounted for, all of the receiver-specific error sources (such as receiver clock bias, receiver phase offset, etc.) are still present in the signal. Unfortunately, some of these errors are more difficult to model and have a much larger impact on the final result than satellite-specific errors. Additionally, there is still an ambiguity term in the single-differenced carrier range model, which means that, once again, it is not useful in directly solving for receiver location.

Double-Differencing Model

Double-differencing — also known as "between-satellites differencing" — is a direct extension of the single-differenced model described in the previous sub-section. Quite literally, it consists of taking the difference between two single-differences [56]. Mathematically, this equates to finding the difference between two receivers' observations to a single satellite, repeating this for a second satellite, and then taking the difference between these two results. This is an extremely useful error-minimizing operation and is actually considered to produce the strongest solution when used in existing localization techniques.

Recall that the full equations for single-differenced pseudorange and carrier range are:

$$\begin{aligned} \Delta P^s_{jk} &= \Delta\rho^s_{jk} + c\Delta\tau_{jk} + \Delta\epsilon^s_{P,jk} \\ \Delta L^s_{jk} &= \Delta\rho^s_{jk} + c\Delta\tau_{jk} + \lambda\Delta B^s_{jk} + \Delta\epsilon^s_{L,jk} \\ &= \Delta\rho^s_{jk} + c\Delta\tau_{jk} + \lambda(\Delta\phi_{jk}(t_0) - \Delta N^s_{jk}) + \Delta\epsilon^s_{L,jk} \end{aligned} \tag{20}$$

Note that the ambiguity term has been expanded in the single-differenced model for carrier range. This is to illustrate that the ambiguity includes a fractional portion that is receiver-specific, just like the receiver clock bias. By double-differencing the observations to two different satellites, m and n, as described above, we can not only eliminate all error sources originating from the satellites, but all receiver-side errors as well [24]:

$$\begin{aligned}
\Delta\nabla P_{jk}^{mn} &= \Delta P_{jk}^{n} - \Delta P_{jk}^{m} \\
&= \Delta\nabla\rho_{jk}^{mn} + \Delta\nabla\epsilon_{P,jk}^{mn} \\
\\
\Delta\nabla L_{jk}^{mn} &= \Delta L_{jk}^{n} - \Delta L_{jk}^{m} \\
&= \Delta\nabla\rho_{jk}^{mn} + \lambda\Delta\nabla N_{jk}^{mn} + \Delta\nabla\epsilon_{L,jk}^{mn}
\end{aligned} \tag{21}$$

In these equations, notice that the only remaining error sources are the unmodeled effects including receiver noise, multipath, and antenna phase center offset which are usually ignored. Even more importantly, note that only the integer portion of the ambiguity term remains. This is an extremely useful property in that it adds an integer constraint to the unknown observation ambiguities, re-framing the problem to include so-called *integer ambiguities* that are unknown but constant through time.

These equations are used as the basis for many relative positioning algorithms. It is important to note that with an arbitrary number of satellites and receivers, there are many more double-differenced observation possibilities than there are raw satellite observations. Since it is impossible to create more information than we started with, it can be seen that a large number of the possibilities are actually linear combinations of one another [5]. As such, they provide no useful information and should be excluded from processing.

In order to ensure that we only include observations which are linearly independent, we introduce the concepts of a *reference receiver* and a *reference satellite*. Quite simply, these reference nodes form the basis for all double-differencing operations for a single epoch. By differencing all satellite observations from a single reference satellite and all receiver observations from a single reference receiver, we ensure that the resulting double-differenced set contains only linearly independent observations [5]. Of course, the best results will be achieved if we use references with the highest quality data, since any unmodeled errors in the reference observations will be included in all double-differenced observations, resulting in a potential bias [5]. By choosing the reference node to be the least likely to include significant error, we ensure that random errors in the non-reference observations will only affect one double-differenced observation apiece.

The best way to choose a reference receiver is to pick the node that is closest to the center of a set of nodes being localized, with the fullest, clearest view of the sky possible to increase satellite visibility and minimize multipath. Likewise, the best choice of reference satellite is the satellite with the highest elevation in the sky. Not only is this satellite the most likely to be visible to all receivers, but it is also guaranteed to experience the least amount of multipath and the least amount of atmospheric delay since it has the most direct line of sight to all receivers and the least amount of atmosphere to traverse [5]. One key thing to keep in mind when choosing a reference satellite, however, is that the satellite constellation is constantly moving, meaning that over time, a better reference satellite may come into view or the current reference satellite may lose its line of sight. When this occurs, any currently fixed integer ambiguities are no longer valid and must be transformed for the new reference satellite. Luckily, this is a fairly trivial operation solved by many existing algorithms.

Triple-Differencing Model

The final differencing model, *triple-differencing*, is, again, a direct extension of the double-differencing techniques just described. It is a temporal continuation in which two double-differences from the same two satellites and receivers but two *different* measurement epochs, usually consecutive, are subtracted [56]. This is almost never performed on pseudorange observables but is mainly a carrier range operation. Its purpose is to completely remove the ambiguity term from the carrier range model. You can see that the difference between two consecutive double-differenced observations will result in:

$$\begin{aligned} \Delta\nabla\Delta L_{jk}^{mn} &= \Delta\nabla L_{jk}^{mn}(t) - \Delta\nabla L_{jk}^{mn}(t-1) \\ &= \Delta\nabla\rho_{jk}^{mn}(t) - \Delta\nabla\rho_{jk}^{mn}(t-1) + \Delta\nabla\epsilon_{L,jk}^{mn}(t) - \Delta\nabla\epsilon_{L,jk}^{mn}(t-1) \\ &= \Delta\nabla\Delta\rho_{jk}^{mn} + \Delta\nabla\Delta\epsilon_{L,jk}^{mn} \end{aligned} \tag{22}$$

This equation includes no error sources, ambiguities, or unknowns other than the change in double-difference and the change in unmodeled errors (receiver noise and multipath) over time. As such, it provides a direct way to solve for the baseline between two receivers using carrier ranges alone.

There are three reasons why this type of differencing is not used exclusively in relative localization. The first and most obvious is because the model, by nature, requires that the two receivers remain completely static for as long as the triple-differencing is carried out. Since only one baseline is being estimated over multiple epochs, the baseline cannot change. As such, it is not suitable as a long-term solution for any sort of mobile receiver deployment.

The second reason is because some of the "unmodeled errors" may actually be increased by this operation. Note that in both the single- and double-differenced models, the solutions are based on observations from a single epoch; thus, unmodeled sources of time-variant error (such as multipath) do not introduce error correlations into the estimated solutions, but rather appear more like noise in the models [7]. In the triple-differenced model, however, we are measuring changes in the observables based on position over time. Therefore, although location-specific errors (such as atmospheric delay) and clock-specific errors (such as clock bias) cancel out, errors based on both location **and** time do not. This is because multipath is not identical (or even similar) for

- Two different receivers,
- The same receiver and two different satellites, or
- The same receiver-satellite pair at two different times [47].

In other words, $\mathbb{E}[(\boldsymbol{\Delta\nabla\Delta\epsilon}_{jk}^{mn}) \cdot (\boldsymbol{\Delta\nabla\Delta\epsilon}_{jk}^{mn})^T] \neq 0$.

The final and most important reason this model cannot be used alone is because by carrying out triple-differencing, we are measuring the change in double-differences and, thus, the change in relative satellite-receiver geometry over time. In other words, localization is now based on a change in geometry instead of the actual geometry itself. Since these changes happen quite slowly relative to the measurement interval, we lose most of the geometric localization information [50], and a weaker geometry results in less accurate position estimates; therefore, this situation is not ideal for baseline determination.

Instead, triple differencing is usually used for either a very course-grained initial baseline estimate or for detecting time-based errors, such as cycle slips. We have shown that, assuming the integer ambiguities in the carrier range model are constant over time, they completely cancel out in the triple-differencing operation. Therefore, by keeping track of the triple-differences over time, any changes should occur slowly and linearly. If a cycle slip occurs, the integer ambiguity for that satellite will lose consistency, causing a dicontinuity; therefore, a "spike" will be present in the triple-differenced output. In such a way, these types of observations are quite useful for conditioning data to make sure that no inexplicable errors or cycle slips have occurred before passing the data to a more accurate and robust localization method.

Relative Positioning Techniques

Using any number of techniques, the previously described differencing models can be utilized to determine the baseline between two or more receivers or to perform a type of localization known as relative positioning. As the name indicates, relative positioning consists of localizing one or more GPS nodes relative to either a reference node or to one another. It should be noted that relative positioning does not preclude results in an absolute geodetic coordinate space; on the contrary, most relative positioning techniques are simply a means by which to provide more accurate absolute receiver coordinates. This is usually achieved via prior knowledge of one or more node locations in an ECEF frame or, at the very least, the presence of at least one static node. Localization in the truly "relative" sense, with a network of roving receivers where none of the node positions are known *a priori*, is a much more recent development and describes the localization paradigm used by the new method in this dissertation.

Differential GPS

The first set of techniques developed to increase the precision of relative positioning using GPS became known collectively as "Differential GPS," or DGPS for short. In most cases, an algorithm is considered to be DGPS-based if it produces a baseline vector between two receivers in a situation where one "reference node" remains static at a location with previously determined coordinates and a second node roams freely around the reference [52, 16]. Relative DGPS techniques do not typically produce continuous position outputs, but rather require some amount of data aggregation and processing delay to achieve a solution. For *stop and go* techniques, in which the roving receiver must remain stationary at certain measurement points before moving on, these delays can range from 5-30 seconds. For *continuous* methods, the delay is only 0.5-5 seconds [16]. This differs from Real-Time Kinematic (RTK) positioning in which relative localization is carried out using **carrier range** measurements, with position outputs being produced in real-time [2].

DGPS is currently the status quo for high-precision surveying techniques with many commercial companies producing DGPS-based solutions [52, 9] — not to mention a national DGPS system currently being implemented in the United States [27]. In most cases, a single reference node – sometimes called a *beacon* – will be erected in a pre-surveyed location with a clear view of the sky. The beacon will begin receiving satellite ranging signals and then retransmitting them to nearby receivers. Any number of mobile, roving nodes can be dispatched around the beacon to receive their own GPS ranging signals along with the ranging signals belonging to the beacon. In such a way, every node has two full sets of ranging data and a known reference position at every epoch. Using this information in a relative context, many of the large sources of error in the satellite ranging signals can be minimized or eliminated to produce a much more accurate baseline than would be possible using single point positioning techniques. Also, since the location of the reference beacon is precisely known, these baselines translate directly into absolute coordinates which makes DGPS an extremely attractive method for surveying.

There are, however, several caveats to using this type of methodology. The first and most obvious is the necessity of having a reference beacon with a precisely known location. It requires some amount of pre-planning and set-up time to find an acceptable location for such a beacon, as well as to allow the beacon to localize its own point position with a high enough accuracy that it can be used as a reference for the roving nodes. This makes it a bad choice for on-the-fly type applications which cannot tolerate long set-up and calibration times [16]. The second caveat is that data from multiple epochs are usually required to achieve the level of precision desired using these methods. As such, the roving receivers must remain stationary at each measurement point for a length of time in order to collect the appropriate amount of data for the specified method. Due to the stop-and-go nature of these types of methods, they are more suitable for applications that use GPS localization primarily as a surveying tool, such as feature mapping or boundary determination.

Real-Time Kinematic (RTK) Positioning

A recent extension to standard DGPS methods which works well in highly dynamic and time-sensitive applications is Real-Time Kinematic (RTK) Positioning. This type of positioning represents a group of algorithms and solutions that use carrier range measurements from two (or more) receivers to provide real-time localization updates. While RTK can be viewed as a subset of DGPS, it is usually considered separately, with the main difference being that RTK algorithms are **necessarily** carrier-range based and always provide instantaneous and continuous receiver coordinates, whereas DGPS algorithms may only use pseudoranges and require post-processing, which precludes them from being used in strictly real-time scenarios. Additionally, the primary observable in DGPS methods is considered to be the pseudorange, because DGPS is concerned with minimizing correlated errors such that a more accurate *absolute* position can be estimated, and the primary observable in RTK is considered to be the carrier range with an emphasis on providing more accurate *relative* localization [31, 20, 16].

Since RTK primarily uses carrier ranges, we are once again faced with the problem that the carrier range model includes an unknown ambiguity term. In order to achieve the centimeter-level accuracy possible with this type of configuration, roving receivers need to be able to produce solutions that are ambiguity-fixed and double-differenced. As such, the main area of research in RTK positioning revolves around solving for the integer ambiguities in the double-differenced carrier range model. Since it is possible that the roving receivers may be in constant motion, this poses a particularly difficult problem when examing most of the “standard” ambiguity fixing algorithms in use today. In fact, it is such a problem that there still exists no real solution able to provide accuracies on par with those achievable if the roving receivers remain stationary during the ambiguity fixing process.

In the majority of cases, RTK algorithms will require an initial, short calibration phase during which the ambiguities can be unequivocally resolved. After that, the receivers are free to move anywhere they like, and assuming that consistent locks on at least three satellites are able to be retained throughout the localization process, the accuracy is unlikely to degrade [16, 20]. If stationary calibration cannot be achieved, some algorithms make use of both the pseudorange and carrier range observables to resolve the ambiguities, leading to a time period of increasing accuracy as the solution algorithm transitions from a strictly ambiguity-float solution to an ambiguity-fixed solution.

It should be noted that the downside to DGPS and RTK algorithms alike is that they both presuppose the presence of a stationary reference node. The case where the locations of **both** nodes is unknown or, even worse, when the positions are unknown and both receivers may be moving, represents a realm of localization that has not yet been satisfactorily solved. The best approach in these cases at the time of writing is a so-called *moving baseline* method in which two receivers are assumed to be in motion, and an RTK-like algorithm attempts to estimate the baseline vector between them at every time epoch [30, 28, 21]. Most current algorithms which fall under this heading are proprietary in nature, and there is yet to be seen an open, standard method for dealing with the uncertainties in this problem. The technique offered by this dissertation aims to fill this gap in relative localization techniques.

CHAPTER IV

RELATIVE TRACKING AND LOCALIZATION ALGORITHMS

As introduced in Chapter I, the goal of this research is to provide a complete centimeter-scale relative localization system using low-cost, single frequency GPS receivers alone. Additionally, we have discussed many of the shortcomings of existing techniques, as well as the needs of certain applications which are not being met by the current state of the art. As such, the following list has been devised, describing the essential requirements that a complete relative localization system should have to garner the largest amount of utility from the widest range of applications:

- Centimeter-precision accuracy for a scalable network of nodes,
- No *stationary* or *explicit* calibration or data collection phase, and
- No explicit reference station or hub.

A centimeter-scale level of precision should be accurate enough to enable a wide range of highly location-dependent new applications. Likewise, the lack of an initialization phase in which the network of receivers must remain stationary will be extremely useful in ad-hoc or impromptu localization setups, where there is either no time to spend on lengthy calibrations, or the nature of the application precludes the component receivers from remaining stationary. Finally, the lack of an explicit reference station will increase both robustness and accuracy, eliminating any single points of failure from the system and ensuring that any unmodeled errors that may be present in a single reference signal remain decorrelated from the signals in the rest of the network.

The approach we take to realizing a complete localization system without violating any of these requirements stems from three key observations:

1. It is impossible to consistently, robustly, and dynamically achieve the level of precision desired without some form of error mitigation and/or implicit calibration,
2. Calibration takes a long time because existing algorithms rely on the slowly changing geometry of satellites to introduce variety into the otherwise stationary observations, and
3. Receiver-based changes in geometry (motion) would greatly speed up the calibration process *if the magnitude and direction of these changes were known precisely.*

The last item listed is the key to making our technique work. It should be noted that there does exist a body of research aimed at increasing the accuracy of localization techniques by integrating them with motion-based sensors such as IMUs, accelerometers, or gyroscopes [60, 44, 62]; however, all of these sensing modalities suffer from rapid bias accumulation and are therefore only useful for increasing the short-term stability of GPS measurements.

It would be much more useful if there were a method that could provide highly accurate relative tracking results over an extended period of time. Not only would this allow for receiver motion during a calibration phase, but it would also provide the much-needed receiver-based change in network geometry required to more rapidly converge to the level of precision desired. Additionally, by knowing the relative 3D motions of the receivers through time, the number of unknowns in the system would not increase past the initial three X,Y,Z-coordinates of the baseline. In other words, the known relative receiver motions can be used to make data observations taken over multiple epochs mathematically equivalent to the stationary case.

Take, for example, a system of receiver-satellite range equations (R_r^s) over time, where each equation is a function of the initial unknown baseline coordinates ($\mathbf{b}(t_0)$) with respect to a "stationary" reference ($\mathbf{x}_{ref}$) at time t_0:

$$\begin{aligned}
R_r^s(t_0) &= f\begin{pmatrix} x_{ref} + b_x(t_0), \\ y_{ref} + b_y(t_0), \\ z_{ref} + b_z(t_0) \end{pmatrix} \\
R_r^s(t_1) &= f\begin{pmatrix} x_{ref} + b_x(t_0) + \Delta b_x(t_1), \\ y_{ref} + b_y(t_0) + \Delta b_y(t_1), \\ z_{ref} + b_z(t_0) + \Delta b_z(t_1) \end{pmatrix} \\
R_r^s(t_2) &= f\begin{pmatrix} x_{ref} + b_x(t_0) + \Delta b_x(t_1) + \Delta b_x(t_2), \\ y_{ref} + b_y(t_0) + \Delta b_y(t_1) + \Delta b_y(t_2), \\ z_{ref} + b_z(t_0) + \Delta b_z(t_1) + \Delta b_z(t_2) \end{pmatrix}
\end{aligned} \tag{23}$$

If the *relative* motions of the receiver to the reference are known (the $\Delta\mathbf{b}$ values), the only unknowns in the equations above are the initial baseline coordinates, ($\mathbf{b}(t_0)$). Additionally, it does not even matter if the reference receiver is stationary, since we are only concerned with the relative baseline between the two nodes. As such, it is clear that including range observations from a number of consecutive epochs increases the amount (and quality) of data available to solve for the initial baseline, or in other words, to calibrate. Quality is said to be increased because data from different epochs correspond to different satellite constellation geometries, and the more geometries that are incorporated, the smaller the region of feasibility for the location of a receiver.

Using this knowledge, we created a technique which first tracks the positions of any node in a network *relative to the receiver performing the localization*, and then uses these tracking results in a baseline determination algorithm, with the end result that each receiver using the localization service has its own highly accurate internal mapping of all of the surrounding nodes taking part in the localization. The rest of the chapter describes this technique in detail.

Error Mitigation

In creation of our GPS tracking solution, it was noted that existing measurement error models tend to characterize standalone-only errors that degrade position accuracy in a single-receiver, absolute-coordinate sense. Since our research focuses on *relative* positioning, it quickly became clear that several sources of error exist specific to multi-receiver localization, and these errors could not be overlooked if we hoped to achieve the centimeter-scale level of accuracy desired.

Dual-Receiver Error Sources

In order to discuss the additional sources of error arising from measurement combinations across multiple receivers and times, we must first categorize the existing known errors into two categories based on their effect on a given satellite observation. These two categories consist of signal propagation errors and time bias errors. Signal propagation errors directly affect the actual transit time of a radio signal, meaning that they introduce a measurable delay in signal reception from the ideal time taken under vacuum conditions. Time bias errors, on the other hand, have no impact on the radio signals themselves, but rather affect **when** we measure them, which is usually at odds with our perception of the measurement times. As such, time bias errors may appear in a measurement observation, but they are not directly estimable since they are not due to physical signal delays.

The reason this is important in terms of GPS is due to the way in which the satellite observables (and corresponding corrections) are formed. Recall that in formation of the pseudorange observable, the exact time of transmission of the received signal is known; however, our inaccurate local clocks are unable to determine precisely when the signal was received. As such, this time bias error makes it impossible to calculate the exact propagation time of the ranging signal from satellite to receiver. In "Section: *Relativistic Effects*" in Chapter III, we discussed how rotation of the Earth during this propagation time could introduce error into the resulting satellite observations. While the corresponding corrections are straightforward and sufficient for a single receiver, there exist several subtleties regarding dual-receiver localization that do not seem to appear in literature at this time of writing, stemming from the fact that the Earth rotates in the presence of both physical signal delays and time biases from GPS time:

1. Earth rotation corrections are calculated from pseudorange observations that include the two types of errors previously mentioned: signal propagation errors and time bias errors. Due to the *bias* errors, we cannot know the exact time of reception of the satellite signals, which means we do not know the real propagation time of the signal, and hence cannot accurately estimate how far the Earth has rotated during this time.
2. Without knowing precisely how far the Earth has rotated, transformation of the satellite positions into the correct ECEF frame will not be exact.
3. In the two-receiver case, if measurements are made at two slightly different times, the satellite positions at the time of each signal transmission will have *actually* been different.
4. Likewise, the signal propagation times will have also been different, and thus, the Earth will have experienced differing amounts of rotation between the two measurement intervals.
5. Since the actual satellite positions (from two different transmit times) as well as the apparent satellite positions (from incorrect Earth rotation corrections) may both be different, a combination of these observations from two receivers would result in an error model that includes second-order effects dependent on satellite orbital offset, relativistic effects, and receiver clock biases.

The result is a model containing many cross-dependencies that are difficult to mitigate mathematically. Specifically, two receivers making observations at apparently the same time (according to their local clocks) may calculate satellite positions up to ~9 m apart from one another with analogous errors in the measurement observables. In reality, these error sources typically amount to 1-3 meters of error per dual-receiver satellite observation.

Ideally, we would like to take the satellite geometry "snapshots" according to both receivers and extrapolate the measurement data to correspond to what the observations and satellite positions *should* have been if taken exactly at the GPS epoch in the absence of any time bias errors. Unfortunately, this requires a slight increase in computational complexity, and extrapolation of measurement data to a specific *reception* time is not a straightforward mathematical operation, as shown in the next subsection.

Time Bias Data Extrapolation

The optimal way to overcome the error subtleties arising from the two-receiver case is to extrapolate the data from two or more receivers to the exact common point in time when the receivers *should* have made their measurements, namely the GPS epoch. In order to begin the extrapolation procedure, it is necessary to first determine the clock bias of each receiver so that we know how far forward or backward to extrapolate the data. This step can be as simple as solving for the standalone 3D position and bias of each receiver using a simple least-squares optimization routine as described in [24].

Once the receiver clock bias is known, the obvious next step would be to simply rotate each of the receivers' coordinate frames and data sets to coincide with the nominal GPS epoch. This will not result in a satisfactory solution however, especially in the multiple receiver case, because although the receivers themselves can be rotated into a cohesive frame, the satellite positions do not "rotate" with the Earth, but rather follow their own orbits. This means that errors due to the discrepancies between satellite positions at actually different transmit times will still be present. Additionally, the amount of Earth rotation experienced during the measured signal propagation interval cannot be assumed to be identical to the amount of rotation that would have been experienced at the extrapolated time.

In order to correctly extrapolate the data, we need to determine the change in signal propagation time (and by extension, the propagation range, R_r^s) if the measurement had been made correctly at the GPS epoch. Fortunately, the instantaneous change in range can be inferred quite accurately using the Doppler shift observable (necessarily computed by all GPS chips), which directly indicates the line-of-sight *range rate* between a satellite and receiver at any given time:

$$\Delta R_r^s(t) = -f_D^s(t) * \lambda_{L1} * \tau_{bias} \tag{24}$$

where $f_D^s(t)$ is the instantaneous Doppler frequency to satellite s reported by the receiver at time t, λ_{L1} is the vacuum wavelength of the carrier signal, and τ_{bias} is the receiver clock bias from GPS system time. Assuming a Doppler shift accuracy of ± 1 Hz, this change in range is guaranteed to be accurate to better than 191 μm.

The pseudorange and carrier range observations to any given satellite can then be extrapolated to the exact GPS epoch by simply adding the $\Delta R_r^s(t)$ correction term to each observable. Likewise, it is trivial to track a receiver's clock bias through time to determine its rate of change, also known as the *clock drift.* Using this drift value, we can extrapolate the local clock bias to the GPS epoch via:

$$\tau'_{bias} = \tau_{bias} * (1 + \tau_{drift}) \tag{25}$$

This step is necessary because we cannot assume the local clock bias at the exact GPS epoch to have remained constant since the incorrect measurement time, and even microseconds of time difference can result in satellite positional errors on the order of decimeters. Put more simply, τ'_{bias} must be used to determine signal reception time *according to the receiver's local clock* if the observations had been made correctly at the GPS epoch to avoid introducing errors in estimation of the extrapolated satellite positions.

Since we now know both the correct receiver clock bias and what the measurement observables *should* have been, we can calculate the time when a signal received at the exact GPS epoch would have been transmitted, and thereby also the corresponding correct satellite positions, $\mathbf{X}^s$, and Earth rotation corrections corresponding to the extrapolated epoch. The following procedure explains this concept in more detail:

Algorithm 1 GPS Extrapolation Procedure

```
1: Determine the receiver clock bias, $\tau_{bias}$, using a simple least-squares point positioning
   solution (see "Section: *Absolute Localization*" in Chapter III). This bias is equal to the
   time difference between the desired measurement time and the actual measurement time
2: Determine an updated receiver clock bias as if the measurement had been made at the
   correct GPS epoch according to $\tau'_{bias} = \tau_{bias} * (1 + \tau_{drift})$
3: Using the updated receiver clock bias, determine what the receive time of the correctly
   received signal would have been *according to the local receiver clock*: $T'_{rx} = t_{epoch} + \tau'_{bias}$
4: Calculate the change in satellite range over the bias time using $\Delta R^s_r(t) = -f^s_D \lambda_{L1} \tau_{bias}$
5: Update the measurement observables by the computed $\Delta R^s_r(t)$ value
6: Calculate the extrapolated signal transmit time using the updated pseudorange observ-
   able, $P^s_r(T'_{rx})$, according to: $t'_{tx} = T'_{rx} - \frac{P^s_r(T'_{rx})}{c}$. (See Figure 2 on Page 13 for an expla-
   nation of why this works)
7: Calculate the satellite position at the new transmit time, $t'_{tx}$
8: Correct the satellite position for the Sagnac Effect using the extrapolated (i.e. *actual*)
   receive epoch, $t_{epoch}$, and the extrapolated transmit time (see [24])
```

The effectiveness of the aforementioned extrapolation procedure is difficult to quantify since it depends not only on our ability to accurately estimate a receiver's clock bias from GPS time, but also on the specific satellite for which the data is being extrapolated. It should be obvious that a satellite moving laterally with respect to a receiver will not require as much correction as a satellite moving primarily radially. This is because lateral motion does not constitute a change in range, whereas radial motion does. Therefore, more positional range error due to receiver clock bias will be present in measurements from satellites with a low elevation in the sky than higher elevation satellites (with satellites at the zenith experiencing almost entirely lateral motion).

Another reason the effectiveness is difficult to quantify is that all observations include a very large error term consisting of the receiver clock bias times the speed of light. As such, we cannot simply compare the extrapolated data observations from two different receivers, since they may still contain many hundreds of meters of clock-based error; additionally, if we compare the extrapolated **and** clock bias-corrected data from two receivers, we are not simply measuring the effectiveness of data extrapolation on the satellite range error, we are also measuring the correctness of our individual estimates of each receiver's clock bias. This borders on irrelevant since first-order receiver clock bias effects are handled separately, as will be seen later in this chapter, and it is only the second-order effects with which we are concerned.

We were able to determine an alternative method of performing quantitative analysis on the effectiveness of this extrapolation procedure via direct comparison of the estimated *satellite positions* from two receivers when the signals corresponding to the extrapolated data would have been transmitted by each satellite. This works by placing two (or more) receivers as close as possible to the exact same position in space and making measurements over a significant period of time. Since the receivers are co-located, all atmospheric and satellite-based errors should be identical, as should the ranges to each of the satellites at a given point in time. Since the ranges are equal, the transmit times of the received signals should also be equal which means the positions of the satellites should be exactly identical for all participating receivers.

As such, we can find the extrapolation error on the measurement observables between receiver a and receiver b's viewpoints of the position of a single satellite, s, according to:

$$\epsilon = \sqrt{(X_b^s - x)^2 + (Y_b^s - y)^2 + (Z_b^s - z)^2} - \sqrt{(X_a^s - x)^2 + (Y_a^s - y)^2 + (Z_a^s - z)^2} \qquad (26)$$

where (x, y, z) are the co-located receiver coordinates, and (X_r^s, Y_r^s, Z_r^s) are the satellite coordinates for satellite s according to receiver r. Based on this analysis, the error results shown in Table 1 were found.

Satellite #	**Mean Difference without Extrapolation**	**Mean Difference with Extrapolation**
6	1.7 m	< 1 mm
14	2.0 m	< 1 mm
15	1.4 m	< 1 mm
21	1.9 m	< 1 mm
22	2.1 m	< 1 mm
27	1.8 m	< 1 mm

Table 1: Average satellite range differences as seen from three co-located receivers at the same GPS epoch over the course of 30 minutes

These results show that Algorithm 1 can be used to extrapolate satellite observations to the GPS epoch with μm accuracy.

Temporal Double-Differencing Model

Now that we have devised a way to ensure that data sets from two different receivers correspond to the same satellite geometries and measurement intervals, we can combine them in a variety of different dual-receiver observation models (including those mentioned in "Section: *Observation Models and Positioning Techniques*" in Chapter III), with the assurance that second-order errors of errors will not be propagated into our system. We mentioned earlier that the classical double-differencing model is generally regarded as providing the strongest relative baseline solution from a geometric point of view; however, it has some limitations. These include:

- Correlated errors present in every data observation due to the use of a reference satellite that is unlikely to be completely error-free,
- The necessity of solving for a set of ambiguity terms that are dependent on satellite, receiver, and reference satellite, which is a complex and relatively slow process,
- Problems whenever the tracking lock to a reference satellite is lost, since any integer ambiguities that have been solved for are only valid with respect to the given reference satellite, requiring either large data transformations or even re-initialization of the localization procedure, and
- The requirement that the receivers remain stationary while the ambiguity fixing process is being carried out (if using carrier ranges), which can take a significant amount of time.

Keeping these shortcomings in mind, we introduce an additional observation model called the "temporal double-differencing model" which is a time-based extension of the single-differencing model. Recall the single-differenced observation equations (here marked to indicate their dependence on time):

$$\begin{aligned}\Delta P^s_{jk}(t) &= \Delta\rho^s_{jk}(t) + c\Delta\tau_{jk}(t) + \Delta\epsilon^s_{P,jk}(t)\\ \Delta L^s_{jk}(t) &= \Delta\rho^s_{jk}(t) + c\Delta\tau_{jk}(t) + \lambda\Delta B^s_{jk} + \Delta\epsilon^s_{L,jk}(t)\end{aligned} \tag{27}$$

In these equations, the correlated errors due to a reference satellite are not present, nor are the mathematical problems arising from the loss of such a reference satellite. In order to maintain these benefits, we avoid the introduction of any sort of reference satellite by differencing not through space, but through time (where ∇ represents a temporal difference instead of a spacial one):

$$\begin{aligned}\nabla\Delta P^s_{jk}(\nabla t) &= \nabla\Delta\rho^s_{jk}(\nabla t) + c\nabla\Delta\tau_{jk}(\nabla t) + \nabla\Delta\epsilon^s_{P,jk}(\nabla t)\\ \nabla\Delta L^s_{jk}(\nabla t) &= \nabla\Delta\rho^s_{jk}(\nabla t) + c\nabla\Delta\tau_{jk}(\nabla t) + \nabla\Delta\epsilon^s_{L,jk}(\nabla t)\end{aligned} \tag{28}$$

We see here that the carrier range model no longer includes an ambiguity term since it was constant through time. Additionally, the change in clock bias over time, $c\nabla\tau_r(\nabla t)$, is known more commonly as the *clock drift*, typically denoted $\dot{\tau}_r$, and it is a much more stable value than the clock bias itself. As such, this temporal double-differencing model includes only the difference between the *change* in ranges of two receivers to one satellite over the course of one epoch and the difference between the two receivers' clock drifts, along with some amount of observation error. Substituting for the clock drift term and dropping the ∇t-nomenclature for simplicity's sake, this results in:

$$\begin{aligned}\nabla\Delta P^s_{jk} &= \nabla\Delta\rho^s_{jk} + c\Delta\dot{\tau}_{jk} + \nabla\Delta\epsilon^s_{P,jk}\\ \nabla\Delta L^s_{jk} &= \nabla\Delta\rho^s_{jk} + c\Delta\dot{\tau}_{jk} + \nabla\Delta\epsilon^s_{L,jk}\end{aligned} \tag{29}$$

The similarity between these equations shows that we now have a model in which the two distinct satellite observation types can be used to mathematically represent the exact same thing. This can be useful for data integrity verification (as in the triple-differencing model) or simply for replacing the observation values in pre-existing pseudorange-based techniques with the more accurate carrier range values, since they are treated identically in this model. The carrier range equation shown above is unique in that it uses highly accurate carrier phase observations to produce **unambiguous** estimates of the change in relative ranges between a satellite and two receivers through time without requiring any sort of reference satellite or node.

Relative GPS Tracking Algorithm

Recall in the beginning of this chapter that we introduced the concept of a high-accuracy relative node tracking algorithm. It was stated that such an algorithm would prove beneficial in enabling dynamically-moving receivers to be localized as if they were stationary over a significant period of time. When the new temporal double-differencing observation model from the previous section is examined more closely, we see that its geometric significance can be stated in an alternate way that may prove more useful in the creation of such a tracking algorithm.

As a reiteration, we note that by subtracting one receiver's carrier phase observation to satellite s from a second receiver's similar observation (i.e. single-differencing), the result is the range difference between the two receivers to that satellite. It can be shown that this concept is approximately equal to projecting the relative baseline between the two receivers onto the line-of-sight unit direction vector from receiver to satellite, as shown in Figure 9.

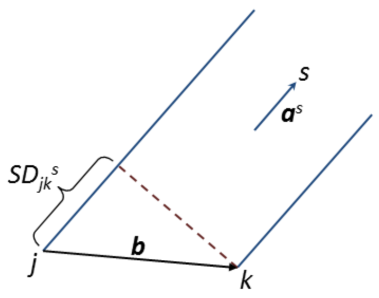


Figure 9: Geometric interpretation of the single-differencing operation

Note that the GPS satellites are such a great distance from the surface of the Earth that the unit direction vectors can be assumed to be identical for both receivers with little effect on the result as long as the two receivers are located in the same geographic region. This assumption has been tested and verified according to [51] with results as indicated in Table 2.

Baseline Length	**Range Error**
2 m	100 nm
100 m	247 μm
1,000 m	2.47 cm
10,000 m	2.48 m

Table 2: Interferometric range errors for differing baseline lengths under the assumption of identical satellite unit direction vectors

For baselines shorter than 1 km, this assumption introduces very little error into the system. In fact, for baselines up to 10,000 km, this data would suggest an increase in error along with baseline distance according to the following power equation:

$$\epsilon = 2.476\times10^{-8}x^2 \tag{30}$$

Thus, the error due to the assumption that satellite unit direction vectors are identical for similarly located receivers can be kept on millimeter scale by limiting it to cases involving baselines less than approximately 635 m. Now that we have seen that the single-differencing operation can be approximated as a function of the baseline between two receivers, we show that a similar statement can be made about the temporal double-differencing model.

If we perform another single-differencing operation at the next time epoch, $t+1$, and then subtract the two results, we will have created a temporal double-differenced value, with model corresponding to Equation 29, repeated here for clarity (and ignoring any receiver measurement noise):

$$\nabla\Delta L^s_{jk} = \nabla\Delta\rho^s_{jk} + c\nabla\Delta\tau_{jk} \tag{31}$$

Since the $\nabla\Delta\rho^s_{jk}$ term indicates the relative change in satellite ranges over the course of one epoch, the model can be expanded as follows:

$$\nabla\Delta L^s_{jk}(t+1) = R^s_k(t+1) - R^s_j(t+1) - R^s_k(t) + R^s_j(t) + c\nabla\Delta\tau_{jk}(t,t+1) \tag{32}$$

For any arbitrary change in time, this is the same as:

$$\nabla\Delta L^s_{jk}(t+\Delta t) = R^s_k(t+\Delta t) - R^s_j(t+\Delta t) - R^s_k(t) + R^s_j(t) + c\nabla\Delta\tau_{jk}(\Delta t) \tag{33}$$

When the range is written recursively through time, we see that:

$$\begin{aligned}
&\nabla\Delta L^s_{jk}(\Delta t)\\
&\quad= (R^s_k(t) + \Delta R^s_k(\Delta t)) - (R^s_j(t) + \Delta R^s_j(\Delta t)) - R^s_k(t) + R^s_j(t) + c\nabla\Delta\tau_{jk}(\Delta t)\\
&\quad= \Delta R^s_k(\Delta t) - \Delta R^s_j(\Delta t) + c\nabla\Delta\tau_{jk}(\Delta t)\\
&\quad= \Delta R^s_k(\Delta t) - \Delta R^s_j(\Delta t) + c\Delta\dot{\tau}_{jk}(\Delta t)
\end{aligned} \tag{34}$$

Similar to our assumption that the unit direction vectors to a single satellite are identical for receivers in a geographically similar region, we can make the argument that the unit direction vectors to a single satellite are also identical over the course of one time epoch. We analyze the error introduced by this assumption numerically for the worst case by noting that a satellite completes a full orbit every 11 hours and 58 minutes. In other words, the largest angle it could trace out in the sky over one second would occur when it passes through the zenith directly overhead. At the standard orbital period, 1 second equates to $0.008357°$. This means that the most influence the change in unit direction could have would be $1.0-(1.0\times\cos 0.008357) = 10.636{\times}10^{-9}$ times the baseline. This is such a small number that it can be disregarded for short baselines, but it is very important to note that the error due to this effect grows with increasing baseline lengths. Anything over 1,000 km will start to introduce errors on the centimeter level; therefore, as was concluded in the single-differencing error analysis, this methodology should only be used for so-called *short baselines*.

Since we have shown that the unit direction vector does not change significantly over consecutive measurement epochs, Equation 34 can be re-written geometrically as the change in receiver-receiver baseline ($\Delta\mathbf{b}$) projected onto the unit direction vector from receiver to satellite ($\mathbf{a}^s_r$), where the Δt terms have been omitted for succinctness:

$$\nabla\Delta L^s_{jk} = \Delta\mathbf{b}\cdot\mathbf{a}^s_r + c\Delta\dot{\tau}_{jk} \tag{35}$$

It is clear from this result that neither of the initial positions of the receivers must be precisely known in order for the relative motions between them to be accurately tracked. Additionally, it is unnecessary to solve for the *absolute* motions of either of the receivers, since Equation 35 allows for a direct solution of the change in baseline based on our general knowledge of the unit direction vector to satellite s.

In order to solve a system of tracking equations (in the form of either Equation 34 or 35), a simple least-squares optimization routine can be set up to estimate the tracked 3D coordinates through time along with the clock drift difference. As long as consistent satellite locks are maintained with at least four satellites, the relative positions of any node after initialization can be determined via simple dead reckoning (i.e. adding the current tracking result to the last relative position estimate).

In addition to allowing for the tracking of relative motions without requiring precise *a priori* knowledge of any of the node locations, our tracking methodology has the added advantage that neither of the nodes must remain stationary during the tracking procedure due to the fact that:

- Absolute changes in position are irrelevant since only relative motion is present in a moving coordinate system,
- The satellite unit direction vectors are virtually identical for a node that has only moved an order of meters in one epoch.

Measurement Error Detection Using Clock Drift Estimation

The tracking methodology just described is able to produce highly accurate results in a low-multipath environment; however, multipath-rich environments tend to not only introduce additional unmodeled errors into one or more of the satellite observations, but also to increase the likelihood that very minute, undetected cycle slips may occur. In either of these cases, the errors themselves are so small that they are mathematically difficult to detect, but their effect on the results of the tracking algorithm can be quite large and accumulate quickly.

In order to provide a robust way to discard these erroneous measurements, we devise a technique whereby the clock drifts of the participating receivers are tracked in their own separate filters through time and then compared to the clock drift term produced by the relative tracking algorithm described above. This is reasonable and simple to do since the clock drifts experienced by the local oscillators tend to be quite stable over relatively long time spans. To begin, an empty list of satellites to ignore due to possibly erroneous measurements is created. The detection algorithm then begins as described in Algorithm 2.

Once no additional erroneous measurements are detected, the "good" satellite observations are used to produce a final tracking result which can then be used by the relative localization algorithm described in the next section. Note that the residual thresholds used to identify erroneous measurements in Algorithm 2 were determined empirically based on many test iterations using large data sets.

Algorithm 2 Erroneous Satellite Observation Detection

```
1: Compute the 3D tracking vector and clock drift term from Equation 34 or 35 using all
   valid, non-ignored satellite observations
2: Compare the computed clock drift term with the tracked clock drift estimate from the
   clock dynamics tracking filter
3: If the difference between the clock drifts (times the speed of light) is greater than 1.5 me-
   ters, OR the maximum error residual from the least-squares tracking solution is greater
   than $\frac{1}{5}th$ the wavelength of the carrier wave, a measurement is most likely erroneous.
   Recompute the 3D tracking vector by setting the clock drift term equal to its estimate
   from the dynamics tracking filter (i.e. remove a single degree of freedom), and add the
   satellite with the resulting maximum residual to an "ignore list," then return to Step 1
4: If this step is reached, no more satellite observations are determined to be in error
```

Relative GPS Localization Algorithm

The tracking results from the previous section provide exceptional accuracy when viewed from a dead-reckoning point of view; however, their real utility comes in allowing data from multiple epochs to be used in a single localization algorithm regardless of any motion between the receivers. For the theory in this section, we will make the assumption that the available tracking results are completely error-free, and we will investigate the real-world impact of the omission of any such errors in Chapter VI.

We form the basis for our relative localization algorithm from the following observations:

1. Accurate tracking results allow us to use localization techniques that would normally be reserved for stationary network topologies,

2. The standard double-differencing observation model provides the strongest geometric solution while including the minimum amount of unmodeled error possible; however, it requires resolution of the so-called *integer ambiguities*,

3. A solution relying solely on the resolution of ambiguities in the carrier phase model is inadvisable, since it will, by necessity, be susceptible to large errors due to undetected cycle slips and will mandate that a certain subset of satellites remain consistently visible through time, and

4. The correct relative position will be characterized by a set of ambiguity values that results in low error residuals when the *modeled* receiver-satellite ranges are calculated and compared to the *actual* satellite observations over a significant period of time.

Recall that a single satellite observation viewed from the paradigm of the double-differenced carrier phase model will include a GPS-reported satellite-receiver carrier phase and an unknown integer ambiguity term, as shown in Figure 10.

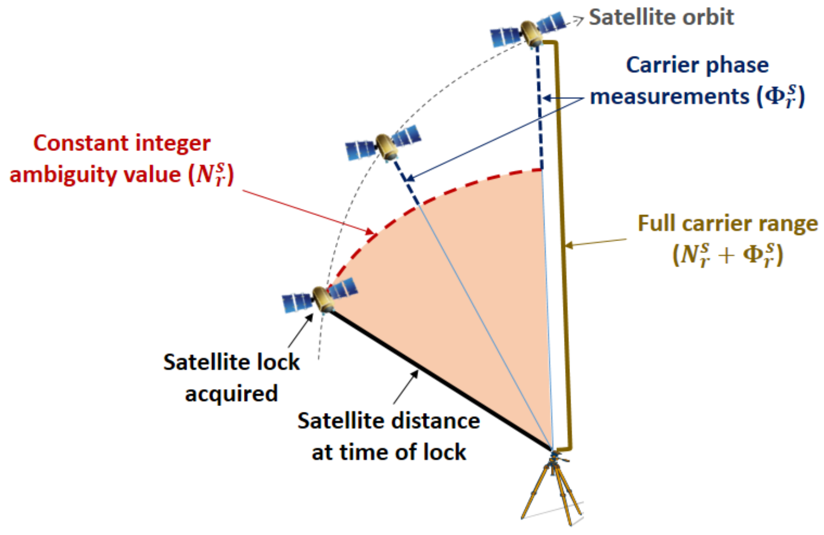

Satellite orbit
Carrier phase measurements (Φ_r^s)
Constant integer ambiguity value (N_r^s)
Full carrier range ($N_r^s + \Phi_r^s$)
Satellite lock acquired
Satellite distance at time of lock

Figure 10: Constituent components of the double-differenced carrier phase observation for a single satellite through time

Since the integer ambiguity is related to the number of whole carrier wave cycles that were present between the satellite and receiver at the time of initialization, we can think of the observation to each satellite as a reported range value plus an unknown bias which is **guaranteed** to be equal to some number of carrier cycle wavelengths (~19 cm), where the actual receiver position must lie on one of the grid lines representing the reported range plus the ambiguity bias, like so:

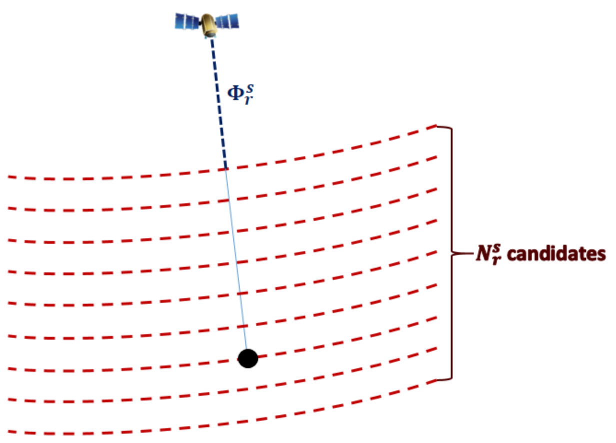
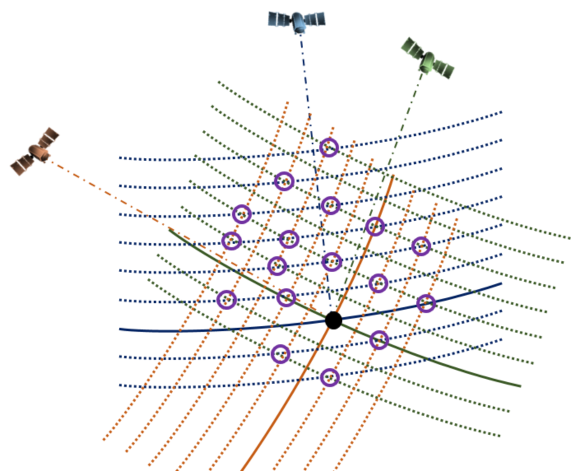


Figure 11: Potential integer ambiguity candidates and their corresponding grid lines

If we extend this model to a three-satellite configuration, as shown in Figure 12a, it is apparent that there are multiple intersecting points which could all be potential receiver positions since they are located on one of the gridlines from each of the satellites, and the relevant gridlines intersect one another very nearly perfectly. However, as the satellites move over time, the slopes of the gridlines will necessarily change such that a new set of possible locations becomes apparent (Figure 12b):


(a) Set of potential receiver locations at time t_0.

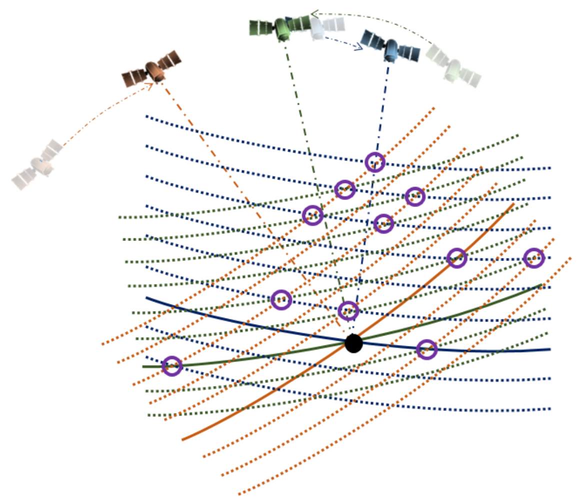

(b) Set of potential receiver locations at time t_n.

Figure 12: Change in candidate receiver locations through time

We can see in this figure that one and only one of the candidate locations has remained constant through the change in satellite geometry. This is obviously the correct receiver position, and as stated in our list of observations above, this can be verified mathematically by noting that the satellite-receiver ranges calculated using the incorrect ambiguity values (lines) from Figure 12a will no longer intersect with one another when recalculated after a change in satellite geometry. In other words, only the correct receiver position (with the correct ambiguity values) will have a continually low error residual when the computed satellite-receiver ranges are compared to the satellite observation values through time.

While this result shows the strength of the double-differenced observation model, we have already said that it can suffer severe drawbacks in that:

- Cycle slips are difficult to detect and will result in either very high error residuals or very wrong position estimates, and
- The double-differenced model includes use of a reference satellite that may go in and out of visibility, undermining the ability of the ambiguity values to be resolved since their values are completely dependent on the reference satellite being used.

As such, we decided to look for other ways of carrying out our localization procedure without requiring resolution of the integer ambiguities.

Ambiguity Function Method Overview

From literature, we found a mathematical concept introduced in 1981 by Counselman and Gourevitch called the "Ambiguity Function Method" (AFM) [10]. This method is unique in that it leverages the "integer-ness" of the ambiguity values in the double-differenced carrier phase model to determine a baseline position which minimizes the range errors in the participating satellites' observations. Recall that the error-free double-differenced carrier phase model looks like:

$$\nabla\Delta L_{jk}^{mn}(t) = \nabla\Delta\rho_{jk}^{mn}(t) + \lambda_{L1}\nabla\Delta N_{jk}^{mn} \tag{36}$$

By isolating the integer ambiguity term on the left-hand side of the equation, we get:

$$\nabla\Delta N_{jk}^{mn} = \frac{\nabla\Delta L_{jk}^{mn}(t) - \nabla\Delta\rho_{jk}^{mn}(t)}{\lambda_{L1}} \tag{37}$$

This shows that at the *correct* baseline coordinates, the entire right-hand side of the above equation will be a perfect integer (in the absence of any errors).

Mathematically, we can test a value for its integer-ness by converting the number to radians and taking its cosine:

$$\cos(2\pi \cdot \text{integer}) \equiv 1.0 \tag{38}$$

For numbers that are not perfect integers, the resulting value will drift further and further from 1.0, reaching a minimum of -1.0 at any integer ± 0.5, or in other words, at the least "integer-like" value possible. Extending this concept to the double-differenced carrier phase model, we see that:

$$\cos(2\pi \cdot \nabla\Delta N_{jk}^{mn}) = 1 \tag{39}$$

And by extension, at the correct 3D baseline coordinates:

$$\cos(2\pi \cdot \frac{\nabla\Delta L_{jk}^{mn}(t) - \nabla\Delta\rho_{jk}^{mn}(t)}{\lambda_{L1}}) = 1 \tag{40}$$

This is an interesting concept in that it allows us to search for a baseline solution in the *position domain*, instead of in the *measurement* or *ambiguity* domains, as is usually the case.

By summing the resulting cosine values for all possible satellite observations and dividing this result by the total number of observations, n, we can calculate an *Ambiguity Function*

Value (AFV) in the range from $-1.0 \leq \text{AFV} \leq 1.0$:

$$\text{AFV} = \frac{\sum_{a=1}^{n} \cos\left(2\pi \cdot \frac{\nabla\Delta L_{jk}^{ma}(t) - \nabla\Delta\rho_{jk}^{ma}(t)}{\lambda_{L1}}\right)}{n} \tag{41}$$

where an AFV closer to 1.0 represents a position in which all of the double-differenced ambiguity values would be very nearly integer.

Since we know that the ambiguity values **must** be integers, the application of this method to GPS is as simple as defining a 3D search space, calculating the AFV for each and every point (down to some pre-defined resolution) in the search space, and picking the point with the highest value. As simple as this sounds, this technique is almost never used in practice for several reasons:

1. Depending on the size of the search space and the desired resolution, the number of points that must be evaluated can become intractable,
2. The method pre-supposes all satellite observations to be error-free; however, a relatively small amount of error in one or more observation can result in an AFV that is quite far from 1.0,
3. Unmodeled errors can result in an incorrect set of coordinates having a higher AFV than the correct baseline coordinates, and
4. The method cannot be used over time unless both receivers remain stationary since we are searching in the position domain.

The reason this method is so intractable is due to the minimum resolution required to guarantee that the correct position is not missed. Since we are searching for positions that make the double-differenced ambiguities the most nearly integer, we must actually evaluate a position that is quite close to the intersecting ambiguity lines in the first place. It is clear, for example, that at a search resolution of 9.5 cm, or one-half the carrier wavelength, it is possible for the set of evaluated locations (in one dimension) to come no closer to the correct integer gridline than 4.75 cm:

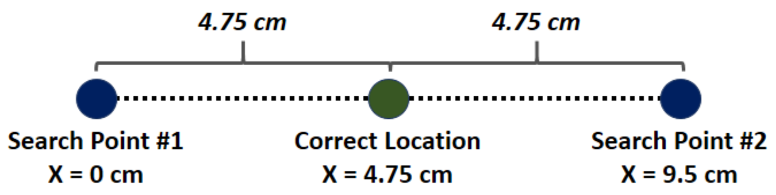


Figure 13: Maximum 1D AFM error due to a search resolution of 9.5 cm

By plugging this worst-case residual value of 4.75 cm into AFV Equation 41, we see this would result in an AFV of 0.0, clearly nowhere near 1.0. In order to achieve a reasonable AFV of at least 0.8, for example, the absolute minimum search resolution required would be nearly 4 cm. Given a typical GPS 3D RMS position error of 4 m, this would require $(\frac{4\ m}{4\ cm})^3 = 1,000,000$ individual search points for a single observation epoch!

In addition to the computational complexity of searching through the entire search space in the absence of measurement noise, real-world errors dictate that either much higher resolutions or much larger search spaces be used to ensure that the correct position is not missed. As such, this method has been all but abandoned for use in modern GPS localization techniques; however, it should be noted that this method alone stands apart from the rest due to its complete immunity to cycle slips. The fact that the method simply takes the integer-ness of a solution into account coupled with the fact that cycle slips are always integer in nature (barring half-cycle slips which will be discussed in "Section: *Extended AFM with Hill Climbing*" in Chapter IV) means that cycle slips which normally wreak havoc on other positioning techniques pose no problems whatsoever for the Ambiguity Function Method.

AFV Peak Tracking (APT) Solution

Because of the AFM's immunity to cycle slips, as well as its direct position-domain search space, it was chosen as the preferred method on which to build our baseline determination algorithm. Unfortunately, due to the errors and complications listed in the previous section, it could not be coupled with our tracking results and simply applied as a direct solution to the centimeter-scale relative localization problem.

Instead, we view the AFM search space in a different light, creating a contour-map topology in which the AFV at a specific point corresponds to the point's overall "fitness" as a potential candidate for the baseline solution:

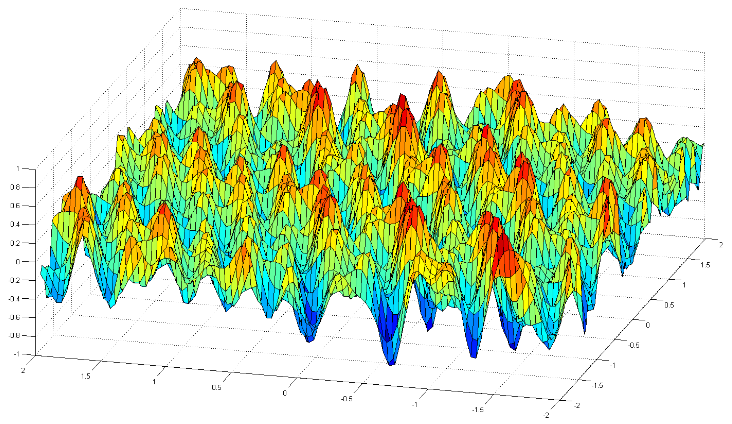


Figure 14: AFM search space as a contour map in two dimensions

Figure 14 shows a graphical representation of such a contour map in two dimensions, where the X and Y axes correspond to the X and Y dimensions in the standard AFM search space, and the Z axis represents the corresponding AFV, or fitness value, at the specified location. Red areas in this figure indicate regions in which the *correct* relative baseline is more likely to fall, while blue areas indicate regions of unfitness. It is clear, although the map is quite jagged and hilly, that the overall fitness function is relatively smooth and continuous over short intervals, with easily identifiable *peaks* and *valleys* present in the topology. It is precisely these peaks that we are referring to in our name "AFV Peak Tracking Solution."

When this same search grid is expanded to include three dimensions, it is much more difficult to visualize the so-called peaks in the topology, but they are there nonetheless and correspond to the darker, redder-colored dots in Figure 15.

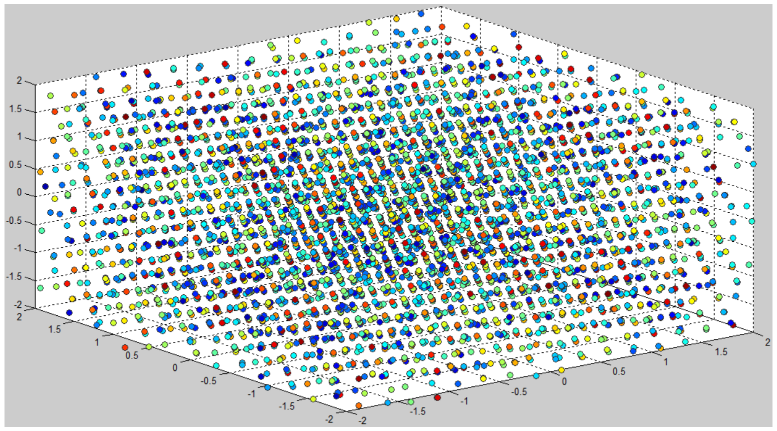


Figure 15: AFM search space as a heat map in three dimensions

The previous section explained how AFVs close to 1.0 correspond to baselines that make the double-differenced integer ambiguities in our model the most integer-like, and therefore, the most "correct" in a mathematical sense. This becomes quite easy to see in either of the figures above by recognizing that the various peaks correspond to likely baseline candidates. We also discussed that, due to multipath and unmodeled errors, there is a high probability that the highest peak will not actually correspond to the correct baseline solution. Recall from Figure 12, however, that the **correct** baseline position will experience continually low error residuals (i.e. will remain a peak) through significant changes in both satellite and receiver geometry. Extending that concept to our AFV search grid, it is clear that peaks at incorrect locations will fade and turn into valleys as the change in geometry causes their fitnesses to decrease. As such, the "tracking" in our AFV Peak Tracking Solution refers to the fact that we not only identify the highest peak(s) at a given point in time, but also track these peaks such that it becomes possible to filter out the erroneous candidate baselines as their corresponding AFVs decrease.

In the simplest of cases, this procedure equates to initially searching over some pre-defined search space for all of the AFV peaks (i.e. baseline candidates) and then filtering through time by evaluating each remaining peak according to the newest epoch of GPS data and removing any baselines from the candidate set that are no longer valid. At some point, there will only be one valid peak remaining, corresponding to the correct relative baseline between the two receivers.

As discussed in the previous section, however, we are not carrying out this technique in an ideal situation in which we can simply ignore the unmodeled errors and assume that we are either completely stationary or that our tracking results are completely error-free. In such a case, we would simply be able to track the *values* of the peaks through time, but in our case, we are required to track the *peak locations* as well as their values, since they are most likely in constant motion, and our knowledge of these motions is not completely without error.

Extended AFM with Hill Climbing

Although the implementation of the algorithms in "Section: *Relative GPS Tracking Algorithm*" in Chapter IV provides highly accurate tracking results, they are nonetheless imperfect. As mentioned earlier, even single centimeters of error can lead to drastically low AFV levels. As such, a method of correcting for or overcoming these inaccuracies must be employed to make the tracking results useful in a motion-enabled, AFM-based technique. As such, we extend the AFM method with a simple hill climbing algorithm to overcome any inaccuracies in our tracking results and derive a new thresholding function to be used in determining an appropriate AFV level under which to filter our results.

Hill climbing algorithms provide a useful methodology for overcoming small optimization discrepancies by searching in a region close to or around an arbitrary initial position estimate for the "local maximum" of some fitness function. In our case, the fitness function is the AFV equation itself, and the location around which to search is an estimate of the baseline between two receivers that has been tracked through time (i.e. the "tracked peak" introduced in the previous section).

Assuming that our tracking results are able to provide sub-centimeter-scale precision *over single epochs of time*, the tracking errors would not be large enough to push a potential baseline candidate so far from its corresponding local AFV maximum that it enters the domain of a different (i.e. wrong) local maximum. In other words, the local maximum of the

AFV function for a given position at time t will be the same local maximum for the updated position at time $t+1$ if our tracking results are used over a single epoch.

This statement is only true for a finite amount of time, however. In order to guarantee that this claim always remains true, we must re-position our baseline estimate after every time step such that it always coincides with its corresponding local maximum. In essence, this is the same as removing any bias that would normally accumulate over time in a dead-reckoning tracking algorithm such as ours.

Fortunately, since the AFV equation represents a continuous, smooth function in the position domain, we can directly use a steepest-ascent hill-climbing technique to quickly find the local maximum given our initial baseline estimate. All that this entails is evaluating the AFV equation for the points immediately around the estimate, and then repositioning it to be the point that experienced the largest increase in AFV from the original estimate. This process is repeated for the updated estimate and so on and so forth until no point is found with a better AFV than the current one. The corresponding point will be the local maximum of the function in the desired region.

At the next time step, we update our baseline estimate with the tracking results from the previous section, and then carry out this hill-climbing technique to once again remove any tracking error we incurred. In such a way, any position estimate (even if it is the wrong one) can be continuously tracked through time according to its fitness within the paradigm of the AFM. There remains to be seen, however, an effective way to remove candidates that no longer fit based on the changing satellite and receiver geometries.

AFV Thresholding Function

We mentioned at the beginning of this section that some sort of thresholding function would be an ideal way to filter out baseline positions that have become unfit candidates over time. Such a thresholding function can be formed by taking into account the fact that:

- Since we are searching over a discrete search space, there will almost always be some amount of error due to the offset between the "correct" position and its nearest search point in the grid, and

- The carrier phase observations used for AFV determination are not perfect measurements and will include some amount of error, due primarily in this case to multipath and receiver noise.

As such, the error term that was omitted to form the simplified carrier phase model in Equation 36 should not be overlooked. Let us first re-derive the AFV equation without ignoring the error term. The full double-differenced carrier-range model is given by Equation 42.

$$\nabla\Delta L_{jk}^{mn}(t) = \nabla\Delta\rho_{jk}^{mn}(t) + \lambda_{L1}[\nabla\Delta N_{jk}^{mn} + \nabla\Delta\epsilon_{jk}^{mn}(t)] \tag{42}$$

Again, we isolate the integer ambiguity on the left-hand side of the equation:

$$\nabla\Delta N_{jk}^{mn} = \frac{\nabla\Delta L_{jk}^{mn}(t) - \nabla\Delta\rho_{jk}^{mn}(t)}{\lambda_{L1}} + \nabla\Delta\epsilon_{jk}^{mn}(t) \tag{43}$$

Then, we take the cosine of its radian form to compute the AFV corresponding to the n^{th} satellite.

$$\cos(2\pi[\frac{\nabla\Delta L_{jk}^{mn}(t) - \nabla\Delta\rho_{jk}^{mn}(t)}{\lambda_{L1}} + \nabla\Delta\epsilon_{jk}^{mn}(t)]) = \text{AFV}^n \tag{44}$$

This leaves us with an equation in which the unknowns are the 3D baseline coordinates (used to determine the $\nabla\Delta\rho_{jk}^{mn}$ term) and the $\nabla\Delta\epsilon_{jk}^{mn}$ error term.

Not immediately apparent is the fact that the λ term may also be unknown. We have discussed the nature of cycle slips earlier in "Section: *Carrier Cycle Slips*" in Chapter III, but some receivers also suffer from what are known as "half-cycle slips." These slips occur for the very same reason as full-cycle slips; however, they appear to add or remove cycles equal to only one-half the wavelength of the carrier wave. The reason these slips occur is because many receivers choose to remove the BPSK navigation messages from the received L1 signal by simply squaring the received waveform. Since a squaring operation makes all negative values appear to be positive, the resulting squared carrier wave will not repeat itself every λ_{L1} meters, but rather every $\frac{\lambda_{L1}}{2}$ meters.

Fortunately, most receivers that operate this way are able to detect when their carrier phase tracking loops get out of sync with the unsquared version of the received signal by a half-wavelength and will return this information along with the raw carrier phase measurement to indicate that ambiguity resolution *may* need to be carried out using half-wavelengths, denoted from here on as $\lambda_{L1/2}$. As such, the λ value in Equation 44 must be treated as a pseudo-variable since its value can change and the AFV equation depends on it, but it will always be known prior to use.

Finally, it was discussed in the previous section that the AFV equation will necessarily contain some amount of error due to the resolution of the search space. We gave an example in one-dimension that showed the worst-case error to be equal to one-half the search resolution ($\frac{res}{2}$); however, GPS operates in three dimensions, so it is possible for the search point location (which determines the calculated $\nabla\Delta\rho$ value) and the correct receiver location (which determines the $\nabla\Delta L$ value) to differ by up to $\sqrt{(\frac{res}{2})^2 + (\frac{res}{2})^2 + (\frac{res}{2})^2} = \sqrt{3 \cdot (\frac{res}{2})^2}$.

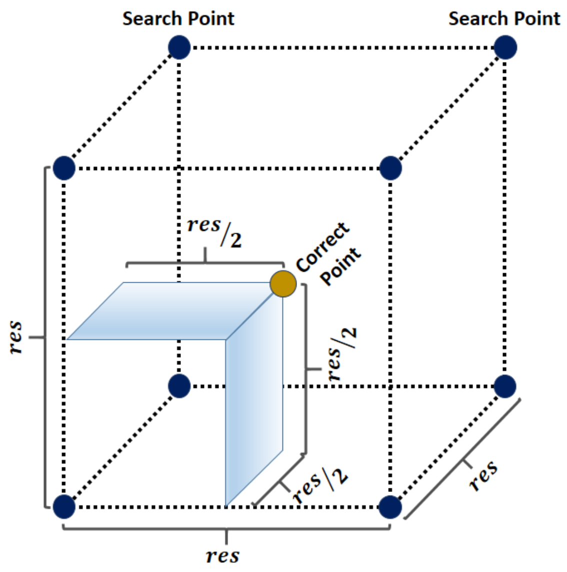


Figure 16: Maximum 3D AFM error due to search resolution

Now that we have determined the worst-case error due to the search resolution, investigated the possibility of a half-cycle slip, and included a catch-all term for any unmodeled errors (dominated by multipath), we can re-write the AFV Equation 44 for satellite s as a single-satellite, worst-case value, like so:

$$\text{AFV}^s_{worst}(res, \lambda, \epsilon) = \cos\left(2\pi\left[\frac{\sqrt{3 \cdot (\frac{res}{2})^2}}{\lambda} + \epsilon\right]\right) \tag{45}$$

Extending this to include all valid observations, n, for a given epoch, we arrive at an AFV thresholding function, dependent on the search resolution used, ambiguity resolution required, and worst-case error present in any given observation:

$$\text{AFV}_{th}(res, \lambda, \epsilon) = \frac{\sum_{a=1}^{n} \cos\left(2\pi\left[\frac{\sqrt{3 \cdot (\frac{res}{2})^2}}{\lambda} + \epsilon\right]\right)}{n} \tag{46}$$

Note in the equation above that the λ value may be different for individual satellite measurements since not all observations require half-cycle ambiguity resolution. Also recall from "Section: *Multipath*" in Chapter III that carrier range errors due to multipath have a maximum error of ~5 cm, but not all observations will experience multipath, and those that do are not likely to experience it in the same direction or with the same magnitude as one another; therefore, a reasonable technique is to scale down the maximum value of multipath error (ϵ^{max}_M) by the number of visible satellites (n) and use the resulting value ($^{\epsilon^{max}_M}/_n = {}^{0.05}/_n$) as the ϵ term in the AFV thresholding function above.

Putting It All Together

This chapter introduced several concepts used by the APT localization algorithm, but we have yet to put them together and state them as a complete methodology for relative baseline localization through time. The explicit steps in APT can be separated into three logical phases of execution: Initialization, Calibration, and Steady-State Localization. The following algorithm details the explicit steps comprising each of these phases:

Algorithm 3 AFV Peak Tracking (APT) Solution

```
Initialization
1:  Evaluate the AFV for all 3D grid coordinates in a given search cube
2:  Identify all peaks from the resulting AFV set
Calibration
3:  Wait until the next epoch of data arrives
4:  for each identified peak do
5:      Update the peak location according to the computed tracking result
6:      Re-evaluate the AFV at the updated peak location
7:      Hill climb by steepest ascent to the local peak maximum
8:  end for
9:  Calculate the worst-case AFV threshold value for the current epoch using Equation 46
10: Filter the updated peak locations based on the computed threshold value
11: If more than one peak remains, go to Step 3
Steady-State Localization
12: Update the peak location according to the computed tracking result
13: Re-evaluate the AFV and hill climb to the local peak maximum
14: if AFV remains at an acceptable level then
15:     Go to Step 12 at the next time epoch
16: else
17:     Go to Step 1
18: end if
```

In essence, the goal of this algorithm is to reach the steady-state phase for every remote node participating in the localization procedure. Once this stage is reached, the relative baseline between two receivers (i.e. the current peak location) can be computed at a high level of accuracy using only minimal computational resources.

This methodology is very easy to understand graphically. At some point in time, t, we have carried out the initialization phase of the APT algorithm and are left with a set of candidate peak locations (represented here by dots):

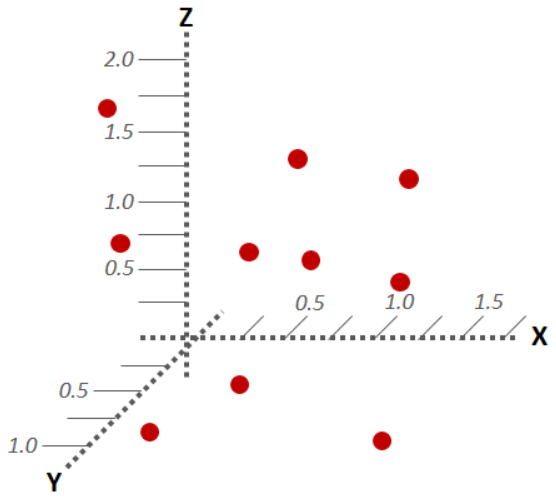


Figure 17: Candidate peak locations at time t

At the next time step, t+1, we update each of the peak locations according to the tracking results, which in this graphic are (0.75, 0.0, 0.25):

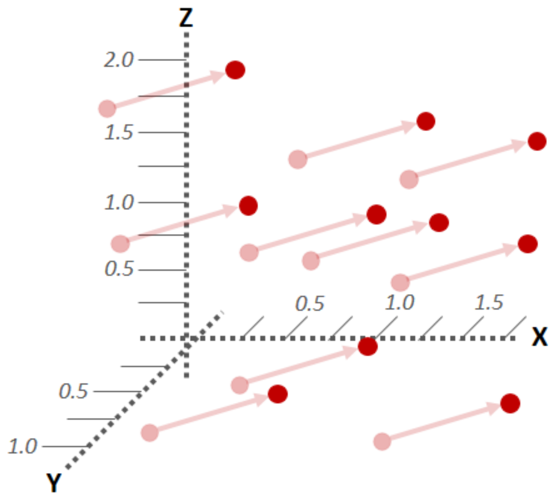


Figure 18: Candidate peak locations at time $t+1$

We then re-evaluate the AFVs for each peak candidate and hill-climb to the local maximum:

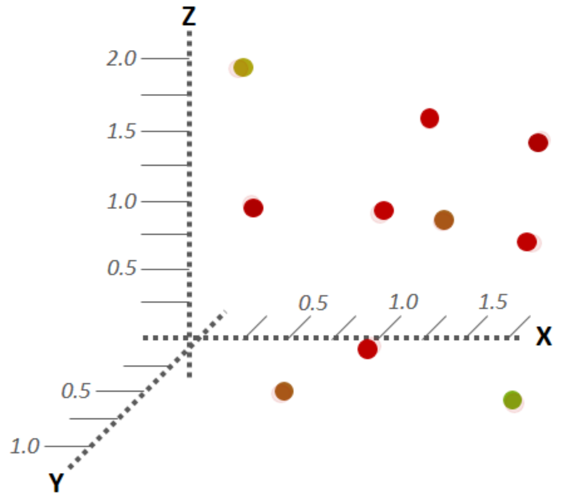


Figure 19: Candidate peak locations after hill-climbing and re-evaluation

From the resulting peak locations and AFVs, we filter out unsuitable results using the calculated threshold value, such that only a subset of the peak candidates remain:

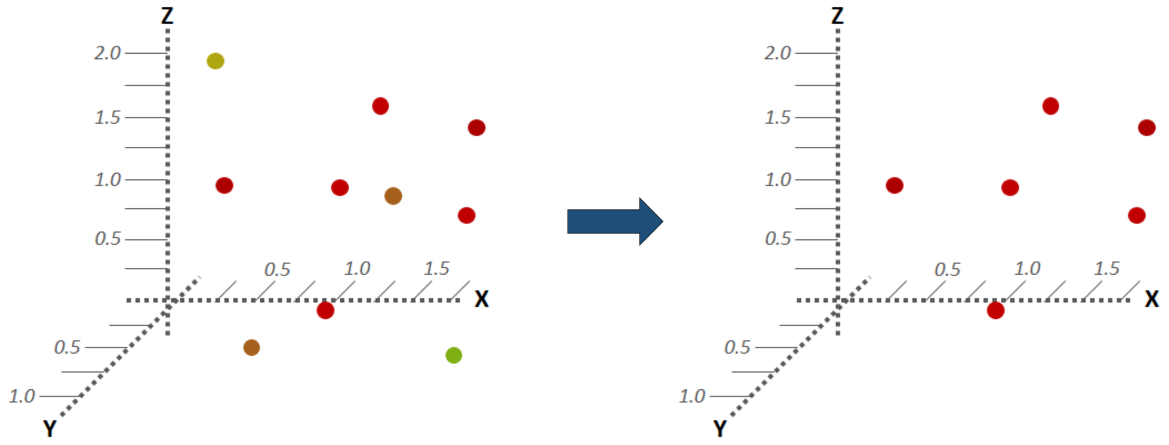


Figure 20: Candidate peak locations before and after threshold filtering

This process repeats itself at every time step until only one peak remains, corresponding to the correct relative baseline between two receivers. This single peak continues to be tracked through time, providing highly accurate relative node location information while requiring only minimal computational complexity.

It should be noted that since we have a set of potential relative coordinates at each time step which, by definition, correspond to sets of integer ambiguities containing minimal error, it is trivial to actually calculate the integer ambiguities for each peak using quite literally any satellite as a reference and then to store them for later use. Then, at the next time step, if the satellite previously used as a reference is still visible and without error, the integer ambiguities can be used to compute the current relative baseline in a least-squares sense, thereby providing a means of both verifying the validity of the location of the tracked peaks (i.e. a sanity check) and also providing additional criteria on which to filter and remove unfit peak candidates.

CHAPTER V

RELATIVE LOCALIZATION ARCHITECTURE AND PRACTICAL CONSIDERATIONS

In order to experimentally verify the validity of the new algorithms and methodologies described in this dissertation, we decided to create a fully functional proof of concept. This chapter describes the system and implementation details of the working prototype, as well an analysis of the various hardware and software components and their impact on the scalability of the system.

System Overview

The following diagram outlines the individual software components comprising our relative localization system and shows how they connect and interact with one another:

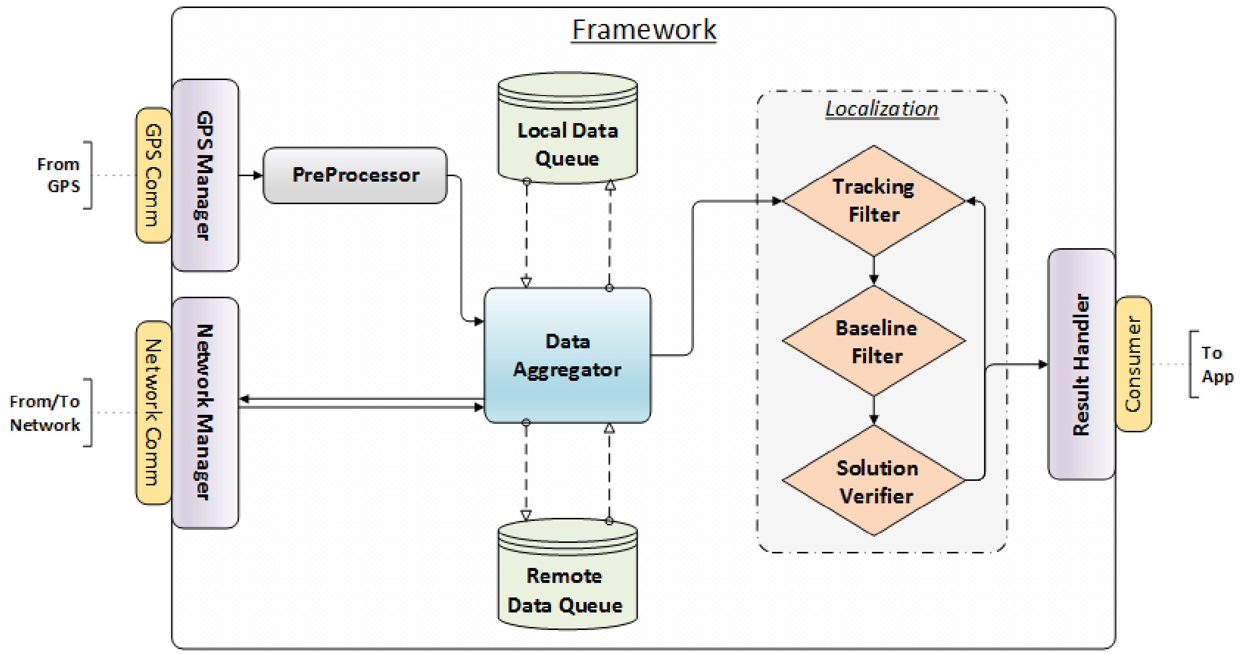


Figure 21: Software framework functional diagram

In Figure 21 above, several conventions are used to aid in understanding how the software framework fits together as a whole, as well as how it interacts with the world around it. Perhaps most important is the solid black line labeled "Framework" that encompasses the majority of the individual software components. This boundary delineates a standalone, platform-independent software service that can be instantiated any number of times on any number of devices without the device or user knowing anything about the inner workings of the framework.

The interfaces into and out of the framework are denoted by yellow rectangles connected to the outside of the framework boundary. These interfaces are just that – interfaces – meaning that they require interface implementations to be written specifically for the device the framework is running on, as well as the inputs coming into the device. For example, the GPS chip that we use in our prototype (see "Section: *Hardware and Platform Components*" in Chapter V) transmits raw satellite data using a proprietary format that must be decoded before being passed into the framework. We therefore write a decoder implementation specific to the data format of the chip and attach it to the "GPS Comm" interface. Likewise, the implementation of the "Network Comm" interface will change depending on the networking technology we are using (e.g., Bluetooth, 3G, UDP Multicast, etc.) and the device the framework is running on. To reiterate, the only platform-specific implementations that must be written correspond to the three interface boxes, making this framework extremely easy to port to virtually any device with the computational power to run it.

The benefits of this black-boxing approach are threefold. Namely, the framework:

- Can easily be ported to run on any device,
- Can be run in multiple instances on the same device (useful for simulation or log-file playback), and
- Can be run across multiple devices and device types, with the guarantee that the results of internal computations will be mathematically identical, regardless of the device it is running on.

In fact, the framework can currently run as both a PC service on any operating system or on any handheld device running Android version 4.0 or later. This proved to be a very useful tool in creating and analyzing experimental results since it allowed us to conduct experiments very early in the project on both mobile and laptop devices and then continue to use the raw log files from these experiments in further research, knowing that the computational results of the research would be identical across devices since only one code base was used.

The individual software components within the framework, like the framework itself, are black-boxed from one another and communicate via unidirectional message passing using a common set of pre-defined messages and message types. Each component is referred to as a "module" throughout the rest of this chapter, with the black-boxed nature of the configuration allowing us to drop in, change, or edit any of several possible module implementations on the fly without requiring reconfiguration of the rest of the software.

Software Components

In this section, we describe each of the individual software components in Figure 21 in detail. All software was written in pure Java, ensuring cross-compatibility between various systems and allowing for ease of benchmarking.

GPS Manager

The GPS Manager is responsible for receiving decoded GPS data, parsing it into a usable format for the rest of the framework, and packaging all of the satellite data corresponding to a single GPS epoch into one cohesive unit before forwarding it to the next module for pre-processing. This module is also responsible for creating any log files, if desired, since it is the only component with full access to the raw satellite data before any processing, parsing, or transformations have been carried out.

Preprocessor

The Preprocessor module handles the bulk of the error mitigation and correction techniques employed by the framework. This includes filtering the data for obvious outliers and error-prone observations, as well as extrapolating all measurement data to the common GPS epoch as outlined in "Section: *Time Bias Data Extrapolation*" in Chapter IV. The first preprocessing step involves removing observations from the data set for satellites:

- That are transmitting a satellite health problem as determined by the Control Segment,
- Whose orbital accuracies (as determined by the CS and transmitted in the navigation messages) are above a threshold of 5 meters of error,
- Whose received signal strengths (carrier-to-noise ratios) are below a minimum threshold of 28 dBHz (from a maximum signal strength of 44 dBHz), or
- Whose elevations in the sky relative to the GPS receiver are lower than 15°.

The remaining observations are then tested for potential cycle slips using Formula 47.

$$\epsilon_{L,threshold} < |\hat{L}(t) - L(t)| \tag{47}$$

with:

$$\hat{L}(t) \approx L(t-1) - \lambda_{L1} \frac{f_D(t) + f_D(t-1)}{2} \tag{48}$$

where:

$\hat{L}$ is the predicted carrier range at the specified time
L is the actual carrier range at the specified time
f_D is the instantaneous Doppler shift at the specified time
λ_{L1} is the wavelength of the carrier signal
$\epsilon_{L,threshold}$ is a threshold value for the difference between the *predicted* and the *actual* carrier range values

Since the **instantaneous** Doppler shift is immune to cycle slips (unlike the carrier phase observable), and the carrier phase is created via integration of the actual, or **continuous**, Doppler shift through time, the change in carrier phase over one epoch should be close to the *average* of the instantaneous Doppler shifts at the limits of the observation period. Since these two values are only approximately equal, we take any deviation over an absolute threshold of 5 wavelengths ($\epsilon_{L,threshold} = 5\lambda_{L1} \approx 1$ m) to indicate the possibility of a cycle slip. These measurements are simply marked as potential slip candidates, and it is up to the individual modules to determine how to handle a slip (e.g., slips adversely affect the tracking algorithm, whereas they are irrelevant for the AFM-based localization solution).

After any slipped measurements have been marked and erroneous measurements discarded, the preprocessor estimates the current clock bias of the receiver from GPS time using a simple least-squares positioning algorithm, as can be found in [24]. The resulting clock bias value is used to extrapolate all of the data from the current observation set to the correct GPS epoch using the algorithm described in "Section: *Time Bias Data Extrapolation*" in Chapter IV. The fully preprocessed data is then sent to the next module for aggregation.

Network Manager

The Network Manager's only two functions are to package preprocessed data into a format that can be easily transmitted over a network link, and to unpack network data received from other remote GPS nodes into the same format as the local preprocessed data. Upon reception of remote data, the data is parsed into the correct format and immediately forwarded to the next module for aggregation.

Data Aggregator

The primary responsibility of the Data Aggregator is to match local and remote data from the same GPS epoch, such that a single cohesive package containing only relevant data points can be sent to the localization modules for further processing. When a local data packet is received from the Preprocessor module, this module simply forwards it to the Network Manager to be transmitted to other remote nodes. It then stores the data in a local database for later use, simultaneously removing any stale data that is five seconds old or older.

Once a remote data packet is received from the Network Manager, this module first stores a copy of it into a database containing all of the remote packets from the corresponding node over the previous five epochs; then, it searches through its databases to find the local preprocessed data corresponding to the GPS epoch of the received packet, as well as the local and remote data corresponding to the previous GPS epoch.

Once all four data packets (i.e. local and remote data for the current and previous GPS epochs) have been located, they are combined into a single "pairwise data unit" that contains the results of all of the various differencing operations that will be needed by the localization algorithms in the future. The reason to have these operations carried out here instead of by the localization algorithms themselves is that, in this case, they only need to be performed once, after which they can be utilized by the various algorithms without incurring additional computational costs. As such, a pairwise data unit is constructed using the following steps:

1. Two reference satellites are chosen from the list of currently visible satellites based on their elevations in the sky and their likelihood to have experienced a cycle slip (where the satellite with the highest elevation and no cycle slips is ranked first),
2. Of the resulting two satellites, one satellite is chosen as a reference which was **not** the reference at the last time epoch. This is done to minimize the potentially correlative effects of multipath (and other error sources) arising from use of the same reference satellite over time,
3. The temporal double difference (see "Section: *Temporal Double-Differencing Model*" in Chapter IV) is calculated for all visible satellites that have experienced no apparent cycle slips between the current and previous epoch,
4. The classical double difference (see "Section: *Double-Differencing Model*" in Chapter III) is calculated using the reference satellite determined in Step 2, and
5. The double-differenced observation is marked to indicate whether it requires half-cycle ambiguity resolution or not, as determined by the GPS chip itself.

After completion of these steps, the resulting pairwise data unit is forwarded to the next module to begin localization as described in the earlier chapters of this dissertation.

It should be noted that this module will never block unless it receives a remote data packet that is newer than any of its local data. This should never happen in a real-time, networked mode of operation since the network latency should be far greater than the latency of the preceding modules running on a local framework; however, this behavior has been noted when playing back log files on a local machine. All module operations, however, begin on a new thread once a data packet is received, so any remote packets that come in and can be matched immediately with a local packet will not be held up by a blocked module. Likewise, if a *local* packet is produced that is newer than the one for which the module is blocking, the framework will assume that the missing epoch included a complete loss of all satellite locks, and it will discard any pending operations for epochs up to the current valid local data packet. As such, this module will not cause a bottleneck under any circumstances.

Tracking Filter

The Tracking Filter implements a smart version of the relative tracking algorithm described in "Section: *Relative GPS Tracking Algorithm*" in Chapter IV, including use of the clock drift estimation and matching procedure described in that section to detect previously unresolved cycle slips and poor observation data due to multipath and other unmodeled errors. Since there is the possibility that pairwise data may be unavailable for relatively short periods of time, such as when walking in a heavily-wooded area or driving underneath an overpass on the interstate, a constant-acceleration motion model is employed to bridge the gap between non-consecutive periods of satellite visibility. The complete tracking filter works as follows:

1. Upon receipt of continuous, consecutive satellite data, the filter carriers out a "standard tracking update," which involves direct implementation of the tracking algorithm described in "Section: *Relative GPS Tracking Algorithm*" in Chapter IV. Since our update rate is 1 Hz, the resulting track is equivalent to the relative velocity in meters per second in our motion model. Likewise, the relative acceleration is equal to the difference between the current relative velocity and the relative velocity of the previous epoch.

2. If non-consecutive satellite data is received, meaning that one or more epochs of data are unavailable, but the missing number of epochs is relatively short (i.e. less than 5 seconds long), the motion model will simply update the relative velocity using its estimate of the relative acceleration.

3. If non-consecutive satellite data is received and the missing number of epochs is greater than 5 seconds, the filter is reset with a relative velocity and acceleration of 0.

In all cases, the resulting relative velocity is checked to ensure that it does not exceed an unrealistic threshold (currently 100 km/s), and its value is passed to the next module as the tracking result for the current epoch.

Baseline Filter

This module uses the previously determined relative tracking results to carry out the APT baseline localization algorithm described in "Section: *Relative GPS Localization Algorithm*" in Chapter IV. The implementation of this algorithm is straightforward and follows directly from the listing in Algorithm 3. Once again, recall that the APT algorithm is unaffected by whole cycle slips; thus, no slip detection is required to maintain a high level of accuracy in this module. Conversely, half-cycle slips **do** affect the solution, but they are detected by the GPS chips themselves. Upon receipt of an observation that *may* contain a half-cycle slip, this module will simply:

- Replace the λ_{L1} value in AFV Equation 41 with $\frac{\lambda_{L1}}{2}$, and
- Use $\frac{\lambda_{L1}}{2}$ as the wavelength argument to the thresholding function, Equation 46

Previous chapters mentioned the dramatic impact the search resolution can have on the resulting Ambiguity Function Values; however, increased resolution has an exponential effect on the computational complexity of the initial peak detection algorithm. As such, we use two different search resolutions depending on the phase of execution we are currently in. In the initialization phase in which we are simply trying to identify the locations of the peaks in a search space, we use a coarse-grained resolution of 4 cm between search points. This represents a high enough resolution to determine whether or not a peak exists in a given region, but cuts down on the number of points that must be evaluated to cover the entire space.

After this initialization step (which lasts only one epoch), we enter the calibration phase in which we hill-climb using a much finer resolution of 5 mm. This ensures that the error due to misalignment between a discrete search point and the *real* baseline position is small and will have a minimal impact on the AFV. Note that this is the resolution that gets passed as an argument to AFV Thresholding Equation 46 to help determine the worst-case value that could arise from the various error sources in this technique.

Finally, once only a single peak candidate remains, we have effectively identified the correct relative baseline between ourselves and the remote receiver being localized. At this point, we enter the steady-state localization phase of the algorithm in which we simply continue to track the peak through time and monitor it for any unexpected errors. Such an error will manifest as a sharp and significant lowering of the peak's associated AFV over the course of one epoch, at which point, we will be required to re-initialize the APT algorithm. Luckily, however, we can assume that, although our peak tracking algorithm has experienced an unknown error (most likely due to multipath or a loss of lock), and the peak can no longer be considered to be valid, it is still likely to be quite close the correct location (i.e. at least within a meter of error). As such, re-initialization can take place over a dramatically smaller search space than was initially used, enabling us to quickly re-determine the correct relative baseline without spending too much processing time in the calibration phase.

Solution Verifier

Solution verification in this case is actually carried out on all potential baseline candidates identified in the previous module and involves comparing the APT baseline solution with a fixed double-differenced solution (see [24]) using the estimated carrier phase ambiguities from the previous epoch of data. As mentioned earlier, the result of a successful iteration of APT will be a relative peak location that would produce a set of carrier phase ambiguity values that are quite close to perfect integers. In our framework, we go one step further and actually compute these integer ambiguities for all peak candidates and store them until the next time epoch.

Once each peak has undergone an iteration of the APT algorithm, the locations of the remaining peaks are then recalculated in a least-squares sense using the fixed ambiguity values from the previous observation epoch. If the APT solution and the fixed solution agree, we know that the current baseline candidate remains a good fit in our overall localization paradigm. If the solutions do not match, then the Ambiguity Function Value corresponding to the fixed solution is computed (the AFV is already known for the APT solution from the previous module) and compared to the APT AFV. In the likely case that a cycle slip has occurred, the AFV from the fixed solution will be significantly lower than the AFV from the APT solution, which should remain above the threshold value calculated from Equation 46. In this case, the APT solution is simply accepted and the ambiguity values are recomputed to remove the cycle slip for the next epoch's data set.

If, on the other hand, the AFV from the fixed solution remains above the threshold value but the APT AFV has dropped significantly, this is indicative of an error in the tracking solution itself. In this case, the location of the peak is updated to be consistent with the fixed solution. In the final case, if both AFVs are below the threshold value, we know that the baseline candidate is simply no longer a good fit, and as such is not worth further consideration. With this in mind, our implementation of the framework described by Figure 21, actually combines the solution verification module with the baseline filter itself, since it became apparent that non-verification of a solution could be a useful additional criterion on which to filter potentially unfit baseline candidates. It should be noted, however, that this module was retained as a separate and additional processing step in the description of our framework because *statistical* analysis and verification of the reported baseline solution from the previous module is currently lacking but is **strongly** advisable in future work, not only as a means to increase system robustness, but also to provide better performance metrics and to handle the potential case when an unknown filtering error occurs and incorrect position information is returned by the system.

Result Handler

The purpose of this module is simply to repackage the baseline solution into a standard, easy-to-read format for use by whatever application is consuming the result. It takes the verified solution, along with the confidence value corresponding to the reported baseline, and sends a single message to the output interface containing the relevant GPS epoch number, the applicable device name or ID of the remote node, the relative 3D baseline vector, and an estimated accuracy value for the result.

Processing Time and Memory Loading

This section provides an overview of the processing times and memory requirements of each of the software modules comprising the relative localization framework, enabling analysis of the scalability of the framework as a whole. Memory requirements were determined using the Eclipse Memory Analysis Toolkit [17], and processing latencies were computed from real-world experimentation. Results denoted by a "PC-based" benchmark were formed from the average of the corresponding operations run 10,000 times each on a:

- Fujitsu Lifebook S Series, Intel Core 2 Duo @ 2.53 GHz, 4 GB RAM, Windows 7
- Dell Precision T1600, Intel Xeon @ 3.30 GHz, 4 GB RAM, Windows 7

Results denoted by a "mobile-based" benchmark were formed from operations run 10,000 times each on a:

- Google Nexus 7 Tablet, ARM Cortex-A9 @ 1.2 GHz, 1 GB RAM, Android 4.3

GPS Manager

The processing time of this module is negligible, and its throughput is limited to the incoming data rate of the GPS packets, which, by design, is bursty with a 1 Hz transmission cycle. Specifically, the latency from packet reception to module output is 6.64 μs for the framework running on a PC and 97.9 μs when running on an Android device.

From a memory standpoint, this module requires a maximum of only 1.52 kB of storage per full epoch of data, determined by the number of visible satellites at any given epoch, with a maximum of 12 satellites visible per unit time. In addition to the observation data, this module must also keep track of the received ephemeris data from each satellite. Fortunately, this memory can be allocated at one time and does not grow with the addition of satellites. The storage requirement for these ephemerides is a constant 24.03 kB of data, making the total maximum memory footprint for this module 25.55 kB.

Preprocessor

Although this module contains a significant number of vital processing steps, the total processing time for all of its operations is a negligible 129.3 μs on a PC-based framework and 4.0 ms on an Android device. The maximum memory requirements for this module are 4.38 kB of storage, with the bulk of the memory being used for constant storage of the full ephemeris data from each satellite. The reason the processing time and memory requirements for this module are so low is because all of the operations can be expressed in the form of deterministic, closed-form algorithms which contain a minimum number of algebraic or matrix operations (3 matrix multiplications and 1 inversion) per epoch.

Network Manager

Since the mathematical functionality of this module is limited to repackaging data into different formats, its average processing time on a PC is only 12.4 μs and 405.3 μs on a mobile device. Likewise, the memory requirements are limited to the amount of storage required to hold both the network- and framework-based formats of one epoch of satellite data, which in this case represents a maximum of 1.36 kB.

Data Aggregator

The primary role of this module is to match data corresponding to the same measurement epoch from different GPS receivers. Since only five epochs of data are stored in the local and remote data queues, epoch-matching is a trivial operation, attributing the bulk of the processing time to carrying out the differencing operations between the various satellites and epochs of data. As such, the total latency for this module from remote packet reception to module output is 19.9 μs for a PC-based framework and 1.08 ms for the Android framework.

The memory requirements for this module are dominated by the data queues used to store the past five epochs of data for both the local and remote nodes. These queues, coupled with the storage necessary to carry out any differencing operations, total a maximum of 66.95 kB of data at any given time for a single remote node. It should be noted that each additional remote receiver that participates in the localization procedure will require another maximum of 33.12 kB of memory to store five epochs of received data, although in reality, this number is likely to be only half as much, since an average of six to seven satellites are usually visible, and this memory value corresponds to the consistent visibility of twelve satellites.

Tracking Filter

This is the first module whose computational complexity depends on the specific data observations passed to it, as some processes iterate multiple times using different parameters and inputs to remove any erroneous measurements and provide the highest possible tracking accuracy for subsequent modules. As such, processing times may vary; however, the *types* of operations performed are still computationally cheap (i.e. 3 matrix multiplications and 1 matrix inversion per iteration). For this reason, the average processing time on a PC-based platform is 24.1 μs and the average on an Android-based platform is 845.4 μs. The standard deviation of the processing times (over many hours of experimental data) is only 10.0 μs on a PC and 412.1 μs on Android, meaning that even in the worst case, the module still provides good throughput and will not become a bottleneck in the system.

Regardless of the number of iterations required to come to an acceptable solution, the memory requirements of this module do not grow over time, but rather re-use data structures from previous iterations for further processing. As such, the maximum memory requirements for this module are 1.68 kB of data for any given epoch.

Baseline Filter

The processing latency of this module varies drastically between the initialization, calibration, and steady-state localization phases of the APT algorithm. The worst-case latency occurs during the initialization phase and is dependent on the initial size of the search space being explored. In general, for a search cube with an area of 8 m^3, the average processing latency for the initialization phase is 46.4 ms on a PC-based framework and 109.7 ms on Android. After initialization, the algorithm will quickly weed out a large majority of the "unfit" peak candidates, such that processing times dramatically decrease. The average calibration latency, measured until one and only one peak candidate remains, is 1.6 ms for the PC and 11.5 ms for Android. Finally, once the correct relative baseline has been identified and the module has entered the steady-state localization phase, processing becomes trivial and the latency averages 850 μs for the PC and 10.3 ms for Android.

The memory requirements of this module, similar to the processing latencies, are highly dependent on the state of the algorithm. In the absence of any peaks, the module requires a maximum baseline of 13.99 kB of memory to function. This is trivial compared to the large amount of memory needed to store the list of peak candidates during the initialization and calibration phases of the localization algorithm. Each peak candidate requires a maximum of 960 bytes of memory to store its relevant information, so in the steady-state, the memory footprint of the module is only 14.98 kB. In the initialization phase, however, the memory requirements are completely dependent on the number of peaks in the search space (which in turn depend on the size of the search space itself). For a search space size of 8 m^3, we have found the average maximum memory load to be 127.1 kB during the initialization and calibration phases of the APT algorithm.

Solution Verifier

Since our framework implementation combined the solution verification module with the baseline filter itself, the processing times and memory requirements of this module were included in analysis of the Baseline Filter module in the previous sub-section.

Result Handler

As with the Network Module, this module serves primarily as a re-packager of existing data, so it's processing and memory requirements are minimal. On both a PC- and Android-based platform, the average processing time for this module's operations is 0.0 ms, and it's maximum memory requirements are 460 bytes of data, corresponding to the amount of memory required to hold both the input (396 bytes) and output (64 bytes) format of the same data.

Summary

To summarize, Table 3 gives a breakdown of the processing time requirements for each of the software components, as well as the total latency of the framework from remote data reception to production of a relative baseline result. Times are given both in absolute terms and as a percentage of the total available time for a given epoch of data, namely 1 s. Note that all of the values shown in the following tables are for a single pair of GPS receivers over one observation epoch, and both the PC and mobile devices were operating under default system conditions (i.e., standard operating system installations with no services disabled).

Module Name	**Processing Time (PC)**	**Processing Time (Mobile)**	**Processing Time % (PC)**	**Processing Time % (Mobile)**
GPS Manager	6.64 μs	97.9 μs	0.00%	0.01%
Preprocessor	129.3 μs	4.0 ms	0.01%	0.40%
Network Manager	12.4 μs	405.3 μs	0.00%	0.04%
Data Aggregator	19.9 μs	1.08 ms	0.00%	0.11%
Tracking Filter	24.1 μs	845.4 μs	0.00%	0.08%
Baseline Filter (Calibrating)	1594.0 μs	11.5 ms	0.16%	1.15%
Baseline Filter (Steady-State)	850 μs	10.3 ms	0.09%	1.03%
Result Handler	0 μs	0 μs	0.00%	0.00%
TOTAL (Calibrating)	1.79 ms	17.93 ms	0.18%	1.79%
TOTAL (Steady-State)	1.04 ms	16.73 ms	0.10%	1.67%

Table 3: Summary of processing times for each software component of the localization framework

Table 4 gives a summary of the maximum memory requirements for each software module and the framework as a whole. Note that since each module can only operate on one epoch of data from a single remote receiver at a time, these values correspond to the maximum amount of storage that may be required by the system at any given time.

Module Name	**Memory Requirements**
GPS Manager	25.55 kB
Preprocessor	4.38 kB
Network Manager	1.36 kB
Data Aggregator	66.95 kB
Tracking Filter	1.68 kB
Baseline Filter (Calibrating)	127.1 kB
Baseline Filter (Steady-State)	14.98 kB
Result Handler	0.45 kB
TOTAL (Calibrating)	227.47 kB
TOTAL (Steady-State)	115.35 kB

Table 4: Summary of the memory requirements for each software component of the localization framework

The previous tables indicate the requirements of the localization framework from the point of view of the dedicated hardware on which it runs; however, since the framework makes use of data sharing among nodes, there exists a minimum amount of communication bandwidth that must be available to avoid bottlenecking. Specifically, each receiver will produce an average of 472 bytes/second of data that must be transmitted across a network link (52 base bytes + 70 bytes per satellite observation with an average of 6 visible satellites = 472 bytes). The maximum network data load corresponds to the visibility of 12 satellites and equals 892 bytes/second, so even for large numbers of participating nodes with perfect visibility, network latency is unlikely to be an issue when used with standard wireless communication protocols such as Bluetooth or WiFi.

Hardware and Platform Components

We implemented an actual working prototype of the distributed system shown in Figure 21 using a network of Android smartphones and tablets. Specifically, three HTC Desire smartphones and three Google Nexus 7 tablets provided us a network of six devices with varying levels of processing power and memory capabilities on which to test our system. Each device was paired with exactly one custom Bluetooth-headset [57] that included a μBlox LEA-6T GPS receiver. We chose this particular L1 receiver because it supplies raw measurement data, whereas many current low-cost receivers do not. The standalone accuracy of the LEA-6T is around 2.5 m with an unobstructed view of the sky [55].

In our setup, all GPS measurements were obtained via an external antenna connected directly to the Bluetooth headset which streams raw GPS data (pseudorange, carrier phase, ephemerides, etc.) over a virtual COM port to its paired Android device using the UBX proprietary protocol of the μBlox GPS receiver. The GPS coordinates computed and reported by the receiver were also streamed to the device to allow for comparison between our methodology and the built-in algorithms supplied by μBlox.

To avoid having a single point of failure, we opted for a distributed, symmetric software architecture whereby each device shares its raw GPS data with the entire network and runs the localization algorithm independently on the data received from its peers, as well as from the local GPS receiver. We relied on 3G data communications and an Amazon-based cloud server for network-wide broadcast of the raw GPS readings. In our prototype, we configured the framework to use UDP over WiFi/3G to send data to the centralized cloud server whose purpose was to keep track of the participating GPS receivers and to re-broadcast any received data packets to all other networked nodes.

In playback mode (i.e. when pre-recorded log files were played using a PC-version of the framework), multiple instances of the framework were instantiated in separate processes to simulate the behavior of the framework running on physically unique devices. Each instance communicated with all other local instances via UDP-based loopback multicast over the PC's network interface. In other words, from the framework's point of view, each instance ran on its own dedicated CPU and communicated via a physical network link. The next chapter summarizes some of the experimental results obtained through use of the software framework and hardware configurations described in this chapter.

CHAPTER VI

EXPERIMENTAL TESTING, RESULTS, AND ANALYSIS

In this chapter, we introduce a number of different experiments which are used to verify the validity and accuracy of the algorithms described in Chapter IV. All results are based on the hardware and software components described in Chapter V, with each experiment being categorized by the methodology it uses, the properties it tests, and the environment in which it was carried out.

Our evaluation approach compares the accuracy of the algorithms introduced in this dissertation to the ground truth, as well as to the results produced by the GPS chips themselves; however, to put the results into context, another methodology is needed to transform the *absolute* coordinates reported by the GPS chips into the *relative* coordinate system used by our localization procedures. To achieve this, we simply subtract the absolute positions of one or more receivers from the absolute position of the local receiver currently being used as a reference (the receiver carrying out the localization and tracking operations). In this way, it is possible to create a map of relative locations with respect to the local receiver's coordinate system, exactly as they appear in our own methodology.

Since a substantial amount of measurement error is dependent primarily on the geographic locations of the receiver nodes, this conversion should result in a relative baseline estimate that is more accurate than the absolute coordinates from which it was derived. Additionally, the μBlox chip used in our prototype represents the top-end of several low-cost, L1 receiver lines, and the set of localization algorithms it implements can be considered a good baseline against which to compare our methodology. For the remainder of this chapter, results obtained through use of the algorithms introduced in this dissertation will be referred to simply as "Dissertation" results. Results used for comparison and obtained from the built-in methods of the μBlox GPS chip will be referred to as "μBlox" results.

Node Tracking Results

All of the experiments in this section were used specifically to test the capabilities of the relative tracking algorithms as described in "Section: *Relative GPS Tracking Algorithm*" in Chapter IV. The first two sub-sections deal with **cumulative** tracking results, or in other words, tracking from a dead-reckoning point of view, and the final sub-section examines short-term tracking results on an epoch-by-epoch basis, as this is the primary way in which tracking results are used in our baseline localization algorithm.

Running Track

Establishing the ground truth for localization experiments is always difficult, even more so when mobility is involved. The site for our first set of experiments, a running track with precisely marked lanes, helped greatly in this regard; hence, we tried out multiple different scenarios. The first experiment was a stationary setup with four nodes placed on the track at the corners of a rectangle approximately 94x68 m in size. The second experiment utilized a 9-foot long pole. We placed one receiver at each end of the pole and walked around the track multiple times in the middle of lane 4. The pole was kept perpendicular to the direction of movement; hence, the nodes were moving approximately on the border between lanes 2 and 3, and between 5 and 6, respectively. We also placed a stationary node at one corner of the track. Finally, we constructed an equilateral triangle of three 9-foot long poles and put a node at each corner. This setup was similar to the single pole configuration with an additional node moving along the center of lane 4 ahead of the other two nodes. Once again, a stationary node was also used.

These experiments are useful because the relative distances of the nodes attached to the poles are precisely known at all times, with only the relative directions changing. As such, they provide us with a way to measure the accuracy of the technique in the "free mobility" case with extreme precision. The additional static node adds range mobility to the experiment, since both the relative directions **and** distances to the nodes are changing continuously when viewed from this receiver. This provides performance statistics on the technique when used with baselines of varying lengths. Finally, the stationary node gives us the ability to create absolute ground truth maps for easier visualization, such as using Google Earth as shown in Figure 24.

Stationary Setup

In this setup, four nodes were placed at the corners of a rectangle approximately 94x68 m in size. The track had trees and a building next to it, so the nodes had a partially obstructed view of the sky. Nevertheless, this can still be considered a benign GPS environment. To test system stability, we let the system run for more than 20 minutes. Note that neither our tracking algorithm nor the built-in μBlox algorithms made any assumption that the nodes were stationary.

To evaluate the accuracy of the system, we set the absolute initial position of each node to be equal to the reported absolute location from the GPS chip. This allows for fair comparison since both our methodology and the internal μBlox algorithms start with identical views of all of the node positions in the network. Each receiver then tracks the motions of the other nodes through time using the pairwise relative tracking vectors between the remote node and

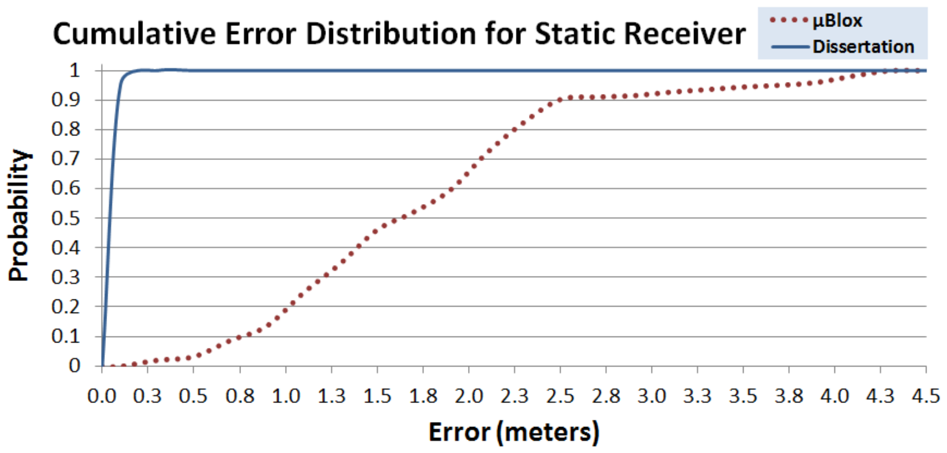
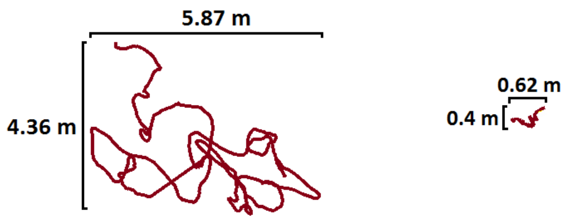


Figure 22: Cumulative error distributions for a stationary receiver

itself. The error is determined from the distance between the computed relative location at any point in time and the ground truth. Since the nodes are completely stationary, the ground truth is simply the starting location of each node.

The results shown in Figure 22 indicate the cumulative error distribution for one of the three remote nodes as viewed from an arbitrarily chosen reference receiver. Note that different choices of reference and/or remote node result in almost identical distributions, so only one of the 12 possible options is shown here. In every case, our algorithm produced very accurate results with mean errors ranging from 7.5-10.3 cm (with standard deviations of 4.9-6.8 cm). The errors from the built-in μBlox algorithms were significantly worse, with mean errors ranging from 1.46-1.82 m and standard deviations from 0.7-0.84 m. This is almost a 20-time improvement in average error, with an extremely substantial increase in stability as evinced by the low standard deviations. Figure 23, taken from an actual track experiment, shows what this type of improvement looks like graphically over a 25-minute period. Note that our tracking algorithm resulted in both significantly slower and smaller error accumulations than the chip-based solution; however, the errors from the μBlox algorithms appear to be zero-mean, whereas our errors indicate a slow bias over time.



(a) μBlox (b) Dissertation

Figure 23: Static tracks using the built-in μBlox algorithms vs. the tracking methodology introduced in this dissertation

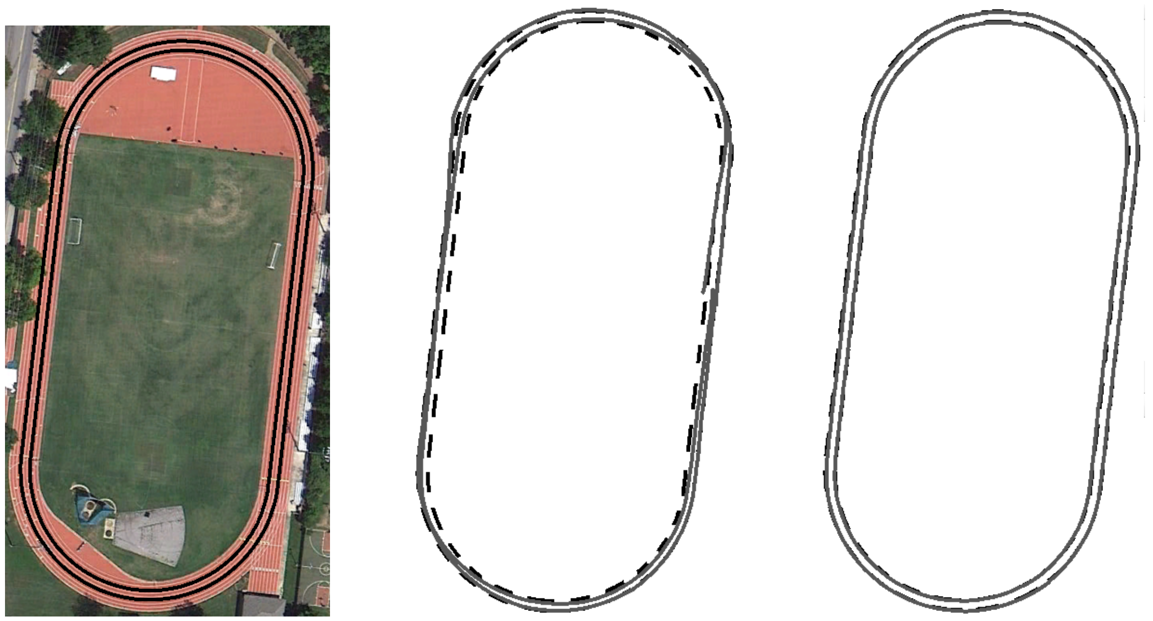

(a) Ground truth **(b)** μBlox **(c)** Dissertation

Figure 24: Tracks of two nodes separated by a constant 9-foot baseline making one lap around a running track

Fixed 9-foot distance

The results of the experiment with two mobile nodes on a 9-foot (2.74 m) pole and a single stationary node are probably the easiest to visualize. Figure 24a shows the ground truth of this experiment—the tracks of the two mobile nodes in Google Earth. Figure 24b shows the tracks estimated by the built-in μBlox algorithms using the stationary node as a reference, under the assumption that its precise absolute location is known. Finally, Figure 24c shows the tracks computed by the dissertation algorithms, indicating the relative location vectors between the stationary node and the two mobile nodes through time (and again assuming that the stationary node is perfectly localized and the starting positions are known). In other words, these results indicate the tracks of the two mobile nodes as seen from the reference with relative ranges varying from 0 to ~125 m. In the latter two figures, the ground truth is shown using a dashed line, and the background image is removed for clarity.

As the figures show, the tracks reported by our dissertation algorithms are generally closer to the ground truth. For example, the two nodes are always perpendicular to the track and finish almost exactly where they started. This is also apparent in Figure 25, which shows the estimated distance between the mobile node pair (as seen from the stationary reference) as it moved around the lap. Notice by comparing Figure 24b to Figure 25 that the tracks from the μBlox algorithms are not only inconsistent in their ability to stay in the correct lane position, but the nodes also frequently move in an orientation that is not perfectly parallel to the track, as indicated by the greater than 9-ft range errors in Figure 25 that correspond to track locations in Figure 24b at times when they visually appear to be *less* than 9-ft, and also in the starting locations of the two nodes in Figure 24b (with a range of close to 5 m as opposed to the expected 2.74 m).

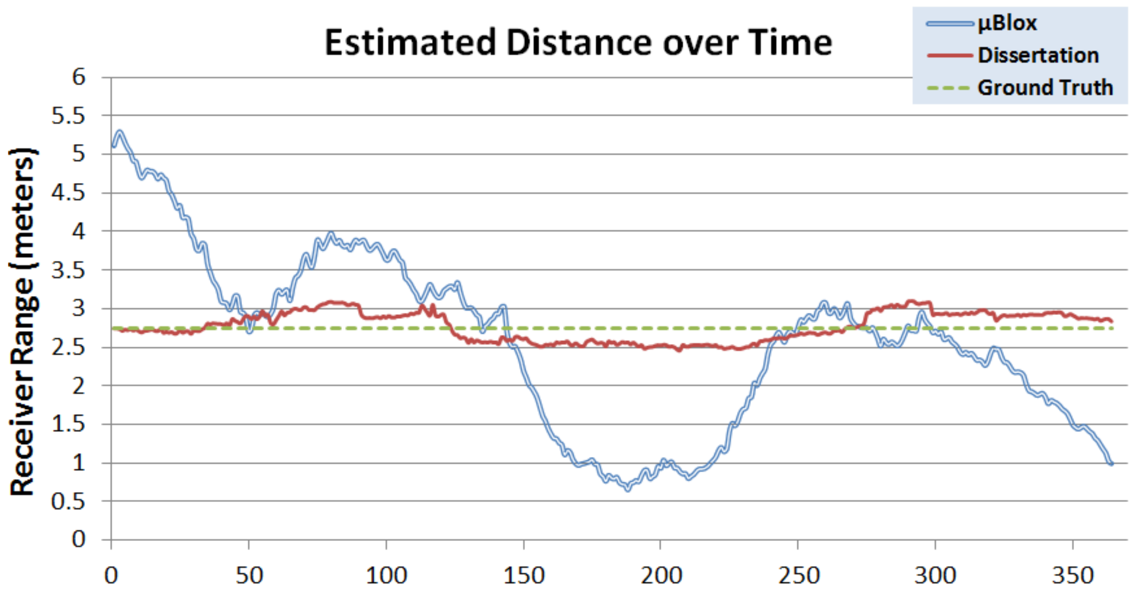


Figure 25: Estimated distance between two mobile nodes (as seen by a stationary node) as a function of time as they moved around the track

Knowing the ground truth between the two mobile nodes to be a fixed 9 feet, or 2.74 m, the mean error using the μBlox algorithms (as seen from the stationary reference node) was 89 cm with a standard deviation of 68.5 cm. Taking the *difference of the vectors* provided by our tracking algorithms between each mobile node and the reference, the mean error was 16.8 cm with a standard deviation of 8.4 cm.

Note that we can also track the relative distance of the two mobile nodes to one another without an explicit reference node. Each of the mobile nodes can consider itself to be the reference and track the relative location of the other. (This is the default operating mode of our tracking algorithms; the stationary reference node was only necessary to give results over changing baseline ranges and to show the absolute track in Figures 24b and 24c.) Results using this method provide a mean error of 14.2 cm according to both receivers, with a standard deviation of 7.8 cm. It is interesting to note that the relative errors are very similar whether we use a reference node up to 125 m away or just the two mobile nodes that are less than 3 m apart. In conclusion, this experiment showed a factor of 6 improvement over the built-in μBlox algorithms, with an almost 10-time improvement in stability as indicated by the standard deviation values.

9-foot equilateral triangle

This experiment was very similar to the previous one with the addition of an extra mobile node. Again, our approach is compared to that of the μBlox algorithms; that is to say, the value computed by taking the difference of the absolute positions reported by the three μBlox receivers involved and the stationary reference. The combined mean error of the μBlox algorithms for all three baselines was 1.12 m, while the mean error of our tracking algorithm was 23.2 cm, an overall improvement of 6.6x in mean error. The standard deviation of our tracking algorithm came to 17.6 cm, which was a 3.5-time improvement over the μBlox standard deviation of 63.2 cm.

Note that in this setup, each mobile node sees the other two at a 60° angle. Since our algorithms compute location vectors, we can estimate this angle easily. It is also straightforward to do with the absolute coordinates provided by the GPS chips. Figure 26 shows the distribution of all three angle estimates throughout the lap. These results show a marked improvement over the μBlox algorithms, with a clear spike in the distribution curve around the 60° mark. The standard deviation of our results was 8°, while the standard deviation of the built-in results was 33°. Note, however, that due to the close proximity of the receivers, a small error in range can result in significant angular errors.

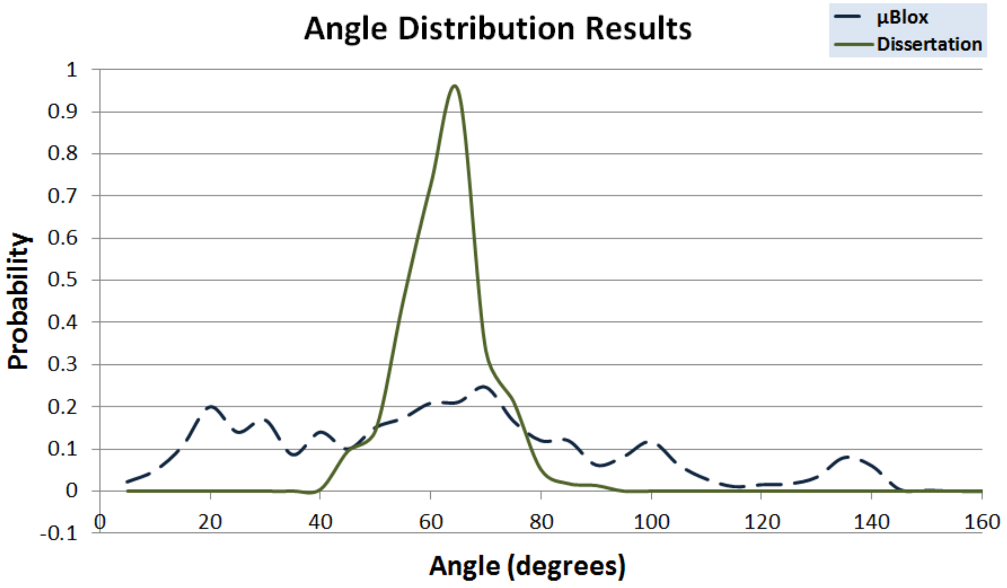


Figure 26: Relative angular distributions between three nodes in an equilateral triangle configuration for one lap around a track

Driving

Since multiple experiments in a fairly benign environment all produced marked improvements over the built-in μBlox algorithms, we decided to test the limits of our system by running experiments in both a partially-obstructed area and under high-dynamic conditions, namely driving in a car. This experiment involved placing three GPS nodes on top of a car and driving in a 12.2 km loop, where the course was broken up into two distinct parts. The first part involved driving at relatively low speeds (<50 km/h) between 2-4 story tall buildings and under numerous leafy trees; in other words, a slightly higher dynamic environment than in previous experiments with significantly more opportunities for multipath, cycle slippage, and losses of lock. This was followed by an extremely high-dynamic situation in which we drove on an interstate highway, where the main obstructions were the numerous overpasses that temporarily blocked satellite visibility. Finally, we exited the interstate and closed the loop to return back to our starting point.

In addition to the three mobile nodes on the roof of the car, we also set up a stationary node at the starting position to test the accuracy of our algorithm when a receiver is tracking multiple nodes many kilometers away (a maximum of 3.5 km in this case).

Driving in an Obstructed Area

Three nodes were placed on top of a vehicle situated in a wide open parking lot with a clear view of the sky, two in the front approximately 1.04 m apart and a third on the back left side approximately 1.35 m behind the front left node. After initialization, the car was driven slowly through a very narrow alleyway with buildings on either side and significant tree cover. The car then drove through a suburban-type road with occasional tree cover, followed by a main road (at approximately 50 km/h) with partial tree cover. The total length of this portion of the experiment was 1.478 km, which took 5 minutes to complete including a few stops due to traffic. Table 5 summarizes the results as they accumulated over time, where "Reference" indicates the pairwise node ranges as computed with reference to the stationary node, and "Direct" indicates the results as computed directly (pairwise) from each of the mobile nodes on the vehicle relative to one another, excluding the stationary node.

Somewhat surprisingly, the overall results were only marginally less accurate than those from the running track, even though this experiment included a much more difficult GPS environment. We would have expected multipath and losses of satellite locks to have more of an impact, but it is possible that the close proximity of the receivers to one another meant that any multipath effects were actually quite similar for all receivers, and therefore canceled out.

Method	**Distance Traveled**	**Mean Error**	**Standard Deviation**
Reference	500 m	15.2 cm	10.5 cm
Direct	500 m	50.7 cm	24.0 cm
μBlox	500 m	93.6 cm	36.5 cm
Reference	1 km	15.0 cm	10.2 cm
Direct	1 km	48.4 cm	24.1 cm
μBlox	1 km	97.5 cm	36.5 cm
Reference	1.48 km	25.3 cm	24.8 cm
Direct	1.48 km	62.2 cm	36.4 cm
μBlox	1.48 km	113.4 cm	55.6 cm

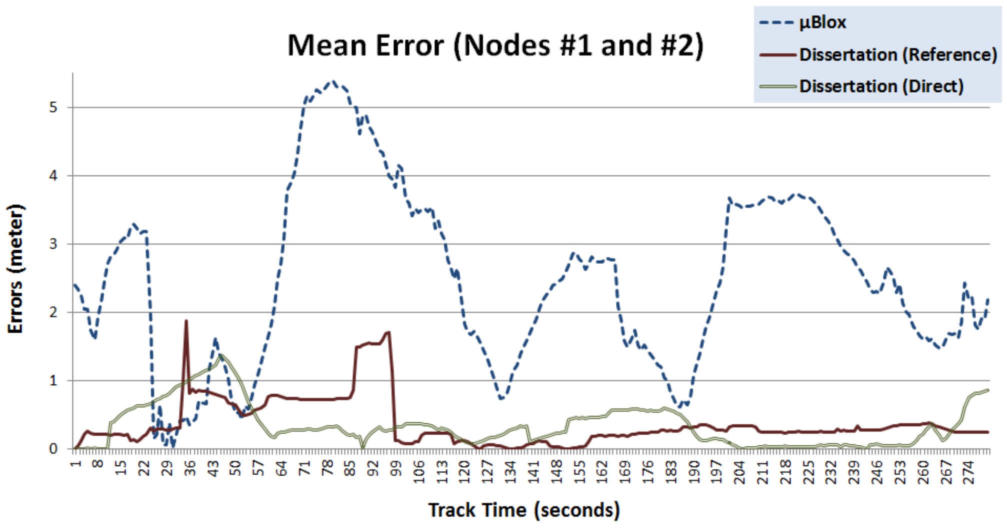
Table 5: Cumulative accuracy for one node pair while driving through a difficult GPS environment. "Reference" results indicate node ranges as computed with respect to a stationary reference node, and "Direct" results indicate ranges computed pairwise between the mobile nodes themselves.

It is interesting to note that the *direct* application of our tracking algorithms from each of the roving nodes to one another resulted in less accurate results than when viewed from the stationary node by a factor of about 3. The most likely reason for this is that the stationary node had a clear view of the sky, meaning it was more likely to have a larger number of visible satellites in common with each of the mobile nodes at any given time than they did with one another. In either case, our methodology outperformed the μBlox algorithms by a factor of 2 for the direct case, and a factor ranging from 4.5-6.5 when using the stationary node as a reference.

High-Speed Driving

For this portion of the driving test, we entered an interstate highway with a clear view of the sky, obstructed only by occasional overpasses. It is important to note here that we did not stop and re-initialize the tracking algorithm, but rather continued on from the end of the previous portion of the driving experiment. Therefore, any error that had accumulated during that time was also present to begin this test.

The total length of the path traversed during this experiment was 7.027 km, taking 4.75 minutes to complete. The receivers were moving at an average speed of 90 km/h during this time. The results for two of the mobile node pairs are summarized in Figure 27. Clearly, the improvement of our tracking methodology over the μBlox algorithms in the mean error graph for mobile nodes #1 and #2 looks significantly better than the results from nodes #2 and #3. We included both sets of results to emphasize an important point, namely to indicate how bias can accumulate in a dead-reckoning algorithm and to show why these tracking results should not be used exclusively in a standalone fashion.



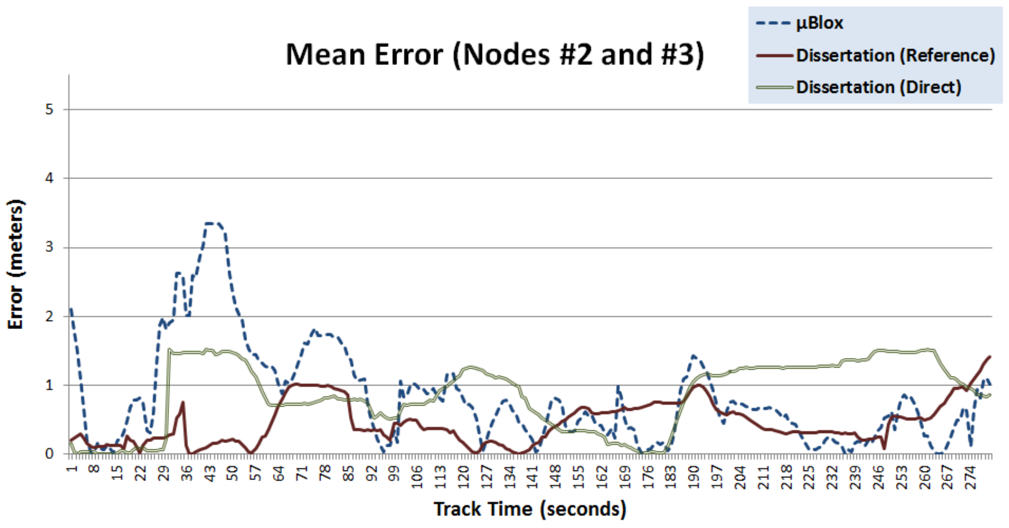


Figure 27: Mean tracking errors over time for two of the mobile node pairs attached to the roof of a car driving along the interstate

Nodes #1 and #2 correspond to the front two receivers on the car, which, from looking at both GPS tracks and analyzing the result data, accrued only a modest amount of error over the first part of the driving experiment. Node #3 was the node on the back of the car directly behind node #2, and it was apparent that this node had accumulated more error than the other two in the directional sense (in other words, the reported ranges between the nodes were quite close to the actual ranges, however the direction vectors from node #3 to the other nodes were slightly incorrect). As such, the tracking algorithm could have been working flawlessly and the comparison to ground truth would still show significant error since the initial positions at the beginning of this portion of the experiment had already accumulated error.

Since the initial positions of nodes #1 and #2 were shown to be more closely representative of the ground truth, the results in the first graph are more indicative of the performance of our tracking algorithm. Once again, it outperformed the μBlox receivers by a significant margin, only this time the direct localization approach (i.e. the positions of the mobile nodes as seen from one other, excluding the stationary node) performed slightly better than the stationary node case (as expected). This again furthers our theory that low satellite visibility in the first part of the experiment was to blame for the decrease in accuracy.

For numerical comparison, the mean error of the μBlox method was 2.47 m with a standard deviation of 1.29 m, whereas the mean error of the dissertation method using a stationary reference node was 37.6 cm with a standard deviation of 34.5 cm, and the mean error of the direct dissertation method was 34.8 cm with a standard deviation of 31 cm. As a frame of reference, this level of accuracy allows for quite obvious feature extraction of important driving events such as lane-changing as shown in Figure 28, approximately 1 km into the experiment:

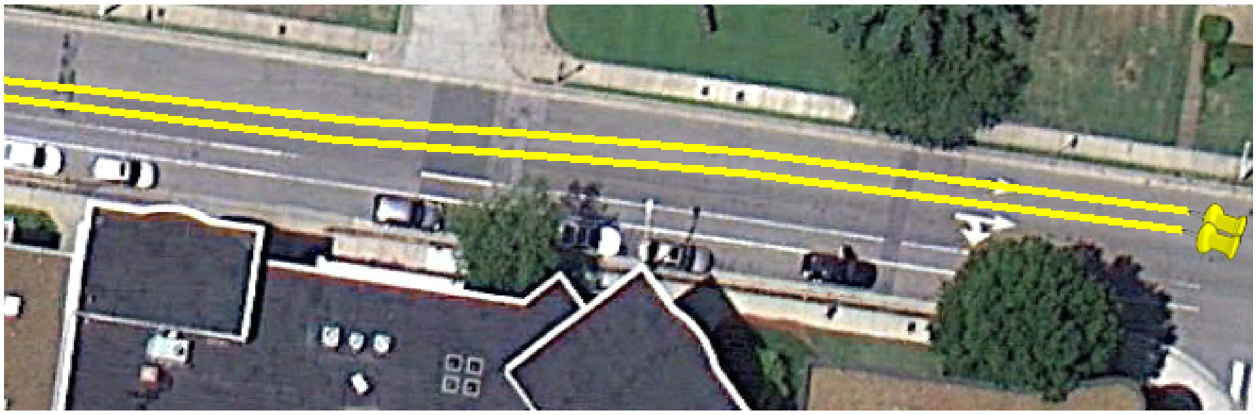


Figure 28: Track of car carrying out a lane change

Closing the Loop

At the very end of the interstate experiment, our vehicle passed under a wide overpass while at the same time subtly changing directions. Also, each of the nodes re-acquired satellite locks at slightly different times. As a result, the motion model used to extrapolate tracking results during periods of low satellite visibility failed in this case, with the three mobile receivers emerging from the overpass with tens of meters of distance between them as seen from the stationary reference node. Recall that the motion model used for extrapolation is quite simplistic, so this sudden increase in error is more indicative of the limitations of that model than it is of the tracking algorithm itself.

Fortunately, one node was able to re-acquire its satellite locks quickly enough to have only accrued a minimal amount of error, enabling us to continue to track its position relative to the stationary node as we closed the loop and returned back to the starting position. This required an additional 3.696 km of driving over 5.2 minutes. At this point, the remaining node appeared 2.3 m away from its starting position. While this number by itself is quite impressive, considering our tracking methodology relies on dead reckoning and the entire experiment lasted 15 minutes and covered 12.2 km of moderate to difficult GPS terrain, we expect that the result may have even been better had the final overpass not temporarily disabled the algorithm.

One final important thing to note about this driving experiment is the level of accuracy that was able to be maintained relative to a stationary reference node, even though the range to that node varied up to 3.5 km from the starting point. This shows that our error analyses and the dual-receiver corrections used in our tracking algorithm produce precise enough satellite observations that accuracy is largely independent of the pairwise distances between the mobile nodes being tracked. As a summary, Table 6 shows the results of all experiments described in this section, with the high-speed driving results ending just before passage underneath the problematic overpass.

Experiment	**Method**	**Mean Error**	**Standard Deviation**
Track: *Stationary*	Reference	9.1 cm	6.1 cm
	μBlox	168.4 cm	77.3 cm
Track: *9-ft Pole*	Reference	16.8 cm	8.4 cm
	Direct	14.2 cm	7.8 cm
	μBlox	89.0 cm	68.5 cm
Track: *Triangle*	Reference	23.2 cm	17.6 cm
	μBlox	112.0 cm	63.2 cm
Driving: *Obstructed/ Multipath*	Reference	25.3 cm	24.8 cm
	Direct	62.6 cm	36.4 cm
	μBlox	113.4 cm	55.6 cm
Driving: *High-Speed*	Reference	37.6 cm	34.5 cm
	Direct	34.8 cm	31.0 cm
	μBlox	247.0 cm	149.0 cm

Table 6: Summary of tracking results for all nodes in each experiment. "Reference" results indicate node ranges as computed with respect to a stationary reference node, and "Direct" results indicate ranges computed pairwise between the mobile nodes themselves. Note that in the stationary track experiment, the Reference and Direct methods of evaluation are synonymous.

Short-Term Tracking

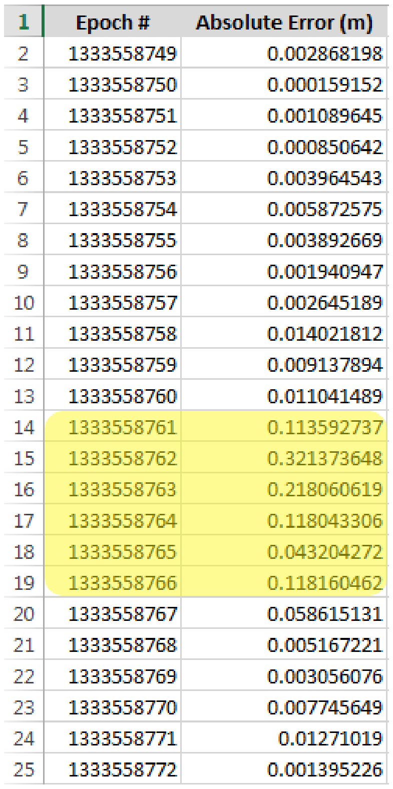
While the results of the tracking experiments show a marked improvement over existing algorithms in a dead-reckoning sense, the benefits of this technique are even more apparent when viewed in the context of the baseline localization algorithm described in "Section: *Relative GPS Localization Algorithm*" in Chapter IV. We discussed that our new APT solution algorithm makes use of single-epoch tracking results to overcome the typical hurdles of motion-based initialization and calibration. We have seen in this chapter that even highly accurate tracking results can suffer from an accumulation of bias over the long-term; however, it has already been discussed how APT uses an extended AFM technique with hill-climbing to remove this bias at each time step, such that it should fail to accumulate over time.

As such, the significance of these tracking results stems from their accuracy over single epochs of measurement. From the experiments described in the previous sub-section, the single-epoch tracking accuracy of the stationary running track experiment is trivial to compute (i.e., it should change by 0 m at every time step). Likewise, in both the 9-ft. pole and triangle running track experiments, as well as both portions of the driving experiment, the direct distance between the mobile nodes remains fixed with only the direction changing; therefore, the single epoch accuracy can be deduced from any range deviations per unit time. Finally, for estimation of the single-epoch tracking error for changing baselines through time, we will have to rely on analyzing the range deviations between two or more remote nodes when the baseline between them is computed relative to a stationary reference, or in other words, by subtracting the position vectors from a single stationary reference node to two remote nodes from one another, thus computing the direct range vector between the remote nodes *as seen from the stationary reference.*

Experiment	**Method**	**Mean Error**	**Standard Deviation**
Track: *Stationary*	Reference	1.1 cm	1.3 cm
Track: *9-ft Pole*	Reference	2.1 cm	2.3 cm
	Direct	2.9 cm	3.4 cm
Track: *Triangle*	Reference	4.9 cm	6.6 cm
	Direct	4.7 cm	5.4 cm
Driving: *Obstructed*	Reference	7.8 cm	15.2 cm
	Direct	6.4 cm	12.0 cm
Driving: *High-Speed*	Reference	4.3 cm	12.7 cm
	Direct	3.9 cm	7.5 cm

Table 7: Summary of single-epoch tracking errors for all nodes in each experiment. "Reference" results indicate node errors as computed with respect to a stationary reference node, and "Direct" results indicate range errors computed pairwise between the mobile nodes themselves.

Table 7 shows the average single-epoch tracking error for each of the aforementioned experiments when calculated in this way. The sub-decimeter results seem quite good, especially given the cumulative tracking errors in Table 6 which indicate that the majority of single-epoch errors must be close to zero mean when aggregated over time; however, an analysis of the individual epoch-by-epoch results indicates that the total error distribution is not Gaussian. In fact, very clear *regions* of error can be discerned amidst a vast number of results containing only millimeters of single-epoch tracking error. Figure 29 shows an example of this phenomenon:



1	Epoch #	Absolute Error (m)
2	1333558749	0.002868198
3	1333558750	0.000159152
4	1333558751	0.001089645
5	1333558752	0.000850642
6	1333558753	0.003964543
7	1333558754	0.005872575
8	1333558755	0.003892669
9	1333558756	0.001940947
10	1333558757	0.002645189
11	1333558758	0.014021812
12	1333558759	0.009137894
13	1333558760	0.011041489
14	1333558761	0.113592737
15	1333558762	0.321373648
16	1333558763	0.218060619
17	1333558764	0.118043306
18	1333558765	0.043204272
19	1333558766	0.118160462
20	1333558767	0.058615131
21	1333558768	0.005167221
22	1333558769	0.003056076
23	1333558770	0.007745649
24	1333558771	0.01271019
25	1333558772	0.001395226

Figure 29: Region of erroneous results in an otherwise stable tracking environment

There are four causes for these types of anomalies:

1. The number of non-erroneous satellite measurements as determined by the tracking algorithm has reached a critical level of 4, meaning the system is no longer over-determined,

2. The number of satellite observations is less than 4, meaning the tracking results are being determined by a motion model instead of the tracking algorithm itself,

3. One or more undetected cycle slips were present in the tracking observations, or

4. The level of multipath during the error-prone time frame was large enough (and affected enough observations) that it translated into a comparable error in the tracking result.

These regions of error often manifest with a noise-like quality (i.e. they tend to be close to zero mean over the long term), and thus, have minimal impact on the tracking algorithm when used in a dead-reckoning framework; however, it is possible for them to be problematic in the APT localization algorithm if not detected and mitigated as much as possible, as "Section: *Ambiguity Function Method Overview*" in Chapter IV demonstrated that even very small tracking inaccuracies can result in a low AFV. Fortunately, the first two causes of inaccuracy can be determined by the tracking algorithm itself and passed to the APT algorithm for special handling. The latter two causes, however, are currently undetectable and will require additional processing and error handling in subsequent baseline localization procedures.

Baseline Localization Results

Due to the typically high accuracy of the single-epoch tracking results from the previous sub-section, we were able to use the results directly as the requisite inputs to the APT baseline solution procedure as described in Algorithm 3. The remainder of this section discusses the results from the same set of experiments and test configurations introduced earlier in the chapter, with the data logs being re-processed by the complete localization system, including all modules, algorithms, and processing steps described in Chapter V.

Running Track

Recall that the setup for this experiment consisted of a stationary node positioned at one corner of a standard running track with a series of mobile nodes roving in various configurations around the track. In all of the experiments in this section, the initial search space was defined to be an 8 m^3 cube, centered on the estimated baseline reported by the GPS receivers themselves. Note that in several experiments, the error in the reported μBlox baseline caused the correct solution to lie *outside* of the 8 m^3 search space. In these cases, an arbitrary initialization point was selected inside of the search space for purposes of evaluating the system; however, in real-world scenarios, either a larger initial search space must be used or improved initial baseline determination methods must be employed (note that many such methods exist, they were simply not utilized in the test prototype for this dissertation).

Stationary Setup

In this experiment, four stationary nodes were placed at the corners of a track, forming a 94x68m rectangle. Since no motion took place, the results are useful for indicating the worst-case calibration times of the system (since receiver motion causes a change in geometry that is beneficial in speeding up the initial baseline estimate). Likewise, since multipath and cycle slips are unlikely to occur, this represents the most benign scenario in terms of dealing with uncorrectable errors.

In this experiment, one of the four stationary nodes kept going in and out of service, and thus, was never able to be successfully calibrated in the first place by the three remaining nodes. As such, it is omitted from the table of results. It should be noted, however, that at every time instant prior to the receiver becoming unavailable, the ambiguity function "peaks" corresponding to the correct baselines from each of the three additional nodes were always present, but sufficient time had not passed for the APT baseline algorithm to filter out the remainder of the incorrect or unfit solution candidates.

Table 8 presents the steady-state results of this experiment in terms of baseline deviations from the known range between pairs of individual nodes over the course of approximately 18 minutes. The low error residuals and standard deviations corresponding to the first two node pairs indicate that the initial baseline calibration was successful and remained steady through time. The results from the third node pair, however, including its unreasonably short calibration time compared to the other nodes, seem to indicate that a slightly incorrect peak was selected during calibration as the correct baseline. While the average error from this faulty detection was quite low, the standard deviation value indicates that the baseline tends to drift (albeit extremely slowly) through time. In such a case, the peak filter would have eventually determined that its current estimate of the relative baseline was incorrect (via consistent lowering of the associated AFV) and re-initialized itself to obtain a better solution.

Node Pair	**Mean Error**	**Standard Deviation**	**Calibration Time**
#1 and #2	2.7 cm	1.8 cm	245 s
#1 and #3	4.1 cm	1.3 cm	179 s
#2 and #3	9.0 cm	7.1 cm	36 s
TOTAL:	5.0 cm	4.6 cm	—

Table 8: Mean errors and calibration times for baseline solution of stationary node pairs

In all cases, notice that the mean errors using the full baseline localization algorithm remain lower than the mean errors from the same experiment when run in a tracking-only mode (see Table 6). Even more importantly, the standard deviations of the errors from the node pairs with successful baseline initializations dropped to negligible single centimeters over the course of the experiment. This indicates that the hill-climbing algorithm is successfully removing any tracking bias from the solution before it can accumulate into a measurable amount of baseline error.

Finally, note that the calibration times from all nodes in this experiment average 153 s. This value is likely to be on the low side since the third node pair obviously latched onto an incorrect peak in an extremely short amount of time. From this, as well as additional stationary localization experiments, a more likely average calibration time would lie somewhere between 3 and 4 minutes. We expect to see this number decrease substantially once the receivers begin moving relative to one another during the calibration phase.

Fixed 9-foot Distance

As in the stationary experiment, a node is placed at one corner of the running track; however, in this setup, two additional receivers are connected to a fixed 9-ft. pole which is walked around the track in a configuration perpendicular to the running lanes. The following graph, Figure 30, shows the estimated baseline between the mobile receivers traversing a lap around the running track as seen by the stationary reference node after its successful calibration of the relative baselines to both of the remote nodes.

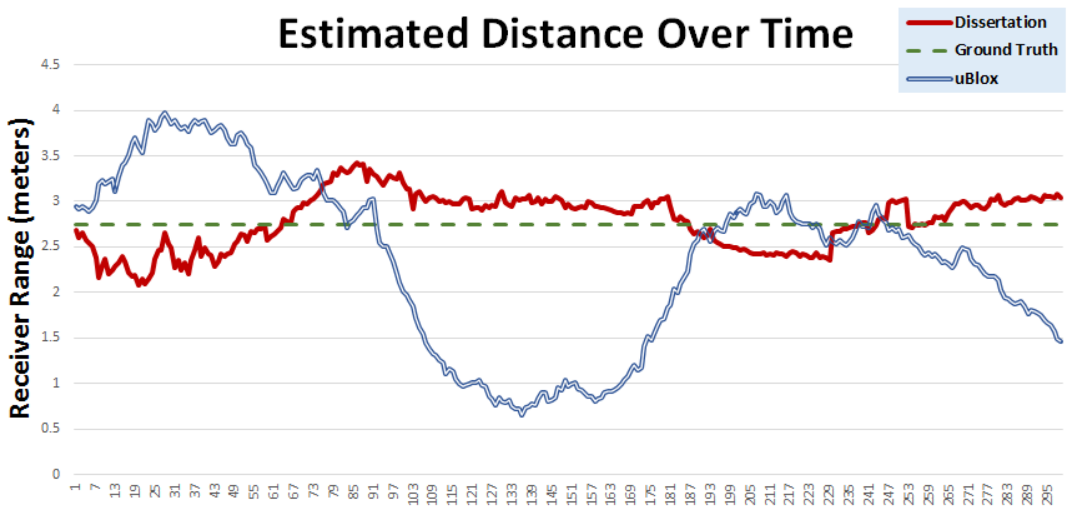


Figure 30: Estimated distance between two mobile nodes (as seen by a stationary node) as a function of time as they moved around the track

Figure 31 on the next page shows the same results as computed pairwise directly between the two mobile nodes.

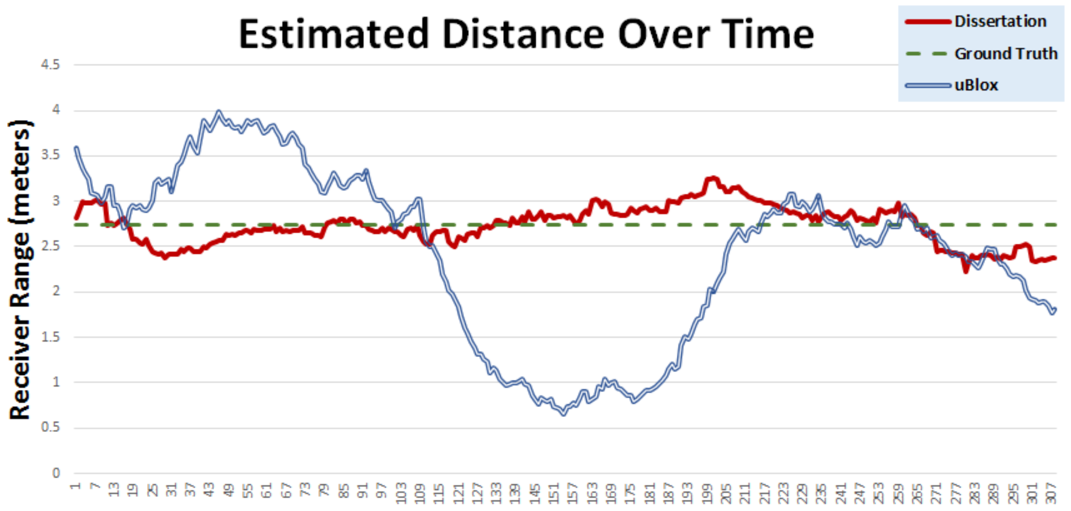


Figure 31: Estimated direct distance between two mobile nodes as a function of time as they moved around the track

In both cases, the results from this experiment indicate a similar level of error to that of the tracking-only experiment, although the standard deviations of the error are approximately doubled. Specifically, the mean error for the relative baseline as seen from the stationary reference node was 26.2 cm with a standard deviation of 15.3 cm. The mean error for the direct baseline solution was 16.9 cm with a standard deviation of 12.3 cm. While these results indicate a significant improvement over the μBlox average of 89 cm of error, we would have expected to see an improvement over the dead-reckoning results from the tracking-only solution as well.

One reason for the slight decrease in accuracy and stability is an overly pessimistic view of the current baseline solution by the APT filter due to the arbitrary regions of abnormally large tracking errors described in the previous sub-section. These tracking errors cause our lock on the peak corresponding to the current baseline solution to be lost, such that the filter begins tracking a nearby peak with a much lower AFV than the "correct" peak. In this case, the peak will quickly fade below the cutoff threshold, causing the filter to re-enter its calibration stage to search for new peak candidates. Since the most recently tracked peak in this case will have corresponded to an incorrect baseline, the error will be proportionally larger during both the erroneous steady-state and subsequent calibration phases until a new, correct peak is identified.

Even with these regions of uncertainty, the relative localization procedure provides a 6-7x improvement over the μBlox solution, which is clearly visible in the previous two figures. Additionally, the algorithm's ability to re-initialize itself on the fly provides a valuable asset for those situations when tracking alone becomes unreliable.

The times spent in the calibration phase for this experiment were significantly shorter than in the stationary case, as expected. The transition from calibration to steady-state localization in this experiment when viewed from the point of view of the stationary node was 38 s, which as a frame of reference, occurred after only 12.64 m of receiver motion. When the baseline was computed directly by the two nodes in motion, the steady-state phase was reached after 72 s of calibration. The reason for the longer calibration time in the direct baseline solution is most likely due to the simple fact that only the baseline *directionality* changes in this case, whereas both the directionality and the *range* are changing from the point of view of the reference node. In either case, these calibration times on the order of a minute indicate that this method may find utility in applications requiring very rapid relative position determination.

9-foot Equilateral Triangle

In the final running track experiment, three nodes were placed in an equilateral triangle configuration, 9 feet apart from one another, and walked in a lap around the track. An additional stationary reference was placed at the corner of the track to provide results over changing baselines. As with the previous running track experiment, this setup showed improvement over the corresponding μBlox solution, as indicated in the following table of range errors and calibration times. Note that the calibration times shown here appear to be longer than those for the previous 9-ft. pole experiment. This is a misleading result in that we spent a considerable number of seconds standing in place at the beginning of this experiment, whereas we did not in the previous one. This means that the portion of calibration time spent during periods of relative receiver motion is substantially shorter than illustrated in the following table.

Node Pair	**Mean Error**	**Standard Deviation**	**Calibration Time**
#1 and #2 (Ref.)	40.9 cm	17.0 cm	101 s
#1 and #3 (Ref.)	31.9 cm	22.6 cm	93 s
#2 and #3 (Ref.)	27.3 cm	15.8 cm	101 s
Combined (Ref.)	33.6 cm	19.5 cm	—
#1 and #2	37.7 cm	46.7 cm	126 s
#1 and #3	67.7 cm	69.8 cm	133 s
#2 and #3	92.9 cm	50.9 cm	134 s
Combined	65.0 cm	60.3 cm	—
μ**Blox**	112.0 cm	63.2 cm	—

Table 9: Mean errors and calibration times for baseline solutions of triangle node pairs. "Reference" pairs indicate baselines as computed with respect to a stationary reference node.

These results indicate a 3.3-time improvement over the μBlox solution when the baselines between the three remote nodes are calculated with respect to a reference receiver, and a 1.7-time improvement when computed directly pairwise to one another. As with the previous experiment, these results represent an improvement over the μBlox solution, but a degradation compared the tracking-only solution which experienced an 8.5-time improvement over the μBlox results in the previous sub-section. Again, the reason for this is simple: slight tracking inaccuracies cause a loss of lock on the correct baseline peak which results in larger range offsets during times of filter re-initialization. However, this decrease in accuracy is a tradeoff for system automation. While accuracy may not be as high as with the tracking-only architecture, the APT baseline solution provides the added advantage that a user no longer has to explicitly define the starting baselines between all node pairs in the system in order for the localization results to have meaning.

While the baseline results obtained using a stationary reference node are straightforward and indicate no significant anomalies in the filtering methodology, the mean errors from the direct results deviate from one another quite substantially. By analyzing the raw epoch-by-epoch data, it is clear that several regions of dramatically incorrect tracking results cause the filter to begin tracking erroneous peaks that are quite a long distance from the correct baseline. This is also illustrated in the physical tracks corresponding to the pairwise baselines as computed by each of the remote nodes in Figure 32. Note that since the receivers are configured in a set triangle configuration and walked 360° (counterclockwise) around a lap, the corresponding baseline tracks should be perfect circles in all cases (although in reality, we stopped walking slightly short of the lap starting position, so the gaps between the start and end points in the tracks do not represent localization errors).

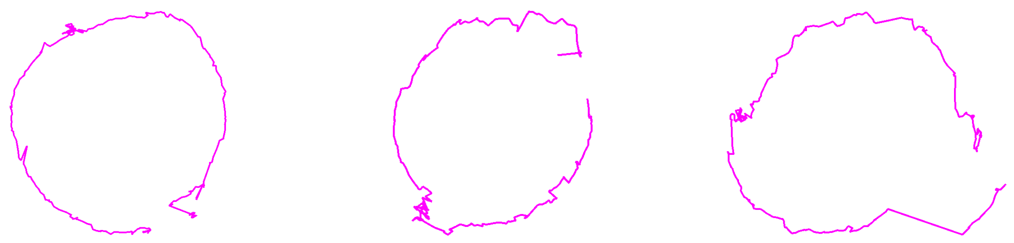
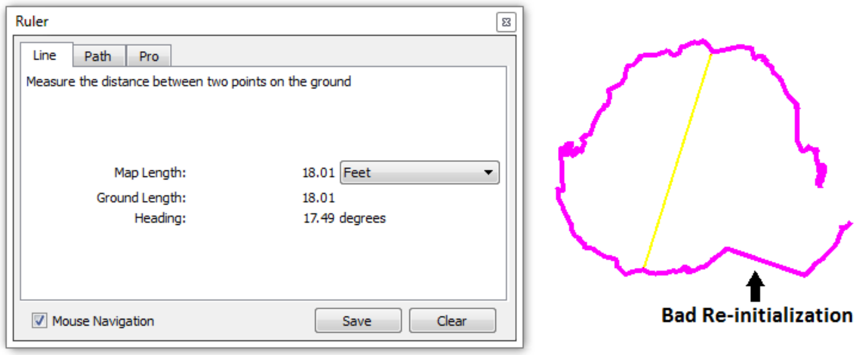

(a) Nodes #1 and #2 **(b)** Nodes #1 and #3 **(c)** Nodes #2 and #3

Figure 32: Direct pairwise baseline tracks between roving nodes in a triangle configuration

The baseline track of nodes #1 and #2 in Figure 32a looks quite good, and this is reflected in the corresponding mean error value in Table 9. Additionally, it is easy to see the physical location where the APT filter transitioned from calibration to steady-state, as indicated by the "jump" in the track to the correct location on the circle. The track of nodes #1 and #3 shows similar regions of good localization results; however, a tracking error occurred as the receivers reached the portion of the experiment corresponding to the lower left-hand corner of Figure 32b, causing a clear jump to an incorrect baseline location which fails to re-initialize throughout the remainder of this experiment. Finally, Figure 32c shows the clearest example of the result of an incorrect peak lock.

The following Figure 33 indicates specifically where this erroneous re-initialization occurs and shows that, prior to the bad tracking results, the localization filter was performing quite well, with a circle radius of approximately 18 ft, exactly as expected:



Measure the distance between two points on the ground

Map Length:	18.01 Feet
Ground Length:	18.01
Heading:	17.49 degrees

[✓] Mouse Navigation

Figure 33: Baseline localization track before and after a "bad re-initialization"

One of the reasons that the results obtained from the stationary reference node are more accurate than the direct baseline results is due to the fact that the satellite measurements themselves were highly error-prone in this experiment. For an unknown reason, the observations contained far more cycle slips than would be expected, and the average satellite signal strengths tended to remain at the lower end of the acceptable range. As such, any additional observation errors introduced into the system, most notably caused by multipath due to the close proximity of the mobile receivers to our bodies, caused many of the measurements at each epoch to become unusable. The stationary node's clear view of the sky and relatively multipath-free environment enabled it to make better use of the remote observation data than the remote nodes themselves.

Even with the low quality of satellite data during this experiment and the moments of seemingly poor baseline initialization values, our methodology was able to outperform the μBlox algorithms by a substantial amount. The next set of driving experiments gives a much better example of the *utility* of this type of localization algorithm and shows why it may be preferable to the tracking-only solution, even at the cost of slightly degraded accuracy.

Driving

For the experiments in this section, recall that our setup included three mobile nodes mounted to the roof of a car with a single stationary node placed in a parking lot at the starting site of the driving trial. The primary purpose of these experiments was to test the accuracy of our methodology under high dynamic situations and at increasing baseline lengths.

Driving in an Obstructed Area

For comparison with the tracking-only results, the driving course was again separated into two distinct parts, the first of which included driving in multipath-rich alleyways and on suburban roads for a total of 1.48 km. In the case that the relative baseline was determined using a stationary reference node, the results from the APT solution algorithm were an order of 2 times *less* accurate than those from the tracking-only solution, but 3 times *more* accurate than the μBlox results. On the other hand, direct localization produced results that had similar or even slightly better accuracies than the tracking-only solution, with cumulative errors as shown in Table 10 for one of the pairs of remote nodes over increasing intervals of 500 m.

Method	Distance Traveled	Mean Error	Standard Deviation
Reference	500 m	39.1 cm	25.5 cm
Direct	500 m	58.3 cm	28.7 cm
μBlox	500 m	93.6 cm	36.5 cm
Reference	1 km	34.7 cm	25.2 cm
Direct	1 km	34.0 cm	32.6 cm
μBlox	1 km	97.5 cm	36.5 cm
Reference	1.48 km	36.5 cm	23.4 cm
Direct	1.48 km	39.7 cm	37.5 cm
μBlox	1.48 km	113.4 cm	55.6 cm

Table 10: Cumulative localization accuracy for one node pair while driving through a difficult GPS environment. "Reference" results indicate node ranges as computed with respect to a stationary reference node, and "Direct" results indicate ranges computed pairwise between the mobile nodes themselves.

These results indicate a quantifiable improvement over the μBlox algorithms, but an unexpected lack of substantial increase in accuracy from the tracking-only results in Table 5. Once again, the number of filter re-initializations was higher than expected and corresponded most commonly to the times when the vehicle had a severely obstructed view of the sky, or in other words, when driving very close to buildings in a narrow alleyway or under leafy trees. As such, it appears that multipath plays a much greater role in the ability of our APT baseline filter to maintain a lock on the correct peak than initially anticipated. Nonetheless, this portion of the experiment ended with all three of the remote nodes in a configuration containing less than 1 meter of error each from the correct ground truth position.

The transition times from calibration to the steady-state phase of the APT localization algorithm were 31 s, 25 s, and 50 s respectively for baseline determination of each of the mobile nodes as calculated by the stationary reference. The average calibration time for direct baseline determination between the roving nodes themselves was 90 s, almost 3 times as long as the average for the reference node. Again, the reason for this dichotomy in calibration times is due to the fact that the baseline *ranges* between the stationary node and the vehicle nodes are changing quite rapidly, whereas the ranges between the vehicle nodes themselves are not changing at all. The decrease in time spent in the calibration phase for the stationary receiver is consistent with our expectation that increased relative motion will result in decreased calibration times.

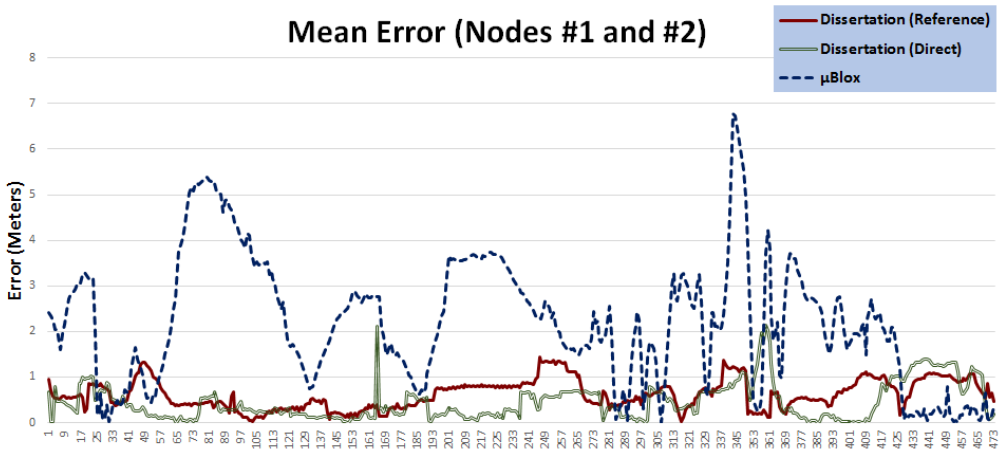
High-Speed Driving

At this point, we turned onto an interstate highway and began driving at an average speed of 90 km/h over an additional 7.027 km of road. For the duration of this experiment, all receivers had consistently clear views of the entire sky (with the exception of an occasional overpass). As such, we expected relatively few tracking errors due to multipath and a correspondingly low number of APT filter re-initializations.

Compare the errors from the dissertation algorithms to those obtained from the μBlox chip in Figure 34 and the increased precision of our localization technique becomes apparent, especially in terms of the baseline between nodes #2 and #3 in which the instantaneous localization errors rarely exceed 50 cm. Additionally, the number of times when the accuracy of the μBlox solution over multiple epochs is better than our own is very small, most notably at the very end of the experiment for nodes #1 and #2.

As with the tracking-only results, this increase in accuracy over the μBlox results is substantial, and we believe the accuracy could have been even higher if not for the overpasses, which almost always caused the localization filter to require re-initialization. In fact, many of the regions of decreased accuracy between nodes #1 and #2 began at times corresponding to the vehicle passing underneath an overpass (note that error spikes in the graph for nodes #2 and #3 seem to be more random, having little correlation with environmental obstacles at corresponding times).

For comparison, the mean range error averaged over all remote nodes on the car when viewed from a reference receiver in the parking lot was 49.9 cm with a standard deviation



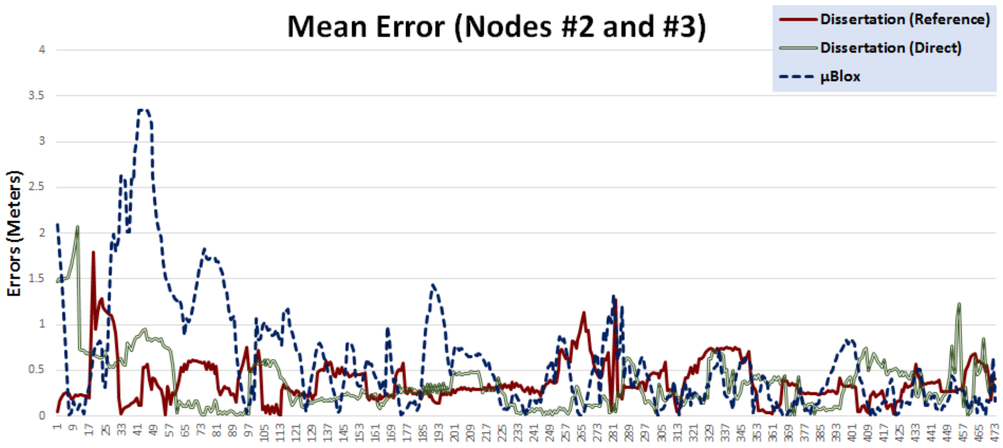


Figure 34: Mean localization errors over time for two of the mobile node pairs attached to the roof of a car driving along the interstate

of 30.4 cm. This represents a 5x improvement over the μBlox solution in terms of both error residuals and stability as evinced by the standard deviation. Likewise, the mean error when the relative baselines are computed directly between the roving nodes themselves was 39.9 cm with a standard deviation of 35.6 cm, almost identical to the results obtained from the tracking-only solution.

Closing the Loop

You will recall from the tracking-only results presented earlier in this chapter that we encountered a problem at the very end of the experiment, namely that we passed underneath a wide overpass while changing directions, causing incorrect tracking updates due to a momentary loss of satellite locks. One of the primary benefits of our APT localization approach is that the baseline filter quickly destabilizes as the AFV of erroneous peaks drops below a threshold value. In this experiment, this occurred when we traveled underneath the final overpass and experienced measurable amounts of tracking error. At this point, the filter attempted to re-initialize itself with a better baseline solution, which in the case of this experiment, proved successful.

The following figure shows a track of the entire experiment, including a zoomed-in snapshot of the starting and ending points, indicating minimal error over the course of the entire experiment. The physical distance between the start and end points of the node track pictured below was 1.54 m; however, we failed to mark the starting point of the experiment, so it was difficult to gauge the precise location at which we should stop driving. As such, it is highly likely that a large portion of this offset was due simply to misalignment of the car at the end of the experiment.

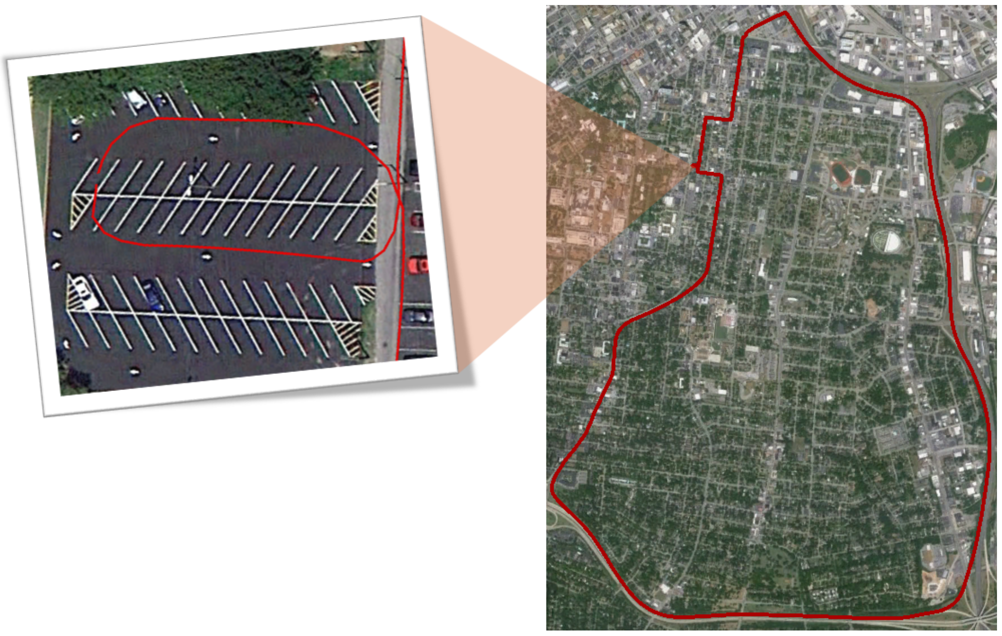


Figure 35: Track of the full driving experiment with emphasis on the starting and ending points of the course

This figure shows the usefulness and benefits of the APT baseline localization algorithm over the tracking-only solution. In cases where substantial amounts of satellite outage are likely to be experienced, the tracking solution, while accurate, will always require human intervention whenever re-initialization is required. The full APT technique, however, allows the system to re-initialize itself, thereby providing a "hands-off" automatic solution in which the system can quite literally be turned on and left to manage itself.

CHAPTER VII

CONCLUSIONS

This dissertation presented a novel approach to GPS-based differential localization of mobile nodes, with the overarching goal of dramatically increasing the precision of relative 3D baseline coordinates using only low-cost, off-the-shelf GPS receivers. In this chapter, we explore the meaning of the work presented in this dissertation, including summarizing the key results achieved by the research, commenting on the contributions the research has made to the field of outdoor relative localization technologies, analyzing the limitations of the system in its current state, providing recommendations for future areas of improvement, and examining some of the broader impacts of the progress made.

Findings and Contributions

By allowing a network of GPS receivers to share their raw satellite measurements with one another, we were able to achieve centimeter-scale relative localization accuracy via:

- Better modeling and correction of error sources unique to the two-receiver localization case, stemming primarily from ill-synchronized local receiver clocks,
- Creation of a new observation model called the *Temporal Double-Differencing* model which is able to use independent satellite observations from two receivers to create a set of unambiguous, highly-accurate carrier phase equations representing the relative motion of a pair of receivers through time and space,
- Development of a tracking algorithm that produces pairwise 3D location vectors between a local node and any number of remote receivers without requiring a reference node, reference satellite, stationary setup or calibration phase, or any *a priori* knowledge of the locations of one or more of the participating receivers,
- Use of highly accurate tracking results in a robust algorithm that allows for motion-enabled, unsurveyed receivers to determine the relative baselines between them in an impromptu, ad-hoc fashion, and
- Synthesis of the aforementioned research results in a working prototype on mobile Android devices for testing and validation.

Unlike most GPS-based navigation solutions, our approach does not snap positions to maps or try to use a dynamic model for the motions of the receivers (other than for tracking during satellite losses of lock). Also unlike other high-precision GPS localization techniques used in applications for which our system may have utility (e.g. Differential GPS, Real-Time Kinematic navigation, or any number of post-processing methods used in applications such as land surveying, precision agriculture, temporal feature extraction, or autonomous driving), our approach does not require a stationary calibration phase, relying instead on a symbiotic feedback relationship between relative tracking and baseline localization to provide real-time coordinate updates.

Prior to investigation into the aforementioned research topics, a list of requirements was created that represented the ideal characteristics a complete relative localization system should embody. This list included:

- Centimeter-precision accuracy for a scalable network of nodes,
- No *stationary* or *explicit* data collection phase, and
- No explicit reference station or hub.

Based on the results of the research presented in this dissertation, all three of these requirements have been met. Our technique allows any GPS receiver to become its own reference, thus negating the need for an explicit "reference station." Likewise, we employ observation models which either do not require a reference satellite or impose no limitations on which satellite must be used as a reference (or for how long). By incorporating tracking results into our baseline determination algorithms, there is no longer the need to pre-survey receiver locations prior to application deployment or for any receiver in the network to remain stationary during calibration; as such, our methodology provides a simple out-of-the-box, ready-when-deployed solution to high accuracy relative localization. Finally, our combination of techniques for dual-receiver error correction and continuous tracking through time has enabled localization algorithms to achieve levels of precision that are limited primarily by the measurement capabilities of the individual GPS receivers themselves.

Summary of Experimental Results

The research presented in this dissertation leverages an incremental approach to relative localization, whereby distinct challenges to GPS-based positioning are handled compartmentally in a linear fashion. Specifically, raw satellite observations are passed through a series of "modules" in a framework, where the results of data manipulation in one module directly affect the ability of subsequent modules to perform their respective tasks. As such, the same experiments were repeated twice, first to examine the effectiveness of our novel tracking algorithm, and then again to examine the accuracy of the complete localization methodology that depends on the precision of the tracking results.

In three different walking experiments, we were able to achieve tracking improvements ranging from 7-20x better than the corresponding "standard" GPS positioning techniques, and sub-meter accuracy was likewise achieved while driving at various speeds, with varying baseline lengths, and under difficult GPS conditions. In fact, the worst-case error encountered in the tracking-only solution was 62.6 cm corresponding to direct pairwise tracking of the roving nodes in an obstructed and multipath-rich driving experiment. Excluding this result, the next worst mean error was only 37.6 cm, with the average error over all experiments equalling 27.95 cm. The demonstrated high-accuracy of our tracking-only technique is notable because the approach uses a form of dead reckoning in which errors accumulate over time.

When the same experiments were repeated using the complete localization system, we achieved baseline accuracies that were once again substantially improved over standard positioning techniques by a factor of 1.7-33x, depending on the test environment and the quality of the satellite observations; however, the precision was slightly degraded from that of the tracking-only solution by a factor of about 1.75, except in the case of stationary localization, where precision was actually improved. Based on additional investigation into single-epoch tracking anomalies, we found that this typical decrease in accuracy was due to very specific regions of higher-than-normal tracking error which caused our APT localization filter to begin tracking incorrect peaks.

Regardless of the high dependence of our localization technique on the precision of the tracking results, our methodology showed that by allowing the filter to re-initialize itself on-the-fly, it is usually possible to reacquire a lock on the "correct" peak, thereby removing the necessity for user intervention in the case of erroneous measurements. Likewise, since initialization is handled by the localization algorithm itself, calibration becomes an *implicit* step requiring no stationary setups or *a priori* knowledge of the precise relative baselines between pairs of receivers. Likewise, the time required for the filter to converge to a high-accuracy, steady-state solution is generally less than two minutes when the receivers are in motion, and anywhere from 3 to 4 minutes when stationary.

Finally, our experiments included receiver-receiver baseline lengths ranging from 0 m all the way up to 3.5 km, with little to no impact on the precision of the results. We conclude, therefore, that our method is quite robust to changing baseline lengths, and the limiting factor may be the various mathematical assumptions regarding the similarity of satellite unit direction vectors for multiple receivers in the same geographic region, as well as the assumption that these direction vectors remain relatively constant over single epochs of receiver motion in the Temporal Double-Differencing model. As such, we anticipate similar levels of accuracy to be achievable for baselines up to ~1000 km in length.

Recommendations for Future Work

One of the primary limitations of the work presented in this dissertation is the strong dependence of the APT localization algorithm on the initial estimate of the baseline between two receivers. The worst-case accuracy of this estimate dictates the minimum size of the search cube that must be traversed when identifying the relevant peaks and baseline candidates for a given pair of receivers. Any increase in baseline uncertainty results in a cubic increase in the number of points that must be searched to find the correct peak. Correspondingly, since the search resolution of APT is so fine, even extremely small increases in the required size of the search space can result in drastically longer processing times, especially for the initial search procedure in which candidate peaks are identified.

In order to cut down on the time required to search through this vast number of points, additional research should be carried out on determining a better estimate of the initial baseline between two receivers before using the algorithms described in this dissertation. Conversely, it would also be beneficial to have a better idea of the confidence interval (or an estimate of the worst-case error) for any given baseline. Better accuracy estimates would ensure that the correct baseline is not missed in the initial peak determination algorithm, as well as minimize the number of extra points that must be evaluated in the case of an overly pessimistic accuracy estimate.

Experiments using the complete APT localization framework highlighted several drawbacks of the hill-climbing technique used to remove accumulated bias from the requisite tracking results. These include a stronger than anticipated dependence on the accuracy of the tracking results themselves, as well as an increased susceptibility to errors caused by multipath. As such, additional research into methods of removing short-term tracking biases should be carried out to ensure that the locality of peaks being tracked by the APT filter remains constant through time.

It would, of course, be beneficial for future applications to develop “smarter” peak searching algorithms, such that only a subset of the search space need be traversed to identify any relevant peaks. This would drastically cut down on the processing time required to initially identify the peaks, as well as the time it takes to track them. As such, the scalability of this research would improve to allow for a larger number of receivers to be localized in real-time without experiencing data bottlenecks or processing latency issues.

In addition to improving the existing algorithms for speed and computational complexity, the most obvious and immediately rewarding path for future work would be to increase the robustness of the techniques presented here for difficult GPS environments, including urban canyons and areas of dense tree cover. A final extension with the potential to improve overall accuracy would be to perform a network-wide localization step using the the individual relative location vectors computed by the nodes. In surveying terms, this is referred to as "closing the loop," whereby the addition of any number of relative vectors which form a closed-loop from one receiver back to itself should equal 0. In a network of n nodes, $(n-1)$ such vectors are sufficient to compute the relative locations of any of the nodes. In our system, we have far more than this required number, resulting in an overdetermined system. Any number of optimization approaches could be used to estimate the node locations while minimizing overall error.

Broader Implications

Our method has several unique characteristics that give it a broad appeal and make it attractive for a wide array of use cases. First, since our localization algorithm works on pairwise sets of satellite data, its mathematical complexity does not grow with the addition of GPS receivers to the network. In fact, from the framework processing times shown in Table 3, it is clear that our system should be able to support up to 60 steady-state mobile receivers simultaneously without running into latency issues. The caveat to this statement is that the "initialization mode" of the localization algorithm requires significantly more processing time than the nominal "steady-state localization mode;" therefore, as additional nodes enter the network, there will be intermittent periods during which position data for the new, calibrating receivers is unavailable. This is an important aspect to consider when examining the applicability of our technique to real-time systems.

Aside from the processing overhead associated with baseline calibration, however, the primary bottleneck in our system is the communication bandwidth required to broadcast the raw satellite measurements among a potentially large number of receivers. Since the amount of data transmitted from a single node every second is actually quite low (on the order of 500 bytes/second), there are better ways (than the naïve cloud-based multicast approach used in our experiments) of approaching this problem if and when communication scalability becomes a problem.

Finally, since each and every node considers itself to be the "reference node" in our technique, the system is symmetric and contains no single points of failure. If any node experiences a fault or error of any kind, it can simply drop out of the localization procedure until the problem is resolved without affecting the reliability of the remaining healthy nodes.

This is not only beneficial for introducing fault-tolerance into networked receiver arrays, but it also allows for the system to work in a truly ad-hoc fashion without any node requiring knowledge of the network topology or connectivity to any of its neighbors.

While the majority of off-the-shelf and embedded GPS receivers currently only employ simple point positioning algorithms, the number of precision devices that rely on GPS as their primary localization service is growing rapidly [33, 22]. The proliferation of smartphones and GPS-enabled mobile sensors, along with the ever-increasing availability of anywhere, anytime network coverage, is creating a substantial and practical platform for the relative positioning method described in this dissertation. As the price of technology continues to decline and our world becomes ever more connected, further research into highly accurate, low-cost relative positioning techniques will continue to find its way even more prominently into our everyday devices, and we hope to see the approach presented in this dissertation spark further research and become only one of a body of ad-hoc relative localization techniques centered around low-cost, commercial GPS receivers.

APPENDIX A

GLOSSARY AND DEFINITIONS

Ambiguity Function Method (AFM): In terms of GPS, a localization method carried out in the position domain that relies on searching for the baseline that most closely maintains the integer nature of the ambiguity terms in the double-differenced model.

Ambiguity Function Value (AFV): A value in the range $-1.0 \leq \text{AFV} \leq 1.0$ resulting from evaluation of a set of 3D coordinates in an ambiguity function, where values closer to 1.0 represent "better," or more consistent, coordinate candidates.

Autocorrelation: A mathematical tool for evaluating how similar a signal is to itself or to some copy of itself shifted in time.

Azimuth: An angular measurement representing the location of an astronomical object on the horizon (with true north as a reference).

Baseline: The 3D vector spanning from one GPS receiver to another.

Binary Phase Shift Keying (BPSK): A digital modulation scheme in which binary bits are encoded onto a carrier sine wave by means of an instantaneous phase shift of either $0°$ or $180°$.

Carrier Phase: In this paper, the accumulated number of carrier wave cycles that have been received from a satellite at a given point in time.

Carrier Range: The carrier phase in terms of meters (i.e. multiplied by the carrier wavelength, λ).

Clock Bias: The difference between a local clock's time measurement and the "true" or "universal" time.

Clock Drift: The change in clock bias over time.

Code Division Multiple Access (CDMA): A channel access method in which multiple streams of data can be transmitted over the same data channel by encoding each data stream with its own unique pre-defined code.

Codephase: The current position in a finite-length PRN code sequence (akin to carrier phase, except the carrier is a binary bit sequence instead of a sine wave).

Cycle Slip: A very short loss of correlation between a receiver's internally-generated PRN sequence and the PRN sequence received from a satellite. Small correlation losses result in the appearance of fewer received carrier cycles.

Dead Reckoning: The process of estimating a current position based on a previous position estimate plus an estimate of the change in position through time.

Differential GPS (DGPS): The set of localization methodologies that rely on transmitted pseudorange corrections from a reference base station with a known position to increase relative baseline accuracy.

Direct Sequence Spread Spectrum (DSSS): A data modulation technique in which a data stream is *spread* over a much wider bandwidth than is strictly necessary for the data being transmitted. A variety of spreading methods exist, such as the CDMA method above.

Doppler Shift: The change in frequency of a waveform as perceived by a receiver moving relative to the signal's transmission source.

Earth-Centered, Earth-Fixed (ECEF): A coordinate system with its origin at the center of the Earth that rotates along with the Earth such that a point on the surface of the Earth always corresponds to the same set of coordinates.

Earth-Centered Inertial (ECI): A coordinate system in inertial space with its origin at the center of the Earth that does not rotate with the Earth.

Earth Rotation Corrections: Updates to estimated satellite ephemerides at a given point in time based on the amount of Earth rotation experienced by a signal traveling from the satellite to an Earth-bound receiver.

Ephemeris (pl. *ephemerides*): A set of values indicating the position of an astronomical object in the sky.

Epoch: A notable moment in time — i.e. a specific moment used as a reference for events occurring immediately before, during, or after the specified point in time.

Gaussian Distribution: A continuous probability distribution that has a bell-shaped probability density function described exclusively by its mean and variance.

Geostationary: Used to describe satellites positioned directly over the Earth's equator with an orbital period equal to the Earth's rotational period. Such satellites appear permanently fixed at a certain point in the sky to an observer on Earth.

Geosynchronous: Used to describe satellites with an orbital period equal to the Earth's rotational period. Such satellites trace out a path in the sky, and appear to be at the exact same point after a period of one sidereal day.

GPS Time: The synchronized, universal time as referenced according to the GPS satellites.

Half-cycle Slip: *See "cycle slip."* Only found in receivers that rely on signal squaring to remove encoded information. When squared, carrier waves repeat every one-half-wavelength, making it possible for cycle slips to occur on some number of half-wavelengths.

Hill-Climbing: An iterative search optimization technique that attempts to locate a local maximum by incrementally changing single solution elements until no improvements are possible.

Integer Ambiguity: The unknown number of full-cycle wavelengths that exist between a satellite and receiver at the time of satellite lock. (Note, this is only an integer in the context of the "double-differenced observation model.")

Keplerian Orbit: An idealized, mathematical approximation of an astronomical orbit at a given time, using the six parameters of eccentricity, semimajor axis, inclination, longitude of ascending node, argument of periapsis, and mean anomaly.

L1/L2: The designated wavelengths for the two GPS carrier signals. L1 represents the civilian signal, and L2 represents the military signal.

Least Squares: Any over-determined approximation or estimation procedure which generates a solution such that the resulting sum of the squares of residual errors is minimized.

Loss of Lock: A total loss of correlation between a receiver's internally generated PRN code and the same PRN code received from a GPS satellite.

Multipath: A propagation phenomenon whereby one radio signal arrives at an antenna at two or more time instants due to physical reflections and path delays.

Pseudorandom Noise (PRN): A sequence of pre-defined bits which are known *a priori* but which have the spectral properties of noise.

Range Rate: The change in range between a satellite and receiver through time.

Real-Time Kinematic (RTK): The set of localization methodologies that rely on carrier phase measurements from two or more receivers to provide real-time localization updates.

Reference Satellite: The satellite that is used to form all double-differenced observations for a single epoch such that all resulting measurements are linearly independent.

Sagnac Effect: In terms of GPS, errors in the apparent satellite position arising from the relative motion of the receiver (Earth rotation) during the time in which a radio signal transmitted by the satellite was in transit.

Sidereal Day: The time it takes for a point on the surface of the Earth to make one full revolution such that an astronomical object appears to be at the exact same point in the sky. The mean sidereal day for Earth is 23 hours, 56 minutes, 4.0916 seconds.

Space Vehicle Number (SVN): The official satellite identification numbers assigned by the Department of Defense.

Temporal Model: Any observation model which relies on the difference between data over two or more epochs.

Time of Flight: The transit time of a radio signal from time of transmission to time of reception.

User-Equivalent Range Error (UERE): The error of a measurement observation in the distance from a receiver to a satellite in units of meters.

White Noise: A random signal with a flat power spectral density in which the signal contains an equal amount of power at all frequencies within a fixed bandwidth at any center frequency.

Zenith: The point directly above a particular location. In terms of GPS receiver, the highest point in the sky visible to it.

APPENDIX B

MATHEMATICAL NOMENCLATURE

General note: *All superscripts relate to satellites, while all subscripts relate to receivers.*

$\mathbf{a}_r^s$: Unit direction vector from receiver r to satellite s

$\mathbf{b}$: Baseline vector between two receivers

B_r^s: Full (integer and fractional) phase ambiguity between receiver r and satellite s

B_{jk}^s: Single-differenced phase ambiguity to satellite s between receivers j and k (i.e. $B_k^s - B_j^s$)

f_0: A nominal frequency.

f_R: A received frequency.

f_D: A Doppler shift.

L_r^s: Carrier range observation from receiver r to satellite s.

ΔL_{jk}^s: Single-differenced carrier range to satellite s between receivers j and k (i.e. $L_k^s - L_j^s$)

$\nabla\Delta L_{jk}^{mn}$: Double-differenced carrier range between satellites m and n and between receivers j and k (i.e. $\Delta L_{jk}^n - \Delta L_{jk}^m$)

$\Delta\nabla\Delta L_{jk}^{mn}$: Triple-differenced carrier range between satellites m and n and between receivers j and k for two consecutive time epochs (i.e. $\Delta\nabla L_{jk}^{mn}(t) - \Delta\nabla L_{jk}^{mn}(t-1)$)

N_r^s: Integer ambiguity between receiver r and satellite s.

P_r^s: Pseudorange observation from receiver r to satellite s.

ΔP_{jk}^s: Single-differenced pseudorange to satellite s between receivers j and k (i.e. $P_k^s - P_j^s$)

$\nabla\Delta P_{jk}^{mn}$: Double-differenced pseudorange between satellites m and n and between receivers j and k (i.e. $\Delta P_{jk}^n - \Delta P_{jk}^m$)

$\Delta\nabla\Delta P_{jk}^{mn}$: Triple-differenced pseudorange between satellites m and n and between receivers j and k for two consecutive time epochs (i.e. $\Delta\nabla P_{jk}^{mn}(t) - \Delta\nabla P_{jk}^{mn}(t-1)$)

Φ_r^s: Carrier phase observation from receiver r to satellite s.

ϕ_r^s: Instantaneous phase measured by receiver r to satellite s.

ρ_r^s: Geometric range from receiver r to satellite s.

ρ_{jk}^s: Single-differenced geometric range to satellite s between receivers j and k (i.e. $\rho_k^s - \rho_j^s$)

T: When capitalized, represents a discrete time that includes some amount of bias or error. In other words, it is a time based on a local clock.

t: When lowercase, represents a discrete time free of any bias or error. In other words, it describes *actual*, universal time.

τ: Describes a clock bias from universal time.

X^s, Y^s, Z^s: X, Y, Z coordinates of satellite s

x_r, y_r, z_r: X, Y, Z coordinates of receiver r

REFERENCES

[1] Federal Aviation Administration. Navigation Services - Wide Area Augmentation System (WAAS). August 2010, http://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/techops/navservices/gnss/waas/.

[2] Cooper E. Allison. GPS and RTK-GPS: A Comparison. EzineArticles, October 31, 2011, http://ezinearticles.com/?GPS-and-RTK-GPS:-A-Comparison&id=6642248.

[3] S.F. Appleyard, R.S. Linford, and P.J. Yarwood. *Marine Electronic Navigation (2nd Edition)*. Routledge and Kegan Paul, 1988.

[4] Neil Ashby. The Sagnac effect in the Global Positioning System. *Relativity in Rotating Frames*, 2010.

[5] Geoffrey Blewitt. Basics of the GPS Technique: Observation Equations. *Geodetic Applications of GPS*, 1997.

[6] Bill Clinton. Statement by the President regarding the United States' decision to stop degrading Global Positioning System accuracy. Office of the Press Secretary, May 1, 2000.

[7] J.M. Codol and A. Monin. Improved triple difference GPS carrier phase for RTK-GPS positioning. *IEEE Statistical Signal Processing Workshop*, pages 61–64, 2011.

[8] Oscar L. Colombo. Ephemeris errors of GPS satellites. *Journal of Geodesy*, 60(1):64–84, 1986.

[9] Southern Avionics Company. DGPS Reference Station Transmitters. 2012, http://www.southernavionics.com/products/differential-gps-transmitters/.

[10] C. Counselman and S. Gourevitch. Miniature interferometer terminals for earth surveying: Ambiguity and multipath with Global Positioning System. *IEEE Transactions on Geoscience and Remote Sensing*, GE-19(4), October 1981.

[11] Peter H. Dana. Global Positioning System Overview. University of Texas at Austin, 1999, www.colorado.edu/geography/gcraft/notes/gps/gps_f.html.

[12] Global Positioning Systems Directorate. Systems engineerings and integration interface specification. IS-GPS-200F, September 2011.

[13] Hamid Ebadi. Positioning with GPS. K.N. Toosi University of Technology, August 2000.

[14] Ahmed El-Sayed El-Rabbany. The effect of physical correlations on the ambiguity resolution and accuracy estimation in GPS differential positioning. *Technical Report No. 170, Geodesy and Geomatics Engineering*, December 1997.

[15] Tamer F. Fath-Allah. A new approach for cycle slips repairing using GPS single frequency data. *World Applied Sciences Journal*, 8(3):315–325, 2010.

[16] Survey Land Advisory Board for State of Washington Department of Natural Resources. GPS guidebook: Standards and guidelines for land surveying using Global Positioning System methods. Version 1, November 2004.

[17] The Eclipse Foundation. Eclipse memory analyzer (mat). 2014, `http://www.eclipse.org/mat/`.

[18] J.G. Garcia, P.I. Mercader, and C.H. Muravchik. Use of GPS carrier phase double differences. *Latin American Applied Research*, 35:115–120, 2005.

[19] Tekmon Geomatics. Antenna phase center offset and variation. 2011-2012, `http://tekmon.gr/tag/phase-center-antenna/`.

[20] Tekmon Geomatics. Network RTK. GPS Network Tutorial, 2011-2012, `http://tekmon.gr/2011/03/network-rtk-2/`.

[21] Waypoint Consulting Inc. Relative moving baseline software (RMBS). March 2000, `http://webone.novatel.ca/assets/Documents/Waypoint/Reports/RelativeMovingBaselineSoftware.pdf`.

[22] Berg Insight. GPS and mobile handsets. Technical Report, LBS Research Series, 2012.

[23] Alan Kaminsky. Trilateration. Rochester Institute of Technology, Department of Computer Science, Ad hoc Networks Class, Module 05: `http://www.cs.rit.edu/~ark/543/module05/trilateration.pdf`.

[24] Elliott D. Kaplan and Christopher J. Hegarty. *Understanding GPS: Principles and Applications*. Artech House, Inc., Norwood, MA, USA, second edition, 2006.

[25] kowoma.de. Sources of errors in GPS. The GPS System, `http://www.kowoma.de/en/gps/errors.htm`.

[26] D. Kuang, B.E. Schutz, and M.M. Watkins. On the structure of geometric positioning information in GPS measurements. *Journal of Geodesy*, 71(1):35–43, 1996.

[27] NASA Jet Propulsion Laboratory. The NASA Global Differential GPS System. NASA, September 2009, `http://www.gdgps.net/system-desc/index.html`.

[28] Simon Lightbody and Gary Chisholm. Techniques in relative RTK GNSS positioning. White Paper, Trimble Marine Group, 2010.

[29] The Decca Navigator Company Limited. *The Decca Navigator: Principles and Performance of the System*. The Decca Navigator Company Limited, July 1976.

[30] Ning Luo. Centimetre-level relative positioning of multiple moving platforms using ambiguity constraints. In *Proceedings of the ION GPS2000 Conference, Salt Lake City, UT*, September 2000.

[31] Cetin Mekik and Murat Arslanoglu. Investigation on accuracies of real time kinematic GPS for GIS applications. *Remote Sensing Journal*, 1:22–35, March 2009.

[32] National Coordination Office for Space-Based Positioning, Navigation, and Timing. Control segment. GPS.gov - Official U.S. Government information about GPS and related topics, June 2012, `http://www.gps.gov/systems/gps/control/`.

[33] Kevin J. O'Brien. Smartphone sales taking toll on GPS devices. New York Times, November 14, 2010, Online at `http://www.nytimes.com/2010/11/15/technology/15iht-navigate.html`.

[34] U.S. Departments of Defense and Transportation. 1994 Federal Radionavigation Plan. National Technical Information Service, Springfield, Virginia, 22161, DOT-VNTSC-RSPA-95-1/DOD-4650.5, May 2005.

[35] U.S. Department of Homeland Security. *NAVSTAR GPS User Equipment Introduction*. U.S. Department of Homeland Security, September 1996.

[36] U.S. Department of the Air Force. Pseudorandom Noise (PRN) Code Assignments. Los Angeles Air Force Base Website, Feb 2011, `http://www.losangeles.af.mil/library/factsheets/factsheet.asp?id=8618`.

[37] U.S. Department of the Navy. Current GPS Constellation. http://tycho.usno.navy.mil/gpscurr.html.

[38] Bradford W. Parkinson. Introduction and heritage of NAVSTAR, the Global Positioning System. *Global Positioning System: Theory and Applications*, 1:3–28, 1996.

[39] Bradford W. Parkinson, James J. Spilker Jr., Penina Axelrad, and Per Enge. *Global Positioning System: Theory and Applications*, volume 164. Progress in Astronautics and Aeronautics, ii edition, January 1996.

[40] Mark Petovello. GNSS solutions: Carrier phase and its measurements for GNSS. InsideGNSS, July/August 2010, `http://www.insidegnss.com/auto/julaug10-solutions.pdf`.

[41] Nam D. Pham. The economic benefits of commercial GPS use in the U.S. and the costs of potential disruption. NDP Consulting, June 2011.

[42] Darius Plausinaitis. GPS signal acquisition. GPS Signals and Receiver Technology MM11, Danish GPS Center, Aalborg University, Denmark, 2009.

[43] Maxim Integrated Products. An introduction to spread-spectrum communications. Application Note 1890 (AN1890), Feb 2003.

[44] Honghui Qi and J.B. Moore. Direct Kalman filtering approach for GPS/INS integration. *IEEE Transactions on Aerospace and Electronic Systems*, 38:687–693, 2002.

[45] Marco Rao and Gianluca Falco. Code tracking and pseudoranges. GNSS Solutions – Inside GNSS: Engineering Solutions from the Global Navigation Satellite System Community, January/February 2012, `http://www.insidegnss.com/node/2898`.

[46] Sabbi Babu Rao. Design and implementation of a GPS receiver channel and multipath delay estimation using teager-kaiser operator. Masters Thesis, Department of Super Computer Education and Research Centre, Indian Institute of Science, Bangalore, India, June 2009.

[47] J.K. Ray and M.E. Cannon. Characterization of GPS carrier phase multipath. In *Proceedings of ION NTM-99, San Diego*, January 1999.

[48] Jagdish Rebello. Four out of five cell phones to integrate GPS by end of 2011. Press Release, IHS Market Research, July 16, 2010.

[49] Zhoufeng Ren, Liyan Li, Jie Zhong, Minjian Zhao, and Yingjie Shen. A real-time cycle-slip detection and repair method for single frequency GPS receiver. In *2nd International Conference on Networking and Information Technology (IPCSIT)*, volume 17. IACSIT Press, Singapore, 2011.

[50] Chris Rizos. Principles and practice of GPS surveying. The University of New South Wales, Syndey, Australia, May 2000, `http://www.gmat.unsw.edu.au/snap/gps/gps_survey/principles_gps.htm`.

[51] J.B. Schleppe. Development of a real-time attitude system using a quaternion parametrization and non-dedicated GPS receivers. M.Sc. Thesis, University of Calgary, Canada, 1996.

[52] United States Geological Survey GPS Committee. USGS Global Positioning System. GPS Applications and Surveying Methods, `http://water.usgs.gov/osw/gps/index.html`.

[53] Rick W. Sturdevant. Chapter 17: NAVSTAR, the Global Positioning System: A sampling of its military, civil, and commercial impact. *Societal Impact of Spaceflight*, NASA SP-2007-4801:331–351, 2007.

[54] National Geodetic Survey. Antenna calibrations. `http://www.ngs.noaa.gov/ANTCAL/`.

[55] u-blox AG. Lea-6t module with precision timing. 2013, `http://www.u-blox.com/de/lea-6t.html`.

[56] Jan van Sickle. *GPS for Land Surveyors.* Taylor and Francis, New York, NY, second edition, 2001.

[57] Péter Völgyesi, Sándor Szilvási, János Sallai, and Ákos Lédeczi. External smart microphone for mobile phones. In *Proceedings of the International Conference on Sensing Technology*, ICST, 2011.

[58] Phillip Ward. The natural measurements of a GPS receiver. In *Proceedings of the 51st Annual Meeting of the Institute of Navigation*, pages 67–85, June 1995.

[59] Clifford M. Will. Einstein's Relativity and Everyday Life. Physics Central, `http://physicscentral.com/explore/writers/will.cfm`, 2012.

[60] Oliver J. Woodman. An introduction to inertial navigation. Technical Report Number 696, University of Cambridge, August 2007.

[61] G. Wubenna, M. Schmitz, F. Menge, V. Boder, and G. Seeber. Automated absolute field calibration of GPS antennas in real-time. In *International Technical Meeting ION GPS-00*. Salt Lake City, Utah, USA, September 2000.

[62] Yudan Yi. On improving the accuracy and reliability of GPS/INS-based direct sensor georeferencing. Thesis for Doctor of Philosophy, Ohio State University, 2007.

[63] Jason Zhang, Kefei Zhang, Ron Grenfell, and Rod Deakin. On the relativistic doppler effect for precise velocity determination using GPS. *Journal of Geodesy*, 80:104–110, 2006.