The Hong Kong Polytechnic University

Department of Aeronautical and Aviation Engineering

2024/25 Semester 2

AAE6102 Satellite Communication and Navigation

Assignment 1

Technical Report

Written by Wenting Luo

Department of Land Surveying and Geo-Informatics

March 2025

Introduction

Developing a comprehensive understanding of GNSS Software-Defined Receiver (SDR) signal processing is crucial for researchers in the GNSS navigation field. You are provided with two real Intermediate Frequency (IF) datasets, which were collected in open-sky area and urban environment, respectively (see Figure 1). The urban dataset likely exhibits typical urban effects such as multipath and non-line-of-sight (NLOS) receptions, which can degrade signal quality and lead to positioning errors. The collected GPS L1 data are described in Table 1. You are required to process and analyze the IF data using any open-source GNSS SDR code.

Installation

Make sure you have a working MATLAB environment. To install GPSSDR, simply unzip the file into a folder, e.g., ./GPSSDR/. The package contains six folders and one technical report. The sample data set is located in the folder called DataSample. The data file is compressed due to its large size, so before using the sample data set, please unzip the data file. The MATLAB source code is located in three folders, acqtckpos and geo and plot. The results for the sample data are found in the folders OpenSky_Result and Urban_Result, respectively. Set the current folder to ./GPSSDR/ in MATLAB, and then run the main program, SDR_main.m.

Usage

GPSSDR is very easy to use. This section presents the usage through two examples. The raw IF data were collected in open-sky and urban environments (see Figure 1). The sampling frequency is 58 MHz and 26 MHz, respectively. The urban dataset likely exhibits typical urban effects such as multipath and non-line-of-sight (NLOS) receptions, which can degrade signal quality and lead to positioning errors. The collected GPS L1 data are described in Table 1.

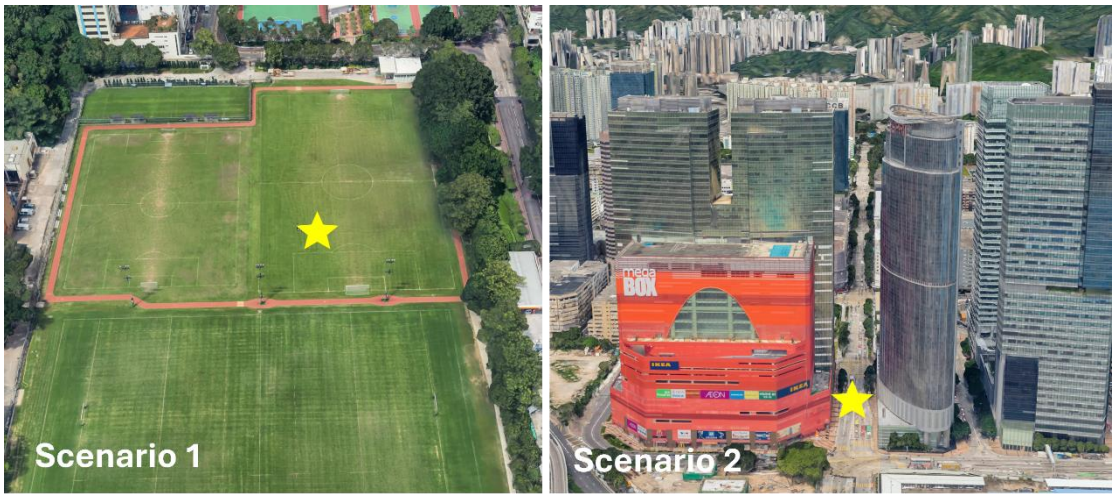

Figure 1. Data collection locations

Table 1 Sample data description

	Opensky	Urban
Carrier frequency	1575.42 MHz	1575.42 MHz
Intermediate frequency	4.58MHz	0
Sampling frequency	58MHz	26MHz
Data Format	8 bit I/Q samples	8 bit I/Q samples
Ground truth	(22.328444770087565, 114.1713630049711)	(22.3198722, 114.209101777778)
Data length	90 seconds	90 seconds

To begin with, it is necessary to understand the folder structure of the software, as shown in Figure 2. This software consists of three separate files: the main program, the initialization functions (initParameters_OpenSky.m and initParameters_Urban.m), and three folders containing baseband signal processing functions, geo-related functions, and plot functions. The baseband signal processing functions handle signal acquisition, tracking, navigation data extraction, etc., while the geo-related functions perform coordinate transformation, atmospheric corrections, and so on. The plot-related functions are responsible for plotting. Each file contains comments that clearly specify the purpose of the respective function.

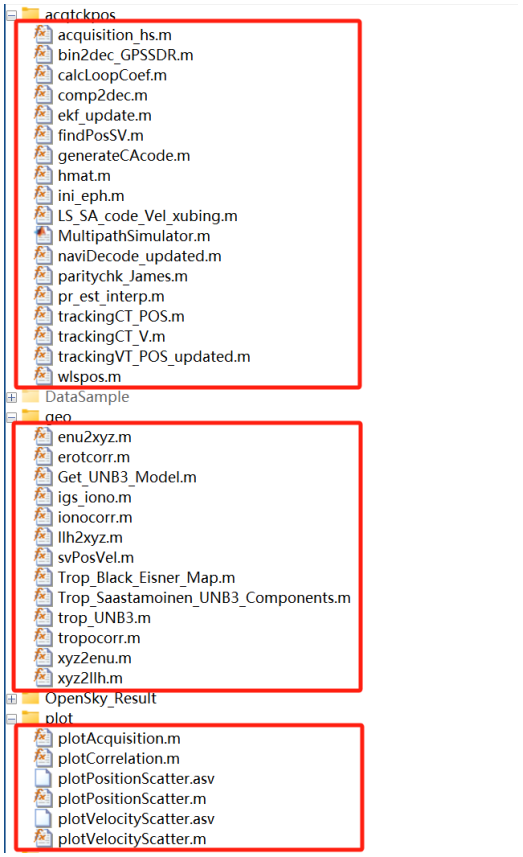

Figure 2 Folder structure.

Initialization parameters

To process the raw IF data, the first step is to initialize necessary parameters for acquisition, tracking, positioning, etc. in the file named initParameters_OpenSky.m and initParameters_Urban.m. Some important initialization parameters are listed in Table 2:

Table2. Initialization parameters.

	Parameter	Description
Raw data parameters	*file.filename*	Raw data file name
	file.skip	Skip time in milliseconds, from when raw data is processed
Acquisition Parameters	acq.prnList	PRNs to be searched
	acq.freqStep	Frequency search step
	acq.freqMin	Minimum frequency to be searched
	acq.freqNum	Frequency bin numbers
	acq.threshold	Acquisition threshold
	acq.L	Numbers of code periods to find fine frequency
Tracking Parameters	track.CorrelatorSpacing	Correlator spacing between the Early and Late code
	track.msToProcessCT	Time to do positioning via LSE/EKF in milliseconds
	track.msToProcessVT	Time to do positioning in vector tracking mode in milliseconds
	track.pdi	Prediction integration time
Navigation Parameters	solu.iniPos	Initial position, latitude (degree), longitude (degree), height (meter)
	solu.mode	Position method: 0-WLS algorism;1-EKF algorism
Common Parameters	cmn.vtEnable	Tracking mode: 0 - Conventional tracking, 1 - Vector tracking
Signal Parameters	file.skiptimeVT	Skip time in milliseconds, from when vector tracking begins
	file.dataType	Raw data type: 1 - I, 2 - I/Q
	file.dataPrecision	Data size: 1 - 'int8', 2 - 'int16'
	signal.IF	Intermediate frequency
	signal.Fs	Sampling rate
	signal.Fc	Carrier frequency
	signal.codeFreqBasis	Code frequency
	signal.Sample	Numbers of samples in one code period
	signal.codelength	Code length

Task 1 – Acquisition

Process the IF data using a GNSS SDR and generate initial acquisition outputs.

After initialization, type 'SDR_main' in the MATLAB command window and press [Return]. This software will first acquire the visible satellites, outputting the satellite number (SV), signal-to-noise ratio (SNR), code phase, and Doppler frequency in the command window, as shown in Figure 3. The acquisition result is saved in the current folder with the name 'Acquired + raw data file name + .mat'.

```
    30


svindex =

    31


svindex =

    32

SV[ 1] SNR = 42.21, Code phase = 22672, Raw Doppler =  1000, Fine Doppler = 1205.000000
 SV[ 3] SNR = 29.71, Code phase =   829, Raw Doppler =  4500, Fine Doppler = 4280.000000
 SV[ 7] SNR = 20.43, Code phase = 10810, Raw Doppler =   500, Fine Doppler = 370.000000
 SV[11] SNR = 26.13, Code phase = 24845, Raw Doppler =   500, Fine Doppler = 410.000000
 SV[18] SNR = 21.80, Code phase = 15421, Raw Doppler =  -500, Fine Doppler = -325.000000
 SV[22] SNR = 16.52, Code phase =  1805, Raw Doppler =  3500, Fine Doppler = 3365.000000
 Acquisition Completed.
```

Figure3. Acquisition results shown in MATLAB command window

To further visualize the acquisition results, use a bar chart to plot the satellite signal acquisition results. The x-axis represents each PNR number in Acquired.sv, and the y-axis corresponds to the SNR value for each satellte. A peak higher than the acquisition threshold indicates a successful acquisition of that satellite. If satellites are not detected, use an 'x' marker. When the acquisition threshold is set to 16, the model acquisition results for OpenSky and Urban data are as follows (Figure4 and 5):

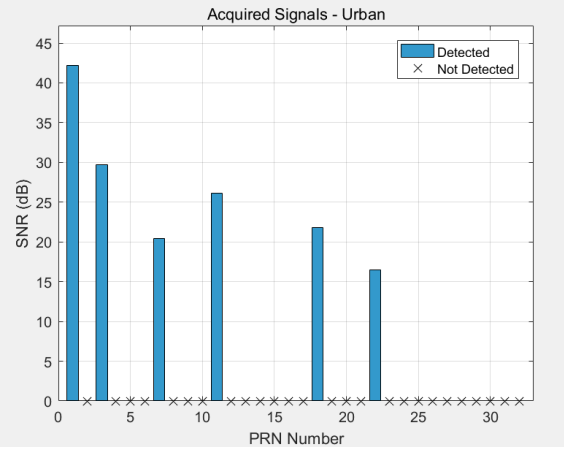

Figure4. Acquisition results of Urban data.

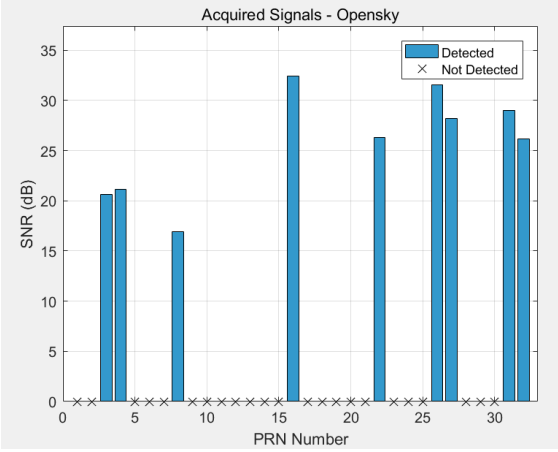

Figure 5. Acquisition results of OpenSky data.

From Figures 4 and 5, when the threshold value was set to 16, the Urban data acquired signals from 6 satellites, and the OpenSky data acquired signals from 8 satellites.

Task 2 – Tracking

Adapt the tracking loop (DLL) to produce correlation plots and analyze the tracking performance. Discuss the impact of urban interference on the correlation peaks. (Multiple correlators must be implemented for plotting the correlation function)

After acquisition, perform signal tracking and obtain satellite ephemeris. In conventional tracking, a DLL with fixed bandwidths is used to track the code. The DLL employs a normalized noncoherent early-minus-late envelope discriminator. A progress bar will appear during the conventional tracking period. The conventional tracking result is saved in the current folder with the name 'TckRstct_Eph + raw data file name + .mat'.

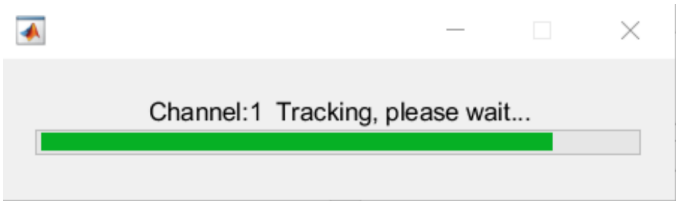

Figure 6. Progress bar during conventional tracking period

To analyze the tracking performance, correlation values against the time delay for each PNR were plotted. The x-axis represents the time delay, with the unit in chips, and the y-axis represents the correlation values. When x = 0, it refers to the prompt time; -0.5 represents 0.5 chips earlier relative to the present, and 0.5 represents 0.5 chips later relative to the present. Different colors are used to represent the correlation values for different satellites.

For the Urban and OpenSky data, 6 and 8 satellites were acquired, respectively. Both datasets have a length of 90 seconds. This means that if 1 ms is recorded for each time step, the Urban and OpenSky data will acquire 540,000 and 640,000 results, respectively. To avoid clutter in the correlation lines, the correlation value line is plotted once for every 1,000 time steps. The OpenSky (Figure 7) and Urban (Figure 8) results are shown as follows:

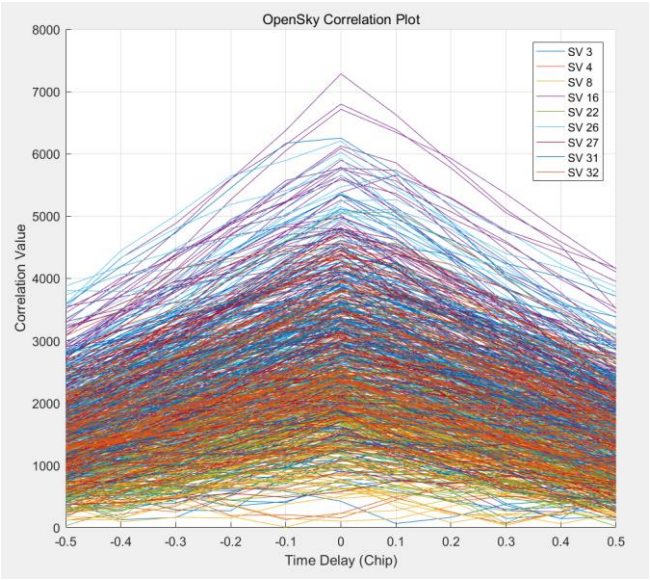

Figure 7. Correlation plots for OpenSky data

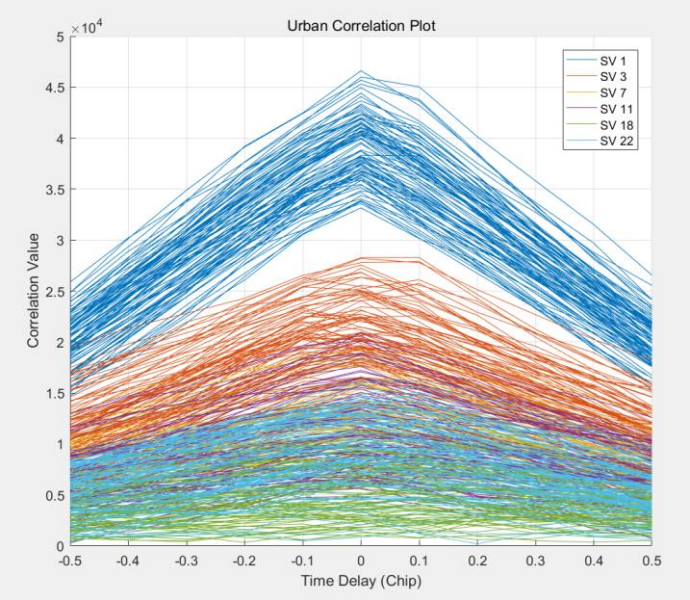

Figure 8. Correlation plots for Urban data

Analysis: Compared to OpenSky data, urban environments introduce multipath effects, where signals reflect off buildings and other structures, causing multiple delayed versions of the signal to arrive at the receiver. This can lead to broader and less distinct correlation peaks, as seen in the urban correlation plot. Additionally, buildings and other obstacles can attenuate the signal, reducing the overall correlation values. This is evident in the urban plot, where the correlation values are generally lower compared to the OpenSky plot.

Task 3 – Navigation data decoding

Decode the navigation message and extract key parameters, such as ephemeris data, for at least one satellite.

After tracking, the decoded satellite ephemeris, named 'eph + raw data file name + .mat', is saved, along with the subframe information, named 'sbf + raw data file name + .mat'. Taking PRN 1 as an example, the decoded key parameters of the OpenSky and Urban data are shown as follows (Table 3):

Table3: Decoded Key Parameters of OpenSky and Urban Data for PRN 1

Parameter	explanation	Value (OpenSky)	Value(Urban)
t_{oe}	Ephemerides reference epoch in seconds within the week	396000	453600
a^0.5	Square root of semi-major axis	5153.75657081604	5153.65564346314
e	Eccentricity	0.00388247426599264	0.00892308494076133
M_0	Mean anomaly at reference epoch	2.74664157628070	0.517930887728971
w	Argument of perigee	0.999580010005351	0.711497598513720
i_0	Inclination at reference	0.970650171544315	0.976127704025529

	epoch		
Ω_0	Longitude of ascending node at the beginning of the week	1.37716785453468	-3.10603580061843
△n	Mean motion difference	4.40196907396673e-09	4.19088885305685e-09
i	Rate of inclination angle	-1.30005415247375e-10	-1.81078971237415e-10
Ω	Rate of node's right ascension	-8.04104922769063e-09	-8.16962601200122e-09
C_{uc}, C_{us}	Latitude argument correction	-5.73135912418366e-06, 6.02193176746368e-06	-6.33485615253449e-06, 5.30108809471130e-06
C_{rc}, C_{rs}	Orbital radius correction	266.031250000000, -111.156250000000	287.468750000000, -120.718750000000
C_{ic}, C_{is}	Inclination correction	-2.60770320892334e-08, -6.33299350738525e-08	-7.45058059692383e-08, 1.60187482833862e-07
a_0	SV clock offset	-0.000324650201946497	-3.48975881934166e-05
a_1	SV clock drift	-1.37561073643155e-11	-9.43600753089413e-12
a_2	SV clock drift rate	0	0

Task 4 – Position and velocity estimation

Using the pseudorange measurements obtained from tracking, implement the Weighted Least Squares (WLS) algorithm to compute user's position and velocity. Plot the user position and velocity, compare it to the provided ground truth values, and comment on the impact of multipath effects on the WLS solution.

Obtain pseudorange measurements from multiple satellites. Start with an initial guess of the user's position and clock bias. Use the solu.mode parameter to set the method for computing the user's position and velocity. If solu.mode is set to 0, it indicates the use of the Weighted Least Squares (WLS) algorithm to compute the user's position and velocity. The position and velocity estimation results are saved in the current folder with the name 'navSolCT_WLS_ + num2str(track.pdi) + ms_ + file.fileneme'.

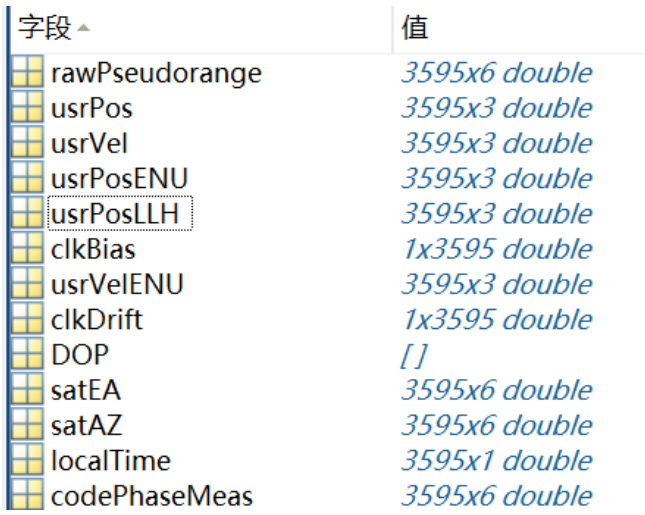

Figure 9: Storage of position and velocity estimation result variables.

For OpenSky and Urban data, the ground truth latitude and longitude coordinates are (22.328444770087565, 114.1713630049711) and (22.3198722, 114.209101777778), respectively. The estimated user position's latitude and longitude results are stored in usrPosLLH, and the specific results are as follows:

navSolutionsCT.usrPosLLH

	1	2	3	4	5	6
1	22.3284	114.1714	2.9998			
2	22.3284	114.1714	2.9924			
3	22.3284	114.1714	2.9796			
4	22.3284	114.1714	2.9613			
5	22.3284	114.1714	2.9376			
6	22.3284	114.1714	2.9085			
7	22.3284	114.1714	2.8739			
8	22.3284	114.1714	2.8338			
9	22.3284	114.1714	2.7884			
0	22.3284	114.1714	2.7374			
1	22.3284	114.1714	2.6811			
2	22.3284	114.1714	2.6193			

Figure 10: Longitude, Latitude, and Height Prediction Results for OpenSky data using the WLS

	1	2	3	4	5
1	22.3198	114.2098	46.2169		
2	22.3198	114.2097	44.6558		
3	22.3197	114.2098	61.7558		
4	22.3197	114.2100	95.0503		
5	22.3199	114.2096	17.4333		
6	22.3199	114.2095	14.6864		
7	22.3199	114.2096	29.4117		
8	22.3196	114.2099	94.0810		
9	22.3198	114.2098	47.4349		
10	22.3197	114.2097	50.9768		
11	22.3196	114.2099	70.9961		
12	22.3196	114.2101	99.9986		
13	22.3197	114.2098	58.2782		
14	22.3197	114.2096	44.5552		
15	22.3198	114.2096	18.6451		
16	22.3198	114.2095	19.5301		
17	22.3199	114.2095	7.7432		

Figure 11: Longitude, Latitude, and Height Prediction Results for Urban data using the WLS

In addition, the error between the estimated position and ground truth values was saved in the 'usrPosENU' field with the ENU coordinate system. Furthermore, the estimation errors of the coordinates were visualized using a scatter plot, where the x-axis represents the user's position offset eastward from the reference point (m), and the y-axis represents the offset northward from the reference point. Different points represent different local estimated reception times (s), with the color of the points becoming bluer as time progresses.

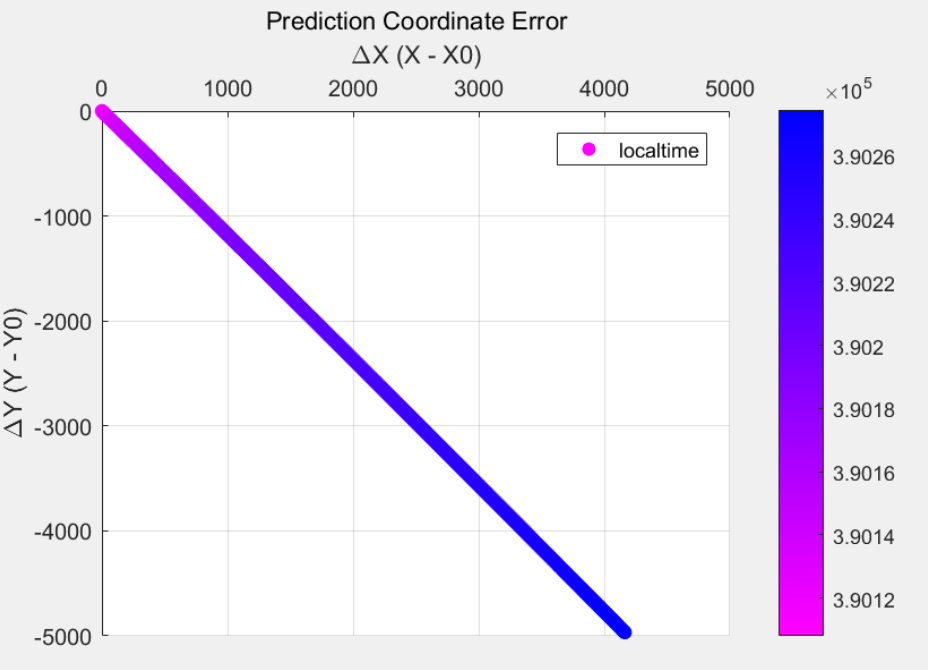

Figure 12. OpenSky position estimation error by WLS algorithm over time

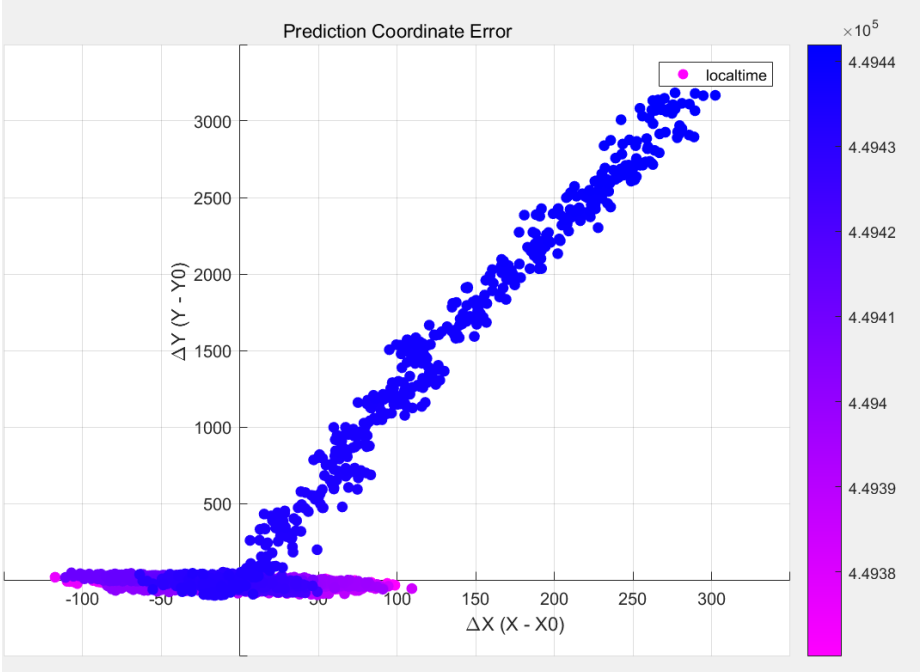

Figure 13. Urban position estimation error by WLS algorithm over time

Analysis: Using the WLS algorithm to estimate the user's position, for the OpenSky environment, the error points are distributed along a clearly linear path, and as time progresses (with the color shifting from purple to blue), the error gradually increases. This could be due to some systematic errors (such as receiver clock drift or satellite orbit errors) that have not been fully corrected.

In the Urban environment, the error points are more dispersed, and as time progresses, the error gradually increases and become more dispersed, showing larger offset. This may not only be influenced by systematic errors, but multipath effects and signal occlusion also significantly affect the positioning accuracy, thus affecting the overall positioning accuracy.

At the same time, the velocity components of the user device in the local East-North-Up (ENU) coordinate system were visualized. In this visualization (Figure 14 and 15), the x-axis represents the user's velocity moving eastward, the y-axis represents the user's velocity moving northward, and points in different colors represent the estimated local reception times.

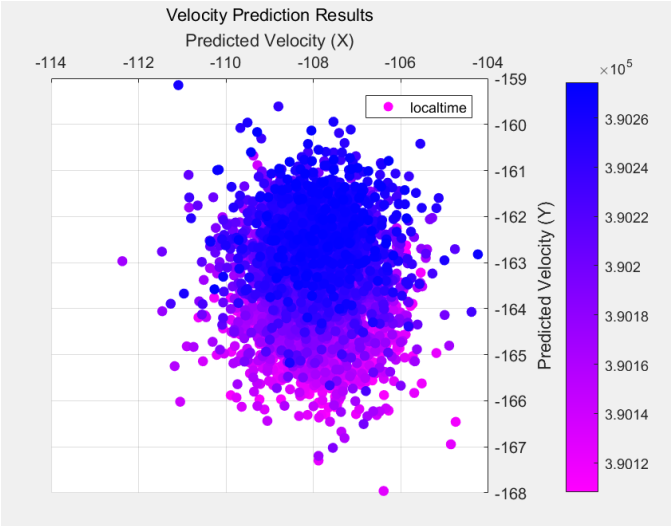

Figure 14. OpenSky data velocity estimation error by WLS algorithm over time

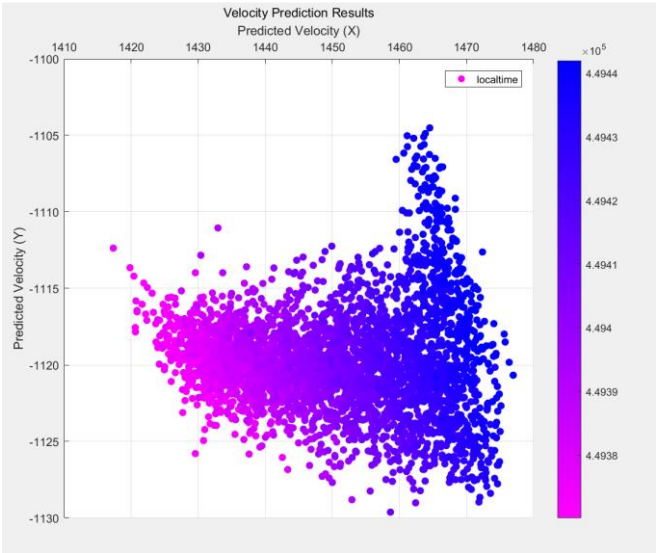

Figure 15. Urban data velocity estimation error by WLS algorithm over time

Analysis: In the OpenSky environment, the velocity estimation points estimated using the WLS algorithm show a relatively uniform distribution along the eastward (X-axis) and northward (Y-axis) directions, and the distribution is concentrated. This may be because the signal propagation path in the OpenSky environment is relatively direct, with high signal quality, leading to smaller velocity estimation errors and a more compact distribution of points. It also suggests that over time, the velocity estimation remains relatively stable with little variation.
However, in the Urban environment, the velocity estimation points show a more dispersed distribution along the eastward and northward directions, with a noticeable bias trend. Due to the presence of multipath effects and signal occlusion in the Urban environment, the signal quality is poorer, which leads to larger velocity estimation errors and a more dispersed distribution of points. Moreover, as time progresses, the velocity estimation fluctuates more significantly, likely due to environmental changes causing variations in signal quality.

Task 5 – Kalman filter-based positioning
Develop an Extended Kalman Filter (EKF) using pseudorange and Doppler measurements to estimate user position and velocity.

If solu.mode is set to 1, it represents using the Extended Kalman Filter (EKF) algorithm to compute the user's position and velocity. The position and velocity estimation results are saved in the current folder with the filename 'navSolCT_EKF_+ num2str(track.pdi) + ms_ + file.fileneme.

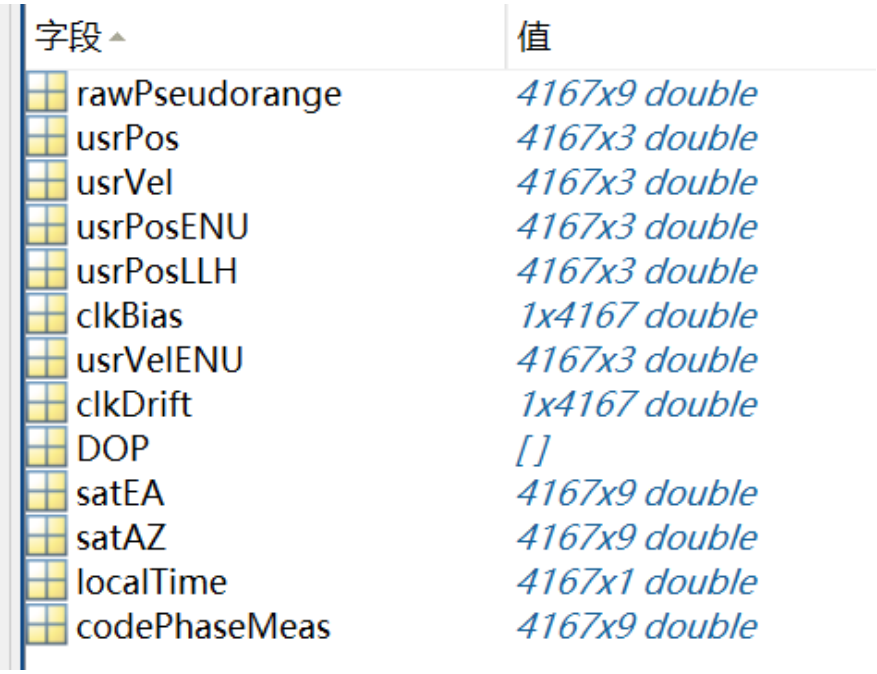

Figure 16: Storage of position and velocity estimation result variables.

For the OpenSky and Urban data, the ground truth latitude and longitude coordinates are (22.328444770087565, 114.1713630049711) and (22.3198722, 114.209101777778), respectively. Using the EKF algorithm, the estimated user position in latitude and longitude is stored in usrPosLLH. The specific results are as follows:

1	2	3	4
22.3284	114.1714	2.9998	
22.3284	114.1714	2.9924	
22.3284	114.1714	2.9796	
22.3284	114.1714	2.9613	
22.3284	114.1714	2.9376	
22.3284	114.1714	2.9085	
22.3284	114.1714	2.8739	
22.3284	114.1714	2.8338	
22.3284	114.1714	2.7884	
22.3284	114.1714	2.7374	

Figure 17: Longitude, Latitude, and Height Prediction Results for OpenSky data using the EKF

1	2	3	4
22.3199	114.2091	2.2930	
22.3199	114.2091	2.2887	
22.3199	114.2091	2.2847	
22.3199	114.2091	2.2791	
22.3199	114.2091	2.2811	
22.3199	114.2091	2.2792	
22.3199	114.2091	2.2801	
22.3199	114.2091	2.2847	
22.3199	114.2091	2.2747	
22.3199	114.2091	2.2775	

Figure 18: Longitude, Latitude, and Height Prediction Results for Urban data using the EKF

In addition, the error between the estimated position and ground truth values is saved in the 'usrPosENU' field using the ENU coordinate system. Furthermore, the estimation error of the coordinates is visualized using a scatter plot, where the x-axis represents the user's displacement eastward from the reference point (in meters), the y-axis represents the displacement northward from the reference point, and different points represent different local estimated reception times (in seconds). The further in time, the bluer the color of the points.

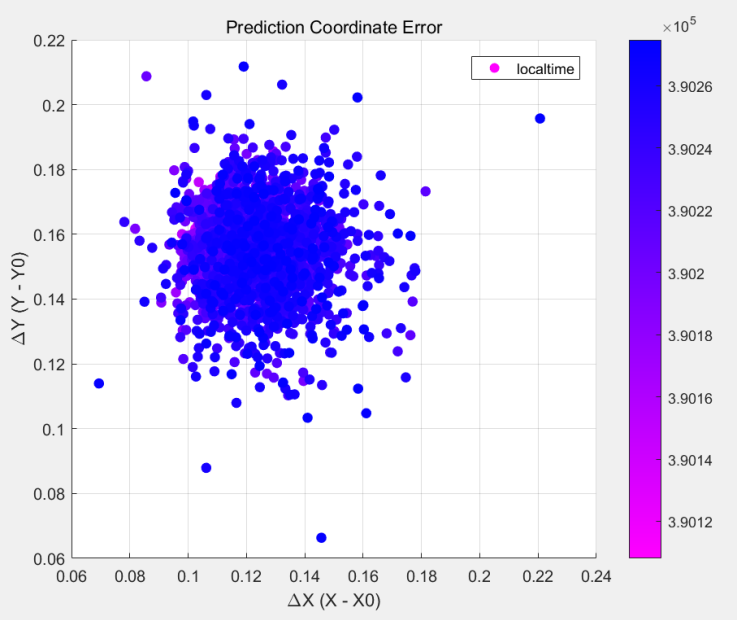

Figure 19. OpenSky data position estimation error by EKF algorithm over time

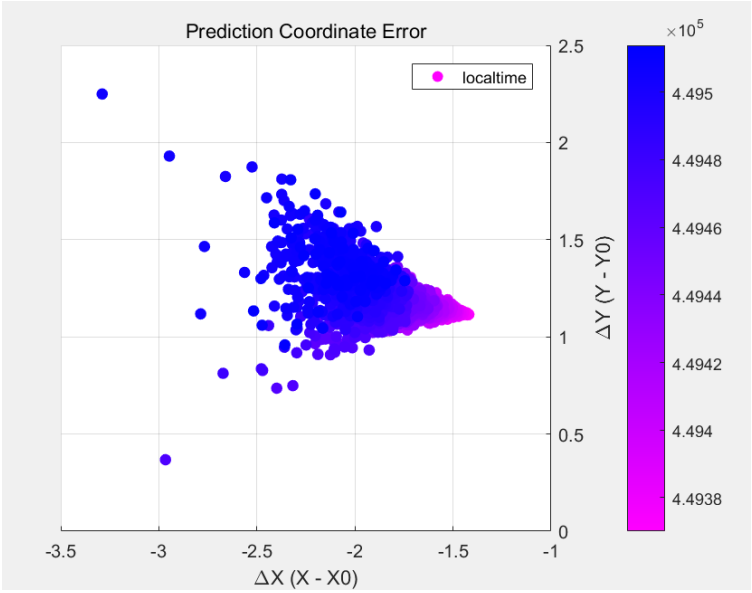

Figure 20. Urban data position estimation error by EKF algorithm over time

Analysis: In the OpenSky environment, the error points are more concentrated, and the error magnitude is relatively small, indicating good positioning accuracy. Furthermore, as time progresses, the error remains relatively stable, demonstrating good system stability and excellent positioning performance. In contrast, in the Urban environment, the error points show greater dispersion, and as time passes, the error not only increases but also exhibits a certain bias trend. This may be due to signal quality fluctuations caused by environmental changes.

Additionally, compared to the WLS algorithm, the use of the EKF algorithm significantly reduces the user position estimation error in both the Urban and OpenSky environments. This could be because the EKF can update the state estimate in real-time, adapting to dynamic environmental changes. Moreover, the EKF is better at handling complex dynamic models and nonlinear relationships, improving positioning accuracy.

At the same time, the velocity components of the estimated user device in the local East-North-Up (ENU) coordinate system were visualized. The x-axis represents the user's velocity moving eastward, the y-axis represents the user's velocity moving northward, and the points of different colors represent the estimated local reception times.

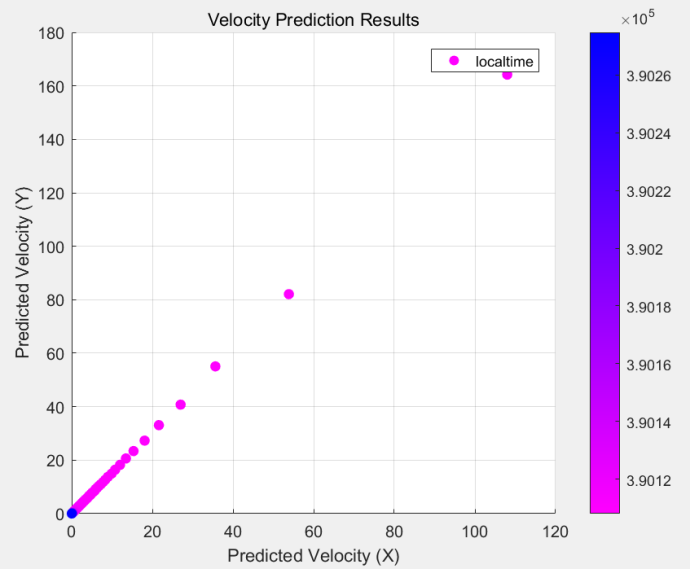

Figure 21. OpenSky data velocity estimation error by EKF algorithm over time

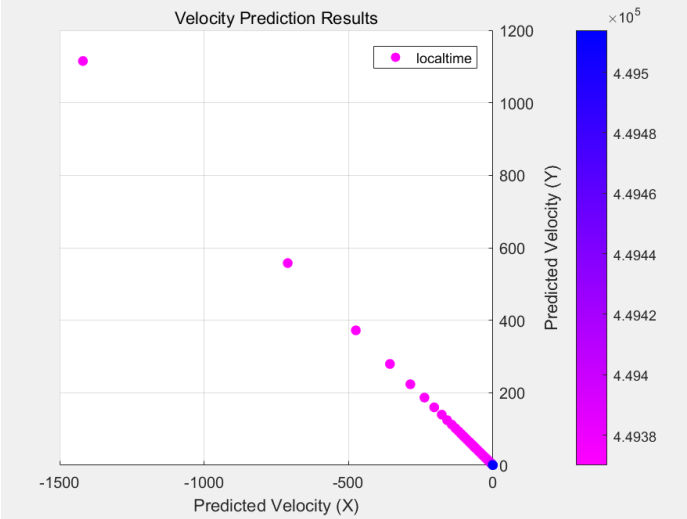

Figure 22. Urban data velocity estimation error by EKF algorithm over time

Using the EKF algorithm to estimate the user's velocity, in the OpenSky environment, where the signal quality is higher, the velocity estimation is quite accurate. As time progresses, the velocity decreases linearly, gradually approaching zero. In the Urban environment, the initial velocity estimation error is relatively large, but over time, the velocity decreases linearly and approaches zero. This indicates that the EKF algorithm can accurately estimate the velocity in both OpenSky and Urban environments. Especially in open spaces with minimal signal obstruction and multipath effects, such as in the OpenSky environment, the velocity estimation results are superior to those of the WLS algorithm.

References

Xu, Bing, and Li-Ta Hsu. "Open-source MATLAB code for GPS vector tracking on a software-defined receiver." GPS solutions 23.2 (2019): 46.

Declaration

The code for this assignment is modified based on the original code (from References, written by Prof. Xu, Bing, and Li-Ta Hsu.).