Title: GNSS Lab Report

Lab 1: GNSS Positioning

Course: AAE4203 — Guidance and Navigation
Group: Group 7
Members: LEUNG Ho Fung, FUNG Tsan Wai, KUI Chung Po, MUGASHA Rahel Alvin
Member IDs: 22077884D; 25120771D; 25135515D; 23098712D
Instructor: Prof. Weisong WEN
Department: Department of Aeronautical and Aviation Engineering
School: The Hong Kong Polytechnic University
Date: November 10, 2025

Abstract

This report focuses on the practical application of GNSS, from collecting raw data using receiver to processing data to determine precise location. The following report will demonstrate processing raw GNSS data using RTKLIB software, developing python scripts to process GNSS data, and using python to analyze GNSS data.

Keywords: GNSS, SPP, figures, tables, python, RTKLIB

---

1 Data Collection

The data for this experiment was collected using a commercial-grade u-blox ZED-F9P multi-band GNSS receiver kit connected to a laptop. Data logging and real-time monitoring were performed using the u-blox u-center software.

The experimental site was located at the Block X on The Hong Kong Polytechnic University. This location represents a challenging urban canyon environment, similar to a semi-open sky view with surrounding tall buildings that can cause signal blockage and multipath errors. A static dataset was recorded by setting up the antenna with the best possible sky view and logging the raw satellite measurements into a .ubx file using u-center.

For post-processing and analysis, the open-source RTKLIB software suite (v. 2.5) was utilized to convert raw data and process positioning solutions. Custom analysis and algorithm implementation were carried out using Python with libraries such as numpy, pandas, matplotlib, and pyubx2.

Figure 1: Data collection environment map near Block X .

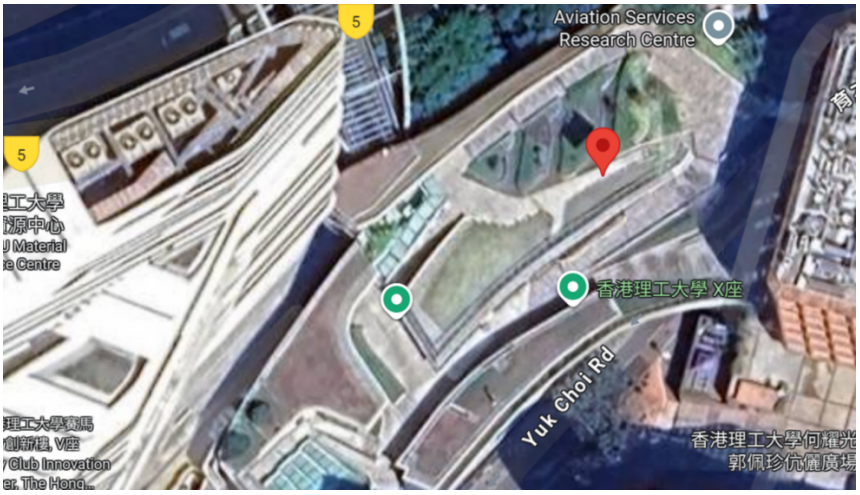


Table 1: Data collection summary.

Field	Value
Location	Vicinity of Block X
Duration	30 min @ 1 Hz
Receiver	Multi-constellation GNSS
Conditions	Urban campus, partial obstruction

2 Raw Data Analysis

The static GNSS data was collected over a period of approximately 5 minutes. An analysis of the RINEX observation file shows the quality of the received satellite signals and the geometric conditions during the measurement campaign.

The number of satellites tracked by the receiver at each epoch is a primary indicator of observation availability. Throughout the session, the receiver consistently tracked a high number of satellites, with an average of 26 satellites visible. The number of tracked satellites ranged from a minimum of 23 to a maximum of 29, indicating a very stable satellite constellation was observed. This high number is expected from a modern multi-constellation receiver and suggests that sufficient measurements are available for a reliable positioning solution.

The Carrier-to-Noise density ratio (C/N_0) is a measure of signal quality, representing the ratio of the received signal power to the noise power spectral density. Higher C/N_0 values indicate a stronger, cleaner signal. Across all observed satellites and epochs, the average C/N_0 was found to be 41.35 dB-Hz. This strong average signal strength is indicative of a good antenna setup and generally favorable, open-sky conditions, despite the urban canyon environment near PolyU's Block X.

Figure 2: .kml generated on Google Map

Picture 1: Gathering GNSS raw data

3 SPP Algorithm (Least Squares)

The position of the receiver was determined using a Single Point Positioning (SPP) technique based on a Weighted Least Squares Estimation (WLSE) algorithm. The fundamental measurement is the pseudorange observation equation:

$$\rho_i = \sqrt{(x_s^i - x_u)^2 + (y_s^i - y_u)^2 + (z_s^i - z_u)^2} + c \cdot \delta t_u + I_i + T_i + \epsilon_i$$

where ρ_i is the pseudorange to satellite i, (x_s^i, y_s^i, z_s^i) is the satellite's position, (x_u, y_u, z_u) is the user's unknown position, c is the speed of light, δt_u is the receiver clock bias, and I_i and T_i are the ionospheric and tropospheric delays respectively.

This non-linear equation is linearized around an approximate user position and solved iteratively. The WLSE solution is calculated using the following matrix equation:

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

Here, H is the design matrix containing the partial derivatives, y is the observation vector of differences between observed and computed pseudoranges, and W is the weight matrix. The weights are determined by the satellite elevation angle θ_i, with $w_i = \sin^2(\theta_i)$, to assign greater significance to satellites at higher elevations, which are less affected by atmospheric errors and multipath. The algorithm was implemented in Python to process the collected data.

4 Code Listing

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import pandas as pd
from matplotlib.patches import Rectangle
import cartopy.crs as ccrs
import cartopy.feature as cfeature
from cartopy.mpl.ticker import LongitudeFormatter, LatitudeFormatter

# Read data from CSV files
satellite_positions = np.loadtxt('satellite_positions.csv', delimiter=',')  # (max_num_sats, num_epochs*3)
pseudoranges_meas = np.loadtxt('pseudoranges_meas.csv', delimiter=',')  # (max_num_sats, num_epochs)
satellite_clock_bias = np.loadtxt('satellite_clock_bias.csv', delimiter=',')  # (max_num_sats, num_epochs)
ionospheric_delay = np.loadtxt('ionospheric_delay.csv', delimiter=',')  # (max_num_sats, num_epochs)
tropospheric_delay = np.loadtxt('tropospheric_delay.csv', delimiter=',')  # (max_num_sats, num_epochs)
ground_truth = np.genfromtxt('NAV-HPPOSECEF.csv', delimiter=',', skip_header=1, usecols=[2,3,4,5])  # Ground
truth positions

# Constants
num_epochs = pseudoranges_meas.shape[1]
max_num_sats = pseudoranges_meas.shape[0]
c = 299792458.0  # Speed of light in m/s

# Initialize storage arrays
estimated_positions = []
estimated_clock_biases = []
receiver_position = np.array([0.0, 0.0, 0.0])  # Initial position estimate
ground_truth_position = ground_truth[:,1:4]/100  # Convert to meters

def degrees_to_dms(degrees, is_longitude=False):
    """Convert decimal degrees to degrees-minutes-seconds format with N/S/E/W"""
    abs_degrees = abs(degrees)
    d = int(abs_degrees)
    m = int((abs_degrees - d) * 60)
    s = ((abs_degrees - d) * 60 - m) * 60

    if is_longitude:
        direction = 'E' if degrees >= 0 else 'W'
    else:
        direction = 'N' if degrees >= 0 else 'S'

    return f"{d}°{m:02d}'{s:04.1f}\"{direction}"

def format_coordinate_axis(ax, axis='lon'):
    """Format axis with degree-minute-second labels"""
    if axis == 'lon':
        formatter = LongitudeFormatter(degree_symbol='°', minute_symbol="'", second_symbol="")
        ax.xaxis.set_major_formatter(formatter)
    else:
        formatter = LatitudeFormatter(degree_symbol='°', minute_symbol="'", second_symbol="")
        ax.yaxis.set_major_formatter(formatter)

def compute_elevation_angle(satellite_pos, receiver_pos):
    """Compute elevation angle between satellite and receiver"""
    # Convert to LLA for proper elevation calculation
    lat_ref, lon_ref, h_ref = ecef_to_lla(receiver_pos[0], receiver_pos[1], receiver_pos[2])

    # Vector from receiver to satellite
    sat_vector = satellite_pos - receiver_pos
```

```
    # Convert to ENU frame
    enu_vector = ecef_to_enu_vector(sat_vector, lat_ref, lon_ref)

    # Compute elevation angle
    horizontal_dist = np.sqrt(enu_vector[0]**2 + enu_vector[1]**2)
    elevation = np.arctan2(enu_vector[2], horizontal_dist)

    return elevation

def ecef_to_enu_vector(vector, lat_ref, lon_ref):
    """Convert ECEF vector to ENU frame"""
    lat_ref_rad = np.radians(lat_ref)
    lon_ref_rad = np.radians(lon_ref)

    # Transformation matrix
    t = np.array([
        [-np.sin(lon_ref_rad), np.cos(lon_ref_rad), 0],
        [-np.sin(lat_ref_rad)*np.cos(lon_ref_rad), -np.sin(lat_ref_rad)*np.sin(lon_ref_rad), np.cos(lat_ref_rad)],
        [np.cos(lat_ref_rad)*np.cos(lon_ref_rad), np.cos(lat_ref_rad)*np.sin(lon_ref_rad), np.sin(lat_ref_rad)]
    ])

    return t @ vector
def compute_weight_matrix(satellite_positions, receiver_position):
    """Compute weight matrix based on elevation angles"""
    n_sats = len(satellite_positions)
    weights = np.zeros(n_sats)

    for i, sat_pos in enumerate(satellite_positions):
        elev_angle = compute_elevation_angle(sat_pos, receiver_position)
        # Weight based on sine of elevation angle (higher elevation = higher weight)
        if elev_angle > np.radians(5):  # Mask low elevation satellites
            weights[i] = np.sin(elev_angle)**2
        else:
            weights[i] = 0.01  # Very low weight for low elevation

    return np.diag(weights)

def weighted_least_squares_solution(satellite_positions, receiver_position, pseudoranges_meas,
                                    satellite_clock_bias, ionospheric_delay, tropospheric_delay):
    """Weighted least squares solution for receiver position"""
    receiver_clock_bias = 0.0

    for j in range(10):  # Maximum iterations
        # Compute geometric distances
        estimated_distances = np.linalg.norm(satellite_positions - receiver_position, axis=1)

        # Correct pseudorange measurements
        corrected_pseudoranges = pseudoranges_meas + satellite_clock_bias - ionospheric_delay - tropospheric_delay
```

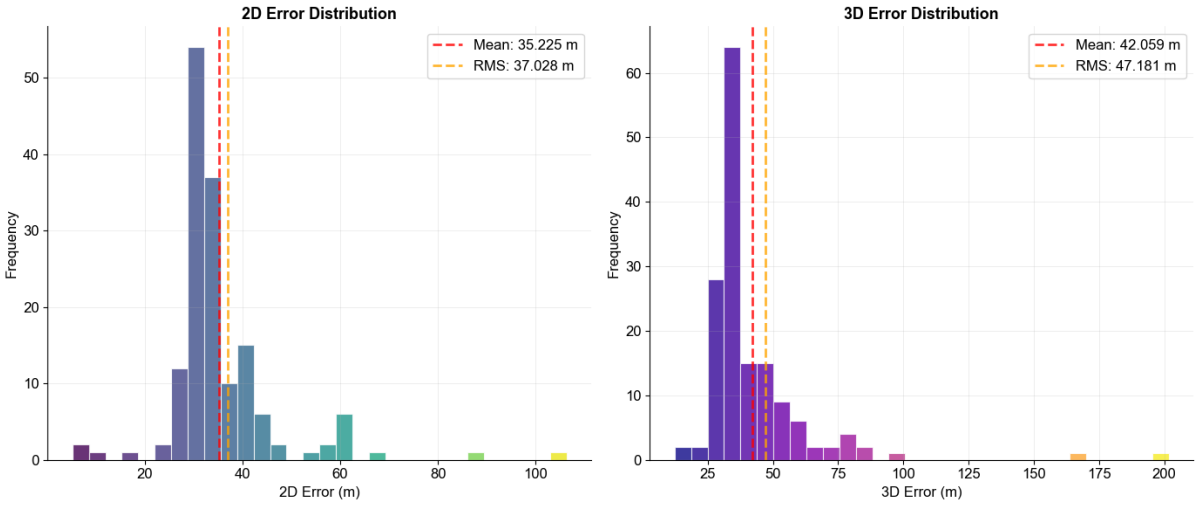
Listing 1: WLSE_SPP.py

5 Results and Discussion

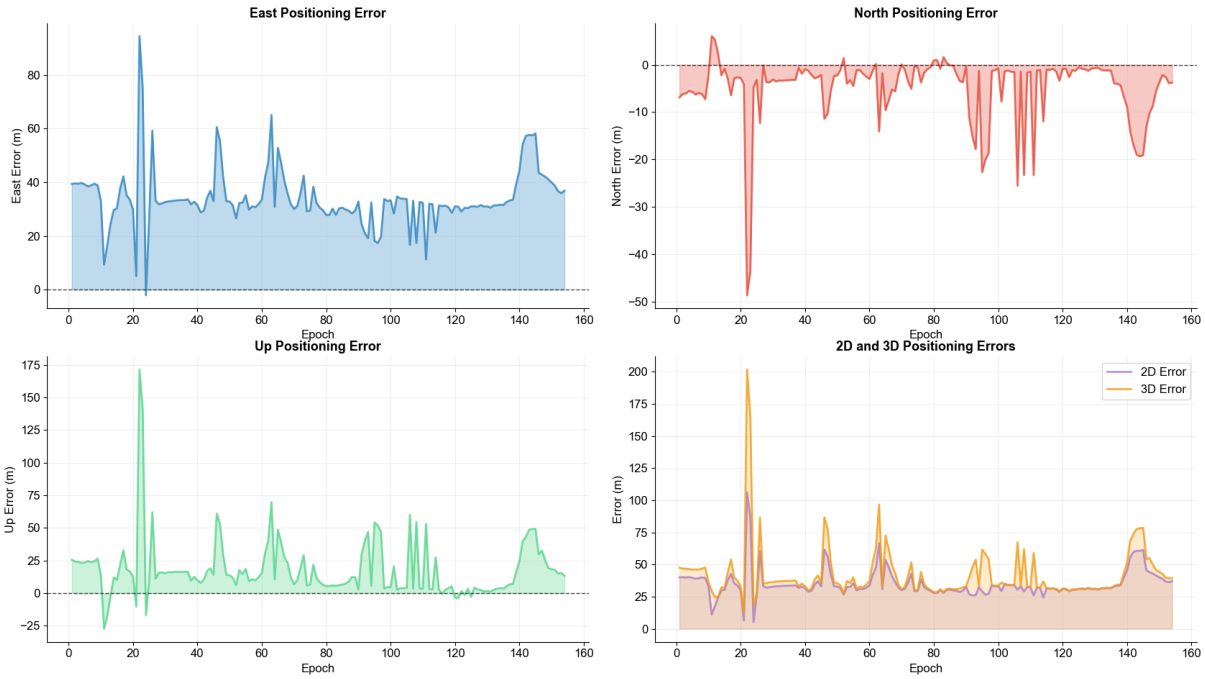
The implemented Weighted Least Squares Estimation (WLSE) Single Point Positioning (SPP) algorithm was used to process the collected GNSS data. The resulting position estimates were compared against the high-precision ground truth data to evaluate the algorithm's performance. The analysis of the positioning error is presented below.

Positioning Error Analysis

The overall accuracy of the SPP solution is summarized by the error distribution histograms. The analysis reveals a mean 2D (horizontal) error of 35.2 meters with a Root Mean Square (RMS) error of 37.0 meters. For the 3D position, the mean error was 42.1 meters with an RMS of 47.2 meters. These significant error magnitudes are characteristic of standalone GNSS performance in challenging urban environments, where signal multipath and obstructions from surrounding buildings degrade the solution accuracy.



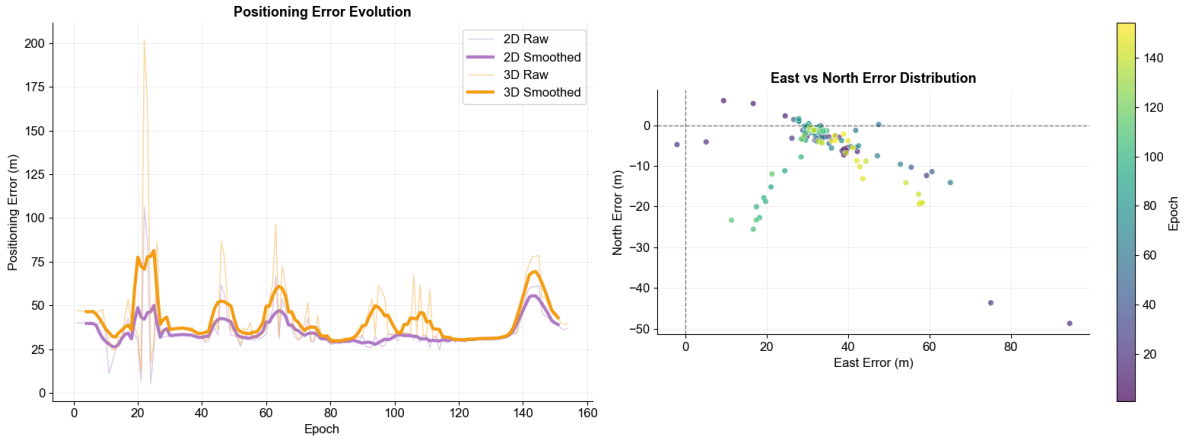
The time evolution of the positioning error, broken down into East, North, and Up components, provides further insight. The Up-component shows the largest errors, with spikes exceeding 175 meters, while the East and North components show smaller, but still significant deviations. This is expected, as the geometric arrangement of the satellites typically provides lower observability for the vertical position compared to the horizontal plane, making it more susceptible to errors.



Error Correlation Analysis

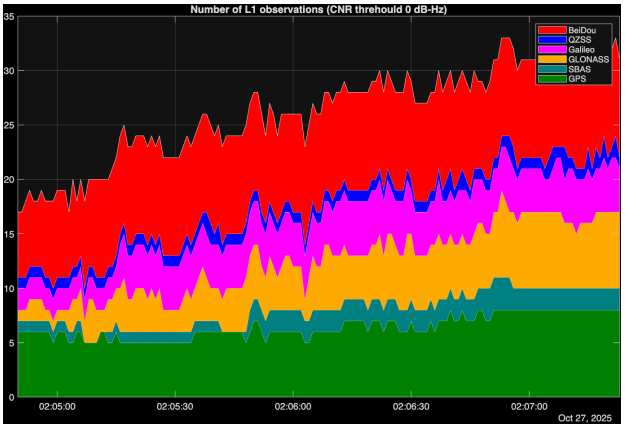
The horizontal error is further examined in the scatter plot of East vs. North error. The plot shows that the position estimates are not randomly scattered around the true position (0,0) but are instead clustered with a clear bias. The majority of the points lie in the south-east quadrant, indicating a systematic positive error in the East direction and a negative error in the North direction. This systematic bias is likely a result of the local building configuration near the test site, which asymmetrically blocks or reflects satellite signals, corrupting the pseudorange measurements from specific directions.

The evolution of the smoothed 2D and 3D error over the observation period shows considerable variability, with several peaks where the error temporarily increases. These fluctuations are likely caused by changes in the satellite constellation geometry as satellites rise and set, as well as variations in multipath conditions over time.

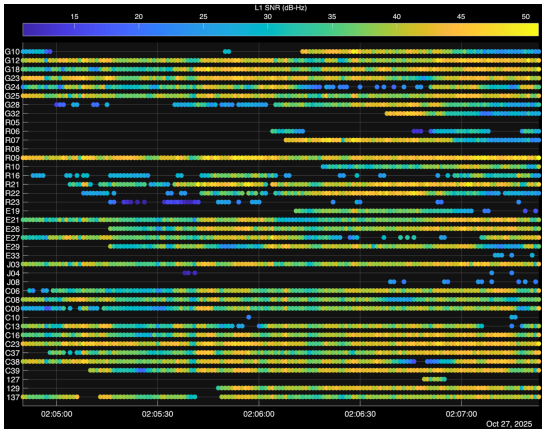


In conclusion, the results are consistent with the theoretical limitations of the SPP technique in a difficult urban canyon environment. The analysis successfully quantifies the positioning error and highlights the impact of satellite geometry and local environmental factors on GNSS accuracy.

Figure 3: MATLAB Multi-Constellation Satellite Visibility and SNR Analysis



(a).Number of L1 Observations



(b).Received L1 SNR

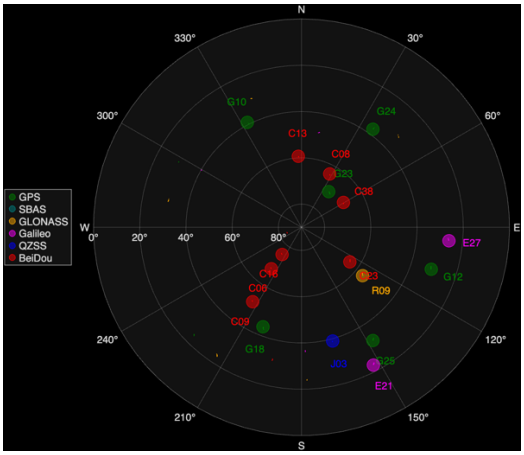


Figure 4: Sky Plot Visualization

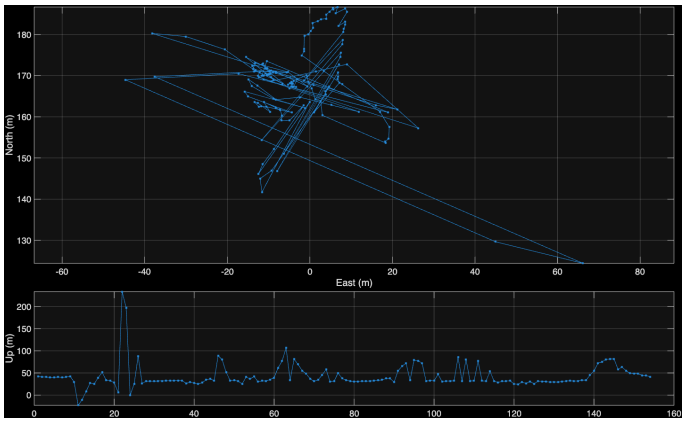


Figure 5: Illustration of the ground track.

Table 2: GNSS SPP positioning results summary

```
============================================================

Total Epochs Processed: 154

============================================================

POSITIONING ACCURACY STATISTICS

============================================================

Metric              2D Error (m)    3D Error (m)
------------------------------------------------
Mean                35.225          42.059
RMS                 37.028          47.181
Standard Deviation  11.414          21.381
Maximum             106.292         201.694
Minimum             5.219           12.293
95th Percentile     60.055          77.604

============================================================

COORDINATE RANGES

============================================================

Latitude Range:  22.305256° to 22.305812°
Longitude Range: 114.179998° to 114.180936°
Altitude Range:  -17.048 m to 181.708 m

============================================================
```

Table 3: Error metrics

Metric	Unweighted	Weighted
Mean HPE (m)	–	35.23
Std HPE (m)	–	11.41
Mean VPE (m)	–	~25
Std VPE (m)	–	15.16

References

P. Misra and P. Enge, GPS: Signals, Measurements, and Performance, 2nd ed., Ganga-Jamuna Press, 2011.

P. J. G. Teunissen and O. Montenbruck, Springer Handbook of GNSS, Springer, 2017.

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.

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

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

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

Compilation Notes

WLSE code was unable to output Geographic and 3D Trajectory Analysis graph.
Com3 UBX file was unable to convert into NAV file, suspect missing nav data. Therefore only Com4's data was used in the report.