ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Autonomous Systems Lab
Prof. Roland Siegwart

Semester-Thesis

Real Time Kinematic GPS for Micro Aerial Vehicles

Spring Term 2012

Supervised by:
Markus Achtelik
Simon Lynen

Author:
Daniel Grieneisen

Declaration of Originality

I hereby declare that the written work I have submitted entitled

Real Time Kinematic GPS for Micro Aerial Vehicles

is original work which I alone have authored and which is written in my own words.[1]

Author(s)

Daniel Grieneisen

Supervising lecturer

Markus Achtelik Simon Lynen

With the signature I declare that I have been informed regarding normal academic citation rules and that I have read and understood the information on 'Citation etiquette' (`http://www.ethz.ch/students/exams/plagiarism_s_en.pdf`). The citation conventions usual to the discipline in question here have been respected.

The above written work may be tested electronically for plagiarism.

Place and date Signature

[1] Co-authored work: The signatures of all authors are required. Each signature attests to the originality of the entire piece of written work in its final form.

Contents

Abstract

For any MAV task, especially exploring unknown environments, it important to have high accuracy localization measurements. For high quality, stationary receivers, real time kinematic (RTK) GPS can achieve centimeter level accuracy. However, solution quality can be degraded due to noisy data or fast accelerations. The size of an MAV limits the size and quality of antenna and receiver that it can carry, and they move with very fast dynamics.
In this project, I explored the potential for use of RTK GPS on MAVs. I used RTKLIB, an open source GPS processing library to evaluate solution quality for a variety of processing methods for both stationary and moving receivers. For real-time localization, the RTK solution performs well, with an RMS error of less than 1 m in horizontal position. With post-processing, the solution was smoothed, but the overall error was not significantly reduced. Therefore, RTK GPS represents a viable real-time localization measurement to be used with an on-board state estimation filter, but it does not provide post-processed ground truth with less error.
Experiments with the rover receiver actually mounted on an MAV revealed that electronic interference from the on board processor greatly degraded the GPS signal quality and the resulting positioning solutions. The GPS antenna must be mounted further away from the processor or some sort of shielding needs to be used to improve signal quality enough for RTK GPS solutions to be viable. Finally, I created a ROS wrapper for RTKLIB so that it can be easily used in future projects at the lab.

Symbols

Acronyms and Abbreviations

AGNES	Automated GNSS Network Switzerland
C/A Code	Coarse Acquisition Code
DGPS	Differential GPS
DOP	Dilution of Precision
ECEF	Earth-Centered, Earth-Fixed
EKF	Extended Kalman Filter
ENU	East North Up
ETH	Eidgenössische Technische Hochschule
EUREF	IAG Subcommission for Europe
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
MAV	Micro Aerial Vehicle
NTRIP	Network Transport of RTCM via Internet Protocol
P Code	Precision Code
PPP	Precise Point Positioning
PPS	Precise Position Service
ROS	Robotic Operating Systeam
RTCM	Radio Technical Committee for Maritime Standards
RTK	Real Time Kinematic
SBAS	Satellite Based Augmentation System
SNR	Signal to Noise Ratio
SPS	Standard Position Service
UTC	Coordinated Universal Time
WADGPS	Wide Area Differential GPS

Chapter 1

Introduction

There are an increasingly large number of uses for micro-aerial vehicles (MAVs), from exploration and map building to autonomous patrol and station keeping. In any of these scenarios, it is important to estimate the exact global position of the MAV, especially when exploring previously unknown environments. Ideally, a position measurement sensor should be able to measure accurate localization to be used either to provide data for real time localization and mapping algorithms, to provide a failsafe in case another position estimation system fails, or to calculate accurate ground truth of the MAV for post-processing. Outdoors, the global positioning system (GPS) can provide accurate position measurements to within 10-15 meters [7]. As the typical MAVs used in the Autonomous Systems Lab at ETH are less than a meter in diameter, this it is not really sufficient to serve any of the previously mentioned purposes. The quality of the GPS solution is a function of a number of noise parameters that encompass everything from the signal delay in the atmosphere to the quality of the amplifiers in the antenna used to receive the signals. There are, however, methods that can correct for and mitigate a number of these noise factors. Using differential GPS, which involves data from two receivers, static position measurements can be improved to the order of a few centimeters [11]. Global position measurements of this accuracy can greatly enhance the localization estimate of an MAV.

Unfortunately, the size of MAVs poses some severe restrictions on the size of onboard sensors. Differential GPS receivers that purport to have sub-decimeter accuracy are typically too heavy to be carried by the MAVs used in the lab, where every gram of mass needs to be minimized. In addition, the algorithms that provide this highly precise localization can be disrupted by rapid accelerations, which are likely to occur in typical MAV flight scenarios.

The goal of this project is to explore the feasibility of using differential GPS techniques, such as real time kinematic (RTK) algorithms, for accurate position measurements on an MAV. It focuses on two primary use cases: real-time localization and post-processed ground truth. The first is useful for real-time MAV state estimation and control. The second is useful for determining the quality of other localization algorithms on the MAV. Additionally, I implemented a ROS software interface for an RTK GPS library to allow easy integration into other projects within the lab.

Chapter 2

Background

2.1 GPS Introduction

The Global Positioning System (GPS) is a satellite based method of accurately determining the three dimensional global position of receiver based on measuring the propagation time of radio signals. It was originally developed by the United States Department of Defense in the 1970s and 1980s as a way to determine the positions of ships and aircraft [3]. GPS relies on calculating the distance from the receiver to a satellite based on the time-of-flight of a radio signal sent from the satellite. Given the range measurements from three different satellites and their spatial locations, the position of the receiver can be calculated via triangulation. However, the range measurements are not perfect. They are based on measuring the time that it takes for the signals to propagate from the satellites to the receiver. All of the satellites contain precise, (relatively) synchronized clocks. The receiver, however, generally has a much less precise crystal oscillator clock which is not synced with the satellites. Therefore, the distance measured to each satellite is corrupted by the timing error between the satellite clocks and the clock on the receiver. This measured distance is referred to as 'pseudorange'. Given that there is now a fourth unknown (timing error in addition to three-dimensional position), at least four satellites are required to compute the position of the receiver. More satellites can be used to provide a more accurate solution.

2.2 GPS Segments

GPS can be broken down into three primary segments: space segment, control segment, and user segment [3].

1. Space Segment - The space segment refers to the constellation of satellites. There are 24 GPS satellites currently in orbit (referred to as the NAVSTAR system). They orbit along 6 different paths approximately 10,000 miles above the Earth with 55 degree inclination angles. The orbital period is 23 hours, 56 minutes. They are arranged such that there are always at least 5 satellites visible from any location on the Earth's surface. They are referred to as "Block II" satellites because they are the second generation launched. Data is transmitted using a pseudorandom code. There are two different types of codes: Coarse Acquisition (C/A) and Precision (P). P codes allow higher accuracy measurements and are less susceptible to jamming, but take longer to acquire a fix. They are used primarily by the military, though civilian applications can now use them. The C/A codes are used by most civilian GPS

receivers. The satellites transmit data over a number of different frequency bands. Most receivers use only the L1 and L2 bands. L1, at 1575.42 MHz, transmits both C/A and P codes, while L2, at 1227.6 MHz, transmits just P codes. The C/A code is used for the Standard Position Service (SPS), while the P code is used for the Precise Position Service (PPS). On top of the code, the satellites transmit data at 50 bits per second. This data includes daily satellite orbit and clock error correction terms.

2. Control Segment - The control segment refers to ground stations that monitor the signals transmitted by the satellites. They calculate the precise orbital and timing errors of the satellites and transmit this information back to the satellites once per day.

3. User Segment - The user segment refers to any end user receivers. A GPS receiver uses a combination of two loops to lock onto the signal from a satellite. The first, the code loop, locks onto the C/A or P code transmitted by the satellite, and generates the code pseudorange measurement. The phase loop locks onto the phase of the carrier signal and provides a highly accurate phase difference measurement. Cheap commercial receivers generally only use the L1 band, while dual frequency receivers use both the L1 and the L2 bands.

2.3 GPS Error Budget

There are a number of sources of error that degrade the quality of GPS solutions. A single SPS solution, such as that produced by a low end consumer receiver, can generally achieve accuracy of around 10-15 meters [3]. The following error sources (summarized in table 2.1) affect this accuracy. Different experts provide different values of the magnitude of error for each of these sources. I have tried to use average values to approximate the amount of expected error.

Table 2.1: Summary of GPS error budget

Error Source	Effect on solution
Selected Availability (now disabled)	50-100 m
Ionosphere	0-30 m
Troposphere	0-30 m
Satellite Ephemeris	1-2 m
Satellite Clock	1-2 m
Multipath	1-2 m
Receiver Noise	1-2 m
Dilution of Precision	Scales expected error

1. Selected Availability (SA) - Selected availability was a United States government policy where timing errors and satellite position estimate errors were purposely introduced into the C/A signals to degrade the quality of SPS solutions. This was to ensure that civilians and foreign countries did not have access to high quality GPS [3]. SA added 50-100 meters in error. However, in May 2000, SA was officially turned off, so it is no longer a source of error [5].

2. Ionospheric Error - Electromagnetic signals are refracted in the ionosphere due to the large number of charged particles there. This introduces signal delay [5]. Though this delay depends on the location of the receiver and time

of day, it can be modeled relatively well. Unknown ionosphere changes, such as strong solar wind, cannot be corrected in real time, but may be accounted for in post-processing. The error due to the ionosphere can range from 0-30 meters, though it is generally less than 5 meters [6].

3. Tropospheric Error - Electromagnetic signals are also refracted in the troposphere, though it is due to different concentrations of water vapor from weather patterns, such as clouds [5]. This again introduces signal delay. With proper modeling, this error can also be mitigated somewhat. The Saastamoinen model is most commonly used to estimate this delay. The error can be anywhere from 0-30 meters, though it is generally less than 4 meters [6].

4. Ephemeris Error - Over time, the satellites accumulate some minor errors in their orbits. Though they send updated orbital information on top of the code signals, there can still be some error between the actual satellite position and the position expected by the receiver. This is referred to as orbital error or ephemeris error, and it is generally less than 2 meters [5].

5. Clock Error - The clocks on the satellites are highly accurate. However, like the satellite orbits, they too accumulate some errors over time. This is calculated daily by the ground control stations and broadcast to the satellites, which then send the information to the receivers with their broadcasts. There can be some error in this clock correction information, and it can lead to position errors of up to 2 meters [5].

6. Multipath Effects -The signal from a satellite can reflect off of objects such as buildings before reaching the receiver. This added time from bouncing off of things can introduce error in position calculations of up to 1-2 meter [6].

7. Receiver Errors - There are a number of error sources within the receiver, such as amplifier noise and rounding errors in calculations. These can add up to 1-2 meter in error [5].

8. Dilution of Precision (DOP) - Dilution of Precision is purely a function of the geometry of the visible satellites. Because position is calculated via triangulation, having a good spread of satellites across the sky results in a more accurate position than having a few satellites close together. DOP is a scale factor that accounts for the extra uncertainty. A position calculation with a DOP of 2 is half as accurate as one with a DOP of 1. There are a few different DOP measurements that typical GPS receivers calculate [5]. Geometric DOP (GDOP) is the overall 3D position and time error. Position DOP (PDOP) is just the 3D position error, horizontal DOP (HDOP) is the 2D position error, and vertical DOP (VDOP) is the position error in altitude.

There are a few different ways to try to mitigate the sources of error in GPS. One is to use dual band receivers. By receiving PPS signals at two different frequencies, the ionosphere errors can be eliminated. Another method is to use correction data for post processing. Accurate ionosphere, troposphere, satellite ephemeris and satellite clock data is published by multiple sources a few days post facto. This information, however, is only available for post processing. The primary means of improving real-time GPS solution quality is through differential GPS.

2.4 Differential GPS

The general idea of differential GPS is rather straight forward. For two receivers placed relatively close to each other, the signals from the satellites will pass through

similar sections of the ionosphere and troposphere. Additionally, they will both be affected by the same variability in the satellite ephemeris and clock data. By recording data at a known fixed location ("base station") and comparing it with the data recorded at the unknown location ("rover"), the effect of these different sources of error can be mostly eliminated, greatly improving accuracy. In calculating accurate rover positions, DGPS uses double-differenced observables. A single-differenced observable is calculated by subtracting two different measurements, either the measurement from the same satellite to two different receivers, or the measurement from the same receiver to two different satellites. A double-differenced observable is the difference between two single-differenced observables. That is, it compares the measurements at two different receivers of two different satellites. By using double-differenced observables, ephemeris, clock, and atmosphere errors can be eliminated.

2.4.1 Code-based Differential GPS

Code-based differential GPS uses the receivers' measurements of the pseudoranges to the satellites based on the code, rather than carrier phase. Given the difference between the measured and actual position of the base station, a correction value for each of the satellites from which it received messages is calculated. These correction values are then used to correct the measured pseudoranges for the signals received by the rover. Note that only the signals from satellites which are seen by the base station can be corrected for the rover[7]. The correction data can be used in post-processing or transmitted in real-time. Using code-DGPS, accuracy of approximately 1-3 meters is achievable.

2.4.2 Carrier-phase Differential GPS

In carrier-phase differential GPS, correction information for the rover is based on the measured phase of the carrier waves of the received signals. Because the L1 carrier wavelength is approximately 19 cm, this can result in highly accurate pseudorange measurements. One problem with measuring just the phase of a satellite signal, however, is that it is not a direct measurement of the pseudorange. The pseudorange is the code-triggered phase edge plus an integer number of wavelengths. Carrier-phase GPS calculates the precise relative position of the rover to the base station by determining the number of wavelengths difference for each of the visible satellites. This is known as the integer ambiguity problem. By resolving this number of wavelengths difference to an integer, much higher accuracy can be achieved. There are a few methods to do this resolution. One is to know the initial location of the rover either by placing it at a surveyed location or by placing it within one wavelength of the base station. Another method is known as Real Time Kinematic GPS (RTK GPS). With this method, the integer ambiguity is constantly trying to be resolved as the rover moves. If it is resolved, but then lock is lost to a satellite for some reason, cycle slip occurs, and the ambiguity must be resolved again. With ambiguities resolved, RTK GPS can be accurate to within 2 cm [7]. Without resolved ambiguities, RTK GPS is less accurate, around 10-30 cm [6]. Initialization of a carrier-phase solution can take up to 30-40 minutes.

2.4.3 Base station choices

There are three primary methods of obtaining differential GPS corrections.

1. Local Base Station - The first is to set up your own base station receiver and data link to the rover. This has the advantage of having a very small baseline distance between the base station and rover. Errors between two

receivers become less correlated the greater the baseline distance grows. As the receivers become farther apart, the signals to them from the same satellite pass through different areas of the atmosphere, resulting in differing ionosphere and troposphere delays. Additionally, the effects of the errors of the satellite ephemeris changes between the receivers with the increased baseline. For distances greater than 250 km, the error can be greater using DGPS than using models to estimate ionosphere and troposphere delay [7]. This is because the error correction for the rover includes all of the errors seen by the base station. Uncorrelated errors add to the error at the rover. Thus, the smaller the baseline, the greater the error correlation, and the better the measurement of rover position. The disadvantage of setting up a base station is that you need to know its position accurately in order to resolve errors. This can be achieved by making observations for a significant amount of time before using it as a base station or by placing the receiver at a pre-surveyed location.

2. NTRIP - The second option is to use a fixed nearby station from a reference station network. A number of these networks exist. In Switzerland, for example, there is the EUREF Permanent Network (EUREF) and the Automated GNSS Network Switzerland (AGNES) [8, 9]. Each of these networks is composed of a number of reference receiver stations, and each receiver's position is known very precisely. The receivers transmit data to a central server, which then transmits data to rover receivers via the internet. This is known as NTRIP, or Networked Transport of RTCM via Internet Protocol [10]. RTCM, or Radio Technical Committee for Maritime Standards, is the standard used for sharing DGPS and RTK correction data [11]. In an NTRIP network, sources (reference stations) transmit their data to servers, which then transmit it to casters. NTRIP casters make the data from all of the sources in their network available to users via the internet. This allows correction data to be used in real time. Casters also archive old data for post-processing.

3. SBAS - The third option is to use Wide Area DGPS (WADGPS) [7]. A WADGPS system has a number of spatially distributed reference receivers. The data from all of the receivers is coordinated to create correction information for a large area that does not depend on the distance to the nearest reference receiver. For many of these WADGPS systems, correction information is sent to rover receivers via satellite signals. This is known as a satellite-based augmentation system (SBAS). The advantage of using SBAS is that, because correction comes via satellite transmission, the rover receiver does not need to have an internet or radio connection to receive it. Additionally, the receiver can use the SBAS signal as another reference for calculating its position. The disadvantage is that the correction data cannot be used for carrier-phase differential GPS. In the US, there is the Wide Area Augmentation System (WAAS), while in Europe, there is the Europe Geostationary Navigation Overlay System (EGNOS).

2.5 RTKLIB

RTKLIB is an open source software library developed by Tomoji Takasu and Akio Yasuda of the Tokyo University of Marine Science and Technology for DGPS processing [12]. It provides both command line and GUI tools for streaming GPS measurement data from a variety of sources and computing position solutions. Data streams can be provided from serial port, tcp connection, NTRIP caster, or saved

files, and solutions can be output to the same type of streams. RTKLIB provides the following solution types:

1. Single - A single solution uses only data from one receiver to calculate its position.
2. Fixed - This algorithm assumes the receiver is stationary, and then averages over the whole measurement time to calculate the fixed position. Note that it does not use differential information.
3. DGPS - This algorithm performs code-based differential GPS. It requires a reference receiver.
4. Kinematic - This is the RTK algorithm. It uses carrier-phase DPGS to calculate the position of a moving rover relative to a base station.
5. Static - This is also a carrier-phase DGPS algorithm, but it makes the assumption that the rover is stationary. The solution provides the best guess at position over time, as opposed to the fixed solution, which just outputs the average value.
6. PPP Kinematic - Precise Point Positioning (PPP) uses very precise satellite clock and ephemeris data made available online 10-12 days after recording [14]. The kinematic version is for a moving receiver. It does not require a reference station, just the precise clock and ephemeris data.
7. PPP Static - The PPP algorithm again, but assuming a stationary receiver.
8. Moving Baseline - RTK solution where the base station is not assumed to be stationary.

RTKLIB can be used in real-time to generate GPS solutions or in post-processing. It has a large parameter space that can be tuned by the user to specific applications. As part of this project, I tried to determine what the best parameter set is for use on MAVs.

Chapter 3

Procedure

3.1 Test Setup

It is inherently difficult to get accurate ground truth for experimentation with GPS. To prove that sub-decimeter accuracy is achievable, one must know the actual location of the receiver to sub-decimeter accuracy. For static location tests, this can be achieved by putting the antenna at a previously surveyed location. For this project, we did not have access to highly accurate GPS equipment that could be used to survey antenna locations, nor were we aware of previously surveyed locations at our testing sites. Therefore, I was not able to test the absolute accuracy of the positioning solutions in this paper. Instead, I looked at the precision of the solutions for a stationary antenna. Given that the antenna is not moving, all measurements should be within a certain range of a nominal value. By measuring the deviation about the nominal value, the precision of the solution can be determined.
For a moving system, such as an MAV, this solution will not work. The precise path of the moving antenna must be known to determine the precision of the solution. Other researchers have had this problem as well. They address it by mechanically constraining the movement of the antenna to a known shape (e.g. on playground equipment [13]) or by moving it on a track with a known position [15]). Luckily, we had access to a highly accurate mobile surveying device made by Leica Geosystems. The Leica Total Station (Leica) provides millimeter-accurate range measurements to a reflective prism. Additionally, it measures the relative azimuth and elevation angles of the prism, yielding a 3D relative position. Furthermore, the Leica will automatically track the prism as it moves. It has been shown by researchers at ASL that this can be used to accurately track the position of a flying MAV. The Leica, therefore, was used to measure the relative position of the rover antenna as it moved. This allowed us to test the precision of the GPS solution for a moving rover without the need for a large mechanical system.
For the tests, the local base station antenna was mounted on top of the Leica. The rover receiver was mounted with the reflective prism, either on a stick, on a tripod, or on an MAV, depending on the test.

3.2 Equipment

For all of the tests, we used two receivers. The local base station was a uBlox LEA-6T receiver in an EVK-6T evaluation kit. The antenna was the standard uBlox ANN-MS. The rover receiver was also a uBlox LEA-6T. Provided by Ascending Technologies, it came mounted on a PCB with an integrated antenna. For RTK GPS, a receiver which provides raw measurement data, such as the LEA-6T, is

required. This is a newer generation of the receiver chip used by the authors of RTKLIB [12]. The MAV used for tests was an Ascending Technologies Firefly hexacopter. As was mentioned in the previous section, a Leica Total System was used to provide relative position measurements for ground truth [1].

3.3 Test Locations

Three different locations were used for testing. They were all chosen for their proximity to ETHZ and their (relatively) clear view of the sky. A fourth location (near ETH Hoenggerberg) was also used for some testing. However, one of the receivers used for this testing was later determined to be faulty, and so no results from this site are included. Figure 3.1 shows a map of these sites.

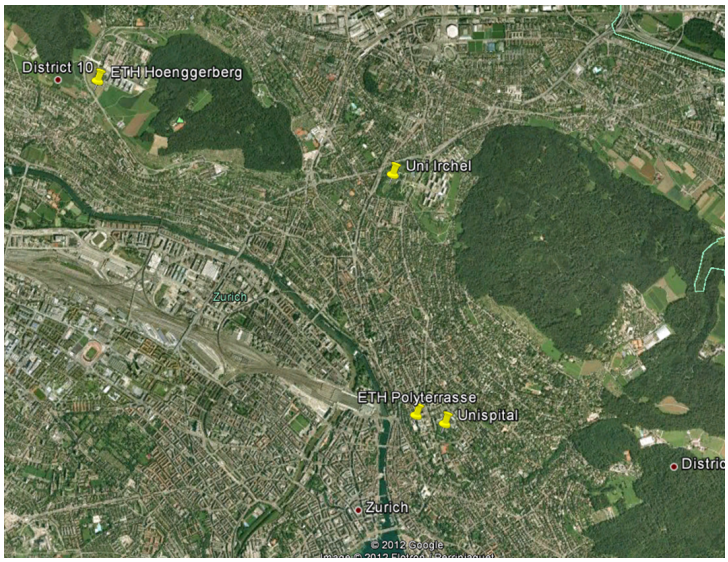

Figure 3.1: Data collection locations. Map created with GoogleMaps (http://maps.google.com).

1. ETH PolyTerrasse - This location provided a relatively clear view of the sky, as the only major obstruction is the Hauptgebaeude. The ground is reinforced concrete, as opposed to grass for the other two locations. This is more reflective to GPS signals, yielding potentially higher multipath errors. Additionally, the PolyTerrasse is frequently crowded, so we were unable to conduct tests with the MAV here.
2. Unispital - Tests were conducted in the small park beside the Unispital. The park is surrounded by trees, and therefore the satellite coverage is not as good as at other locations.
3. Uni Irchel - Finally, tests were conducted in the park between the Uni Irchel campus and the Milchbuck tram stop. This park is has wider open areas than the Unispital park, and therefore has better satellite coverage.

3.4 Data Analysis

For all of the tests, GPS solutions were generated offline using RTKLIB. The algorithms, except where noted in the results section, could be used for real-time

processing. Two NTRIP reference stations were used. The first (NTRIP Caster ZIM20) is located in Zimmerwald, Switzerland, approximately 110 km away from Zurich. This is the closest station that is part of the EUREF network. The second (NTRIP Caster ETH2) is located at ETH Hoenggerberg, approximately 4 km from the Uni Irchel test location. This station is part of the AGNES network, run by Swiss-topo, which does not normally provide free data access. We were able to get data from this station for analysis only for the testing at Uni Irchel. When calculating differential GPS solutions, the location of the base station receiver needs to be accurately known. The static reference stations publish their location along with measurement information. For the experiments using a local base station that we set up, the base station's position was first calculated using data from a static reference station, and then used to calculate the rover position.

Chapter 4

Results

4.1 Local Base Station

For the first set of analysis, I looked at the what is the best way to accurately determine the location of a stationary local base station. This can then be used, as described earlier, for RTK with the rover. The goal is to know the location as quickly and accurately as possible.

4.1.1 Solution Types

I first looked at how well different solution types converge, given a stationary receiver. Figure 4.1 shows the results for a single data set using all of the real-time solution methods available in RTKLIB. The solution types are described in the previous section. The data set was taken at Uni Irchel, and the NTRIP caster in Zimmerwald (ZIM20) was used as the reference station for the differential methods (DGPS, RTK, and static). All of the solution methods except for static and RTK have a significant amount of error, between 1 and 3 m RMS. This amount of noise does not allow the user to determine a single, stationary position for the receiver, which is necessary for its use as a base station. The fixed solution is essentially an average of the single solution position. This does not take into account any differential correction information, and as such, is likely less accurate than the differential solutions. The static solution uses a strong prior (assuming that the receiver is stationary), and as such, has the most stable output. Therefore, when determining the local base station localization, it is best to use the static solution. The drawback of this is that it requires access to NTRIP data. Without an internet connection, the best option would be to use the single solution with SBAS correction data and average the position of a long time.

4.1.2 Solution Convergence Time

Having determined that a static solution will yield the least noisy base station localization, I next looked at how quickly this solution type will converge to a stable position. This is important, because the local base station position needs to be known before it can be used as a reference station. Therefore, the longer it takes to converge, the earlier that the base station will need to be set up. Figure 4.2 shows the horizontal position (east and north) convergence time for four different tests. As can be seen, all the solutions are within 0.5 meters of their final value after 30 minutes. Therefore, one needs to set up the local base station receiver only 30-40 minutes ahead of time to let the static solution converge. The data from Unispital is noisier and takes longer to converge the than other data. This is because of the

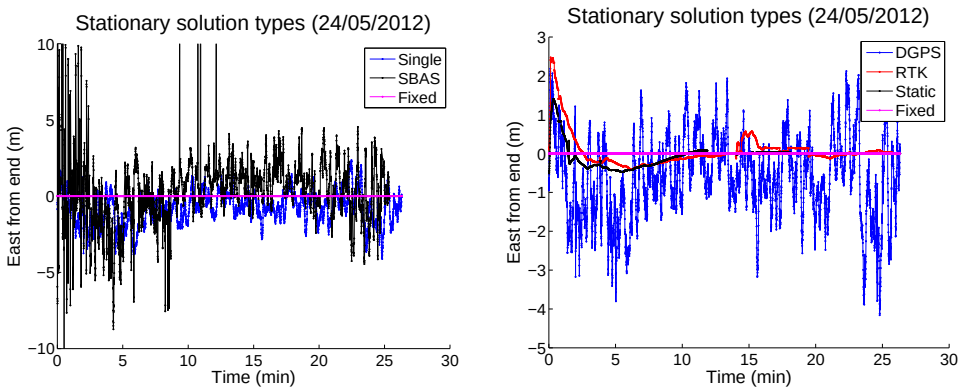

Figure 4.1: Solution types for stationary base station.

number of trees obstructing the clear view of the sky there. In a situation such as this, it would be prudent to allow 50-60 minutes for the solution to converge to a good value.

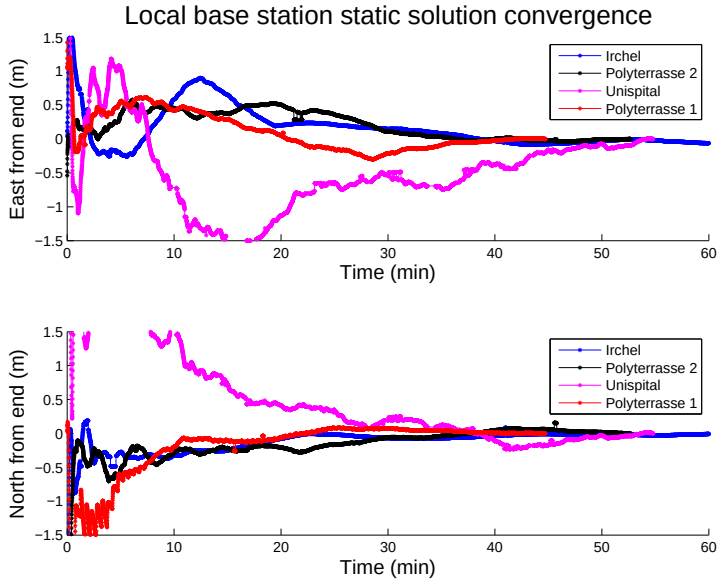

Figure 4.2: Convergence of static solutions for a stationary base station.

4.2 Rover Localization

Once the local base station position is determined, it can be used as the reference station for an RTK solution for the rover. I next looked at the quality of localization for both a stationary and a moving rover.

4.2.1 Base Station Selection

The first factor that I considered in rover solution quality was base station selection. Differential GPS should work better the closer the rover is to the base station. However, the quality of the base station receiver will effect the quality of the rover

solution. I therefore looked at using three different base stations: the local base station that we set up, the NTRIP reference station at ETH Hoenggerberg (ETH2), and the NTRIP reference station in Zimmerwald (ZIM20). I compared RTK solution quality with the rover sitting in a stationary position. The results are shown in figure 4.3. All three base stations yield similar solutions. The solutions all remain within a bound of approximately 2 m, except for the spike near the end of the data set. The local base station solution, however, is slightly worse, with a greater horizontal standard deviation (0.89 m), compared to ETH2 (0.72 m) and ZIM20 (0.74 m). This is likely do to the poor quality antenna and receiver used for the local base station, as well as the error in the measured position of this station.

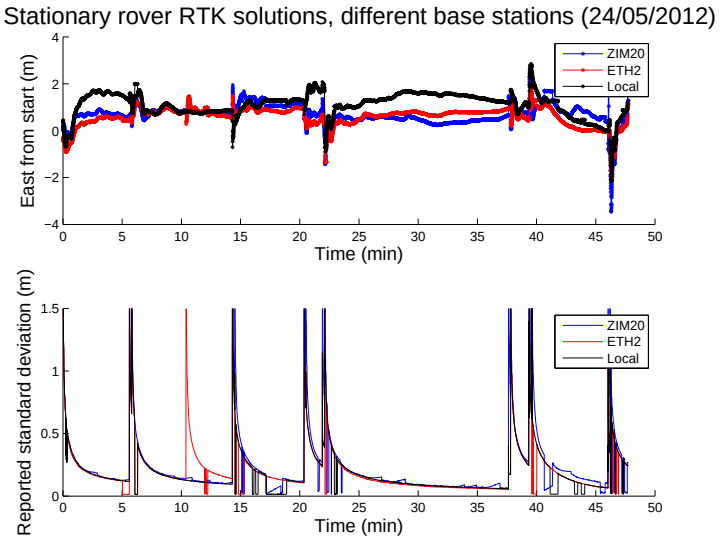

Figure 4.3: RTK solutions for a stationary rover using three different base stations.

The bottom plot in figure 4.3 shows the standard deviation of the solutions, as reported by RTKLIB. Note the spikes. These are likely caused by a loss of phase lock on a satellite for the rover receiver. When the phase lock is lost, a carrier phase measurement cannot be made, which in turn means that a good RTK solution cannot be calculated at this point in time. The standard deviation spikes and the rover position estimate jumps. This is observed regardless of which base station is selected, and is therefore a function of the rover receiver. The spikes cannot be eliminated. However, by monitoring the solution standard deviation, a state estimation system on an MAV can determine when a jump in position is likely caused by this loss of phase lock, rather than the rover actually moving, and filter accordingly.

4.2.2 Integer Ambiguity Resolution

Another important parameter in the quality of the rover RTK solution is integer ambiguity resolution. As explained in section 2.4, the range from the receiver to a satellite should be an integer number of wavelengths plus the measured phase. The initial RTK solution is calculated using a float number of wavelengths. If this float can be resolved to an integer number of wavelengths (integer ambiguity resolution), then the solution can be much more precise. However, the ambiguity resolution is lost (cycle slip) whenever the receiver loses phase lock to a satellite, it accelerates too quickly, or the received signals are too noisy.

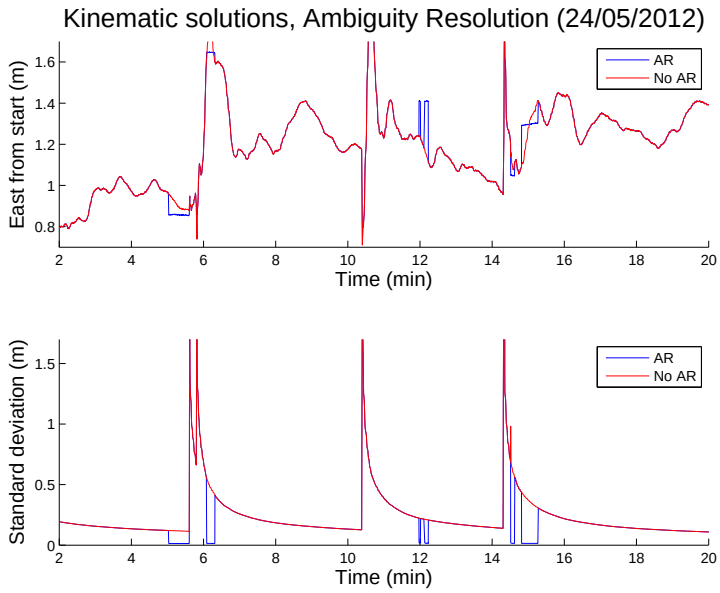

Figure 4.4: Kinematic solutions for a stationary rover with and without integer ambiguity resolution.

Figure 4.4 shows a comparison between a solution calculated with integer ambiguity resolution and one with it disabled. For this data set the rover was in a stationary location. What is immediately obvious is that the two solutions are nearly identical. This is because the ambiguity is resolved sporadically, and only for very short periods of time. This can be seen most clearly in the standard deviation plot. The ambiguity is resolved whenever the AR standard deviation drops to nearly zero, and cycle slip occurs when it returns to the same value as the No AR standard deviation. Note how infrequently this happens. This is likely due, in part, to the relatively poor quality antenna used on the rover. When ambiguity resolution happens, the solution position jumps and then jumps back when cycle slip occurs. As the figure shows, the solution is not necessarily more accurate, as the actual position does not change throughout the whole data set. Therefore, if the ambiguity resolution pulled the solution to a more accurate value, then each time that it is resolved, the solution should jump to approximately the same value. Instead, each time the ambiguity is resolved, the solution jumps to a slightly different position. This is likely due to noisy signals, as the observation data shows no drop in satellite visibility when cycle slip occurs. However, if the signals are too noisy, then the optimization which resolves the ambiguities cannot determine a solution and AR is lost. With a higher quality antenna, the data will be less noisy, and AR can likely improve solutions.
The discontinuous position jumps in the solution, coupled with sudden drop in standard deviation, will likely cause problems in the MAV state estimator that uses this output. Thus, I recommend turning off integer ambiguity resolution for the RTK solutions for current MAV application.

4.2.3 Moving Rover

In the end, the rover receiver will be mounted on a moving MAV, not one that sits in a stationary location. The next step is to look at solution quality for a moving rover. Relative ground truth for the rover was recorded by the Leica Total Station.

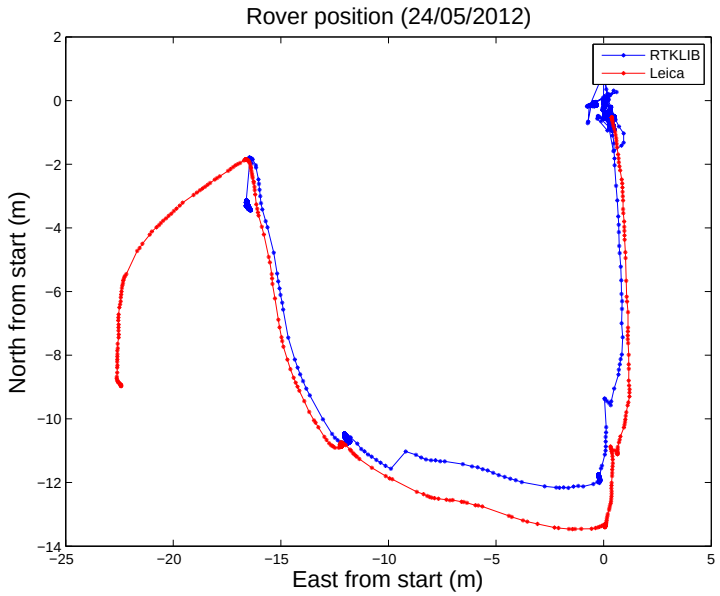

Figure 4.5: RTKLIB solution and Leica measured ground truth for a moving rover at Uni Irchel.

Figure 4.5 shows a comparison between the RTK solution and Leica ground truth for a data set taken on 24/05/2012 at Uni Irchel. It is important to note here there is some small amount of unknown error in the alignment between the Leica and RTK solution coordinate frames. The RTK solution is in local east-north -up (ENU) coordinates, with the origin at the local base station receiver. This receiver is mounted on the Leica, so the origins should be aligned. However, the Leica frame's axes are not aligned to east and north. This angular offset is estimated from the initial rover position. The rover was set in a stationary position for over half an hour. The ENU position was calculated with an RTK solution. Given this and the position measured with the Leica, the two reference frames were aligned. The RTK solution position is not exact, and so there is some amount of error in the frame alignment.

At first glance, the solutions appear to be well aligned for most of the dataset. The notable exception to this is the 2 meter offset between the RTKLIB solution and ground truth at the bottom of the figure. This could have occurred due to an uncorrelated error between the rover and the base station or a loss of satellite visibility. There was a grove of trees just south of the test location, and this offset occurs when the rover was closest to these trees. Looking at the observation data, there does not appear to be a loss of satellite visibility. Therefore, the most likely explanation is extra multipath error due to the trees.

Figures 4.7 and 4.8 show a similar comparison between an RTK solution and Leica ground truth for a data set taken on 30/04/2012 on the PolyTerrasse. Note that the solution is noisier than the one for the Uni Irchel data set. This is most probably due to significantly more multi-path error. The PolyTerrasse is made of concrete on top of a metal support structure, which will reflect electromagnetic waves at the GPS frequency much better than the grass and dirt ground in the park at Uni Irchel. More reflected signals reach the receiver, resulting in a greater error.

Table 4.1 shows the absolute and RMS errors in position (relative to Leica ground truth) for both tests. Note that horizontal error is listed separately from vertical error, as GPS solutions tend to be less accurate in vertical position. The horizontal

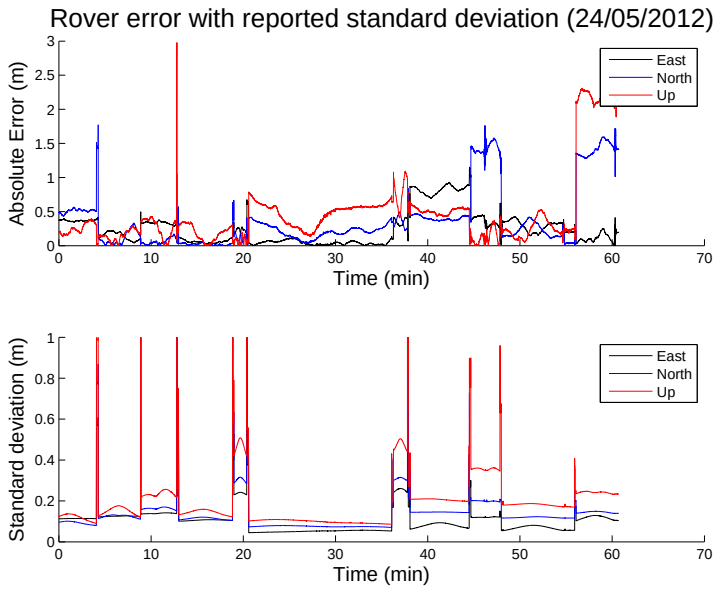

Figure 4.6: Absolute error compared to Leica Ground Truth and the standard deviation reported by RTKLIB for a moving rover at Uni Irchel.

Table 4.1: RTK Real-time Solution Position Error.

	Uni Irchel (24/05/2012)	PolyTerrasse (30/04/2012)
Max Horizontal Error (m)	3.06	6.80
RMS Horizontal Error (m)	0.80	1.02
Max Vertical Error (m)	5.89	7.90
RMS Vertical Error (m)	1.04	2.66

RMS error is approximately 1 m for both data sets. The vertical RMS, however, is 2.5 times greater for the PolyTerrasse data set.

4.2.4 Post-Processing

In addition to real-time localization, it is useful for MAV research to get accurate ground truth through post-processing. Post-processing the RTK GPS solutions has a few advantages over real-time solutions. First, the algorithm can be run both forwards and backwards in time, which can significantly smooth the solution. Second, the user that is post-processing the data can choose to exclude some satellites to improve the solution. For example, if one satellite is seen only intermittently, then every time the receiver loses view of the satellite, the solution position will jump. Excluding this satellite from processing will prevent this from happening. Finally, different sources publish ionosphere, troposphere, satellite ephemeris, and satellite clock correction information based on logged data, which can be used to improve the solution.

Figures 4.9 and 4.10 show real-time and post-processed solutions for the two data sets analyzed in the previous section. Note how the post-processed solutions are significantly smoother. The solution position transients that occur with phase-lock loss are mostly eliminated. Though the position value still changes (while the actual

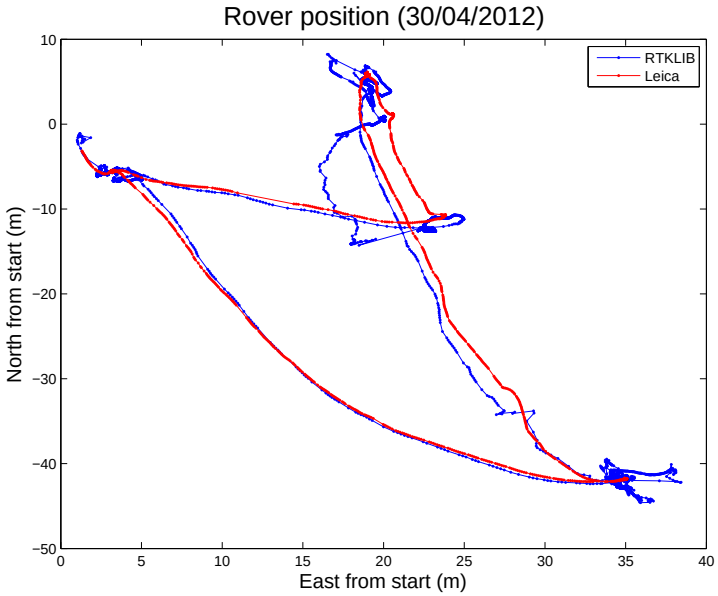

Figure 4.7: RTKLIB solution and Leica-measured ground truth for a moving rover on the ETH PolyTerrasse.

position remains stationary), it immediately settles on the new position estimate.

Table 4.2: RTK Post-processed Solution Position Error.

	Uni Irchel (24/05/2012)	PolyTerrasse (30/04/2012)
Max Horizontal Error (m)	1.82	6.52
RMS Horizontal Error (m)	0.71	1.10
Max Vertical Error (m)	2.98	7.60
RMS Vertical Error (m)	0.70	2.87

Table 4.2 shows the absolute and RMS errors for the two data sets using post-processing solutions. Post-processing only had a minor effect on the error. While the data is smoother than the real-time solution, it is not much more accurate.

4.3 Rover receiver on MAV

For another test, the rover receiver was mounted on an MAV. The MAV was first set in stationary locations, and then flown around. As the goal of this project is to ultimately calculate MAV position, it is crucial to determine whether any problems arise in mounting the rover receiver on an MAV, rather than a tripod.

4.3.1 Influence of Noise

When analyzing the data from the MAV test, it became immediately apparent that there was a problem. The quality of the RTK solution, whether the local base station or ZIM20 reference station was used, was significantly worse than the quality of solution for the tripod mounted receiver. The solution had horizontal position errors of greater than 5 meters. Figure 4.11 shows a comparison between the RTK

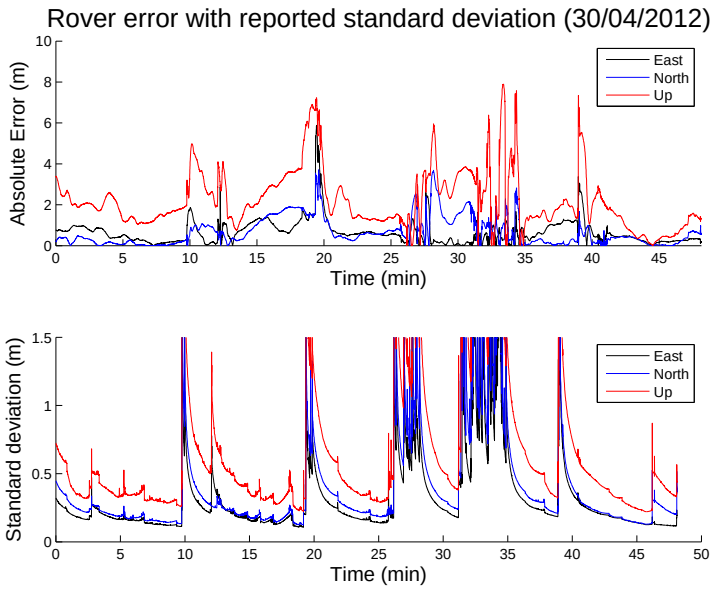

Figure 4.8: Absolute error compared to Leica ground truth and the standard deviation reported by RTKLIB for a moving rover on the ETH PolyTerrasse.

solution and the solution produced by the uBlox receiver. Note that both vary by a significant amount, and around UTC 14:37, the solution jumps by over 50 meters and is lost for a few minutes. The uBlox solution is significantly smoother than the RTK solution. This is because the uBlox receiver filters the positions using a well-tuned motion model. This yields a much smoother, but not necessarily more accurate, solution.

The reason for the poor quality of the RTK solution is noise in the GPS frequency band produced by the MAV. The most likely noise source is the on-board processor, which has a clock rate close the GPS L1 carrier frequency. Figure 4.12 lends credence to this explanation. It shows the signal to noise ratio (SNR) of the GPS signals received by the local base station. At approximately UTC 14:00:00, the center of the MAV was moved to within 15-20 cm of the base station antenna. The SNR drops significantly, and then rises again when the MAV was moved away 30 seconds later.

This noise problem effectively renders the RTK solution unusable. As such, it must be solved before RTK GPS can be used on this type of MAV. The simplest solution is to get a separate antenna and mount it on a mast above the MAV. It can hopefully be mounted in a position such that the noise is no longer an issue. New antennas were ordered, but they did not arrive in time to be tested.

4.3.2 Data Loss

In addition to the high level of noise, I discovered that the rover receiver dropped GPS measurements. Each GPS position solution is recorded with the corresponding UTC time of the raw measurements used to calculate the solution. Figure 4.13 shows the time between subsequent measurements for local base station and rover receivers. Further investigation revealed this data dropout was not due to RTKLIB solution processing errors, as the raw observations display the same pattern of missing data. Additionally, the data dropout was observed whenever the rover receiver was used, both when it was mounted on the MAV and when it was mounted on a

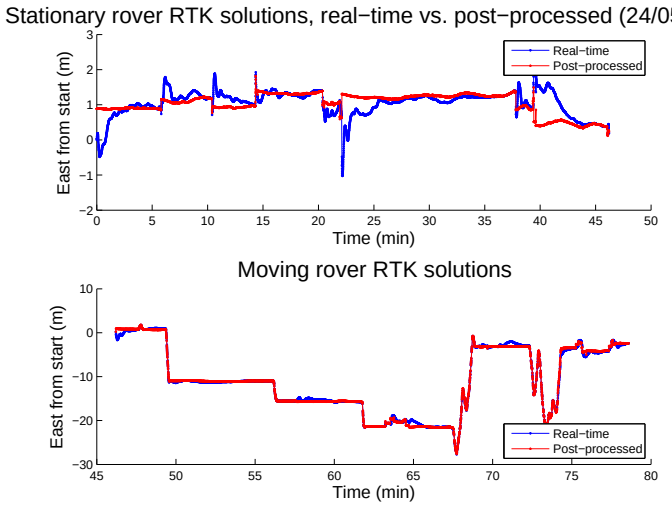

Figure 4.9: Real-time and post-processed solutions for the Uni Irchel data set.

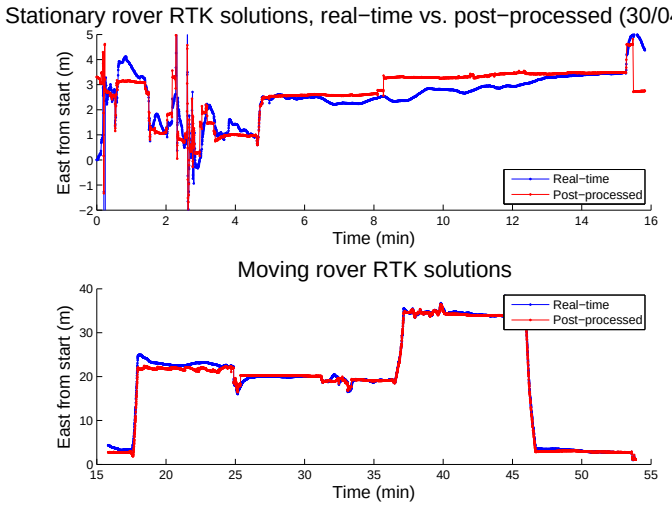

Figure 4.10: Real-time and post-processed solutions for a PolyTerrasse data set.

tripod. It also did not matter whether the raw data were recorded using the ROS interface or the RTKLIB GUI. Therefore, I can conclude that the problem likely lies somewhere in the hardware or the firmware configuration. It is important to resolve this problem before attempting to use RTK GPS in for real-time localization on an MAV. Having seemingly random delays in new position calculations of up to 1.5 seconds is non-optimal for MAV position estimation.

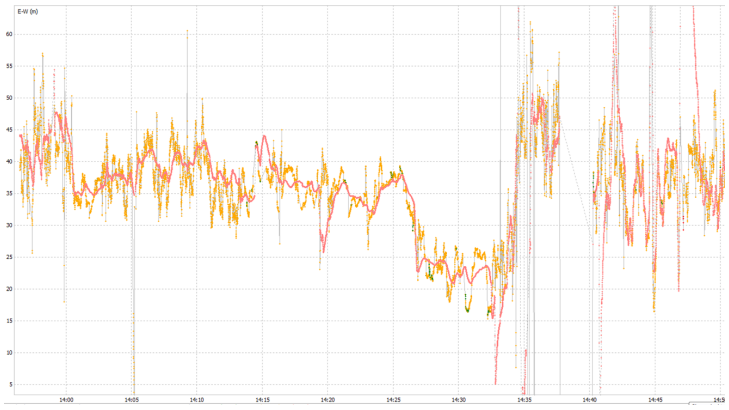

Figure 4.11: Comparison of RTKLIB and uBlox solutions for the receiver mounted on the MAV.

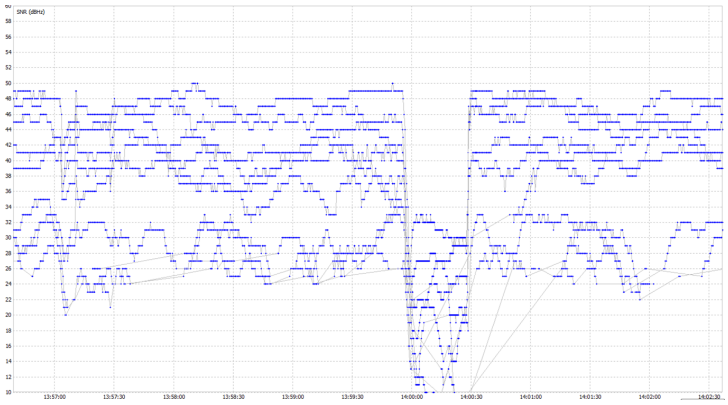

Figure 4.12: Signal to noise ratio for the base station receiver. Note the sharp dip at 14:00:00 when the MAV placed near it.

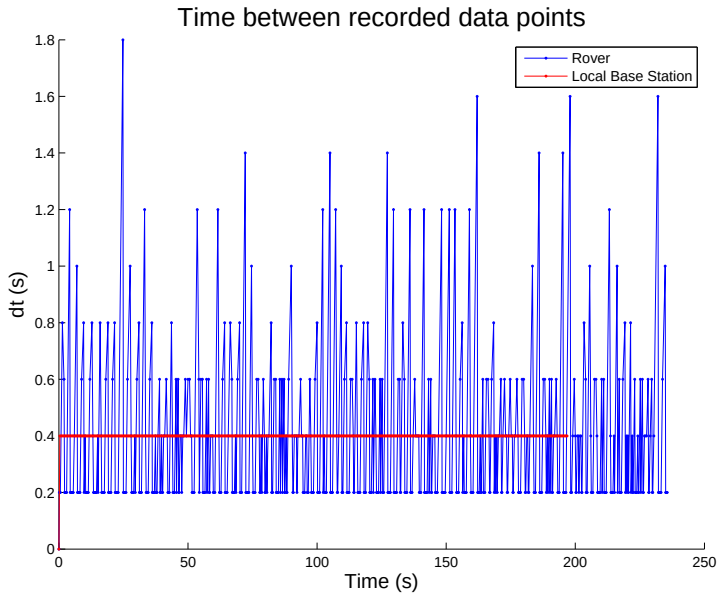

Figure 4.13: Time difference between successive GPS measurements.

Chapter 5

ROS Implementation

As part of this project, I created a ROS wrapper for RTKLIB. Many of the projects in the Autonomous Systems Lab at ETH use ROS, and so, having a ROS wrapper for RTKLIB allows RTK GPS solutions to be easily integrated into other lab projects. I started with a basic implementation provided by Simon Lynen, and then modified it to be more easily used.

RTKLIB is compiled such that two different node types can be launched via ROS launch files. The first is a measurement stream node. This node will read data from an input stream, such as a serial port, and send it to an output stream, such as a tcp socket. This is useful for streaming data from one processor connected to a receiver to another on which the RTKLIB server is running to compute the GPS solution. For example, this node can stream data from the receiver on board an MAV to a ground station for processing.

The second node type is the RTKLIB server. The server receives data streams and computes the solution. I modified the RTKLIB server so that solution parameters can be loaded on launch via a .conf file or a .yaml file. A .conf file can be generated from the RTKLIB GUI, recording the current solution parameters. If one uses the RTKLIB GUI to post-process data and determine a desired set of parameters, they can be saved to a .conf file and loaded when the server node is launched. Solution parameters can also be specified with a .yaml file. The .yaml file loads solver parameters to the ROS parameter server. This allows a user to display the current solution parameters via ROS. The parameters can also be modified on the parameter server. Once changes are made, they must be loaded into the RTKLIB server node via two ROS service callbacks. Parameters are fetched from the parameter server with rtklib_fetch_params. The RTKLIB server must then be reset using rtklib_reset to start using the new parameters. Finally, the RTKLIB server node publishes the solution via two ROS topics. One topic, /baseline, gives the position of the rover relative to the base station. The other topic, /latlon gives the exact latitude, longitude, and height of the rover.

RTKLIB uses an Extended Kalman Filter (EKF) to generate its solutions. One option for RTKLIB is to use a simple model of receiver dynamics in the EKF to try to help improve the solution. Solutions could be improved by providing acceleration or velocity measurements to this filter, rather than just estimating the acceleration from the GPS position updates. I looked at how the EKF implemented and wrote a sample function to allow velocity measurements to be incorporated. This function has not been tested with real or simulated data, but it should hopefully provide a starting point for future development.

Chapter 6

Conclusion

Real time kinematic GPS is feasible on an MAV. While it is not as accurate as RTK GPS with high quality expensive receivers, it does improve upon the single receiver solutions.
The first use case for MAV is real-time localization. For this task, RTKLIB seems adequate.

- In a low multi-path environment, the horizontal RMS error for an RTK solution was only 0.8 m. In a higher multi-path environment, the RMS error for RTK was 1.0 m. These are both significantly lower than the 10-15 m expected for a single receiver solution. The vertical error is worse (1.0 m and 2.7 m). This is to be expected from GPS, and an additional altitude sensor should be used for state estimation on an MAV.

- RTKLIB publishes the current solution covariance matrix. If a user wants to integrate the RTKLIB output into a real-time state estimator, then knowing this covariance matrix ensures that when RTKLIB is unsure of the solution, it does not count as much in the estimator. When trying to incorporate GPS solutions from a standard receiver, there is no published covariance, and thus good quality solutions cannot be distinguished from poor quality solutions.

- Using a reference station from an NTRIP network, such as EUREF and AGNES, provides slightly better solutions than using a local base station set up by the user. This difference in precision is most likely due to the difference in the quality of antennas and receivers used for the reference stations. The local base station used in this test was a cheap receiver with a cheap antenna. A higher quality antenna would likely improve the solution. That being said, the difference in performance was not large (0.7 m vs. 0.8 m RMS error in a stationary position), and so either solution is adequate. It may be possible to improve this somewhat by using a base station receiver with a high quality antenna as close to the rover receiver as our local base station.

- The measurement of the local base station's position converges to a stable, usable value in 30-40 minutes, when using a static solution type with an NTRIP reference station. This means that, with some internet access, one does not have to go into the field more than 40 minutes ahead of time to get a good measurement of the base station position.

The second use case for the MAV is post-processed ground truth. Though post-processing the data smooths out the solutions, it does not necessarily make them more accurate. The RMS error is almost the same for the real-time and post-processed solutions. For some tasks, 1 m level accurate ground truth may be

sufficient. However, for most MAV tasks, this is likely not accurate enough to use as the sole measure of ground truth. This could likely be improved by using a higher quality antenna with the rover receiver.
Problems arose when the receiver was mounted on the MAV. These were most likely due to noise from the processor. A better GPS antenna and mounting location will hopefully mitigate this problem. Without some noise mitigation, RTK GPS will perform worse than the solution generated by the uBlox receiver.

Appendix A

RTKLIB ROS Documentation

Here is a copy of the documentation that will be included in RTKLIB ROS. It is also available in the rtklibros ROS package.

```
ROS Wrapper for RTKLIB GPS processing library.
Wrapper authors:  Daniel Grieneisen (gridanie@student.ethz.ch)
                  Simon Lynen (simon.lynen@mavt.ethz.ch)
Version 0.1
Date:  June 01, 2012

RTKLIB:
    -Website: http://www.rtklib.com/
    -Authors:  T. Takasu and A. Yasuda
    -version: 2.4.1

================== INTRODUCTION ================================

RTKLIB is an open source software library for GPS processing.  The library
comes with command line and GUI programs for a variety of real-time and post-
processing GPS tasks.  For more information, as well as the original library,
please see http://www.rtklib.com.  This is a ROS wrapper for the library, designed
so that the RTK server can be started via launch files and solver parameters can
be set using the ROS parameter server.

For an overview of RTKLIB, visit http://www.rtklib.com or read the paper
Takasu_RTKLIB.pdf in the documentation folder.

For a brief background on differential GPS, as well as test results geared
towards using RTKLIB with an MAV, please see the paper Grieneisen_SA_Final.pdf in
the documentation folder.

================== GPS MEASUREMENT NODE ========================

This node is used to stream GPS measurement data from a receiver to the
RTKLIB server.

To start the "rover" gps measurements, launch gps_meas.launch.
This will start a service reading from COM and publishing data to TCP port 8600.
```

```
===> adapt parameters such as serial port config in the launchfile


================ RTKLIB SERVER NODE ============================

This node starts an RTKLIB server for processing gps data and publishing
solution results.

To start the RTKLIB server, use one of the following launch files:
-rtkrcv_conf.launch
-rtkrcv_param.launch

These two launch files demonstrate two different methods for setting the
RTKLIB solver parameters.

----------------- rtkrcv_conf.launch ----------------------
This launch file loads parameters via an RTKLIB .conf file.  If you analyze data
via the RTKLIB GUI tools, solution parameters can be saved in a .conf file.  Using
this launch file, you can load parameters this way.

===> you probably want to adjust the settings in ./conf/rtk_server.conf

IMPORTANT VALUE CHANGES: (flagged with @ADAPT in the config file)

line 68: ant2-pos1  (deg|m) = LAT STATION2 POSITION
line 69: ant2-pos2  (deg|m)  = LON STATION2 POSITION
line 70: ant2-pos3  (m|m) = HEIGHT STATION2 POSITION

line 89: inpstr1-type = ROVER MEAS SOURCE
line 90: inpstr2-type = BASE MEAS SOURCE
line 97: inpstr1-path = ROVER MEAS SOURCE PARAM
line 98: inpstr2-path = BASE MEAS SOURCE PARAM

----------------- rtkrcv_param.launch ----------------------
This launch file loads parameters via a .yaml file.   Example yaml file for the
following use cases are provided in the params directory.

  -BaseStatic.yaml:  Runs static solution algorithm for a stationary receiver.
        Designed to determine a local base station position as quickly as possible
        for use with a rover.  Position convergence time at least 20-40 minutes.
  -RoverSingle.yaml:  Runs SBAS algorithm for a rover receiver.  Note that receiver
        must be configured to receive SBAS data for this to work.  Otherwise,
        a single solution is calculated.
  -RoverBase.yaml:  Runs RTK algorithm for a rover receiver sending data over TCP
        and a local base station connected via serial.

The yaml files are organized so that parameters that most likely need to be
adapted to get up and running are at the top, and those that likely do not
need to be changed are at the bottom.  These can be modified to improve solution
quality.  However, the default values seem to produce acceptable results.

Scroll down to see a summary of parameters that should be adapted.
```

```
----------------- ROS Services -------------------------------
The following two rosservice callbacks are available:
  -rtklib_fetch_params:  Fetches new parameters from the parameter server.  Note
        that the RTKLIB server does not start solving with these parameters until
        it is reset.
  -rtklib_reset:  Resets the RTKLIB server, incorporating newly loaded parameters

----------------- ROS Topics --------------------------------
The following ROS message topics are published:
  -/baseline - position (ENU) relative to base station (PoseWithCovarianceStamped)
  -/latlon - solution latitude, longitude, altitude (PoseWithCovarianceStamped)


----------------- Parameter summary -------------------------
Here are the parameters that should be adapted for different uses.  Note that
parameter names are "foo_bar" if set in a .yaml files, but "foo-bar" if set in
a .conf file.


## Solver parameters
pos1_posmode: static
# (0:single,1:dgps,2:kinematic,3:static,4:movingbase,5:fixed,6:ppp_kine,7:ppp_static)
    -- Set the solution mode.  Solution modes are as follows:
        -single:  No base station needed.
        -dgps:  Code-based differential GPS.  Needs a base station.
        -kinematic:  RTK differential GPS.  Needs a base station.
        -static:  static differential GPS.  Assumes rover is not moving.  Needs
            a base station.
        -movingbase:  RTK GPS.  Does not assume base station is stationary.
        -fixed:  Outputs best estimate of fixed location.  No base station needed.
        -ppp_kine:  Precise Point Positioning algorithm for a moving receiver.
            Needs high quality ephemeris and clock data.
        -ppp_static:  Precise Point Positioning algorithm for a stationary receiver.
            Needs high quality ephemeris and clock data.

pos1_frequency: l1         # (1:l1,2:l1+l2,3:l1+l2+l5,4:l1+l2+l5+l6,5:l1+l2+l5+l6+l7)
    -- Set the frequency bands of the receivers.  L1 is used for most cheap
        receivers, L1+L2 for dual frequency, etc.
pos1_soltype: forward    # (0:forward,1:backward,2:combined)
    -- Set the solution direction in time.  Must be set to forward for real-time.
        combined runs the solver forwards and then backwards for smoothing a
        solution in post-processing.
pos1_elmask: 15         # (deg)
    -- Set the minimum elevation for a satellite to use in the solution.
pos1_snrmask: 0          # (dBHz)
    -- Set the minimum SNR for a satellite to use in the solution.
pos1_dynamics: !!str off        # (0:off,1:on)
    -- Set to include a simple model of receiver dynamics in the solution filter.
        ** NOTE:  must use "!!str off" and "!!str on" instead of "off" or "on".
                Without '!!str', this will get interpreted by ROS as a boolean
                instead of a string, which is what it should be.
pos2_armode: continuous # (0:off,1:continuous,2:instantaneous,3:fix_and_hold)
    -- Set how integer ambiguities should be resolved.  When using 'static' or
        'kinematic' mode, solution is calculated using carrier wave phase
```

```
        measurement.  Distance to a satellite is the phase measurement plus an
        integer number of wavelengths.  The solution is initially calculated with
        a floating point number of wavelengths.  Resolving this float to an int
        for each of the satellites is called ambiguity resolution.  When resolved,
        solutions can be much more precise.  However, AR can be lost if the
        receiver moves to quickly or loses view of a satellite.  Whenever
        ambiguities are resolved or lost, the solution position tends to jump.
        Using AR allows higher precision solutions when ambiguities can be
        resolved, but turning it off avoids these position jumps.

## Input streams

# Rover input

inpstr1_type: serial     # ROVER
(0:off,1:serial,2:file,3:tcpsvr,4:tcpcli,7:ntripcli,8:ftp,9:http)
    -- Set the input stream type for the rover receiver.

inpstr1_path: 'ttyACM0:57600:8:n:1:off' #
    -- Set the stream path for the rover receiver.  Path formats are as follows:
        (specified in src/stream.c)  [] are optional parameters
        -serial: port[:brate[:bsize[:parity[:stopb[:fctr]]]]]
                    port  = COM?? (windows), tty??? (linuex, omit /dev/)
                    brate = bit rate     (bps)
                    bsize = bit size     (7|8)
                    parity= parity       (n|o|e)
                    stopb = stop bits    (1|2)
                    fctr  = flow control (off|rts)
        -file: file_path[::T][::+start][::xseppd][::S=swap]
                    ::T   = enable time tag
                    start = replay start offset (s)
                    speed = replay speed factor
                    swap  = output swap interval (hr) (0: no swap)
        -tcpsvr:  :port
        -tcpcli:  address:port
        -ntripcli: [user[:passwd]]@address[:port][/mountpoint]
        -ftp:  [user[:passwd]]@address/file_path[::T=poff[,tint[,toff,tret]]]
        -http:  address/file_path[::T=poff[,tint[,toff,tret]]]
                    poff  = time offset for path extension (s)
                    tint  = download interval (s)
                    toff  = download time offset (s)
                    tret  = download retry interval (s) (0:no retry)

inpstr1_format: ubx
# (0:rtcm2,1:rtcm3,2:oem4,3:oem3,4:ubx,5:ss2,6:hemis,7:skytraq,8:gw10,9:javad,15:sp3)
    -- Set the rover stream format.  Supports a variety of different manufacturers'
        formats (e.g. ubx = uBlox).

# Base station input

inpstr2_type: ntripcli
# BASE (0:off,1:serial,2:file,3:tcpsvr,4:tcpcli,7:ntripcli,8:ftp,9:http)
    -- Set the base station input stream type.
```

```
inpstr2_path: 'user:pword@www.euref-ip.net:2101/ZIM20' #
    -- Set the base station stream path (see above for options).

inpstr2_format: rtcm2
#(0:rtcm2,1:rtcm3,2:oem4,3:oem3,4:ubx,5:ss2,6:hemis,7:skytraq,8:gw10,9:javad,15:sp3)
    -- Set the base station stream format.

# Base station location
    -- Set basestation location
ant2_postype: rtcm        # (0:llh,1:xyz,2:single,3:posfile,4:rinexhead,5:rtcm)
    -- Set the base station location type.
        -llh:  latitude, longitude, height
        -xyz:  xyz in ECEF
        -single:  calculate position via single solution
        -posfile:  get position from a .pos file
        -rinexhead:  get position from the head of a rinex file.  This is used in
            post-processing, as most NTRIP stations allow datalogs to be downloaded
            using the RINEX file format, which includes the position in the header.
        -rtcm:  get the position from an RTCM stream.  If the base station stream
            is using the RTCM format, then it may include the position as part of
            the data that is sent (e.g. NTRIP casters include this in their data
            streams).  Uses this published position.

ant2_pos1: 0  # (deg|m) # BASE POSITION
    -- Set latitude (llh) or x (xyz) position of base station.

ant2_pos2: 0  # (deg|m) # BASE POSITION
    -- Set longitude (llh) or y (xyz) position of base station.

ant2_pos3: 0  # (m|m) # BASE POSITION
    -- Set height (llh) or z (xyz) position of base station.

## Output streams (separate from ROS messages)

outstr1_type: tcpsvr
# (0:off,1:serial,2:file,3:tcpsvr,4:tcpcli,6:ntripsvr)
    -- Set first output stream type

outstr1_path: ':@:8601/:'
# SOLUTION IS OUTPUT HERE AND via ros @ topic "/baseline"
    -- Set first output stream path

outstr1_format: enu        # (0:llh,1:xyz,2:enu,3:nmea)
    -- Set first output stream format


## Logging output streams

logstr1_type: file
# (0:off,1:serial,2:file,3:tcpsvr,4:tcpcli,6:ntripsvr)
    -- Set type of logging stream for rover data.

logstr1_path: '$(find rtklib)/logs/%Y%m%d_%h%M_rover_rtk_server'
    -- Set path for rover data logging.
```

```
logstr2_type: file
# (0:off,1:serial,2:file,3:tcpsvr,4:tcpcli,6:ntripsvr)
    -- Set type of logging stream for base station data.

logstr2_path: '$(find rtklib)/logs/%Y%m%d_%h%M_base_rtk_server'
    -- Set path for base station data logging.
```

Bibliography

[1] *Leica Viva TS15 Datasheet*. Leica Geosystems AG. 05/06/2012. http://www.leica-geosystems.com/downloads123/zz/tps/Viva%20TS15/brochures-datasheet/Leica%20Viva%20TS15%20Datasheet_en.pdf.

[2] "GPS Accuracy." *Earth Measurement*. Earth Measurement Consulting. 25 May 2012. http://earthmeasurement.com/GPS_accuracy.html.

[3] S. Pace, et al: *The Global Positioning System: Assessing National Policy*. RAND Corporation. 1995. http://www.rand.org/pubs/monograph_reports/MR614.html.

[4] J. Clynch: "GPS Accuracy Factors." *Department of Oceanography, NPS*. July 2000. Naval Postgraduate School. 25 May 2012. http://www.oc.nps.edu/oc2902w/gps/accfact.html.

[5] "The GPS System." *Kowoma*. 19 Apr. 2009. http://www.kowoma.de/en/gps/errors.htm.

[6] "Introduction to the Global Positioning System for GIS and TRAVERSE". *CMTInc*. 1996. Corvallis Microtechnology. 25 May 2012. http://www.cmtinc.com/gpsbook/.

[7] R. Sabatini, G.B. Palmerini: *Differential Global Positioning System (DGPS) for Flight Testing*. In RTO-AG-160 AC/323(SCI-135)TP/189 Volume 21, NATO Research and Technology Organization. ¡ftp://ftp.rta.nato.int//PubFullText/RTO/AG/RTO-AG-160-V21/¿.

[8] *EUREF Permanent Network*. 19 Apr. 2012. EPN Central Bureau. 25 May 2012. http://www.epncb.oma.be/.

[9] *Agnes*. 18 Aug. 2010. Federal Office of Topography (swisstopo). 25 May 2012. http://www.swisstopo.admin.ch.

[10] E. Lenz: *Networked Transport of RTCM via Internet Protocol - Application and Benefit in Modern Surveying Systems*. From FIG Working Week 2004, Athens, Greece, 2004. http://www.fig.net/pub/athens/papers/ts03/ts03_2_lenz.pdf.

[11] M.Chivers: "Differential GPS Explained". *ArcUser Online*. ESRI. 25 May 2012. http://www.esri.com/news/arcuser/0103/differential2of2.html.

[12] T. Takasu: *RTKLIB: Open Source Program Package for RTK-GPS*. FOSS4G 2009 Tokyo, Japan, November 2, 2009.

[13] D. Odijk, et al: *Two Precision GPS Approaches Applied to Kinematic Raw Measurements of Miniaturized L1 Receivers*. In 20th International Technical Meeting of the Institute of Navigation Satellite Division, ION GNSS 2007.

[14] B. Witchayangkoon: *Elements of GPS Precise Point Position.* Ph.D. dissertation, The University of Maine, Dec. 2000.

[15] W. Stempfhuber, M. Buchholz: *A Precise Low-Cost RTK GNSS System For UAV Applications.* In International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. XXXVIII-1/C22. Presented at UAV-g 2011, Conference on Unmanned Aerial Vehicle in Geomatics, Zurich, Switzerland.