Open Source Matlab Code for a Vector Tracking-based GPS Software-Defined Receiver

User Manual

Written by Bing Xu

The Hong Kong Polytechnic University
Hong Kong, China

January 2019

Contents

Introduction

Due to its increased immunity to receiver dynamics, interference and jamming, as well as its ability to bridge short signal outages, vector tracking has received increased attention in the GNSS community in recent years. This software implements the vector delay lock loop (VDLL) algorithm on a software-defined receiver (SDR), and aims at providing users with a tool to investigate the pros and cons of vector tracking in various applications and under various environments. This software is called GPSSDR_vt hereinafter for convenience.

In GPSSDR_vt, an "equivalent" conventional tracking SDR based on an extended Kalman filter (EKF) is implemented, which shares the same state, system and measurement models and noise tuning methods with that used by vector tracking. This baseline provides users with a tool to compare the performance of vector tracking and conventional tracking on common ground. GPSSDR_vt is developed in MATLAB, an easy-to-use programming language, so that users can focus more on testing newly developed algorithms related to vector tracking based on this platform.

Requirements

GPSSDR_vt is currently developed and tested in MATLAB environments on Windows platforms. It can also work on a Linux operating system. It does not use any MATLAB toolboxes, but the MATLAB version is required to be greater than 7.6.

Installation

Make sure you have a working MATLAB environment. To install GPSSDR_vt simply unzip the file into a folder, e.g., *./GPSSDR_vt*/. There are two folders and one user manual in the package. The sample data set is in the folder called *sample data*. The data file is compressed due to its large size. Before using the sample data set, please unzip the data file. The MATLAB source code is in the folder called *source code*. Open the MATLAB application and set the current folder as *./GPSSDR_vt/source code/*. Then run the main program, *SDR_main.m*.

Supported Front-ends

GPSSDR_vt supports various front-ends, e.g., IP-Solutions, bladeRF X40, NSL Stereo, etc. Users need to make appropriate configurations for the front-end used, in the script *initParameters.m.*

Processing Modes

GPSSDR_vt is currently only working in the post-processing mode. Raw IF data should be first collected by an RF front-end, then processed using this software.

Main Functionalities

The flowchart of GPSSDR_vt is given in Figure 1, together with the name of the script for each functionality. Main functionalities include initialization, acquisition, conventional tracking and vector tracking, which are described in detail as follows.

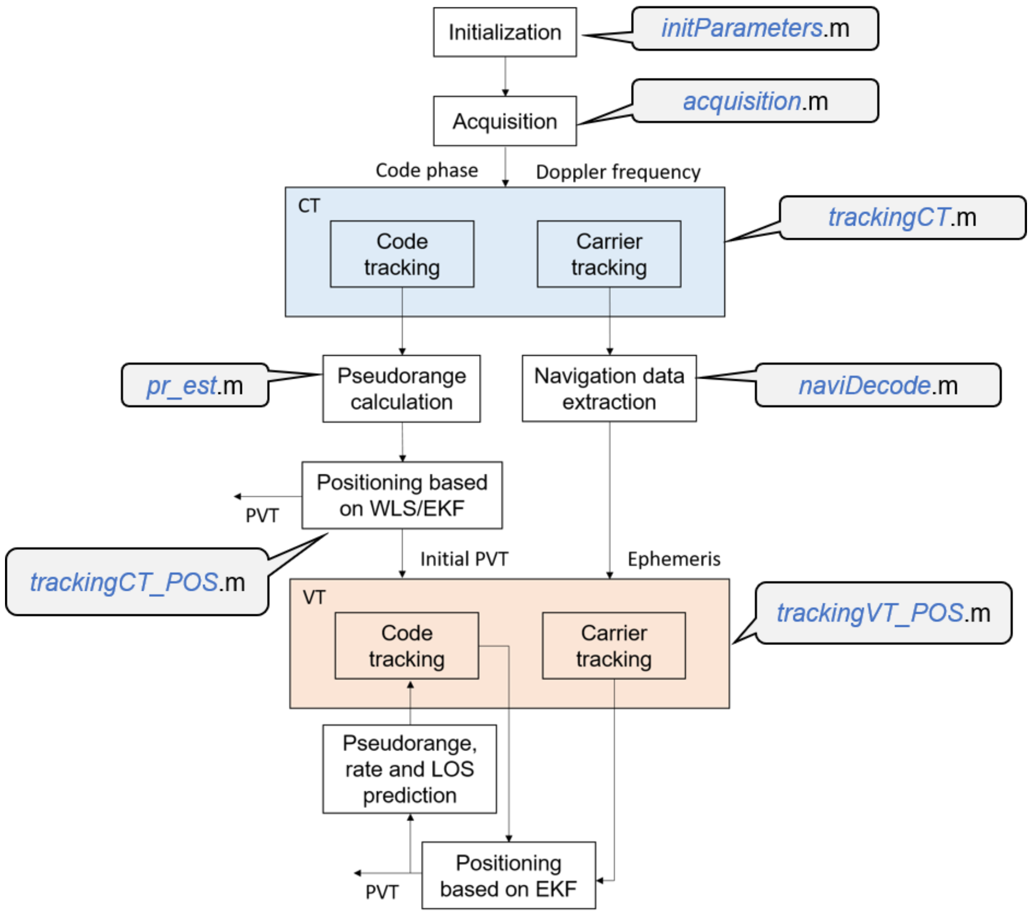


Figure 1. Software flowchart.

Initialization

The first step to use this software is to complete configurations such as the sampling rate and intermediate frequency of the raw signal, the frequency step and band to be searched in the acquisition process, etc.

Acquisition

The second module is signal acquisition, which determines code phase and Doppler frequency of visible satellites. A two-step coarse-to-fine acquisition method is used in this software. In the first step, the 4-ms data is used to detect the code phase and Doppler frequency coarsely via the parallel code phase search acquisition algorithm, as shown in Figure 2. The second step utilizes the long C/A code-stripped data to find the carrier frequency accurately via the fast Fourier transform.

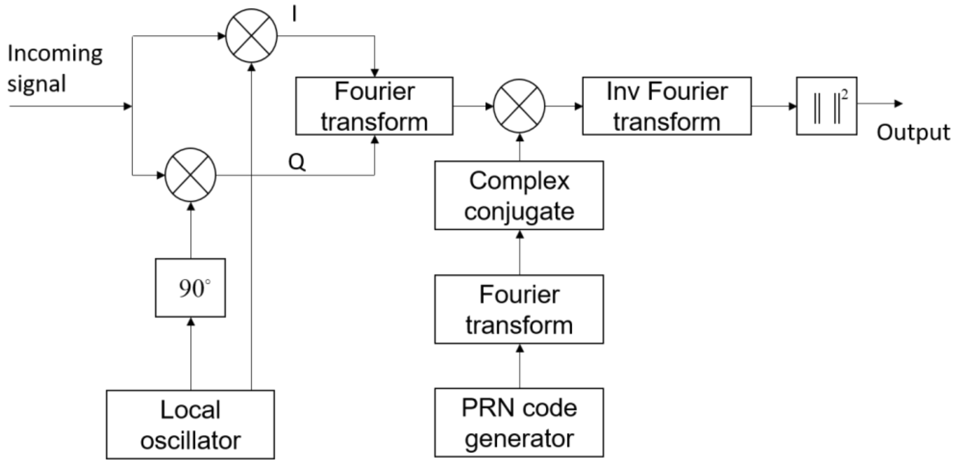


Figure 2. Block diagram of the parallel code phase search algorithm (Van Nee and Coenen, 1991).

Conventional Tracking

After obtaining the code phase and Doppler frequency, these two parameters should be refined in the tracking stage so that satellite ephemeris data can be decoded. Measurements of pseudorange and pseudorange rate can also be obtained during tracking. A second-order DLL and PLL is used in this software, as shown in Figure 3.

In conventional tracking loops, each acquired satellite is allocated to an individual tracking channel. Each channel has two closed loops, one for code and one for carrier. All tracking channels are independent of each other, i.e., no interaction between channels, and no information exchange between signal tracking and navigation processors. The pseudorange range and rate measurements are fed forward to the positioning module, e.g., EKF, to compute the navigation solution.

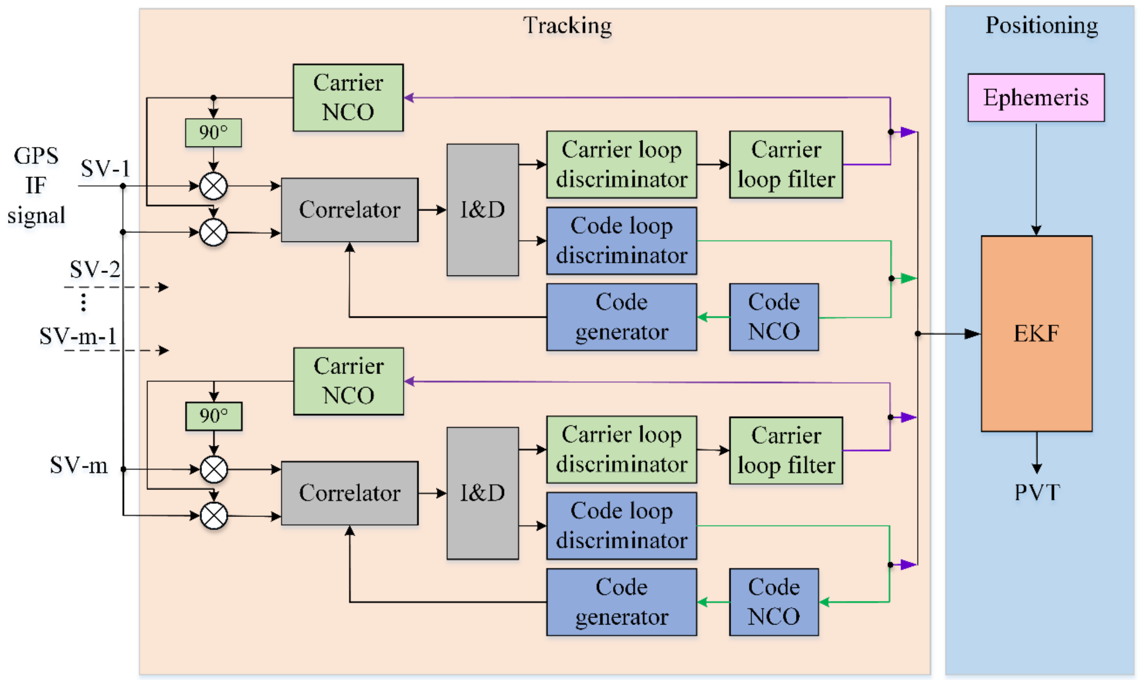


Figure 3. Conventional tracking architectures in GPSSDR_vt.

Vector Tracking

To start vector tracking, initialization parameters, such as ephemeris data, initial receiver PVT, etc., should be provided. The pseudorange error and pseudo-range rate error extracted from the code and carrier tracking loops are used as the measurements of the EKF. The estimated receiver PVT is then used to predict the pseudorange, rate and the LOS vectors at the next epoch, closing the loop finally. The block diagram of vector tracking is shown in Figure 4.

Each acquired satellite in the incoming intermediate frequency signal is allocated to one tracking channel. In each channel, IF signals are first multiplied with the locally generated carrier replica in both in-phase and quadrature arms. Correlation is then performed between the code replicas and the received ones. In this software, three code replicas spacing of 0.5 chips are generated. Afterwards, correlation results are integrated and dumped. The output of these integrations is used as the input to the carrier/code loop discriminator to find the phase error of the local carrier and code replicas. In each carrier loop, the carrier discriminator output is filtered and fed back to the carrier numerical controlled oscillator (NCO), so as to modify the frequency of local carrier replica. For the code tracking loop, code discriminator outputs of all channels are forwarded to the navigation processor. In this software, an EKF is used. The output of the carrier loop filter, i.e., Doppler shift frequency information, is also fed into the EKF. Note that in practice the EKF update time is not necessary to be the same as the coherent integration time (typically 1 ms for GPS L1 signal). A pre-filter can be used to average the code discriminator outputs over multiple integration time, e.g., 20 ms.

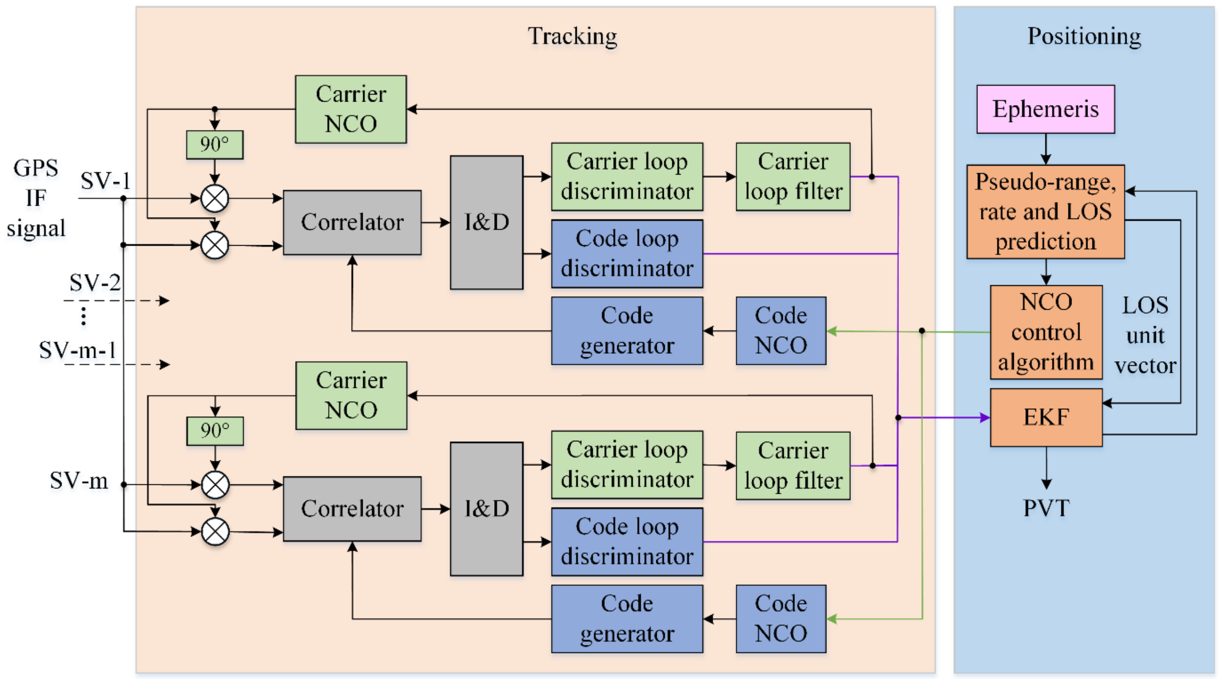
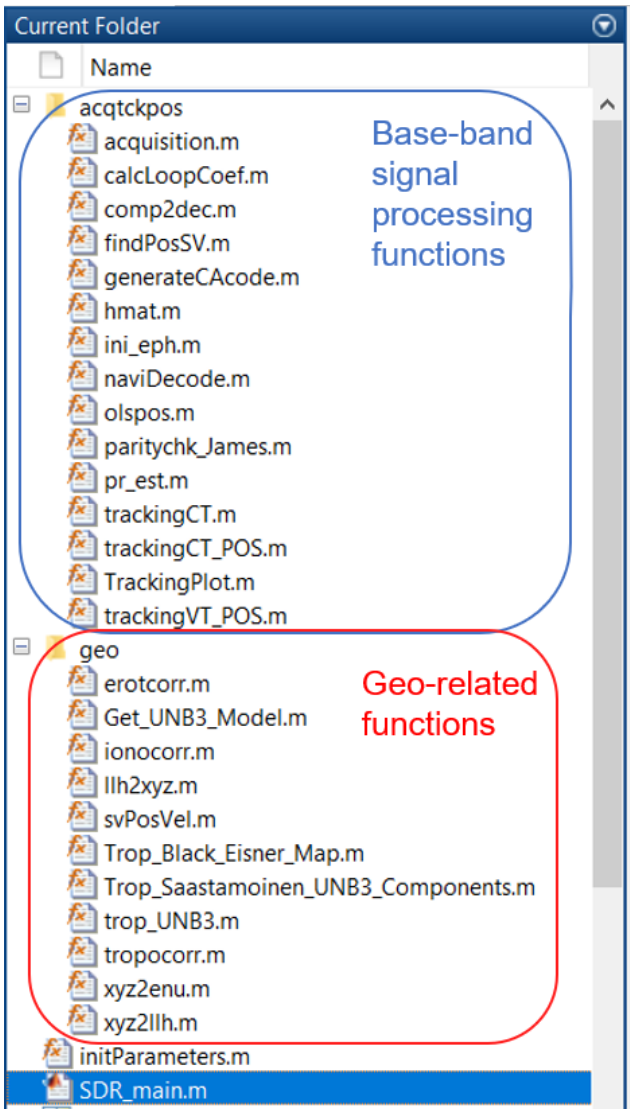


Figure 4. Tracking architecture of GPSSDR_vt.

Usage

GPSSDR_vt is very easy to use. This section presents the usage via an example. The raw IF data was collected in an urban area in Hong Kong using the NSL Stereo front-end. The sampling frequency and IF are 26 MHz and 6.5 MHz, respectively.

To begin with, it is necessary to know the folder structure of the software, as shown in Figure 5. This software is comprised of two separated files, which are the main program and initialization function, respectively, and two folders containing baseband signal processing functions, geo-related functions and plot functions, respectively. Baseband signal processing functions perform signal acquisition, tracking, navigation data extraction, etc., while the geo-related functions do coordinate transformation, atmospheric corrections, and so on. In each file, there's a comment that clearly specifies the purpose of that function.



```
acqtckpos
    acquisition.m
    calcLoopCoef.m
    comp2dec.m
    findPosSV.m
    generateCAcode.m
    hmat.m
    ini_eph.m
    naviDecode.m
    olspos.m
    paritychk_James.m
    pr_est.m
    trackingCT.m
    trackingCT_POS.m
    TrackingPlot.m
    trackingVT_POS.m
geo
    erotcorr.m
    Get_UNB3_Model.m
    ionocorr.m
    llh2xyz.m
    svPosVel.m
    Trop_Black_Eisner_Map.m
    Trop_Saastamoinen_UNB3_Components.m
    trop_UNB3.m
    tropocorr.m
    xyz2enu.m
    xyz2llh.m
initParameters.m
SDR_main.m
```

Figure 5. 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.m*. Some important initialization parameters are listed in Table 1:

Table 1. 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

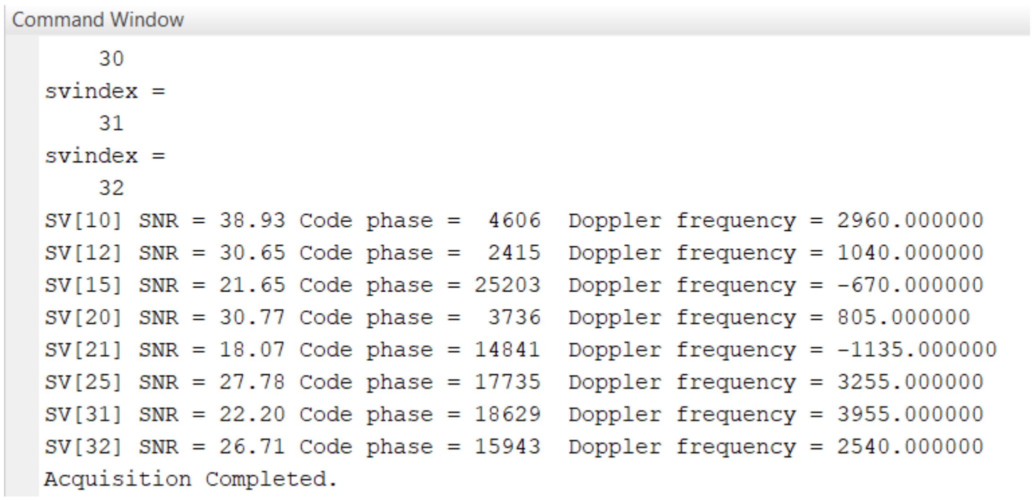
	file.skiptimeVT	Skip time in milliseconds, from when vector tracking begins
	file.dataType	Raw data type: 1 - I 2 - I/Q
	file.dataPrecision	Data size: 1 - ‘int8’ 2 - ‘int16’
Signal parameters	*signal.IF*	Intermediate frequency
	signal.Fs	Sampling rate
	signal.Fc	Carrier frequency
	signal.codeFreqBsis	Code frequency
	signal.Sample	Numbers of samples in one code period
	signal.codelength	Code length
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.msEph	Time needed to decode satellite ephemeris, in milliseconds
	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.rate	Navigation solution update rate in Herts
Common parameters	*cmn.vtEnable*	Tracking mode: 0 - Conventional tracking 1 - Vector tracking
	cmn.doy	Day of year

It should be noted that the inonospheric delay correction parameters (*ALPHA* and *BETA* vectors) should be set according to the RINEX file. Another important parameter is *cmn.vtEnable* which indicates the tracking mode of the SDR, i.e., conventional tracking (*cmn.vtEnable=0*) and vector tracking (*cmn.vtEnable=1*). When doing conventional tracking, navigation solutions are estimated using the LSE or an EKF. In vector tracking, navigation solution estimation and signal tracking are coupled together using an EKF. In this software, the EKF in CT and VT share the same system state vector, process and measurement models.

Acquisition results

After the initialization, type "*SDR_main*" in the MATLAB command window and press [Return]. This software will first acquire the visible satellites, outputting the satellite number, signal-to-noise ratio, code phase and Doppler frequency in the command window, as shown in Figure 6.



```
    30
svindex =
    31
svindex =
    32
SV[10] SNR = 38.93 Code phase =  4606  Doppler frequency = 2960.000000
SV[12] SNR = 30.65 Code phase =  2415  Doppler frequency = 1040.000000
SV[15] SNR = 21.65 Code phase = 25203  Doppler frequency = -670.000000
SV[20] SNR = 30.77 Code phase =  3736  Doppler frequency = 805.000000
SV[21] SNR = 18.07 Code phase = 14841  Doppler frequency = -1135.000000
SV[25] SNR = 27.78 Code phase = 17735  Doppler frequency = 3255.000000
SV[31] SNR = 22.20 Code phase = 18629  Doppler frequency = 3955.000000
SV[32] SNR = 26.71 Code phase = 15943  Doppler frequency = 2540.000000
Acquisition Completed.
```

Figure 6. Acquisition results shown in MATLAB command window.

To visualize the acquisition result, one can dig into the acquisition function to see the intermediate acquisition results. Figure 7 shows the 3D acquisition results of PRN 10. A peak higher than the acquisition threshold claims a successful acquisition of that satellite. In this software, the acquisition threshold is set as the SNR calculated. Users can set this value in the script, *initParameters.m*. The acquisition result is saved in the current folder with the name of *Acquired + raw data file name + .mat*. This software checks the existence of acquisition results of this raw data file according to its name, so that users have no need to acquire signals repeatedly in further development.

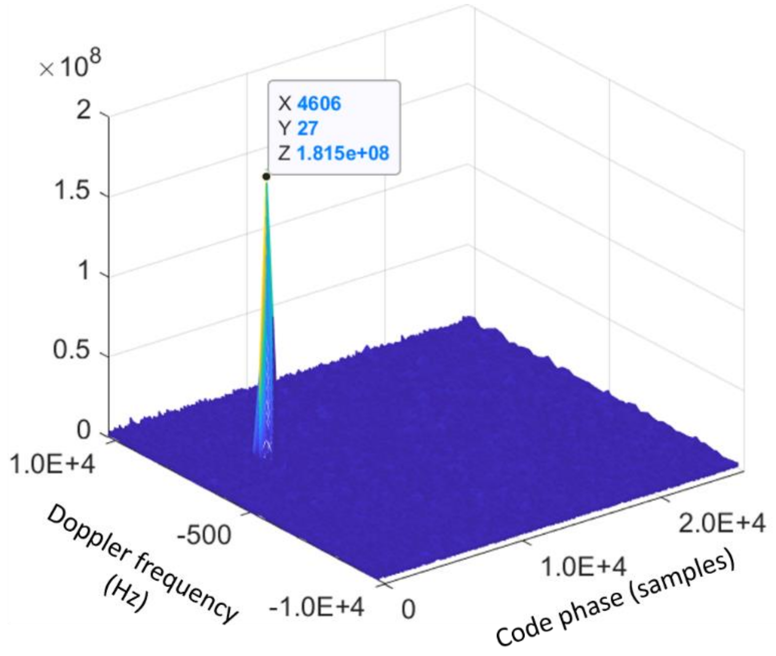


Figure 7. 3D acquisition results of PRN 10.

Tracking results

After the acquisition, do signal tracking and obtain satellite ephemeris. In the conventional tracking, GPSSDR_vt uses a DLL and PLL, both with fixed bandwidths, to track the code and carrier, respectively. The DLL employs a normalized noncoherent early-minus-late envelope discriminator, while the second-order PLL uses a two-quadrant arctangent phase discriminator. A progress bar will appear during the conventional tracking period, as shown in Figure 8. At least 30 seconds of data need to be tracked so as to decode five subframes. The conventional tracking result is saved in the current folder with the name of *TckRstct_forEph +raw data file name + .mat*. The decoded satellite ephemeris, named *eph + raw data file name + .mat*, is also saved, together with the subframe information, named *sbf + raw data file name + .mat*, which contains the information of the first navigation bit transition point and the beginning of subframe 1. This information is useful in the subsequent vector tracking.

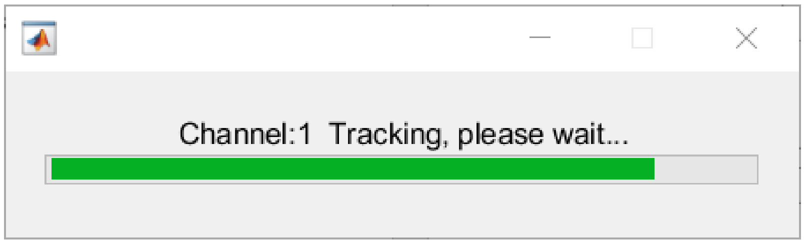

Channel:1 Tracking, please wait...

Figure 8. Progress bar during conventional tracking period.

Figure 9 shows the various tracking results of PRN 10. As seen in the top right panel in Figure 9, the navigation bit steam can be found in the in-phase output of the prompt channel. To obtain the satellite ephemeris, we have to decode these navigation bits into the navigation message using the script named *naviDecode.m*.

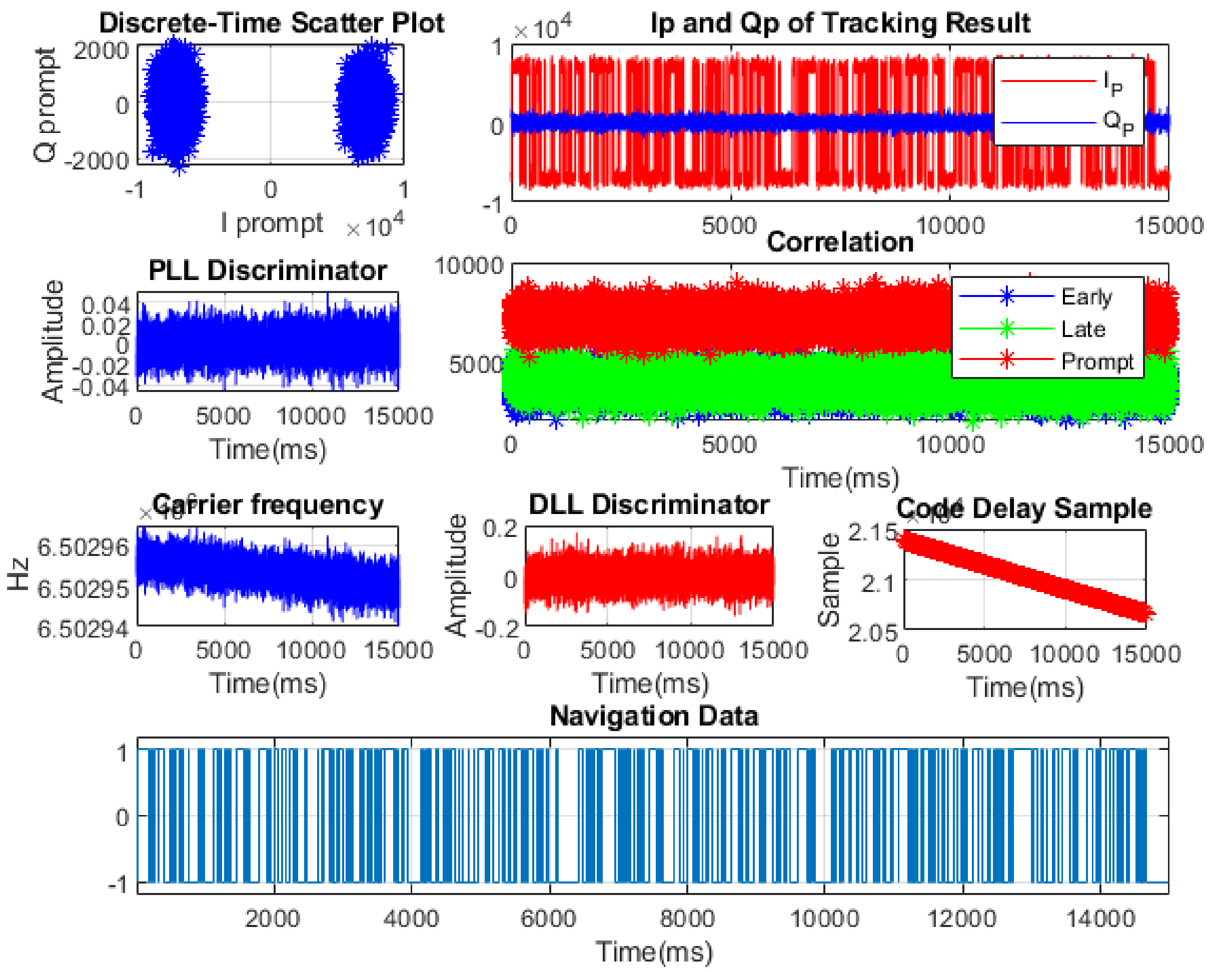
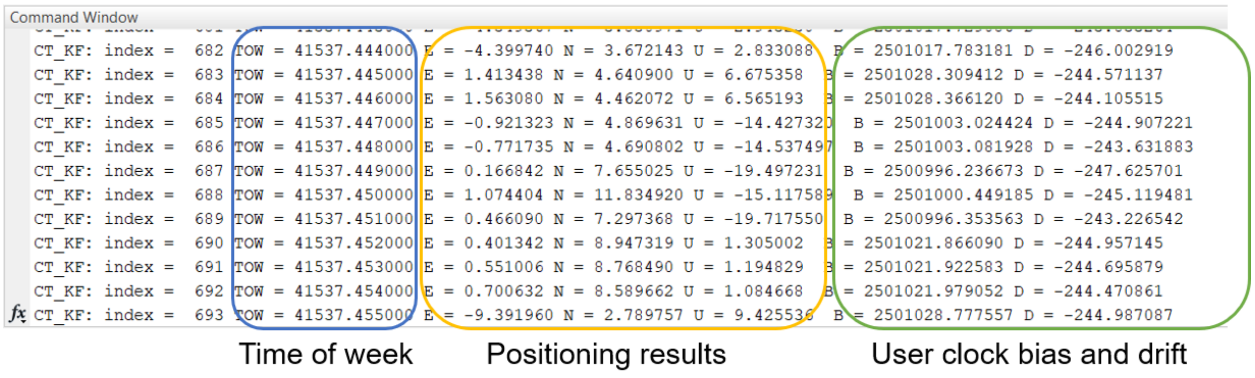


Figure 9. Tracking results of PRN 10.

Positioning results

If *cmn.vtEnable* is set as 0, navigation solutions are calculated using the EKF based on conventional tracking results. On the contrary, *cmn.vtEnable* = 1 means navigation solutions are estimated in the vector tracking mode. To this end, we need the conventional tracking and positioning results and satellite ephemeris to start the vector tracking.

During the positioning process, information such as the time of week, positioning results in East, North and Up directions (with respect to the initial a priori position given in the *initParameters.m* file), and the user clock bias and drift solution will appear in the MATLAB Command Window, as shown in Figure 10. It is helpful for readers to watch the results during the positioning process.



```
CT_KF: index =  682 TOW = 41537.444000 E = -4.399740 N = 3.672143 U = 2.833088  B = 2501017.783181 D = -246.002919
CT_KF: index =  683 TOW = 41537.445000 E = 1.413438 N = 4.640900 U = 6.675358  B = 2501028.309412 D = -244.571137
CT_KF: index =  684 TOW = 41537.446000 E = 1.563080 N = 4.462072 U = 6.565193  B = 2501028.366120 D = -244.105515
CT_KF: index =  685 TOW = 41537.447000 E = -0.921323 N = 4.869631 U = -14.427320  B = 2501003.024424 D = -244.907221
CT_KF: index =  686 TOW = 41537.448000 E = -0.771735 N = 4.690802 U = -14.537497  B = 2501003.081928 D = -243.631883
CT_KF: index =  687 TOW = 41537.449000 E = 0.166842 N = 7.655025 U = -19.497231  B = 2500996.236673 D = -247.625701
CT_KF: index =  688 TOW = 41537.450000 E = 1.074404 N = 11.834920 U = -15.117589  B = 2501000.449185 D = -245.119481
CT_KF: index =  689 TOW = 41537.451000 E = 0.466090 N = 7.297368 U = -19.717550  B = 2500996.353563 D = -243.226542
CT_KF: index =  690 TOW = 41537.452000 E = 0.401342 N = 8.947319 U = 1.305002  B = 2501021.866090 D = -244.957145
CT_KF: index =  691 TOW = 41537.453000 E = 0.551006 N = 8.768490 U = 1.194829  B = 2501021.922583 D = -244.695879
CT_KF: index =  692 TOW = 41537.454000 E = 0.700632 N = 8.589662 U = 1.084668  B = 2501021.979052 D = -244.470861
fx CT_KF: index =  693 TOW = 41537.455000 E = -9.391960 N = 2.789757 U = 9.425536  B = 2501028.777557 D = -244.987087
```

Figure 10. Information in the MATLAB Workspace window during positioning process. The positioning results are with respect to the initial a priori position given in the initParameters.m file.

Figure 11 shows the horizontal positioning results of both conventional tracking and vector tracking, with respect to the initial a priori position given in the *initParameters.m* file.

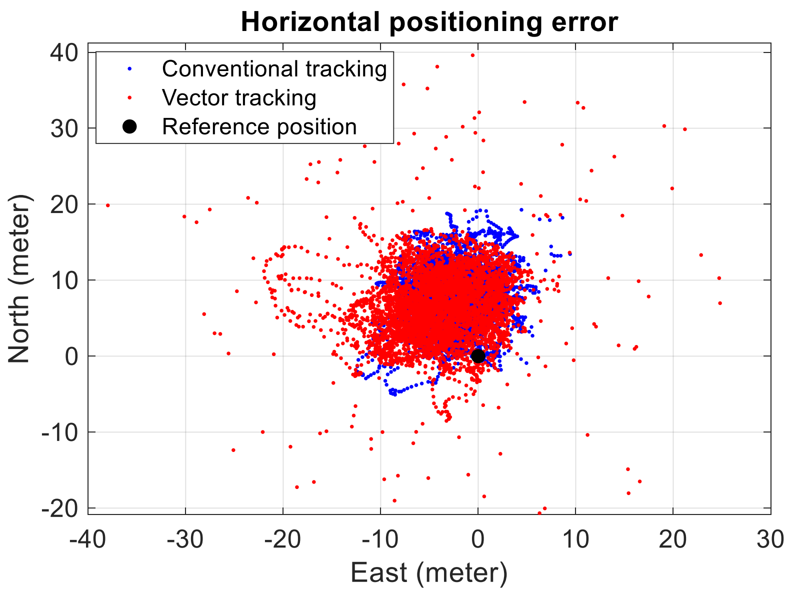
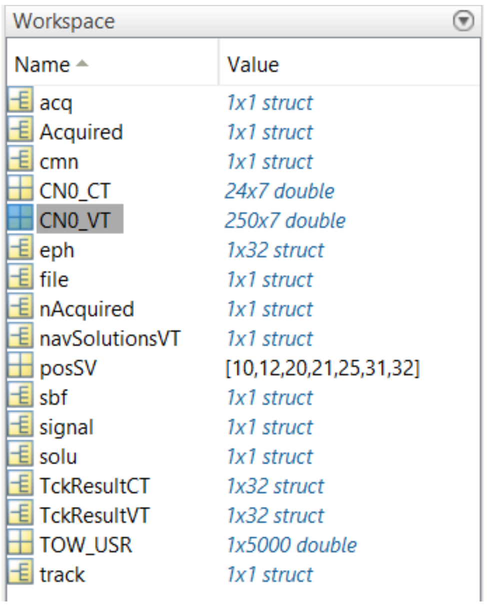


Figure 11. Horizontal positioning results with respect to the initial a priori position given in the initParameters.m file.

In addition, users can plot various results which can be found in the MATLAB Workspace Window after the program execution, as shown in Figure 12.


Workspace

Name	Value
acq	1x1 struct
Acquired	1x1 struct
cmn	1x1 struct
CN0_CT	24x7 double
CN0_VT	250x7 double
eph	1x32 struct
file	1x1 struct
nAcquired	1x1 struct
navSolutionsVT	1x1 struct
posSV	[10,12,20,21,25,31,32]
sbf	1x1 struct
signal	1x1 struct
solu	1x1 struct
TckResultCT	1x32 struct
TckResultVT	1x32 struct
TOW_USR	1x5000 double
track	1x1 struct

Figure 12. Results in the Workspace window after the program execution.

References

Van Nee D, Coenen A (1991) New fast GPS code-acquisition technique using FFT Electronics Letters 27:158-160

Contact Information

This is the first version of this software. Please do not hesitate to contact us if you come across any bugs or have any comments, suggestions or corrections. We will reply you by e-mail as soon as possible.

Bing XU, Li-Ta HSU

pbingxu@polyu.edu.hk, lt.hsu@polyu.edu.hk

Interdisciplinary Division of Aeronautical and Aviation Engineering,

The Hong Kong Polytechnic University, Hong Kong, China