
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich


Semester Project

# Permasense GPS Postprocessing

ETH Zurich
Computer Engineering and Networks Laboratory

*Samuel Zumtaugwald*

supervised by
Dr. Jan Beutel
Matthias Meyer

Spring Term 2016

## Abstract

In order to compute the position of a GPS receiver, several different data is required. The data is put available for download by the IGS.
The aim of this project was to develop a software tool that computes the solution for a GPS receiver within the Permasense Network. The resulting program is a bash script called *compute_solution.sh* taking a minimum of two mandatory arguments. Firstly, a configuration file in which, amongst other parameters, the user defines the receiver and a base station from within the Permasense GPS network in the specific deployment ≪Dirruhorn≫. The second argument is the date for which the GPS position is desired.
The tool then automatically downloads and converts all required data and finally returns the GPS position in ECEF coordinates.
For the tasks of converting and computing, an external open-source program library, called rtklib, is used.

**Keywords** GPS Solution Calculation, Permasense, Rtklib

# Contents

# 1 Introduction

GPS positioning is part of everyone's daily life. Such position computation however is not trivial. Scientific workers in the Permasense project (see 2.1) analyze displacements of GPS receivers every day and therefore regularly need to compute positions of the latter.
Consequently it is beneficial to use a simple and quick program that allows to compute the position of a GPS receiver without having to manually collect all required data and perform calculations on it.
The solution is to use an existing open-source collection of tools doing the necessary calculations to obtain a GPS position. This tool collection is called *Rtklib* and is shortly described in section 2.4.
Even tough the tools provided are easy to use, the solution computation requires many data that must be supplied to the program by the user and therefore has to be downloaded first. This is a task that could be automated, which has been realized during this semester project: The development of a software tool that automates download, conversion and processing of the data needed in order to compute the position of a GPS receiver belonging to the Permasense Network.
The target audience to use the implemented tool are scientific coworkers at the TIK institute of ETH Zurich who quickly need to have a solution computed for a specific day and a specific GPS node within the network.
Such a tool was created, its main entrance point is a bash script called *compute_solution.sh* that will be described later in section 3.1. This tool is the first version and still leaves plenty of space for extension and improvement.

# 2 Project Overview

## 2.1 Embedded in the Permasense Project

The Permasense project is a scientific collaboration between several research institutes in Switzerland. Its research areas cover measurements of clima related geological effects in alpine regions.
One of the focuses lies on the rock movements due to thawing permafrost in the Matterhorn valley and other alpine regions in Switzerland. On each research site, some dozens of GPS sensor nodes are installed and organized in small wireless networks. Each network consists of about ten sensor nodes of which one serves as so called base station, while the other ones are called receivers or rover stations. All the raw GPS data collected by the rover stations and the base station itself are finally stored in a database on a remote server.
In this semester project the main focus lies on the so called ≪Dirruhorn≫ deployment in the Matterhorn valley. And more precisely on a subnetwork at

the «Grabengufer», a rocky place above the village of Randa where rock falls repeatedly occur due to climate changes.

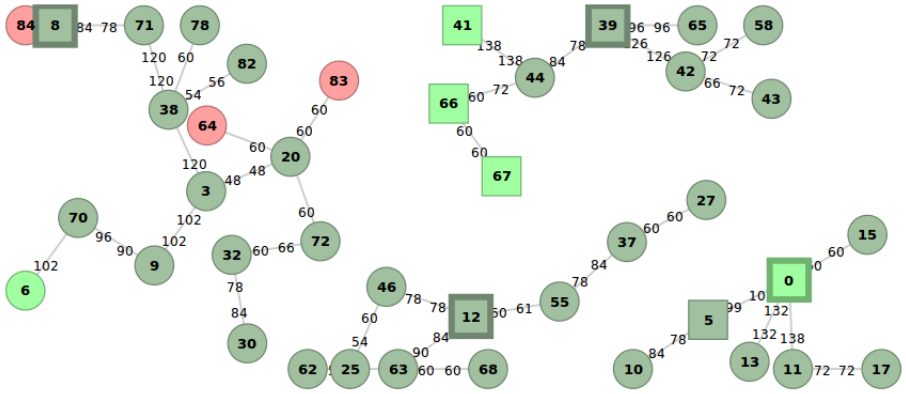


Figure 1: The topology of the wireless network at the «Dirruhorn» deployment. Not all nodes are GPS receivers (e.g. webcams). Position number 39 is the networks base station. [Screenshot from *http://data.permasense.ch/topology.html#topology*]

This semester project is strongly related to the Permasense project. Its aim is to develop a software tool that offers an easy way to calculate a receiver's position also called solution for a specific day.
As it will become clear, in order to calculate a solution for a GPS receiver not only the receiver's raw GPS data is required (see 2.2.2). The tool's task is to perform the final calculation but also to make sure that all required data is collected, downloaded and converted before the actual calculation is done.

## 2.2 GPS Position Calculation Data

The raw GPS data is stored in the database on the server *data.permasense.ch*. It contains information such as the travel time of the signals between the receiver and the satellites. Yet this is not enough to compute its position. In order to achieve this, further information is needed. Mainly the positions of the satellites but also additional clock information or atmospheric conditions a the time of the signal exchange.
The solution calculation in this project only takes into account the very essential data.

### 2.2.1 GPS Data

**Raw GPS Data** The above mentioned raw GPS data coming from the rover stations is stored following the ubx-format. Ubx-files aren't ASCII for-

mated and cannot be proceeded by the computation software in this project. Therefore, data stored in this format must be converted before it can be used.

**Observation Data** The raw GPS data is converted into the so called RINEX (receiver independent exchange) format using a software tool described later (see section 2.4). Once transformed, this data is called observation data.

### 2.2.2 IGS Data

The information about GPS satellite orbits as well as satellite and station clocks is collected by the International GNSS Service (IGS) and available for download on various servers.

**Satellite orbit data** Information about the GPS satellites' positions and clock record are collected in satellite orbit files according to RINEX format with the file extension *.sp3* (Standard Product # 3).
This data is essential for position calculation since the satellites of whose positions it consists are communicating with the rover stations. Of course not all satellites listed in the file are actually needed since many of them aren't visible to the receivers. One of many download sources for these orbit informations, a NASA managed website[1] was used in this project. The clock correction data (see below) can be downloaded from the same server.
For further information about this data format on can can consult the official documentation [2].

**Navigation Data** Another source of satellite orbits (ephemerides) is used in this work. This so called navigation data is available the day it is measured, unlike the *sp3* data which is more precise but is provided with a delay from approximately two weeks.
In this project, the data is downloaded from the Crustal Dynamics Data Information System (CDDIS) website[3].

**Clock Correction** Due to clock synchronism errors referring to GNSS time scale and relativistic effects due to the high velocity of the satellites the clocks running on satellite and receivers have to be synchronized. The required synchronization information is provided in clock correction files with file extension *.clk*. Again, more information about the data format is written in the official documentation[4].

[1] ftp://cddis.gsfc.nasa.gov/gnss/products
[2] https://igscb.jpl.nasa.gov/igscb/data/format/sp3_docu.txt
[3] ftp://cddis.gsfc.nasa.gov/gnss/data/daily
[4] https://igscb.jpl.nasa.gov/igscb/data/format/rinex_clock302.txt

**Differential Code Bias** In order to correctly use the clock correction data provided by the IGS, some more informmation is necessary. Since GPS sensor nodes are able receive signals on different frequencies. These signals can have different delays. In order to take into account this difference in delays, so called Differential Code Bias (*.DCB*) data is required. DCB files are published monthly and as source in this work a server[5] managed by the University of Berne was used.

**Antenna Correction** Absolute IGS phase center correction for receiver as well as satellite antenna is stored in an antenna correction file (*.atx* extension). In this project the file *igs08.atx* is used.

## 2.3 File Naming Convention

**File Naming** As one can see, for a solution calculation, many data has to be collected for that specific date. To assure that the correct data is downloaded, a strict naming convention is necessary. Else, the software would be looking for non existing files as well on the download servers as on the local machine. In the following table ( 1)the file names for the above mentioned data are explained.

| **Type of Data** | **Name of File** | **Explanation** |
|---|---|---|
| Raw GPS | *labelddd*0.ubx | label : name of receiver<br>ddd : day of year |
| Observation | *labelddd*0.yyO | label : name of receiver<br>ddd : day of year<br>yy : the year |
| Orbit | igs*wwwwd*.sp3 | wwww : GPS week<br>d : day of week (0-6) |
| Navigation | brdc*ddd*0.*yy*n | ddd : day of year<br>yy : the year |
| Clock Correction | igs*wwwwd*.clk | wwww : GPS week<br>d : day of week (0-6) |
| Differntial Code Bias | P1P2*yymm*.DCB | yy : year<br>mm : month |
| Solution | *label*00CHE*yyyymmdd*.pos | label : name of the receiver<br>yyyy : year<br>mm : month<br>dd : day |

Table 1: The naming convention used in this project.

[5] ftp://ftp.unibe.ch/aiub/CODE

## 2.4 RTKLIB

Rtklib is an open source library containing several CUI and GUI applications for various operations on different GNSS data types. In this project, the following two command line tools were used [6]:

**Convbin** This is a tool converting raw GPS data from a receiver into rinex conform observation data.

**Rnx2rtkp** This tool is used in the last part of the *compute_solution.sh* script. Namely in the actual solution computation for the required position. Many additional parameters can be given in order to have more accurate output.

[6] For more information: www.rtklib.com

# 3 Results

## 3.1 The Scripts

**compute_solution.sh** The main script to call in order to have a a solution computed is the so called *compute_solution.sh*. As arguments it takes a ≪-p≫ followed the path to a required parameter file (see section 3.2) and a the date for which the solution shall be calculated. All further information, such as the receiver for which the solution is desired, the base station against which the receiver's position is computed, the servers from which the necessary data has to be downloaded and several directories indicating where to store the created data and output must be given in the parameter file.

Once the script is called, it first creates a temporary folder at location *../* relative to the directory in which it is launched. The GPS and IGS files are downloaded into this temporary folder and and also the logfile for the entire script execution is written here. Next, the existence of the given directories is verified. If all indicated directories are found, the script proceeds to download the required data.
If still no errors have occurred, the conversion of the GPS raw data is done, using the *convbin* tool mentioned above. If the conversion worked well, the script finally executes the actual solution computation calling the *rnx2rtkp* tool to which the downloaded IGS data are passed as command line arguments. The resulting solution as well as the observation and IGS data are then stored in their respective directories (indicated in the parameter file) and the temporary directory is deleted.
Besides the mandatory two, the user can set the following options:

-b: This option set, the script will omit all download and conversion concerning the base station. This means that the observation data for the base station must be stored in the directory
*gps_data_dir/basestation_deployment/basestation_label*

-r: This option set, the script will omit all download and conversion concerning the rover station.This means that the observation data for the rover station must be stored in the directory
*gps_data_dir/roverstation_deployment/roverstation_label*

-d : This means that the required IGS data is already stored locally and its download is therefore omitted. In case the needed data is not found where it should be stored (*igs_data_dir* in the parameter file) the script stops running.

-c: If this option is set, the conversion of the raw GPS data is skipped. Obviously this presumes that the observation data for the rover station as well as for the base station must already be stored in the correct directory (*gps_data_dir*) in the parameter file. Like for the ≪-d≫ option, if the data is not found, the script stops its execution.
A flowchart of the script is shown in figure 3.

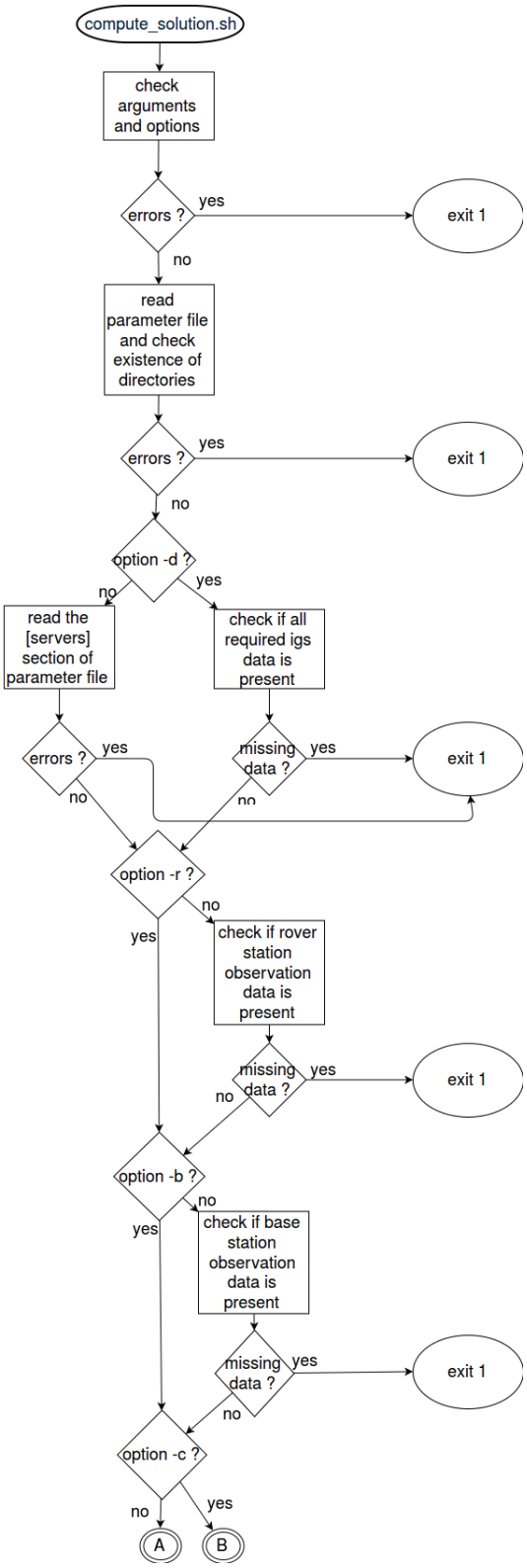


Figure 2: The execution flow of the main script *compute_solution.sh* 1/2

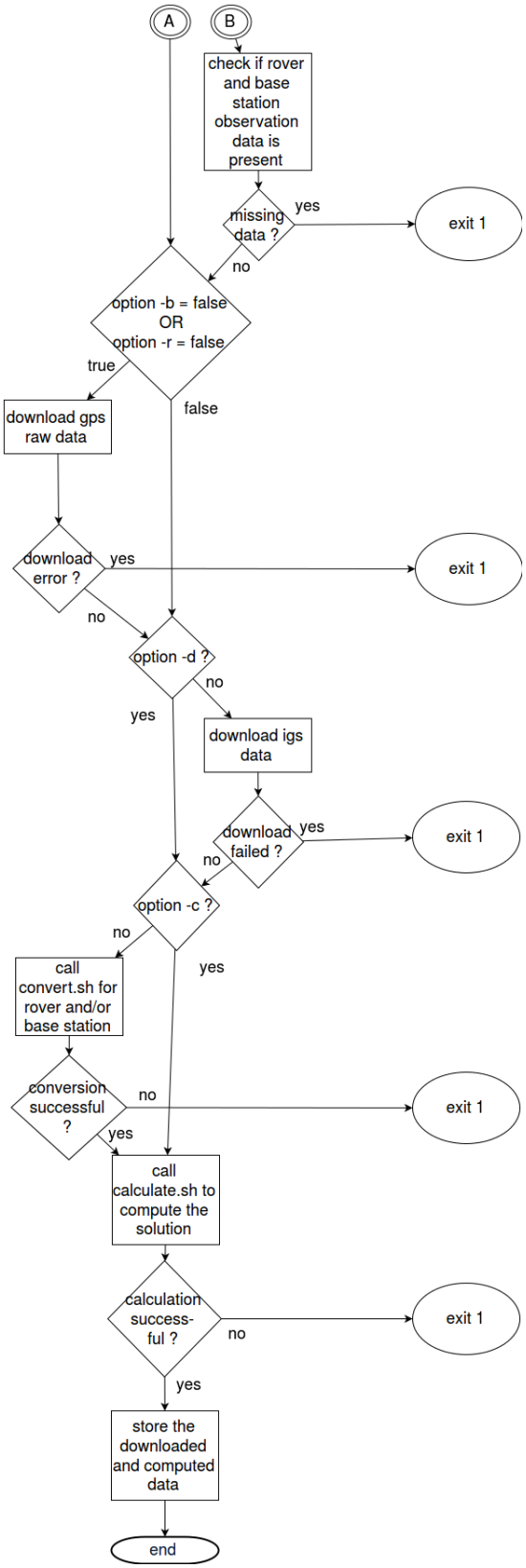


Figure 3: The execution flow of the main script *compute_solution.sh* 2/2

**convert.sh** This script is called from inside the *compute_solution.sh* and is not designed to be executed alone. It merely checks if the tool *convbin* is where it should be (in the *rtklib_dir*) and then executes it. This tool takes only one argument, namely the raw GPS data file to be converted. The call of the convbin tool in the *compute_solution.sh* is as follows:

```
$rtklib_dir'/convbin' -d $temp_dir -o $rover_station_obs_file
    $temp_dir'/'$rover_station_gps_file
```

This command calls the *convbin* tool (stored in *rtklib_dir*) and stores the observation file in the temporary folder (*temp_dir*) under tha name assigned to the variable *rover_station_obs_file*.
The raw data as well as the converted observation file are stored in the temporary folder (from where they will be removed at the end of the compute_solution.sh execution). It also accepts options ≪-b≫ and ≪-r≫ omitting the conversion of gps raw data for the base or the rover station respectively.

**calculate.sh** Another bash script to be only called from within the *compute_solution.sh* script, since it also needs the paths to several directories defined in it. Very much like the *convert.sh* script, this one mainly checks if the required tool *rnx2rtkp* is found and then executes it. In contrast to the *convbin* tool, *rnx2rtkp* needs to be passed several arguments:

```
$rtklib_dir'/rnx2rtkp -x 0 -k '$rtklib_options_dir'/'
    $rnx2rtkp_config_file' -o '$out_dir'/'$solution_file' -p 3 -c
    -e -r '$ref_pos_x' '$ref_pos_y' '$ref_pos_z' '$temp_dir'/'
    $rover_station_obs_file' '$temp_dir'/'$base_station_obs_file'
    '$temp_dir'/'$nav_file' '$temp_dir'/'$sp3_file' '$temp_dir'/'
    $clk_file' '
```

This command as it appears in the *calculate.sh* script takes several arguments. The most important ones are quite obvious though and the reader assumingly can easily guess whats assigned to a variable called *ref_pos_x* for example. It can be seen that for the solution computation the observation data of the rover station (*rover_station_obs_file*) as well as of the base station(*base_station_obs_file*) the navigation (*rover_station_obs_file*) orbit (*sp3_file*) and clock correction data (*clk_file*) are required. Furthermore the position of the reference (base) station is passed to the tool (*ref_pos_x* , *ref_pos_y* ,*ref_pos_z*).
Another very important argument is the *rnx2rtkp*'s own configuration file (*rnx2rtkp_config_file*). Herein many optional files (e.g. atmospheric data) can be inserted into the calculation, leading to more accurate solutions. The *-k* option indicates that input option from the configuation file are taken into account.

Furthermore, some small python scripts are called from inside those three bash scripts. They have to be stored in the same folder as them. Namely the required python scripts are:

**ConfigReader.py** This script is called at the beginning of *compute_solution.sh*. And as the name indicates, it is responsible to read a configuration file, i.e. the parameter file given as argument to *compute_solution.sh*.

**DateConverter.py** Since GPS time and the Gregorian calendar time are not quite the same dates have to be converted from the classic year/month/day format to GPS time. Moreover, to name the different files according to the given convetion (see section 1) it is required to know what day of the year or what day of the week this is. The *DateConverter.py* script is a collection of such functions calculating the GPS Week, day of week or day of year for classic format dates.

**download_gps_data.py and download_igs_data.py** The name actually says everything for those two scripts. The former downloads raw GPS data for a certain receiver from the Permasense database (data.permasense.ch) and the latter downloads the required IGS data for a given date from the servers indicated in the parameter file.

**get_file_name.py** This is a collection of function returning the date-depending file paths or file names.

**GetUbxFileName.py** A script returning the correct name for a file containing the raw GPS data for a certain position (label, not number) and a certain date.

**GetObsFileName.py** A script returning the correct name for a file containing observation data for a certain position (label, not number) and a certain date.

**LogFileWriter.py** A function called many times in any of the three bash scripts. It takes two arguments. The first one is the path to the logfile in which it has to write the log entry that is passed as string in the second argument.

**update_file.py** The *rnx2rtkp* tool requires its own configuration file (given in the parameter file). Some of the files assigned to variables in this file however are date-depending. In the most basic case treated in this project, only one file name has to be updated. Namely the *file-dcbfile*. So if one desires to call the *compute_solution.sh* several times with the same parameter file without manually changing this entry every time, this function is responsible to update the path to the date-depending *file-dcbfile*. The file itself is downloaded together with the other IGS data.

## 3.2 The Parameter File

As mentioned in section 3.1 the main entrance script *compute_solution.sh* takes two mandatory arguments. One is the date for which the solution is desired and the other one is the path to a parameter file. It is very important that this parameter file is completed correctly.
The file itself is divided into several sections that are all mandatory and shortly described in the following paragraphs:

**Positions** The section [positions] lets the user decide for which rover station and with respect to which base station he/she wants to have a solution. The following table ( 2) shows the key-value pairs to be filled in in the *Positions* section.

| **Key** | **Description** |
|---|---|
| roverstation_deployment | The Name of the deployment where the roverstation is |
| rover_station_nr | The number assigned to this rover station |
| rover_station_label | The name assigned to this rover station |
| basestation_deployment | The Name of the deployment where the basestation is |
| base_station_nr | The number assigned to this base station |
| base_station_label | The name assigned to this base station |
| ref_pos_x<br>ref_pos_x<br>ref_pos_x | The position of the basestation. In ECEF coordinates relative to the reference position |

Table 2: The keys in the [positions] section of the parameter file.

**Servers** In this section the user can define from where the required IGS data will be downloaded.
Very important to note is that not all servers use the same directory structure and this tool is designed in order to work well with the servers in the sample parameter file (see appendix B). It is not guaranteed that the tool downloads correctly from other servers, since it automatically navigates through the subdirectories that might be date-depending.
For example, if the given server to download the clock correction data file is *ftp://cddis.gsfc.nasa.gov/gnss/products* and the date for which the data is needed is $26^{th}$ of January 2015, the actual location of the required file is *ftp://cddis.gsfc.nasa.gov/gnss/products/**1829**/igs18291.clk.Z* where the subdirectory *1829* is due to the fact that this date is in the GPS week 1829.
Overall three servers have to be indicated by the user as shows table 3

| **Key** | **Description** |
|---|---|
| dataserver_1 | The server from which *sp3* and *clk* files are downloaded |
| dataserver_2 | The server from which navigation file is downloaded |
| dataserver_3 | The server from which DCB file is downloaded |

Table 3: The keys in the [servers] section of the parameter file.

**Directories** In this section the directories in which the downloaded data as well as the results are stored after the solution calculation.
It is crucial for the correct execution of the script, that all of the given directories really exist.
This section's keys are descibed in table 4

| **Key** | **Description** |
|---|---|
| gps_data_dir | The directory in which the observation data is stored |
| igs_data_dir | The directory in which the IGS provided data is stored |
| output_dir | The directory into which the resulting solution and the logfile are stored at the end of the scripts execution |
| rtklib_dir | The directoy in which the *convbin* and the *rnx2rtkp* tools must be |
| rtklib_options_dir | The directory in which the configuration file for the *rnx2rtkp* tool must be stored |

Table 4: The keys in the [directories] section of the parameter file.

Another thing that must be kept in mind and verified before launching the script is that the GPS data directory must have the following subdirectory structure:

gps_data_dir
    /roverstation_deployment
        /rover_station_label
    /basestation_deployment
        /base_station_label

This is necessary because in case the program is called with options set, then some data download is omitted. In that case, the program will look for the data in the directories given in this section. The reason for this directory substructure is that the program is adapted to a database on a server at the TIK institute that uses this structure.

**Files** In this section only one parameter must be assigned. This is the name of the configuration file the *rnx2rtkp* has to use.

If one of the keys is not found or has no value assigned to it, the *compute_solution.sh* stops its execution. A sample parameter-file is included is the appendix B.

**The Logfile** During the execution of the program, a logfile is written in the temporary folder created at the beginning. It is named the following way: *log_yyyymmdd.txt* where *yyyymmdd* indicates the year, the month and the day for which the solution is calculated.
Furthermore, the output created by *convbin* and *rnx2rtkp* during their respective execution is written into their own logfiles. These are called *log_convbin_bs.txt*, *log_convbin_rs.txt* and *log_rnx2rtkp.txt* respectively. The *rs* and the *bs* in the *convbin*-logfile names stand for rover- and base station. Once the script terminates, the logfile is stored together with the solution in the output directory. The *log_convbin_rs.txt*, *log_convbin_bs.txt* and *log_rnx2rtkp.txt* are deleted in case of successful solution calculation, since their main purpose is that in case an error occurs while executing one of those tools the user can read what happened.

# 4 Usage ot the Solution Calculation Tool

## 4.1 Pre Requirements

Before one can run the program (i.e. call the *compute_solution.sh* script) to compute a GPS solution for a specific receiver within the Permasense Network several prerequisites have to be met.

The **parameter file** (see 3.2) has to be completed. All given directories in it must exist before the launch of the script. The servers from which the IGS data shall be downloaded must be specified (for example the ones in the sample file in appendix B. As mentioned in section 3.2 it must be taken care of the fact that the subdirectories in which the data lies on the server might be date-depending and therefore the path completion performed by this program might not be suitable for the specific servers. In this case, one would have to adjust the functions *get_clk_folderName* and *get_sp3_folderName* in the *get_file_name.py* script so they would complete the paths correctly.
The **Rtklib** tools *convbin rnx2rtkp* tools must be stored in the given directory and the configuration file used by the latter must be supplied the necessary filepath to the antenna correction file *igs08.atx*.
For more information about the possible options to be specified in that file, the reader might consult the official manual at *http://www.rtklib.com/prog/manual_2.4.2.pdf* .
Once these requirements are fullfilled the rest is not very complicated. The possible command line options are explained in section 3.1 and the rest is presented in the following sections with the aid of examples.

## 4.2 Examples

**Basic** In case the user doesn't yet have any data but wants to compute a solution for the sample day yyyy mm dd he/she has to do the following prompt:

```
$ bash compute_solution.sh -p [path-to-parameter-file] yyyy mm dd
```

This is done in the following listing for February $13^{th}$ 2014:

Listing 1: Output of basic usage of the program

```
1  $ bash compute_solution.sh -p /home/samuel/semester_project/tool/
       config/parameter_file.txt 2014 2 13
2  opt_bs: false
3  opt_rs: false
4  opt_dl: false
5  opt_cv: false
6  Year Month Day: 2014 02 13
7  Created temporary folder /home/samuel/semester_project/tool/
       scripts/../temp_20140213
8  Load configuration ...
9  Retrieving the file paths from the [directories] section within
       the configuration file
10 Retrieving the positioning data from the [positions] section
       within the configuration file
11 Retrieving the file adresses from the [servers] section within the
        configuration file
12 End of configuration reading.
13
14 Gps raw data download...
15 Igs data download...
16 Call convert script
17 Close convert script
18 Call calculate script
19 The calculate script was called for rover GG01 (position 43) and
       base station RG01 (position 39) for day 2014 02 13
20 /home/samuel/rtklib_2.4.2/cuis/rnx2rtkp -x 0 -k /home/samuel/
       rtklib_2.4.2/conf/rtkpost_dirruhorn_static_v21.conf -o /home/
       samuel/semester_project/results/GG0100CHE_20140213.pos -p 3 -c
        -e -r 4392040.8940 602680.4440 4574388.7730 /home/samuel/
       semester_project/tool/scripts/../temp_20140213/GG010440.14O /
       home/samuel/semester_project/tool/scripts/../temp_20140213/
       RG010440.14O /home/samuel/semester_project/tool/scripts/../
       temp_20140213/brdc0440.14n /home/samuel/semester_project/tool/
       scripts/../temp_20140213/igs17794.sp3 /home/samuel/
       semester_project/tool/scripts/../temp_20140213/igs17794.clk
21 Solution calculation was successful.
```

```
22 The solution for rover GG01 (position 43) as well as the logfile (
       log_20140213.txt) are now located at /home/samuel/
       semester_project/results
```

In line 1 the program is called. In lines 2 until 12 the parameter file (at `/home/samuel/semester_project/tool/config/parameter_file.txt` ) is read. While in line 7 the ourput informs the user where to find the temporary directory. This is of importance in case any error leading to execution abortion occurs. All data that was downloaded and created until then are stored in that directory. However, here no errors occured.
Lines 14 and 15 indicate that the scripts responsible for the data download are executed. Once the required data is downloaded, the raw GPS data is converted into observation data (line 16-17) before the most important part is done when computing the solution by the command written in line 20. A long command, since it contains all files used with their absolute path.
In the end (line 22), the solution (*GG0100CHE_20140213.pos*) and the logfile (*log_20140213.txt*) are moved into the output directory that was defined in the parameter file.
Both, the *GG0100CHE_20140213.pos* and the *log_20140213.txt* as well as the used *parameter_file.txt* are completely shown in appendix A.1.
Nevertheless, the solution file shall be presented shortly in here.

```
% program   : RTKLIB ver.2.4.2
% inp file  : /home/samuel/semester_project/tool/scripts/../temp_20140213/GG010440.14O
% inp file  : /home/samuel/semester_project/tool/scripts/../temp_20140213/RG010440.14O
% inp file  : /home/samuel/semester_project/tool/scripts/../temp_20140213/brdc0440.14n
% inp file  : /home/samuel/semester_project/tool/scripts/../temp_20140213/igs17794.sp3
% inp file  : /home/samuel/semester_project/tool/scripts/../temp_20140213/igs17794.clk
% obs start : 2014/02/13 00:01:00.0 GPST (week1779 345660.0s)
% obs end   : 2014/02/14 00:00:00.0 GPST (week1779 432000.0s)
% pos mode  : static
% freqs     : L1+L2+L5
% solution  : combined
% elev mask : 10.0 deg
% dynamics  : off
% tidecorr  : on
% ionos opt : estimation
% tropo opt : est ztd
% ephemeris : precise
% amb res   : continuous
% val thres : 3.0
% antenna1  :                     ( 0.0000  0.0000  0.0000)
% antenna2  :                     ( 0.0000  0.0000  0.0000)
% ref pos   :  4392040.8940   602680.4440  4574388.7730
%
% (x/y/z-ecef=WGS84,Q=1:fix,2:float,3:sbas,4:dgps,5:single,6:ppp,ns=# of satellites)
%  GPST                  x-ecef(m)      y-ecef(m)      z-ecef(m)   Q  ns   sdx(m)   sdy(m)   sdz(m)  sdxy(m)  sdyz(m)  sdzx(m) age(s)  ratio
2014/02/13 00:10:30.000   4391926.3790    602616.3559   4574411.9242   1   4   0.0011   0.0009   0.0010  -0.0002   0.0001   0.0010   0.00   82.3
```

Figure 4: The solution file created in this example

The meaning of the header lines (beginning with ≪%≫) are explained in the following table 5.

| **Key** | **Description** |
|---|---|
| program | The program used to compute the solution |
| inp file | The additional igs data used for the calculation |
| obs start | Beginning of the measurements in this solution |
| obs end | End of the measurements in this solution |
| pos mode | The position mode used. Static promises the most accurate position |
| freqs | The frequency bands used for signal transmission between satellite and receiver |
| solution | The Solution type option. Combined for higher numerical stability |
| elev mask | Satellites below this angle are neglected for the data collection |
| dynamics | Computes the position taking into accout velocity and acceleration Not possible for static mode |
| tidecorr | Earth tidal displacementt taken into account |
| ionos opt | Indicates how the ionospherical condition parameters (e.g. total electron content) is obtained |
| tropo opt | The ZTD (Zenith Tropospheric Delay) i.e. the influence of the troposphere's composition on the signal travel time is estimated from the GPS data |
| ephemeris | Indicates what kind of ephemeris data is used (*.sp3*) |
| amb res | Integer ambiguity resolution option |
| val thres | Integer ambiguity validation option |
| ref pos | Position of the antenna of the base station |

Table 5: Description of the solution file header

These first header lines in table 5 are relevant for the precision of the computed position written in the last line and described in table 6.

| **Key** | **Description** |
|---|---|
| GPST | The calendar timefor which this position is calculated |
| x-ecef(m)<br>y-ecef(m)<br>z-ecef(m) | The x, y and z components (in meters) of the rover station's position Computed according to the positioning options |
| Q | Flag indicating the solution's quality |
| ns | The number of valid (visible) satellites used for this solution estimation |
| sdx(m)<br>sdy(m)<br>sdz(m)<br>-sdx(m)<br>-sdy(m)<br>-sdz(m) | Standard deviation (in meters) of the different position's components |
| ratio | The ratio factor for the integer ambiguity validation |

Table 6: Description of the solution file body

For more precise information about this output format, the reader is invited to consult official documentation or the rtklib manual[7].

Comparing the first and last line of the logfile, one can see that this solution calculation took three minutes. This is of course too long if several dates shall be computed together (as in the next example). For this purpose, the user wight want to distribute the computation over multiple cores.

**Several days** If one is interested in seeing how a single receiver moves from day to day, he/she simply has to call the main script in a loop containing the dates in question.
The following call, for example, computes a solution for every day in February 2014:

```
1 for d in {1..28} ; do bash compute_solution.sh -p /home/samuel/
    semester_project/tool/config/parameter_file.txt 2014 2 $d ;
    done
```

After this is executed, the output directory (e.g. `results`) looks as follows:

Listing 2: Content of the output directory after the script was executed for the whole February 2014

```
~/semester_project/results$ ls -l
insgesamt 336
-rw-rw-r-- 1 samuel samuel 1384 Jun 30 20:21 GG0100CHE_20140201.
    pos
-rw-rw-r-- 1 samuel samuel 1384 Jun 30 20:25 GG0100CHE_20140202.
    pos
-rw-rw-r-- 1 samuel samuel 1384 Jun 30 20:28 GG0100CHE_20140203.
    pos
-rw-rw-r-- 1 samuel samuel 1384 Jun 30 20:32 GG0100CHE_20140204.
    pos
-rw-rw-r-- 1 samuel samuel 1384 Jun 30 20:36 GG0100CHE_20140205.
    pos
-rw-rw-r-- 1 samuel samuel 1384 Jun 30 20:39 GG0100CHE_20140206.
    pos
-rw-rw-r-- 1 samuel samuel 1384 Jun 30 20:43 GG0100CHE_20140207.
    pos
-rw-rw-r-- 1 samuel samuel 1384 Jun 30 20:47 GG0100CHE_20140208.
    pos
-rw-rw-r-- 1 samuel samuel 1384 Jun 30 20:51 GG0100CHE_20140209.
    pos
-rw-rw-r-- 1 samuel samuel 1384 Jun 30 20:54 GG0100CHE_20140210.
    pos
```

[7] http://www.rtklib.com/prog/manual_2.4.2.pdf

```
-rw-rw-r-- 1 samuel samuel 1384 Jun 30 20:58 GG0100CHE_20140211.
    pos
-rw-rw-r-- 1 samuel samuel 1384 Jun 30 21:02 GG0100CHE_20140212.
    pos
-rw-rw-r-- 1 samuel samuel 1384 Jun 30 21:06 GG0100CHE_20140213.
    pos
-rw-rw-r-- 1 samuel samuel 1384 Jun 30 21:10 GG0100CHE_20140214.
    pos
-rw-rw-r-- 1 samuel samuel 1384 Jun 30 21:14 GG0100CHE_20140215.
    pos
-rw-rw-r-- 1 samuel samuel 1384 Jun 30 21:18 GG0100CHE_20140216.
    pos
-rw-rw-r-- 1 samuel samuel 1384 Jun 30 21:22 GG0100CHE_20140217.
    pos
-rw-rw-r-- 1 samuel samuel 1384 Jun 30 21:26 GG0100CHE_20140218.
    pos
-rw-rw-r-- 1 samuel samuel 1384 Jun 30 21:30 GG0100CHE_20140219.
    pos
-rw-rw-r-- 1 samuel samuel 1384 Jun 30 21:33 GG0100CHE_20140220.
    pos
-rw-rw-r-- 1 samuel samuel 1384 Jun 30 21:37 GG0100CHE_20140221.
    pos
-rw-rw-r-- 1 samuel samuel 1384 Jun 30 21:41 GG0100CHE_20140222.
    pos
-rw-rw-r-- 1 samuel samuel 1384 Jun 30 21:45 GG0100CHE_20140223.
    pos
-rw-rw-r-- 1 samuel samuel 1384 Jun 30 21:49 GG0100CHE_20140224.
    pos
-rw-rw-r-- 1 samuel samuel 1384 Jun 30 21:53 GG0100CHE_20140225.
    pos
-rw-rw-r-- 1 samuel samuel 1384 Jun 30 21:57 GG0100CHE_20140226.
    pos
-rw-rw-r-- 1 samuel samuel 1384 Jun 30 22:01 GG0100CHE_20140227.
    pos
-rw-rw-r-- 1 samuel samuel 1384 Jun 30 22:05 GG0100CHE_20140228.
    pos
-rw-rw-r-- 1 samuel samuel 5914 Jun 30 20:21 log_20140201.txt
-rw-rw-r-- 1 samuel samuel 5914 Jun 30 20:25 log_20140202.txt
-rw-rw-r-- 1 samuel samuel 5914 Jun 30 20:28 log_20140203.txt
-rw-rw-r-- 1 samuel samuel 5914 Jun 30 20:32 log_20140204.txt
-rw-rw-r-- 1 samuel samuel 5914 Jun 30 20:36 log_20140205.txt
-rw-rw-r-- 1 samuel samuel 5914 Jun 30 20:39 log_20140206.txt
-rw-rw-r-- 1 samuel samuel 5914 Jun 30 20:43 log_20140207.txt
-rw-rw-r-- 1 samuel samuel 5914 Jun 30 20:47 log_20140208.txt
-rw-rw-r-- 1 samuel samuel 5914 Jun 30 20:51 log_20140209.txt
-rw-rw-r-- 1 samuel samuel 5919 Jun 30 20:54 log_20140210.txt
-rw-rw-r-- 1 samuel samuel 5919 Jun 30 20:58 log_20140211.txt
-rw-rw-r-- 1 samuel samuel 5919 Jun 30 21:02 log_20140212.txt
-rw-rw-r-- 1 samuel samuel 6252 Jun 30 21:06 log_20140213.txt
-rw-rw-r-- 1 samuel samuel 5919 Jun 30 21:10 log_20140214.txt
-rw-rw-r-- 1 samuel samuel 5919 Jun 30 21:14 log_20140215.txt
-rw-rw-r-- 1 samuel samuel 5919 Jun 30 21:18 log_20140216.txt
-rw-rw-r-- 1 samuel samuel 5919 Jun 30 21:22 log_20140217.txt
-rw-rw-r-- 1 samuel samuel 5919 Jun 30 21:26 log_20140218.txt
```

```
-rw-rw-r-- 1 samuel samuel 5919 Jun 30 21:30 log_20140219.txt
-rw-rw-r-- 1 samuel samuel 5919 Jun 30 21:33 log_20140220.txt
-rw-rw-r-- 1 samuel samuel 5919 Jun 30 21:37 log_20140221.txt
-rw-rw-r-- 1 samuel samuel 5919 Jun 30 21:41 log_20140222.txt
-rw-rw-r-- 1 samuel samuel 5919 Jun 30 21:45 log_20140223.txt
-rw-rw-r-- 1 samuel samuel 5919 Jun 30 21:49 log_20140224.txt
-rw-rw-r-- 1 samuel samuel 5919 Jun 30 21:53 log_20140225.txt
-rw-rw-r-- 1 samuel samuel 5919 Jun 30 21:57 log_20140226.txt
-rw-rw-r-- 1 samuel samuel 5919 Jun 30 22:01 log_20140227.txt
-rw-rw-r-- 1 samuel samuel 5919 Jun 30 22:05 log_20140228.txt
```

Now it would be interesting to visualize these positions somehow. A short instruction guide for this purpose is given in section 4.3 and the corresponding images can be found in appendix B.

## 4.3 Plot the solution

Once the solution for a day or several days are obtained one might want to have a quick look at the positions. For this purpose Rtklib also offers a tool called *rtkplot*.

**Rtklplot** This is a very intuitive GUI and is used to plot solutions. In this paragraph only the most important aspects of its usage are described, for more detailed information, one should consult the official manual [8].
For a rudimentar use, one only needs to follow these steps:
1. Open the *rtkplot* GUI application.
2. In the menu ≪File≫-≪Open Solution≫ choose the solutions to be plotted.
Figure 7 shows the *rtkplot* GUI home screen and in appendix B some more screenshots showing plots of the position of receiver number *43* with label *GG01* are pictured.

[8] http://www.rtklib.com/prog/manual_2.4.2.pdf

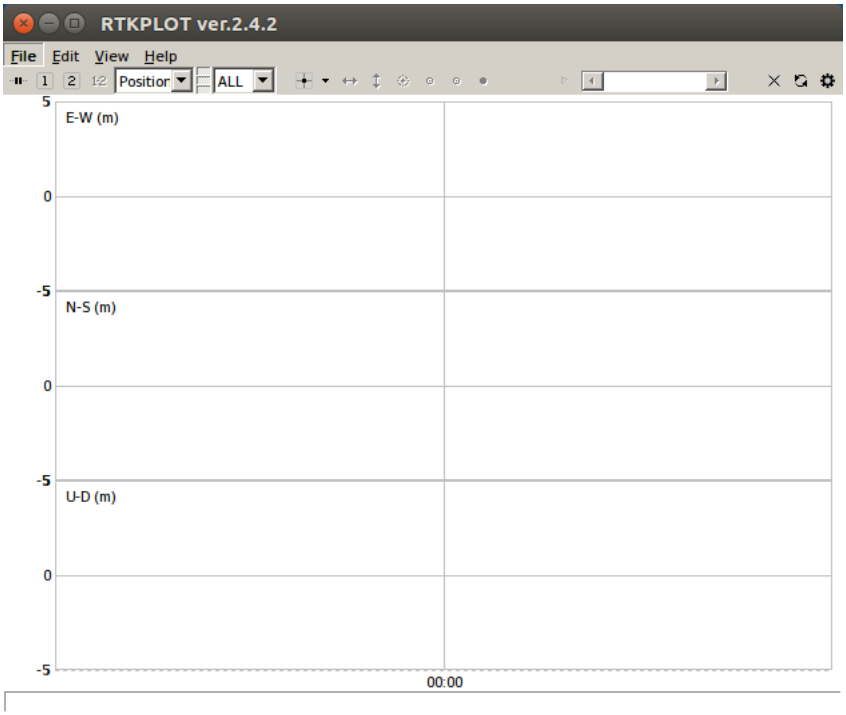


Figure 5: The *rtkplot* GUI

# 5 Outlook

**More IGS Data** In order to obtain a more accurate final solution, more additional data provided by the IGS could be taken into account. Such as for example earth rotation parameters or troposheric and ionospheric composition. An overview of the different products offered by the IGS can be found online[9].
In order to include these additional information in the solution computation the IGS data download part of the tool must be expanded and the downloaded data must be passed to the *rnx2rtkp* tool in the *calculate.sh* script.

**Multiple Cores** The execution of the program takes between two and three minutes. In order to obtain the final solution faster, the step to take could be that the program is launched using several cores instead of one.

**Default temporary directory** The program creates a new temporary directory while running. All IGS and GPS data are downloaded into it and

[9]https://igscb.jpl.nasa.gov/components/prods.html

removed before the end, because the directory is eventually deleted. The individual files vary in size from a few hundreds of bytes up to some Megabytes (e.g. the clock correction data). The data transport and therefore the whole executin of the program could be accelerated if the tool would use the */tmp* directory exixsing on every computer running a Unix system.

# 6 Learning effects

When starting this semester project I had not programmed in Python before, neither have I ever written any bash script. So I had at first to get a bit familiar with these two languages. And finally I handled both more or less well. So I can say that from a programming point of view I have learned enough.

Second thing I had never used before was the *svn Subversion* control tool and the connection to remote servers via ssh. Even though this wasn‘t that hard a piece of work I am glad to have used it since I am expecting to use it again.

Furthermore it was not too easy to start working with the Rtklib tools, since there I wasn't able to find good documentation on them and some error messages even only appeared in Japanese. Moreover I had no idea about GPS computation relative to the reference station as RTK does. Nevertheless in the end and thanks to much help from my supervisor I was able to use them correctly.

A part that posed more problems was the GPS measurement per se. Here I had no idea before starting the project. Although I fortunately hadn‘t to go too deep into this matter I learned some basics in this area too. As for example the ECEF positioning format in which the final solution is given as well as the fact that for precise positioning it is not enough for a receiver to communicate with less than four satellites.
And finally what I encountered to be another hard piece of work was to get familiar with the many different GNSS-related data formats. Some of them weren‘t ASCII based so I had no idea about its real content. And those in ASCII weren‘t really understandable either. Also I couldn't imagine why one needs several different files in order to compute a solution for only one position. That is now very clear and actually quiet plausible. Another question that bothered me quite a long time was that I always had the feeling to need too many information or even to use some information twice. As for example I couldn‘t see what the difference between the required *sp3* and the navigation file was. Again I profited from my supervisor who has a much brighter background in this field and helped me wherever he could. Even though I am far from considering me an expert now, I learned some interesting facts.

# Appendices

# A Input and Output files of the Examples

## A.1 Basic Example

Listing 3: The parameter file *parameter_file.txt* used in the basic example

```
# sample parameter file for the compute_solution.sh script

# IMPORTANT:    the gps data directory (gps_data_dir) must have
    the following subdirectory structure:
#
#               gps_data_dir
#                  /roverstation_deployment
#                    /rover_station_label
#                  /basestation_deploymentelative to the reference
     station as RTK does.
#                    /base_station_label
#
#               Because the downloaded data will be stored in
    those subdirectories.

[positions]

roverstation_deployment:        dirruhorn
rover_station_nr:               43
rover_station_label:            GG01

basestation_deployment:         dirruhorn
base_station_nr:                39
base_station_label:             RG01

ref_pos_x:                      4392040.8940
ref_pos_y:                      602680.4440
ref_pos_z:                      4574388.7730


[servers]

#dataserver_1: clk and sp3 files
#dataserver_2: nav (.%yn)
#dataserver_3: DCB

dataserver_1=ftp://cddis.gsfc.nasa.gov/gnss/products
dataserver_2=ftp://cddis.gsfc.nasa.gov/gnss/data/daily
dataserver_3=ftp://ftp.unibe.ch/aiub/CODE
```

```
[directories]

gps_data_dir:           /home/samuel/semester_project/gps_data
igs_data_dir:           /home/samuel/semester_project/igs_data
output_dir:             /home/samuel/semester_project/results
rtklib_dir:             /home/samuel/rtklib_2.4.2/cuis
rtklib_options_dir:     /home/samuel/rtklib_2.4.2/conf



[files]

rtklib_conf_file:       rtkpost_dirruhorn_static_v21.conf
```

Listing 4: The configuration file for the *rnx2rtkp* tool (*rtkpost_dirruhorn_static_v21.conf*) used in the basic example

```
# rtkpost options (2016/04/11 12:09:16, v.2.4.2)

pos1-posmode       =static     # (0:single,1:dgps,2:kinematic,3:
    static,4:movingbase,5:fixed,6:ppp-kine,7:ppp-static)
pos1-frequency     =l1+l2+l5      # (1:l1,2:l1+l2,3:l1+l2+l5,4:l1+
    l2+l5+l6,5:l1+l2+l5+l6+l7)
pos1-soltype       =combined   # (0:forward,1:backward,2:combined)
pos1-elmask        =10         # (deg)
pos1-snrmask_r     =off        # (0:off,1:on)
pos1-snrmask_b     =off        # (0:off,1:on)
pos1-snrmask_L1    =0,0,0,0,0,0,0,0,0
pos1-snrmask_L2    =0,0,0,0,0,0,0,0,0
pos1-snrmask_L5    =0,0,0,0,0,0,0,0,0
pos1-dynamics      =off        # (0:off,1:on)
pos1-tidecorr      =on         # (0:off,1:on)
pos1-ionoopt       =est-stec       # (0:off,1:brdc,2:sbas,3:dual-
    freq,4:est-stec,5:ionex-tec,6:qzs-brdc,7:qzs-lex,8:vtec_sf,9:
    vtec_ef,10:gtec)
pos1-tropopt       =est-ztd    # (0:off,1:saas,2:sbas,3:est-ztd,4:
    est-ztdgrad)
pos1-sateph        =precise    # (0:brdc,1:precise,2:brdc+sbas,3:
    brdc+ssrapc,4:brdc+ssrcom)
pos1-posopt1       =off        # (0:off,1:on)
pos1-posopt2       =off        # (0:off,1:on)
pos1-posopt3       =off        # (0:off,1:on)
pos1-posopt4       =off        # (0:off,1:on)
pos1-posopt5       =off        # (0:off,1:on)
pos1-exclsats      =           # (prn ...)
pos1-navsys        =1          # (1:gps+2:sbas+4:glo+8:gal+16:qzs
    +32:comp)
pos2-armode        =continuous # (0:off,1:continuous,2:
    instantaneous,3:fix-and-hold)
pos2-gloarmode     =on         # (0:off,1:on,2:autocal)
pos2-arthres       =3
pos2-arlockcnt     =0
pos2-arelmask      =0          # (deg)
pos2-arminfix      =10
pos2-elmaskhold    =0          # (deg)
pos2-aroutcnt      =5
pos2-maxage        =30         # (s)
pos2-syncsol       =off        # (0:off,1:on)
pos2-slipthres     =0.05       # (m)
pos2-rejionno      =30         # (m)
pos2-rejgdop       =30
pos2-niter         =1
pos2-baselen       =0          # (m)
```

```
pos2-basesig       =0          # (m)
out-solformat      =xyz        # (0:llh,1:xyz,2:enu,3:nmea)
out-outhead        =on         # (0:off,1:on)
out-outopt         =on         # (0:off,1:on)
out-timesys        =gpst       # (0:gpst,1:utc,2:jst)
out-timeform       =hms        # (0:tow,1:hms)
out-timendec       =3
out-degform        =deg        # (0:deg,1:dms)
out-fieldsep       =
out-height         =ellipsoidal # (0:ellipsoidal,1:geodetic)
out-geoid          =internal   # (0:internal,1:egm96,2:egm08_2
    .5,3:egm08_1,4:gsi2000)
out-solstatic      =single     # (0:all,1:single)
out-nmeaintv1      =0          # (s)
out-nmeaintv2      =0          # (s)
out-outstat        =off        # (0:off,1:state,2:residual)
stats-eratio1      =100
stats-eratio2      =100
stats-errphase     =0.003      # (m)
stats-errphaseel   =0.003      # (m)
stats-errphasebl   =0          # (m/10km)
stats-errdoppler   =10         # (Hz)
stats-stdbias      =30         # (m)
stats-stdiono      =0.03       # (m)
stats-stdtrop      =0.3        # (m)
stats-prnaccelh    =10         # (m/s^2)
stats-prnaccelv    =10         # (m/s^2)
stats-prnbias      =0.0001     # (m)
stats-prniono      =0.001      # (m)
stats-prntrop      =0.0001     # (m)
stats-clkstab      =5e-12      # (s/s)
ant1-postype       =llh        # (0:llh,1:xyz,2:single,3:posfile
    ,4:rinexhead,5:rtcm)
ant1-pos1          =0          # (deg|m)
ant1-pos2          =0          # (deg|m)
ant1-pos3          =0          # (m|m)
ant1-anttype       =
ant1-antdele       =0          # (m)
ant1-antdeln       =0          # (m)
ant1-antdelu       =0          # (m)
ant2-postype       =llh        # (0:llh,1:xyz,2:single,3:posfile
    ,4:rinexhead,5:rtcm)
ant2-pos1          =46.0902331253624 # (deg|m)
ant2-pos2          =7.81338958264756 # (deg|m)
ant2-pos3          =3025.62992172179 # (m|m)
ant2-anttype       =
ant2-antdele       =0          # (m)
ant2-antdeln       =0          # (m)
ant2-antdelu       =0          # (m)
misc-timeinterp    =off        # (0:off,1:on)
misc-sbasatsel     =0          # (0:all)
misc-rnxopt1       =
misc-rnxopt2       =
file-satantfile    =/home/samuel/Semesterarbeit/testumgebung/igs/
```

```
    igs08.atx
file-rcvantfile    =/home/samuel/Semesterarbeit/testumgebung/igs/
    igs08.atx
file-staposfile    =
file-geoidfile     =
file-ionofile      =/usr/whymper/data-05/permasense_vault/gps/
    external_dataproducts/igs_data/igsg%n0.%yi
file-dcbfile       =/home/samuel/semester_project/igs_data/
    P1P21402.DCB
file-eopfile      =/usr/whymper/data-05/permasense_vault/gps/
    external_dataproducts/igs_data/igs%W7.erp
file-blqfile       =
file-tempdir       =
file-geexefile     =
file-solstatfile   =
file-tracefile     =
```

Listing 5: The logfile *log_20140213.txt* created in the basic example

```
Logfile started at: 30/06/2016 18:08

Called the compute_solution.sh script for the date: 2014 2 13



::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: :::
    ::: ::: :::

Read the [directories] section of configuration file

All required keys in the [directories] section were found.

Check the existence of the given directories:

All directories have been found.

Read the [files] section of configuration file
All required keys in the [files] section were found.


Read the [positions] section of configuration file

All required keys in the [positions] section were found.

Read the [servers] section of configuration file
All required keys in the [servers] section were found.



::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: :::
    ::: ::: :::

Download the gps data from the permasense public server:

Download the raw gps data for the roverstation GG01 for 2014 2 13
Downloading GPS data for Position  43
From 13/02/2014 01:00:00 to 14/02/2014 00:59:59 CET
Download gps data for station with label GG01 and position
    number43
Download from: http://data.permasense.ch/multidata?vs[1]=
    dirruhorn_gps_raw__binary__mapped&field[1]=gps_raw_data&
    download_format=binary&order=asc&c_join[1]=and&c_vs[1]=
    dirruhorn_gps_raw__binary__mapped&c_field[1]=position&c_min
    [1]=42&c_max[1]=43&c_vs[2]=dirruhorn_gps_raw__binary__mapped&
    c_join[2]=and&c_field[2]=gps_missing_sv&c_min[2]=-inf&c_max
    [2]=0&c_vs[3]=dirruhorn_gps_raw__binary__mapped&c_join[3]=and&
```

```
    c_field[3]=gps_unixtime&c_min[3]=1392249600000&c_max
    [3]=1392335999000&timeline=gps_unixtime&time_format=unix HTTP
    /1.1

The downloaded data are written into file named: GG010440.ubx
Downloaded the ubx file to /home/samuel/semester_project/tool/
    scripts/../temp_20140213/GG010440.ubx
Download into the temporary folder successful
Download the raw gps data for the basestation RG01 for 2014 2 13
Downloading GPS data for Position  39
From 13/02/2014 01:00:00 to 14/02/2014 00:59:59 CET
Download gps data for station with label RG01 and position
    number39
Download from: http://data.permasense.ch/multidata?vs[1]=
    dirruhorn_gps_raw__binary__mapped&field[1]=gps_raw_data&
    download_format=binary&order=asc&c_join[1]=and&c_vs[1]=
    dirruhorn_gps_raw__binary__mapped&c_field[1]=position&c_min
    [1]=38&c_max[1]=39&c_vs[2]=dirruhorn_gps_raw__binary__mapped&
    c_join[2]=and&c_field[2]=gps_missing_sv&c_min[2]=-inf&c_max
    [2]=0&c_vs[3]=dirruhorn_gps_raw__binary__mapped&c_join[3]=and&
    c_field[3]=gps_unixtime&c_min[3]=1392249600000&c_max
    [3]=1392335999000&timeline=gps_unixtime&time_format=unix HTTP
    /1.1

The downloaded data are written into file named: RG010440.ubx
Downloaded the ubx file to /home/samuel/semester_project/tool/
    scripts/../temp_20140213/RG010440.ubx
Download into the temporary folder successful


::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: :::
    ::: ::: :::

Download the igs data from servers:
elative to the reference station as RTK does.
Download data from: ftp://cddis.gsfc.nasa.gov/gnss/products/1779/
    igs17794.clk.Z
Write the data into igs17794.clk.Z
The file igs17794.clk.Z was successfully downloaded!
Download data from: ftp://cddis.gsfc.nasa.gov/gnss/products/1779/
    igs17794.sp3.Z
Write the data into igs17794.sp3.Z
The file igs17794.sp3.Z was successfully downloaded!
Download data from: ftp://cddis.gsfc.nasa.gov/gnss/data/daily
    /2014/044/14n/brdc0440.14n.Z
Write the data into brdc0440.14n.Z
The file brdc0440.14n.Z was successfully downloaded!
Download data from: ftp://ftp.unibe.ch/aiub/CODE/2014/P1P21402.DCB
    .Z
Write the data into P1P21402.DCB.Z
The file P1P21402.DCB.Z was successfully downloaded!

The downloaded files are temporarily stored at /home/samuel/
    semester_project/tool/scripts/../temp_20140213
```

```
::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: :::

Update the rnx2rtkp configuration file
found the key file-dcbfile

Update config-file : the key file-dcbfile gets the new value /home
    /samuel/semester_project/igs_data/P1P21402.DCB assigned.
The update was successful.


::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: :::

Conversion of gps raw data into rinex format of observation data:


Conversion of the rover station gps raw data: GG010440.ubx:
Converted the rover station's gps file.

Conversion of the base station gps raw data: RG010440.ubx:
Converted the base station's gps file.



::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: :::

Calculation of the solution for the date 2014 2 13:


Solution calculation for rover GG01 (position 43) and base station
     RG01 (position 39) for day 2014 2 13


rnx2rtkp call: /home/samuel/rtklib_2.4.2/cuis/rnx2rtkp -x 0 -k /
    home/samuel/rtklib_2.4.2/conf/rtkpost_dirruhorn_static_v21.
    conf -o /home/samuel/semester_project/results/
    GG0100CHE_20140213.pos -p 3 -c -e -r 4392040.8940 602680.4440
    4574388.7730 /home/samuel/semester_project/tool/scripts/../
    temp_20140213/GG010440.14O /home/samuel/semester_project/tool/
    scripts/../temp_20140213/RG010440.14O /home/samuel/
    semester_project/tool/scripts/../temp_20140213/brdc0440.14n /
    home/samuel/semester_project/tool/scripts/../temp_20140213/
    igs17794.sp3 /home/samuel/semester_project/tool/scripts/../
    temp_20140213/igs17794.clk

The calculation of the rover GG01 (position 43) was successful.


::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: ::: :::
```

```
Store the downloaded igs data.

A File named igs17794.clk is already stored at /home/samuel/
    semester_project/igs_data. The freshly downloaded one is
    discarded.
A File named igs17794.sp3 is already stored at /home/samuel/
    semester_project/igs_data. The freshly downloaded one is
    discarded.
A File named brdc0440.14n is already stored at /home/samuel/
    semester_project/igs_data. The freshly downloaded one is
    discarded.
A File named P1P21402.DCB is already stored at /home/samuel/
    semester_project/igs_data. The freshly downloaded one is
    discarded.
A File named RG010440.14O is already stored at /home/samuel/
    semester_project/gps_data/dirruhorn/RG01. The freshly
    converted one is discarded.
A File named GG010440.14O is already stored at /home/samuel/
    semester_project/gps_data/dirruhorn/GG01. The freshly
    converted one is discarded.
Delete the temporary folder /home/samuel/semester_project/tool/
    scripts/../temp_20140213.
Logfile closed at: 30/06/2016 18:11
```

Listing 6: The solution file *lGG0100CHE_20140213.pos* created in the basic example

```
% program   : RTKLIB ver.2.4.2
% inp file  : /home/samuel/semester_project/tool/scripts/../
    temp_20140213/GG010440.14O
% inp file  : /home/samuel/semester_project/tool/scripts/../
    temp_20140213/RG010440.14O
% inp file  : /home/samuel/semester_project/tool/scripts/../
    temp_20140213/brdc0440.14n
% inp file  : /home/samuel/semester_project/tool/scripts/../
    temp_20140213/igs17794.sp3
% inp file  : /home/samuel/semester_project/tool/scripts/../
    temp_20140213/igs17794.clk
% obs start : 2014/02/13 00:01:00.0 GPST (week1779 345660.0s)
% obs end   : 2014/02/14 00:00:00.0 GPST (week1779 432000.0s)
% pos mode  : static
% freqs     : L1+L2+L5
% solution  : combined
% elev mask : 10.0 deg
% dynamics  : off
% tidecorr  : on
% ionos opt : estimation
% tropo opt : est ztd
% ephemeris : precise
% amb res   : continuous
% val thres : 3.0
% antenna1  :                       ( 0.0000  0.0000  0.0000)
% antenna2  :                       ( 0.0000  0.0000  0.0000)
% ref pos   :  4392040.8940   602680.4440  4574388.7730
%
% (x/y/z-ecef=WGS84,Q=1:fix,2:float,3:sbas,4:dgps,5:single,6:ppp,
    ns=# of satellites)
%  GPST                      x-ecef(m)      y-ecef(m)      z-ecef(
    m)   Q  ns   sdx(m)   sdy(m)   sdz(m)  sdxy(m)  sdyz(m)  sdzx(
    m) age(s)  ratio
2014/02/13 00:10:30.000   4391926.3790    602616.3559
    4574411.9242   1   4   0.0011   0.0009   0.0010  -0.0002
    0.0001   0.0010   0.00   82.3
```

# B Rtkplot screenshots

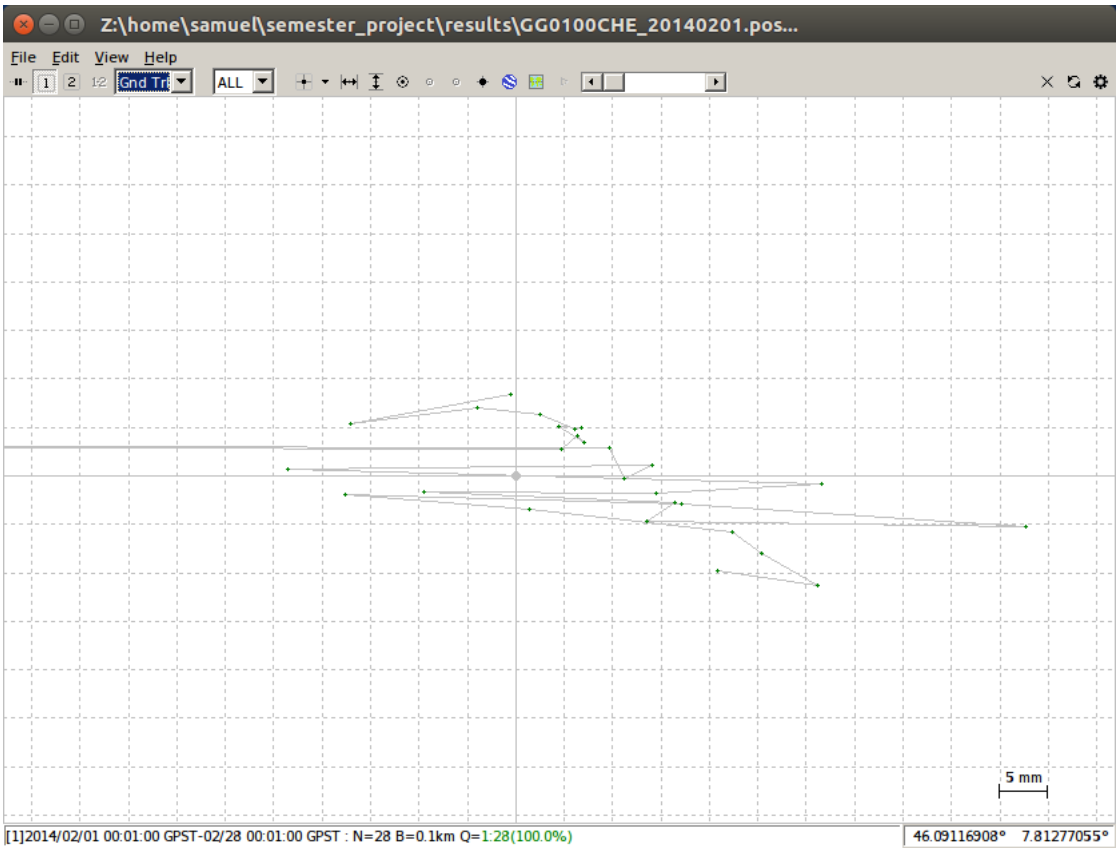


Figure 6: Plot of the ground tracking for position GG01 relative to the reference position during February 2014. The plotted solutions were calculated in section 4.2 under the paragraph «Several Days»

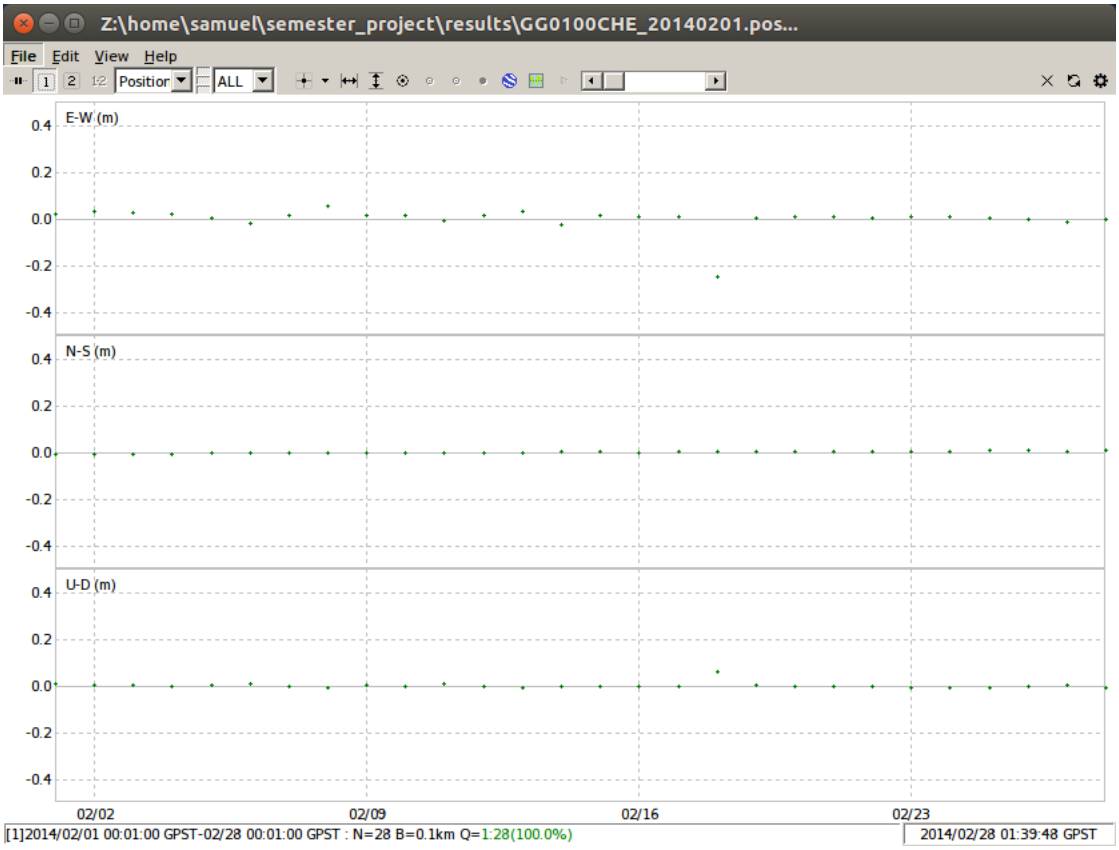


Figure 7: Plot of the east-west, north-south and up-down movement of position GG01 relative to the reference position during February 2014. The plotted solutions were calculated in section 4.2 under the paragraph ≪Several Days≫