CHALMERS
UNIVERSITY OF TECHNOLOGY

Real-time kinematic positioning of UAS - possibilities and restrictions

Master's thesis in Complex Adaptive Systems

SVEN ERIKSSON

Department of Earth and Space Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2016

Master's thesis 2016

Real-time kinematic positioning of UAS - possibilities and restrictions

SVEN ERIKSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Earth and Space Sciences
Space Geodesy and Geodynamics
Chalmers University of Technology
Gothenburg, Sweden 2016

Real-time kinematic positioning of UAS - possibilities and restrictions
SVEN ERIKSSON

Supervisor: Thomas Hobiger, Department of Earth and Space Sciences
Supervisor: Rüdiger Haas, Department of Earth and Space Sciences
Examiner: Thomas Hobiger, Department of Earth and Space Sciences

Master's Thesis 2016
Department of Earth and Space Sciences
Space Geodesy and Geodynamics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Picture of the quadcopter IRIS+ with the Emlid Reach antenna set up for flight tests.

Typeset in LaTeX
Gothenburg, Sweden 2016

Real-time kinematic positioning of UAS - possibilities and restrictions
SVEN ERIKSSON
Department of Earth and Space Sciences
Chalmers University of Technology

Abstract

A pair of RTK capable GNSS receivers (Emlid Reach) and related software (RTKLIB) are used in order to get better position estimates for UAS (unmanned aircraft system) applications. Necessary correction data are either sent in real-time between the receivers through a network realized by two Raspberry Pis or downloaded after a flight for post-processing. Performed test on the ground show an accuracy and precision of a few centimetres when raw data was post-processed.
Ground based tests were performed with a professional grade RTK receiver as reference or with the tested receiver moving in a known pattern. The used software reports good results in for flight tests but that has not been verified due to lack of a moving reference. A simple application for UAS and RTK was achieved and demonstrated.

Keywords: UAS, RTK, Emlid Reach, drone.

Acknowledgements

I would like to thank my supervisors Thomas Hobiger and Rüdiger Haas for aiding me in various way during this project. I would also like to thank Joakim Strandberg and the other Ph.D. students at the Onsala Space Observatory for the aid they provided during different tests. I would also like to thank the staff at the workshops at the observatory for making the equipment I needed and for letting me use their equipment.

Sven Eriksson, Gothenburg, december 2016

Contents

List of Figures

List of Tables

1

Introduction

1.1 Background

A good position accuracy is useful or even necessary for many applications. Some applications related to UAS (Unmanned Aircraft Systems) are navigation and mapping. For navigation the position estimations are required in real-time while for other applications it might be enough to be able to produce good position estimations after the flight.

Nowadays good position accuracy is often realized with expensive differential GNSS (Global navigation satellite system) equipment. One of the used techniques is Real-Time Kinematic.

During the last couple of years using UAS (Unmanned Aircraft Systems) or drones as a hobby has become increasingly popular. This has pushed the development of cheaper components and now also GNSS receivers capable of Real-time Kinematic.

The Earth and Space Sciences department does currently own an IRIS+ quadcopter from 3D Robotics and a S900+ hexacopter from DJI. Both of these are controlled by Pixhawk autopilots running the autopilot software ArduCopter. While they do currently have ordinary GPS receivers for navigation better position estimation would allow for new applications.

1.2 GNSS (Global Navigation Satellite Systems)

Will utilize the American GPS and the Russian GLONASS. In addition to these two there are a few others GNSS systems in development. The Chinese BeiDou and the European Galileo satellite systems are expected to become fully operational in the next 10 years.

1.2.1 GPS (Global Positioning System)

The GPS is operated by U.S. Department of Defence. It has been operational since mid 1990s and it uses 24 satellites in 6 orbital planes. The orbital plane inclination for the satellites is 55 degrees. The satellites move in an almost circular orbit with a semi-major axis of 26 578 km [2].

The GPS satellites have three carrier frequencies over which it transmits its data: L1 (1575.42MHz), L2 (1227.60MHz) and L5 (1176MHz). Pseudo random noise (PRN) code, satellite clock and other data superimposed onto L1, L2 and L5. These are used by the receiver for positioning [2].

1.2.2 GLONASS (Globalnaya navigatsionnaya sputnikovaya sistema)

The GLONASS is operated by the Ministry of Defense of the Russian Federation and it uses 21 satellites spread out in 3 orbital planes. There are 3 spare satellites in orbit. The satellites move in an almost circular orbit with semi-major axis of 25 510 km and an orbital inclination angle of 64.8 degrees. [2].

The system has two frequency bands for its carrier frequencies: 1602 - 1615.5 MHz and 1246 - 1256.5MHz, with frequency interval of 0.5625MHz and 0.4375MHz. Each GLONASS satellite shares its frequency with the one on the other side of the earth. The GLONASS satellites are identified by their frequency instead of a PRN code [2].

1.2.3 RTK

Real Time Kinematics (RTK) is a differential GNSS technique that is based on measuring the phase difference in the carrier waves between two GNSS receivers. The two receivers share the main sources of error in standalone positioning, satellite clock error, atmospheric conditions etc. but by using the carrier phase difference the error from these sources cancel out [3, 4].

The difference in length between the two receivers to the satellite can be calculated if one knows the phase difference and the difference in number of full wavelengths for the two receivers [3]. This is sometimes referred to as the carrier-phase ambiguity. Different techniques are available to calculate the phase ambiguities [4].

For a given satellite, a simplified expression for the carrier phase observation is [3]:

$$\phi = \rho - I + Tr + c(b_{Rx} - b_{Sat}) + N\lambda + \epsilon_\phi \tag{1.1}$$

Where:
c is the speed of light.

I is the signal path delay due to ionosphere.
T_r is the signal path delay due to the troposphere.
b_{Rx} is the receiver clock offset form the reference time.
b_{Sat} is the satellite clock offset form the reference time.
λ is the wavelength of the carrier frequency.
N is the ambiguity of the carrier-phase (integer number).
ϵ_ϕ is measurement noise.
ρ is the geometrical distance between the receiver and satellite.

The expression for the double difference observable for two receivers a and b and two satellites 1 and 2 is [3]:

$$\phi_a^{12} - \phi_b^{12} = \rho_a^{12} - \rho_b^{12} - I_a^{12} + I_b^{12} + Tr_a^{12} - Tr_b^{12} + \lambda(N_a^{12} - N_b^{12}) + \epsilon_a^{12} - \epsilon_b^{12} \quad (1.2)$$

As $I_a^{12} \approx I_b^{12}$ and $Tr_a^{12} \approx Tr_b^{12}$ due to the two receivers being relatively close to each other, these terms cancel out. A even more simplified expression is then:

$$\phi_a^{12} - \phi_b^{12} = \rho_a^{12} - \rho_b^{12} + \lambda(N_a^{12} - N_b^{12}) + \epsilon_a^{12} - \epsilon_b^{12} \quad (1.3)$$

As ϕ_a^{12}, ϕ_b^{12} can be measured at the receivers, one will get the difference in $\rho_a^{12} - \rho_b^{12}$ by solving the carrier-phase ambiguity $\lambda(N_a^{12} - N_b^{12})$, assuming that the errors $\epsilon_a^{12} - \epsilon_b^{12}$ are small. By doing this for multiple pairs of satellites one will get the difference in position in multiple directions.

1.2.4 RTKLIB

RTKLIB is an open-source software that is developed mainly by Tomoji Takasu [5]. It is currently being distributed under the BSD 2-clause licence with additional clauses [6].

It supports different positioning algorithms and GNSS. It is capable of RTK position for using GPS and GLONASS. It is able to do this in real-time or in post-processing. RTKLIB supports external communication using serial, TCP/IP and other protocols [5].

RTKLIB makes use of an extended Kalman filter to estimate the next state in terms of position and other variables. When using post-processing this filter can be used going through states both forward and backward in time. This improves the solution and the setting is refereed to as combined [6]. In real-time applications the Kalman filter can only estimate values based on previous solutions. In its calculations it uses float values for the carrier-phase ambiguity. It tries to fit these as integer values as it increases accuracy and convergence time [4]. It validates the integer values using a ratio-test, if it succeeds the position estimation is reported as fixed, otherwise as float [4].

For more details on how RTKLIB works, see "Development of the low-cost RTK-GPS receiver with an open source program package RTKLIB" [4] or appendix E in the RTKLIB manual [6].

1.2.5 Emlid Reach

Emlid Reach is small module from which the manufacturer claims that it is claimed able to provide position estimations with centimeter precision using RTK [7]. This requires the use of two units where both can be Emlid Reach units but this isn't necessary. Emlid Reach was released to the market in the fall of 2015. Its hardware part consist of an Intel Edison computer and a GNSS receiver from u-blox (model NEO-M8T) as well as the board on which these components and connectors are placed. Emlid Reach is delivered with an external Tallysman antenna [1].

The software running on the Intel Edison computer is mainly RTKLIB and ReachView. ReachView is Emlids web interface that gives the user web access to RTKLIB and its settings [1].

We choose to test Emlid Reach as it was at the time the project started one of the few available RTK capable modules in its price range. At the time of writing this report a pair of two reach units costs $570.

Emlid Reach uses only one of the carrier frequencies (L1) for GPS for RTK [1].

A picture of Emlid Reach unit can be seen in figure 1.1. Emlid Reach is 45.5mm x 27mm x 9.2mm large and weighs 14g. Emlid Reach is powered with 5V and it draws at most 500mA [1].

1.2.5.1 Alternatives to Emlid Reach

When the Reach units were ordered the only competitor we found was Piksi. It is slightly older (available April 2014) and costs twice as much as the Emlid Reach [8].

But during the spring and summer of 2016 several competitors have either started selling their products or released information about future RTK capable units for a similar cost.

A French company called drotek announced their product SMARTNAV L1 RTK GNSS in February. It is very similar to Emlid Reach in that it also runs RTKLIB and a custom web interface on a Intel Edison computer connected to a u-blox NEO-M8T GNSS receiver. The difference is that drotek has attached the receiver and the Intel Edison directly to the antennas ground plane. It does also feature some additional connectors such as an Ethernet port. Cost is 423€ per unit [9].

PRECIS-BX305 is a unit from Tersus-GNSS that is capable of using the L2 frequency as well as carrier frequencies that Reach can handle. It is configured using custom,

Figure 1.1: A picture of Emlid Reach (taken form the Emlid Reach documentation with permission. [1]).

proprietary software. Cost for a pair was at the time of writing $1,999 [10].

U-blox the manufacturer of GNSS receivers that are used in some of the other products that have been mentioned have released evaluation kits and engineering samples of their new NEO-M8P receiver chip. This receiver is used in conjunction with software from U-blox and is claimed to be able to provide cm-accuracy using RTK [11]

1.3 Purpose

We will also test the precision, accuracy, and availability of the Emlid Reach units. And to utilize the Emlid Reach units RTK capabilities in conjunction with a Pixhawk autopilot.

Precision:
Defines, how close measured quantity values are in repeated measurements with the same conditions [12]. Standard deviation will be used to represent a numerical value of precision.

Accuracy
Defines, how close a measured quantity values are to the true value [12]. A numerical value will be produced by comparing the average value to a reference value measured with professional equipment with a known accuracy.

Availability
Defines the ratio of fixed to float solutions and how often Emlid Reach and RTKLIB is able to achieve fixed and float solutions.

1.4 Limitation

This project involves several different components in terms of both software and hardware. The success depend on that they will work together. However we are bounded by two constraints, i.e.:

- Emlid Reach will be used.
- Pixhawk will be used.

This means that we are also limited to RTKLIB and the software that Emlid has developed and their models that are used to estimate positions. As stated earlier there are a few different methods of solving the integer ambiguity but we will now be limited to use those methods that RTKLIB uses. Since a Pixhawk will be used limits us to the interfaces and software available for that particular autopilot.

The project was also done with the limitation that neither RTKLIB nor Ardupilot, the software for Pixhawk, was to be modified. All of these choices were done in order to reduce the scope of the thesis to design and test one system for RTK positioning.

1.5 Research question

- Test the precision of Emlid Reach (and RTKLIB) modules in RTK mode.
- Test the accuracy of Emlid Reach (and RTKLIB) modules in RTK mode.
- Set up a network / data link between the two Emlid reach modules that complies with the regulations in Sweden and rules at Onsala. A longer network for air to ground communication.
- Use the Emlid reach modules in RTK mode as a "GPS" for a pixhawk autopilot. (Work in progress by Emlid themself.)

2

Implementation and Methods

2.1 Real-time vs Post-processing

A good position estimation using RTK can be achieved in real-time as well as in post-processing. In order to use the RTK technique, data from two units are needed. One often refers to the unit with a known and static location as the base and the moving unit with an unknown location as rover.

In post-processing one starts by downloading raw data logs from both of the two reach units. But in real-time application this data needs to be transferred from the reach unit acting as base to the rover unit. The data from the base unit is then referred to as correction data.

RTKLIB and ReachView have the functionality to let data be transferred by using a TCP server/client relation. The base unit does then act as the server and the rover as the client. The data from the base unit is transferred and RTKLIB's calculation are done in real-time in the rover unit [1]. It is also possible to save the raw data from the rover and base on the rover when operating in this mode. This raw data can later be downloaded from the rover for post-processing.

2.2 Network

Even though the Emlid Reach unit contains a WiFi antenna it was deemed to be too weak and thus only useful over very short range. Emlid Reach does also support ethernet-over-usb [1].

In order to overcome some drawbacks, each Reach unit was connected through ethernet-over-usb to a separate Raspberry Pi 2. The Raspberry Pis were then connected to the same network using ethernet cables or WiFi adapters. By using a software called iptables on the Raspberry Pis it was possible to reroute the TCP calls from the rover to the base through the Raspberry Pis and their shared network. This setup removes any previous range limitation as Raspberry Pi is able to use different networks, including 4G and several wireless data links with directed

antennas.

During testing we used either ethernet cables or WiFi with a third Raspberry Pi acting as a router.

2.2.1 Emlid Reach and Pixhawk

If the improved position estimations are to be used in real-time, the Emlid Reach units need to transfer the position estimations to the Pixhawk autopilot as well as establish a data link between the rover and base unit. Emlid solved this problem by implementing what they call ERB (Emlid Reach Binary protocol). It has been implemented on the Reach units in one of the software updates and it has also been included in the beta version of the next release of the ArduCopter software [1].

The autopilot does often have a data-link to a ground station, usually a computer. This datalink does normally consist of a pair of radio modems transmitting telemetry and instructions over 433MHz (in Europe). Emlid has developed a protocol so that the correction data from the Reach unit acting as base can be included with this telemetry data and then provided to the Reach acting as rover using a serial port on the Pixhawk. The rover does also return its position estimation over that same serial port.

Almost all GNSS receivers that are used with Pixhawk are using a serial port. The rover unit will also have to use that port to provide the Pixhawk the position estimations. But instead of sending the correction data over the telemetry data link it is possible to transmit the telemetry data in parallel to the correction data using TCP over a different network. This setup gives the user access to the rovers ReachView application while Emlids solution does not. It does also allows for potentially longer ranges.

2.3 Method

Professional RTK equipment was used to measure reference points. The receiver was a Leica GRX1200+ with a Leica AR10 antenna. This was connected to network RTK service provided by Lantmäteriet. This should give an accuracy that is off by at most 5cm in any direction [13, 14]. As no moving reference was available with enough accuracy most tests were done while both the base and rover where static. The antennas were attached to a circular ground plane with a diameter of 10 cm using electrical tape. The antenna was then placed at one of reference points. As this was done by hand without actually seeing the mark underneath the ground plate this introduced a small error. In figure 2.1 a reference point that will later be referred to as C is shown.

Both the real-time position estimations and the raw data were then downloaded

Figure 2.1: A picture of the reference point C used in the static measurements. Reference point C is one of the six points used in the static test. The antenna and its ground plate was placed on the pink area. As the ground plate blocked the view of the pink area it was sometimes difficult to place the ground plate directly over the center of the pink area.

from the Reach units to a PC and RTKLIB was run on the desktop in order to get position estimations from post-processing.

A python script was developed to calculated the average error and standard deviation in different directions for each test.

Once an estimation for the accuracy and precision for fixed and float values were achieved some kinematic tests were done. These where done mainly to see how the fraction for fixed vs float changed when the units were moving.

2.4 Coordinate system

This project utilizes mainly two coordinate systems. The earth centered earth fixed (ECEF) with its origin in the earth's center of mass. $\vec{z}$ is parallel to the mean rotational axis. $\vec{x}$ points towards the mean Greenwich meridian and $\vec{y}$ is orthogonal to the others in such a way so that we have a right hand coordinate system ($\vec{x}$, $\vec{y}$, $\vec{z}$) [2, p. 7-8]. This coordinate system makes it easy to calculate distances between points as any point is represented by a vector with 3 components. Each component expressed in meters.

However, RTK results are often displayed in the local East, North, up (ENU) coor-

dinate system as they have a distinct height component.

To convert between the two coordinate system one uses a rotational matrix [15]

$$\begin{pmatrix} E \\ N \\ U \end{pmatrix} = \begin{pmatrix} -\sin\lambda & \cos\lambda & 0 \\ -\cos\lambda\sin\phi & -\sin\lambda\sin\phi & \cos\phi \\ \cos\lambda\cos\phi & \sin\lambda\cos\phi & \sin\phi \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{2.1}$$

where λ is the latitude and ϕ is the longitude of the measured point.

3

Results

During the course of this thesis work several software updates for both Emlid Reach and RTKLIB were released. Without showing measurements done with previous versions of the software, we saw a significant improvement with certain updates. The tests presented here are done with Emlid Reach image 1.2 and Reachview v0.4.9. Processed with RTKLIB beta version 2.4.3 b25. The settings that were used for these softwares can be seen in appendix A.

3.1 Static test

A test of accuracy and precision for static measurements was done on 2016-09-21. The Reach acting as rover was moved between stationary points and staying at each point for at least 3 min. The first and last 10 seconds at each point was excluded when calculating the average difference to the reference value and standard deviation due to potential influence from movement. All of these position estimations are the result of post-processing of the raw data downloaded from each receiver.

The result for position estimates marked as fixed can be seen in table 3.1 and in table 3.2 for position estimates marked as float. The only reference point at which we have a large fraction of position estimations reported as float is "1 - 3". The position estimates reported as float at reference point "1-3" are also much worse than the others.

The position estimates that are reported as fixed have a low mean distance to their respective reference value with a deviation of most 2.5 cm in each direction.

The raw data that used for these calculations was split up into several files due to reboots caused by power issues with one of the batteries. One of these files relate to 1-3 and might be the cause to why it has a larger percentage of position estimations marked as float.

Table 3.1: Statistics done on the position estimations marked as fixed in the static test performed on 2016-09-21. Raw data from the GNSS receivers was recorded on each Emlid Reach for at least 180 seconds at each location and later post-processed using RTKLIB. Mean distance to reference value and standard deviation is measured in centimetres and calculated in the ENU coordinate system.

Ref	#samples	mean_{East}	mean_{North}	mean_{Up}	std_{East}	std_{North}	std_{Up}	%Fixed
1 - 1	4486	0.08	0.64	0.99	0.13	0.12	0.28	99.8
7 - 1	886	-0.45	-1.74	1.06	0.06	0.09	0.23	98.4
1 - 2	378	-1.47	0.90	0.77	0.06	0.07	0.16	100
1 - 3	219	-0.73	1.03	-0.34	0.11	0.15	0.38	33.5
7 - 2	839	0.43	-0.77	2.15	0.42	0.38	1.18	94.4
B - 1	815	-0.82	0.24	2.29	0.07	0.07	0.15	100
A - 1	403	0.44	-1.28	1.88	0.11	0.14	0.27	81.2
C - 2	803	-0.69	-0.09	2.43	0.09	0.13	0.33	90.5
A - 2	819	-0.07	-1.53	1.39	0.11	0.13	0.30	100
B - 2	901	-1.49	-0.79	1.72	0.12	0.14	0.50	100

Table 3.2: Statistics done on the position estimations marked as float in the static test performed on 2016-09-21. Raw data from the GNSS receivers was recorded on each Emlid Reach for at least 180 seconds at each location and later post-processed using RTKLIB. Mean distance to reference value and standard deviation is measured in centimetres and calculated in the ENU coordinate system.

Ref	#samples	mean_{East}	mean_{North}	mean_{Up}	std_{East}	std_{North}	std_{Up}
1 - 1	8	5.15	5.30	-3.73	0.09	0.08	0.35
7 - 1	14	-3.09	-4.98	0.76	0.21	0.45	0.35
1 - 3	435	-21.11	87.71	131.39	1.53	4.48	21.20
7 - 3	50	1.41	1.37	7.28	0.27	0.61	1.29
A - 1	93	-7.41	-6.17	-0.14	1.23	0.80	0.32
C - 2	84	-0.09	0.28	3.38	0.06	0.08	0.16

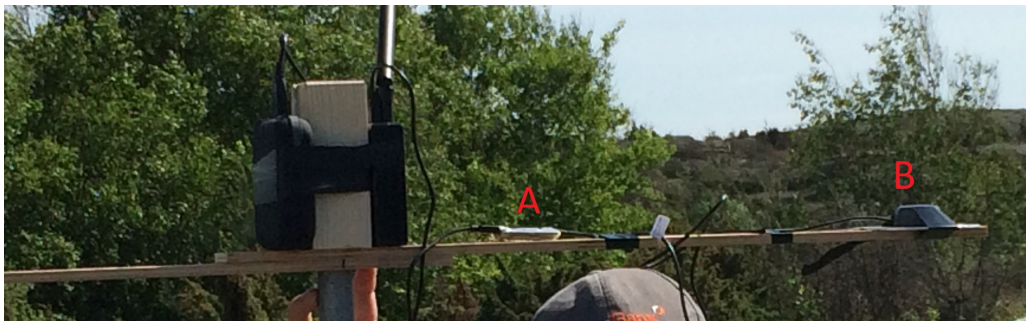

Figure 3.1: The setup for the kinematic test - circle. The Emlid Reach acting as rover is mounted at A and its antenna is mounted at B. The antenna is restriced to move in a circular pattern due to how the wooden beam is attached to the metal pole.

3.2 Kinematic test - circle

While the performance of Emlid Reach while stationary is interesting, the purpose of the Reach unit is to be used in conjunction with different kinds of moving systems.

One way to construct a reference for a moving test is to make the Reach acting as rover to move in a pattern that is known with good accuracy. As the is no moving reference it is impossible to know the position for a given time and data point. But it is still useful to compare the reported pattern to the known pattern.

Such a test was constructed by placing the rover on a beam that was then spun around a metal pole. The rover was restricted to move in a circle in the horizontal plane. The velocity was not controlled as it was spun by hand but estimations were done so that velocity comparable to e.g. drone flights. The setup of this test can be seen in figure 3.1.

In figure 3.2 one can see several different circles when the movement was restricted to a single circle. Fixed solution is achieved when both receivers record the raw data that is later post-processed. When both the raw data from the rover and the real-time correction data from the base is recorded on the rover we do only get float values in post-processing. For the fixed solution the difference between the circle is <3 cm. For the float solution the difference is <15 cm. For comparison the figure does also show the result when the raw data is post-processed as a single GNSS receiver. When no RTK or other differential GNSS methods are used there is no resemblance of a circle.

3.2.1 Statistics of the kinematic test

In order to calculate the standard deviation regarding how much a position estimation differs in length from the middle of the circle, we must first fit a circle to the position estimations.

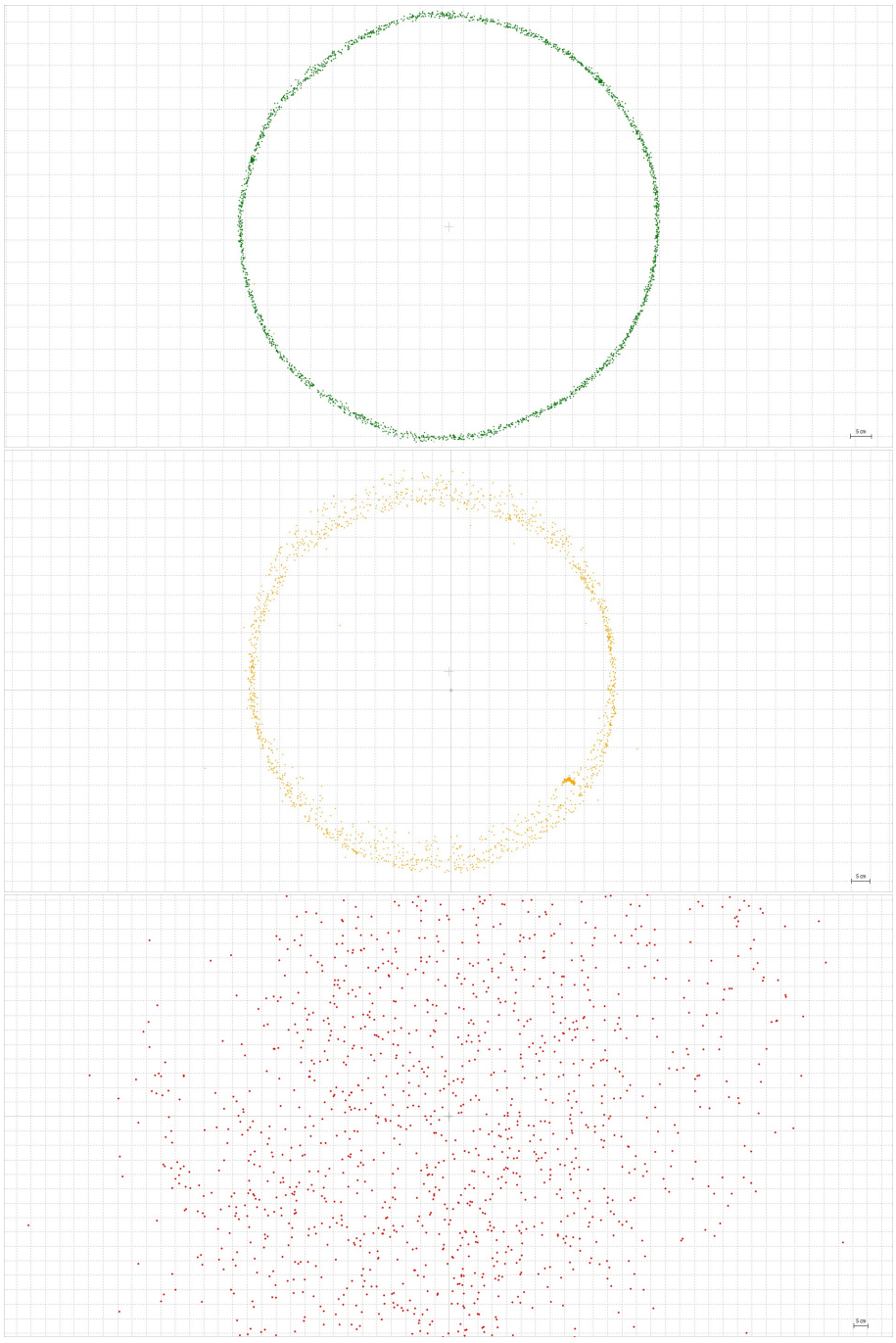

Figure 3.2: The rover rotating in a single circle around a metal pole. The upper figure is the result from post-processing using RTK based on raw data when both units were in single mode. The middle figure is the result of post-processing based on raw data when the units were in rover and base mode and transmitting corrections over wifi. The lower figure is based on the raw data from the rover and post-processed as a single GNSS receiver. All dotted squares have the side of 5 cm. Results are shown in the horizontal plane.

This was done by using numerical optimization methods in MATLAB to minimize the following with regards to $\vec{x_0}$ and R:

$$min_{\vec{x_0},R} \sum_i (|\vec{x_i} - \vec{x_0}| - R)^2$$

where:
R is the fitted radius of the circle
$\vec{x_0}$ is the fitted center of the circle
$\vec{x_i}$ is each individual position estimation

For each set of test data the standard deviation of the following is then calculated:

$$|\vec{x_i} - \vec{x_0}| - R$$

Using the position estimation previously shown in figure 3.2 we get the following values. Here the few data points that were not marked as fixed have been removed from the result with >99% fixed.

Type	STD radius	R
Fixed	0.48cm	48cm
Float	2.03cm	48cm

The set of position estimations marked as fixed refers to the case where the raw data downloaded from each receiver was post-processed. The set marked as float refers to the test where the correction data from the base was transferred in real-time to the rover and then stored there. It was then downloaded from the rover together with the raw data recorded on the rover for later post-processing.

3.3 Kinematic test - flight

As we have no reference data available for positions during a flight the purpose of this test is to determine if an RTK or at least post-processed kinematic solution can be achieved.

All of the flights were performed with an IRIS+ from 3D Robotics with the Reach unit and antenna placed above the rest of the electronics. They were all performed by letting the Pixhawk fly autonomously in a pattern similar to the one shown in figure 3.3. Different flights had different flight speeds, which varied between 2m/s to 4m/s.

Two flights were done on 2016-09-19: Post-processed kinematic solution of the raw data from the two reach units achieved 97.8% and 100% of the position estimations as fixed. These can be seen in figure 3.4 and 3.5.

Two flights were done on 2016-09-21: Post-processed kinematic solution of the raw

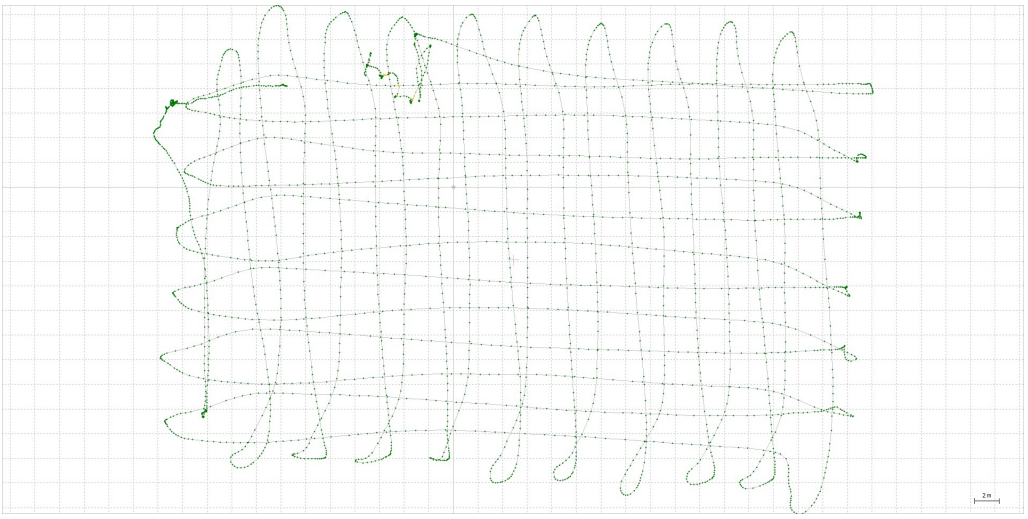

Figure 3.3: The second flight flown on 2016-09-21 shown in the horizontal plane. The speed was 4 m / s. Green indicates position estimations marked as fixed.

data from the two reach units achieved 100% and 98.7% of the position estimations as fixed. These can be seen in figure 3.6 and 3.7.

Neither of the figures show any discontinuities or jumps in the graphs. Such a jump would indicate that RTKLIB provided an incorrect solution for the integer ambiguity.

3.3.1 Height comparison to barometer

Even though the barometer of the autopilot is not the most reliable sensor it can be interesting to compare the heigh estimations from the barometer and the post-processed position estimations from Emlid Reach. The barometer is mounted on the autopilot and in the center of the IRIS+. As the antenna for the Emlid Reach is mounted on a pole the height difference between the two sensors will vary depending on the drones pitch and roll angles. The different height values can be seen in figure 3.8.

The first figure is troubling as there is a height difference of at least one meter between the two sensors. The second figure shows that the post-processed Emlid Reach data gives a smother result.

3.3.2 An application: Height mapping

By using the position estimations provided by Emlid Reach and RTKLIB we were able to make a rather simple height mapping of the area covered by the IRIS+ in one of the flights. This was done by connecting a laser range finder to the Pixhawk in the IRIS+ and combining the logs from the Pixhawk with the position estimations

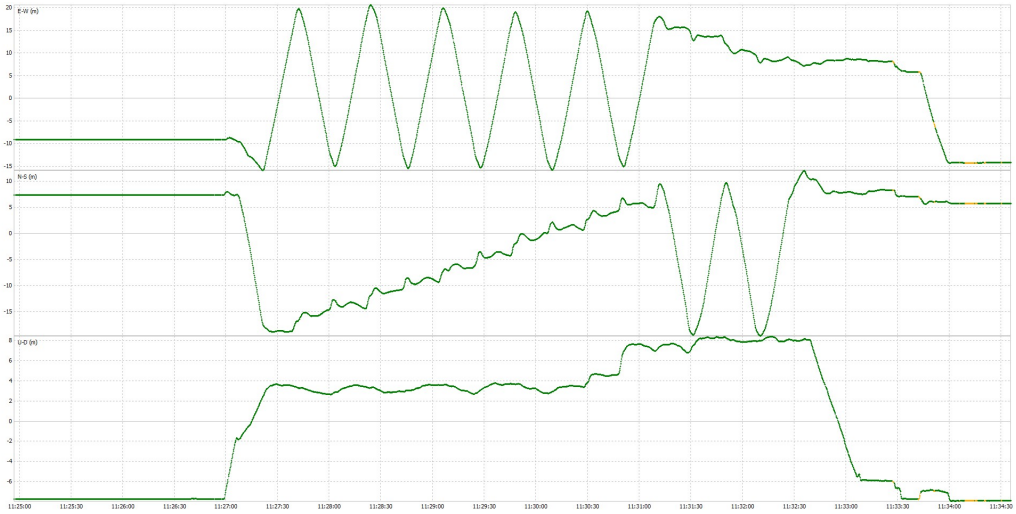

Figure 3.4: The first flight flown on 2016-09-19 shown in the ENU coordinate system. The speed was 2 m / s. Green indicates position estimations marked as fixed. The upper graph shows east-west, the middle north-south and the third one up-down. All in meters.

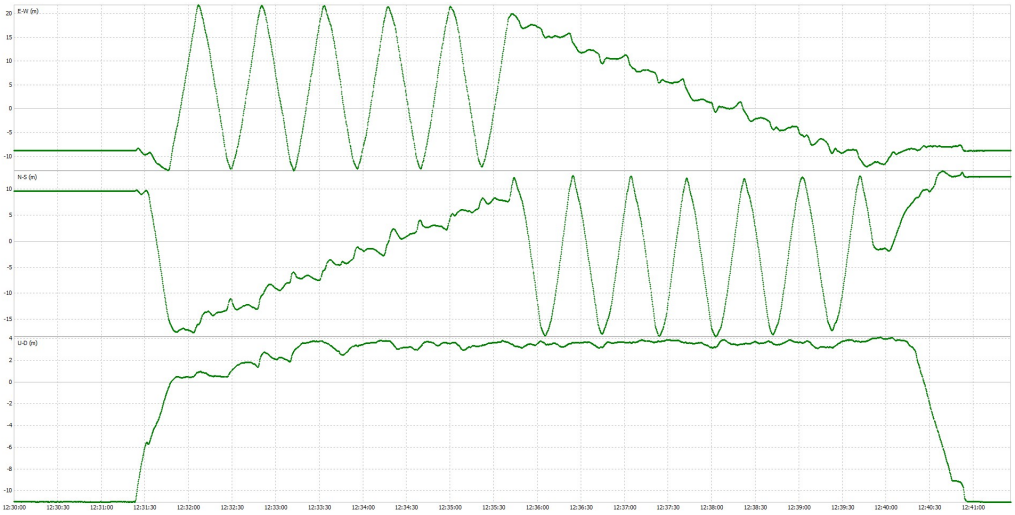

Figure 3.5: The second flight flown on 2016-09-19 shown in the ENU coordinate system. The speed was 2 m / s. Green indicates position estimations marked as fixed. The upper graph shows east-west, the middle north-south and the third one up-down. All in meters.

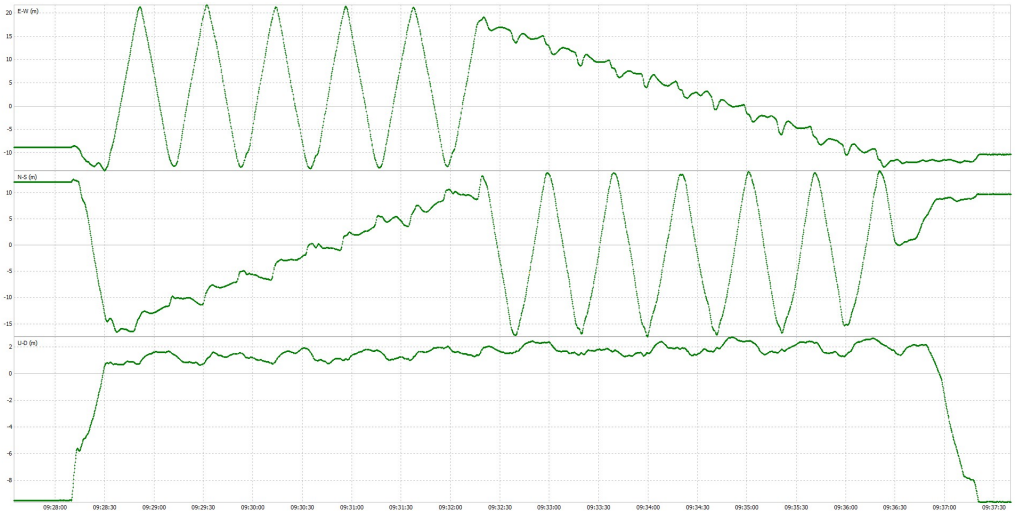

Figure 3.6: The first flight flown on 2016-09-21 shown in the ENU coordinate system. The speed was 2 m / s. Green indicates position estimations marked as fixed. The upper graph shows east-west, the middle north-south and the third one up-down. All in meters.

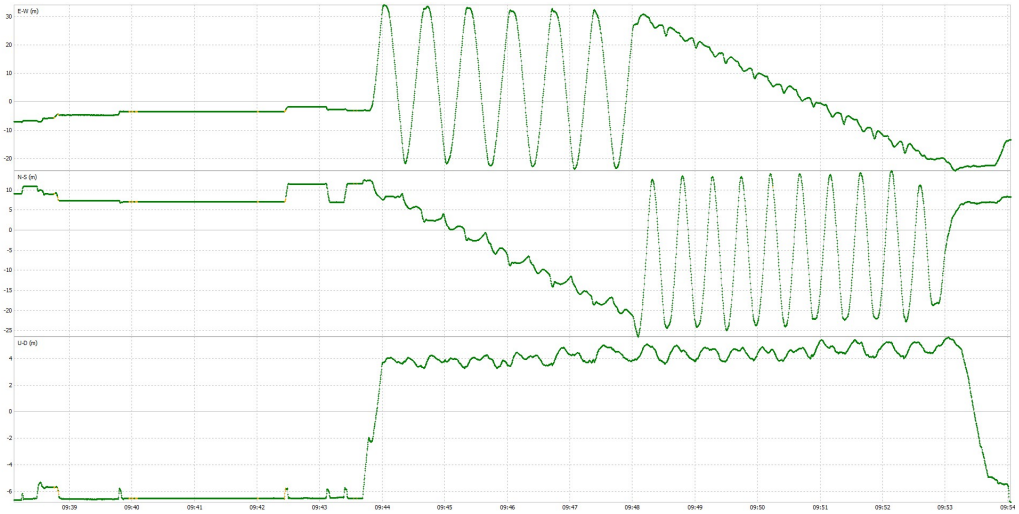

Figure 3.7: The second flight flown on 2016-09-21 shown in the ENU coordinate system. The speed was 4 m / s. Green indicates position estimations marked as fixed.

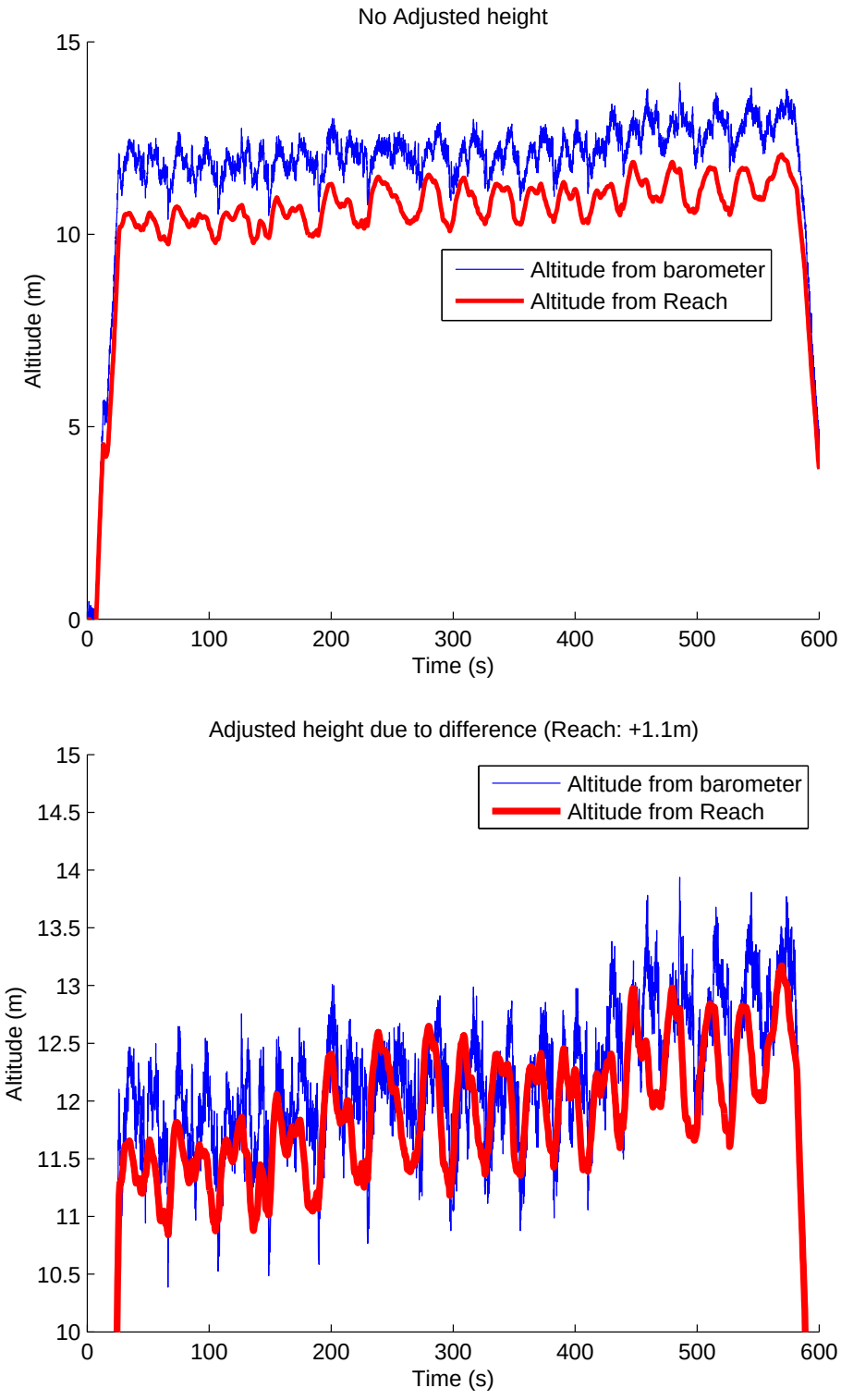

Figure 3.8: Both figures show height values from the barometer as well as the height value from post-processed Emlid Reach data. Both series have been adjusted so that they have the value 0 at the time of takeoff. In the lower figure the Emlid Reach data has been additionally adjusted by adding 1.1m. No correction has been done in order to account for varying physical heigh difference due to roll and pitch angles.

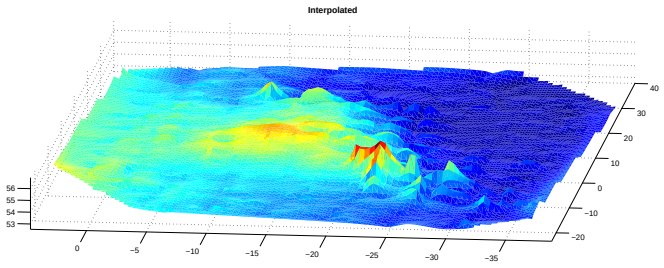

Figure 3.9: A height map from the north of the area covered in one of the flights. The IRIS+ uses a laser rangefinder and its autopilot to log a distance and the drones orientation. Emlid Reach is used in oder to determine the position of the drone. The picture was created by Joakim Strandberg from the Department of Earth and Space Sciences.

from RTKLIB.

The logs from the Pixhawk contain the ranges measured by the rangefinder as well as the orientation of the drone. By knowing the position and orientation as well as the range measurement it is possible to calculate the height of the point on the ground that was hit by the laser. A height map of the area can be seen in figure 3.9.

4

Conclusion and Outlook

4.1 Conclusion

Emlid Reach has provided cm accuracy during static tests and cm precision during static and kinematic tests for post-processed solutions. This is better than a single GNSS receiver.

Flight tests have shown a large percentage of positions estimation marked as fixed. If one deems RTKLIB, which has a larger user base than Emlid Reach, trustworthy one should probably also trust the position estimations reported as fixed in post-processing. The comparison between the barometer data and the data from Emlid Reach and RTKLIB is slightly contradictionary. Further testing should probably be done with an alternative reference to better determine the precision and accuracy during a flight.

The rather simple height mapping application demonstrates some of the potential UAS application have by better positioning provided via the Emlid Reach. This could be improved by using a rotating lidar sensor and / or an external IMU and rangefinder to provide ranges and orientation data at a higher frequency. Positions can rather easily be interpolated due to the fact that the vehicle is moving in a fairly simple pattern. However it is hard to interpolate orientation data as even small changes in this gives a large impact on the result.

Another potential application of cheap RTK positioning could be the easier creation of 3D maps as one will know where each picture was taken or other forms of mapping applications using different sensors. If several UASs are able to use Emlid Reach for real-time positioning it would allow for better relative positioning and formation flight. This could be important for some applications where one UAS would send out a signal and the other one would record the echo of that signal.

Emlid Reach is currently one of the cheaper options for RTK positioning. Many of its competitors are still in development or were recently been released so it is currently unclear if there is a large difference in performance between the options.

It should be noted that we have experienced some problems with the early versions of the software in Emlid Reach that gave incorrect position estimations. The tests

shown in this thesis were all done using the same version of the software, i.e. a version were the previous problem had disappeared. Future software updates might improve the performance as well. Which needs to be considered since for example at the time of writing this a new software version has been released.

4.2 Outlook

While this thesis indicates that Emlid Reach and RTKLIB performs well in post-processing, additional tests are required to show the accuracy and precision of position estimations for flying applications.

This would require access to professional GNSS systems with a good accuracy and precision as well as a UAS to carry it and the Emlid Reach unit with antennas. Another option would be to use a more powerful UAS to carry equipment so that is could be tracked from the ground using cameras or lasers. Unfortunately we did not have access to any of these two options.

Similar testing should probably be done for the competitors of Emlid Reach in order to determine if one is superior to the rest.

If further testing confirms what this thesis indicates, Emlid Reach and RTK can be used in many UAS applications by many users due to its comparably low cost to previous professional equipment.

Bibliography

[1] Emlid, "Reach RTK docs." https://docs.emlid.com/reach/, 2016. Accessed: 2016-09-11.

[2] G. Xu, *GPS, Theory, Algorithms and Applications*. Springer, 2nd ed., 2007.

[3] "Navipedia: Rtk fundamentals." http://www.navipedia.net/index.php/RTK_Fundamentals. Accessed: 2016-09-11.

[4] T.Takasu and A.Yasuda, "Development of the low-cost rtk-gps receiver with an open source program package rtklib," in *International Symposium on GPS/GNSS, International Convention Center Jeju, Korea*, 2009.

[5] T.Takasu, "RTKLIB: an open source program package for gnss positioning." http://www.rtklib.com/. Accessed: 2016-09-11.

[6] T.Takasu, "Rtklib ver. 2.4.2 manual." http://www.rtklib.com/prog/manual_2.4.2.pdf. Accessed: 2016-09-11.

[7] Emlid, "Reach." https://emlid.com/reach/. Accessed: 2016-09-11.

[8] S. Navigation, "Introducing piksi." http://swiftnav.com/piksi.html. Accessed: 2016-09-11.

[9] Drotek, "Smartnav l1 rtk gnss." https://drotek.com/shop/en/home/762-l1-rtk-gnss.html. Accessed: 2016-09-11.

[10] Tersus-GNSS, "Precis-bx305." http://www.tersus-gnss.com/products/precis-b01. Accessed: 2016-09-11.

[11] U-blox, "Neo-m8p." https://www.u-blox.com/en/product/neo-m8p. Accessed: 2016-09-11.

[12] JCGM/WG 2, "International vocabulary of metrology — basic and general concepts and associated terms (vim)." http://www.bipm.org/utils/common/documents/jcgm/JCGM_200_2008.pdf. Accessed: 2016-09-26.

[13] Lantmäteriet, "Swepos - förväntad mätosäkerhet." https://swepos.lantmateriet.se/tjanster/realtid/natverksrtk/matosakerhet.aspx. Accessed: 2016-11-30.

[14] Leica Geosystems, "Leica spider hardware, technical data." http://hds.leica-geosystems.com/downloads123/zz/general/general/brochures-datasheet/Spider_Hardware_technical_data_en.pdf. Accessed: 2016-11-30.

[15] "Navipedia: Transformations between ecef and enu coordinates." http://www.navipedia.net/index.php/Transformations_between_ECEF_and_ENU_coordinates. Accessed: 2016-09-11.

A

Configuration files

A.1 Configuration file for Emlid Reach

Setting file for RTK mode:

```
# rtkrcv options for rtk (v.2.4.2)

console-passwd      =admin
console-timetype    =gpst # (0:gpst,1:utc,2:jst,3:tow)
console-soltype     =dms # (0:dms,1:deg,2:xyz,3:enu,4:pyl)
console-solflag     =off # (0:off,1:std+2:age/ratio/ns)
inpstr1-type        =serial # (0:off,1:serial,2:file,3:tcpsvr,4:tcpcli
    ,7:ntripcli,8:ftp,9:http) ## Input source for onboard receiver
inpstr2-type        =tcpcli # (0:off,1:serial,2:file,3:tcpsvr,4:tcpcli
    ,7:ntripcli,8:ftp,9:http) ## Input source for base corrections
inpstr3-type        =off # (0:off,1:serial,2:file,3:tcpsvr,4:tcpcli,7:
    ntripcli,8:ftp,9:http)
inpstr1-path        =ttyMFD1:230400:8:n:1:off
inpstr2-path        =192.168.1.254:9000
inpstr3-path        =
inpstr1-format      =ubx # (0:rtcm2,1:rtcm3,2:oem4,3:oem3,4:ubx,5:ss2,6:
    hemis,7:skytraq,8:sp3) ## Input format for onboard receiver
inpstr2-format      =rtcm3 # (0:rtcm2,1:rtcm3,2:oem4,3:oem3,4:ubx,5:ss2
    ,6:hemis,7:skytraq,8:sp3) ## Input format for base corrections
inpstr3-format      =rtcm3 # (0:rtcm2,1:rtcm3,2:oem4,3:oem3,4:ubx,5:ss2
    ,6:hemis,7:skytraq,8:sp3)
inpstr2-nmeareq     =off # (0:off,1:latlon,2:single) ## Transmit NMEA
   GPGGA messages to Base station
inpstr2-nmealat     =0 # (deg) ## Latitude to send
inpstr2-nmealon     =0 # (deg) ## Longitude to send
outstr1-type        =file # (0:off,1:serial,2:file,3:tcpsvr,4:tcpcli,10:
    bluetooth) ## Solution 1 output path
outstr2-type        =serial # (0:off,1:serial,2:file,3:tcpsvr,4:tcpcli
    ,10:bluetooth) ## Solution 2 output path
outstr1-path        =/home/reach/logs/sol_%Y%m%d%h%M.pos
outstr2-path        =ttyMFD2:38400:8:n:1:off
outstr1-format      =llh # (0:llh,1:xyz,2:enu,3:nmea,5:erb) ## Solution
    1 output format
outstr2-format      =erb # (0:llh,1:xyz,2:enu,3:nmea,5:erb) ## Solution
    2 output format
logstr1-type        =file # (0:off,1:serial,2:file,3:tcpsvr,4:tcpcli,10:
```

```
    bluetooth) ## Raw data log for onboard receiver
logstr2-type       =file # (0:off,1:serial,2:file,3:tcpsvr,4:tcpcli,10:
    bluetooth) ## Raw data log for base corrections
logstr3-type       =off # (0:off,1:serial,2:file,3:tcpsvr,4:tcpcli)
logstr1-path       =/home/reach/logs/rov_%Y%m%d%h%M.ubx
logstr2-path       =/home/reach/logs/ref_%Y%m%d%h%M.rtcm3
logstr3-path       =cor_%Y%m%d%h%M.log
misc-svrcycle      =10 # (ms)
misc-timeout       =30000 # (ms)
misc-reconnect     =30000 # (ms)
misc-nmeacycle     =5000 # (ms)
misc-buffsize      =32768 # (bytes)
misc-navmsgsel     =rover # (0:all,1:rover,1:base,2:corr)
misc-startcmd      =
misc-stopcmd       =
file-cmdfile1      =../GPS_GLONASS_5Hz.cmd # (0:../GPS_1Hz.cmd,1:../
    GPS_5Hz.cmd,2:../GPS_10Hz.cmd,3:../GPS_14Hz.cmd,4:../
    GPS_GLONASS_1Hz.cmd,5:../GPS_GLONASS_5Hz.cmd,6:../GPS_BEIDOU_1Hz.
    cmd,7:../GPS_BEIDOU_5Hz.cmd) ## u-blox configuration file
file-cmdfile2      =../GPS_GLONASS_5Hz.cmd # (0:../GPS_1Hz.cmd,1:../
    GPS_5Hz.cmd,2:../GPS_10Hz.cmd,3:../GPS_14Hz.cmd,4:../
    GPS_GLONASS_1Hz.cmd,5:../GPS_GLONASS_5Hz.cmd,6:../GPS_BEIDOU_1Hz.
    cmd,7:../GPS_BEIDOU_5Hz.cmd) ## Base u-blox configuration file
file-cmdfile3      =
pos1-posmode       =kinematic # (0:single,1:dgps,2:kinematic,3:static
    ,4:movingbase,5:fixed,6:ppp-kine,7:ppp-static) ## Positioning mode
pos1-frequency     =l1 # (1:l1,2:l1+l2,3:l1+l2+l5) ## Carrier
    frequencies
pos1-soltype       =forward # (0:forward,1:backward,2:combined) ##
    Filter type
pos1-elmask        =15 # (deg) ## Elevation mask angle in deg
pos1-snrmask       =35 # (dBHz) ## SNR mask(minimum satellite level)
pos1-dynamics      =off # (0:off,1:on) ## Dynamics model of the rover(
    Kinematic and dgps only)
pos1-tidecorr      =off # (0:off,1:on) ## Apply earth tides corrections
pos1-ionoopt       =off # (0:off,1:brdc,2:sbas,3:dual-freq,4:est-stec)
    ## Ionosphere corrections
pos1-tropopt       =off # (0:off,1:saas,2:sbas,3:est-ztd,4:est-ztdgrad)
    ## Troposphere corrections
pos1-sateph        =brdc # (0:brdc,1:precise,2:brdc+sbas,3:brdc+ssrapc
    ,4:brdc+ssrcom) ## Type off satellite ephemeris
pos1-exclsats      = # (prn
pos1-navsys        =5 # (1:gps+2:sbas+4:glo+8:gal+16:qzs+32:comp) ##
    Used positioning systems
pos2-armode        =continuous # (0:off,1:continuous,2:instantaneous,3:
    fix-and-hold) ## GPS integer ambiguity resolution mode
pos2-gloarmode     =on # (0:off,1:on,2:autocal) ## GLONASS integer
    ambiguity resolution mode
pos2-arthres       =2 ## Integer ambiguity validation threshold
pos2-arlockcnt     =0 ## Minimum lock count to fix integer ambiguity
pos2-arelmask      =0 # (deg) ## Minimum elevation angle to fix integer
    ambiguity
pos2-aroutcnt      =5
pos2-arminfix      =10
pos2-slipthres     =0.05 # (m)
pos2-maxage        =30 # (s)
```

```
pos2-rejionno      =30 # (m)
pos2-niter         =1
pos2-baselen       =0 # (m)
pos2-basesig       =0 # (m)
out-solformat      =llh # (0:llh,1:xyz,2:enu,3:nmea)
out-outhead        =on # (0:off,1:on)
out-outopt         =off # (0:off,1:on)
out-timesys        =gpst # (0:gpst,1:utc,2:jst)
out-timeform       =tow # (0:tow,1:hms)
out-timendec       =3
out-degform        =deg # (0:deg,1:dms)
out-fieldsep       =
out-height         =ellipsoidal # (0:ellipsoidal,1:geodetic)
out-geoid          =internal # (0:internal,1:egm96,2:egm08_2.5,3:
    egm08_1,4:gsi2000)
out-solstatic      =all # (0:all,1:single)
out-nmeaintv1      =0 # (s)
out-nmeaintv2      =0 # (s)
out-outstat        =off # (0:off,1:state,2:residual)
stats-errratio     =100
stats-errphase     =0.003 # (m)
stats-errphaseel   =0.003 # (m)
stats-errphasebl   =0 # (m/10km)
stats-errdoppler   =1 # (Hz)
stats-stdbias      =30 # (m)
stats-stdiono      =0.03 # (m)
stats-stdtrop      =0.3 # (m)
stats-prnaccelh    =1 # (m/s^2)
stats-prnaccelv    =0.1 # (m/s^2)
stats-prnbias      =0.0001 # (m)
stats-prniono      =0.001 # (m)
stats-prntrop      =0.0001 # (m)
stats-clkstab      =5e-12 # (s/s)
ant1-postype       =single # (0:llh,1:xyz,2:single,3:posfile,4:
    rinexhead,5:rtcm)
ant1-pos1          =0 # (deg|m)
ant1-pos2          =0 # (deg|m)
ant1-pos3          =0 # (m|m)
ant1-anttype       =
ant1-antdele       =0 # (m)
ant1-antdeln       =0 # (m)
ant1-antdelu       =0 # (m)
ant2-postype       =rtcm # (0:llh,1:xyz,2:single,3:posfile,4:rinexhead
    ,5:rtcm) ## Base antenna coordinates
ant2-pos1          =57.43885 # (deg|m) ## Base antenna latitude
ant2-pos2          =12.02090 # (deg|m) ## Base antenna longitude
ant2-pos3          =79 # (m|m) ## Base antenna height
ant2-anttype       =
ant2-antdele       =0 # (m)
ant2-antdeln       =0 # (m)
ant2-antdelu       =0 # (m)
misc-timeinterp    =on # (0:off,1:on)
misc-sbasatsel     =0 # (0:all)
file-satantfile    =../../../data/igs05.atx
file-rcvantfile    =../../../data/igs05.atx
file-staposfile    =../../../data/station.pos
```

```
file-geoidfile      =
file-dcbfile        =../../../data/P1C1_ALL.DCB
file-tempdir        =../../../data/temp
file-geexefile      =
file-solstatfile    =
file-tracefile      =
```

A.2 Configuration file for RTKLIB

Setting file for the kinematic mode:

```
[set]
timestart=0
timeend=0
timey1=2000/01/01
timeh1=00:00:00
timey2=2000/01/01
timeh2=00:00:00
timeintf=0
timeint=0
timeunitf=0
timeunit=24
inputfile1=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160910\6\fromRaw
    \rover\rov_201609101728.obs
inputfile2=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160910\6\fromRaw
    \ref\rov_201609101728.obs
inputfile3=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160910\6\fromRaw
    \rover\rov_201609101728.nav
inputfile4=
inputfile5=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160910\6\fromRaw
    \rover\rov_201609101728.sbs
outputdirena=1
outputdir=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160910\6\solution
outputfile=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160910\6\
    solution\rov_201609101728.pos
[hist]
inputfile1_000=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160910\6\
    fromRaw\rover\rov_201609101728.obs
inputfile2_000=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160910\6\
    fromRaw\ref\rov_201609101728.obs
inputfile3_000=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160910\6\
    fromRaw\rover\rov_201609101728.nav
outputfile_000=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160910\6\
    solution\rov_201609101728.pos
inputfile1_001=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160910\6\
    rov_201609101728\RINEX\rov_201609101728.obs
inputfile2_001=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160910\6\
    ref_201609101728\RINEX\rov_201609101728.obs
inputfile3_001=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160910\6\
    rov_201609101728\RINEX\rov_201609101728.nav
inputfile5_000=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160910\6\
    fromRaw\rover\rov_201609101728.sbs
```

```
outputfile_001=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160910\1\
    solution\rov_201609101628.pos
inputfile1_002=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160910\1\
    rov_201609101628\RINEX\rov_201609101628.obs
inputfile1_003=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160910\1\
    fromRaw\rover\rov_201609101628.obs
inputfile1_004=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160910\3\
    rov_201609101652\RINEX\rov_201609101652.obs
inputfile2_002=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160910\1\
    ref_201609101628\RINEX\ref_201609101628.obs
inputfile2_003=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160910\1\
    fromRaw\ref\ref_201609101628.obs
inputfile2_004=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160910\3\
    ref_201609101652\RINEX\rov_201609101652.obs
inputfile3_002=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160910\1\
    rov_201609101628\RINEX\rov_201609101628.nav
inputfile3_003=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160910\1\
    fromRaw\rover\rov_201609101628.nav
inputfile3_004=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160910\3\
    rov_201609101652\RINEX\rov_201609101652.nav
inputfile5_001=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160910\6\
    rov_201609101728\RINEX\rov_201609101728.sbs
inputfile5_002=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160910\1\
    rov_201609101628\RINEX\rov_201609101628.sbs
outputfile_002=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160910\3\
    solution\rov_201609101652.pos
outputfile_003=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160526
    _spinning\solutionGLO\ref_201605261322.pos
inputfile1_005=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160526
    _spinning\fromRaw\ref\ref_201605261322.obs
inputfile1_006=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160526
    _spinning\fromRaw\rover\rov_201605261322.obs
inputfile2_005=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160526
    _spinning\fromRaw\ref\ref_201605261322.obs
inputfile2_006=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160526
    _spinning\fromRaw\ref\ref_201605261322.obs
inputfile3_005=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160526
    _spinning\fromRaw\ref\ref_201605261322.nav
inputfile3_006=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160526
    _spinning\fromRaw\rover\rov_201605261322.nav
inputfile5_003=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160910\1\
    fromRaw\rover\rov_201609101628.sbs
inputfile5_004=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160910\3\
    rov_201609101652\RINEX\rov_201609101652.sbs
outputfile_004=C:\Users\Sven Eriksson\Dropbox\EXJOBB\rtk\20160526
    _spinning\solutionGLO\rov_201605261322.pos
[opt]
posmode=2
freq=0
solution=2
elmask=15
snrmask_ena1=0
snrmask_ena2=0
snrmask_1_1=0
snrmask_1_2=0
snrmask_1_3=0
```

```
snrmask_1_4=0
snrmask_1_5=0
snrmask_1_6=0
snrmask_1_7=0
snrmask_1_8=0
snrmask_1_9=0
snrmask_2_1=0
snrmask_2_2=0
snrmask_2_3=0
snrmask_2_4=0
snrmask_2_5=0
snrmask_2_6=0
snrmask_2_7=0
snrmask_2_8=0
snrmask_2_9=0
snrmask_3_1=0
snrmask_3_2=0
snrmask_3_3=0
snrmask_3_4=0
snrmask_3_5=0
snrmask_3_6=0
snrmask_3_7=0
snrmask_3_8=0
snrmask_3_9=0
ionoopt=0
tropopt=0
rcvbiasest=0
dynamicmodel=1
tidecorr=0
satephem=0
exsats=
navsys=7
posopt1=0
posopt2=0
posopt3=0
posopt4=0
posopt5=0
mapfunc=0
ambres=3
gloambres=2
bdsambres=1
validthresar=3
thresar2=0,9999
thresar3=0,25
lockcntfixamb=0
fixcntholdamb=10
elmaskar=0
elmaskhold=0
outcntresetbias=5
slipthres=0,05
maxagediff=30
rejectgdop=30
rejectthres=30
numiter=1
codesmooth=0
baselinelen=0
```

```
baselinesig=0
baselineconst=0
solformat=0
timeformat=1
timedecimal=3
latlonformat=0
fieldsep=
outputhead=1
outputopt=1
outputdatum=0
outputheight=0
outputgeoid=0
solstatic=0
debugtrace=0
debugstatus=0
measeratio1=100
measeratio2=100
measerr2=0,003
measerr3=0,003
measerr4=0
measerr5=10
satclkstab=5E-12
prnoise1=0,0001
prnoise2=0,001
prnoise3=0,0001
prnoise4=10
prnoise5=10
rovpostype=0
refpostype=3
rovpos1=3,9325782161665E-12
rovpos2=0
rovpos3=21384,6857451801
refpos1=3,9325782161665E-12
refpos2=0
refpos3=21384,6857451801
rovantpcv=0
refantpcv=0
rovant=
refant=
rovante=0
rovantn=0
rovantu=0
refante=0
refantn=0
refantu=0
rnxopts1=
rnxopts2=
antpcvfile=
intprefobs=1
sbassat=0
netrscorr=0
satclkcorr=0
sbascorr=0
sbascorr1=0
sbascorr2=0
sbascorr3=0
```

```
sbascorr4=0
sbascorrfile=
precephfile=
satpcvfile=
staposfile=
geoiddatafile=
ionofile=
eopfile=
dcbfile=
blqfile=
googleearthfile=C:\Program Files\Google\Google Earth\googleearth.exe
rovlist1=
rovlist2=
rovlist3=
rovlist4=
rovlist5=
rovlist6=
rovlist7=
rovlist8=
rovlist9=
rovlist10=
baselist1=
baselist2=
baselist3=
baselist4=
baselist5=
baselist6=
baselist7=
baselist8=
baselist9=
baselist10=
exterr_ena0=0
exterr_ena1=0
exterr_ena2=0
exterr_ena3=0
exterr_cerr00=0,3
exterr_cerr01=0,3
exterr_cerr02=0,3
exterr_cerr03=0,3
exterr_cerr04=0,3
exterr_cerr05=0,3
exterr_cerr10=0,3
exterr_cerr11=0,3
exterr_cerr12=0,3
exterr_cerr13=0,3
exterr_cerr14=0,3
exterr_cerr15=0,3
exterr_cerr20=0,3
exterr_cerr21=0,3
exterr_cerr22=0,3
exterr_cerr23=0,3
exterr_cerr24=0,3
exterr_cerr25=0,3
exterr_perr00=0,003
exterr_perr01=0,003
exterr_perr02=0,003
```

```
exterr_perr03=0,003
exterr_perr04=0,003
exterr_perr05=0,003
exterr_perr10=0,003
exterr_perr11=0,003
exterr_perr12=0,003
exterr_perr13=0,003
exterr_perr14=0,003
exterr_perr15=0,003
exterr_perr20=0,003
exterr_perr21=0,003
exterr_perr22=0,003
exterr_perr23=0,003
exterr_perr24=0,003
exterr_perr25=0,003
exterr_gloicb0=0
exterr_gloicb1=0
exterr_gpsglob0=0
exterr_gpsglob1=0
[conv]
timespan=0
timey1=2000/01/01
timeh1=00:00:00
timey2=2000/01/01
timeh2=00:00:00
timeintf=0
timeint=0
trackcolor=5
pointcolor=5
outputalt=0
outputtime=0
addoffset=0
offset1=0
offset2=0
offset3=0
compress=0
[viewer]
color1=0
color2=16777215
fontname=Courier New
fontsize=9
```