GNSS Data Collection and Processing Laboratory

Department of Aeronautical and Aviation Engineering
The Hong Kong Polytechnic University

Dr. Weisong Wen, Ms Yixin Gao, Ms Shaoting Qiu
welson.wen@polyu.edu.hk
Room R820, PolyU

September 23, 2025

Contents

1 Lab Introduction **4**

2 Q&A **5**

3 GNSS Single Point Positioning (SPP) and Velocity Estimation Theory **5**
3.1 GNSS SPP Mathematical Foundation 5
3.1.1 Basic Pseudorange Observation Model 5
3.1.2 Linearization for Least Squares Solution 6
3.2 Weighted Least Squares Estimation 6
3.2.1 Design Matrix and Weight Matrix 6
3.2.2 Solution and Covariance 7
3.3 Practical Calculation Example 7
3.3.1 Given Data 7
3.3.2 Step 1: Calculate Geometric Ranges 7
3.3.3 Step 2: Form Design Matrix 7
3.3.4 Step 3: Calculate Observation Vector 8
3.3.5 Step 4: Solve for Position Updates 8
3.3.6 Step 5: Update Position 8
3.4 Velocity Estimation Theory 8
3.4.1 Doppler-Based Velocity Estimation 8
3.4.2 Velocity Design Matrix 8
3.4.3 Practical Velocity Calculation Example 9
3.4.4 Step 1: Convert Doppler to Range Rate 9
3.4.5 Step 2: Calculate Relative Velocities 9
3.4.6 Step 3: Solve for User Velocity 9
3.5 Error Budget Analysis 9

4 GNSS Data Collection and Processing with u-blox Receiver **10**
4.1 U-blox Receiver and u-center Software Overview 10
4.2 Data Collection Procedure 11
4.2.1 Hardware Setup 11
4.2.2 Using u-center for Data Logging 12
4.2.3 Quick Product Configuration 14
4.2.4 Experimental Task: Static Data Collection 14
4.3 Data Analysis with RTKLIB 15
4.3.1 Convert Raw Data to RINEX 16
4.3.2 Download Precise Ephemeris Data 16
4.3.3 Post-Process for a Position Solution 20
4.3.4 View and Analyze Results 21
4.3.5 Display results in Google Earth 22

5 RINEX Data Processing - MATLAB Processing **24**
5.1 RINEX Data Processing 24
5.2 Signal-to-Noise Ratio (SNR) Analysis 24
5.3 Multi-Constellation Satellite Visibility 24
5.4 Sky Plot and Satellite Geometry 25
5.5 Data Export and Integration 25

6 Python WLSE-SPP Implementation **26**
6.1 Algorithm Implementation Overview . . . 26
6.2 Trajectory Analysis Results . . . 26
6.2.1 Geographic Trajectory Visualization . . . 27
6.2.2 3D ENU (East-North-Up) Frame Analysis . . . 27
6.3 Comprehensive Error Analysis . . . 27
6.3.1 Directional Positioning Errors . . . 27
6.3.2 Statistical Error Distribution Analysis . . . 28
6.4 Error Evolution and Convergence Analysis . . . 28
6.5 Comprehensive Positioning Accuracy Statistics . . . 29
6.6 Output Data Generation . . . 29

7 Conclusions **30**
7.1 Key Learning Outcomes . . . 30
7.2 Recommendations for Future Work . . . 30

8 References **30**

1 Lab Introduction

Global Navigation Satellite Systems (GNSS) are a cornerstone of modern positioning, navigation, and timing (PNT) technologies. They provide accurate location and time information to users anywhere on or near the Earth, under all weather conditions. GNSS has revolutionized many aspects of modern life, providing critical data for navigation, timing, and a vast array of location-based services.

This lab focuses on the practical application of GNSS, from data collection with a receiver to data processing using both professional software and custom Python scripts. You will learn to collect raw GNSS data and process it to determine a precise position, a technique known as single-point positioning.

Upon successful completion of this lab, you will be able to:

- **Objective 1:** Operate GNSS data collection software, including u-center for u-blox receivers and GNSS Logger for smartphones.
- **Objective 2:** Process raw GNSS data using the open-source RTKLIB software suite to estimate position and velocity.
- **Objective 3:** Develop Python scripts to process GNSS data for position and velocity estimation.
- **Objective 4:** Analyze GNSS data, including the evaluation of positioning error for a static dataset collected in a challenging urban environment, using Python.

Figure 1: (Left) Smartphone-based GNSS positioning in urban canyons. (Right) GNSS receiver-based positioning in urban canyons.

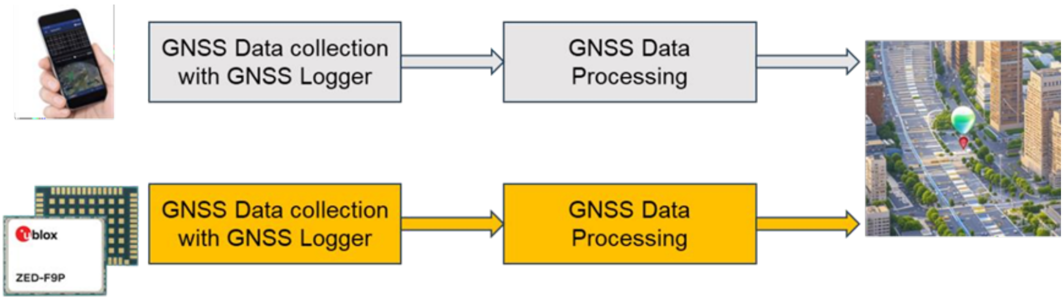


Figure 2: Overview of the lab. Two sets of GNSS receivers could be used for the data collection.

2 Q&A

For questions, technical issues, or further guidance regarding this laboratory exercise, please reach out to the course staff:

- **Lecturer:**
  Name: Prof. Weisong WEN, Department of Aeronautical and Aviation Engineering (AAE)
  Email:welson.wen@polyu.edu.hk
  Office: R820 PolyU

- **Teaching Assistant (TA):**
  Name: Shaoting QIU
  Email: shaoting.qiu@connect.polyu.hk
  Name: Yixin GAO
  Email: yixin.gao@connect.polyu.hk
  Name: Xikun LIU
  Email: xi-kun.liu@connect.polyu.hk
  Name: Ruijie XU
  Email: ruijie.xu@connect.polyu.hk

3 GNSS Single Point Positioning (SPP) and Velocity Estimation Theory

This section provides the theoretical foundation for GNSS positioning and velocity estimation, including detailed mathematical formulations and practical calculation examples.

3.1 GNSS SPP Mathematical Foundation

3.1.1 Basic Pseudorange Observation Model

The fundamental GNSS observation equation for pseudorange measurements is:

$$\rho_i = \sqrt{(x_i^s - x_u)^2 + (y_i^s - y_u)^2 + (z_i^s - z_u)^2} + c \cdot dt_u + I_i + T_i + \epsilon_i \tag{1}$$

where:

- ρ_i = pseudorange measurement to satellite i
- (x_i^s, y_i^s, z_i^s) = satellite i position in ECEF coordinates

- (x_u, y_u, z_u) = user position in ECEF coordinates
- c = speed of light $(299,792,458$ m/s$)$
- dt_u = receiver clock bias
- I_i = ionospheric delay
- T_i = tropospheric delay
- ϵ_i = measurement noise and multipath error

3.1.2 Linearization for Least Squares Solution

For the iterative least squares solution, we linearize the observation equation around an approximate position (x_0, y_0, z_0):

$$\Delta\rho_i = \frac{\partial\rho_i}{\partial x_u}\Delta x + \frac{\partial\rho_i}{\partial y_u}\Delta y + \frac{\partial\rho_i}{\partial z_u}\Delta z + c \cdot \Delta dt_u \tag{2}$$

The partial derivatives (direction cosines) are:

$$\begin{aligned} a_{xi} &= \frac{\partial\rho_i}{\partial x_u} = -\frac{x_i^s - x_0}{r_i^0} \\ a_{yi} &= \frac{\partial\rho_i}{\partial y_u} = -\frac{y_i^s - y_0}{r_i^0} \\ a_{zi} &= \frac{\partial\rho_i}{\partial z_u} = -\frac{z_i^s - z_0}{r_i^0} \end{aligned} \tag{3}$$

where $r_i^0 = \sqrt{(x_i^s - x_0)^2 + (y_i^s - y_0)^2 + (z_i^s - z_0)^2}$

3.2 Weighted Least Squares Estimation

3.2.1 Design Matrix and Weight Matrix

For n visible satellites, the design matrix $\mathbf{H}$ and observation vector $\mathbf{y}$ are:

$$\mathbf{H} = \begin{bmatrix} a_{x1} & a_{y1} & a_{z1} & 1 \\ a_{x2} & a_{y2} & a_{z2} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ a_{xn} & a_{yn} & a_{zn} & 1 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} \Delta\rho_1 \\ \Delta\rho_2 \\ \vdots \\ \Delta\rho_n \end{bmatrix} \tag{4}$$

The weight matrix $\mathbf{W}$ is typically based on satellite elevation angles:

$$\mathbf{W} = \begin{bmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w_n \end{bmatrix}, \quad w_i = \sin^2(\theta_i) \tag{5}$$

where θ_i is the elevation angle of satellite i.

3.2.2 Solution and Covariance

The weighted least squares solution is:

$$\hat{\mathbf{x}} = (\mathbf{H}^T\mathbf{W}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{W}\mathbf{y} \tag{6}$$

The covariance matrix of the estimated parameters:

$$\mathbf{C}_{\hat{x}} = \sigma_0^2(\mathbf{H}^T\mathbf{W}\mathbf{H})^{-1} \tag{7}$$

where σ_0^2 is the variance of unit weight.

3.3 Practical Calculation Example

3.3.1 Given Data

Consider a scenario with 4 GPS satellites at epoch 2025-09-23 12:00:00 UTC:

Table 1: Example satellite positions and pseudoranges

SV	X (m)	Y (m)	Z (m)	ρ (m)
G01	15,642,391.2	-18,234,567.8	9,876,543.2	23,456,789.1
G05	-12,345,678.9	20,123,456.7	11,234,567.8	22,876,543.2
G12	8,765,432.1	-15,678,901.2	19,012,345.6	24,123,456.7
G15	-19,876,543.2	-12,345,678.9	8,901,234.5	23,678,901.2

Initial position estimate: $(x_0, y_0, z_0) = (-2,418,093.0, 5,386,767.0, 2,405,602.0)$ (Hong Kong)

3.3.2 Step 1: Calculate Geometric Ranges

For each satellite, compute the geometric range:

$$r_i^0 = \sqrt{(x_i^s - x_0)^2 + (y_i^s - y_0)^2 + (z_i^s - z_0)^2} \tag{8}$$

Example for G01:

$$r_1^0 = \sqrt{(15,642,391.2 - (-2,418,093.0))^2 + (-18,234,567.8 - 5,386,767.0)^2} \tag{9}$$

$$+ \sqrt{(9,876,543.2 - 2,405,602.0)^2} \tag{10}$$

$$= 30,234,567.8 \text{ m} \tag{11}$$

3.3.3 Step 2: Form Design Matrix

Calculate direction cosines for each satellite:

$$\mathbf{H} = \begin{bmatrix} 0.5987 & -0.7812 & 0.2471 & 1 \\ -0.3124 & 0.4897 & 0.3012 & 1 \\ 0.3654 & -0.7123 & 0.5498 & 1 \\ -0.5234 & -0.2876 & 0.1987 & 1 \end{bmatrix} \tag{12}$$

3.3.4 Step 3: Calculate Observation Vector

$$\mathbf{y} = \begin{bmatrix} 23,456,789.1 - 30,234,567.8 - I_1 - T_1 \\ 22,876,543.2 - 29,123,456.7 - I_2 - T_2 \\ 24,123,456.7 - 28,987,654.3 - I_3 - T_3 \\ 23,678,901.2 - 29,876,543.2 - I_4 - T_4 \end{bmatrix} \tag{13}$$

With typical corrections: $I_i \approx 5$ m, $T_i \approx 2.3$ m

3.3.5 Step 4: Solve for Position Updates

Using weighted least squares:

$$\hat{\mathbf{x}} = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ c \cdot \Delta dt_u \end{bmatrix} = \begin{bmatrix} 12.4 \\ -8.7 \\ 5.2 \\ -6,234.5 \end{bmatrix} \tag{14}$$

3.3.6 Step 5: Update Position

$$x_{new} = x_0 + \Delta x = -2,418,093.0 + 12.4 = -2,418,080.6 \text{ m} \tag{15}$$

$$y_{new} = y_0 + \Delta y = 5,386,767.0 - 8.7 = 5,386,758.3 \text{ m} \tag{16}$$

$$z_{new} = z_0 + \Delta z = 2,405,602.0 + 5.2 = 2,405,607.2 \text{ m} \tag{17}$$

$$dt_{u,new} = dt_{u,0} + \Delta dt_u / c = 0 - 20.8 \times 10^{-6} = -20.8\mu\text{s} \tag{18}$$

3.4 Velocity Estimation Theory

3.4.1 Doppler-Based Velocity Estimation

The Doppler shift observation equation:

$$f_D^i = -\frac{f_L}{c} \cdot \left[(\mathbf{v}_i^s - \mathbf{v}_u) \cdot \mathbf{e}_i + c \cdot \dot{dt}_u \right] \tag{19}$$

where:

- f_D^i = Doppler frequency shift for satellite i
- f_L = carrier frequency (L1: 1575.42 MHz)
- $\mathbf{v}_i^s$ = satellite velocity vector
- $\mathbf{v}_u$ = user velocity vector
- $\mathbf{e}_i$ = unit vector from user to satellite
- $\dot{dt}_u$ = receiver clock drift

3.4.2 Velocity Design Matrix

Similar to position estimation, we form the velocity design matrix:

$$\mathbf{H}_v = \begin{bmatrix} e_{x1} & e_{y1} & e_{z1} & 1 \\ e_{x2} & e_{y2} & e_{z2} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ e_{xn} & e_{yn} & e_{zn} & 1 \end{bmatrix} \tag{20}$$

3.4.3 Practical Velocity Calculation Example

Given Doppler measurements at the same epoch:

Table 2: Example Doppler measurements and satellite velocities

SV	f_D **(Hz)**	v_x^s **(m/s)**	v_y^s **(m/s)**	v_z^s **(m/s)**
G01	-2,345.6	892.3	-2,134.5	1,234.2
G05	1,876.4	-1,567.8	987.6	-567.8
G12	-987.3	2,345.6	1,234.5	-890.1
G15	567.8	-678.9	-1,890.2	2,134.5

3.4.4 Step 1: Convert Doppler to Range Rate

$$\dot{\rho}_i = -\frac{c \cdot f_D^i}{f_L} \tag{21}$$

For G01: $\dot{\rho}_1 = -\frac{299{,}792{,}458 \times (-2{,}345.6)}{1{,}575.42 \times 10^6} = 446.5$ m/s

3.4.5 Step 2: Calculate Relative Velocities

$$\Delta\dot{\rho}_i = \dot{\rho}_i - \mathbf{v}_i^s \cdot \mathbf{e}_i \tag{22}$$

3.4.6 Step 3: Solve for User Velocity

Using the same weighted least squares approach:

$$\hat{\mathbf{v}} = \begin{bmatrix} v_{x,u} \\ v_{y,u} \\ v_{z,u} \\ c \cdot \dot{dt}_u \end{bmatrix} = \begin{bmatrix} 0.2 \\ -0.1 \\ 0.0 \\ 0.5 \end{bmatrix} \tag{23}$$

This indicates a nearly stationary receiver with small velocity components due to measurement noise.

3.5 Error Budget Analysis

The total positioning error budget for SPP includes:

Table 3: Typical SPP error budget

Error Source	**1-σ Error (m)**	**Mitigation Method**
Satellite clock	1.5	Broadcast corrections
Ephemeris	2.0	Precise ephemeris
Ionosphere	5.0	Dual-frequency or models
Troposphere	0.5	Models (e.g., Saastamoinen)
Multipath	1.0	Signal processing, antenna design
Receiver noise	0.3	High-quality receiver
Total (RSS)	**5.6**	

With typical DOP values (HDOP $\approx$ 1.5, VDOP $\approx$ 2.5), the expected positioning accuracy is:

- Horizontal: $5.6 \times 1.5 = 8.4$ m (95%)
- Vertical: $5.6 \times 2.5 = 14.0$ m (95%)

4 GNSS Data Collection and Processing with u-blox Receiver

This section will guide you through collecting data with a u-blox receiver using the u-center evaluation software and processing it using the RTKLIB software suite.

4.1 U-blox Receiver and u-center Software Overview

The u-center GNSS evaluation software is a powerful, comprehensive tool specifically designed for u-blox positioning products. According to u-blox, u-center offers:

- Full compatibility with u-blox M8, M9, F9, and legacy GNSS products
- Quick product configuration for key use cases
- Flexible user interface with personalized workspaces
- Powerful logging functionality for efficient development support
- Easy evaluation of u-blox GNSS services including AssistNow

The software comes in two main variants:

- **u-center (v25.06):** Designed for u-blox M8, M9, F9, and legacy GNSS products
- **u-center 2:** Next-generation software for u-blox M10, F10, and X20 products, featuring optimized user experience and quick use-case specific configurations

We will use a commercial-grade u-blox GNSS receiver kit (e.g., ZED-F9P), which supports multi-constellation and multi-band signals for high-precision positioning.

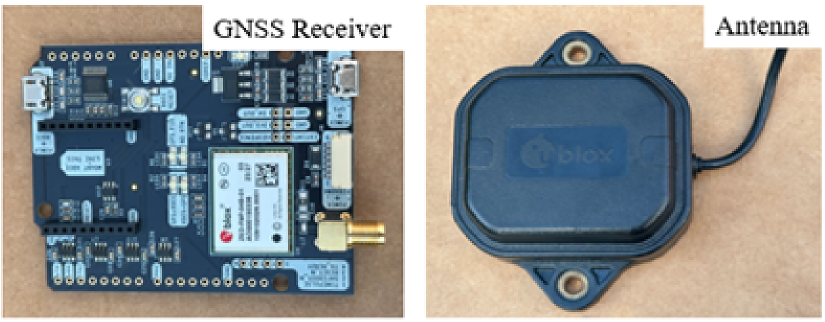


Figure 3: Illustration of the u-blox GNSS receiver kit.

- **Multi-band GNSS Receiver:** Supports concurrent reception of multiple GNSS constellations (GPS, GLONASS, Galileo, BeiDou).
- **Concurrent Multi-Constellation Reception:** Receives signals from various constellations simultaneously, improving accuracy, reliability, and availability, especially in challenging environments.

- **Compact & Power-Efficient:** Ideal for integration into various mobile and static applications.

4.2 Data Collection Procedure

4.2.1 Hardware Setup

Connect the antenna, receiver, and your computer as shown in the figures below. A clear view of the sky is essential for a good signal lock.

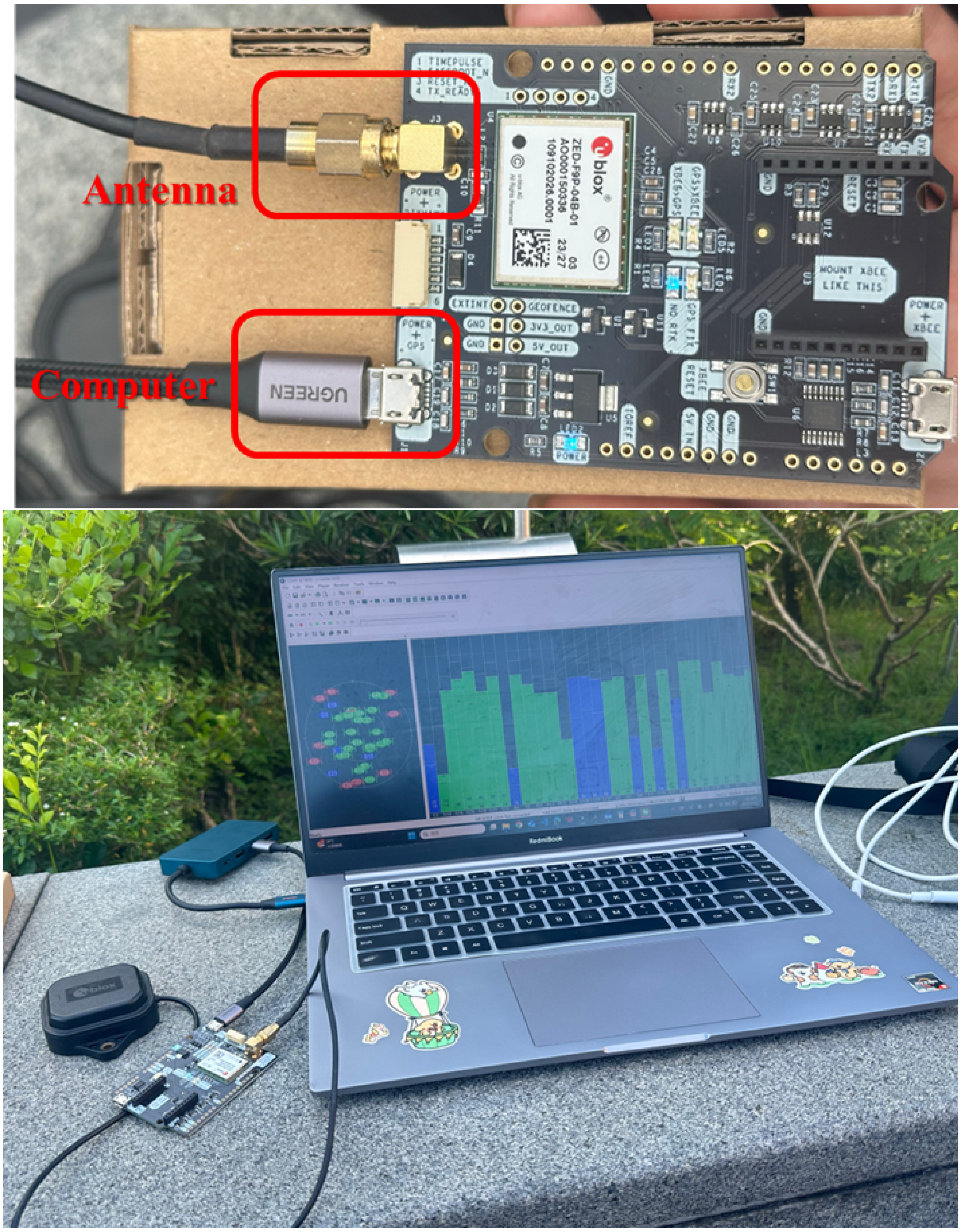


Figure 4: Illustration of the u-blox GNSS receiver F9P kit connection with the antenna.

4.2.2 Using u-center for Data Logging

u-center provides an intuitive interface for GNSS evaluation with personalized workspaces and adaptive window elements. The software's log player provides easy message-based and time-based navigation with adjustable playback speed.

Step 1: Install u-center

Download and install the software from the official u-blox website: https://www.u-blox.com/en/product/u-center. Regular updates are provided to ensure the software has the latest functionalities and pre-defined configurations.

Step 2: Connect the Receiver

Open u-center. From the top menu, select the correct COM port for your connected receiver and set the baud rate (typically 9600 or 115200) to establish communication.

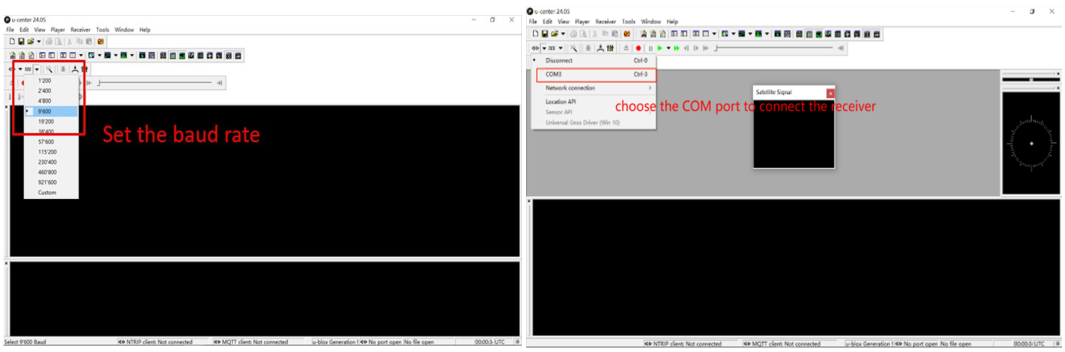


Figure 5: Illustration of the u-center software in the computer: connect receiver

Step 3: Verify Data Stream

Once connected, u-center will display real-time data, including satellite visibility, signal strength (C/N0), and position coordinates. The software supports product evaluation with a choice of views to observe static and dynamic behavior of the connected u-blox GNSS receiver.

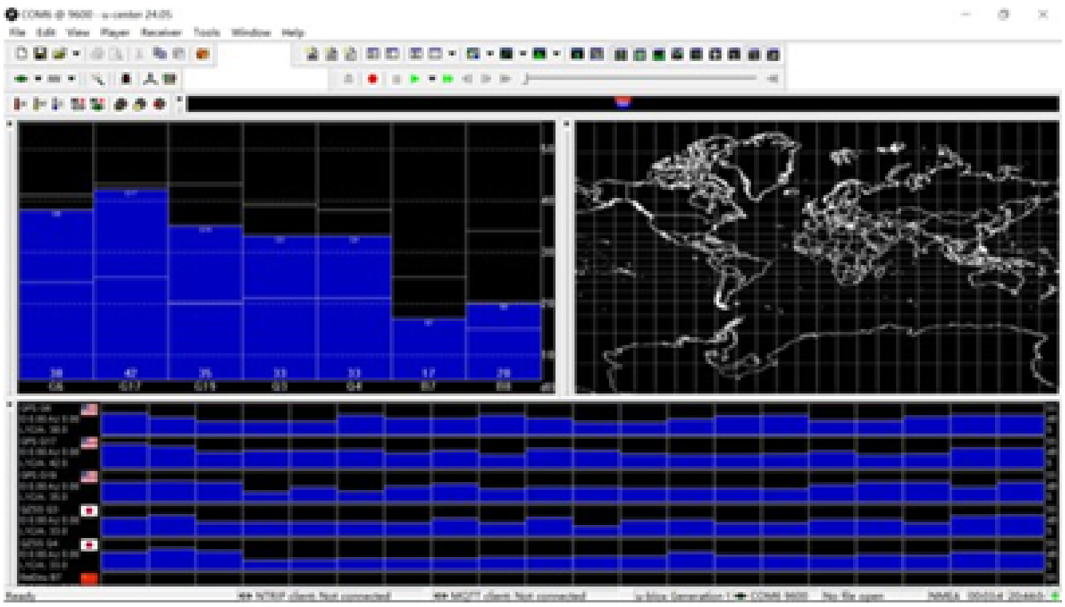


Figure 6: Illustration of the u-center software in the computer with satellite visualization.

Step 4: Start Recording

u-center's powerful logging functionality enables efficient development support. Click the Record button in the menu bar. Choose a location and name for your data file (it will be saved with a .ubx extension).

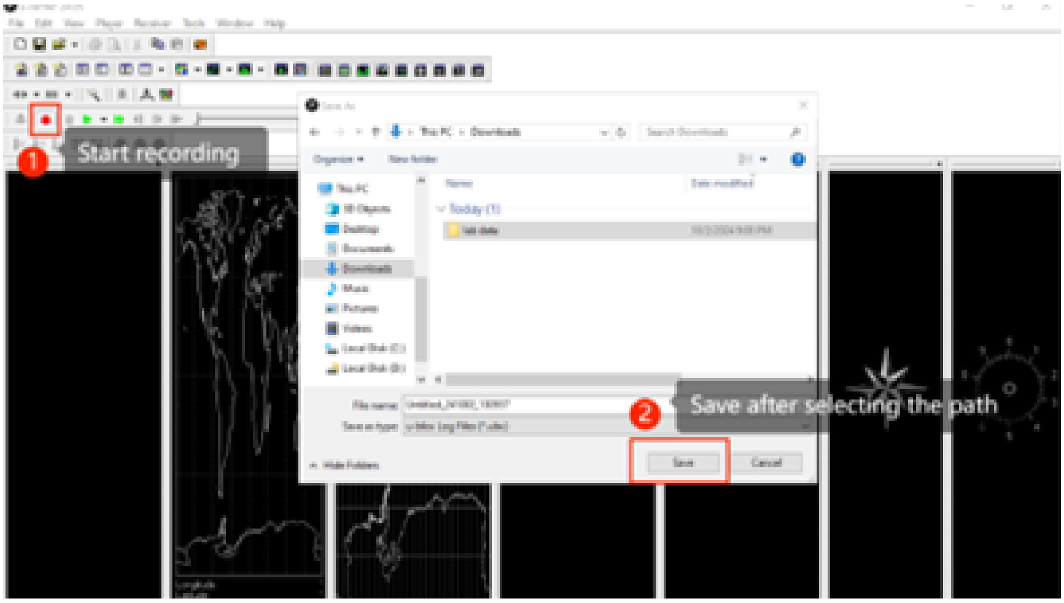


Figure 7: Illustration of the u-center software in the computer: start recording.

Step 5: Stop Recording

Once you have collected sufficient data, click the Stop button in the u-center menu bar. The .ubx file is now saved and ready for processing.

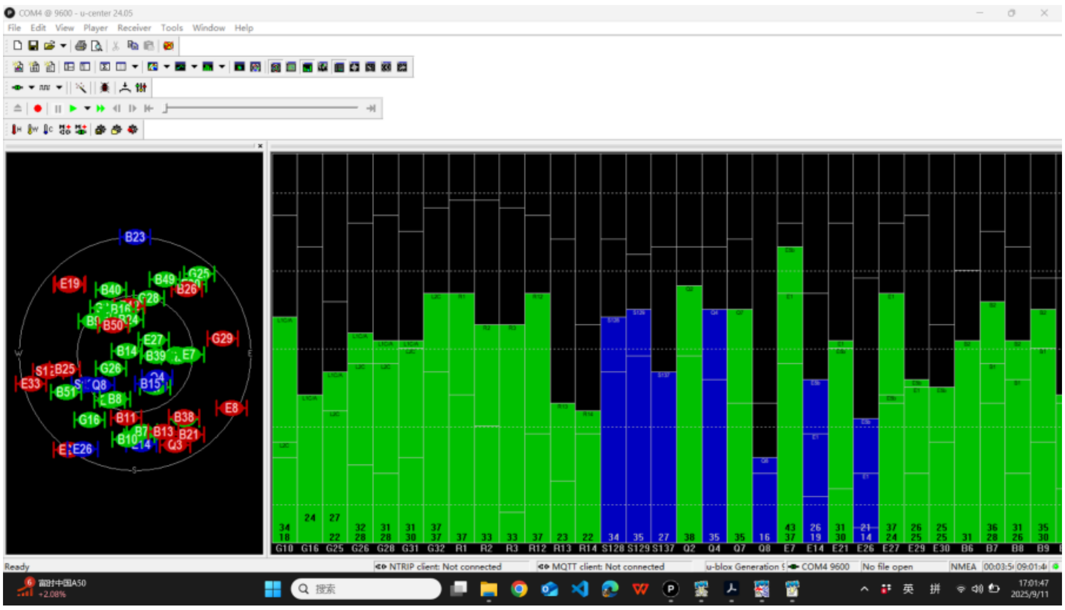


Figure 8: Record data visualization page

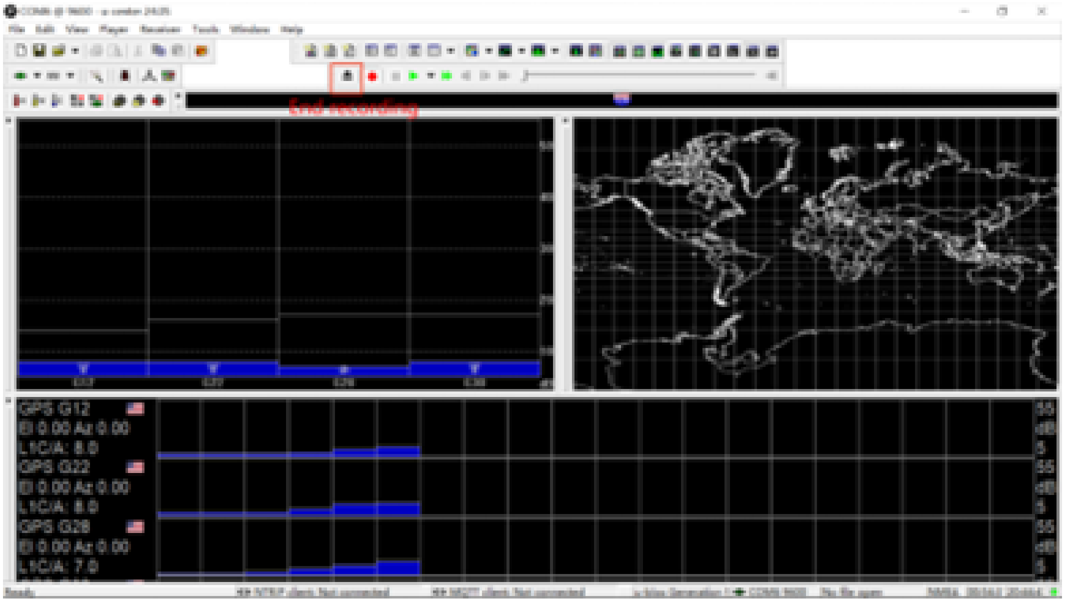


Figure 9: Illustration of the u-center software in the computer: stop recording.

4.2.3 Quick Product Configuration

u-center 2's quick product configuration feature allows users to define or apply GNSS product configurations for specific use cases. Saving, restoring, or sharing configurations between different products and users is easy, making it ideal for standardized testing scenarios.

4.2.4 Experimental Task: Static Data Collection

The primary goal is to collect a static dataset in a typical urban environment to analyze positioning performance.

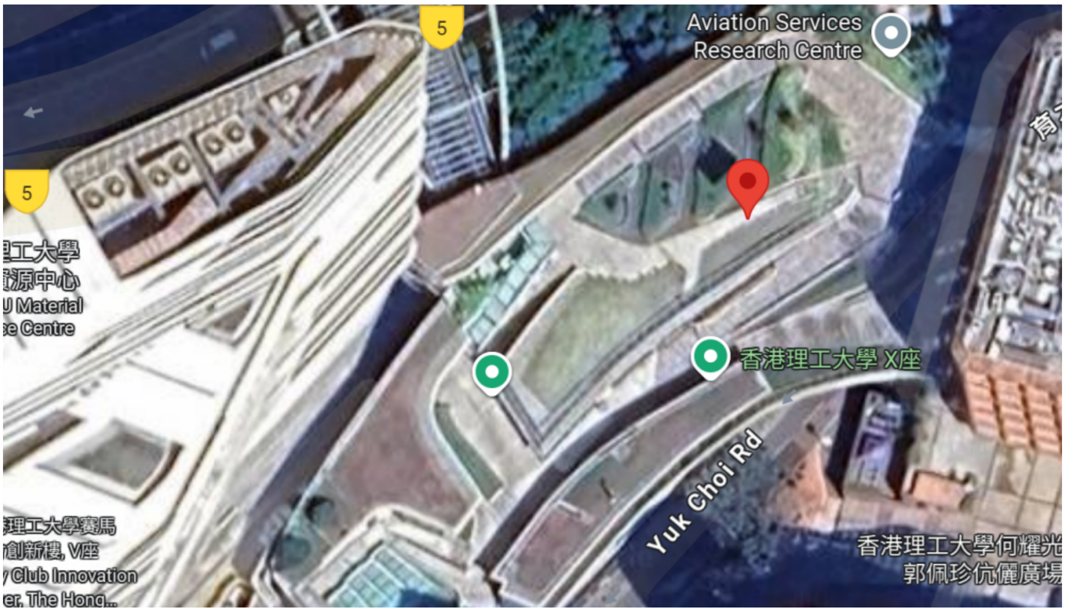


Figure 10: Collection location examples

Experimental Site: The Hong Kong Polytechnic University.

Figure 11: Collection environment examples

Experimental Environment: This location is a semi-open area surrounded by tall buildings, creating a challenging "urban canyon" environment where GNSS signals can be blocked or reflected (multipath).

Procedure:

1. Set up the GNSS receiver and antenna at the designated location.
2. Ensure the antenna has the best possible sky view.
3. Begin recording data using u-center as described above.
4. The software enables easy setup and evaluation of u-blox GNSS services such as AssistNow for improved time-to-first-fix.

4.3 Data Analysis with RTKLIB

RTKLIB is an open-source software package for standard and precise GNSS positioning.

Step 1: Install RTKLIB

Download the recommended version from the official source: RTKLIB Download.

Step 2: Launch RTKLIB

Unzip the files and run rtklaunch.exe. This opens a control panel for all RTKLIB tools.

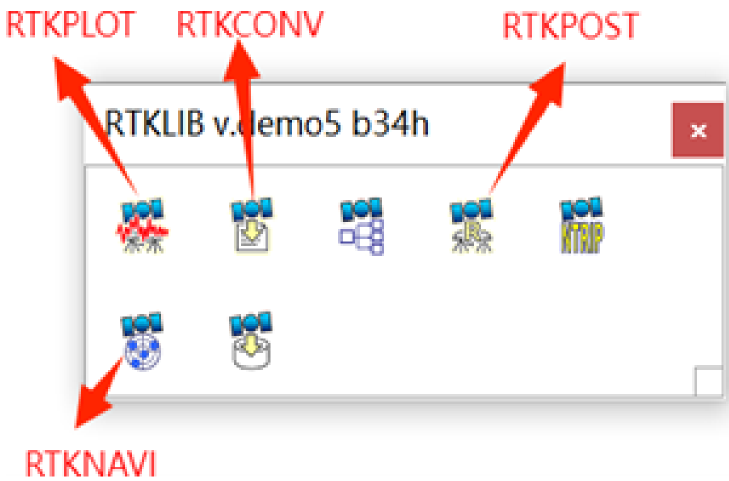


Figure 12: Illustration of the rtklaunch in the computer.

4.3.1 Convert Raw Data to RINEX

RINEX (Receiver Independent Exchange Format) is the standard format for raw GNSS data. We must convert our proprietary .ubx file.

Step 3: Open RTKCONV from the RTKLIB launcher.

- In the first field (labeled "RTCM, RCV RAW or RINEX OBS"), select your .ubx log file.
- Ensure the format is set to u-blox (UBX).
- In the output directories, make sure the .obs (observation) and .nav (navigation/ephemeris) files are checked.
- Click Convert. This will generate the necessary RINEX files.

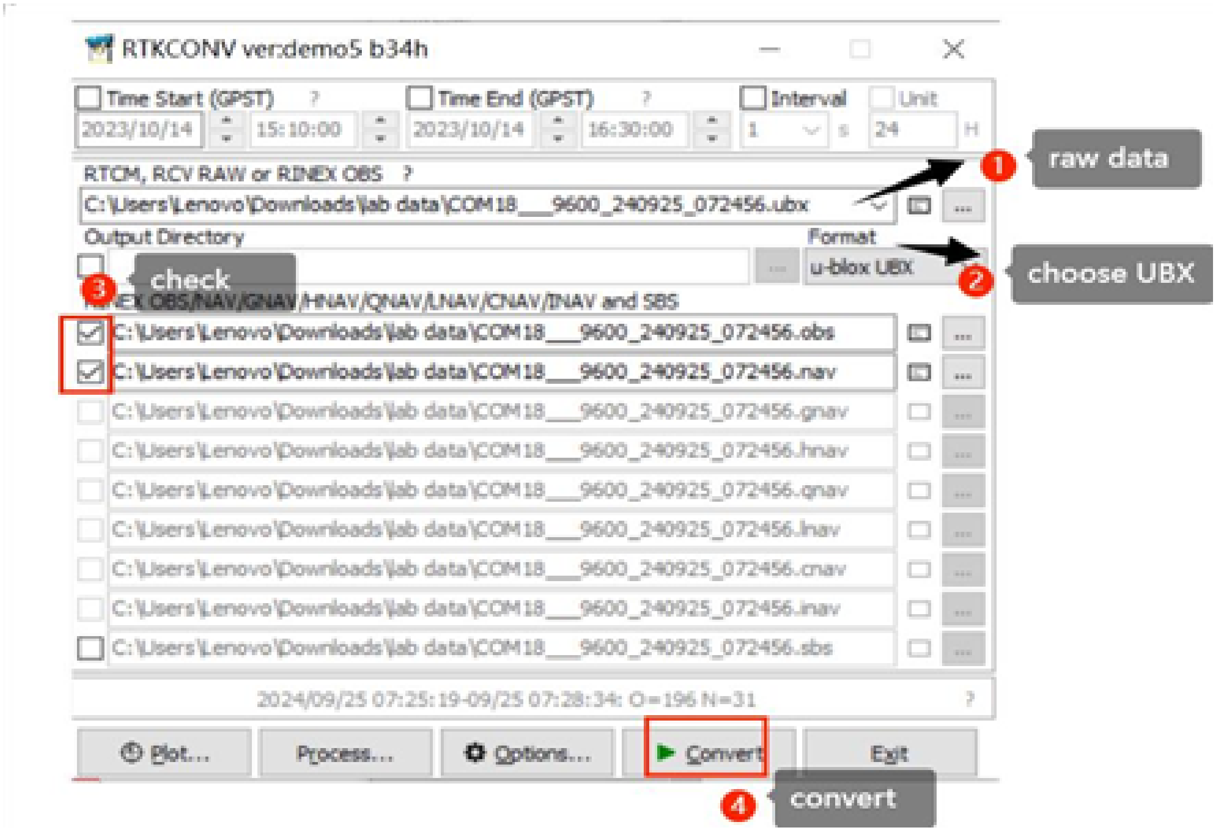


Figure 13: Illustration of the rtkconv software in the computer: convert .ubx to RINEX.

4.3.2 Download Precise Ephemeris Data

For better accuracy, we can use precise ephemeris data from a local reference station.

Step 4: Navigate to the Hong Kong Geodetic Survey Services website: HK SatRef RINEX Data.

Step 5: Select the reference station and set the data type as needed.

Step 1. Select the required stations

Hong Kong SatRef Station			
☐HKKS Kau Sai Chau	☐HKKT Kam Tin	☐HKLM Lamma Island	☐HKLT Lam Tei
☐HKMW Mui Wo	☐HKNP Ngong Ping	☐HKOH Obelisk Hill	☐HKPC Peng Chau
☑HKSC Stonecutters Island	☐HKSL Siu Lang Shui	☐HKSS Shap Sze Heung	☐HKST Sha Tin
☐HKTK Sha Tau Kok	☐HKWS Wong Shek	☐HKCL Chek Lap Kok	☐HKQT Quarry Bay
☐T430 Trig 430	☐HKFN Fanling	☐KYC1 Kau Yi Chau	TCHK* Tate's Cairn
Macao MoSRef Station			
☐COAL Coloane Alto	☐DSMG Taipa Grande	☐FOMO Monte Fortress	

*Note: 1. Satellite Reference Station summary has been replaced by Satellite Reference Station Coordinates List since 1 Apr, 2016.
2. Data of TCHK is not available

Step 2. Select Data Format

File Length	Interval	Data Format	
		RINEX 2.11	RINEX 3.02
1 Hour	1 Second	Not Select	Selected
	5 Seconds	Not Select	Not Select
24 Hours	5 Seconds	Not Select	Not Available
	30 Seconds	Not Select	Not Select

Figure 14: Illustration of step1 and step 2 of downloading ephemeris

Step 6: Open your .obs file (from Step 3) with a text editor and note the start and end times of your data collection.

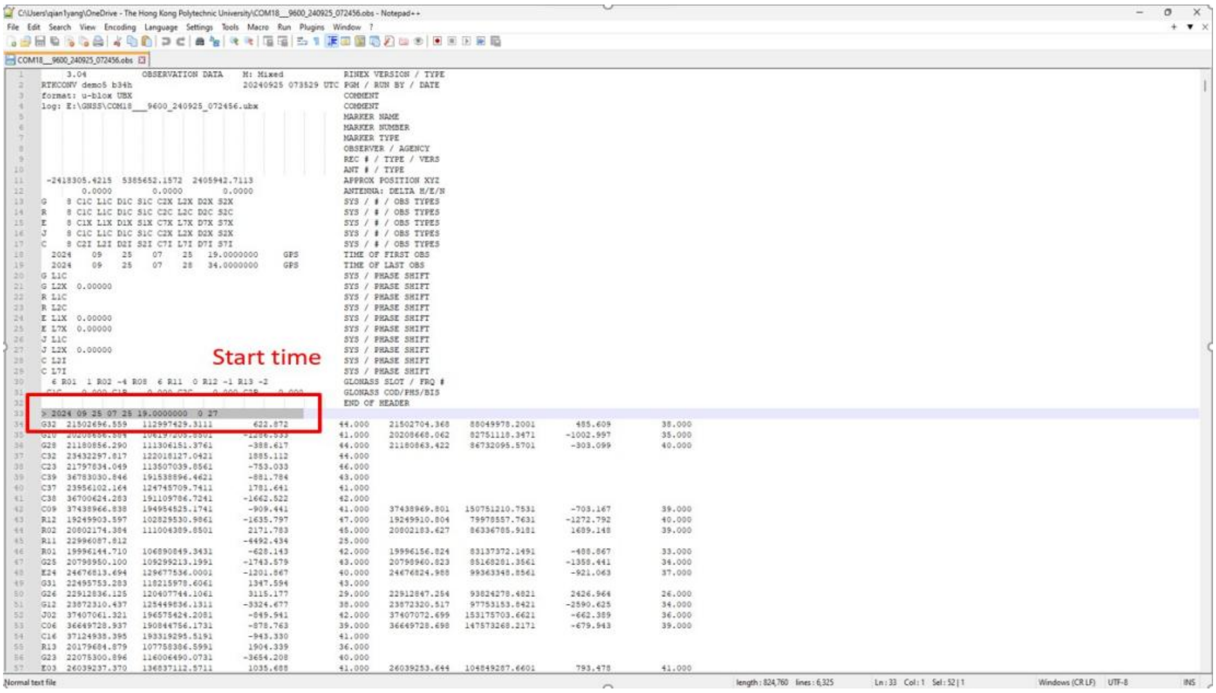


Figure 15: Illustration of the start time.

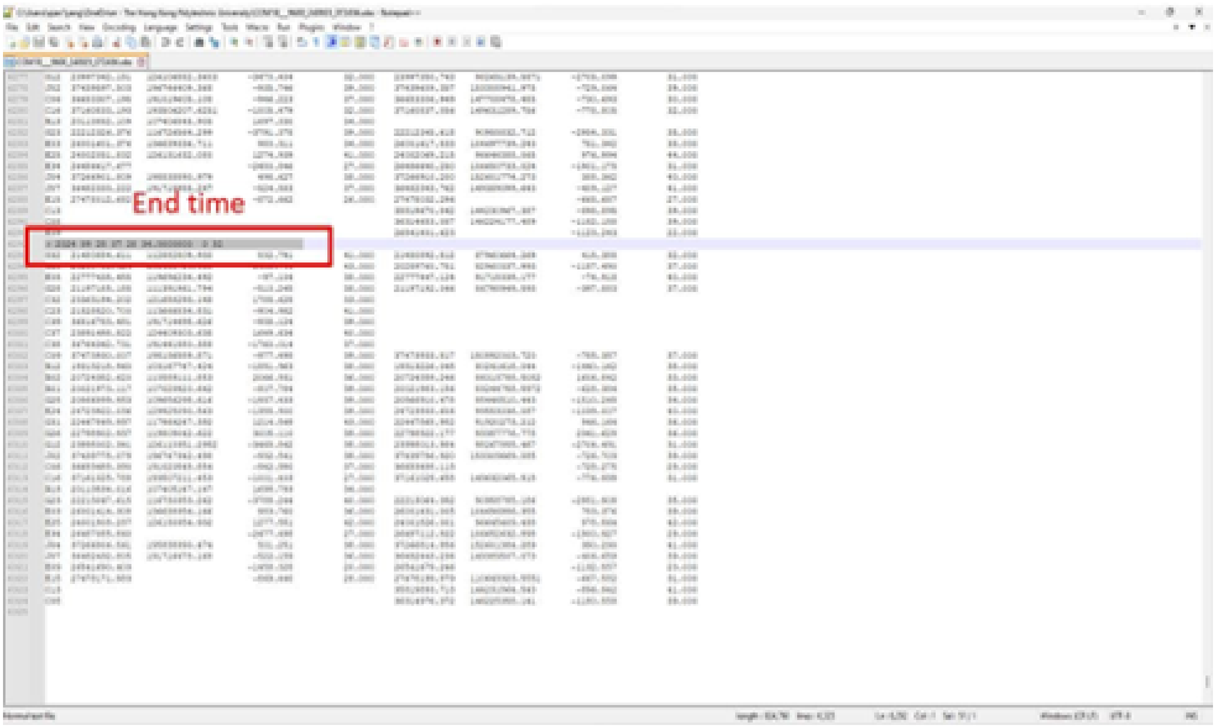


Figure 16: Illustration of the end time.

Step 7: On the website, enter the corresponding UTC start and end times for your data collection period and submit.

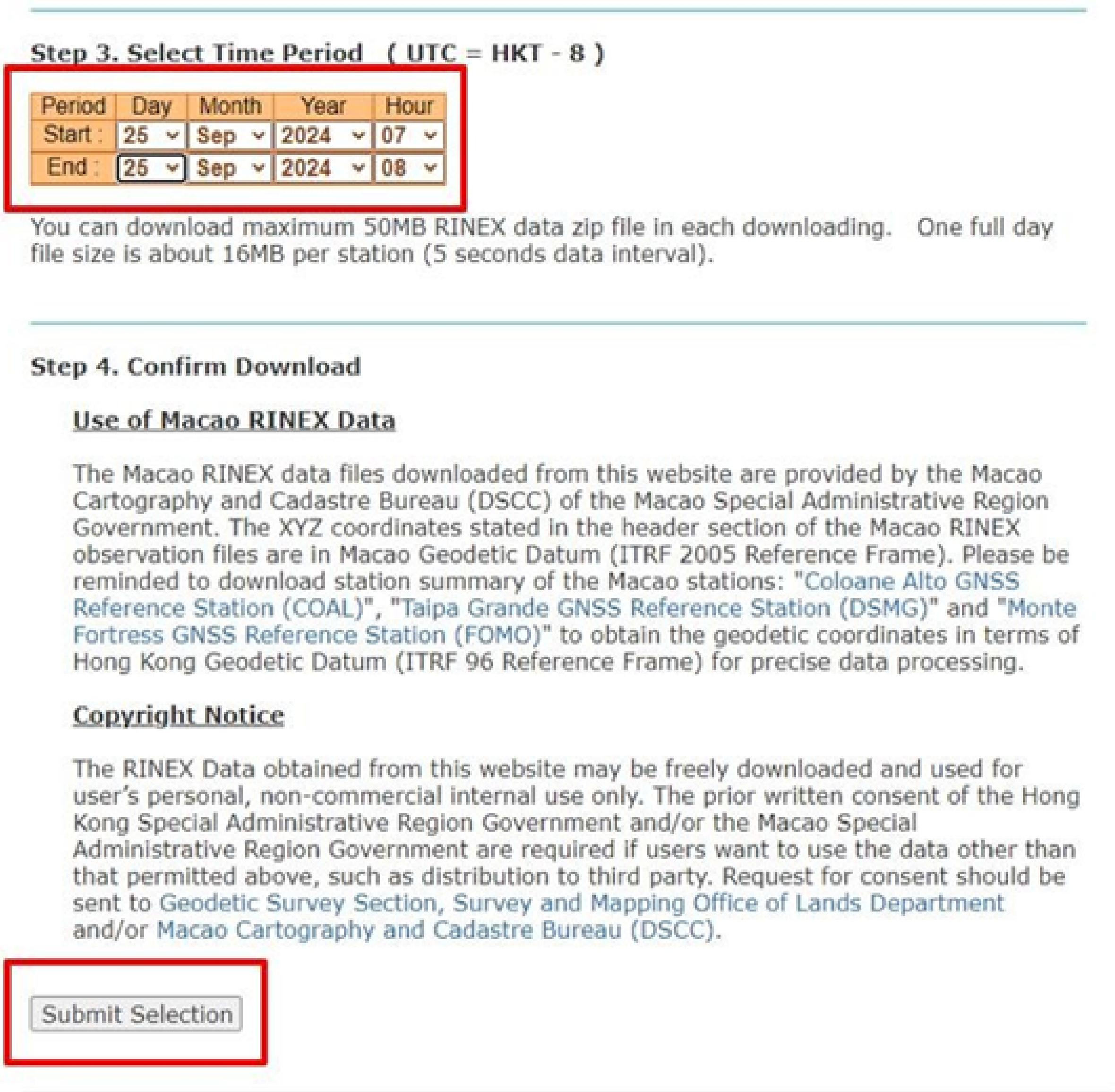


Figure 17: Illustration of step3 and step 4 of downloading ephemeris.

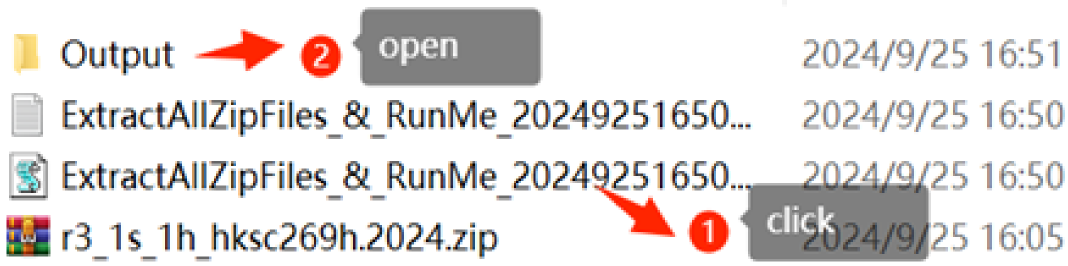


Figure 18: Illustration of how to get output folder.

Figure 19: Illustration of ephemeris.

4.3.3 Post-Process for a Position Solution

Now we will use RTKPOST to compute the position.

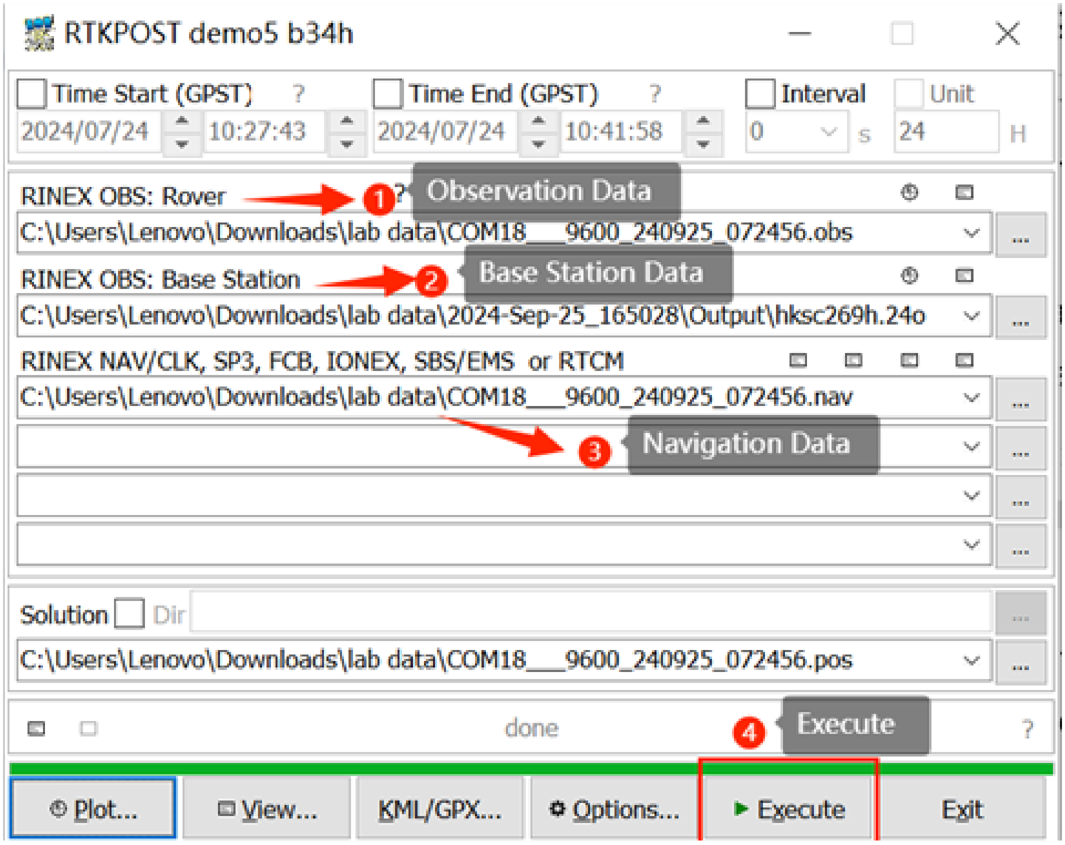


Figure 20: Illustration of the usage of rtkpost.

Step 10: Open RTKPOST from the RTKLIB launcher.

- **RINEX OBS (Rover):** Select your .obs file generated by RTKCONV.
- **RINEX OBS (Base Station):** Select the .24o (or similar) file downloaded from the HK SatRef website.
- **RINEX NAV/CLK:** Select the .nav file generated by RTKCONV.

4.3.4 View and Analyze Results

Step 11: Plot Results

In RTKPOST, click Plot to open RTKPLOT. This tool visualizes the results, showing the ground track, position deviation over time, satellite visibility, and more. For a static test, the ground track should be a tight cluster of points.

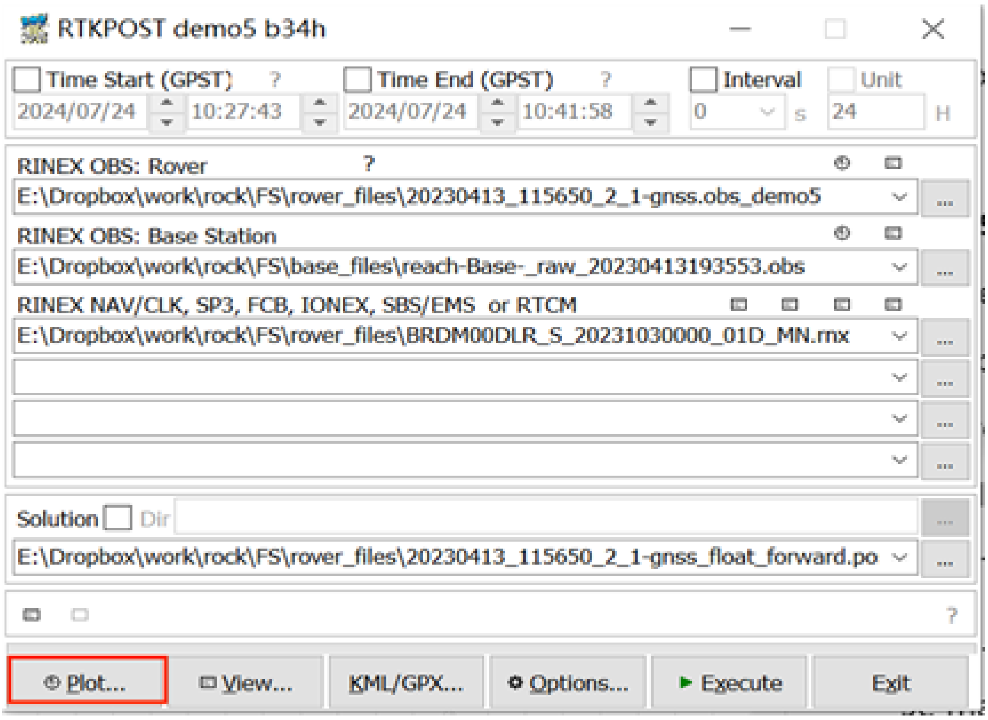


Figure 21: Illustration of plot button in rtkpost.

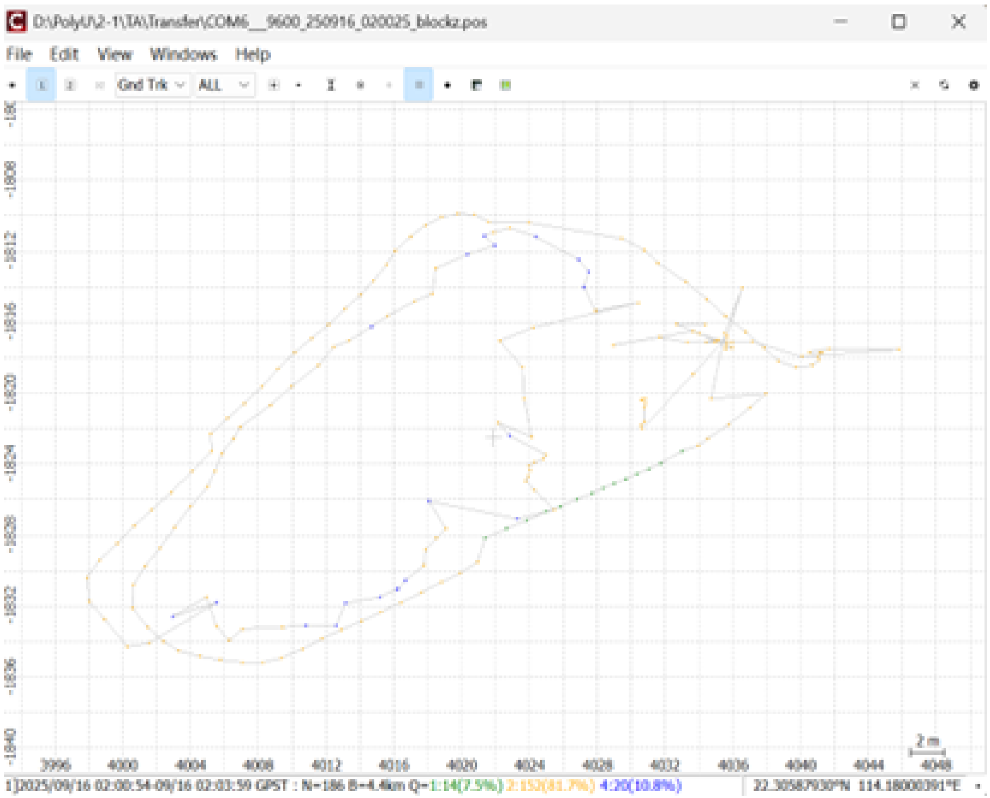


Figure 22: Illustration of the ground track.

4.3.5 Display results in Google Earth

Step 12: View in Google Earth

1. In RTKPOST, go to Tools → Convert to KML. Select your .pos file as input and convert it.
2. Open the Google Earth website.
3. In the menu, go to Projects → New Project → Import KML file from computer and upload your generated KML file to see your calculated track overlaid on the map.

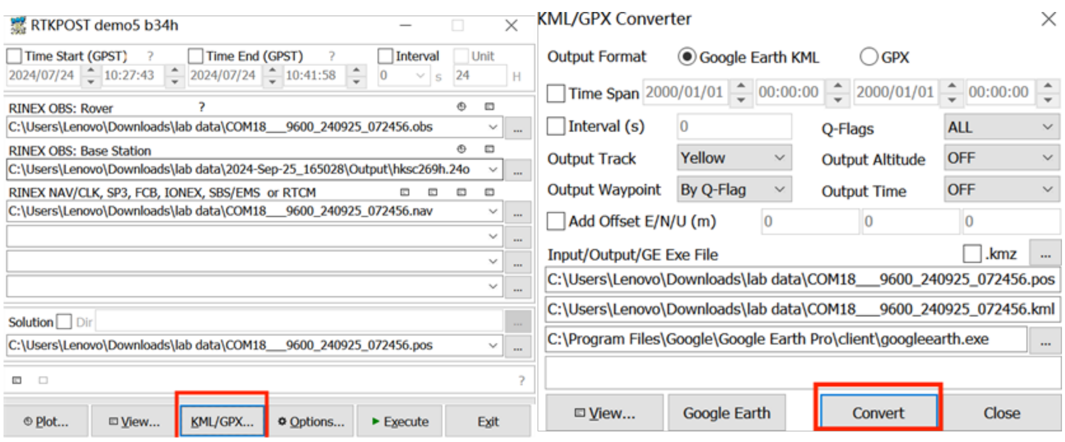


Figure 23: Illustration of rtkpost: convert .pos to KML.

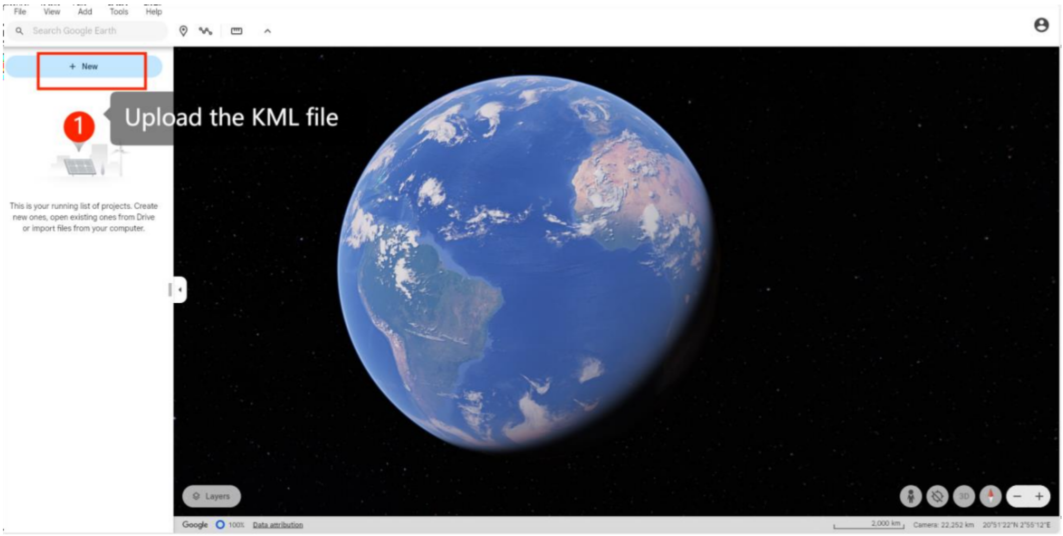


Figure 24: Illustration of google earth.

Figure 25: .kml generated on Google Map

5 RINEX Data Processing - MATLAB Processing

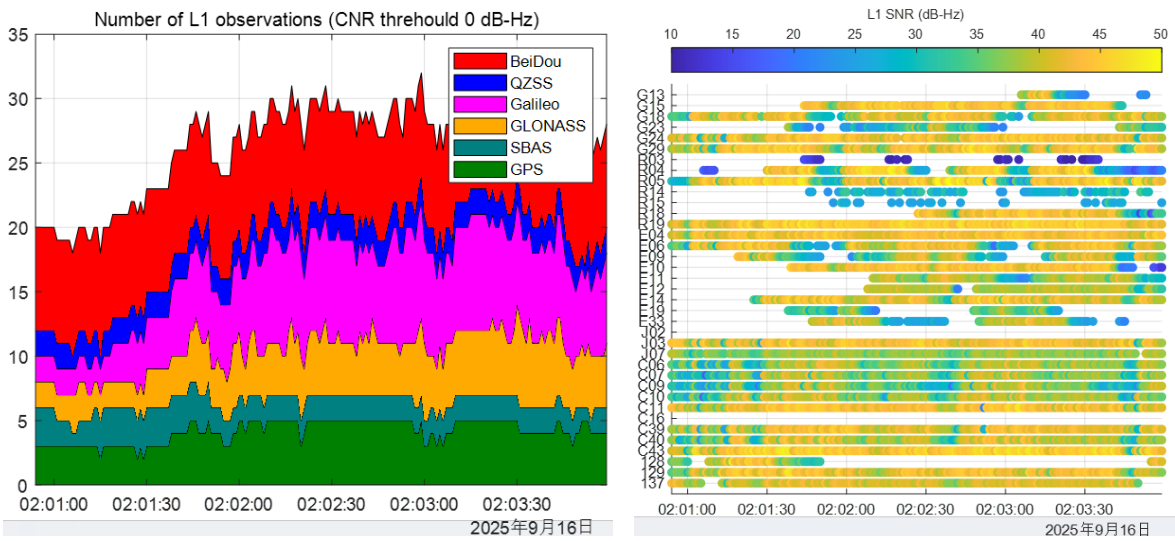
MATLAB RTKLIB provides a powerful framework for comprehensive GNSS data analysis. This section demonstrates the implementation of RTKLIB functions for processing RINEX observation and navigation files, followed by detailed signal quality and satellite geometry analysis.

5.1 RINEX Data Processing

The MATLAB implementation begins with reading RINEX observation and navigation files using RTKLIB functions.

5.2 Signal-to-Noise Ratio (SNR) Analysis

The L1 SNR analysis reveals the signal quality characteristics over the observation period. The SNR plot shows variations in signal strength ranging from 10 to 50 dB-Hz, with color-coded representation indicating signal quality levels.



(a).Number of L1 Observations (b).Received L1 SNR

Figure 26: MATLAB Multi-Constellation Satellite Visibility and SNR Analysis

The analysis shows:

- High SNR values (45-50 dB-Hz) indicate excellent signal reception
- Medium SNR values (30-40 dB-Hz) represent good signal quality
- Low SNR values (10-25 dB-Hz) suggest potential multipath or obstruction effects

5.3 Multi-Constellation Satellite Visibility

The satellite visibility analysis demonstrates the contribution of different GNSS constellations throughout the observation period:

Table 4: Multi-constellation satellite visibility statistics

Constellation	Max Satellites	Avg Contribution	Color Code
BeiDou	12-15	35%	Red
GPS	8-10	25%	Green
QZSS	2-3	15%	Blue
Galileo	4-6	15%	Magenta
GLONASS	3-5	10%	Orange

5.4 Sky Plot and Satellite Geometry

The sky plot visualization provides critical insights into satellite geometry and potential dilution of precision (DOP) effects. The plot shows satellite positions in polar coordinates with azimuth (0°-360°) and elevation angles.

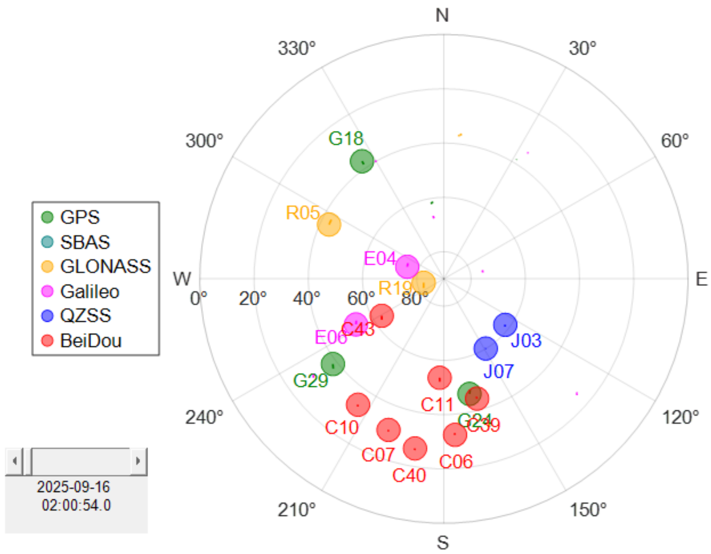


Figure 27: Sky Plot Visualization

Key observations include:

- Satellites distributed across different elevation angles from 20° to 80°
- Good azimuthal distribution ensuring geometric diversity
- BeiDou constellation provides dominant coverage in the Asian region

5.5 Data Export and Integration

The MATLAB analysis generates CSV files for further processing and integration with Python algorithms:

- pseudoranges_meas.csv - Observed pseudorange measurements of satellites
- satellite_clock_bias.csv - Computed satellite clock bias
- `satellite_positions.csv` - Computed satellite positions

- `ionospheric_delay.csv` - Modeled ionospheric corrections
- `tropospheric_delay.csv` - Modeled tropospheric corrections

6 Python WLSE-SPP Implementation

The Python implementation focuses on advanced positioning algorithms using Weighted Least Squares Estimation for Single Point Positioning. This approach provides robust position estimation with comprehensive error analysis and statistical validation.

6.1 Algorithm Implementation Overview

The WLSE-SPP algorithm processes GNSS observations through the following key components:

1. Pseudorange observation preprocessing
2. Satellite position computation from ephemeris data
3. Atmospheric error modeling (ionospheric and tropospheric delays)
4. Weighted least squares position estimation
5. Statistical validation and accuracy assessment

6.2 Trajectory Analysis Results

The trajectory analysis demonstrates the algorithm's performance in both geographic and local coordinate systems:

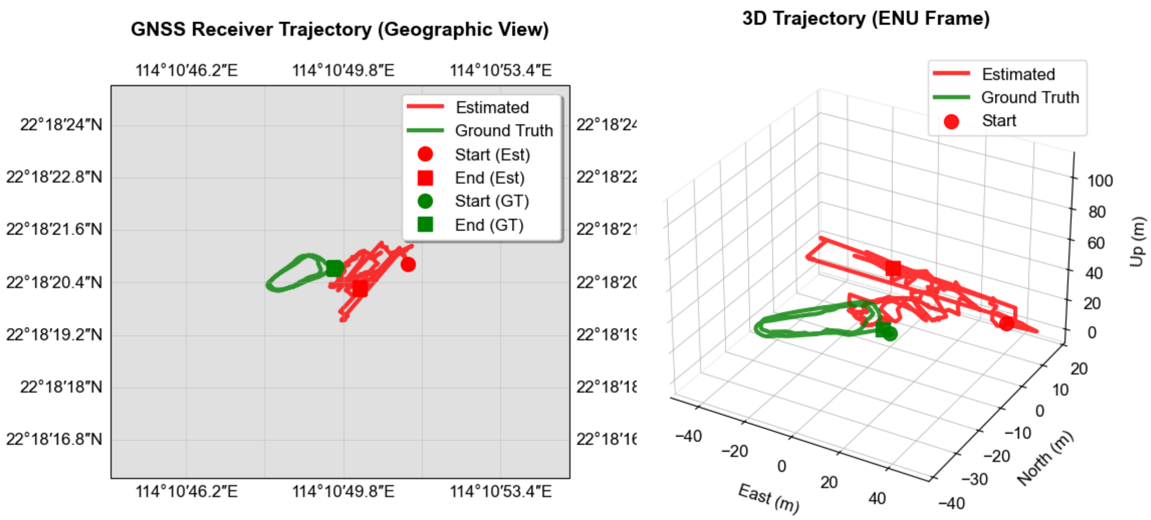


Figure 28: Python WLSE-SPP Trajectory Analysis (Geographic View and 3D ENU Frame)

6.2.1 Geographic Trajectory Visualization

The geographic view presents the estimated trajectory compared to ground truth data over the coordinate range:

- Latitude Range: 22.305614° to 22.305839° (approximately 55 meters)
- Longitude Range: 114.180019° to 114.180466° (approximately 45 meters)
- Altitude Range: 53.697 m to 135.184 m

6.2.2 3D ENU (East-North-Up) Frame Analysis

The 3D trajectory analysis in the ENU coordinate system reveals the positioning behavior in local coordinates, with distinct start and end points clearly marked. The trajectory shows convergence behavior typical of static positioning scenarios.

6.3 Comprehensive Error Analysis

The error analysis provides detailed insights into positioning performance across different dimensions and temporal characteristics.

6.3.1 Directional Positioning Errors

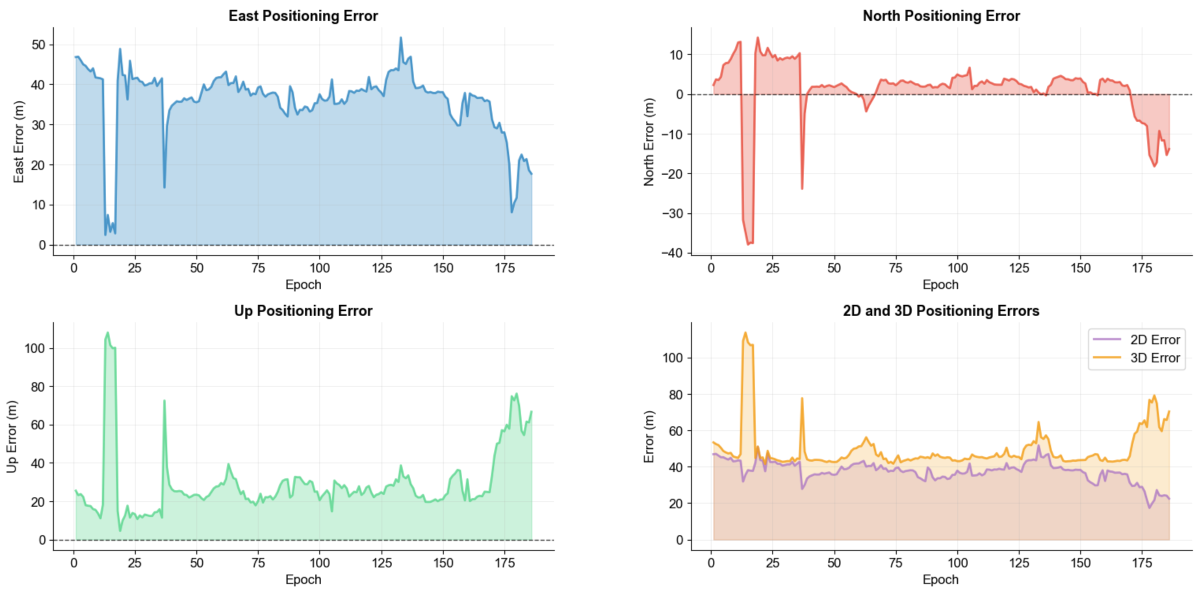


Figure 29: Directional Positioning Errors Analysis (East, North, Up, and 2D/3D Errors)

Individual error components demonstrate different characteristics:

Table 5: Directional error characteristics

Error Component	Characteristics	Typical Range
East Error	Decreasing trend over epochs	5-50 meters
North Error	Relatively stable with occasional excursions	-30-10 meters
Up Error	High variability, characteristic altitude challenges	0-100 meters

6.3.2 Statistical Error Distribution Analysis

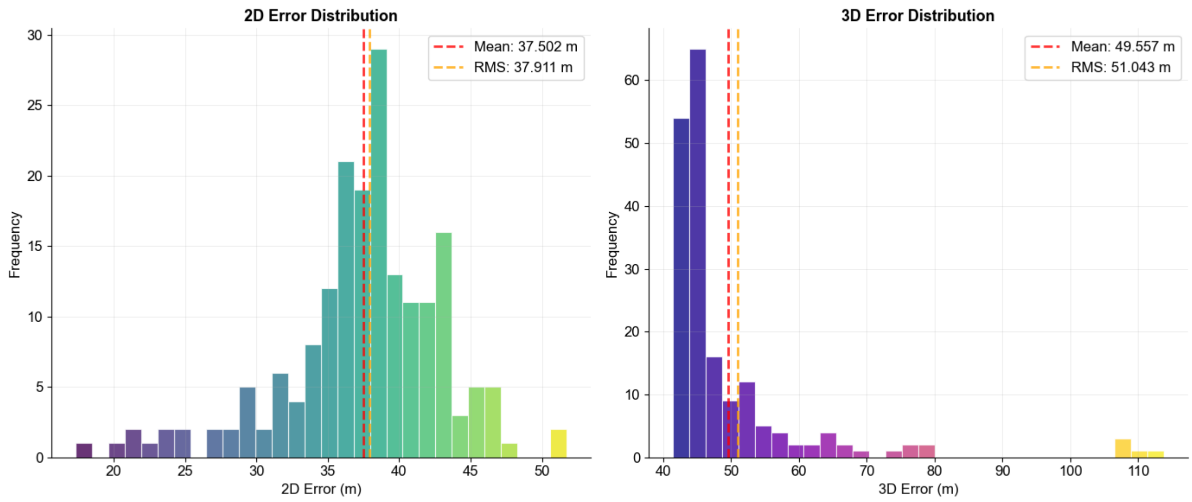


Figure 30: Statistical Error Distribution Analysis (2D and 3D Error Histograms)

The comprehensive statistical analysis reveals important positioning characteristics:

Table 6: Statistical error summary

Error Type	Mean (m)	RMS (m)
2D Error	37.502	37.911
3D Error	49.557	51.043

6.4 Error Evolution and Convergence Analysis

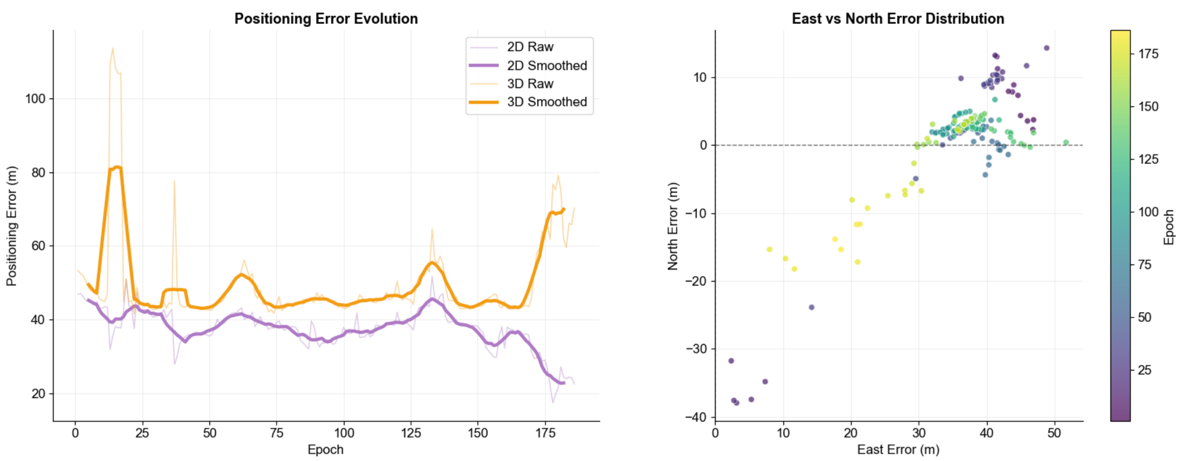


Figure 31: Error Evolution Analysis and East vs North Error Distribution

The positioning error evolution analysis demonstrates the convergence characteristics of the WLSE-SPP algorithm:

- Initial convergence period showing higher errors in the first 25 epochs
- Stabilization phase from epochs 25-150 with consistent performance
- Final degradation period potentially due to satellite geometry changes

6.5 Comprehensive Positioning Accuracy Statistics

The final analysis summary provides complete positioning accuracy statistics for the 186 processed epochs:

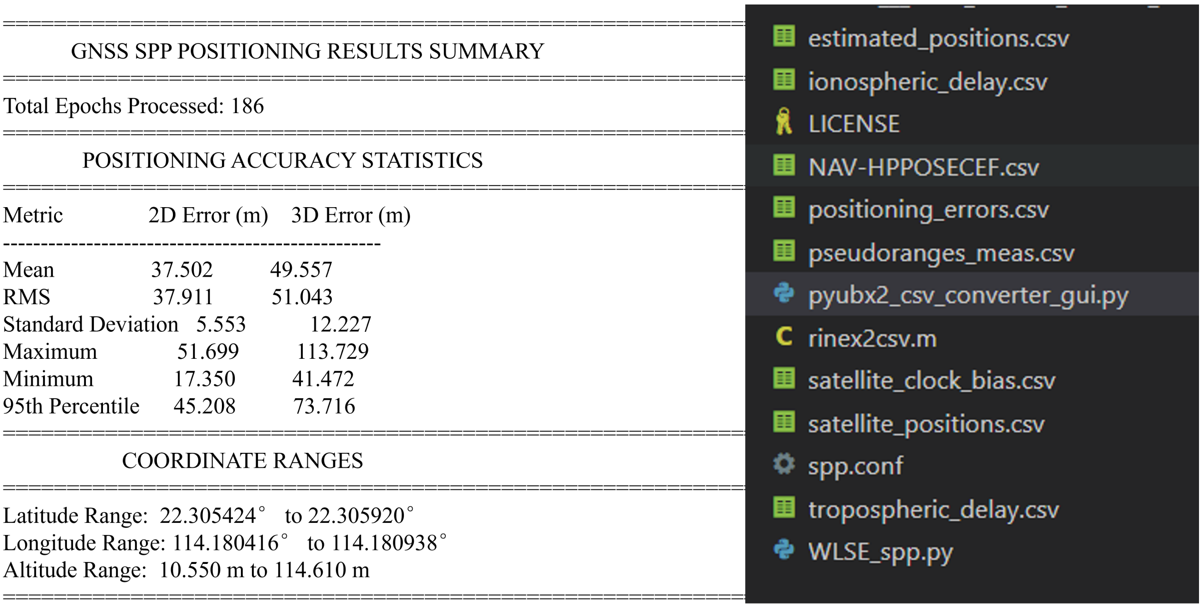


Figure 32: GNSS SPP Positioning Results Summary and Generated Output Files

Table 7: Comprehensive positioning accuracy statistics

Metric	**2D Error (m)**	**3D Error (m)**
Mean	37.502	49.557
RMS	37.911	51.043
Standard Deviation	5.553	12.227
Maximum	51.699	113.729
Minimum	17.350	41.472
95th Percentile	45.208	73.716

6.6 Output Data Generation

The Python WLSE-SPP implementation generates comprehensive output files for further analysis:

- `estimated_positions.csv` - Final position estimates for each epoch
- `positioning_errors.csv` - Detailed error analysis by epoch

7 Conclusions

7.1 Key Learning Outcomes

Through this comprehensive laboratory exercise, students achieve the following learning outcomes:

1. Understanding of multi-constellation GNSS signal characteristics and quality assessment
2. Practical experience with professional GNSS processing software (u-center) and custom algorithm implementation
3. Comprehensive error analysis techniques and statistical validation methods
4. Comparative evaluation of different positioning approaches and their trade-offs
5. Data integration and visualization skills across multiple software platforms

7.2 Recommendations for Future Work

Based on the analysis results, several areas for improvement and extension are identified:

- Implementation of advanced multipath mitigation techniques
- Integration of precise orbit and clock corrections
- Development of real-time positioning capabilities
- Extended urban canyon environment testing
- Integration with u-blox GNSS services such as AssistNow for improved performance

8 References

References

[1] T. Takasu, *RTKLIB ver.2.4.2 Manual*, 2013. Official RTKLIB documentation (SPP/RTK algorithms and configuration guidelines). https://www.rtklib.com/rtklib_document.htm.

[2] T. Takasu, *RTKLIB — Open Source Program Package for GNSS Positioning*, GitHub repository, https://github.com/tomojitakasu/RTKLIB.

[3] E. D. Kaplan and C. J. Hegarty (eds.), *Understanding GPS/GNSS: Principles and Applications*, 3rd ed., Artech House, 2017.

[4] European Space Agency (ESA), *GNSS Data Processing, Volume I: Fundamentals and Algorithms*, ESA TM-23/1, 2013. Available via ESA Navipedia.

- **Additional Resources:**
  RTKLIB 2.5.0: Download Link
  MATLAB:Download Link.