JPNT 8(1), 1-12 (2019)
https://doi.org/10.11003/JPNT.2019.8.1.1

JPNT Journal of Positioning, Navigation, and Timing

Implementation and Experimental Test Result of a Multi-frequency and Multi-constellation GNSS Software Receiver Using Commercial API

Jin-Su Han, Jong-Hoon Won†

Department of Electrical Engineering, Inha University, Incheon 22212, Korea

ABSTRACT

In this paper, we implement a navigation software of a Global Navigation Satellite System (GNSS) receiver based on a commercial purpose GNSS software receiver platform and verify its performance by performing experimental tests for various GNSS signals available in Korea region. The SX3, employed in this paper, is composed of an application program and a Radio Frequency (RF) frontend, and can capture and process multi-constellation and multi-frequency GNSS signals. All the signal processing procedure of SX3 is accessible by the receiver software designer. In particular for an easy research and development, the Application Programing Interface (API) of the SX3 has a flexible architecture to upgrade or change the existing software program, equipped with a real-time monitoring function to monitor all the API executions. Users can easily apply and experiment with the developed algorithms using a form of Dynamic Link Library (DLL) files. Thus, by utilizing this flexible architecture, the cost and effort to develop a GNSS receiver can be greatly reduced.

Keywords: GNSS software receiver, multi-constellation and multi-frequency GNSS, API, navigation unit

1. INTRODUCTION

Currently, global navigation satellite system (GNSS) receivers are applied to various applications such as ground, aviation, and maritime. As of the early 2000s when only the global positioning system (GPS) in the USA and Global Navigation Satellite System (GLONASS) in Russia were present, the mainstream of implementing receivers was the use of single GPS signals or combined signals of GPS and GLONASS. With the research and development by space power nations, new GNSSs such as Galileo in Europe and BeiDou in China were placed in the outer space so various GNSS satellites are now transmitting signals toward the ground. A large number of satellites to run GNSSs such as GPS, Galileo, GLONASS, and BeiDou are orbiting around the earth, and the aim is set to have around 120 satellites in space and more than 90 satellites visible in open sky by 2020 (Ebner 2014).

Thus, the multi-constellation GNSS receiver technology used by receiving multiple GNSS signals have been widely studied from the early 2010s in pace with the system development speed in each country. That is, as the hardware development time of receivers is rapidly shortened, receivers can now be operated using a multi-constellation GNSS method that can process GPS and GLONASS as well as multiple GNSSs such as Galileo and BeiDou (Mattos & Pisoni 2013). However, existing hardware receivers had drawbacks that cannot change the design values according to the changing environment time to time. On the other hand, a large number of design values in GNSS software receivers can be parameterized and changeable according to different applications. Due to this, GNSS software receivers provide highly efficient and convenient methods in terms of cost effectiveness when collecting and tracking new GNSS signals and facilitate easy modifications as well (Abbasian & Petovello 2008). As a result of such advantages, the performance of receiver system can be upgraded easily and

Received Nov 19, 2018 **Revised** Feb 14, 2019 **Accepted** Feb 20, 2019
†**Corresponding Author**
E-mail: jh.won@inha.ac.kr
Tel: +82-32-860-7406 Fax: +82-32-863-5852
Jin-Su Han https://orcid.org/0000-0002-4204-5077
Jong-Hoon Won https://orcid.org/0000-0001-5258-574X

the receiver system can be expanded to an integrated receiver of GPS, Galileo, GLONASS, and BeiDou based on the basic architecture of GPS L1 CA code receiver.

In the meantime, research and development on GNSS software receivers based on universal software radio platform mounted with commercial radio frequency (RF) and additional processing unit have been conducted in recent years (Anghileri et al. 2007). For the development of GNSS receivers, prior developments on overall receiver systems covering communication and navigation technologies such as RF unit, correlator, and navigation unit should be carried out first. The GNSS software receiver requires highly complex and complicate calculation in the real-time mode, and the intermittent data problem is another issue that disturbs real-time processing of the receiver. In general, filtering, down-conversion, and analog-to-digital conversion are conducted with GNSS signals in a GNSS software receiver, which starts at the radio frequency frontend (RF-frontend). The converted digital intermediate frequency (IF) signal data stream is transferred to the main processing unit for de-spreading, tracking, and positioning tasks, where the signal buffer is used to collect digitalized IF data and transfer them to the personal computer (PC) (Chen & Juang 2007).

In summary, a new challenge has emerged in the design of GNSS receivers with the emergence of various and new GNSSs and changes in demand by receiver users. The GNSS software receivers have more advantages than existing GNSS receivers. One of the advantages in GNSS software receiver is to re-design and test a new algorithm by users (Wu et al. 2009). In addition, the GNSS software receiver has an ability to access all steps in the internal signal processing chain that is in general not possible for existing hardware-based receivers. This can be useful to research and development on satellite navigation systems. For this purpose, the GNSS software receiver provides a flexible application programming interface (API) that can add, replace, or extend the functions of the receiver thereby reducing the effort of advanced receiver designers and the development cost and duration, significantly (Heinrichs et. al. 2007).

In this paper, a navigation unit is implemented by using the SX3 API, which is a GNSS software receiver for professional uses and commercially available in the market. Positioning using the least square estimation in each of GPS L1 C/A, GPS L2 P(Y), Galileo E1, GLONASS L1 C/A, and BeiDou B1 is conducted. Furthermore, positioning is conducted using the multi-constellation GNSSs by combining five GNSSs. By implementing the navigation unit using GNSS software API in the above process, this paper proposes a development method that can prove the reduction in development cost and time dramatically within a short period of research effort.

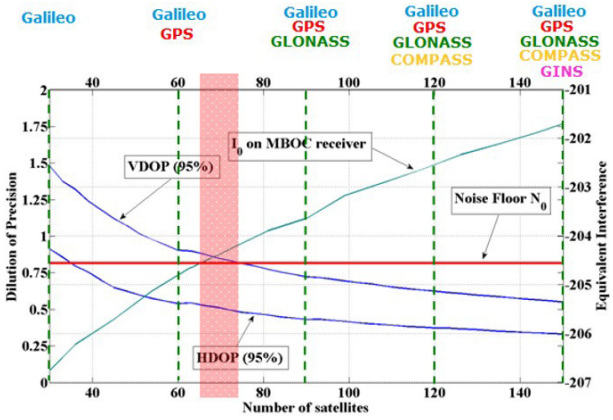

Fig. 1. GNSS Environment in 2020 from user point of view (Ebner 2014).

This paper is organized as follows: in Chapter 2, the hardware configuration of SX3 GNSS receiver and software graphical user interface are described and the structure of the API is explained. In Chapter 3, the non-linear least square estimation (NLSE) is introduced as one of the methods that obtains the positioning results. In Chapter 4, The NLSE algorithm is applied to the API to obtain the positioning results and the results are outputted and analyzed. Then, the effects of the positioning development method using GNSS software API and the direction that is applicable to the future research are presented.

2. MULTI–CONSTELLATION ENVIRONMENT

The multi-constellation environment significantly influences the reception performance of GPS receivers. Currently, a large number of GNSS satellites from several countries are orbiting around the earth. Those nations are increasing the number of GNSS satellites to provide a better service performance and improving the multi-constellation environment by the interoperability, i.e. the inter-operation of different GNSS satellites simultaneously.

Fig. 1 shows GNSS environment as of 2020 according to the user's viewpoint, in which the light red shading part refers to the section that the intra-system interference effect starts to become larger than the noise floor. The goal is to have about 120 satellites in space and more than 90 visible satellites in open sky by 2020. This will definitely bring more improved interoperability but may cause a serious intra-system interference between systems. That is, as the number of satellites increases according to the multi-constellation (the horizontal axis in Fig. 1), dilution of precision (DOP), which represents a theoretical position error according to the geometric arrangement between satellite and receiver,

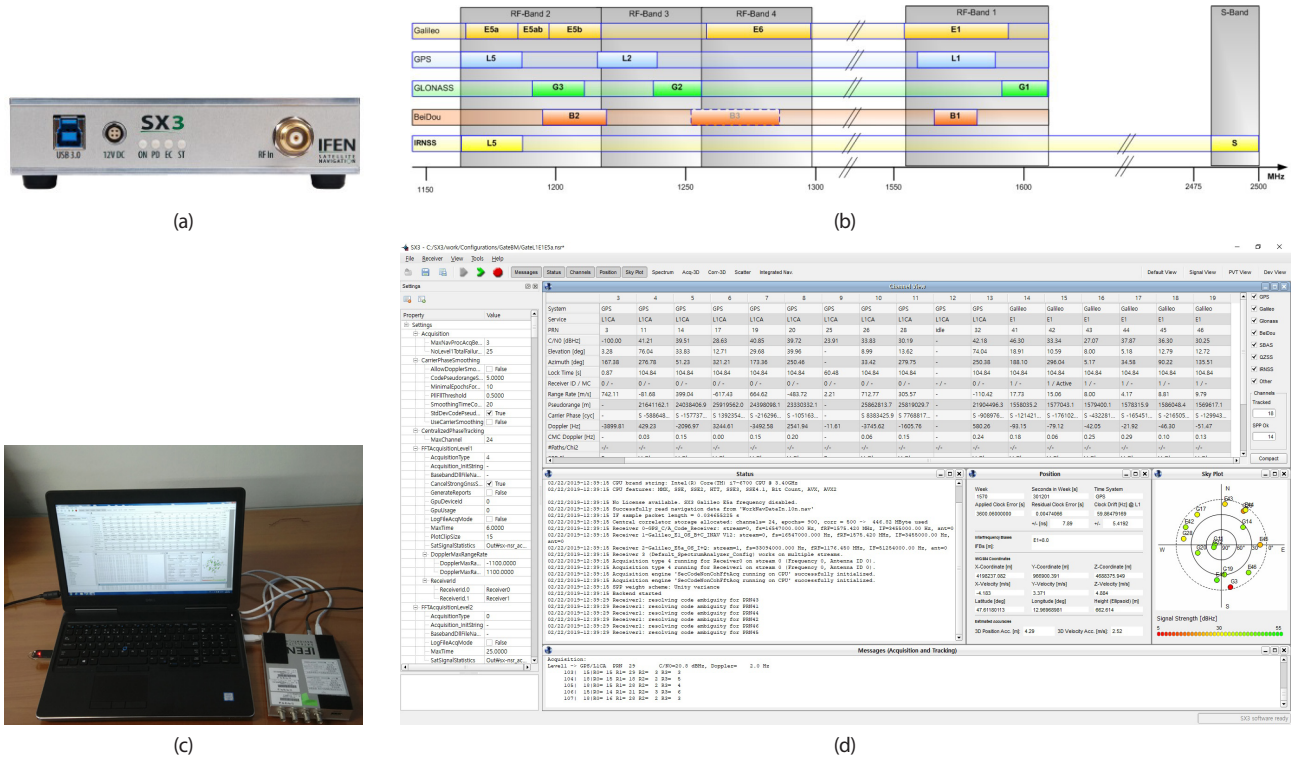

Fig. 2. (a): Front panel of the SX3 frontend housing (single RF version), (b): SX3 frontend frequency plan overview, (c): SX3 Test set-up, (d): Running to SX3 GNSS receiver.exe (IFEN GmbH 2016).

is reduced whereas the intra-system interference effect becomes large. As shown in Fig. 1, the interference effect (equivalent interference) becomes larger than the noise floor when the number of satellites is around 70, resulting in performance degradation. Numerically, the operation of more than three GNSSs is critically difficult from the user's viewpoint when processing all GNSS signals, but users have no significant advantages in contrast (Ebner 2014).

3. SX3 GNSS SOFTWARE RECEIVER

3.1 SX3 Setup

Fig. 2 shows the components of hardware and software in the SX3 GNSS software receiver from IFEN GmbH in Germany. The SX3 frontend can process all GNSS signals inside the L-band and S-band, and programming can be done by connecting it to a host computer through the USB 3.0. As shown in Fig. 2a, the SX3 frontend used in this paper is a single RF version, which consists of single RF-In where antennas can be connected (dual RF version consists of two RF-Ins.), and four out of five RF bands are supported as shown in Fig. 2b. The standard RF bandwidth in the RF path (RF band 1 to 4 and S-band) is 50 MHz (IFEN GmbH 2016).

The SX3 supports real-time and post-processing mode. Fig. 2c shows the experiment set connection, and the SX3 hardware consists of antenna, RF-frontend, and connected PC. The SX3 software receiver can process IF data. Accordingly, it provides a convenient measurement that is needed to the algorithm development by supporting play-back using the post-processing mode. Fig. 2d shows the capture of UI provided by SX3. The UI of SX3 supports real-time and post-processing function, through which environments of reception and monitoring of GNSS signals and execution of developed algorithms are provided according to GNSS type. In addition, SX3 provides a dynamic link library (DLL) file (.dll) execution environment written by C or C++ that can be developed through the API. A user can set up necessary input and output in real time through the property located in the left side of the UI. If the input and output that are preferred by users are located in the same UI of the channel measurements that indicate the satellite characteristics, the input and output can be processed by programming through the API. For the execution of the API, whether DLL algorithm in the API is executed or not is set in the UI property. Once it is set, the execution results can be monitored through the corresponding UI. Thus, users can develop an algorithm of real-time and post-processing mode in C and C++ environment through various functions of SX3

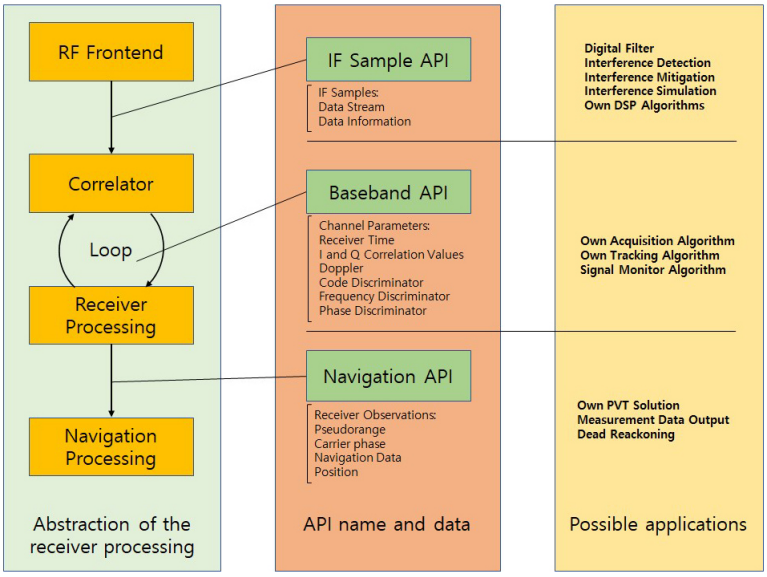

Fig. 3. Accessible data and possible application for the IF Sample API, the baseband API and the navigation API (IFEN GmbH 2016).

Table 1. Application programing interface overview (IFEN GmbH 2016).

Number	Name	Description
1	IF sample API	Gives access to the IF samples, which allows displaying, storing and analyzing the samples. Furthermore one may modify the IF samples (for interference mitigation, signal band limiting etc.).
2	Baseband API	Allows to control signal acquisition and tracking within the SX3 software receiver.
3	Navigation API	Provides raw measurements which may be used to implement own PVT solutions or GNSS/INS integration schemes.
4	Assisted API	Offers an interface for extracting and providing navigation data bits.
5	Sensor API	Allows the retrieval of sensor data captured by the SX3 software receiver.
6	Utility API	Gives access to several software receiver internal parameters and functions. The data transfer gateway is located within this API as well.

Table 2. API function (IFEN GmbH 2016).

Number	API Name	Init	Worker	Closing
1	IF sample	initIfSampleApi	processSamples	closeIfSampleApi
2	Baseband	initAcquisitionApi initTrackingApi	acquire track	closeAcquisitionApi closeTrackingApi
3	Navigation	initNavigationApi	processRawMeasurement	closeNavigation
4	Assisted	initAssistedApi	exportAssistanceData	closeAssistedApi
5	Sensor	initSensorApi	processSensorData	closeSensorApi

and execute the algorithm after changing the completed algorithm into DLL file (.dll) format through the UI's property. In this manner, users can effectively reduce the effort and time to develop the real-time receiver algorithm.

3.2 SX3 API Program

The SX3 GNSS software receiver provides API according to the signal processing step of GNSS. The processing procedure is shown in Fig. 3, and the descriptions on the individual APIs are presented in Table 1. Processing can be done step by step according to the processing structure of the receiver. If processing is not set, default setting is run.

Since the purpose of this paper is to implement the navigation unit, Fig. 3 deals with the API name and data: navigation API, and position, velocity, and time (PVT) solutions are designed and accessed in the possible application. The definitions of the functions in the API are presented in Table 2. In Table 2, the functions in the navigation step are three: InitNavigationApi for initialization, processRawMeasurements that processes the input measurement in earnest, and closeNavigationApi which is run when the navigation step is complete. Through these, the navigation unit can be implemented using the inputs which

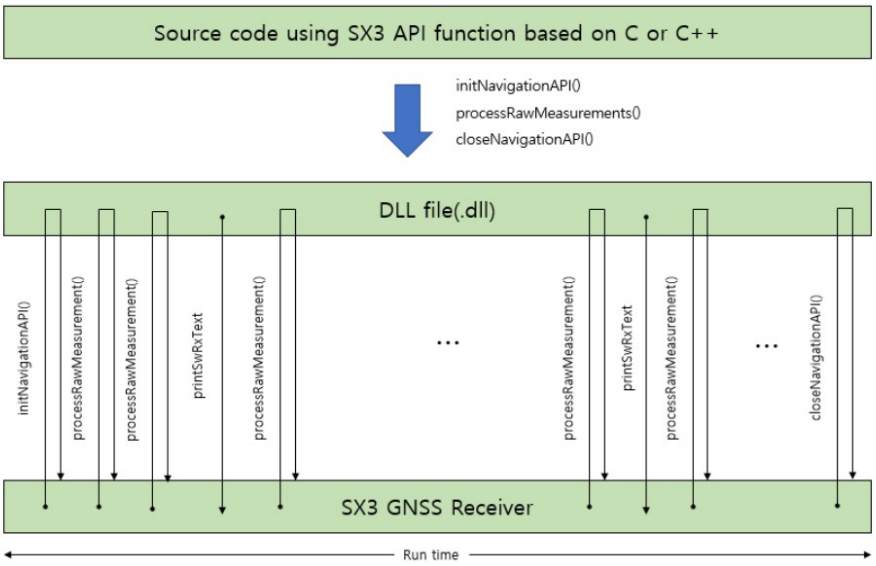

Fig. 4. Function sequence for SX3 API function "processRawMeasurement" (IFEN GmbH 2016).

will be available at the navigation step.

3.3 Execution process of SX3 API

Fig. 4 shows the execution process of the functions in the API briefly. In the SX3 API, which is based on C and C++, provided interfaces and functions are defined according to each of the procedures. The navigation unit is implemented using the navigation API. The navigation unit consists of functions that provide three functions: initNavigationApi, processRawMeasurements, and closeNavigationApi through the navigation API. The DLL files (.dll) are generated through a debugger and then called from the SX3 GNSS software receiver file. Then, SX3 performs the task as written in the API. The initialization process initNavigationApi is activated as soon as the API starts, and various variables and outputs can be set up prior to executing the remaining procedures in initNavigationApi. After the initialization is complete via initNavigationApi, processRawMeasurements function that can utilize various navigation data and printSwRwText are iteratively activated and executed before the closeNavigationApi is activated. The user's algorithm can be applied using various data received from the processRawMeasurements function. And, its result can be outputted to the status window of SX3 execution file using the printSwRwText function. The closeNavigationApi function is activated after all executions are complete. This function is convenient for error control due to iterations through memory management for the next output prior to termination. The above API's functions and variables are defined in the provided header file (*.h).

4. NON-LINEAR LEAST SQUARE ESTIMATION

The final goal of implementing the navigation unit is positioning. The positioning process in this paper uses the NLSE. The NLSE can estimate PVT solutions of the receiver using pseudorange measurements and positions of multiple visible satellites with simple implementations, which is conducted using the equation for pseudorange measurements. When the number of visible satellites is N, the k-th (k=1, 2, ..., N) satellite's pseudorange ρ_k can be expressed as presented in Eq. (1) (Misra & Enge 2011).

$$\rho_k = \sqrt{(x_k - x)^2 + (y_k - y)^2 + (z_k - z)^2} + b + \varepsilon_k \quad (1)$$

where, (x_k, y_k, z_k) and (x, y, z) are the k-th satellite position and user (or receiver) location in the earth-centered earth-fixed coordinate system. In addition, b represents the equivalent clock bias and ε_k refers to the unknown error due to many combined errors (combined effect of the residual errors). Thus, if the value to be estimated is set to the state vector $\underline{\theta} = [x, y, z, b]^T$, the measurement equation $h_k(\underline{\theta})$ with respect to Eq. (1) can be written as Eq. (2).

$$h_k(\theta) = \sqrt{(x_k - x)^2 + (y_k - y)^2 + (z_k - z)^2} + b \quad (2)$$

Here, when the measurement vector $Z = [z_1, z_2, ..., z_N]^T$ that has N pseudorange measurement values is inputted at an arbitrary measurement time from the receiver, measurement residual vector $\underline{e}$ and Jacobian vector H are expressed as the following Eqs. (3) and (4), respectively (Mendel 1995).

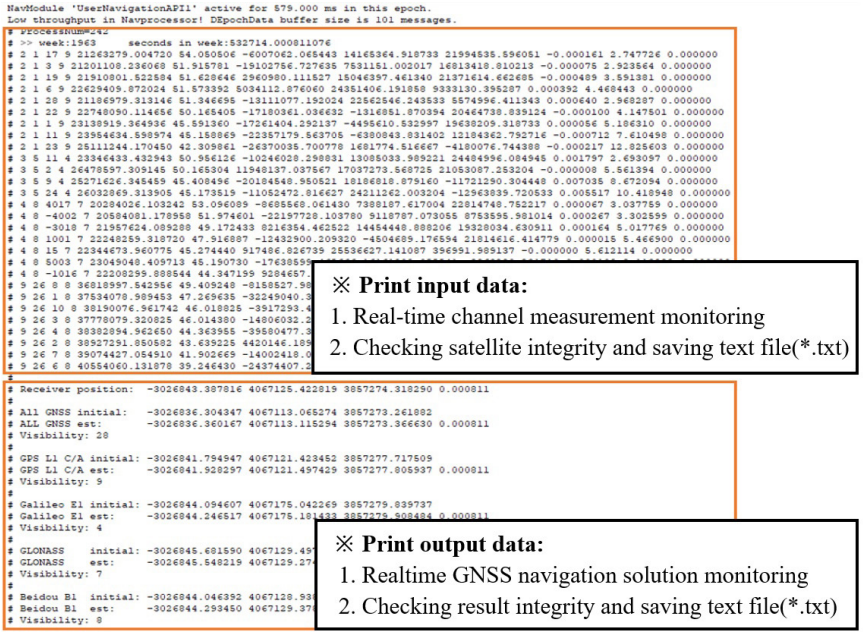

Fig. 5. Real-time operation the designed API-based GNSS software receiver program.

$$\underline{e} = Z - \underline{h}(\hat{\underline{\theta}}) \tag{3}$$

$$H = \begin{bmatrix} \frac{\partial e_1}{\partial x} & \frac{\partial e_1}{\partial y} & \frac{\partial e_1}{\partial z} & \frac{\partial e_1}{\partial b} \\ \frac{\partial e_2}{\partial x} & \frac{\partial e_2}{\partial y} & \frac{\partial e_2}{\partial z} & \frac{\partial e_2}{\partial b} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial e_N}{\partial x} & \frac{\partial e_N}{\partial y} & \frac{\partial e_N}{\partial z} & \frac{\partial e_N}{\partial b} \end{bmatrix} \tag{4}$$

where, $\hat{\theta}$ is the estimated state vector, and $\underline{h}(\hat{\underline{\theta}}) = [h_1(\hat{\underline{\theta}}), h_1(\hat{\underline{\theta}}), \ldots, h_N(\hat{\underline{\theta}})]^T$ represents the vector that enumerates the substitution results of estimated value into the measurement equation. Since it is non-linear with regard to $\underline{\theta}$, it can be calculated using a numerical method, e.g. the Gauss iterative method. After setting the initial value $\hat{\underline{\theta}}_0$, the calculation of Eq. (7) is performed recursively. Here the state vector's estimated value is $\underline{\hat{\theta}} = \underline{\hat{\theta}}_{i+1} \cong \underline{\hat{\theta}}_i$, which is the final estimate value.

$$\underline{\hat{\theta}}_{i+1} = \underline{\hat{\theta}}_i - [H_i^T W H_i]^{-1} H_i^T W \underline{e}_i \tag{7}$$

$$\text{Note that, } H_i = \frac{\partial \underline{e}_i}{\partial \underline{\theta}}\Big|_{\underline{\theta}=\underline{\hat{\theta}}_i} = \frac{\partial}{\partial \underline{\theta}}\left(Z - \underline{h}(\underline{\theta})\right)\Big|_{\underline{\theta}=\underline{\hat{\theta}}_i} = -\frac{\partial}{\partial \underline{\theta}}\underline{h}(\underline{\theta})\Big|_{\underline{\theta}=\underline{\hat{\theta}}_i} \left\| \underline{\hat{\theta}}_{i+1} - \underline{\hat{\theta}}_i \right\| < \delta \tag{8}$$

where, the subscript i refers to the i-th (i = 1, 2, ...) iteration, and the increment δ is set to a sufficiently low value. And, W is the weight of each measurement, which can be configured using C/N_0 value that is related to the reception environment in this paper.

5. MEASUREMENT RESULTS

5.1 Description of the Written Program CODE

The SX3 API provides convenience for users who only perform program coding by selecting hardware variables such as antenna-power from the UI option and writing algorithms with C and C++, compared to existing hardware receivers. These study results are the outcomes of the implementation using the SX3 software receiver's API conducted for about 30 days by one designer. The NLSE algorithm code was written by C and C++ using the API provided within the SX3 software receiver, and DLL files (.dll) were generated, which were then executed in the UI of the receiver operated in real time. The written code is executed in the UI and the results are monitored at the same time. The monitored results are stored as a text file format (.txt). In this study, the results are analyzed based on the configuration of the written code, developed contents, and measurement results stored as a text file format.

The receiver position is estimated through the NLSE process using the data inputted from the receiver. During this process, there are some problems to be solved. The procedure was summarized in the form of the pseudo codes in the Appendices. All channel measurements introduced to the receiver through the API were received as values and processed thereby implementing the navigation unit by writing the source code that performed positioning through

the NLSE procedure. The source code in the navigation unit was designed as it was generalized to enable processing of universal GNSS measurements. Thus, it was designed to have a flexible structure that can reconfigure specific GNSS and frequency receivers by adjusting the setting of the external configuration file.

Fig. 5 shows the program running screen of GNSS software receiver based on the designed API. The measurement data received from the antenna are monitored in real time and multi-constellation GNSS positioning results are monitored in detail at the same time. The integrity of all data outputted during the execution is checked and the data are stored as a text file (*.text) format. All data outputted in the execution screen can be processed with a file format, and signals fed into the receiver can be processed individually to calculate a position solution. There were two problems when obtaining the PVT solution from the signals. First, whether the received data is reliable or not had to be determined. This reliability check function using the single point positioning (SPP) residual during the processing procedure was provided by the SPP flag in SX3. The SPP flag returns OK (without problem) or Raw (sufficient signal is not received) after the range check on each of the satellites. Since all GNSSs had quite more OK returns of SPP flag than Raw, it was designed to receive data whose SPP flag was OK as an input during the NLSE operation as much as possible. Second, the ambiguity problem occurred when signals were received. If GNSS signals had nearly similar directions with one another, it caused a problem in the inverse matrix process during the NLSE process that calculated the position solution. This serious problem occurred during the processing procedure by the API. This refers to a case when matrix equation H for NLSE is close to zero due to satellites whose direction is similar around the receiver during the data processing procedure. A method that reduced the direction ambiguity was adopted by processing H matrix. To solve the above two problems, signals were processed. During this process, GNSS signals were designed to be excluded in case unreliable signals were received and in case signals caused the ambiguity problem. In this way, more reliable signals were used although the API did not utilize input signals in all GNSSs.

Fig. 6. Fig. 6. Mapping experiment position using Google Earth. GNSS antenna on the roof top of the engineering building at Inha University (Latitude: 37.45053046°, Longitude: 126.65743787°, Altitude: 106.217 [m], Date: 18 May 2018, Time: 17:19:01 ~ 18:49:25 (about 90 minute)).

5.2 Measurement Results

The measurement experiments were conducted for 20 min. through antennas installed in the rooftop of Hightech Center building at Inha University as shown in Fig. 6. Each of the satellite signals were processed individually to produce a file form output. The processing results of all signals of GPS L1 C/A, GPS L2 P(Y), Galileo E1, GLONASS L1 C/A, and BeiDou B1 were displayed. The positioning results using the API and SX3 were compared and analyzed with the data captured in the measurement campaign. The results are shown in Figs. 7-9, and presented in Table 3. Table 3 briefly presents the positioning errors. Fig. 7 shows the positioning results including the SX3's navigation solutions according to all GNSS satellites of GPS L1 C/A, GPS L2 P(Y), Galileo E1, GLONASS L1 C/A, and BeiDou B1 using two-dimensional scatter plots, in which yellow point refers to the SX3 position and blue refers to the mean of the position solutions of the SX3. Fig. 8 shows the sky plot that shows each of the satellites around the receiver. Fig. 9 shows the position error of each satellite, GDOP, and satellite's visibility over time.

The mean of GDOP in Table 3 quantifies the distribution characteristics of the satellites according to five types of satellite signals. As stated, the analysis was very convenient using various variables in the API of GNSS software receiver, which produced the measurement experiment results about the program that was described in the above. Overall, the performance results using the API were lower than that of

Table 3. Position solution errors.

	API						SX3
	GPS L1 C/A	GPS L2 P(Y)	Galileo E1	GLONASS L1 C/A	Beirou B1	All GNSS	Receiver solution (All GNSS)
Mean of horizontal error [m]	0.83	6.51	4.00	3.92	6.31	5.37	1.95
Mean of vertical error [m]	5.27	10.45	0.08	2.91	7.44	11.13	4.48
Mean of position error [m]	5.35	12.39	21.57	7.02	10.03	12.73	5.01
Mean of GDOP [m]	2.99	3.00	5.90	2.45	4.99	1.24	
Mean of visibility	10	9.9	4.5	7.7	8	30	

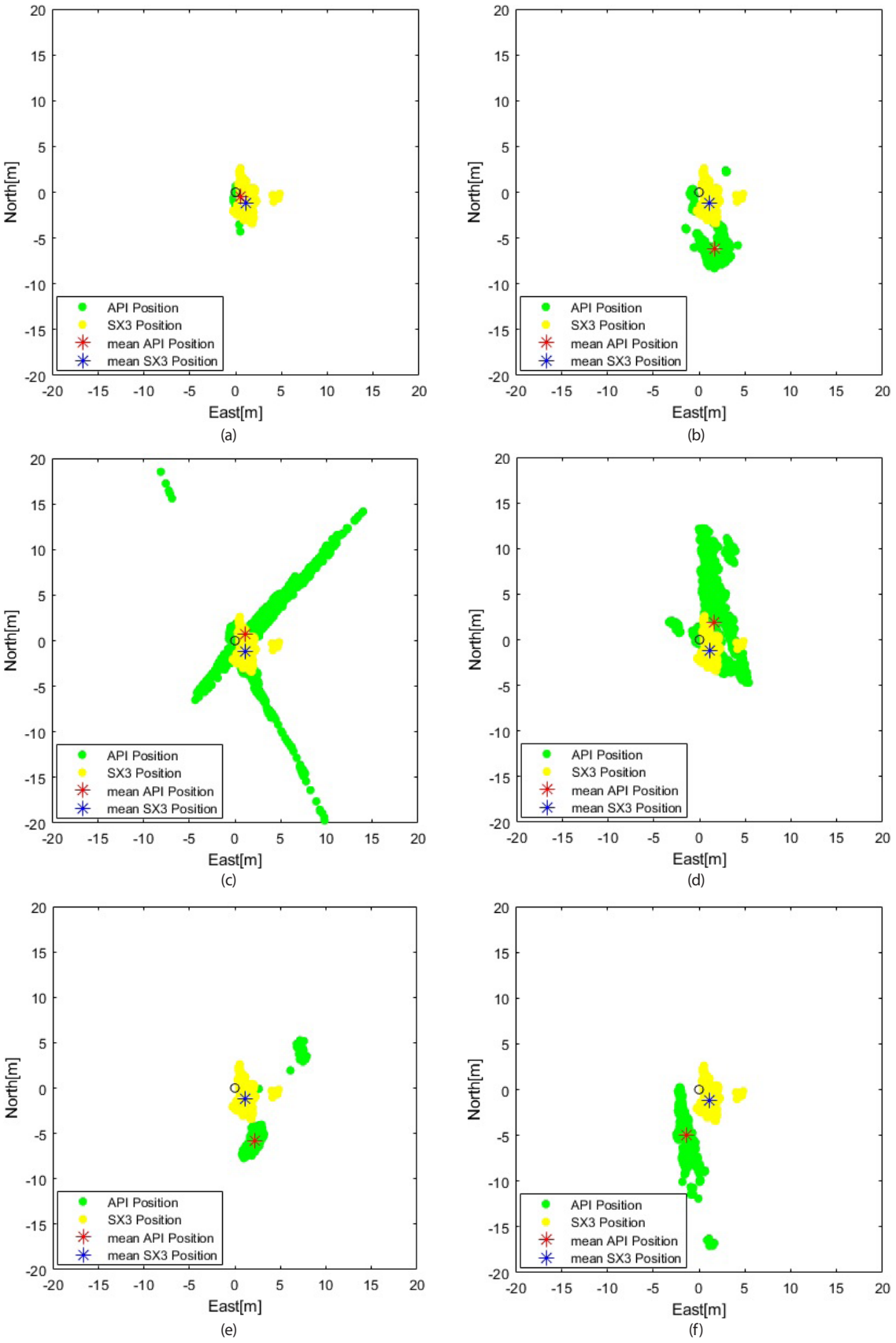

Fig. 7. Position result of API program. scale: ±20 [m].

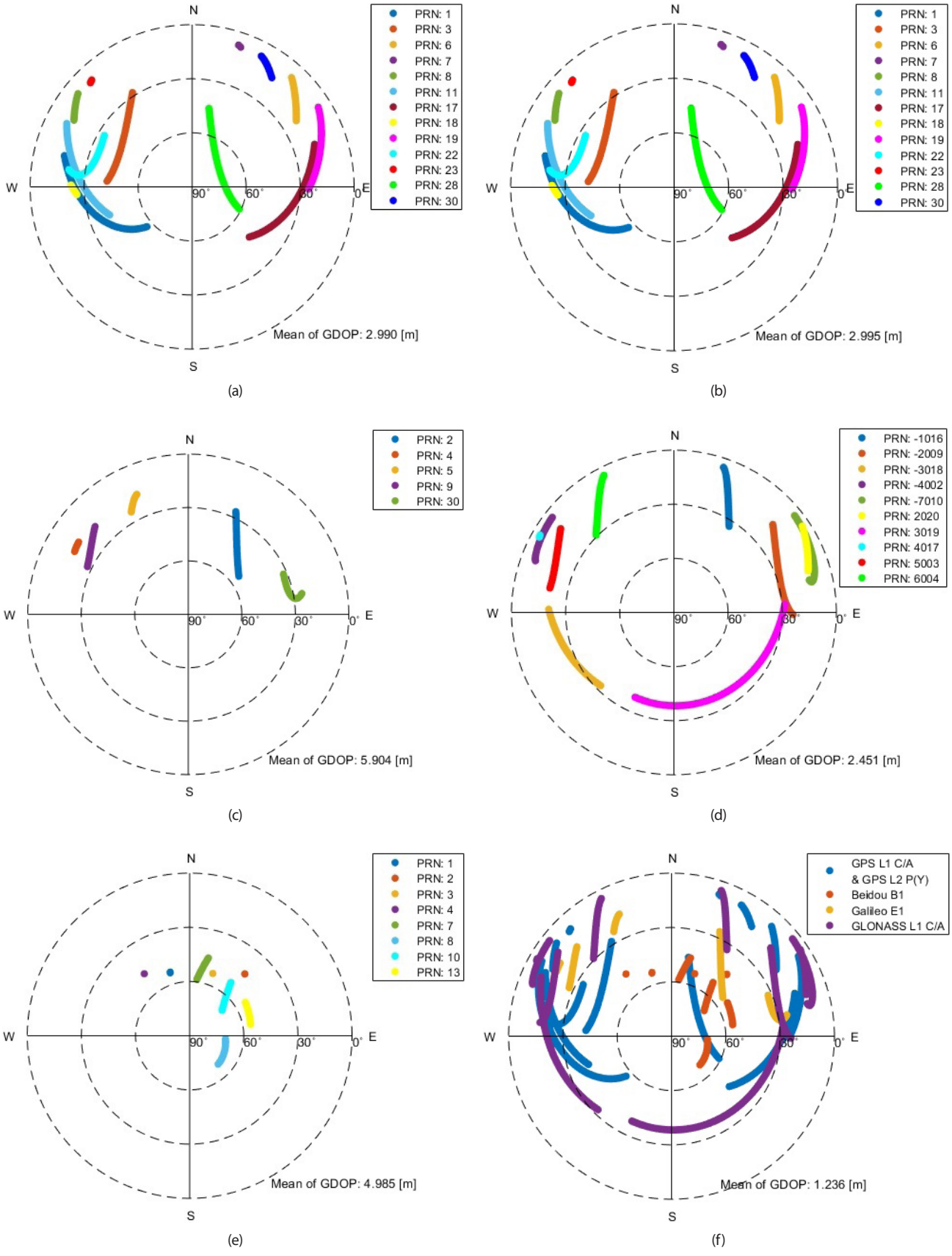

Fig. 8. Sky plot of API program.

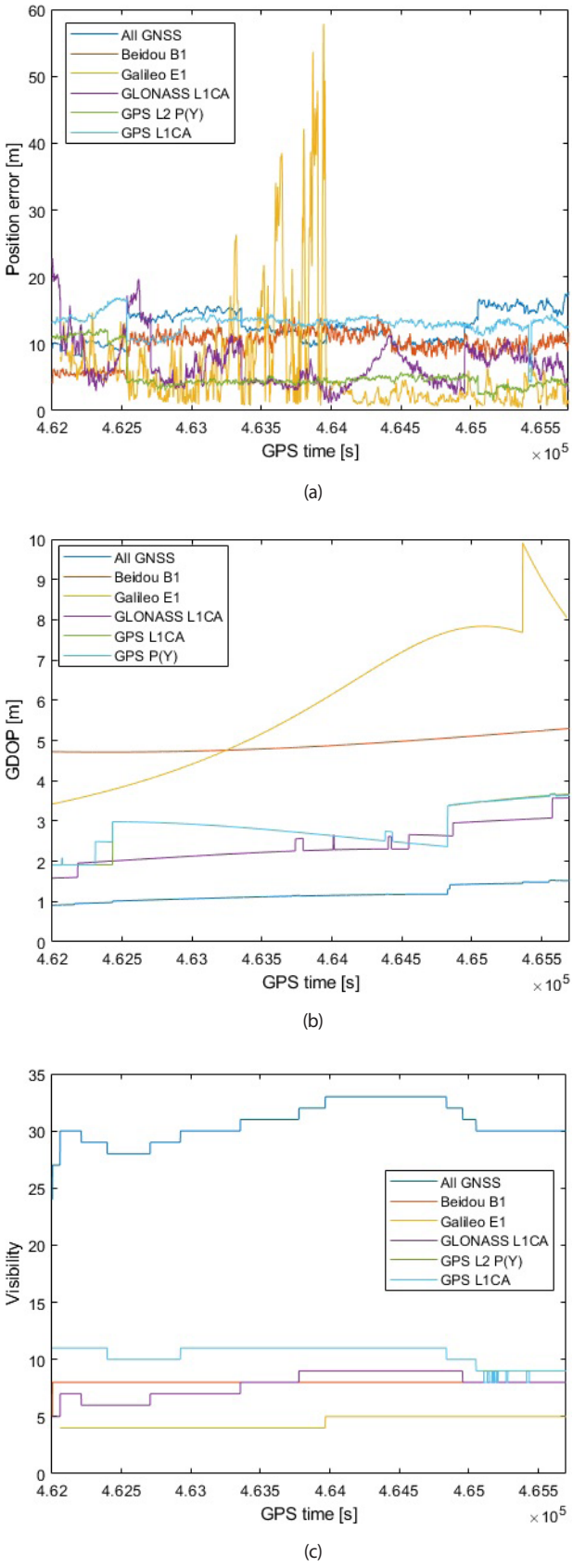

Fig. 9. Result of API program.

SX3. In addition, the position error of the GPS using the API was the lowest as 5.35 m. The horizontal errors of the GPS and GLONASS were relatively low because the satellites were arranged at various angles around the receiver.

In Fig. 7, the navigation results of each system exhibited a level of few meter error, but the trend of the variance differed according to each satellite. The GPS L1 C/A in Fig. 7a and GPS L2 P(Y) in Fig.7b show that the GPS had a small variance of error, but the mean positions of the two signals differed, which were due to the difference in the data reflected by the SPP flag, the criterion that determined whether signals were reliable. Since the satellites of GLONASS L1 C/A in Fig. 7d was distributed in a multi-direction compared to those of Galileo E1 in Fig. 7c, the variance was smaller than that of Galileo E1. The satellites of BeiDou B1 in Fig. 7e were displayed in many positions which were quite far away from the mean position. It meant the increase in variance. Fig. 7f shows the integration of all signals of GPS L1 C/A, GPS L2 P(Y), Galileo E1, GLONASS L1 C/A, and BeiDou B1, which was derived from the results that reflected the means and error characteristics of five types of satellite signals.

The precision of the navigation solution is related with the satellite arrangement. The sky plots of GPS L1 C/A in Fig. 8a and GPS L2 P(Y) in Fig. 8b verified that the satellite arrangement was distributed evenly at various angles overall. In contrast, the satellites of Galileo E1 in Fig. 8c were arranged only in the north direction, which diminished the precision of the navigation solution. Some satellites of GLONASS L1 C/A in Fig. 8d and BeiDou B1 in Fig. 8e were arranged in the south but their numbers were smaller than those arranged in the north. Thus, the variance range was larger than that of GPS L1 C/A.

The GPS revealed a relatively lower error rate than that of GLONASS, which was due to the higher visibility of the GPS than that of GLONASS based on the current satellites floating in space orbit. When analyzing Galileo with only average data in Table 3, relatively low horizontal and vertical errors were exhibited. However, as shown in Figs. 9a and b, its error rate and GDOP were high overall, which meant that the deviation of the value was large. This was due to the low satellite visibility caused by the delay of the Galileo program development, which could not secure the sufficient number of satellites currently. The visibility of all GNSS was 30, which was the highest, and the GDOP was 1.24 m accordingly, which was the lowest. This concluded that the value's deviation was very small. Although the position error was relatively high, all GNSS had relatively higher robustness than that of any other single GNSS due to the visibility of the largest number of satellites. The above study results verified that the reasonable outcomes on multi-frequency

and multi-constellation GNSSs could be derived through the analysis on measurement results within a short period of implementation time as 30 days with two researchers, which proved the reduction of research effort via the API.

6. CONCLUSIONS

The API of the GNSS software receiver is the architecture with a flexible structure. This paper was aimed at implementing the navigation unit using the API. It implemented a multi-frequency and multi-constellation GNSS receiver with small research efforts within a short period of time. It also presented the measurement results in Korea regions. The performance evaluation was concluded that the accuracy of the system implemented with the API was relatively lower than that of SX3. Nonetheless, the multi-frequency and multi-constellation GNSS required a great study efforts due to the drawback that was difficult to handle flexibly although it was more robust than single GNSS. To overcome this, this study proposed a flexible method through the API platform. Our implementation showed that the cost and time required to develop GNSS receivers could be dramatically reduced. As described in the above, the flexibility of the API platform can be a great benefit in terms of a convenient analysis as well as a convenience of performance upgrade of the system and scalability of the integrated receiver.

ACKNOWLEDGEMENT

This research was supported by the Space Core Technology Development Program of the National Research Foundation (NRF) funded by the Ministry of Science & ICT, S. Korea (No. NRF-2017M1A3A3A02016715)

AUTHOR CONTRIBUTIONS

Han, J.S. and Won, J.H. contributed to the design and implementation of the research, to the analysis of the results and to the writing of the manuscript.

"Conceptualization, Han, J.S and Won, J.H.; Methodology, Han, J.S; Software, Han, J.S; Validation: Han, J.S; Formal analysis, Han, J.S. and Won, J.H.; Investigation: Han, J.S; Resources: Han, J.S and Won, J.H.; Data curation, Han, J.S.; Writing-original draft Preparation: Han, J.S.; Writing-review and editing: Han, J.S. and Won, J.H.; Visualization, Han, J.S.; Supervision, Won, J.H.; Project administration, Won, J.H.; Funding Acqusition, Won, J.H.;

REFERENCES

Anghileri, M., Pany, T., Guixens, D. S., Won, J. H., Ayaz, A. S., et al. 2007, Performance Evaluation of a Multi-frequency GPS/Galileo/SBAS Software Receiver, Proceedings of the 20th ION GNSS International Technical Meeting of the Satellite Division, Sep 25-28, 2007, Fort Worth, TX, pp.2749-2761

Abbasian, N. S. & Petovello, M. G. 2008, Multichannel Dual Frequency GLONASS Software Receiver, Proceedings of the 21st ION GNSS International Technical Meeting of the Satellite Division, Sep 16-19, 2008, Savannah, GA, pp.1719-1729

Chen, Y. H. & Juang, J. C. 2007, A GNSS Software Receiver Approach for the Processing of Intermittent Data, Proceedings of the 20th ION GNSS International Technical Meeting of the Satellite Division, Sep 25-28, 2007, Fort Worth, TX, pp.2772-2777

Ebner, H. 2014, European GNSS evolution, EU Global Navigation Satellite System (GNSS) Research and Technology, HR2020 Stakeholder Consultation Workshop, Brussels, 4 Jun 2014

Heinrichs, G., Restle, M., Dreischer, C., & Pany, T. 2007, NavX - NSR- A Novel Galileo/GPS Navigation Software Receiver, Proceedings of the 20th ION GNSS International Technical Meeting of the Satellite Division, Sep 25-28, 2007, Fort Worth, TX, pp.1329-1334

IFEN GmbH 2016, SX3 Navigation Software Receiver, User Manual version 3.2 (Poing, Germany: IFEN GmbH)

Mattos, P. G. & Pisoni, F. 2013, Quad Constellation Receiver - GPS, GLONASS, Galileo, BeiDou, Proceedings of the 26th ION GNSS International Technical Meeting of the ION Satellite Division, Sep 16-20, 2013, Nashville, TN, pp.176-181

Misra, P. & Enge, P. 2011, Global Positioning System: Signals, Measurements, and Performance, 2nd ed. (Massachusetts: Ganga-Jamuna Press)

Mendel, J. M. 1995, Lessons in Estimation Theory for Signal Processing, Communications, and Control, 1st ed. (New Jersey: Prentice-Hall International, Inc.)

Wu, C., Qian, Y., Cui, X., & Lu, M. 2009, The Optimized Method and Algorithms in the CPU&GPU-Based GNSS Software Receiver, Proceedings of the 22nd ION GNSS International Technical Meeting of the Satellite Division, Sep 22-25, 2009, Savannah, GA, pp,339-343

APPENDICES

1. Pseudo code: NLSE

```
Algorithm NLSE
Input: Initial Position, H matrix, Pseudorange,
       Signal Power, Tropo Error, Iono Error, Clock Error,
       GPS Time
Output: Estimation Position
Estimation_Position = Initial_Position
Estimation_Position = Initial_Position
While delta < 10^-5
    e = Pseudorange – H_matrix * Initial_Position
    G = inv( H_matrix_transposed * Weighting * H_matrix )
        // inv function : Inverse matrix.
    delta = G * H_matrix_transposed * W * e
    Estimation_Position = Estimation_Position – delta
EndWhile
Return Estimation_Position
```

2. Pseudo code: API Program

```
Algorithm Processing Channel Measurement Data
Input: Satellite Position, Pseudorange, Signal Power,
       Tropo Error, Iono Error, Clock Error, GPS Time
Output: Position (Infinite loop)
While 1
    IF First_Process Then
        Initial_Position = function Initial_Position
                           (Satellite_Position)
    IF SPP OK Then
        n = The_Number_of_data
        H_matrix[4][n] = {Satellite_Position_array(X,Y,Z,1)}
        H_matrix = function Integrity_check(H_matrix)
        Estimation_Position = function NLSE
                              (Initial_Position, H_matrix,
                               Pseudorange, Signal_Power,
                               Tropo_Error, Iono_Error,
                               Clock_Error, GPS_Time)
        Initial_Position = Estimation_Position
    EndIF
EndWhile
```

Jin-Su Han received the B.S. degree from Inha University, Korea, in 2017. He is currently in the M.S. degree course in Department of Electrical Engineering at Inha University, Korea. His research interests include software receiver and INS/DR.

Jong-Hoon Won received the ph.D.degree in the Department of Control Engineering from Ajou University, Korea, in 2005. After then, he had worked with the Institute of Space Application at University Federal Armed Forces (UFAF) Munich, Germany. He was nominated as Head of GNSS Laboratory in 2011 at the same institute, and involved in lectures on advanced receiver technology at Technical University of Munich (TUM) since 2009. He is currently an assistant professor of the Department of Electrical Engineering at Inha University. His research interests include GNSS signal design, receiver, navigation, target tracking systems and self-driving cars.

https://doi.org/10.11003/JPNT.2019.8.1.1