TU Delft

Bachelor thesis

The accuracy of a GNSS-RTK smartphone system in the field

Lars van den Brand
Teeffelen
June 2021

Bachelor thesis

The accuracy of a GNSS-RTK smartphone system in the field

By

Lars van den Brand
Student number: 4966902

in partial fulfilment of the requirements for the degree of

Bachelor of Science
in Civil Engineering

at the Delft University of Technology

Supervisors: Ir. M. van den Berg TU Delft
Dr.ir. C.C.J.M. Tiberius TU Delft

TU Delft

Preface

In this report you will read about the accuracy of relatively cheap smartphone GNSS-RTK systems and the applicability of these systems in engineering practice. The project has been executed to complete the bachelor phase of the study Civil Engineering at the Delft University of Technology.

Engineers who are interested in the theory behind different GNSS positioning techniques, can find this information in chapter 2. Land surveyors can find the methodology of the field experiments in chapter 3. The results of the experiments can be found in chapter 4.

With the knowledge and help of my supervisors Ir. M. van den Berg and Dr.ir C.C.J.M. Tiberius I was able to complete this study. I would like to sincerely thank them for all their efforts.

Teeffelen, June 2021
Lars van den Brand

Summary

In engineering practice, the need exists for small-scale GNSS receivers that are able to take real-time measurements to deliver centimetre accuracy positioning. In hydraulic engineering for example, it is of great importance to know the speed and direction of the flow. When these flow characteristics need to be determined for a small waterway with a width of just a few metres, centimetre accuracy is required to obtain decent results. This study focuses on a low-cost, small-size, multi-frequency, multi-constellation GNSS receiver and investigates whether it can deliver this required centimetre accuracy in real-time. To this end, the static accuracy, the kinematic accuracy and the influence of the surroundings are determined.

First of all, two locations on the campus of the Delft University of Technology that lie approximately four meters apart, are chosen to act as ground truths in the field experiments. Benchmarks are put into the ground on the chosen locations. Subsequently, the coordinates of the benchmarks are accurately determined with high-end GNSS equipment.

To perform the static experiment, the GNSS antennas of two u-blox receivers are each placed on top of one of the benchmarks. The static accuracy is determined by comparing the position solutions from the u-blox receivers with the ground truth coordinates of the benchmarks. To determine the position solutions, the u-blox receivers use Real-Time Kinematic (RTK) positioning. RTK systems are composed of two receivers which both measure the phase of the carrier wave of satellites' radio signals. First, the carrier phase measurements of the same satellites are differenced between both receivers. Then, the obtained single differences are differenced between two satellites. This eliminates numerous error sources and allows the relative position of the rover receiver, with respect to the position of the reference receiver, to be determined with centimetre accuracy. By adding the obtained baseline vector to the accurate position of the reference receiver, the absolute position of the rover receiver can also be determined with centimetre accuracy. To enable RTK positioning, the reference receiver broadcasts correction messages, which consist of its raw GNSS measurements. These correction messages are received by the u-blox receivers, which can then perform the RTK positioning.

Three different applications are used to compare the position solutions from the u-blox receivers with the ground truth coordinates, namely: u-center, SW Maps and RTKPLOT. The function of u-center and SW Maps is simply to log the data. These applications both contain an NTRIP Client through which a connection can be made to an NTRIP Broadcaster. In this study, the correction messages from the DLF1 permanent receiver from the Delft University of Technology have been used, as the baseline between the u-blox receivers and this reference receiver is only a few kilometres. With these correction messages, the ZED-module of the u-blox receiver is able to fix the carrier phase ambiguity to an integer value. The resulting position solutions are logged and stored by u-center and SW Maps for further analysis. The remaining application, RTKPLOT, contains its own RTK engine and just uses the raw GNSS measurements from the u-blox receivers. RTKPLOT is used with two different settings. It is used with the DLF1 reference receiver and it is used with one of the two u-blox receivers as the reference receiver.

For the kinematic experiment a track has been built, which is centred above the benchmarks. This way, the ground truth for the kinematic experiment is the line connecting the ground truth coordinates of the benchmarks. The track consists of two plywood beams with an opening in between them. The GNSS antenna is mounted onto a plywood block, which can be pulled through the opening in the beams.

The influence of the surroundings is studied by testing how quickly the u-blox receiver is able to fix the carrier phase ambiguity again after the GNSS signals have been blocked for a while. To block the GNSS signals, the antenna is covered by human hands for a period of 15 seconds. After these 15 seconds, the antenna is left uncovered for again a period of 15 seconds. During this period, the u-blox receiver has time to restore the integer fix of the carrier phase ambiguity. This sequence is continuously repeated for a time span of 15 minutes.

The analyses of the static experiments show that RTKPLOT is not suitable for engineering applications, at least not in combination with the u-blox C099-F9P application board. With both analyses in RTKPLOT, the carrier phase ambiguities of at least 25% of the data are not fixed. This is acceptable for a static experiment, but not for a kinematic experiment in which there is just one measurement for each measured location. U-center and SW Maps on the other hand are very well suitable for engineering applications. With a fix ratio of 100% both applications are able to provide accurate position solutions throughout the entire measurement. Furthermore, both applications can be used to display the result in real-time. The required centimetre accuracy is also met with the RMSE in the North-direction being 0.90 cm, the RMSE in the East-direction being 0.56 cm and the RMSE in the Up-direction being 2.00 cm. These are the average values from the u-center analysis and the SW Maps analysis. The results of the kinematic experiment are comparable to the ones of the static experiment (even slightly better), meaning that the accuracy does not seem negatively influenced when going from static to kinematic measurements.

Symbols and Abbreviations

Symbols

dN	Northing
dE	Easting
dH	Differential Ellipsoidal height
ΔN	Differential value of the local North
ΔE	Differential value of the local East
ΔU	Differential value of the local Up

Abbreviations

CA	Coarse Acquisition
2D	Two-Dimensional
3D	Three-Dimensional
GNSS	Global Navigation Satellite Systems
GPS	Global Positioning System
HTTP	Hyper Text Transfer Protocol
LAMBDA	Least squares Ambiguity Decorrelation Adjustment
LOS	Line-Of-Sight
NLOS	Non-Line-Of-Sight
NRTK	Network Real-Time Kinematic
NTRIP	Networked Transport of RTCM via Internet Protocol
OTG	On-The-GO
PC	Personal Computer
PCO	Phase Centre Offset
PPK	Precise Point Kinematic
PPP	Precise Point Positioning
PRN	Pseudo Random Noise
RF	Radio Frequency
RFI	Radio Frequency Interference
RMSE	Root Mean Squared Error
RTCM	Radio Technical Commission for Maritime Services
RTK	Real-Time Kinematic
SPS	Standard Positioning Service
UHF	Ultra High Frequency
USB	Universal Serial Bus
VRS	Virtual Reference Station
ZTD	Zenith Tropospheric Delay

Table of contents

1 Introduction

1.1 Background

In our daily life, we frequently use the Global Navigation Satellite system (GNSS). For example, car drivers frequently use a navigation system to reach their destination. These daily life applications usually do not require very accurate position measurements. If the navigation system has an accuracy in the order of a few metres, it will still lead you to your destination. However, in engineering projects, positions often need to be determined with centimetre accuracy. On top of that, many engineering applications require real-time solutions. In hydraulic engineering for example, it is important to know the state of the flow. Ships want to pass a waterway when the speed and direction of the flow are favourable, as this saves time and fuel (Barr, 2019). GNSS drifters can be used to obtain information about these flow conditions (Sabet & Barani, 2011). When deployed at sea, receivers with a metre accuracy suffice due to the large scale of the measurements. However, in smaller waterways with a width of a couple of metres, centimetre accuracy is required to obtain a proper indication of the flow parameters. Furthermore, real-time solutions are preferred over solutions that need to be post-processed. The captain of a ship needs real-time data to make a decision on whether or not to pass a certain waterway.

Real-Time Kinematic (RTK) positioning systems can provide the required centimetre accuracy in real-time. These systems are composed of two receivers which both measure the phase of the carrier wave of satellites' radio signals. First, the carrier phase measurements of the same satellites are differenced between both receivers. Then, the obtained single differences are differenced between two satellites. This eliminates numerous error sources and allows the relative position of the rover receiver, with respect to the position of the reference receiver, to be determined with centimetre accuracy. By adding the obtained baseline vector to the accurate position of the reference receiver, the absolute position of the rover receiver is also determined with centimetre accuracy (Montenbruck & Teunissen, 2017).

1.2 Problem statement

A drawback of conventional RTK positioning systems is that the set-up, consisting of a rover receiver and a support structure with a height in the order of metres, is rather impractical due to its considerable size and weight. Among other application sectors, the large rover is rather inconvenient in hydraulic engineering, especially for the design of the aforementioned drifters. To support the weight and size of the rover receiver, the dimensions of the drifter should also be in the order of metres. This is possible for large scale applications at sea, however a smaller drifter is required for small scale applications in narrow waterways. Currently, a more practical GNSS-RTK system is available, allowing for a drifter with dimensions in the order of decimetres. The system consists of an antenna, a relatively cheap and small GNSS receiver (with a low power consumption), a reference receiver and an Android smartphone (Tiberius, 2020). Just like the conventional RTK system, it should be capable of performing measurements with centimetre accuracy. However, the system has not yet been tested in the field. Before it can be used in practise, the assumption that it has centimetre accuracy should be verified.

1.3 Objectives

The aim of this report is to answer the following research question: *How suitable is the GNSS-RTK smartphone system for practical applications that require centimetre accurate, real-time position measurements?* The answer to this research question will be obtained by answering the following three sub-questions:

1. How accurate is the GNSS-RTK smartphone system when performing static measurements?
2. How is the accuracy of the GNSS-RTK smartphone system affected when switching from static to kinematic measurements?
3. How is the accuracy of the GNSS-RTK smartphone system affected by the surroundings?

The answer to these sub-questions will be obtained through field experiments. These experiments will focus on the static and kinematic accuracy of the smartphone system and on the influence of the surroundings on this accuracy.

1.4 Structure overview

First, the theoretical framework is presented in chapter 2. At the beginning of the chapter, the composition of GNSS signals is explained. The remainder of this chapter deals with different GNSS positioning techniques. Next, the methodology is presented in chapter 3. In the first subchapter, the used GNSS receiver is described. The remaining subchapters each describe the set-up of one of the field experiments. Then, the results of the fieldwork are presented in chapter 4. This chapter consists of three subchapters. Each subchapter contains the results of one of the field experiments. Finally, conclusions are drawn in chapter 5. The conclusions are based on the results and contain the answers to the sub-questions. These answers lead to the answer to the research question, which is the final conclusion.

2 Theoretical framework

This chapter starts with the composition of GNSS signals. Then, four different GNSS positioning techniques are described. Next, the RTK positioning smartphone system is discussed in more detail. Finally, the influence of the surroundings on GNSS measurements is described.

2.1 GNSS signals

GNSS signals are electromagnetic waves with frequencies between approximately 1.2 and 1.6 GHz. These frequencies belong to the L-band of the radio spectrum. The GNSS signals consist of a carrier wave with two modulations. One of the modulations adds the Pseudo Random Noise (PRN) spreading code to the carrier wave, which allows receivers to distinguish between different satellites using the same frequency and to measure the moment of transmission. The PRN spreading code consists of chips with a value of either '0' or '1' and has a frequency of about 1 – 10 Mchips/s. The other modulation adds the navigation data message to the carrier wave. The navigation data message consists of bits and has a frequency of about 50 bits/s. It contains information about the satellite clock offset and the satellite's orbit (Montenbruck & Teunissen, 2017). An example of a GNSS signal is given in figure 2.1. The PRN spreading code and the navigation data message have been represented by the values '+1' and '-1'.

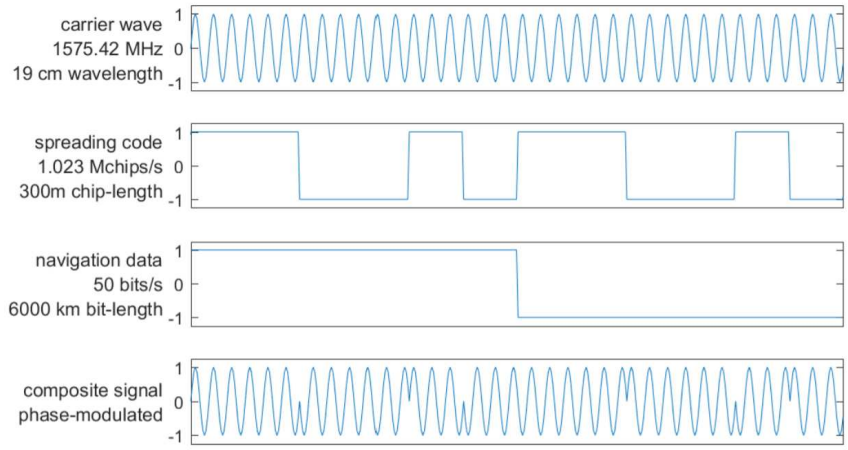

Figure 2.1: The composition of a GNSS signal (Tiberius, 2020)

2.2 Positioning modes

This subsection will deal with different GNSS positioning techniques. First, two point positioning techniques are described: the Standard Positioning Service (SPS) and Precise Point Positioning (PPP). Then, the following two relative positioning techniques are described: Real-Time Kinematic (RTK) positioning and Post-Processed Kinematic (PPK) positioning.

2.2.1 Standard Positioning Service (SPS)

All GNSS positioning techniques require observations of the distance between a receiver and the satellites. In SPS, also known as standalone positioning, the distance to the satellites is determined with pseudorange measurements. A receiver compares the incoming PRN spreading code with a local replica. The codes are aligned by shifting the replica in time and frequency (to compensate for the Doppler effect). The signal travel time follows from the applied time shift. The pseudorange can now be calculated by multiplying the travel time with the speed of light.

To obtain the position of the receiver, the three unknown position coordinates and the unknown receiver clock offset need to be solved simultaneously. As there are four unknowns, the pseudorange measurements of at least four satellites are required. The process of determining the position coordinates and the receiver clock offset is depicted in figure 2.2 for a 2D situation. In this case, just three satellites are needed as there are only two unknown position coordinates. The green circles represent the measured pseudoranges. The other circles are obtained by assuming different values for the receiver clock offset. The blue circles are the only ones that all intersect at one point. Therefore, they represent the real distances and the point of intersection represents the position of the receiver. Under favourable conditions, the accuracy of SPS is about 5 - 15 metres (Tiberius, 2020).

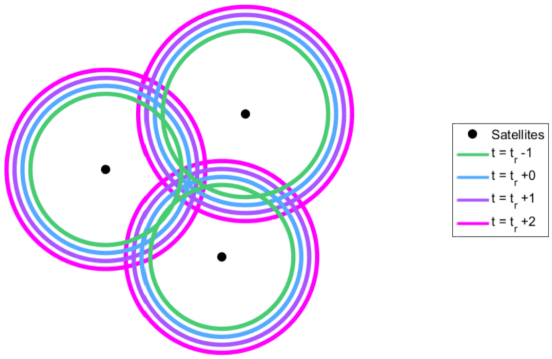

Figure 2.2: Determination of the receiver's position coordinates and clock offset (Tiberius, 2020)

2.2.2 Precise Point Positioning (PPP)

PPP relies on a network of reference receivers spread around the globe. The ultimate goal of PPP, and of all other positioning techniques, is to determine the position coordinates of a local receiver. PPP does this quite well, as it is capable of providing centimetre accurate position solutions. To reach this accuracy, several error sources need to be dealt with. Error sources influence GNSS signals and can cause the position solution to be off by a couple of meters (Tiberius, 2020). This is where the network of reference receivers comes into play. The data from the reference receivers is used to compute the clock offsets and orbit errors of the satellites. This information is stored in correction messages, which can be downloaded by local receivers, see figure 2.3. Next to these satellite related errors, the other major error sources are the receiver clock error and the delay along the travel path. This delay consists of two components: the delay in the ionosphere and the delay in the troposphere. By using dual frequency receivers, the ionospheric delay can be eliminated. This way, only the tropospheric delay remains. The tropospheric delay and the receiver clock error cannot be eliminated. However, they are modelled to reduce their influence.

The correction for the error sources alone is not enough to reach centimetre accuracy. Instead of just collecting pseudorange data, the local receiver also performs carrier phase measurements. The carrier phase consists of the number of periods of a carrier wave that have been observed by a receiver since the start of tracking. To be able to determine the distance to the satellites, the carrier phase ambiguity, which is the initial number of periods of the carrier wave, is also required. The carrier phase ambiguity cannot be fixed to an integer value, but it can be estimated. To do this, PPP exploits the fact that the carrier phase ambiguity is constant as long as the tracked signal is not interrupted.

Finally, the correction data from the network of reference receivers, the models for the remaining error sources and the pseudorange and carrier phase measurements from the local receiver itself are all combined to determine the position solution. Dual frequency PPP can reach centimetre accuracy after a convergence period of 20 - 30 minutes. The cheaper single frequency PPP can only reach an accuracy of the decimetre level, as it is not able to eliminate the ionospheric delay. Instead, it models this delay (Montenbruck & Teunissen, 2017).

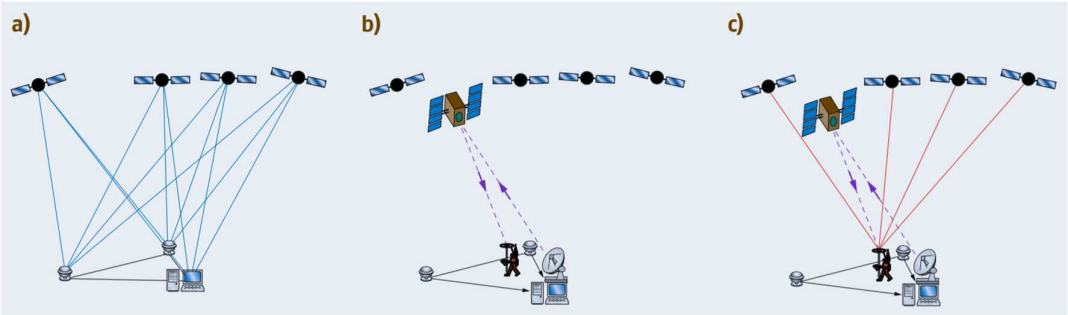

Figure 2.3a-c: The basic principles of PPP: a) GNSS correction parameters are determined based on the data from the global network of reference receivers; b) The parameters are uploaded by the network and downloaded by a local receiver; c) The local receiver computes its own position coordinates based on the correction parameters and its own GNSS measurements (Montenbruck & Teunissen, 2017)

2.2.3 Real-Time Kinematic (RTK) and Post-Processed Kinematic (PPK) positioning

RTK and PPK rely on the same basic principles. The difference is that RTK provides the position solution right on the spot and that PPK requires post-processing of the data to obtain the position solution. RTK and PPK systems are composed of two receivers: a reference receiver and a rover receiver. The position of the reference receiver has already been determined with an accuracy of the centimetre level or better. The position of the rover receiver has yet to be determined. Both receivers measure the carrier phase of satellites' radio signals. Double differencing is applied to the carrier phase measurements to eliminate the clock offsets and phase biases of both the receiver and the satellite and to reduce orbit errors and atmospheric delay errors. Double differencing means that the carrier phase measurements of the same satellite made by two distinct receivers are first differenced between both receivers. Then, this single difference is also determined for another satellite. Ultimately, the single differences are differenced between both satellites to obtain the double difference.

When the distance between the reference and rover receiver, the baseline, is 10 kilometres or less, the double differenced orbit errors and atmospheric delay errors are so small that they can be neglected. Therefore, the remaining unknown parameters in the carrier phase observation equations are the carrier phase ambiguity and the double differenced geometric range between receiver and satellite. In the next step, the unknown parameters are modelled. The obtained non-linear model is linearised to allow for least squares estimation. Finally, the resulting baseline vector is combined with the position coordinates of the reference receiver to obtain the position coordinates of the rover receiver. This technique is called single base, short-baseline RTK. The basic principles of RTK positioning are depicted in figure 2.4.

The carrier phase ambiguity is first estimated as a float. A technique called LAMBDA (Least squares AMBiguity Decorrelation Adjustment) is applied to fix the ambiguity to the right integer value. The fixed solution is more accurate than the float solution. Nowadays, ambiguity resolution takes about tens of seconds. The obtained accuracy of the baseline vector is of the centimetre level. As the position of the reference receiver has been determined with centimetre accuracy or better, the position of the rover receiver also has an accuracy of the centimetre level (Montenbruck & Teunissen, 2017).

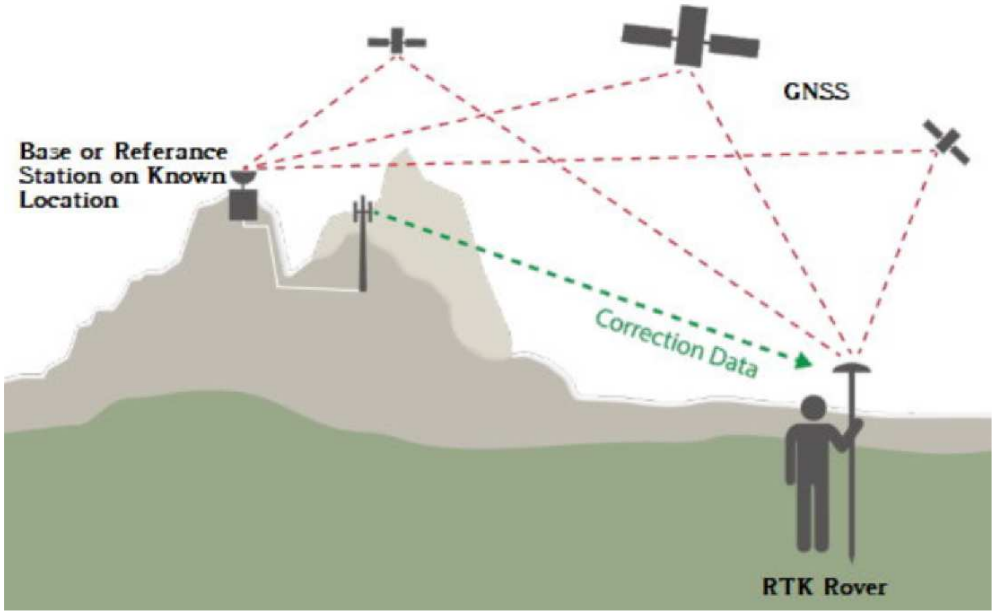

Figure 2.4: The visualisation of RTK positioning (DATAGNSS, 2021)

The maximum baseline length of 10 kilometres is a limiting factor. This length is limited due to the effect of the distance-dependent errors: the orbit error, the tropospheric delay and the ionospheric delay. However, by using a network of reference receivers, these errors can be modelled properly to create real-time, distance-dependent correction messages. This is exploited in a technique called Network RTK (NRTK). The reference receivers can be 40 – 50 kilometres apart.

The reference stations all send their raw GNSS measurement data to a central data processing centre. There, the carrier phase ambiguity is fixed by double differencing the measurements from the reference stations. Also, models are constructed to compensate for the distance-dependent error sources. When the rover receiver wants to obtain real-time correction messages, it sends its approximate location coordinates to the data processing centre. Then, by using these approximate coordinates and by applying the compensations based on the error models, the data processing centre creates a Virtual Reference Station (VRS) in the vicinity of the rover receiver, see figure 2.5. Now, the position solution of the rover receiver can be determined with the single base RTK technique. The raw GNSS data and the location of the VRS are sent to the rover receiver. The rover receiver combines this information with its own GNSS measurement data to compute its own location coordinates. The obtained accuracy is comparable to the accuracy of single base RTK, so NRTK also has an accuracy of the centimetre level (Montenbruck & Teunissen, 2017).

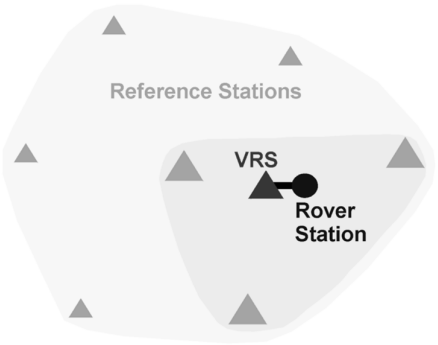

Figure 2.5: The location of the VRS (Wanninger, 2008)

2.3 The GNSS-RTK smartphone system

The GNSS-RTK smartphone system consists of three main components: an RTK rover receiver, a reference receiver and a smartphone. The communication between the different components is schematised in figure 2.6. The purpose of the system is to compute centimetre accurate real-time position solutions for the location of the antenna of the rover receiver. To do this, correction messages obtained from the reference receiver are sent to the rover receiver. These correction messages are communicated using NTRIP (Networked Transport of RTCM via Internet Protocol). NTRIP is an HTTP-based (Hyper Text Transfer Protocol) data streaming technique for GNSS data (Weber et al., 2005). RTCM stands for Radio Technical Commission for Maritime Services and is the standard streaming format for correction messages. In a default single base set-up, an NTRIP Broadcaster collects correction messages from the appropriate NTRIP Source (the reference receiver) and sends them to the NTRIP Client on the smartphone via HTTP streams. Subsequently, the NTRIP Client sends the correction messages to the rover receiver. The communication via HTTP streams is realised by using sockets. This implies that a port number is required to connect to the NTRIP Broadcaster (Oracle, n.d.).

Based on its own GNSS measurement data and on the obtained correction messages, the rover receiver computes the centimetre accurate position solution. This solution is then sent to the smartphone, which uses a certain application to display and save the results. The communication between the smartphone and the rover receiver can be organised in several ways. One option is to connect the devices with a USB OTG (On-The-Go) cable. The connection can also be wireless, for example via Bluetooth or Wi-Fi.

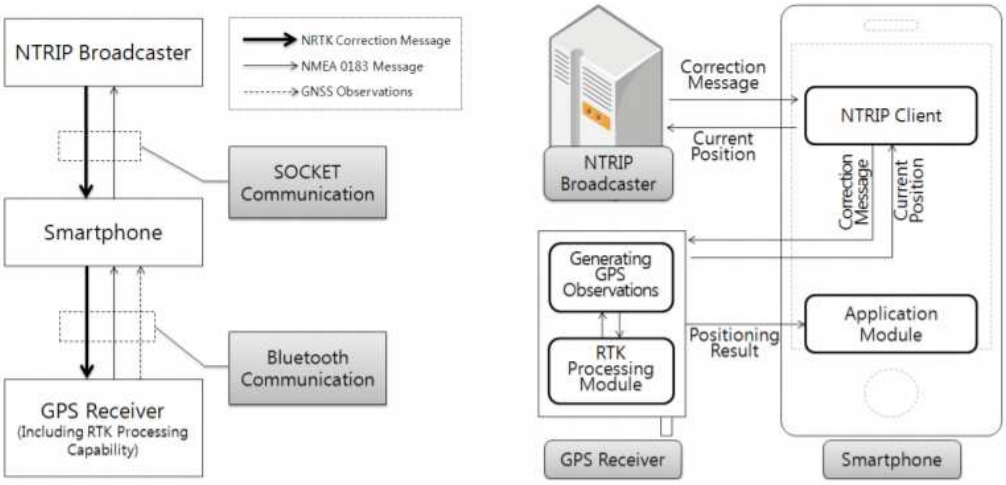

Figure 2.6: The communication within the GNSS-RTK smartphone system (Hwang et al., 2012)

2.4 Error sources

Numerous error sources affect GNSS signals. These error sources can be subdivided in three categories:

- Error sources due to the generation and broadcast of the GNSS signal at the satellite.
- Error sources that affect the GNSS signal during its propagation to earth.
- Error source in the vicinity of the receivers

The last category of error sources can be influenced by the way in which the measurements are done in the field. Therefore, this category will be looked at in more detail. The main error sources of this category are: receiver noise, multipath and blockage and Non-Line-Of-Sight (NLOS) reception. When working with relative positioning techniques, the Phase Centre Offset (PCO) of GNSS antennas can also introduce errors.

Receiver noise

Receiver noise consists of two components: the noise generated by the receiver and the noise from the surroundings. When a GNSS signal arrives at a receiver, it has to be processed. Many electrical component are involved in the processing of the signal, including cables, connectors and the antenna. Imperfections in these components lead to small random errors in the pseudorange and carrier phase measurements. These errors can also be caused by background noise from the surroundings. When this noise is in or near the frequency range of GNSS signals, it can cause Radio Frequency Interference (RFI) and disturb signal tracking (Montenbruck & Teunissen, 2017).

Multipath

GNSS signals do not just arrive at a GNSS receiver; they illuminate a large part of the Earth's surface, including the area in the vicinity of the receiver. From there, the signals can get reflected and still end up on the antenna of the receiver, see figure 2.7. When the receiver measures both the reflected signal and the direct line-of-sight (LOS) signal, multipath effects arise. The code and phase measurements of the LOS signal are affected by the superposition of the signals and the extended path of the reflected signal (Montenbruck & Teunissen, 2017).

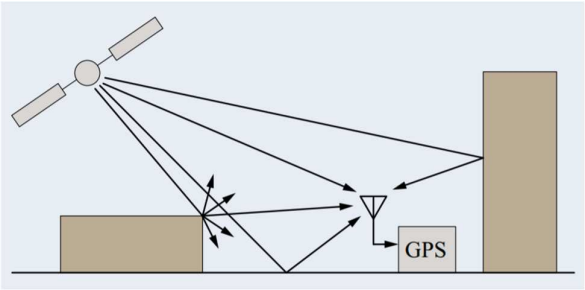

Figure 2.7: Multipath signals (Montenbruck & Teunissen, 2017)

Blockage and NLOS reception

As explained above, GNSS receivers in the vicinity of reflective surfaces will probably observe some reflected GNSS signals. However, when the direct LOS signal is blocked, the reflected signals are the only observations of the receiver. This phenomenon is called NLOS reception, see figure 2.8. As superposition does not take place, the pseudorange measurement error is just equal to the extra path length of the reflected signal (Petovello, 2013). Another possibility is that all GNSS signals, including the reflected ones, are blocked. This is called blockage. Due to a lack of data, the receivers are unable to calculate the position solution.

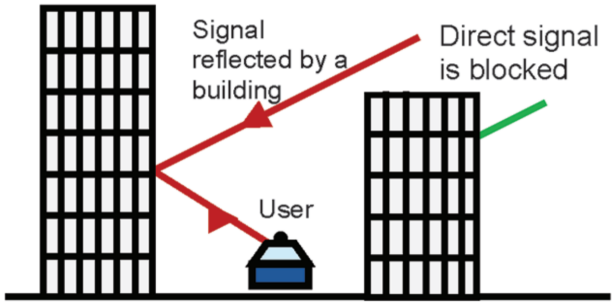

Figure 2.8: NLOS reception (Petovello, 2013)

Phase Centre Offset

The PCO of a GNSS antenna is the vector between the actual point of the antenna where satellites' GNSS electromagnetic signals are received, the phase centre, and the physical point of the antenna to which the coordinates are referenced and for which they are reported. Commonly, the reported coordinates belong to a point at the bottom of the antenna. In relative positioning techniques, the calculated relative baseline is the line connecting the PCO's of the antennas. This baseline should be added to the coordinates of the reference receiver to obtain the coordinates of the rover receiver. The known coordinates of the reference receiver represent the reference point (at the bottom of the antenna) and not the phase centre. Therefore, to avoid the introduction of an error, the difference in the PCO's of the reference and rover receiver should be accounted for in the calculation (Ordnance Survey, n.d.).

3 Methodology

In this chapter, the set-up and the procedure of the field experiments is explained. Three types of experiments have been carried out: static experiments, kinematic experiments and an experiment to measure the influence of the surroundings on the measurements. Before going into detail about the experiments, the used GNSS receiver will be introduced first.

3.1 GNSS receiver

The aim of this study is to test a practical, small-scale GNSS-RTK system that is compatible with a smartphone. As explained in chapter two, an RTK system consists of a rover receiver and a reference receiver. The rover receiver that is used in this experiment is the low-cost, low-power consumption, multi-frequency and multi-constellation C099-F9P application board from u-blox, see figure 3.1. The main component of the application board is the ZED-F9P high precision positioning module. This module combines the correction messages from a reference receiver with the measurements from the u-blox receiver itself and computes the position solution. In other words, the ZED-module is the component that contains the RTK technique. Next to the ZED-module, the board contains a micro-USB port to connect to external devices, such as a smartphone or a PC. The board also contains RF connections (Radio Frequency connections) to both the ZED-module and the ODIN-module. The RF connection to the ZED-module enables a GNSS antenna to be connected. The antenna that is used is the u-blox ANN-MB-00 multi-band GNSS antenna. The ODIN-module is used for wireless communication, so the RF connection to this module enables the connection of a Wi-Fi and Bluetooth antenna (Digi-Key, 2020).

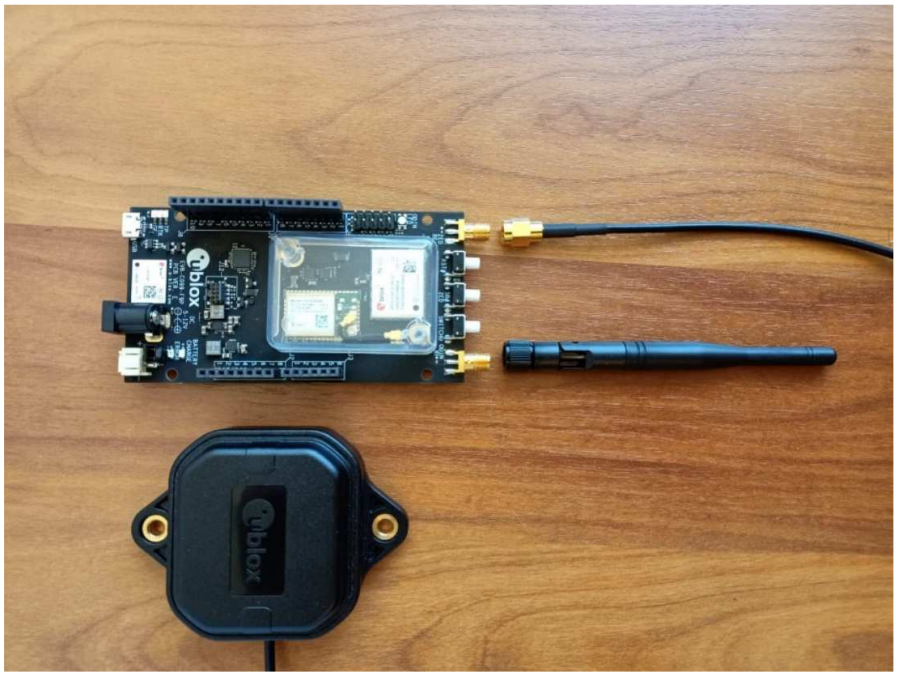

Figure 3.1: The u-blox C099-F9P application board and the antennas

3.2 Static accuracy

The static experiments are performed to deduce the accuracy of the u-blox GNSS receiver. A static experiment is chosen for this purpose, as this yields numerous measurements of the same location. By averaging these measurements, a proper statistical indication of the accuracy of the u-blox receiver can be obtained. Different processing applications are used to turn the measurements into position solutions. These are: u-center, SW Maps and RTKPOST.

3.2.1 Reference points

Two u-blox receivers are used for the static experiments. Both receivers measure the coordinates of a different static point in space, the reference points. To determine the accuracy of the u-blox receiver, the coordinates of the reference points are compared with the position solutions from the u-blox receivers. To act as a ground truth, the coordinates of the reference points must be known with a very high accuracy. The assumption is that the u-blox receivers are capable of performing measurements with centimetre accuracy. Therefore, the ground truth coordinates of the reference points must be millimetre accurate, ideally at least one order higher than the anticipated accuracy of the u-blox receivers. Two locations at the Delft University of Technology, that lie several metres apart, are chosen as the reference points. Benchmarks are created to mark these locations, see figure 3.2.

(a)

(b)


(c)

Figure 3.2a-c: a) The location of the reference points on the TU Delf campus, located to the south-east of the city centre, indicated by the red circle; b) The exact location; c) One of the benchmarks

The locations of the reference points are determined using high-end GNSS equipment, see figures 3.3 and 3.4. In figure 3.3 the antenna is depicted together with the tripod. The tripod ensures that the antenna is levelled and centred right above the reference point. The antenna in this figure is the Trimble Zephyr Geodetic (TRM41249.00). Figure 3.4 shows the Trimble R7 system. This is a high-end multi-band, multi-frequency GNSS receiver and contains an UHF (Ultra High Frequency) radio (Trimble, 2007). To obtain the required millimetre accuracy, the measurements from the Trimble systems are post-processed using the Netherlands Positioning Service (NETPOS). NETPOS consists of a network of reference stations spread around the Netherlands (NSGI, n.d.).

Figure 3.3: Trimble antenna and tripod

Figure 3.4: The Trimble R7 system

3.2.2 Analysis with u-center

U-center is used to analyse the measurements from the u-blox receivers. U-center is an application provided by u-blox that is designed for the analysis of GNSS data. Before going into further details about u-center, the set-up is explained. The set-up is depicted in figures 3.5 and 3.6. Figure 3.5 depicts one of the GNSS antennas positioned right on top of the first benchmark. The antennas are placed on top of a ground plane to block reflected GNSS signals. Because the benchmarks are made from metal and because the ground plane is magnetic, the antenna is kept in position quite well and the introduced error is only in the order of millimetres. The surface surrounding the benchmarks is not completely flat. Therefore, folded pieces of paper towel are used to level the antenna. Figure 3.6 depicts the connection between the PC running u-center and the u-blox receiver. This connection is established via a USB cable. As explained in section 3.1, the GNSS antenna is connected to the receiver via the ZED RF connection.

Figure 3.5: The GNSS antenna set-up

Figure 3.6: Connection to a PC

To obtain enough measurements, the antennas are left in the static position for at least one hour. As they collect GNSS data with a frequency of 1 Hz, they have collected approximately 3600 measurements at the end of the experiment, which is enough to draw grounded statistical conclusions. The data collection rate is one of the numerous settings that can be altered in u-center. Another setting that has to be set in u-center is the collection of RTCM correction messages from a reference receiver. This setting is crucial, as it enables the RTK positioning technique. To set up the connection with a reference receiver, u-center contains an NTRIP Client. This NTRIP Client can communicate with an NTRIP Broadcaster that sends out correction messages. In this experiment, the NTRIP Client is used to establish a connection with the TU-Delft permanent GNSS receiver 'DLF1'. This reference station is chosen as the baseline to the benchmarks at the campus is only a couple of kilometres.

The ZED-module of the u-blox receiver can now combine the correction messages with its own measurements to compute the position solutions. These solutions are displayed in u-center. Next to these solutions, u-center also reports the status of the solution. The status tells whether the carrier phase ambiguity can be fixed to an integer value or not. Furthermore, u-center reports the number of satellites that are used to compute the position solution. These functionalities allow to drop the measurements without a carrier phase ambiguity fix and with a low number of used satellites. The more specific u-center settings can be found in Appendix A. In this appendix, it can be seen that the raw GNSS measurement data is also stored by u-center. This data is used in subsection 3.2.4 for the analysis with RTKPOST.

3.2.3 Analysis with SW Maps

SW Maps is a smartphone application capable of collecting and presenting geographic data. The only difference between the set-up for the u-center analysis and the SW Maps analysis, is that the u-blox application board is now connected to a smartphone. The connection to the smartphone is depicted in figure 3.7. An extra cable is required with respect to the u-center set-up. This cable must be an USB OTG cable, with an USB-A connection on one side to connect to the USB cable coming from the u-blox receiver and either a micro-USB or an USB-C connection at the other side to connect to a smartphone.

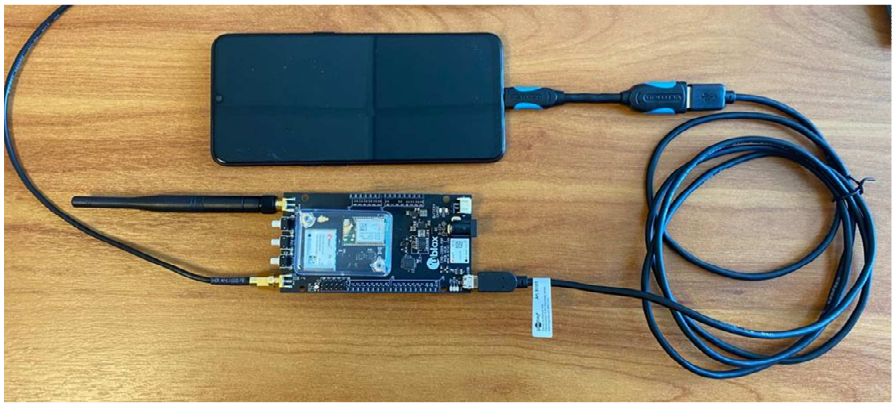

Figure 3.7: Connection between application board and smartphone

As the u-blox application board has just one USB connection, the data collection for SW Maps cannot take place at the same time as the data collection for the u-center analysis. Therefore, after logging data for one hour for the u-center analysis, the antenna needs to stay put on exactly the same location to log data for one hour for the SW Maps analysis. Just like u-center, SW Maps contains an NTRIP Client to establish a connection

with an NTRIP Broadcaster. Again, the correction messages from the DLF1 permanent GNSS receiver are used. Another similarity between SW Maps and u-center is that both applications are just used to display the position solutions calculated by the ZED-module of the u-blox receiver. To ensure high quality results, SW Maps contains the option to only log the results with a fixed carrier phase ambiguity. This option needs to be enabled. The remaining settings can be found in Appendix B. When the measurement is done, SW Maps is used to export the logged data to a KML file. KML is an abbreviation of Keyhole Markup Language and is just a file format to store geographic data.

3.2.4 Analysis with RTKPOST

RTKPOST is one of the applications of the program package RTKLIB. RTKPOST can compute position coordinates from raw GNSS measurements. In this study, it has been used to perform two different analyses. First of all, it has been used to process the measurements from the u-blox receivers in RTK positioning mode. As the raw measurements from the u-blox receivers are used, the RTK technique of RTKPOST is applied and not the RTK technique of the ZED-module of the receiver itself. Therefore, the results of this analysis can be used to check the performance of the ZED-module. To process data in RTK positioning mode in RTKPOST, the option ‘Kinematic’ must be selected, see figure C2 from Appendix C. Furthermore, three files containing GNSS data must be uploaded. At the first file entry, the observation file from the u-blox rover receiver at location 1 is uploaded. To obtain this observation file, CONVBIN from RTKLIB is used. Using this application, the u-blox log-file, obtained from the first hour of measurements, is converted into an observation file and a navigation file. Observation files contain information that is specific for the measurement location, e.g. the measured pseudoranges and carrier phases. Navigation files contain the navigation data message, which is specific for each satellite (Gurtner & Estey, 2007). On the web page ‘dgpa’, monitored by the Delft University of Technology, RINEX files can be downloaded which contain the raw GNSS measurements from numerous reference receivers spread throughout the Netherlands (van der Marel, 2021). These RINEX files can be converted into observation and navigation files. The observation file from the DLF1 reference receiver is uploaded into the second file entry. The navigation file of the DLF1 receiver is uploaded into the third entry. Another key setting is the filter type that is applied. By selecting ‘Forward’, the data is only processed in the forward direction, as if the measurements and the processing is taking place in real-time. The remainder of the applied settings can be found in Appendix C.

The second analysis with RTKPOST concerns the rover base configuration consisting of two u-blox receivers. With this set-up, the reference data from the DLF1 reference receiver is no longer necessary. The observation data from the first benchmark is still uploaded into the first file entry. However, the second and third file entry are now used to upload the observation and navigation file from the u-blox receiver at the second benchmark. In this analysis, the raw measurements of the two u-blox receivers are used. Another possibility would be to connect the u-blox receivers in real-time, enabling the real-time streaming of correction messages. However, this study does not focus on this possibility, as its practical application is limited. For example, a real-time connection is not possible when one of the u-blox receivers is covering multiple kilometres inside a drifter. Therefore, the filter type in RTKPOST is changed from ‘Forward’ into ‘Combined’, which means the GNSS data is processed in both the forward and the backward direction. This increases the chances of fixing the integer ambiguity, improving the quality of the position solutions. The remainder of the settings can be found in Appendix C.

3.3 Kinematic accuracy

The kinematic experiment is carried out to determine how the static accuracy of the u-blox receiver is influenced when the antenna is moving. As for the static experiments, a ground truth is required. Regarding the relatively short duration of this study, it is impossible to create ground truth speed measurements. However, by using a specially designed track, the ground truth path of the antenna can be determined, see figure 3.9. The track is built of plywood. It consist of a base with a width of approximately 20 centimetres. Plywood beams with a width of 5 centimetres are added on either side of the base to guide the antenna. The remaining space in between the beams has a width of 10 centimetres. A small plywood square with a base of 10 x 10 centimetres is used to pull the antenna through the track.

Figure 3.9: The track

As mentioned before, the idea behind the track is to have a ground truth path on which the measured positions should lie. To obtain this path, the track is positioned right above the benchmarks. First of all, the front of the track is centred right above the centre of the first benchmark, see figure 3.10(c). The track has a length of five metres, whereas the benchmarks lie approximately four metres apart. Therefore, the track completely covers the second benchmark. To position the centre of the track right above the centre of the second benchmark, the width of the track is measured. The ground to the left and right of the second benchmark is marked at a distance equal to half the width of the track, see figures 3.10(a-b). When putting the track in position, the sides of the track are put right on top of the marks on the ground, ensuring that the centre of the track also lies right above the centre of the second benchmark. Furthermore, the track prevents the sideways motion of the antenna. Therefore, the measured positions of the u-blox receiver have to lie on the line connecting the benchmarks. As the coordinates of the benchmarks have been determined with the high-end GNSS equipment, this line is accurate enough and suitable to act as a ground truth.

(a)

(b)

(c)

Figure 3.10a-c: a-b) Positioning of the track right above benchmark 2. c) Positioning of the start of the track right above benchmark 1.

U-center is used to display the measurements. Again, the NTRIP Client is used to receive the correction messages from the DLF1 permanent receiver and the RTK technique of the u-blox receivers ZED-module is used to compute the position solutions. Based on the resulting coordinates, the distance to the ground truth path can be calculated. The applied settings in u-center are the same as the ones used for the static experiment and can be found in Appendix A.

3.4 Influence of the surroundings

The influence of the surroundings is studied by testing how quickly the u-blox receiver is able to fix the carrier phase ambiguity after the GNSS signal has been blocked for a while. After performing a small experiment, it seemed that human hands are more effective at blocking the incoming GNSS signals than a large metal casing. To carry out the experiment, the most stable u-blox antenna is used. In this case that is the antenna on top of benchmark 2. To block the GNSS signals, the antenna is covered by human hands for a period of 15 seconds, see figure 3.11. After these 15 seconds, the hands are removed from the antenna and it is left uncovered for again a period of 15 seconds. During this period, the u-blox receiver has time to restore the RTK fix of the carrier phase ambiguity. This sequence is continuously repeated for a time span of 15 minutes. U-center is used to log and store the measurements.

Figure 3.11: The cover up period of the RTK fix experiment

4 Results

In this chapter, the results of the field experiments are presented. The position solutions of all experiments are presented in ellipsoidal coordinates. As it is more convenient to interpret the results in units of metres than in units of degrees, the latitude and longitude have been converted into Northing (dN) and Easting (dE) (van der Marel, 2020). The Python code for this conversion has been added to Appendix D. The Northing and Easting are displayed with respect to the ground truth coordinates of the measured benchmark. By also displaying the ellipsoidal height with respect to ground truth ellipsoidal height of the benchmark, in approximation a so-called local topocentric coordinate system is created.[1] This coordinate system is used to display the results.

The majority of the position solutions is based on the reference data from the DLF1 permanent receiver. The Phase Centre Offset in the Up-axis of its local topocentric coordinate system has a value of 15.8 cm (NGS, 2017). Also, the PCO in the Up-axis of the u-blox receiver is 0.8 cm (u-blox, personal communications, n.d.). A positive PCO of the reference receiver results in a too big height difference of the baseline and a positive PCO of the rover receiver results in a too small height difference. Therefore, a value of 15.8 – 0.8 = 15.0 cm should be added to the position solutions that are computed with the u-blox rover receiver and the DLF1 reference receiver.

4.1 Static accuracy

Just like in chapter 3, the results of the static accuracy experiment are subdivided into the three different analysis applications: u-center, SW Maps and RTKPOST.

4.1.1 Analysis with u-center

To extract the results from u-center, the measurements of both u-blox receivers are replayed and stored as a CSV file. To ensure that only the reliable and accurate position solutions are used in the processing, all solutions without a fixed carrier phase ambiguity resolution are dropped from the data. For both benchmarks the data only contains fixed solutions, so no measurements have been dropped. The minimum number of used satellites has been determined as well. For the measurements of the first benchmark, the minimum number of satellites is 23 and for the second benchmark this is 24. Moreover, the average number of used satellites is 26.22 for the first benchmark and 26.44 for the second one. The number of used satellites is sufficient, so again no data has been dropped.

Based on the differential Northing, Easting and height coordinates, several statistical values have been calculated. The calculations have been done in Python and the code as has been added to Appendix D. The results of these calculations are averaged for the two u-blox receivers and presented in table 4.1.

Table 4.1: The statistical quantities from the u-center analysis

	Bias (m)	Standard deviation (m)	Root Mean Squared Error (m)
Northing	-0.0021	0.0066	0.0069
Easting	-0.0007	0.0057	0.0059
Differential height	0.0182	0.0136	0.0229
2D-distance (dN,dE)	0.0080	0.0042	0.0091
3D-distance	0.0216	0.0118	0.0246

[1] A local topocentric coordinate system actually contains North (ΔN), East (ΔE) and Up (ΔU) values. However, as the distance between the measurements is small, the following applies: $\Delta N \approx dN$, $\Delta N \approx dN$ and $\Delta U \approx dH$ (dH is the differential ellipsoidal height) (van der Marel, 2020).

Next, the measured quantities are plotted against the epoch number. As the measurements are taken with a frequency of one Hertz, each epoch represents one second. These plots can be found in Appendix E. The measurements are also plot in 3D in the local topocentric coordinate system. Finally, a scatter plot has been made, in which the Easting of the measurements has been plotted against the Northing. These plots are also added to Appendix E.

4.1.2 Analysis with SW Maps

Just like u-center, SW Maps logs the position solutions that have been calculated by the ZED-module of the u-blox application board. These position solutions have been exported to a KML file. To be able to analyse the data, the KML file is converted into a TXT file. SW Maps has been used to log the measurements from the antenna on top of benchmark 1. To calculate the statistical quantities, the Python script for the static accuracy from Appendix D has been used.

SW Maps has been configured in such a way that it only saves the position solutions with an RTK fix of the carrier phase ambiguity. Measurements have been made for approximately one hour. The resulting data contains 3665 epochs, corresponding to 1.018 hours of measurements. This means that the u-blox receiver has been able to obtain an RTK fix during the entire experiment. The remaining results of the analysis are presented in table 4.2. The plots from the SW Maps analysis are displayed in Appendix F.

Table 4.2: The statistical quantities from the SW Maps analysis

	Bias (m)	Standard deviation (m)	Root Mean Squared Error (m)
Northing	-0.0099	0.0049	0.0111
Easting	-0.0044	0.0029	0.0052
Differential height	0.0037	0.0089	0.0097
2D-distance	0.0114	0.0045	0.0122
3D-distance	0.0143	0.0057	0.0154

4.1.3 Analysis with RTKPOST

To analyse the results from RTKPOST, the data has been converted to KML files. Just the measurements with an RTK fix have been included in this conversion. For the first analysis with RTKPOST, with the reference data from the DLF1 reference station, this means that only 47.3% of the position solutions remains, corresponding to 1823 epochs and 0.506 hours. For the second analysis, using the u-blox reference receiver, this means that still 73.4% of the measurements remains, corresponding to 2830 epochs and 0.786 hours. The statistical values of the analyses are presented in the tables 4.3 and 4.4. The plots are displayed in Appendix G.

Table 4.3: The statistical quantities from the RTKPOST analysis with the DLF1 reference receiver

	Bias (m)	Standard deviation (m)	Root Mean Squared Error (m)
Northing	-0.0022	0.0057	0.0061
Easting	-0.0027	0.0053	0.0059
Differential height	0.0080	0.0114	0.0140
2D-distance	0.0074	0.0041	0.0085
3D-distance	0.0140	0.0082	0.0163

Table 4.4: The statistical quantities from the RTKPOST analysis with the u-blox reference receiver

	Bias (m)	Standard deviation (m)	Root Mean Squared Error (m)
Northing	0.0002	0.0054	0.0054
Easting	-0.0025	0.0032	0.0040
Differential height	-0.0081	0.0102	0.0130
2D-distance	0.0058	0.0033	0.0067
3D-distance	0.0128	0.0070	0.0146

4.2 Kinematic accuracy

In the kinematic experiment, the track is used to obtain a ground truth. If the u-blox receivers were perfect, all measurements would have to lie on the ground truth line connecting the benchmarks. By calculating the distance of the different position solutions to this line and by comparing this distance with the results from the static accuracy tests, conclusions can be drawn on the influence of the kinematic aspect of the measurements. This is done in chapter 5. In this paragraph, the results of the distance calculation are presented. The distance calculation itself is done with Python. The used Python code is added to Appendix D. As can be seen in the code, a value of 0.036 m has been added to the height of the benchmarks. This is to compensate for the height of the track. The base of the track and the plywood square that is used to pull the antenna through the track both have a thickness of 0.018 m.

A scatter plot of the Easting and Northing of the measurements has been included in Appendix H. The ground truth line between the ground truth coordinates is plotted in the same graph. The time series of the 2D- and 3D-distance and the cross-track error are also displayed in Appendix H.

Table 4.5: The statistical quantities from the kinematic experiment

	Bias (m)	Standard Deviation (m)	Root Mean Squared Error (m)
2D-distance	0.0045	0.0034	0.0056
Cross-track error	0.0015	0.0054	0.0056
3D-distance	0.0152	0.0113	0.0189

4.3 Influence of the surroundings

To investigate the influence of the surroundings, the statistical quantities have been calculated, as in the other analyses. Furthermore, the time series of the solution status and of the number of used satellites in the position solution are plotted and displayed in Appendix I. The average number of used satellites is 24.44 and the minimum number is 19 used satellites. Moreover, the u-blox receiver has managed to compute an RTK fix solution for 65.35% of the measurements. The time series of the Northing, the Easting and the height containing just the measurements with an RTK fix and the time series containing all measurements are also displayed in Appendix I.

Table 4.6: The statistical quantities based on all measurements

	Bias (m)	Standard deviation (m)	Root Mean Squared Error (m)
Northing	-0.0075	0.3717	0.3716
Easting	-0.0252	0.2207	0.2220
Differential height	-0.0422	0.8511	0.8517
2D-distance	0.1683	0.3990	0.4329
3D-distance	0.3799	0.8771	0.9554

Table 4.7: The statistical quantities based on the measurements with an RTK fix

	Bias (m)	Standard deviation (m)	Root Mean Squared Error (m)
Northing	-0.0085	0.0286	0.0299
Easting	0.0010	0.0332	0.0332
Differential height	-0.0045	0.0789	0.0790
2D-distance	0.0273	0.0353	0.0447
3D-distance	0.0541	0.0729	0.0907

5 Conclusion

The goal of this study is to determine how suitable a small-scale GNSS receiver, in this case the u-blox C099-F9P application board, is for engineering applications that require centimetre accurate, real-time position solutions. To this end, the static accuracy and the kinematic accuracy have been calculated and an experiment has been performed to determine the influence of the surroundings. Furthermore, the static accuracy has been calculated by using different analysis methods to find out which method is the best.

The analysis of the static experiment with u-center has yielded very accurate position solutions. The Root Mean Squared Error (RMSE) of the distance to the ground truth in the local East-North plane is just 9.1 mm. For the analysis with SW Maps, this value is 12.2 mm. However, the RMSE of the 3D-distance to the ground truth is 9.2 mm lower for the SW Maps analysis. All in all, the results from both analyses are comparable. The data for the analysis of SW Maps has been gathered one hour after the data collection for the u-center analysis. Therefore, it is logical that the results are not exactly the same. The differences that arise are small and give no reasons to believe that one of the applications is better than the other. This makes sense as SW Maps and u-center are both just used to log the data; the RTK technique is applied by the ZED-module of the u-blox receiver. An advantage of SW Maps is that it can be run on a smartphone. A disadvantage is that it can only convert the data to a KML file, which just contains the position solutions. If the engineering application allows to use a PC, u-center is the better option as it offers more possibilities to analyse the data.

Contrary to the analysis with SW Maps, the analysis with RTKPOST is based on the same data as the u-center analysis. However, the RTK technique of RTKPOST experiences difficulties in fixing the carrier phase ambiguity. This has resulted in a drop of 52.7% of the data for the analysis with the DLF1 reference receiver and in a drop of 26.6% of the data for the analysis with the u-blox reference receiver, causing the datasets and thus also the expected results to differ from each other. The results from the RTKPLOT analysis are slightly more accurate than the u-center analysis. The RMSE of the 2D-distance is 0.6 mm smaller for the DLF1 reference receiver and 2.4 mm smaller for the u-blox reference receiver. The RMSE of the 3D-distances are 8.3 mm and 10.0 mm smaller, respectively. This can be explained by the smaller datasets. If 50% of the least accurate position solutions of the u-center analysis would be dropped, the results would be more comparable. Still, the differences between the RTKPLOT and the u-center analysis are small. Therefore, it is assumed that these differences are caused by the differences in the length of the data and that the RTK techniques of the ZED-module and RTKPOST have a comparable accuracy. Regarding the fact that the ZED-module has been able to obtain an RTK fix for all measurements, it can be concluded that the overall performance of the ZED-module is better. Therefore, it is advisable to use either u-center or SW Maps in engineering applications.

The kinematic experiment has been analysed with u-center, so the statistical quantities are compared with the results from the static u-center analysis. All quantities, the bias, the standard deviation and the root mean squared error, are lower for the kinematic experiment than for the static experiment. This means that the accuracy of the u-blox receiver does not drop when going from static to kinematic measurements.

However, the time series plots of the kinematic experiment are more spiky than the plots from the static u-center experiment, see Appendices H and E, respectively. A possible explanation for this spiky behaviour is the inconstant force that is exerted on the rope connected to the plywood block underneath the antenna. Every time the person pulling the antenna steps forward, a force is exerted on the on the small block. This

causes it to move sideways a little bit. The sideways motion is not completely restricted as the block must be able to slide through the track smoothly. The block also gets lifted up a little bit every time force is exerted.

During the RTK fix experiment, the u-blox antenna has been covered for half of the time. However, the fix ratio is 65.35%. This means that, during the time that the antenna was covered, sometimes the u-blox receiver was still able to compute an integer carrier phase ambiguity resolution. This also means that an RTK fix solution has been computed for the for the vast majority of the time that the antenna was not covered.

The minimum number of satellites that has been used for the position solutions is 19 and the average is 22.44. For the static u-center analysis of benchmark 2, the average number of used satellites is 26.44. Even though the antenna was covered or 50% of the time, it was still able to use a decent amount of satellites. On average, the difference with the static experiment is just 4 satellites. So, if the GNSS signals to the u-blox antenna are blocked for a while and if the integer fix of the carrier phase ambiguity gets lost, the u-blox receiver experiences no difficulties in fixing the carrier phase ambiguity again when the GNSS signals are restored.

In conclusion, the u-blox C099-F9P has proven to be very suitable for engineering applications. Regarding the average of the static u-center and SW Maps analyses, the RMSE in the North-direction is 0.90 cm, the RMSE in the East-direction is 0.56 cm and the RMSE in the Up-direction is 2.00 cm. So, the u-blox receiver is capable of providing measurements with centimetre accuracy. Furthermore, u-center and SW Maps offer the possibility to display the results in real-time. Finally, SW Maps is compatible with smartphones, allowing for a practical small-scale set-up.

References

Barr, K. R. (2019, November 22). *Importance of Tides*. Sciencing. https://sciencing.com/importance-tides-7751713.html

DATAGNSS. (2021). *About RTK*. https://docs.datagnss.com/d303-docs/common/about-rtk/

Digi-Key. (2020, May 26). *C099-F9P Application Board*. https://www.digikey.nl/nl/product-highlight/u/u-blox/c099-f9p-application-board

Gurtner, W., & Estey, L. (2007, November). *RINEX*. https://epic.awi.de/id/eprint/29985/1/Gur2007a.pdf

Hwang, J., Yun, H., Suh, Y., Cho, J., & Lee, D. (2012). Development of an RTK-GPS Positioning Application with an Improved Position Error Model for Smartphones. *Sensors, 12*(10), 12988–13001. https://doi.org/10.3390/s121012988

Montenbruck, O., & Teunissen, P. (2017). *Springer Handbook of Global Navigation Satellite Systems*. Springer International Publishing AG. https://doi.org/10.1007/978-3-319-42928-1

NGS. (2017). *Antenna Calibrations*. National Geodetic Survey. https://www.ngs.noaa.gov/ANTCAL/#

NSGI. (n.d.). *NETPOS*. https://www.nsgi.nl/netpos

Oracle. (n.d.). *What Is a Socket?* Oracle Java Documentation. https://docs.oracle.com/javase/tutorial/networking/sockets/definition.html

Ordnance Survey. (n.d.). *Antenna phase centre offsets*. https://www.ordnancesurvey.co.uk/business-government/tools-support/os-net/antenna

Petovello, M. (2013). GNSS Solutions: Multipath vs. NLOS signals. *InsideGNSS*, 40–44. https://www.insidegnss.com/auto/novdec13-Solutions.pdf

Sabet, B. S., & Barani, G. A. (2011). Design of small GPS drifters for current measurements in the coastal zone. *Ocean & Coastal Management, 54*(2), 158–163. https://doi.org/10.1016/j.ocecoaman.2010.10.029

Tiberius, C. C. J. M. (2020). *Introduction to GPS*. Faculty of Civil Engineering and Geosciences - Delft University of Technology.

Trimble. (2007, May 22). *Trimble Introduces Modular GNSS Survey System and Innovative Target*. https://www.trimble.com/news/release.aspx?id=052207a

van der Marel, H. (2020). *Reference Systems for Surveying and Mapping*. Faculty of Civil Engineering and Geosciences - Delft University of Technology.

van der Marel, H. (2021). *Dutch Permanent GNSS Array (DPGA)*. Dgpa. http://gnss1.tudelft.nl/dpga/

Wanninger, L. (2008, June). *Introduction to network RTK*. International Association of Geodesy. http://www.wasoft.de/e/iagwg451/intro/introduction.html

Weber, G., Dettmering, D., Gebhard, H., & Kalafus, R. (2005). Networked Transport of RTCM via Internet Protocol (Ntrip) . . . IP-Streaming for Real-Time GNSS Applications. *Proceedings of the 18th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2005)*, 2243–2247.

Appendix A – u-center configuration

Toolbar

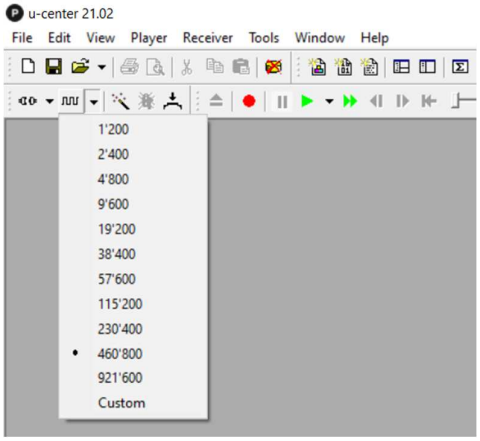

Figure A1: Baudrate

Configuration View

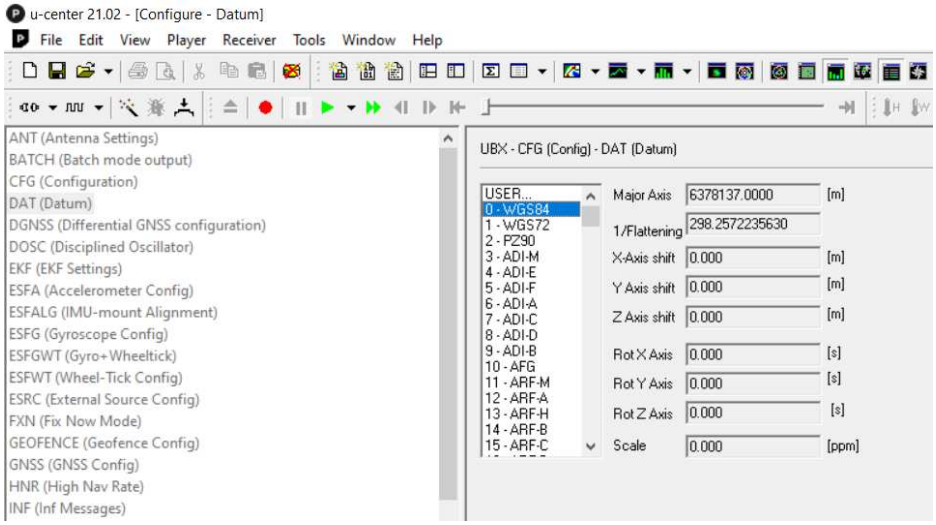

Figure A2: Datum

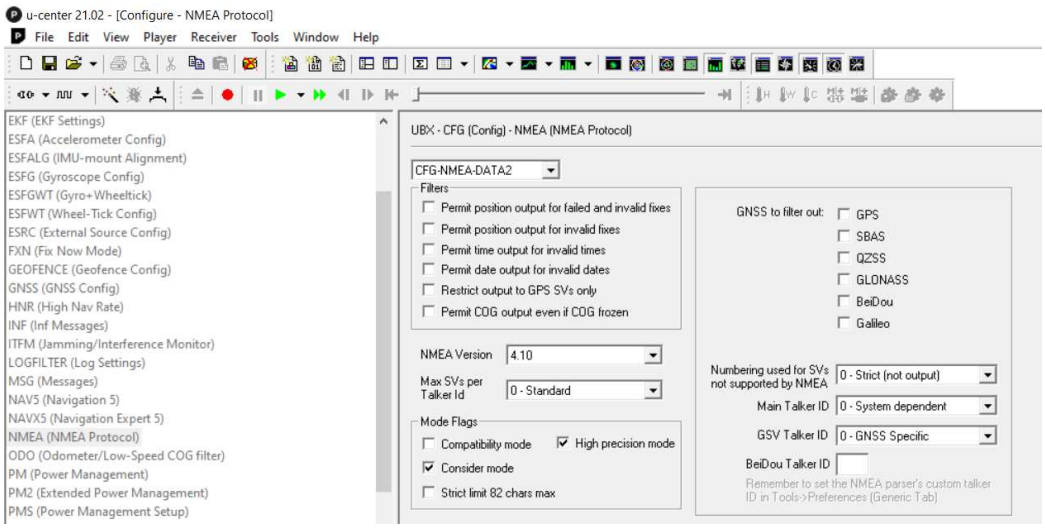

Figure A3: NMEA Protocol

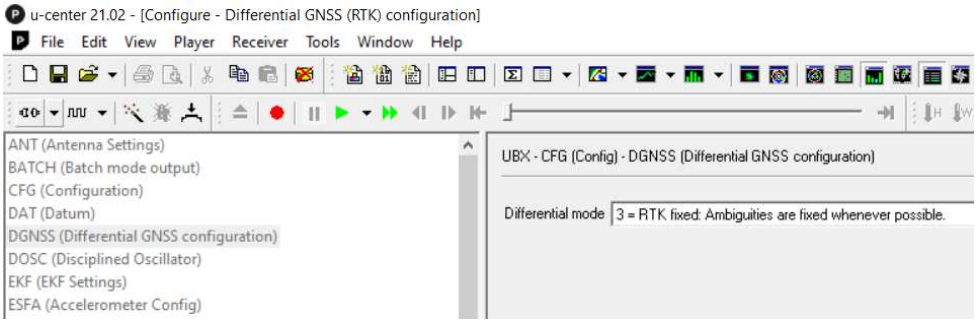

Figure A4: Differential mode

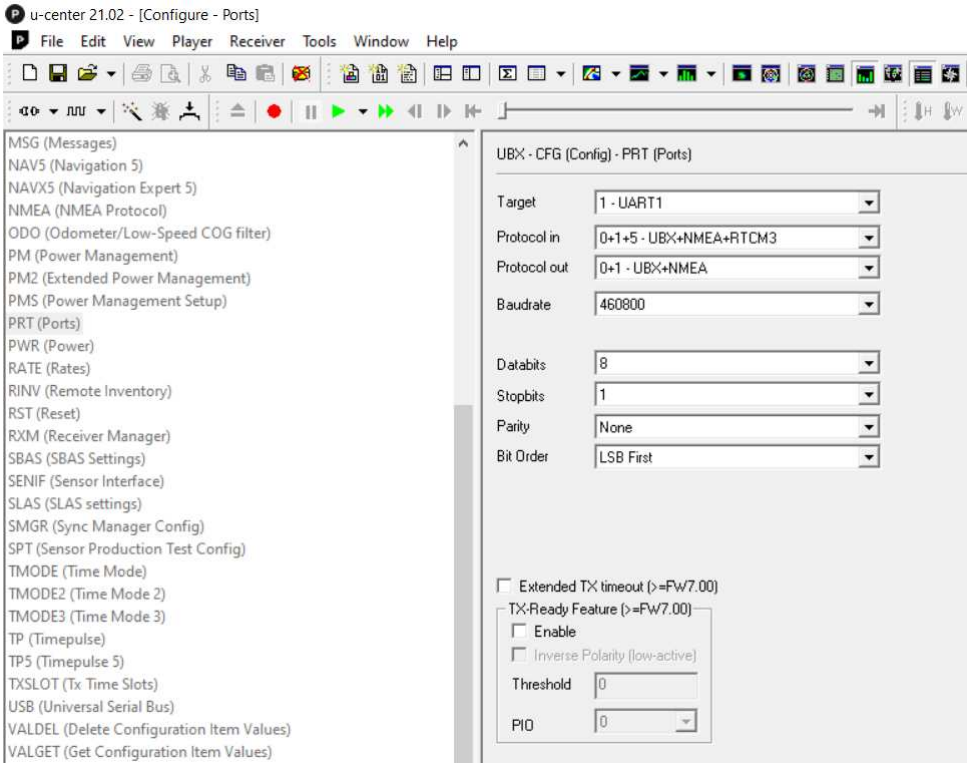

Figure A5: Ports

Messages View

The NMEA and UBX messages should be enabled in the Messages View by right clicking on the message type and selecting ‘Enable Child Messages’.

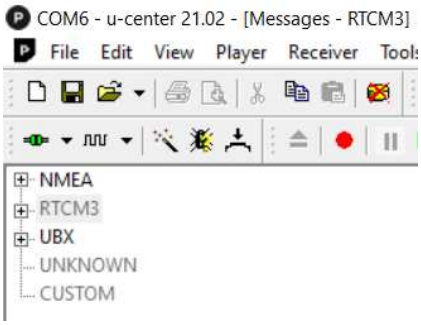

Figure A6: The messages view

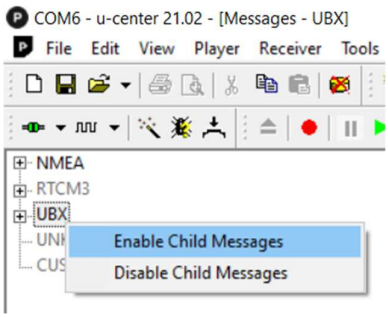

Figure A7: Enable Child Messages

Appendix B – SW Maps configuration

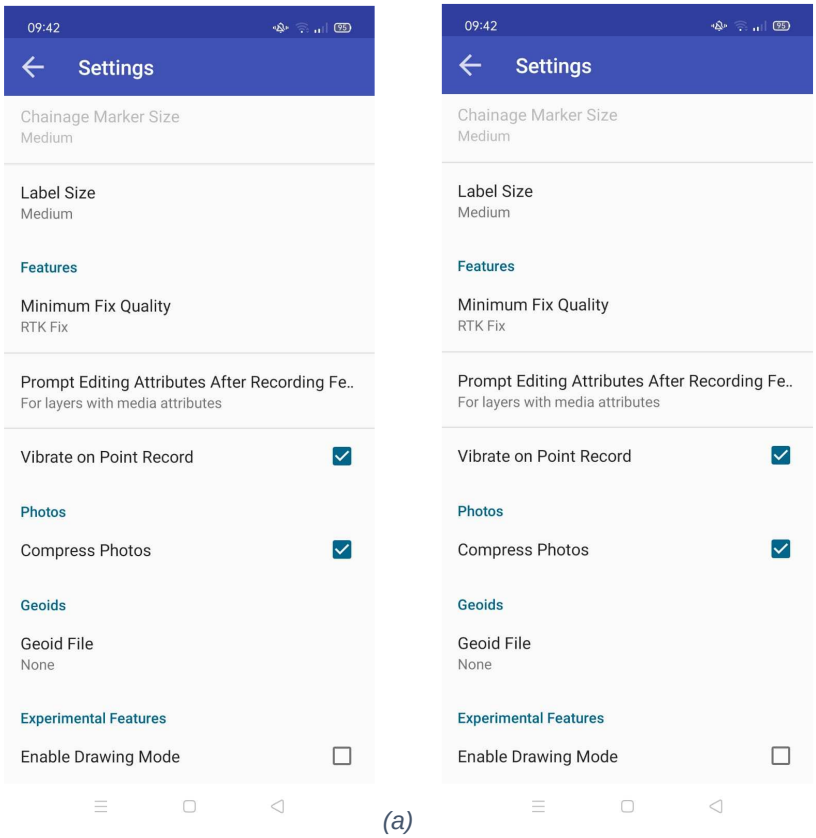

Figure B1a-b: General settings

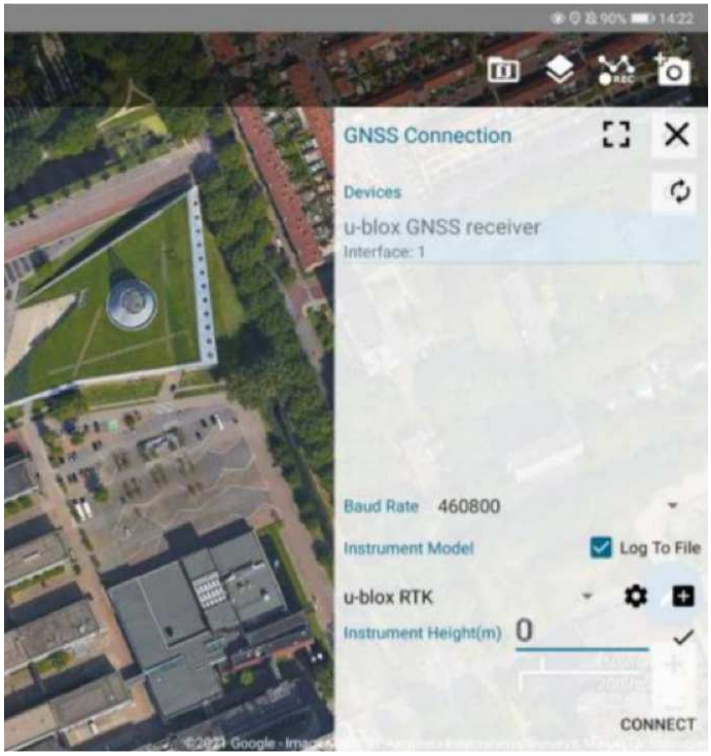

Figure B2: USB Serial GNSS connection settings (Tiberius, personal communications, February 5 2021)

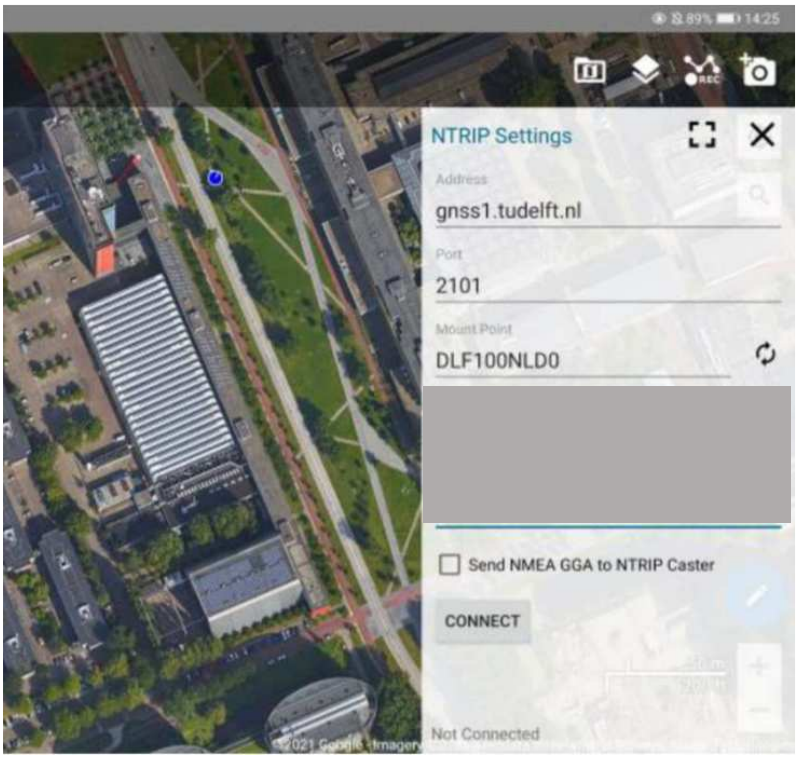

Figure B3: NTRIP settings (Tiberius, personal communications, February 5 2021)

Appendix C – RTKPOST configuration

DLF1 reference receiver

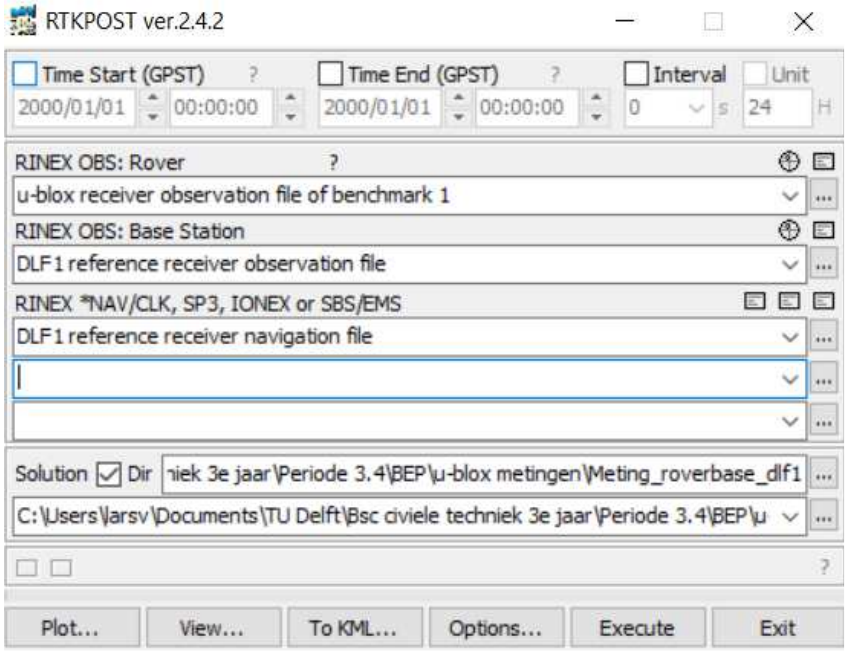

Figure C1: File entries

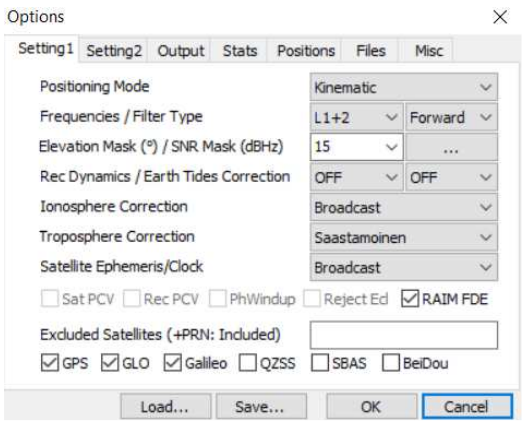

Figure C2: Setting 1 from Options

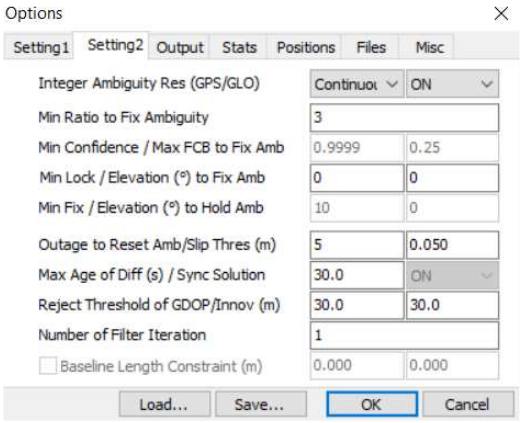

Figure C3: Setting 2 from Options

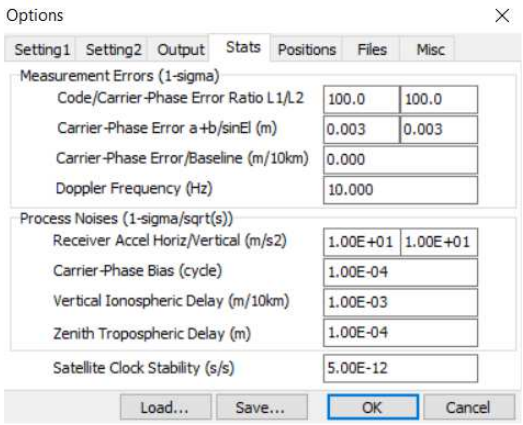

Figure C4: Stats from Options

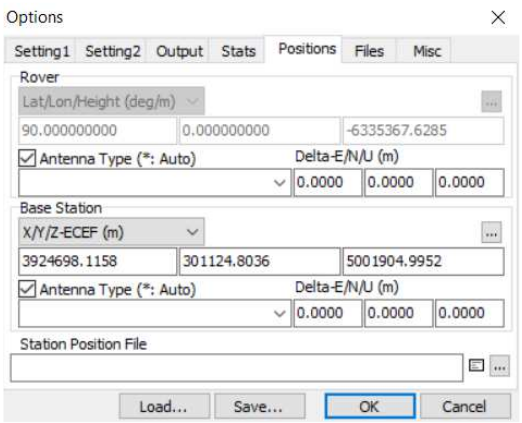

Figure C5: Positions from Options

U-blox reference receiver

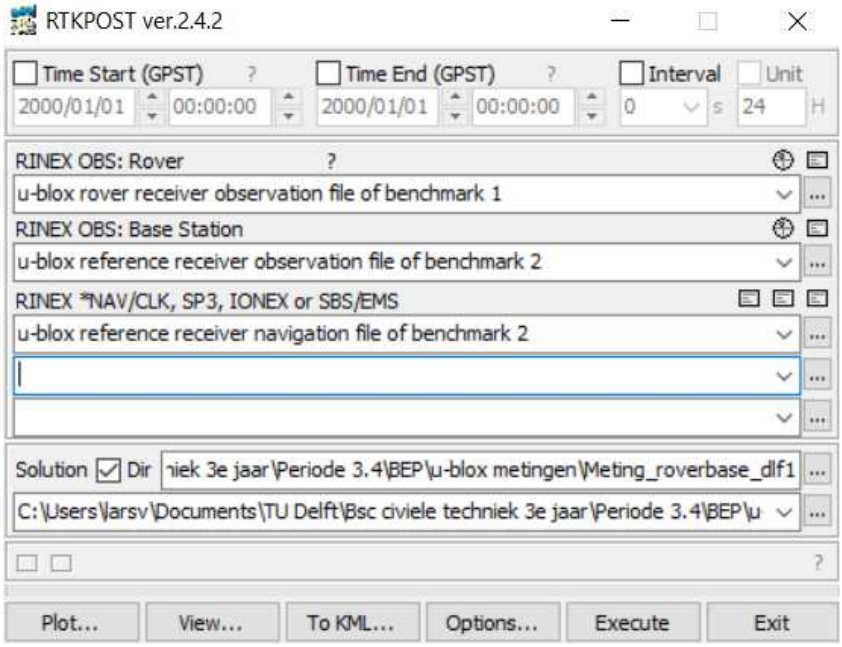

Figure C6: File entries

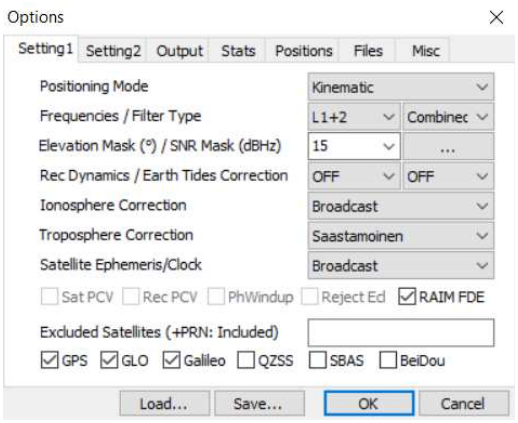

Figure C7: Setting 1 from Options

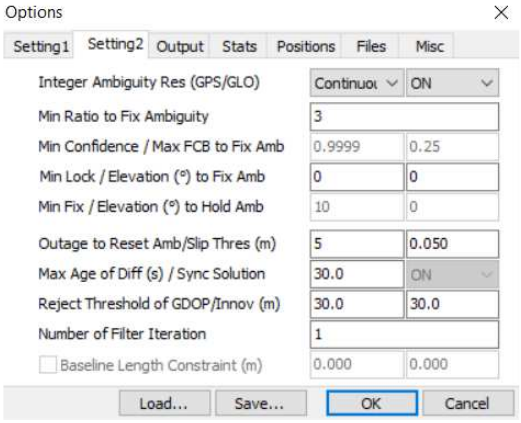

Figure C8: Setting 2 from Options

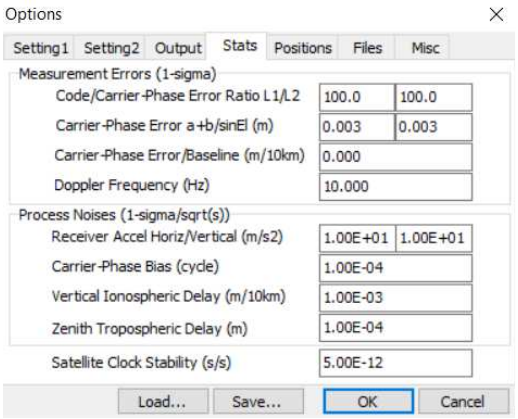

Figure C9: Stats from Options

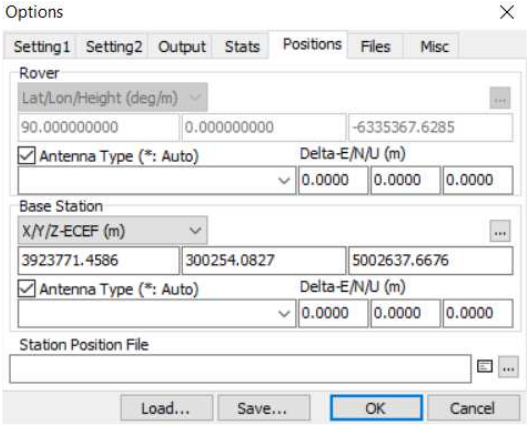

Figure C10: Positions from Options

Appendix D – Python scripts

Static accuracy

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
import pandas as pd
from pandas import read_csv


## Reading the data
# Reading the data from a u-center CSV file
data = read_csv('u-center.csv', index_col=0, sep=',')
display(data)

if len(data['Carrier Range Status'][data['Carrier Range Status'] == 2]) == len(data):
    print(f'No float solutions.')
else:
    print(f'Float solutions exist.')
print(f'The minimum number of used satellites is: {data["SVs Used"].min()}.\n')

Lat = data.Lat
Lon = data.Lon
Alt = data['Alt (HAE)'] + 0.15

# # Reading the data from a KML file  //  Only one of the two codes to read the data is used.
# data = np.loadtxt('KML.txt')
# Lat = data[:, 0]
# Lon = data[:, 1]
# Alt = data[:, 2] + 0.15


## Ground-truth coordinates
lat1_gt, lon1_gt, alt1_gt = 51.99711689, 4.37586839, 43.1605
lat2_gt, lon2_gt, alt2_gt = 51.99708460, 4.37584883, 43.2020
X1_gt, Y1_gt, Z1_gt = 3923768.5081, 300255.2041, 5002639.8468
X2_gt, Y2_gt, Z2_gt = 3923771.4586, 300254.0827, 5002637.6676

# Benchmark 1
lat_gt, lon_gt, alt_gt = lat1_gt, lon1_gt, alt1_gt
X_gt, Y_gt, Z_gt = X1_gt, Y1_gt, Z1_gt

# # Benchmark 2
# lat_gt, lon_gt, alt_gt = lat2_gt, lon2_gt, alt2_gt  # Either the coordinates from benchmark 1 are used or the ones from
# X_gt, Y_gt, Z_gt = X2_gt, Y2_gt, Z2_gt              # benchmark 2.


## Conversion from shperical to XYZ-coordinates
def spherical_XYZ(latitude, longitude, altitude, a, rf):
    """Convert spherical coordinates to Cartesian coordinates.
    a is the value of the semi-major axis in meters and rf is
    the reciprocal of the flattening."""

    X = np.zeros(len(latitude))
    Y = np.zeros(len(latitude))
    Z = np.zeros(len(latitude))
    e2 = 2 / rf - (1 / rf) ** 2  # eccentricity squared
    for i in range(len(latitude)):
        phi = np.radians(latitude[i])
        lamda = np.radians(longitude[i])
        n = a / np.sqrt(1 - e2 * np.sin(phi) ** 2)  # radius of curvature in the prime vertical
        X[i] = (n + altitude[i]) * np.cos(phi) * np.cos(lamda)
        Y[i] = (n + altitude[i]) * np.cos(phi) * np.sin(lamda)
        Z[i] = (n * (1 - e2) + altitude[i]) * np.sin(phi)
    return [X, Y, Z]

X, Y, Z = spherical_XYZ(Lat, Lon, Alt, 6378137, 298.257222101)


## Conversion from degrees to meters
a = 6378137  # semi-major axis
rf = 298.257222101  # reciprocal flattening
e2 = 2 / rf - (1 / rf) ** 2  # eccentricity squared
def n(phi):
    n = a / np.sqrt(1 - e2 * np.sin(np.radians(phi)) ** 2)  # radius of curvature in the prime vertical
    return n
def m(phi):
    m = a * (1 - e2) / (1 - e2 * np.sin(np.radians(phi)) ** 2) ** 1.5  # radius of curvature in the meridian
    return m

dN = np.zeros(len(data))
dE = np.zeros(len(data))
for i in range(len(data)):
    dphi = np.radians(Lat[i] - lat_gt)
    dlamda = np.radians(Lon[i] - lon_gt)
    dN[i] = (m(Lat[i]) + Alt[i]) * dphi
    dE[i] = (n(Lat[i]) + Alt[i]) * np.cos(np.radians(Lat[i])) * dlamda
dH = Alt - alt_gt

```

```python
## Mean
# Spherical coordinates
lat_mean = np.mean(Lat)
lon_mean = np.mean(Lon)
alt_mean = np.mean(Alt)

# Northing, Easting and differential height (or Up)
dN_mean = (m(lat_mean) + alt_mean) * np.radians(lat_mean - lat_gt)
dE_mean = (n(lat_mean) + alt_mean) * np.cos(np.radians(lat_mean)) * np.radians(lon_mean - lon_gt)
dH_mean = np.mean(dH)

# Distance to ground-truth coordinates
dist_2d = np.zeros(len(data))
dist_3d = np.zeros(len(data))
for i in range(len(data)):
    dist_2d[i] = np.sqrt(dE[i] ** 2 + dN[i] ** 2)
    dist_3d[i] = np.sqrt((X[i] - X_gt) ** 2 + (Y[i] - Y_gt) ** 2 + (Z[i] - Z_gt) ** 2)
dist_2d_mean = np.mean(dist_2d)
dist_3d_mean = np.mean(dist_3d)


## Standard deviation and Root Mean Squared Error
# Spherical coordinates
dN_var = 1 / (len(data) - 1) * np.sum((dN - dN_mean) ** 2)  # Unbiased sample variance
dN_std = np.sqrt(dN_var)  # Standard deviation
dN_MSE = 1 / len(data) * np.sum(dN ** 2)  # Empirical MSE
dN_RMSE = np.sqrt(dN_MSE)  # RMSE

dE_var = 1 / (len(data) - 1) * np.sum((dE - dE_mean) ** 2)
dE_std = np.sqrt(dE_var)
dE_MSE = 1 / len(data) * np.sum(dE ** 2)
dE_RMSE = np.sqrt(dE_MSE)

dH_var = 1 / (len(data) - 1) * np.sum((dH - dH_mean) ** 2)
dH_std = np.sqrt(dH_var)
dH_MSE = 1 / len(data) * np.sum(dH ** 2)
dH_RMSE = np.sqrt(dH_MSE)

# Distance to ground-truth coordinates
dist_3d_var = 1 / (len(data) - 1) * np.sum((dist_3d - dist_3d_mean) ** 2)
dist_3d_std = np.sqrt(dist_3d_var)
dist_3d_MSE = 1 / len(data) * np.sum(dist_3d ** 2)
dist_3d_RMSE = np.sqrt(dist_3d_MSE)
dist_2d_var = 1 / (len(data) - 1) * np.sum((dist_2d - dist_2d_mean) ** 2)
dist_2d_std = np.sqrt(dist_2d_var)
dist_2d_MSE = 1 / len(data) * np.sum(dist_2d ** 2)
dist_2d_RMSE = np.sqrt(dist_2d_MSE)


## Plots
# Northing
plt.figure(figsize=(16, 8))
plt.plot(np.arange(0, len(data), 1), dN, label='Northing')
plt.plot(np.arange(0, len(data), 1), np.full(len(data), dN_std+dN_mean), color='red', ls='--', label='Standard deviation')
plt.plot(np.arange(0, len(data), 1), np.full(len(data), -dN_std+dN_mean), ls='--', color='red')
plt.plot(np.arange(0, len(data), 1), np.full(len(data), dN_mean), color='black', ls=':', label='Mean')
plt.ylim(-0.04, 0.04)
plt.legend(loc='best')
plt.ylabel('Northing (m)', fontsize=12)
plt.xlabel('Epoch', fontsize=12)
plt.title('Northing with respect to the groud-truth latitude of benchmark 1', fontsize=15);

# Easting
plt.figure(figsize=(16, 8))
plt.plot(np.arange(0, len(data), 1), dE, label='Easting')
plt.plot(np.arange(0, len(data), 1), np.full(len(data), dE_std+dE_mean), color='red', ls='--', label='Standard deviation')
plt.plot(np.arange(0, len(data), 1), np.full(len(data), -dE_std+dE_mean), ls='--', color='red')
plt.plot(np.arange(0, len(data), 1), np.full(len(data), dE_mean), color='black', ls=':', label='Mean')
plt.ylim(-0.04, 0.04)
plt.legend(loc='best')
plt.ylabel('Easting (m)', fontsize=12)
plt.xlabel('Epoch', fontsize=12)
plt.title('Easting with respect to the ground-truth longitude of benchmark 1', fontsize=15)

# 2D-distance
plt.figure(figsize=(16, 8))
plt.plot(np.arange(0, len(data), 1), dist_2d, label='2D-distance')
plt.plot(np.arange(0, len(data), 1), np.full(len(data), dist_2d_std+dist_2d_mean), color='red', ls='--', label='Standard de
plt.plot(np.arange(0, len(data), 1), np.full(len(data), -dist_2d_std+dist_2d_mean), ls='--', color='red')
plt.plot(np.arange(0, len(data), 1), np.full(len(data), dist_2d_mean), color='black', ls=':', label='Mean')
plt.ylim(-0.04, 0.04)
plt.legend(loc='best')
plt.ylabel('2D-distance (m)', fontsize=12)
plt.xlabel('Epoch', fontsize=12)
plt.title('2D-distance to the ground-truth coordinates of benchmark 1', fontsize=15);

# 3D-distance
plt.figure(figsize=(16, 8))
plt.plot(np.arange(0, len(data), 1), dist_3d, label='3D-distance')
plt.plot(np.arange(0, len(data), 1), np.full(len(data), dist_3d_std+dist_3d_mean), color='red', ls='--', label='Standard de
plt.plot(np.arange(0, len(data), 1), np.full(len(data), -dist_3d_std+dist_3d_mean), ls='--', color='red')
plt.plot(np.arange(0, len(data), 1), np.full(len(data), dist_3d_mean), color='black', ls=':', label='Mean')
plt.ylim(-0.05, 0.1)
```

```
plt.legend(loc='best')
plt.ylabel('3D-distance (m)', fontsize=12)
plt.xlabel('Epoch', fontsize=12)
plt.title('3D-distance to the ground-truth coordinates of benchmark 1', fontsize=15)

# Height
plt.figure(figsize=(16, 8))
plt.plot(np.arange(0, len(data), 1), Alt-alt_gt, label='Height')
plt.plot(np.arange(0, len(data), 1), np.full(len(data), dH_std+dH_mean), color='red', ls='--', label='Standard deviation')
plt.plot(np.arange(0, len(data), 1), np.full(len(data), -dH_std+dH_mean), ls='--', color='red')
plt.plot(np.arange(0, len(data), 1), np.full(len(data), dH_mean), color='black', ls=':', label='Mean')
plt.ylim(-0.05, 0.1)
plt.legend(loc='best')
plt.ylabel('Height (m)', fontsize=12)
plt.xlabel('Epoch', fontsize=12)
plt.title('Height with respect to the ground-truth height of benchmark 1', fontsize=15)

# Scatter plot - Easting vs Northing
plt.figure(figsize=(10, 10))
plt.plot(dE, dN, color='black', marker='o', ls='', ms=1, label='Measurements')
plt.plot(0, 0, marker='*', ls='', ms=10, label='Ground-truth')
plt.plot(dE_mean, dN_mean, color='orange', marker='o', ls='', ms=8, label='Mean')
plt.plot(np.full(1000, dE_mean+dE_std), np.linspace(-0.04, 0.04, 1000), color='red', ls='--', label='Standard deviation')
plt.plot(np.full(1000, dE_mean-dE_std), np.linspace(-0.04, 0.04, 1000), color='red', ls='--')
plt.plot(np.linspace(-0.04, 0.04, 1000), np.full(1000, dN_mean+dN_std), color='red', ls='--')
plt.plot(np.linspace(-0.04, 0.04, 1000), np.full(1000, dN_mean-dN_std), color='red', ls='--')
plt.xlim(-0.04, 0.04)
plt.ylim(-0.04, 0.04)
plt.legend()
plt.xlabel('Easting (m)', fontsize=12)
plt.ylabel('Northing (m)', fontsize=12)
plt.title('Easting vs Northing', fontsize=15)

# 3D-plot
fig = plt.figure(figsize=(10, 10))
ax = plt.axes(projection='3d')
ax.plot3D(dE, dN, Alt-alt_gt, color='black', ls='', marker='o', ms=1, label='Measurements');
ax.scatter3D(0, 0, 0, color='red', marker='*', s=40, label='Ground-truth')
ax.scatter3D(dE_mean, dN_mean, alt_mean-alt_gt, color='yellow', marker='o', s=40, label='Mean')
plt.legend()
ax.set_xlim(-0.04, 0.04)
ax.set_ylim(-0.04, 0.04)
ax.set_zlim(-0.05, 0.1)
ax.set_xlabel('Easting (m)', fontsize=12)
ax.set_ylabel('Northing (m)', fontsize=12)
ax.set_zlabel('Height (m)', fontsize=12)
plt.title('3D-plot of the measurements of benchmark 1', fontsize=15, loc='right')
ax.view_init(30, 130)
```

Kinematic accuracy

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
import pandas as pd
from pandas import read_csv


## Load u-center data from the kinematic test
data = read_csv('Kinematic.csv', index_col=0, sep=',')

if len(data['Carrier Range Status'][data['Carrier Range Status'] == 2]) == len(data):
    print(f'No float solutions.')
else:
    print(f'Float solutions exist.')
print(f'The minimum number of used satellites is: {data["SVs Used"].min()}')

Lat = data.Lat
Lon = data.Lon
Alt = data['Alt (HAE)'] + 0.15


## Ground-truth coordinates
lat1_gt, lon1_gt, alt1_gt = 51.99711689, 4.37586839, 43.1605 + 0.036
lat2_gt, lon2_gt, alt2_gt = 51.99708460, 4.37584883, 43.2020 + 0.036


## Conversion from degrees to meters
a = 6378137  # semi-major axis
rf = 298.257222101  # reciprocal flattening
e2 = 2 / rf - (1 / rf) ** 2  # eccentricity squared
def n(phi):
    n = a / np.sqrt(1 - e2 * np.sin(np.radians(phi)) ** 2)  # radius of curvature in the prime vertical
    return n
def m(phi):
    m = a * (1 - e2) / (1 - e2 * np.sin(np.radians(phi)) ** 2) ** 1.5  # radius of curvature in the meridian
    return m

dN = np.zeros(len(data))
dE = np.zeros(len(data))
```

```
for i in range(len(data)):
    dphi = np.radians(Lat[i] - lat2_gt)  # Nail 2 as reference
    dlamda = np.radians(Lon[i] - lon2_gt)  # Nail 2 as reference
    dN[i] = (m(Lat[i]) + Alt[i]) * dphi
    dE[i] = (n(Lat[i]) + Alt[i]) * np.cos(np.radians(Lat[i])) * dlamda

dN_nail1 = (m(lat1_gt) + alt1_gt) * np.radians(lat1_gt - lat2_gt)
dE_nail1 = (n(lat1_gt) + alt1_gt) * np.cos(np.radians(lat1_gt)) * np.radians(lon1_gt - lon2_gt)
dN_nail2 = 0
dE_nail2 = 0


## Plot
# Easting vs Northing
plt.figure(figsize=(16, 10))
plt.plot(dE, dN, marker='o', ls='', ms=2, label='Measurements')
x = np.linspace(np.min(dE)-0.03, np.max(dE)+0.03, 1000)
a = (dN_nail2 - dN_nail1) / (dE_nail2 - dE_nail1)
b = a * dE_nail1 - dN_nail1
y = a * x - b
plt.plot(x, y, label='Ground-truth')
plt.legend()
plt.xlabel('Easting (m)', fontsize=12)
plt.ylabel('Northing (m)', fontsize=12)
plt.title('Easting vs Northing', fontsize=15);


## 2D-distance (Northing and Easting) and cross-track error calculation
p1 = np.asarray((x[0], y[0]))
p2 = np.asarray((x[-1], y[-1]))
dist_2d = np.zeros(len(data))
crosste = np.zeros(len(data))
check_y = np.zeros(len(data))
for i in range(len(data)):
    p3 = np.asarray((dE[i], dN[i]))
    dist_2d[i] = np.abs(np.cross(p2-p1, p1-p3)) / np.linalg.norm(p2-p1)
    check_y[i] = a * dE[i] + b
    if dN[i] <= check_y[i]:
        crosste[i] = -np.abs(np.cross(p2-p1, p1-p3)) / np.linalg.norm(p2-p1)
    else:
        crosste[i] = np.abs(np.cross(p2-p1, p1-p3)) / np.linalg.norm(p2-p1)

## 3D-distance calculation
# Ground-truth line: y = ax + b
# Perpendicular line from a random measurement to ground-truth line: y = -1/a * x + d

dist_nail12 = np.sqrt((dE_nail2 - dE_nail1) ** 2 + (dN_nail2 - dN_nail1) ** 2)  # Distance between benchmark 1 and 2
height_nail12 = alt2_gt - alt1_gt  # Height difference between benchmark 1 and 2
dist_3d = np.zeros(len(data))
for i in range(len(data)):

    # Obtain intersection between ground-truth line and line perpendicular to random measurement
    d = dN[i] + 1 / a * dE[i]
    int_x = (a * d - a * b) / (a ** 2 + 1)
    int_y = a * int_x + b

    # Determine height of the intersection point
    dist_nail1 = np.sqrt((int_x - dE_nail1) ** 2 + (int_y - dN_nail1) ** 2)  # Distance intersection point to benchmark 1
    int_height = dist_nail1 / dist_nail12 * height_nail12 + alt1_gt  # Height ideal line at intersection point

    # Determine difference of the height of the intersection point ('ideal height') and
    # the heigth of the measurement point ('real height')
    height_dif = int_height - Alt[i]

    # Absolute 3D-distance
    dist_3d[i] = np.sqrt(dist_2d[i] ** 2 + height_dif ** 2)


## Statistical quantities
# 2D-distance
dist_2d_mean = np.mean(dist_2d)
dist_2d_var = 1 / (len(dist_2d) - 1) * np.sum((dist_2d - dist_2d_mean) ** 2)
dist_2d_std = np.sqrt(dist_2d_var)
dist_2d_MSE = 1 / len(dist_2d) * np.sum(dist_2d ** 2)
dist_2d_RMSE = np.sqrt(dist_2d_MSE)

# Cross-track error
crosste_mean = np.mean(crosste)
crosste_var = 1 / (len(crosste) - 1) * np.sum((crosste - crosste_mean) ** 2)
crosste_std = np.sqrt(crosste_var)
crosste_MSE = 1 / len(crosste) * np.sum(crosste ** 2)
crosste_RMSE = np.sqrt(crosste_MSE)

# 3D-distance
dist_3d_mean = np.mean(dist_3d)
dist_3d_var = 1 / (len(dist_3d) - 1) * np.sum((dist_3d - dist_3d_mean) ** 2)
dist_3d_std = np.sqrt(dist_3d_var)
dist_3d_MSE = 1 / len(dist_3d) * np.sum(dist_3d ** 2)
dist_3d_RMSE = np.sqrt(dist_3d_MSE)
```

```
## Plots
# 2D-distance
plt.figure(figsize=(16, 8))
plt.plot(np.arange(0, len(data), 1), dist_2d, label='2D-distance')
plt.plot(np.arange(0, len(data), 1), np.full(len(data), dist_2d_std+dist_2d_mean), color='red', ls='--', label='Standard de
plt.plot(np.arange(0, len(data), 1), np.full(len(data), -dist_2d_std+dist_2d_mean), ls='--', color='red')
plt.plot(np.arange(0, len(data), 1), np.full(len(data), dist_2d_mean), color='black', ls=':', label='Mean')
plt.ylim(-0.04, 0.04)
plt.legend(loc='best')
plt.ylabel('2D-distance (m)', fontsize=12)
plt.xlabel('Epoch', fontsize=12)
plt.title('2D-distance to the ground-truth line through the benchmarks', fontsize=15)

# Cross-track error
plt.figure(figsize=(16, 8))
plt.plot(np.arange(0, len(data), 1), crosste, label='Cross-track error')
plt.plot(np.arange(0, len(data), 1), np.full(len(data), crosste_std+crosste_mean), color='red', ls='--', label='Standard de
plt.plot(np.arange(0, len(data), 1), np.full(len(data), -crosste_std+crosste_mean), ls='--', color='red')
plt.plot(np.arange(0, len(data), 1), np.full(len(data), crosste_mean), color='black', ls=':', label='Mean')
plt.ylim(-0.04, 0.04)
plt.legend(loc='best')
plt.ylabel('Cross-track error (m)', fontsize=12)
plt.xlabel('Epoch', fontsize=12)
plt.title('Cross-track error to the ground-truth line through the benchmarks', fontsize=15)

# 3D-distance
plt.figure(figsize=(16, 8))
plt.plot(np.arange(0, len(data), 1), dist_3d, label='3D-distance')
plt.plot(np.arange(0, len(data), 1), np.full(len(data), dist_3d_std+dist_3d_mean), color='red', ls='--', label='Standard de
plt.plot(np.arange(0, len(data), 1), np.full(len(data), -dist_3d_std+dist_3d_mean), ls='--', color='red')
plt.plot(np.arange(0, len(data), 1), np.full(len(data), dist_3d_mean), color='black', ls=':', label='Mean')
plt.ylim(-0.05, 0.1)
plt.legend(loc='best')
plt.ylabel('3D-distance (m)', fontsize=12)
plt.xlabel('Epoch', fontsize=12)
plt.title('3D-distance to the ground-truth line through the benchmarks', fontsize=15);
```

Influence of the surroundings

The majority of the code that has been used for this experiment is the same as the code for the analysis of the static accuracy. The code that is specific for the RTK fix experiment is presented below.

```
## Reading the data
data = read_csv('Snelheid_fix.csv', sep=',')

# All measurements
Lat = data.Lat
Lon = data.Lon
Alt = data['Alt (HAE)'] + 0.15

# Just the measurements with an RTK fix  //  Either this code should be used or the code for 'All measurements'.
Latt = []
Lonn = []
Altt = []
for i in range(len(data)):
    if data['Carrier Range Status'][i] == 2:
        Latt.append(data.Lat[i])
        Lonn.append(data.Lon[i])
        Altt.append(data['Alt (HAE)'][i] + 0.15)
Lat = np.array(Latt)
Lon = np.array(Lonn)
Alt = np.array(Altt)


## Solution status
Stat = data['Carrier Range Status']
Stat_mean = np.mean(Stat)

Stat_fix = 0
for i in range (len(data)):
    if Stat[i] == 2:
        Stat_fix += 1
    else:
        Stat_fix += 0
fixratio = Stat_fix / len(data)


## Number of satellites used in the position solution
SVused = data['SVs Used']
SVused_mean = np.mean(SVused)
SVused_min = np.min(SVused)


## Plots
# Solution status
plt.figure(figsize=(16, 8))
plt.plot(np.arange(0, len(data), 1), Stat, marker='o', ls='', ms=2)
plt.ylabel('Soltution status', fontsize=12)
plt.xlabel('Epoch', fontsize=12)
plt.title('Solution status of the RTK fix experiment', fontsize=15);

# Number of satellites used in the position solution
plt.figure(figsize=(16, 8))
plt.plot(np.arange(0, len(data), 1), SVused)
plt.ylabel('Number of satellites used', fontsize=12)
plt.xlabel('Epoch', fontsize=12)
plt.title('Number of satellites used in the position solution', fontsize=15);
```

Appendix E – u-center analysis

Benchmark 1

For all the plots that follow, also in the other appendices, two different scales are used. For the Northing, Easting and 2D-distance (in the Northing Easting plane) plots, the scale on the axes runs from -0.04 m – 0.04 m. For the height and the 3D-distance plots, the scale on the axes runs from -0.05 m – 0.1 m.

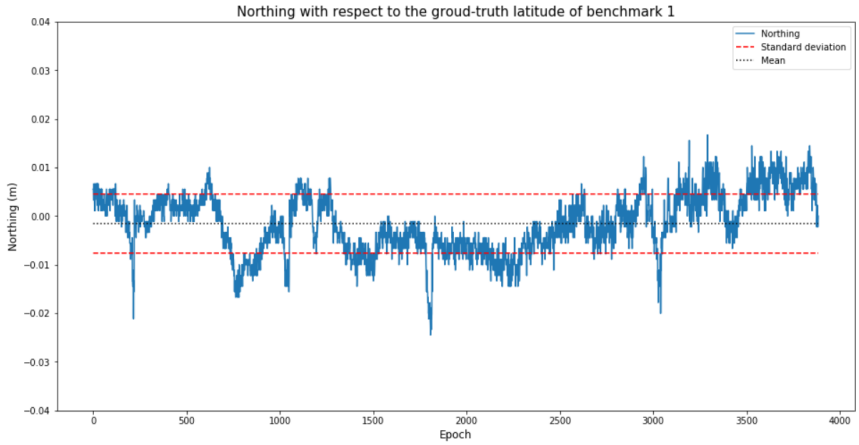

Figure E1: The Northing of benchmark 1 (u-center)

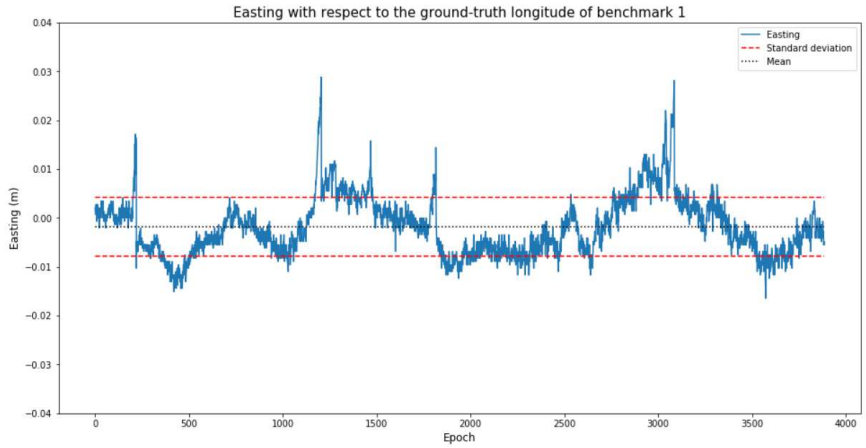

Figure E2: The Easting of benchmark 1 (u-center)

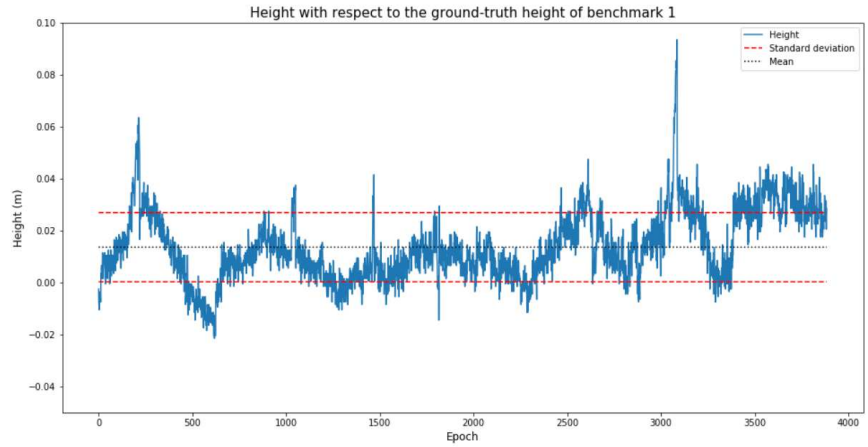

Figure E3: The height of benchmark 1 (u-center)

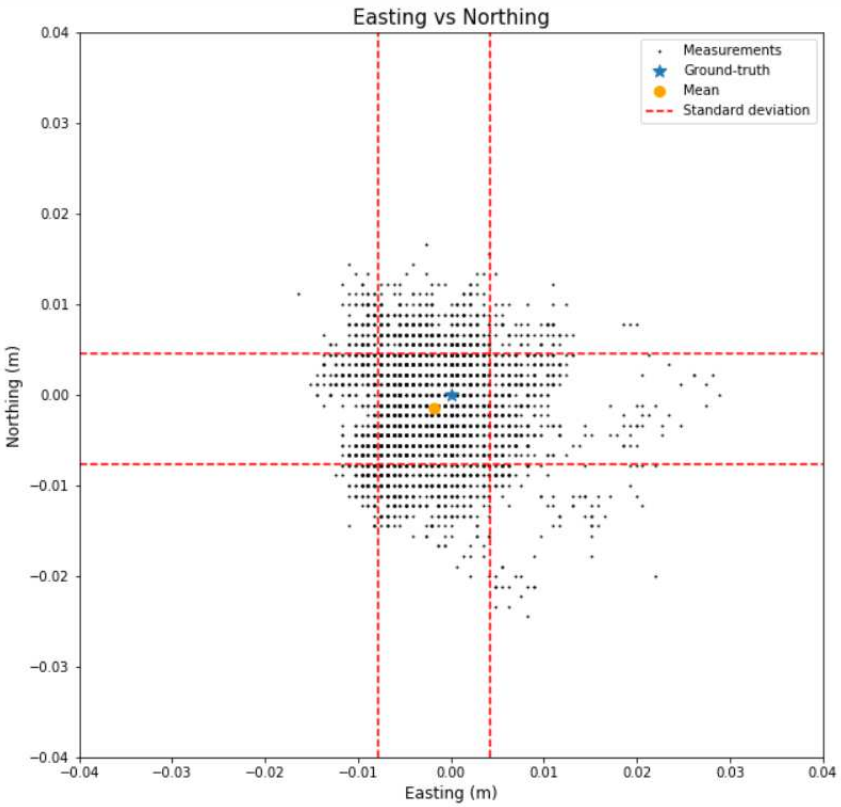

Figure E4: Scatter plot of benchmark 1 (u-center)

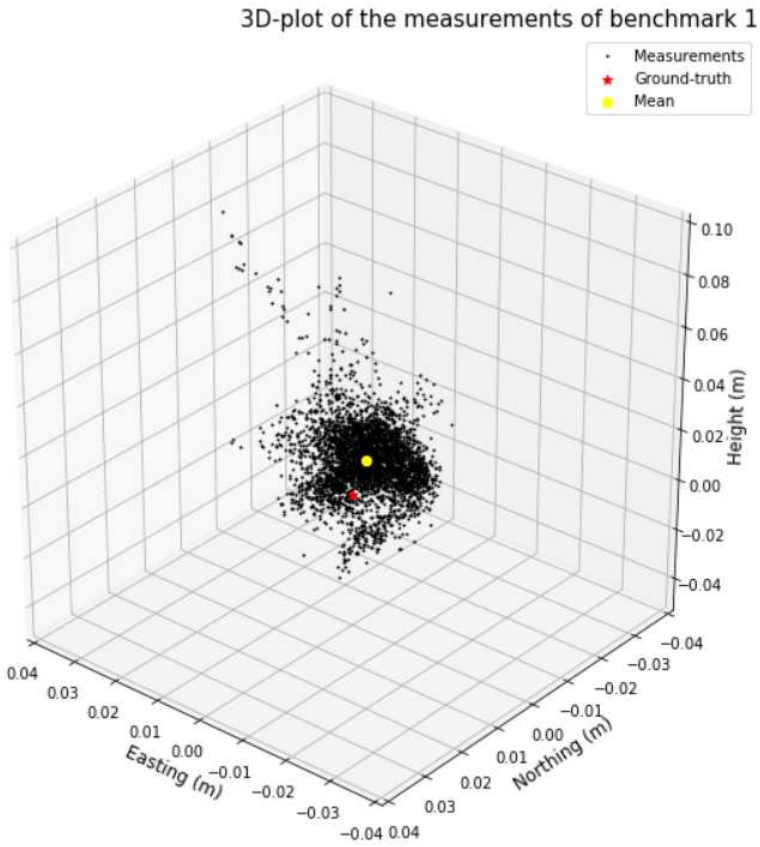

Figure E5: 3D-plot of the measurements of benchmark 1 (u-center)

Benchmark 2

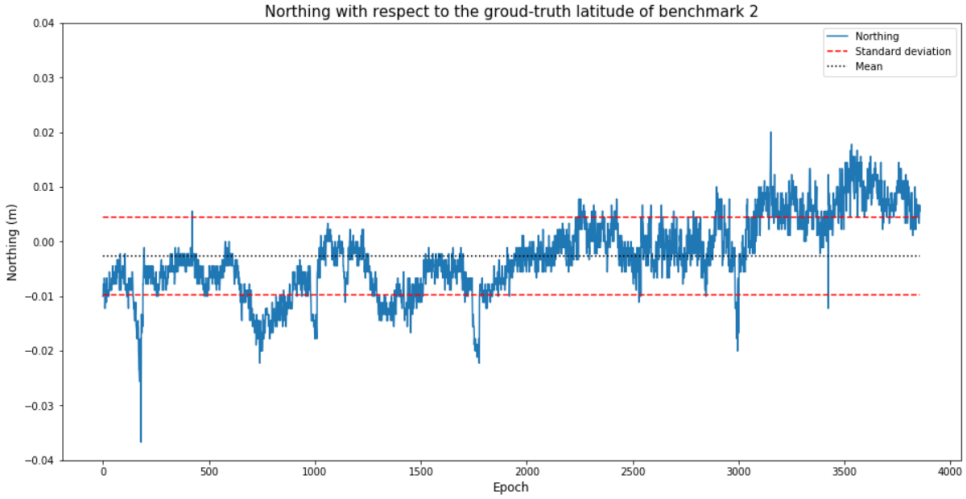

Figure E6: The Northing of benchmark 2 (u-center)

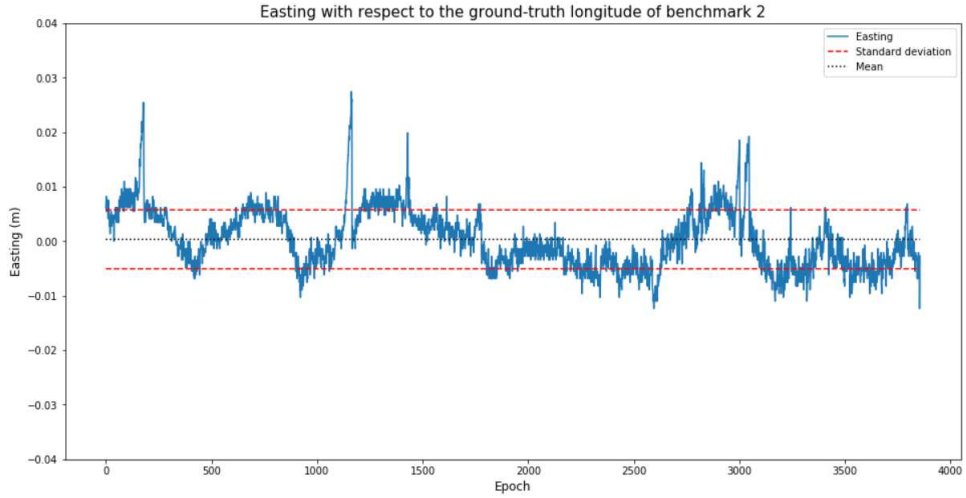

Figure E7: The Easting of benchmark 2 (u-center)

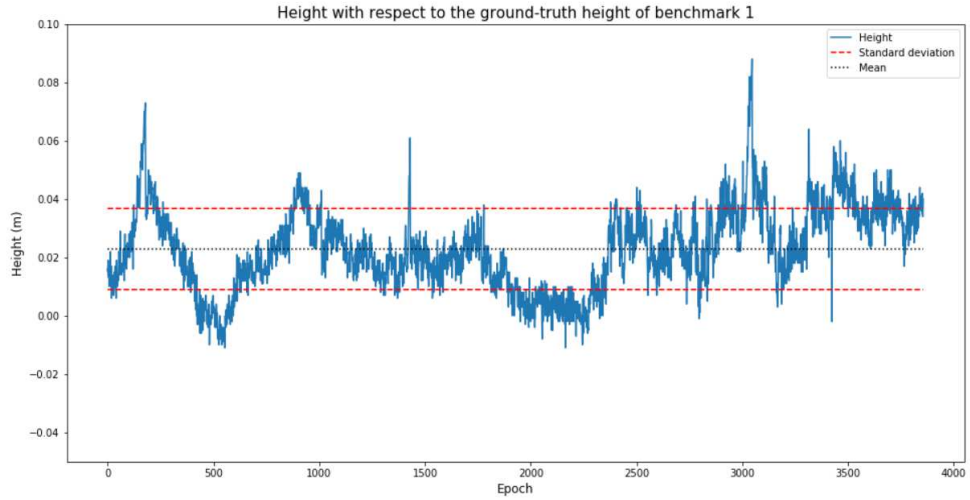

Figure E8: The height of benchmark 2 (u-center)

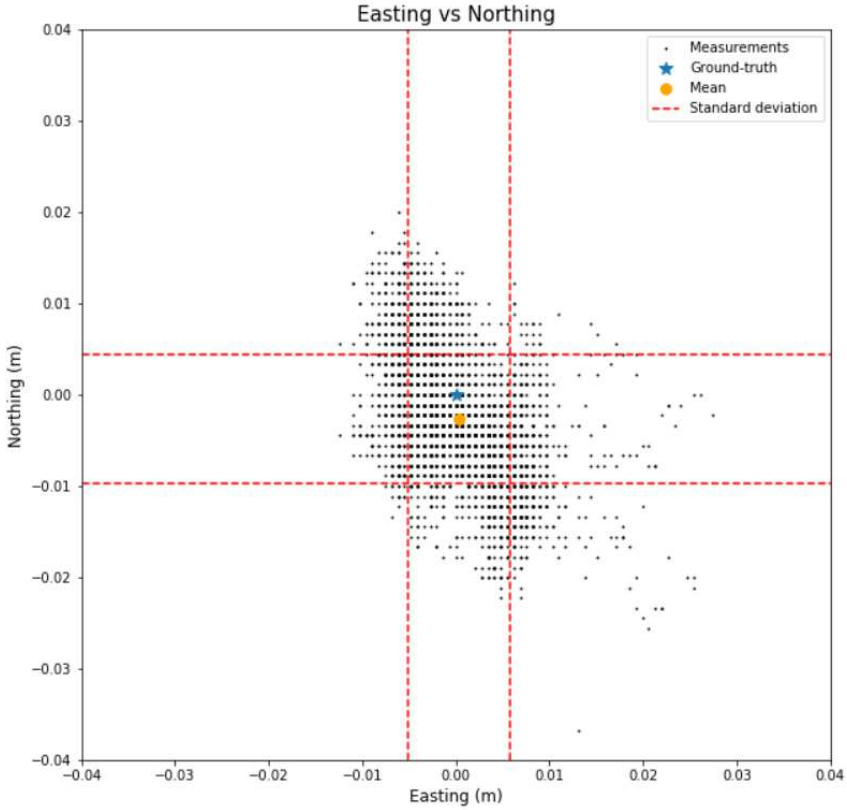

Figure E9: Scatter plot of benchmark 2 (u-center)

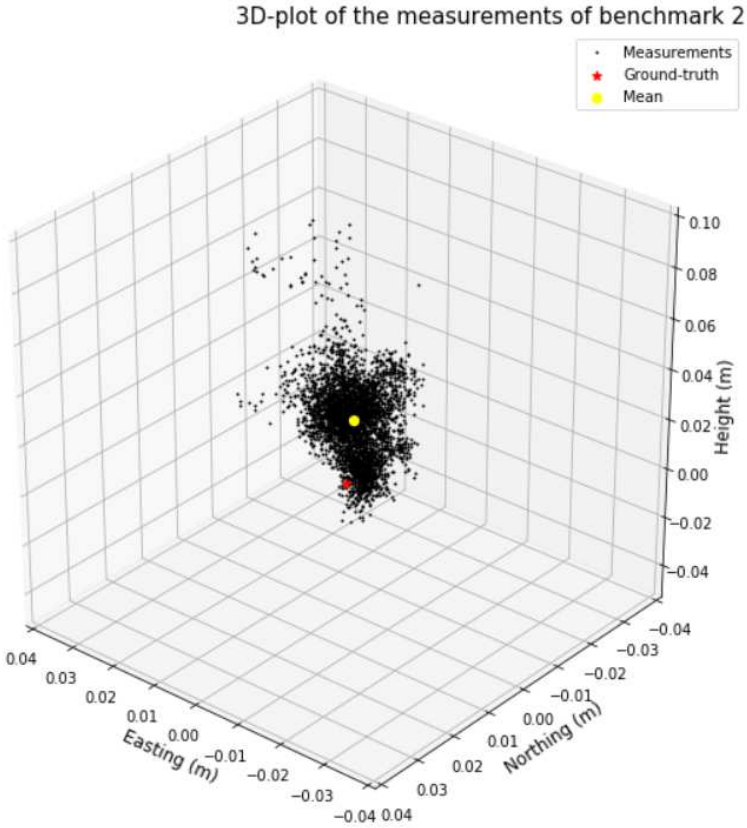

Figure E10: 3D-plot of benchmark 2 (u-center)

Appendix F – SW Maps analysis

Benchmark 1

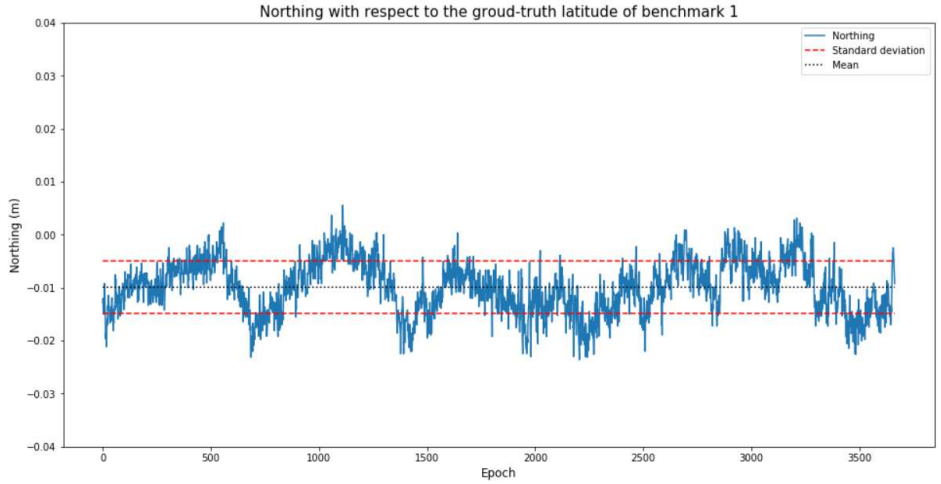

Figure F1: The Northing of benchmark 1 (SW Maps)

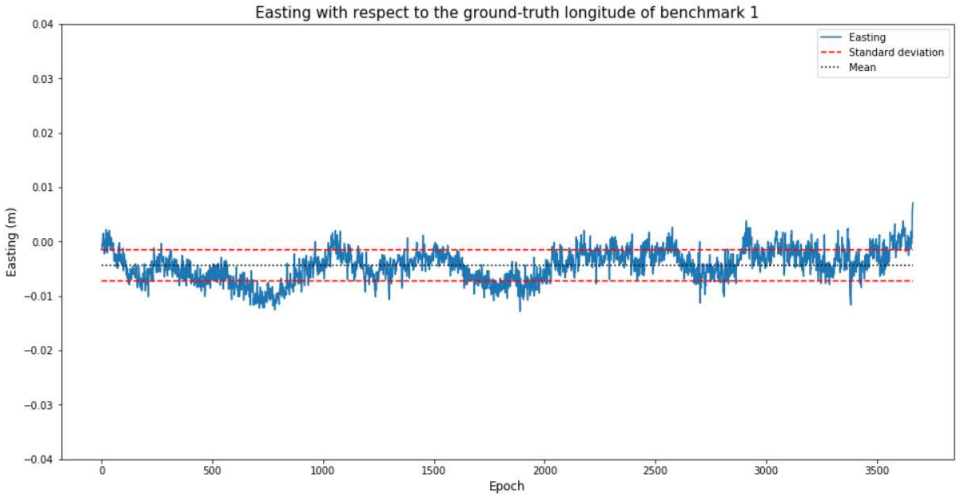

Figure F2: The Easting of benchmark 1 (SW Maps)

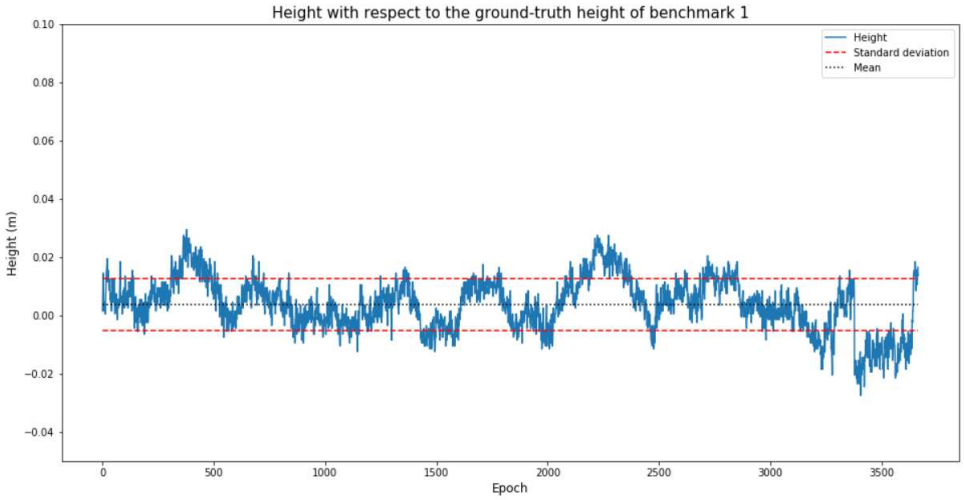

Figure F3: The height of benchmark 1 (SW Maps)

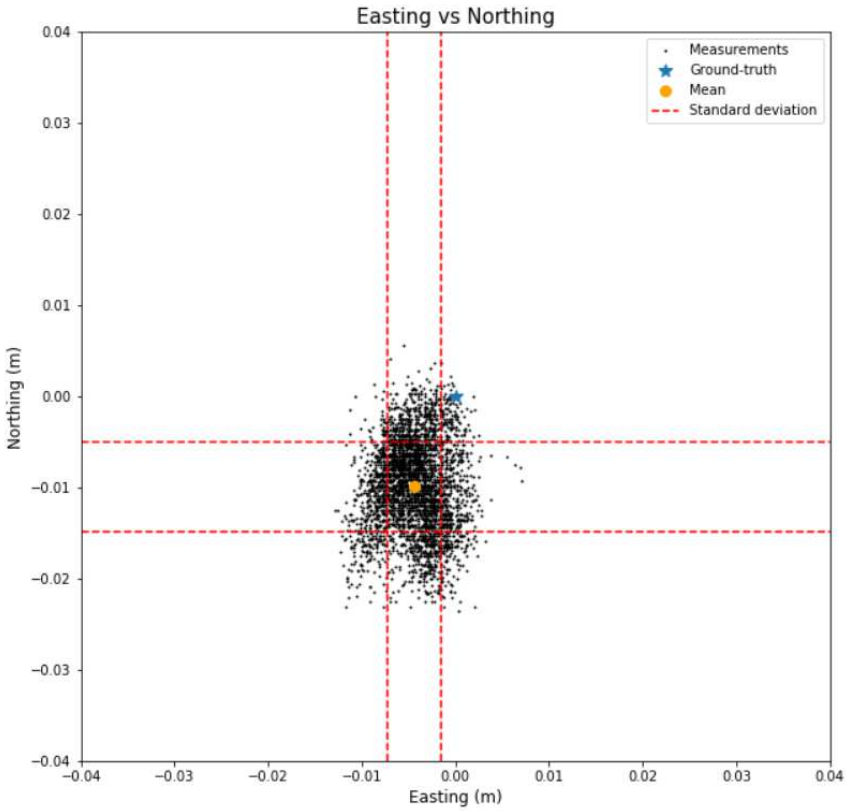

Figure F4: Scatter plot of benchmark 1 (SW Maps)

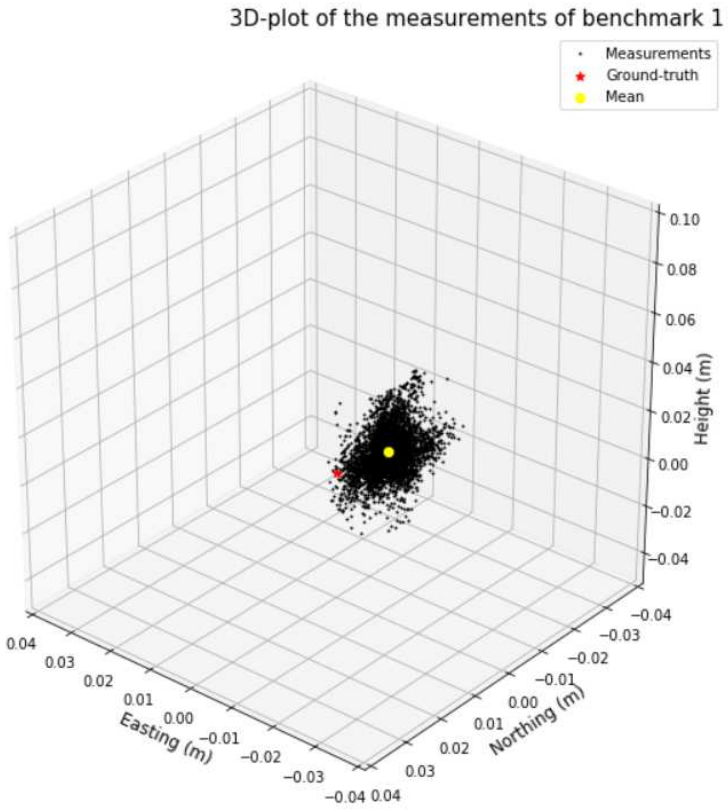

Figure F5: 3D-plot of the measurements of benchmark 1 (SW Maps)

Appendix G – RTKPOST analysis

Benchmark 1 with the DLF1 reference receiver

The fix ratio with this set-up is 47.3%. The measurements for which the carrier phase ambiguity could not be fixed, are left out of the results. The remaining fixed solutions are presented as a continuous data set.

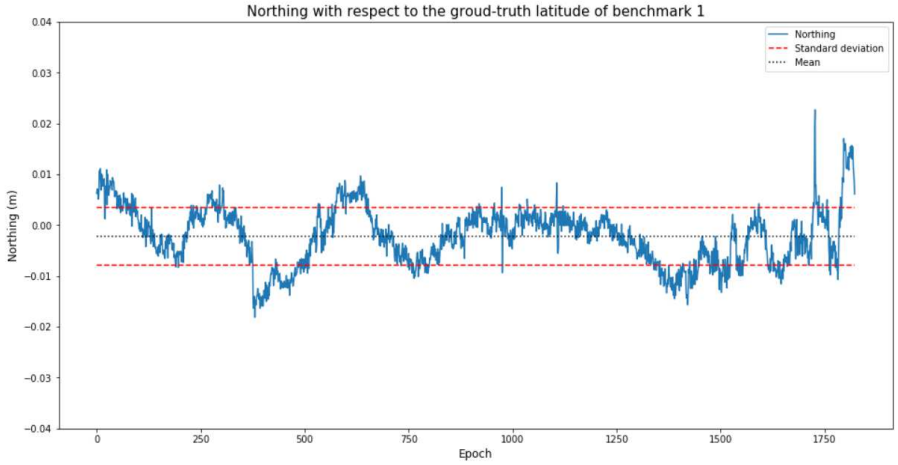

Figure G1: The Northing of benchmark 1 (RTKPOST – DLF1)

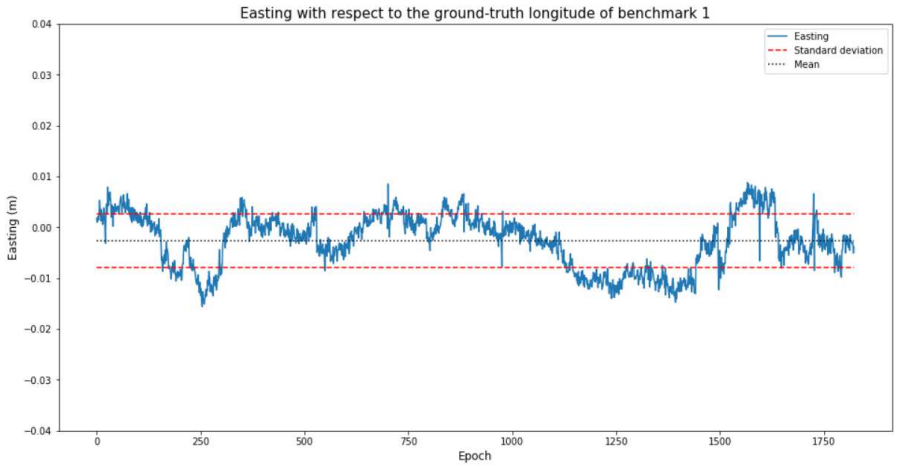

Figure G2: The Easting of benchmark 1 (RTKPOST – DLF1)

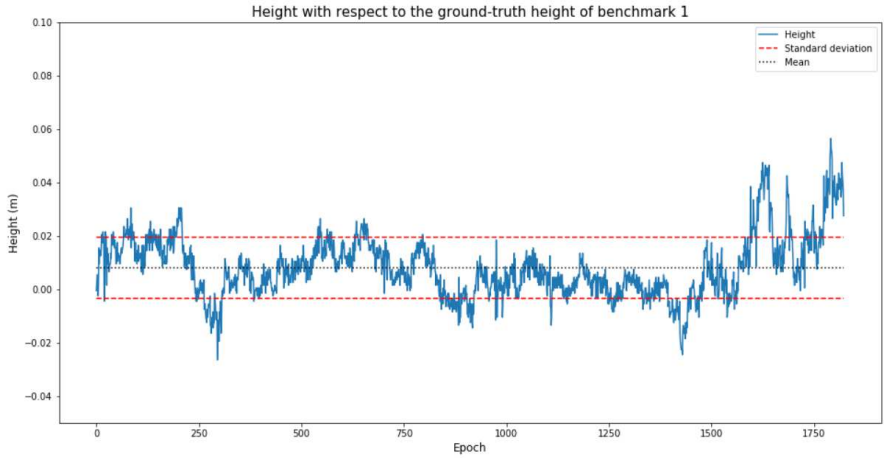

Figure G3: The height of benchmark 1 (RTKPOST – DLF1)

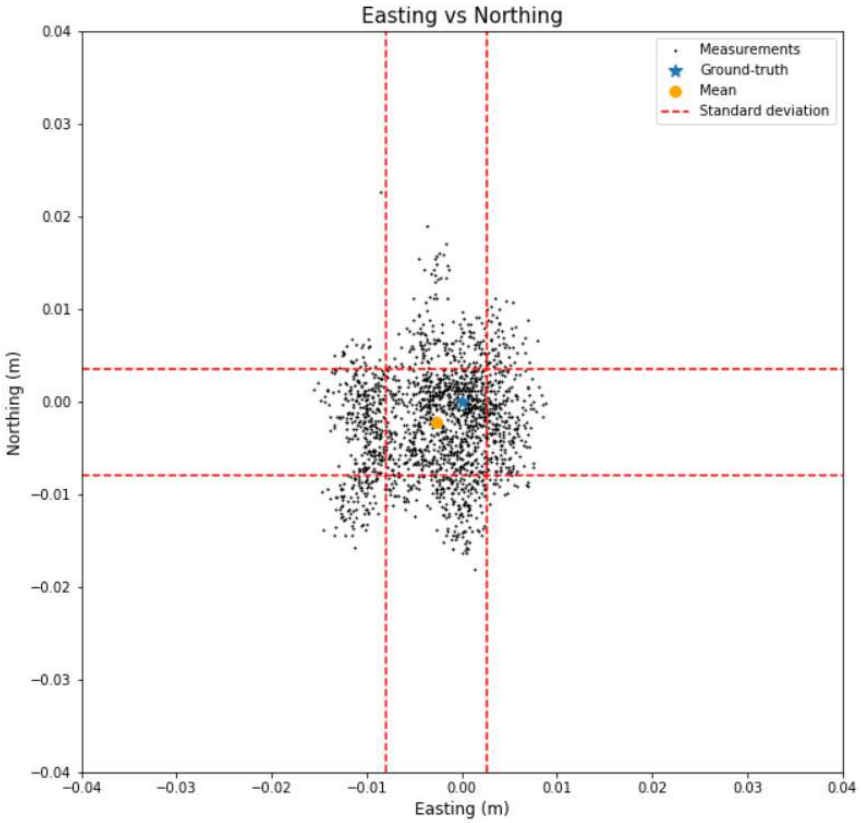

Figure G4: Scatter plot of benchmark 1 (RTKPOST – DLF1)

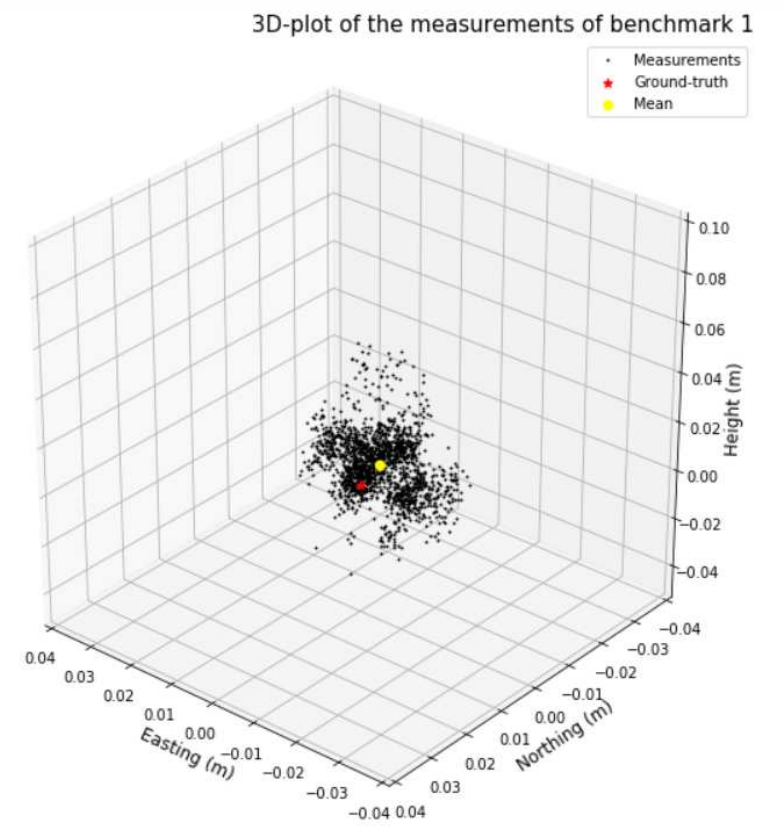

Figure G5: 3D-plot of the measurements of benchmark 1 (RTKPOST – DLF1)

Benchmark 1 with the u-blox reference receiver

The fix ratio with this set-up is 73.4%. Again, the measurements for which the carrier phase ambiguity could not be fixed, are left out of the results. The remaining fixed solutions are presented as a continuous data set.

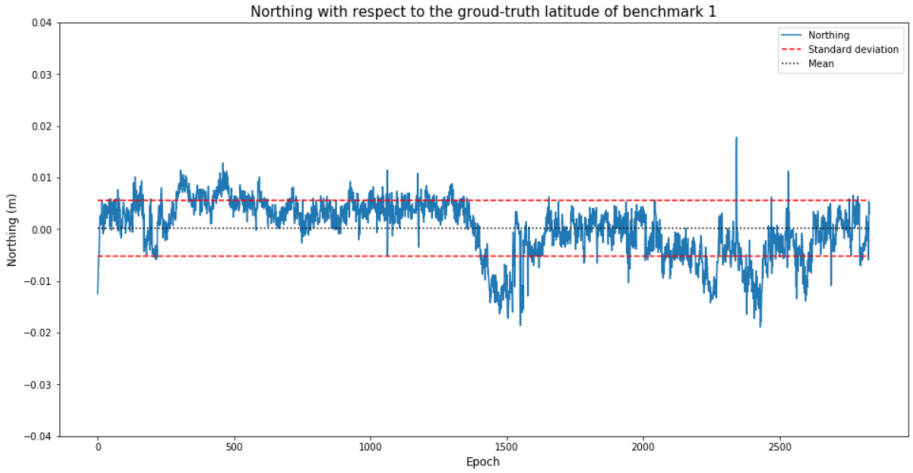

Figure G6: The Northing of benchmark 1 (RTKPOST – u-blox)

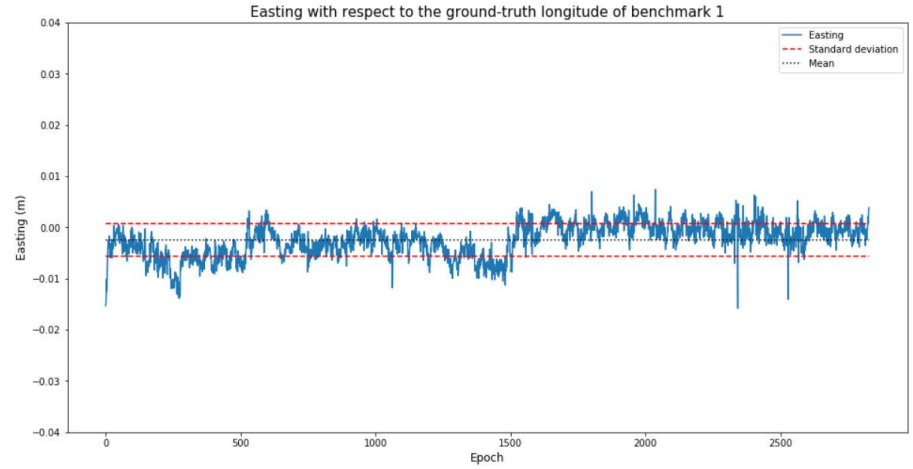

Figure G7: The Easting of benchmark 1 (RTKPOST – u-blox)

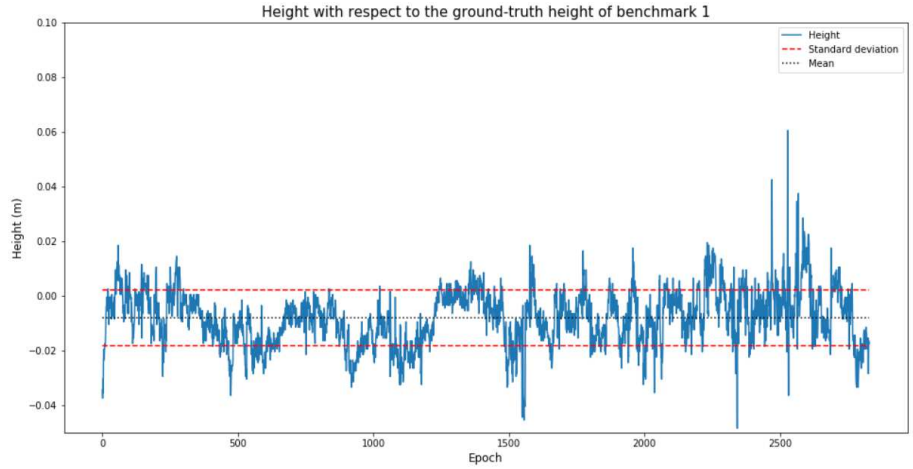

Figure G8: The height of benchmark 1 (RTKPOST – u-blox)

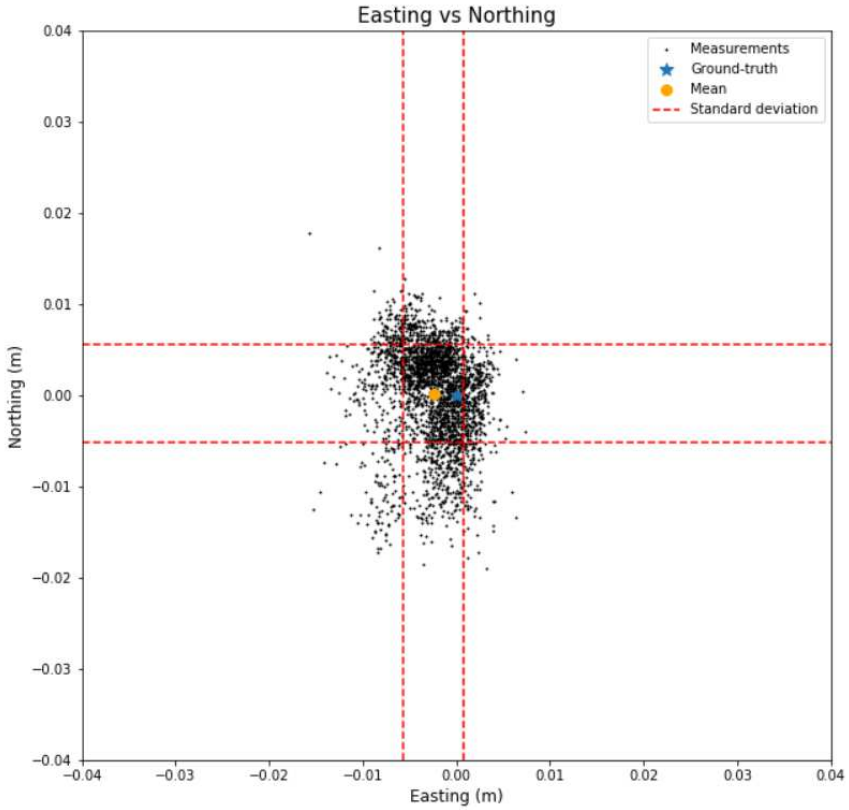

Figure G9: Scatter plot of benchmark 1 (RTKPOST – u-blox)

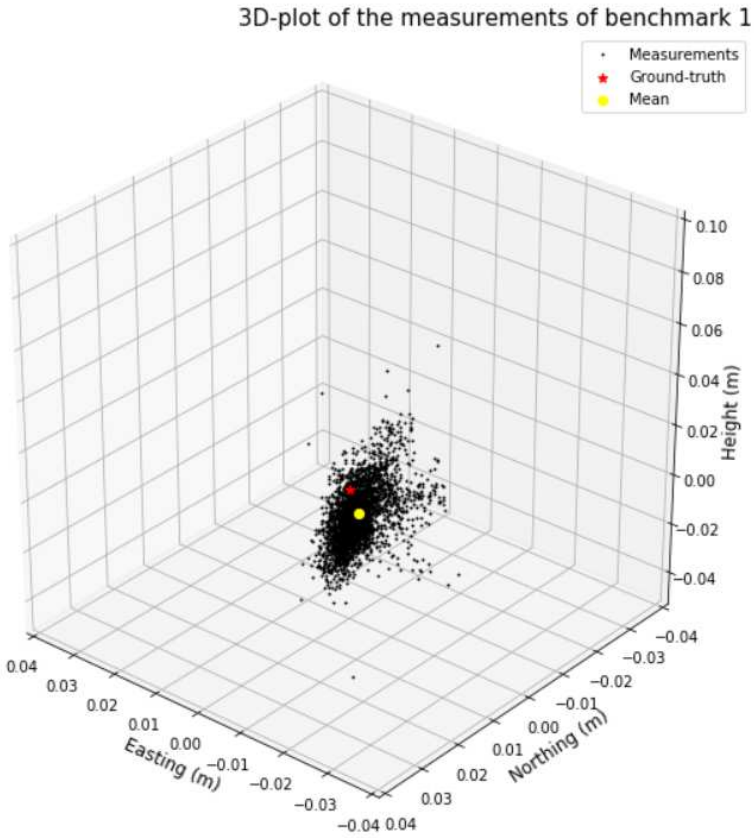

Figure G10: 3D-plot of the measurements of benchmark 1 (RTKPOST – u-blox)

Appendix H – Kinematic accuracy

Scatter plot with ground truth line

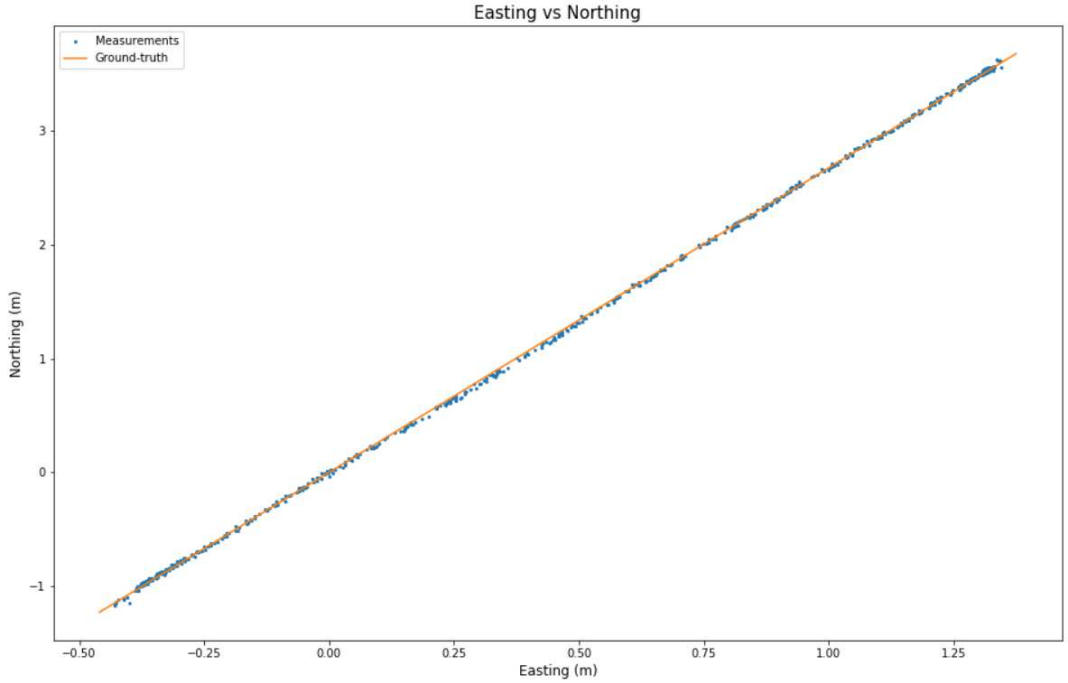

Figure H1: Scatter plot and ground truth line through the benchmarks

Time series

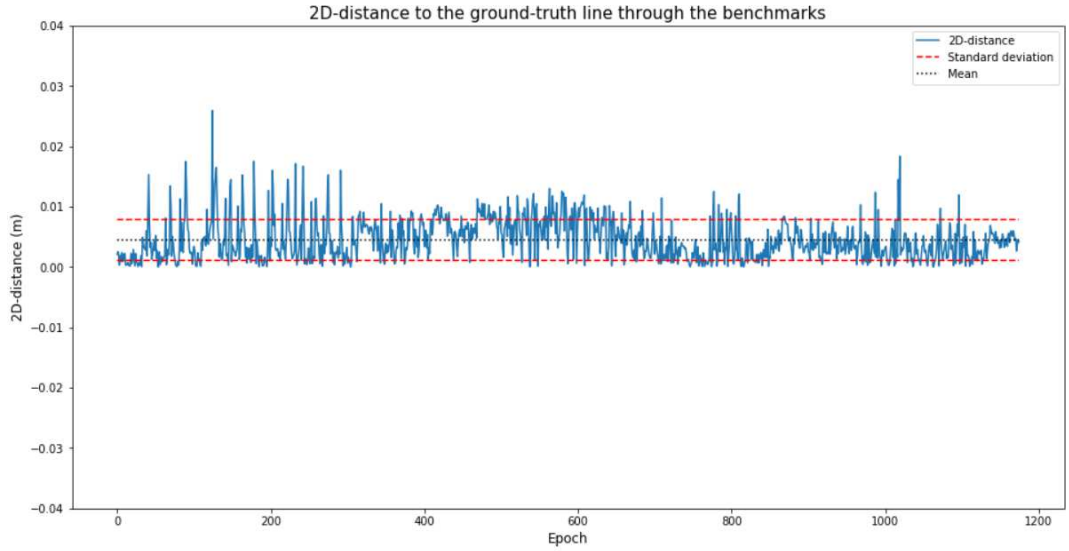

Figure H2: 2D-distance to the ground truth line

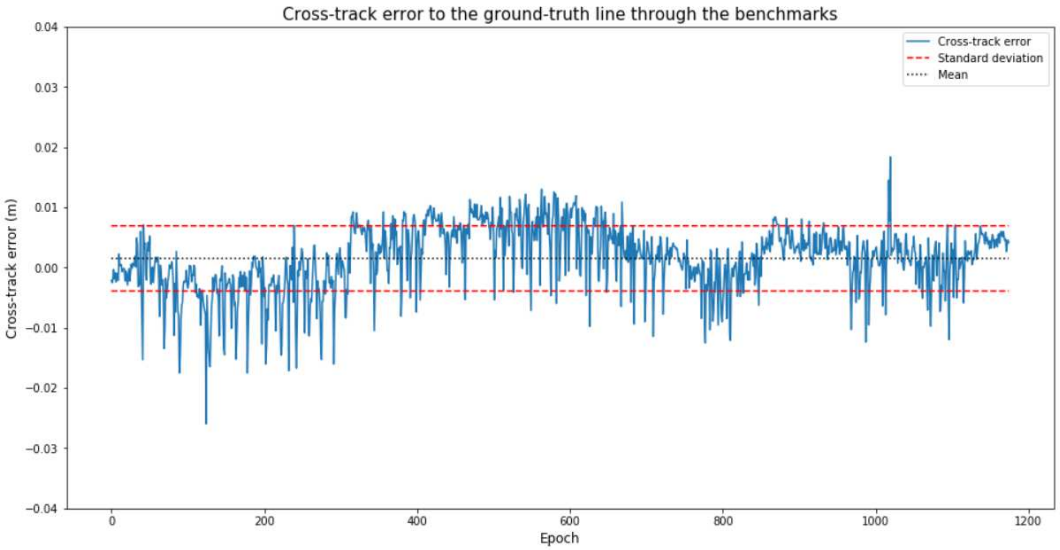

Figure H3: Cross-track error to the ground truth line

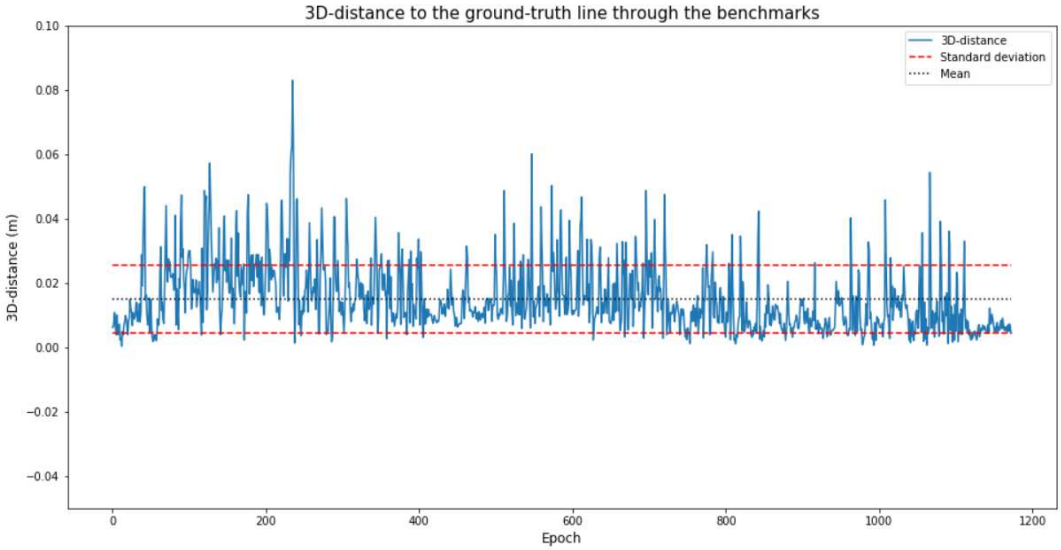

Figure H4: 3D-distance to the ground truth line

Appendix I – Influence of the surroundings

First of all, the time series of the solution status and the used number of satellites are plotted. As can be seen in the time series of the solution status, not all solutions are based on a fixed carrier phase ambiguity. The fix ratio is 63.35%. The time series of the Northing, the Easting and the height are plotted both for all measurements as well as just for the measurements with an RTK fix.

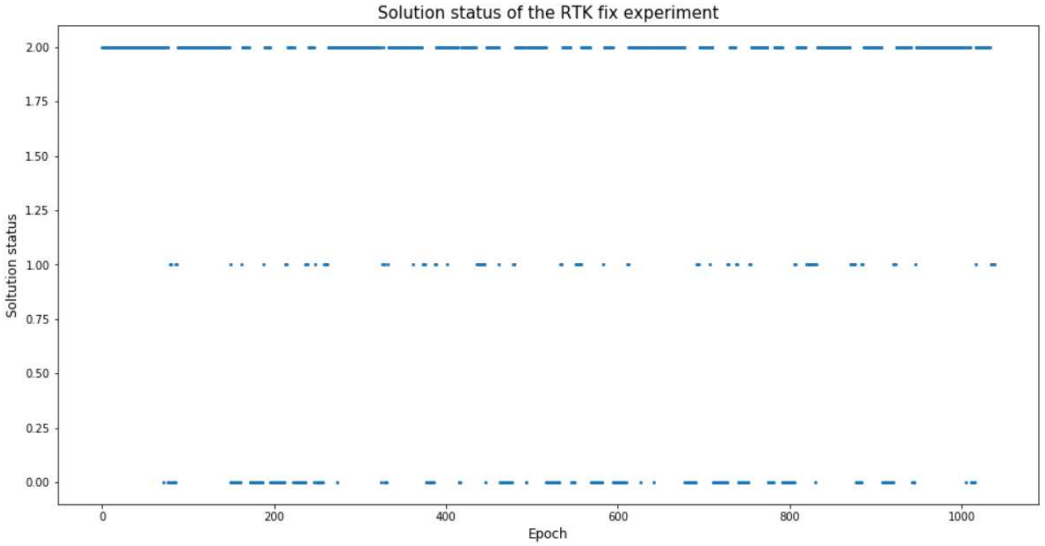

Figure I1: The solution status of the RTK fix experiment (2 = RTK fix, 1 = RTK float, 0 = no RTK solution)

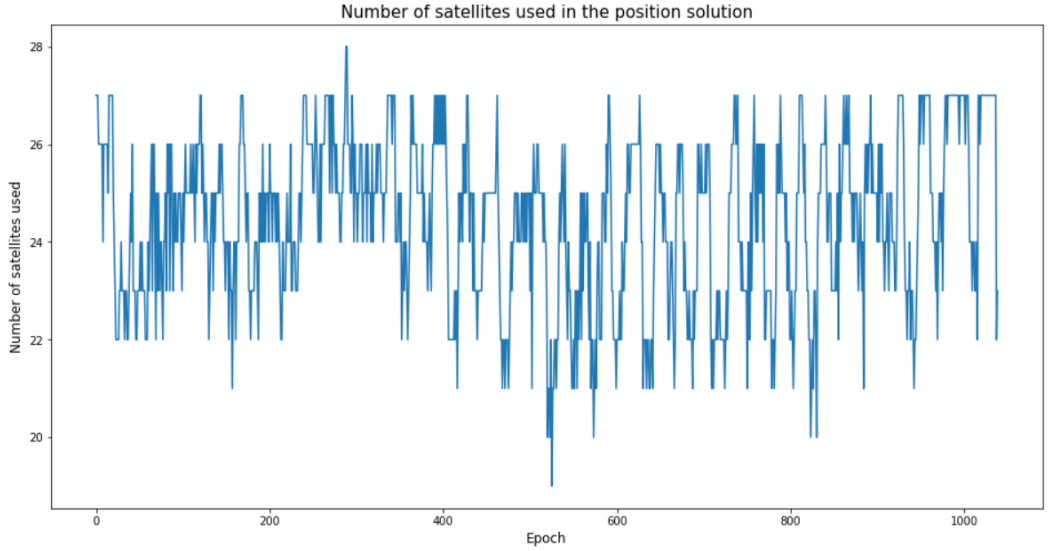

Figure I2: The number of satellites used in the position solutions of the RTK fix experiment

All measurements

One should note that the scales used in these plots differ from the scales of the other plots.

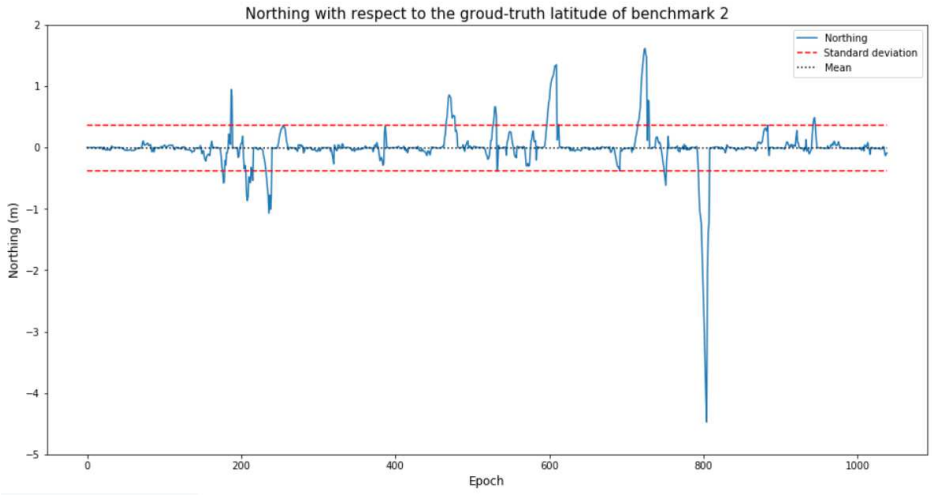

Figure I3: The Northing of benchmark 2 (RTK fix experiment – all measurements)

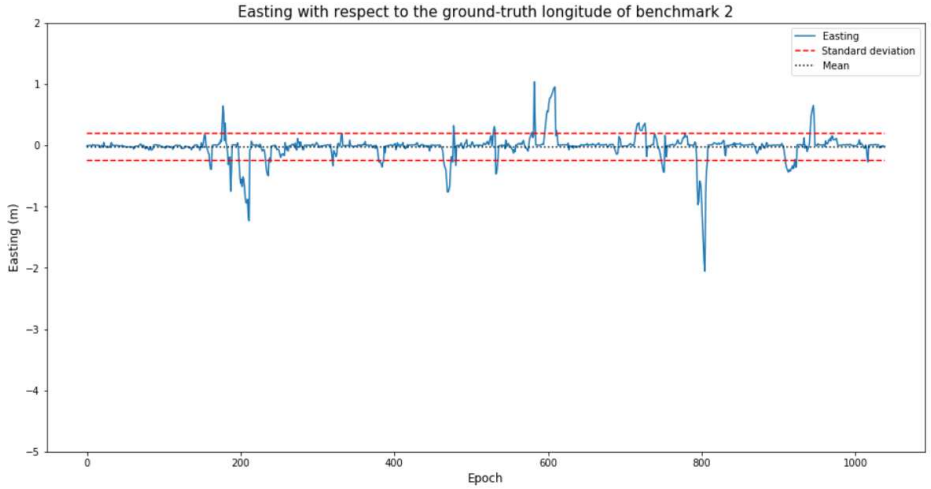

Figure I4: The Easting of benchmark 2 (RTK fix experiment – all measurements)

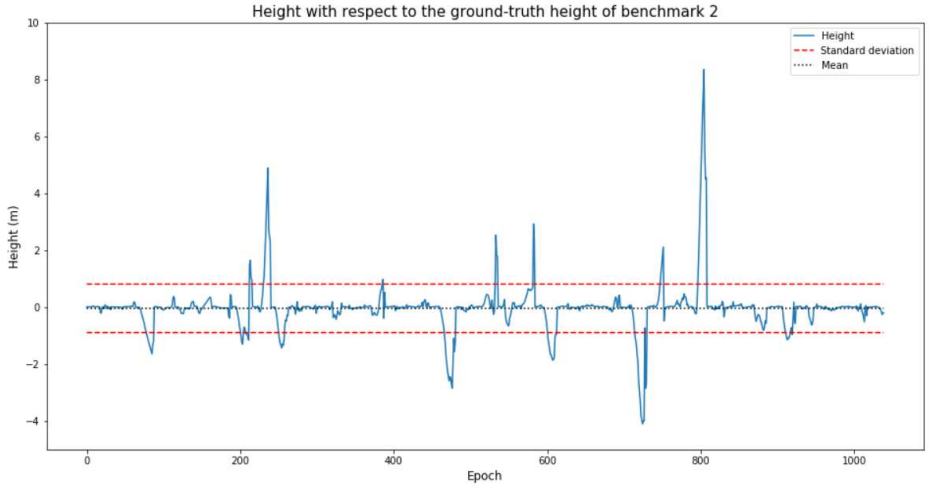

Figure I5: The height of benchmark 2 (RTK fix experiment – all measurements)

The measurements with an RTK fix

The scales of these plots also differ from the scales of the other plots.

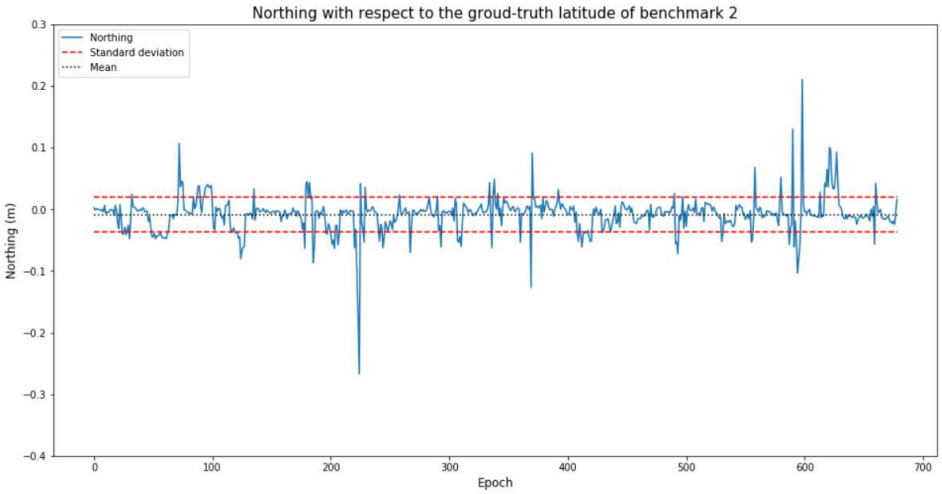

Figure I6: The Northing of benchmark 2 (RTK fix experiment – measurements with an RTK fix)

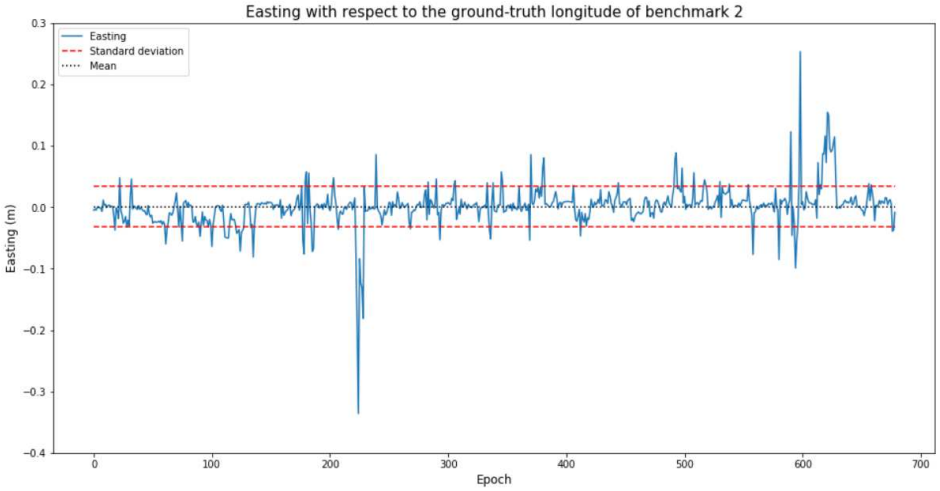

Figure I7: The Easting of benchmark 2 (RTK fix experiment – measurements with an RTK fix)

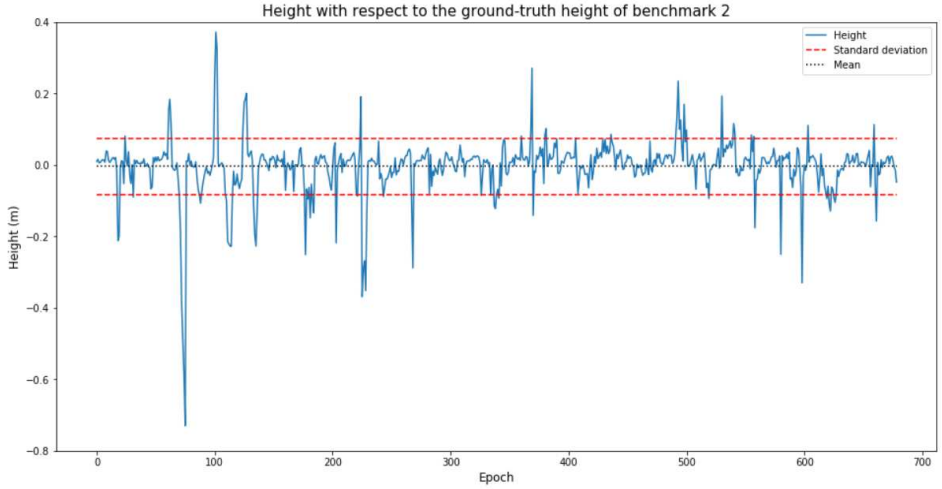

Figure I8: The height of benchmark 2 (RTK fix experiment – measurements with an RTK fix)