Android Raw GNSS Measurement Datasets for Precise Positioning

Guoyu (Michael) Fu, Mohammed Khider, Frank van Diggelen, *Google Inc.*

Biography (IES)

Guoyu (Michael) Fu is a software engineer in the Android GNSS team at Google. His works aim to provide precise and robust positioning performance in both highway and urban environments for billions of Android devices. He received a Ph.D. degree in computer engineering from Texas A&M University in 2019, and a B.E. degree in automation from Nanjing University in 2013.

Mohammed Khider is a Technical Lead and Manager at Google's Android Location & Context team. Within Android, he is a member of a team that works on improving positioning and navigation accuracy of mobile devices in challenging indoor and urban canyon environments. He received his PhD in Communication Engineering with focus on "Multisensor based Positioning for Pedestrian Navigation" from the University of Ulm, Germany. Prior to his role at Google, he was a Research Associate at the Institute of Communication and Navigation at the German Aerospace Center (DLR) where he worked on various research projects related to positioning, navigation and context aware computing.

Frank van Diggelen is a Principal Engineer at Google, where he leads the Android GNSS team. Previously he worked at Broadcom and Global Locate where he and a small group of friends created the technology that put GNSS in smartphones. Their chips were the first GPS in any smartphone, the first in iPhone, the first GPS+GLONASS, and the first full GNSS in phones. He has over 90 issued U.S. patents on A-GNSS, and is the author of the textbook: "A-GPS". He received his Ph.D. at Cambridge University, and he teaches GPS at Stanford University.

Abstract

Raw GNSS measurement APIs (Application Programming Interfaces) in the Android system are a bridge between the advances of positioning techniques and the applications of positioning for billions of Android users in the mass market. The publication trend since the debut of these APIs in 2016 has shown an increasing attention from both academia and industry. Among all GNSS topics, high precision positioning on smartphones may be the most attractive one, not just because it spawns over 1/3 of publications, but more importantly, it may trigger a plethora of possible novel applications, such as lane-level driving, mapping, and precise augmented reality. To facilitate this process, in this paper, Google releases 60+ datasets collected from phones in the Android GPS team, together with corrections from SwiftNavigation Inc. and Verizon Inc. These datasets were collected on highways in the US San Francisco Bay Area in the summer of 2020. They will be released at g.co/gnsstools by November 2020. Ground truth files collected by a NovAtel SPAN system will also be provided in NMEA file format. Furthermore, Google also provides the source codes in Matlab to evaluate the positioning performance. This paper provides details of these datasets, corrections, ground truth files, and utilities.

1. Introduction

Since Google made GNSS raw measurement outputs publicly available in Android N, in 2016, the raw measurements have been an important source for the development of smartphone positioning algorithms and engines. As shown in Figure 1, out of almost 6000 published research papers on raw GNSS APIs [1], over 2000 papers focus on precise positioning algorithms such as PPP (Precise Point Positioning) and RTK (Real Time Kinematic) [2]. Most of them were published in the last 3 years, thanks to the launch of dual-frequency GNSS chips in phones, and recent improvements of smartphone antennas in some models. The trend indicates that decimeter or even centimeter positioning service could be the next major milestone of the smartphone GNSS industry. Such an innovation may have a profound impact on the daily lives of billions of smartphone users, and may enable unprecedented applications like lane-level driving navigation.

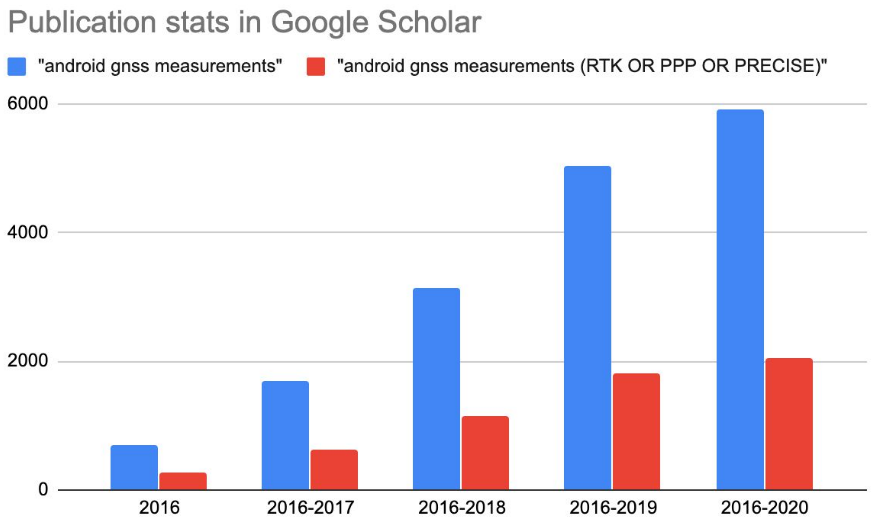


Figure 1 Recent publication stats about Android GNSS in Google Scholar [1-2]

In this paper, we release raw GNSS measurements collected by the Android GNSS team using test Android devices, and their ground truth trajectories when driving on highways, so that researchers and developers can use them as the benchmark for their own positioning engines. Moreover, we also release useful utility tools such as an evaluation script to evaluate customized position outputs. With these benchmark datasets and utilities, one can process the RINEX observation files or Google's GnssLogger files (also referred to as GnssLog), output the position solutions into NMEA files, and finally generate result metrics.

We provide scripts and tools for researchers to analyze the data and evaluate the results. For instance, a script is provided to compare two NMEA files and compute the key metrics of position solutions, which include the time to 1 meter (TT1M), accuracy at 1 minute (AA1M), 50-percentile and 95-percentile horizontal distance error, cross-track error and along-track error. The generated result table can be shared to compare with peers' results on a consistent basis.

The datasets and tools will be released and announced in g.co/gnsstools by November 2020. With the benchmark datasets and test suites, we hope to free GNSS researchers and developers from experimenting in different settings and places, collecting data in different formats, and evaluating them in different metrics. They can then invest more time on the development of new algorithms (as depicted in Figure 2) to tackle the challenges brought by phone GNSS antennas, and satisfy the requirements of efficiency, simplicity, and generality of the massive variety and number of smartphones. As the research advances, we will add more datasets and utilities to the pool, to address new requirements and challenges.

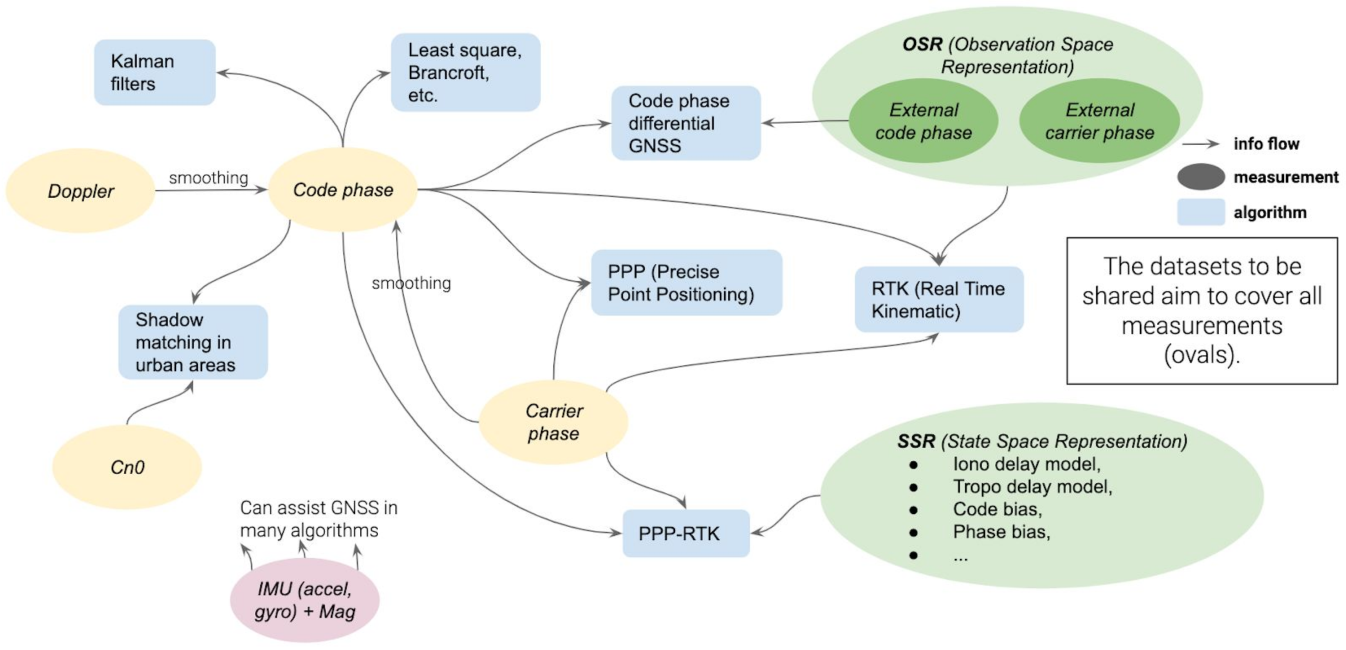

Kalman filters
Least square, Brancroft, etc.
Code phase differential GNSS
OSR (Observation Space Representation)
External code phase
External carrier phase
info flow
measurement
algorithm
Doppler
smoothing
Code phase
smoothing
PPP (Precise Point Positioning)
RTK (Real Time Kinematic)
The datasets to be shared aim to cover all measurements (ovals).
Shadow matching in urban areas
Carrier phase
Cn0
SSR (State Space Representation)
Iono delay model,
Tropo delay model,
Code bias,
Phase bias,
...
Can assist GNSS in many algorithms
PPP-RTK
IMU (accel, gyro) + Mag

Figure 2 GNSS measurements (ovals) and relationships with common positioning algorithms

2. Literature Review and Existing Tools

Over the past 4 years, high precision positioning in Android phones has grown as a booming topic in academia. In 2016, Google implemented APIs to make raw GNSS measurements collected by smartphone's GNSS chipset and antenna available to all Android phones with N or higher versions. Google also provided an App - GnssLogger and a Matlab program - GnssAnalysisTools as example applications to log and process raw GNSS measurements, respectively [3]. Discussions about the feasibility of decimeter- or even centimeter-level smartphone positioning abound in literature. Humphreys et al are ones of the earliest researchers collecting data from smartphones, and share their insights in [4]. They pointed out 5 anomalies in phase observations and concluded that most of them can be recognized and remedied, but local multipath is a challenge to the phase quality.

In 2017, Realini et al [5] used the goGPS software to process double differenced phase observation of GPS L1, and concluded that decimeter-level error is achievable when the phone is stationary. In the same year, an acclaimable result was presented by Pirazzi et al [6], where the trajectory of a smartphone writes 6 English characters - "VADASE" on A4 papers. This relative positioning demo gives confidence to the readers that the quality of phase measurement is accurate at centimeter-level.

2018 is another remarkable year for the history of smartphone GNSS, where the world's first dual-frequency phone (Xiaomi Mi8 with Broadcom 4775) made its debut. Google introduced new APIs (related to Accumulated Delta Ranges, or ADR) in Android to enable high precision GNSS positioning [7]. We updated the GnssAnalysisTools in [8] accordingly to enable Android users to anatomize the raw characteristics of GNSS measurements.

In 2019, the number of relevant publications increased rapidly, and they showed more promising positioning results. Chen et al [9] tested PPP without external correction on a stationary Xiaomi Mi8 and received sub-meter RMS (Root Mean Squared) horizontal error within 30 seconds. Li and Geng [10] modified a Nexus 9 by connecting to an external antenna, and its 3D positioning error is 0.5m or less for 100% of time. It's worth mentioning here that this team achieves <3cm for a Mi8 phone using the PPP-AR algorithm this year in 2020 [11]. Darugna from Geo++ Inc. put a smartphone on top of a choke ring antenna dragging by a train on track [12]. The rejection of ground reflected multipath makes it an excellent demonstration of centimeter positioning on a phone in motion. Liu and his colleagues in FMI Inc. implemented an RTK+IMU algorithm on Mi8, and tested it with the phone on the dashboard of a driving car in a suburban area [13]. They obtained only 1 meter error horizontally at the 95-percentile for two trial drives. These and more unmentioned publications and presentations jointly become a strong signal of the feasibility of sub-meter positioning for smartphones with in-phone antenna and centimeter positioning for smartphones with external antenna.

Techniques and innovations about high precision GNSS positioning in general receivers are established for decades and continue to push the limits of GNSS accuracy and robustness. Many useful packages are available online. For example, NGS maintains a website listing all open-source GNSS packages (https://www.ngs.noaa.gov/gps-toolbox/exist.htm). For high-precision GNSS, we use Table 1 to list some relevant open-source libraries and closed-source tools that can serve as references for the users to build on and process the released datasets. There are some famous online services (APPS from Jet Propulsion Lab [25], GAPS at University of New Brunswick [26], CSRS-PPP from Natural Resources Canada [27], OPUS from National Geodetic Survey [28], CenterPoint RTX from Trimble [29], etc.) that are not listed in this table, but are worth for the readers to explore as well.

Table 1 Some libraries and tools for high-precision GNSS positioning

Organization / Author	Name	Last Update	Open sourced?	Programming language	Constellations	Receiver motion	Core	Link to source codes / download
Tomoji Takasu	RTKLIB [14]	2019-08	Yes	C	GPS, GLO, GAL, QZS, BDS, SBAS	Static, dynamic	WLS, RTK, PPP, NTRIP client/server	http://www.rtklib.com/
Tim Everett	RTKLIB Demo5 [15]	2020-02	Yes	C	GPS, GLO, GAL, QZS, BDS, SBAS	Static, dynamic	WLS, RTK, PPP, NTRIP client/server	https://rtkexplorer.com/
Shandong University of Science and Technology	GAMP [16]	2018	Yes	C	GPS, GLO, GAL, QZS, BDS, SBAS	Static, dynamic	PPP with undifferenced and uncombined observations	https://geodesy.noaa.gov/gps-toolbox/GAMP.htm
GoGPS Inc.	goGPS [17]	2020-09	Yes	Matlab	GPS, GLO, GAL, QZS, BDS, IRNSS	Static, dynamic	Bancroft, WLS, PPP	https://github.com/goGPS-Project/goGPS_MATLAB
Chinese Academy of Sciences, Shandong University of Science and Technology	MG-APP [18]	2020-02	Yes	C++	GPS, GLO, GAL, BDS	Static, dynamic	WLS, PPP, Support single and multiple stations, KF, Square Root Information Filter, Pseudorange smoothing, batch processing	https://github.com/XiaoGongWei/MG_APP
Wuhan University	PRIDE PPP-AR [19]	2019	Yes	Fortran 95	GPS	Static, dynamic	undifferenced PPP with ambiguity resolution; based on least-square.	https://geodesy.noaa.gov/gps-toolbox/PRIDE.htm
Hacettepe University, Turkey	PPPH [20]	2018	Yes	Matlab	GPS, GLO, GAL, BDS	Static, dynamic	PPP	https://geodesy.noaa.gov/gps-toolbox/PPPH.htm
ARL at UT-Austin	GPSTk [21]	2020	Yes	C++	GPS, GLO, GAL, QZS, BDS, SBAS	Static, dynamic	PPP	https://gitlab.com/sgl-ut/GPSTk
European Space Agency (ESA)	gLAB [22]	2019-05	No	Windows, Linux, Mac	GPS, GLO, GAL, QZS, BDS, SBAS, IRNSS	Static, dynamic	PPP, DGNSS, WLS	https://gage.upc.edu/gLAB/
Yize Zhang	Net_Diff [23]	2020-07	No	Windows, and online service	GPS, GLO, GAL, QZS, BDS, IRNSS	Static, dynamic	PPP, RTK, DGNSS, WLS	https://github.com/YizeZhang/Net_Diff
Rokubun Inc.	Jason [24]	2020	No	Online service	GPS, GAL	Static, dynamic	PPP, RTK, WLS	https://jason.rokubun.cat/#!/
Google Inc.	GPS Measurement Tools [3]	2016	Yes	Java, Matlab	GPS	Static, Dynamic	WLS	https://github.com/google/gps-measurement-tools

2. Datasets

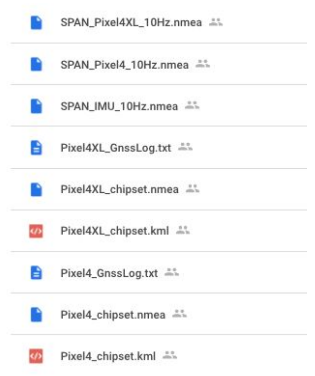
2.1 Overview

Specifically, the benchmark datasets include raw GNSS measurement and raw inertial sensors' readings, using several dual-frequency and ADR (Accumulated Delta Range) enabled smartphones (Xiaomi Mi 8, Google Pixel 4, etc.) in driving scenarios, collected in the US San Francisco Bay Area. This section explains the definitions of raw GNSS values and inertial readings in detail, so that the readers can easily interpret them using standard terms defined in a RINEX observation file. Each dataset comes with a ground truth NMEA file collected by NovAtel SPAN system in a time synchronized manner. The ground truth file has compensated the lever arm offset and is referenced to the smartphone itself. RINEX observation files from nearby reference stations are provided for each dataset, so that a standard RTK algorithm can be performed as a starter. SSR (State Space Representation) corrections in the nearby area are also provided to enable PPP algorithms. For better performance, we encourage the readers to also try observation files or correction products from public reference networks (e.g. CORS) or their proprietary correction service. Datasets and corrections to be released are organized in the following directory structure (Table 2).

Table 2 Directory structure and example of released datasets

- **Datasets/YYYY-MM-DD-{Country code in 2 letters}-{Region}-{Collection index}** (e.g. 2020-05-14-US-MTV-1)
  - **{Phone name}_GnssLog.txt**: GnssLogger log file
  - **SPAN_{Phone name}_{Report rate}.nmea**: Ground truth positions at the phone's position
  - **{Phone name}_chipset.nmea**: Chipset positions in NMEA format
  - **{Phone name}_chipset.kml**: Chipset positions in KML format
- **Corrections/YYYY-MM-DD-{Country code in 2 letters}-{Region}-{Collection index}/**
  - **OSR/**: RINEX files of nearby base stations by Verizon
  - **SSR/:** SwiftNav SSR corrections in JSON

Example:



SPAN_Pixel4XL_10Hz.nmea
SPAN_Pixel4_10Hz.nmea
SPAN_IMU_10Hz.nmea
Pixel4XL_GnssLog.txt
Pixel4XL_chipset.nmea
Pixel4XL_chipset.kml
Pixel4_GnssLog.txt
Pixel4_chipset.nmea
Pixel4_chipset.kml

2.2 GnssLog Format

The GnssLog file contains raw GNSS measurements, location fixes, and motion sensors (accelerometer, gyroscope, magnetic field) collected in the entire trace. The format is a text based Comma-Separated Values (CSV) file. A GnssLog has the header part defining each type of message in the front, followed by the content part containing measurements at each row. With this format, one can conveniently import a GnssLog file into Google Sheet, and perform statistical analysis to obtain basic metrics, such as the number of total measurements, or the ratio of L1 to L5 signal counts, etc.

GnssLog format is different from the standard GNSS format - RINEX in the following aspects:

1. GnssLog contains location fix results and motion sensor readings whereas RINEX does not.
2. GnssLog contains more unprocessed GNSS measurement. It provides signal arrival and transmit times while RINEX directly provides the distance equivalent of their difference - pseudorange. Moreover, from GnssLog, one has deeper insights of the receiver status - clock states and uncertainties, CN0 values at antenna and baseband port, etc.
3. RINEX has precision restrictions, which forces the measurement and times to be rounded to fit the digit-length requirement. For example, the epoch arrival time in RINEX must be a multiple of 100 nanoseconds. In GnssLog, these values can be freely described.

To preserve these advantages, this dataset features the GnssLog format. Considering that most existing positioning engines take RINEX format as inputs, we also provide RINEX observation files converted from GnssLog, as detailed in Section 2.3.

a) Message Types and Content

Table 3 details 5 types of messages contained in each GnssLog. Each message has up to tens of fields that match the APIs defined in Android HAL (Hardware Abstraction Layer) or Google Play Services (e.g. Fused Location Provider). Each message is written in the sequence as their arrivals in the Android App level, which might be inconsistent with the ordering of physical events. Therefore, we put extra timestamps in each message, for developers to synchronize these measurements as close as possible. Among these messages, the "Raw" message is the most complicated and therefore is further detailed in the next section.

Table 3 Message Types in GnssLog

Message Type (Prefix)	Description	Report Rate	Android API Source
"Raw"	Each line contains the raw GNSS measurements of one signal at one epoch.	N times per second, where N is the number of received signals	GnssClock, GnssMeasurement
"UncalAccel" or "Accel"	Each line contains the uncalibrated accelerometer/gyroscope/magnetic field reading at one epoch. The "utcTimeMillis" field is the sum of "elapsedRealtimeNanos" and the estimated device boot time at UTC, after a recent NTP (Network Time Protocol) sync.	100-400 Hz	SensorEvent#TYPE_ACCELEROMETER_UNCALIBRATED
"UncalGyro" or "Gyro"			SensorEvent#TYPE_GYROSCOPE_UNCALIBRATED
"UncalMag" or "Mag"			SensorEvent#TYPE_MAGNETIC_FIELD_UNCALIBRATED
"Fix"	Each line contains a location computed from GNSS chipset ("GPS"), Network Location Provider (NLP), Fused Location Provider (FLP).	1Hz for each kind of location	LocationManager#GPS_PROVIDER, FusedLocationProviderClient, LocationManager#NETWORK_PROVIDER
"Status"	Each line contains the status of one signal at one epoch. The "UnixTimeMillis" field equals the latest UTC time of a GPS location.	≥N times per second, where N is the number of received signals. Depending on the GNSS chipset manufacturer, GnssStatus messages could be reported more than once per second.	GnssStatus

b) "Raw" Message

Table 4 explains the API sources of each field in the "Raw" message, and also provides tips about their relationship to common GNSS concepts, and typical abnormal behaviors. These tips are merely basic checks. To design an effective positioning engine, one should be aware that GNSS measurements by a smartphone are much noisier, and of higher error-rate, than that by a commodity-grade GNSS receiver even with an inexpensive patch antenna. A thorough investigation of error characteristics is necessary and the resultant outlier rejector should have more sophisticated checks. More explanations about Android raw GNSS APIs can be found in [31][32].

Table 4 Explanations of each field recorded in the RAW message

Index	Field name	Definition	Description
0	**"Raw"**	Prefix of sentence	
1	**utcTimeMillis**	Milliseconds since UTC epoch (1970/1/1), converted from GnssClock	
2	**TimeNanos**	GnssClock#getTimeNanos()	local estimate of GPS time (signal arrival time) = TimeNanos - (FullBiasNanos + BiasNanos). Measurements should be discarded if: ● FullBiasNanos is zero or invalid ● Arrival time is negative or unrealistically large ● TimeNanos is empty
3	**LeapSecond**	GnssClock#getLeapSecond()	
4	**TimeUncertaintyNanos**	GnssClock#getTimeUncertaintyNanos()	
5	**FullBiasNanos**	GnssClock#getFullBiasNanos()	
6	**BiasNanos**	GnssClock#getBiasNanos()	
7	**BiasUncertaintyNanos**	GnssClock#getBiasUncertaintyNanos()	Quality indicators of receiver clock at current epoch Measurements should be discarded if: ● BiasUncertaintyNanos is too large
8	**DriftNanosPerSecond**	GnssClock#getDriftNanosPerSecond()	
9	**DriftUncertaintyNanosPerSecond**	GnssClock#getDriftUncertaintyNanosPerSecond()	
10	**HardwareClockDiscontinuityCount**	GnssClock#getHardwareClockDiscontinuityCount()	
11	**Svid**	GnssMeasurement#getSvid()	
12	**TimeOffsetNanos**	GnssMeasurement#getTimeOffsetNanos()	
13	**State**	GnssMeasurement#getState()	Measurements should be discarded if: ● State is neither STATE_TOW_DECODED nor STATE_TOW_KNOWN
14	**ReceivedSvTimeNanos**	GnssMeasurement#getReceivedSvTimeNanos()	Signal transmit time in satellite time scale. Its difference to signal arrival time is the time equivalent to pseudorange. Measurements should be discarded if: ● ReceivedSvTimeUncertaintyNanos is high
15	**ReceivedSvTimeUncertaintyNanos**	GnssMeasurement#getReceivedSvTimeUncertaintyNanos()	
16	**Cn0DbHz**	GnssMeasurement#getCn0DbHz()	
17	**PseudorangeRateMetersPerSecond**	GnssMeasurement#getPseudorangeRateMetersPerSecond()	Equivalent to Doppler measurement
18	**PseudorangeRateUncertaintyMetersPerSecond**	GnssMeasurement#getPseudorangeRateUncertaintyMetersPerSecond()	
19	**AccumulatedDeltaRangeState**	GnssMeasurement#getAccumulatedDeltaRangeState()	Carrier phase measurement. Measurements should be discarded if: ● AdrState violates this condition: ADR_STATE_VALID == 1 & ADR_STATE_RESET == 0 & ADR_STATE_CYCLE_SLIP == 0 ● AdrUncertaintyMeters is too large
20	**AccumulatedDeltaRangeMeters**	GnssMeasurement#getAccumulatedDeltaRangeMeters()	
21	**AccumulatedDeltaRangeUncertaintyMeters**	GnssMeasurement#getAccumulatedDeltaRangeUncertaintyMeters()	
22	**CarrierFrequencyHz**	GnssMeasurement#getCarrierFrequencyHz()	
23	**CarrierCycles**	GnssMeasurement#getCarrierPhase() Deprecated in API level 28 (Android P in 2018)	Obsolete APIs for carrier phase measurement. One should not use them for recent phones.
24	**CarrierPhase**	GnssMeasurement#getCarrierPhase() Deprecated in API level 28 (Android P in 2018)	
25	**CarrierPhaseUncertainty**	GnssMeasurement#getCarrierPhaseUncertainty() Deprecated in API level 28 (Android P in 2018)	
26	**MultipathIndicator**	GnssMeasurement#getMultipathIndicator()	
27	**SnrInDb**	GnssMeasurement#getSnrInDb()	
28	**ConstellationType**	GnssMeasurement#getConstellationType()	Measurements should be discarded if the constellation is unknown.
29	**AgcDb**	GnssMeasurement#getAutomaticGainControlLevelDb()	
30	**BasebandCn0DbHz**	GnssMeasurement#getBasebandCn0DbHz() Added in API level 30 (Android 11 in 2020)	It is different from the normal Cn0DbHz.
31	**FullInterSignalBiasNanos**	GnssMeasurement#getFullInterSignalBiasNanos() Added in API level 30 (Android 11 in 2020)	These fields are meant to provide convenience for developers. For standard positioning requirements, one can directly use these values without computing ISB by themselves.
32	**FullInterSignalBiasUncertaintyNanos**	GnssMeasurement#getFullInterSignalBiasUncertaintyNanos() Added in API level 30 (Android 11 in 2020)	
33	**SatelliteInterSignalBiasNanos**	GnssMeasurement#getSatelliteInterSignalBiasNanos() Added in API level 30 (Android 11 in 2020)	
34	**SatelliteInterSignalBiasUncertaintyNanos**	GnssMeasurement#getSatelliteInterSignalBiasUncertaintyNanos() Added in API level 30 (Android 11 in 2020)	
35	**CodeType**	GnssMeasurement#getCodeType() Added in API level 29 (Android 10 in 2019)	
36	**ChipsetElapsedRealtimeNanos**	GnssClock#getElapsedRealtimeNanos() Added in API level 29 (Android 10 in 2019)	This timestamp is designed to synchronize timestamps of other measurements (e.g. SensorEvents). Please note that many phones didn't populate this field until 2020.

2.3 RINEX File Converted from GnssLog

Table 5 lists the mapping between the key measurements defined in RINEX v3.03 [30] and Android raw GNSS measurement APIs. We convert all GnssLog files to standard RINEX observation files based on this mapping table. During conversion, we have to drop some readings (for example, clock uncertainties, half cycle slips), and adjust the measurements using Doppler to align with the floored signal arrival time. Therefore, RINEX content is a subset of GnssLog content.

Table 5 Mapping of observation values between RINEX v3.03 and Android GNSS APIs

RINEX Data [30]	**Android GnssMeasurementEvent APIs**
Epoch time	Process GnssClock to compute the signal arrival time ("arrivalTime"). Floor it to 100 nanosecond level (F11.7). Adjust measurements using Doppler.
Code pseudorange (C)	1. Process GnssClock to compute arrivalTime. 2. Use GnssMeasurement.getReceivedSvTime*() to compute satellite transmit time. 3. Compute raw pseudorange by subtracting the two times above.
Carrier phase (L) cycles (Required in cycles. Please refer Page 11 Section 3.3 [30])	As per ADR(m) = wavelength * ADR(cycles), GnssMeasurement.getAccumulatedDeltaRangeMeters() / wavelength
Doppler (D) in Hz (Table A2 in [30])	-1 * GnssMeasurement.getPseudorangeRateMetersPerSecond() / wavelength
Signal Strength (S) in dB-Hz	GnssMeasurement.getCn0DbHz()
Loss of Lock Indicator bit	ADR state is retrieved using GnssMeasurement.getAccumulatedDeltaRangeState() ● Blank = Unknown loss of lock status / ADR state is invalid or reset. ○ ADR state invalid or ADR state Reset ● 0 = OK ○ Default ● bit 1 (least significant bit) = loss of lock/cycle slip (observation should be discarded) ○ ADR state cycle slip ● bit 2 = half-cycle slip (observation should be discarded) ○ ADR state half cycle reported
Signal Strength Indicator bit	This is straightforward from Section 5.7 of [30]

2.4 Error Correction Data

For each trace, two kinds of error corrections are provided: Observation Space Representation (OSR) and State Space Representation (SSR). OSR files in RINEX format are collected by 6 reference stations owned by Verizon Inc., around the US San Francisco Bay Area. They enable users to adopt single-baseline RTK and small scale network RTK for precise positioning. SSR files in JSON format containing precise satellite orbit, clock, biases, and atmospheric corrections are provided by the Skylark network of SwiftNav Inc.

3. Ground Truth NMEA File

NovAtel SPAN system and Waypoint Inertial Explorer are used to generate the ground truth result, via the tightly coupling of IMU and GNSS measurements. To eliminate offsets caused by different coordinate frames and tectonic movement, the ground truth is all generated in WGS84 datum with all base stations' coordinates solved in the year of 2020. A NMEA file is provided in correspondence with each GnssLog file, to represent the reference position and velocity of the smartphone. 10Hz instead of 1Hz is adopted here to capture the acceleration of smartphones, especially at turns. It is worth mentioning that the lever arms from the SPAN system to smartphones have been compensated. Next section illustrates this process in the vehicle coordinate frame. Section 3.2 details the NMEA format.

3.1 Lever arm

Figure 3 shows how the smartphones are placed above the dashboard of the test vehicle. The SPAN IMU system is secured at the back passenger seat, right above which at the roof is the GNSS antenna of SPAN. Figure 4 illustrates this geometry using a top view (left diagram) and a front view (right diagram). The SPAN GNSS system solves position at the phase center of its antenna. The SPAN IMU speculates pose at the navigation center of the IMU system. But the position of a smartphone is the position of interest. Therefore, these three entities and their offsets (lever arms) must be carefully defined: SPAN GNSS antenna, SPAN IMU system, and the smartphone under test.

Let us first understand the vehicle coordinate frame defined by SPAN. The origin is at the navigation center of SPAN IMU (with marks on the box). The x-axis points to the right of the vehicle, y-axis to the front, and z-axis to the sky. Secondly, we can define the *IMU-to-Antenna lever arm* to be a (x, y, z) tuple from IMU navigation center to SPAN antenna's phase center. This lever arm is used when the Inertial Explorer performs tightly coupling IMU-RTK algorithm. With this lever arm applied, the output position refers to the IMU navigation center. Thirdly, we define the *IMU-to-smartphone lever arm* to be a (x, y, z) tuple from IMU navigation center to the smartphone's coordinate center (i.e. the geometric center of screen). This lever arm is used to translate reference positions to locate smartphone's.

In practice, we have compensated these lever arms. However, several centimeters of errors are inevitable because the lever arms are coarsely measured with regular rulers.

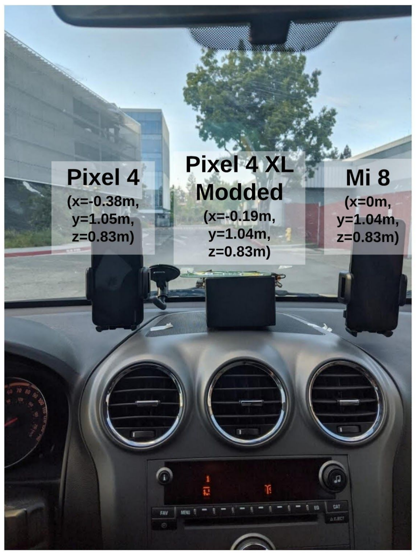


Figure 3 Example setup of phones on the vehicle dashboard

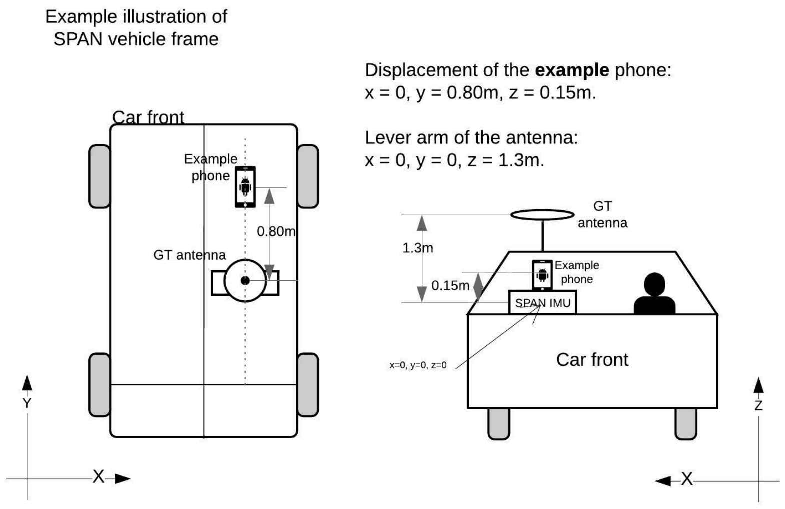
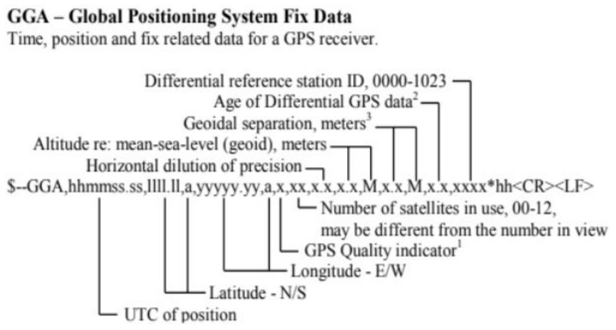

Example illustration of SPAN vehicle frame
Displacement of the **example** phone:
x = 0, y = 0.80m, z = 0.15m.
Lever arm of the antenna:
x = 0, y = 0, z = 1.3m.
Car front
Example phone
0.80m
GT antenna
Y
X
GT antenna
1.3m
0.15m
Example phone
SPAN IMU
x=0, y=0, z=0
Car front
Z
X

Figure 4 The ground truth system coordinate frame and lever arm examples

3.2 NMEA format

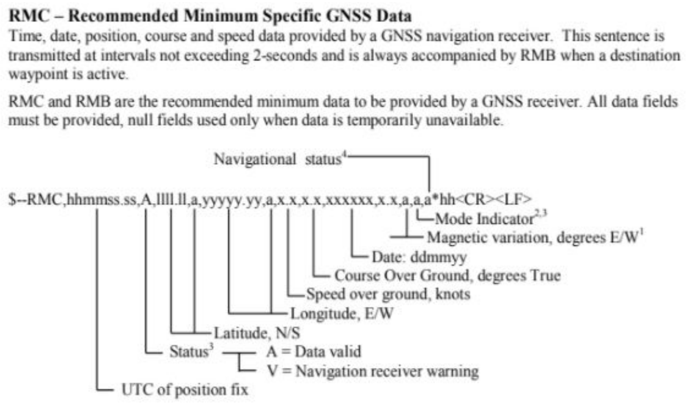
In the NMEA file, each location is described jointly by a pair of GGA and RMC sentences, as depicted in Figures 5 and 6. It is not a requirement by our NmeaUtils (detailed in the next section) to populate all fields in both sentences. Specifically, in GGA sentence, the latitude, longitude, and altitude are used, while in RMC sentence, the latitude, longitude, speed over ground, and course over ground are used. It's feasible with both sentences to project the location onto local plane, along-track, and cross-track directions.


GGA – Global Positioning System Fix Data

Time, position and fix related data for a GPS receiver.

Differential reference station ID, 0000-1023
Age of Differential GPS data[2]
Geoidal separation, meters[3]
Altitude re: mean-sea-level (geoid), meters
Horizontal dilution of precision
$--GGA,hhmmss.ss,llll.ll,a,yyyyy.yy,a,x,xx,x.x,x.x,M,x.x,M,x.x,xxxx*hh<CR><LF>
Number of satellites in use, 00-12, may be different from the number in view
GPS Quality indicator[1]
Longitude - E/W
Latitude - N/S
UTC of position

Figure 5 Format of NMEA GGA sentence [33]


RMC – Recommended Minimum Specific GNSS Data

Time, date, position, course and speed data provided by a GNSS navigation receiver. This sentence is transmitted at intervals not exceeding 2-seconds and is always accompanied by RMB when a destination waypoint is active.

RMC and RMB are the recommended minimum data to be provided by a GNSS receiver. All data fields must be provided, null fields used only when data is temporarily unavailable.

Navigational status[4]
$--RMC,hhmmss.ss,A,llll.ll,a,yyyyy.yy,a,x.x,x.x,xxxxxx,x.x,a,a,a*hh<CR><LF>
Mode Indicator[2,3]
Magnetic variation, degrees E/W[1]
Date: ddmmyy
Course Over Ground, degrees True
Speed over ground, knots
Longitude, E/W
Latitude, N/S
Status[3]
A = Data valid
V = Navigation receiver warning
UTC of position fix

Figure 6 Format of NMEA RMC sentence [33]

4. NmeaUtils Source Codes

The NmeaUtils are written in Matlab and opensourced, in order for most researchers and developers to evaluate the standard NMEA files generated by their positioning engines. Given an NMEA file of the device under test, together with a given ground truth NMEA file, two main functions can be called to analyze the positioning performance: *Nmea2RtkMetrics.m*, and *Nmea2ErrorPlot.m*.

4.1 Nmea2RtkMetrics

Table 6 gives an example of the output, as well as the definitions of positioning metrics. The first group of metrics is about the horizontal errors in 50-, 80-, and 95-percentiles of the Cumulative Distribution Functions (CDF). The second group measures the convergence performance, which is to answer "how quickly can the user feel the precise positioning performance?" The third group is horizontal positioning errors projected into cross-track (or x-track) and along-track directions. Cross-track error is the most interesting metric to lane-level mapping, traffic monitoring, and navigation. For example, 1.5m or less of the 95-percentile cross-track error over each trace is regarded as a good performance for lane-level applications.

Table 6 Example output of Nmea2RtkMetrics.m

NumValidPts	50% (m)	80% (m)	95% (m)	TT1M (s)	TA1M (s)	AA5S (m)	AA10S (m)	XTrack 50% (m)	XTrack 80% (m)	XTrack 95% (m)	AlongTrack 50% (m)	AlongTrack 80% (m)	AlongTrack 95% (m)
1,629	0.80	1.26	1.73	49.00	5.49	1.29	1.51	0.27	0.70	1.16	0.55	1.11	1.57

Metrics definitions:
NumValidPts: Number of valid solution points.
50% (m): 50-percentile of horizontal error (meters)
80% (m): 80-percentile of horizontal error (meters)
95% (m): 95-percentile of horizontal error (meters)
TT1M (s): Time (seconds) to 1-meter horizontal error
TA1M (s): Average consecutive time (seconds) of 1 meter or less horizontal error
AA5S (m): Horizontal accuracy (meters) at the 5th second
AA10S (m): Horizontal accuracy (meters) at the 10th second
X-track 50% (m): 50-percentile of cross-track horizontal error (meters)
X-track 80% (m): 80-percentile of cross-track horizontal error (meters)
X-track 95% (m): 95-percentile of cross-track horizontal error (meters)
Along-track 50% (m): 50-percentile of along-track horizontal error (meters)
Along-track 80% (m): 80-percentile of along-track horizontal error (meters)
Along-track 95% (m): 95-percentile of along-track horizontal error (meters)

4.2 Nmea2ErrorPlot.m

Figure 7 is an example output of Nmea2ErrorPlot program. This tool is useful for debugging positioning performance. In this example, the horizontal error at the 95-percentile is only 2.1m whereas it is 35.4m at the 100-percentile. It implies that there are at least 1 point where the positions are far away from the ground truth. Then one can zoom in the upper view to locate them.

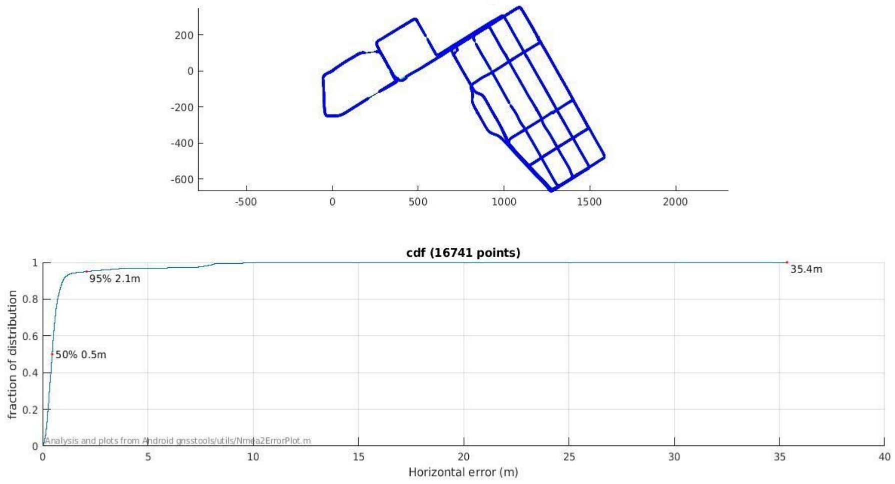


Figure 7 Example output of Nmea2ErrorPlot.m
(*Upper plot*: top view of trajectories horizontally. *Lower plot*: the cumulative distribution function of horizontal errors. *Unit*: meters)

Disclaimer

The measurement data we share in this paper, including sensor data and ground truth data, is collected from Google owned phones and truth-reference systems, by Google employed operators, in public areas. OSR and SSR correction data are collected by Verizon Inc., and SwiftNav Inc., respectively.

Acknowledgments

The authors would like to thank SwiftNav Inc. for providing SSR correction data, Verizon Inc. for OSR files, Shirish Chavan, Charles Zamora, Prasad Chitturi, Steve Malkos, and other Googlers who contributed to GNSS tools and datasets presented in this work.

References

1. Google Scholar, searches with "android gnss measurements", url: https://scholar.google.com/scholar?q=android+gnss+measurements+&hl=en&as_sdt=0%2C5&as_ylo=2016&as_yhi=2019
2. Google Scholar, searches with "android gnss measurements (RTK OR PPP OR PRECISE)", url: https://scholar.google.com/scholar?as_ylo=2016&q=android+gnss+measurements+(RTK+OR+PPP+OR+PRECISE)&hl=en&as_sdt=0,5
3. M. Khider, F. van Diggelen. "GNSS Measurement Tools". *GitHub*. Url: https://github.com/google/gps-measurement-tools

4. T. E. Humphreys, M. Murrian, F. van Diggelen, S. Podshivalov, and K. M. Pesyna. "On the feasibility of cm-accurate positioning via a smartphone's antenna and GNSS chip." In *Position, Location and Navigation Symposium (PLANS)*, pp. 232-242. IEEE/ION, 2016.
5. E. Realini, S. Caldera, L. Pertusini, and D. Sampietro. "Precise GNSS positioning using smart devices." *Sensors 17*, no. 10 (2017): 2434.
6. G. Pirazzi, A. Mazzoni, L. Biagi, and M. Crespi. "Preliminary performance analysis with a GPS+ Galileo enabled chipset embedded in a smartphone." In *Proceedings of the 30nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2017)*, pp. 101-115.
7. F. van Diggelen, and R. Want, and W. Wang. "How to get one-meter location-accuracy from Android devices (Google I/O '18)?" *Google Developers I/O, 2018*. YouTube url: https://www.youtube.com/watch?v=vywGgSrGODU&feature=youtu.be
8. F. van Diggelen, and M. Khider. "GNSS analysis tools from Google." *Inside GNSS 13*, no. 2 (2018): 48-57.
9. B. Chen, C. Gao, Y. Liu, and P. Sun. (2019). "Real-time precise point positioning with a Xiaomi MI 8 android smartphone." *Sensors*, 19(12), 2835.
10. G. Li, and J. Geng. "Characteristics of raw multi-GNSS measurement error from Google Android smart devices." *GPS Solutions 23*, no. 3 (2019): 90.
11. Q. Wen, J. Geng, G. Li, and J. Guo. "Precise point positioning with ambiguity resolution using an external survey-grade antenna enhanced dual-frequency android GNSS data." *Measurement 157* (2020): 107634.
12. F. Darugna, J. Wübbena, A. Ito, T. Wübbena, G. Wübbena, M. Schmitz, "RTK and PPP-RTK Using Smartphones: From Short-Baseline to Long-Baseline Applications," *Proceedings of the 32nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2019)*, Miami, Florida, September 2019, pp. 3932-3945.
13. J. Liu, Z. Xu, Y. Wang, L. Zhao, and R. Zhang. "Toward Achieving Robust Sub-meter Kinematic Positioning on Android with Multi-constellation GNSS." In *Proceedings of the 32nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2019)*, pp. 89-108.
14. T. Takasu, and A. Yasuda. "Development of the low-cost RTK-GPS receiver with an open source program package RTKLIB." In *International symposium on GPS/GNSS*, pp. 4-6. International Convention Center Jeju Korea, 2009.
15. T. Everett. "Exploring high precision GPS/GNSS with low-cost hardware and software solutions". *Rtkexplorer*. Url: http://rtkexplorer.com/
16. F. Zhou, D. Dong, W. Li, X. Jiang, J. Wickert, and H. Schuh. "GAMP: An open-source software of multi-GNSS precise point positioning using undifferenced and uncombined observations." *GPS Solutions 22*, no. 2 (2018): 33.
17. A. M. Herrera, H. F. Suhandri, E. Realini, M. Reguzzoni, and M. Clara de Lacy. "goGPS: open-source MATLAB software." *GPS solutions 20*, no. 3 (2016): 595-603.
18. G. Xiao, G. Liu, J. Ou, G. Liu, S. Wang, and A. Guo. "MG-APP: an open-source software for multi-GNSS precise point positioning and application analysis." *GPS Solutions 24* (2020): 1-13.
19. J. Geng, X. Chen, Y. Pan, S. Mao, C. Li, J. Zhou, and K. Zhang. "PRIDE PPP-AR: an open-source software for GPS PPP ambiguity resolution." *GPS Solutions 23*, no. 4 (2019): 91.
20. B. Bahadur, and M. Nohutcu. "PPPH: a MATLAB-based software for multi-GNSS precise point positioning analysis." *GPS solutions 22*, no. 4 (2018): 113.
21. R. B. Harris, and R. G. Mach. "The GPSTk: an open source GPS toolkit." *GPS Solutions 11*, no. 2 (2007): 145-150.
22. D. Ibáñez, A. Rovira-García, J. Sanz, J. M. Juan, G. Gonzalez-Casado, D. Jimenez-Baños, C. López-Echazarreta, I. Lapin. "The GNSS Laboratory Tool Suite (gLAB) updates: SBAS, DGNSS and Global Monitoring System." *The 9th ESA Workshop on Satellite Navigation Technologies (NAVITEC 2018)*, Noordwijk, The Netherlands. December 5 - 7, 2018. DOI: 10.1109/NAVITEC.2018.8642707.
23. Y. Zhang. "Net_Diff: A Software for GNSS Download, Positioning and Analysis". *GNSS Analysis Center at Shanghai Astronomical Observatory (SHAO)*. Url: http://center.shao.ac.cn/shao_gnss_ac/Net_diff/Net_diff.html.
24. Rokubun Inc. "Rokubun JASON: Cloud-based Precise Positioning-as-a-Service." Url: https://jason.rokubun.cat/#!/.
25. Y. Bar-Sever. "APPS: The automatic precise point positioning service from JPL's global differential GPS (GDGPS) system." *ION GNSS 2008 meeting, Savannah, USA, September (http://apps. gdgps. net)*. 2008.

26. R. F. Leandro, M. C. Santos, and R. B. Langley. “GAPS: The GPS Analysis and Positioning Software – A Brief Overview.” *Proceedings of the 20th International Technical Meeting of the Science for a Sustainable Planet*, International Association of Geodesy Symposia, 139. 2017.
27. P. Tétreault, J. Kouba, P. Héroux, and P. Legree. "CSRS-PPP: an internet service for GPS user access to the Canadian Spatial Reference Frame." *Geomatica 59*, no. 1 (2005): 17-28.
28. US National Geodetic Survey. "OPUS: Online Positioning User Service". Url: https://geodesy.noaa.gov/OPUS/.
29. Trimble Inc. "Trimble CenterPoint RTX Post-Processing" Url: https://trimblertx.com/UploadForm.aspx.
30. IGS, RTCM-SC104. "RINEX-The Receiver Independent Exchange Format (Version 3.03)." URL: ftp://igs. org/pub/data/format/rinex303. pdf (2015).
31. The GSA GNSS Raw Measurements Task Force. [White Paper] "Using GNSS Raw Measurements on Android devices", 2017. URL: https://www.gsa.europa.eu/newsroom/news/available-now-white-paper-using-gnss-raw-measurements-android-devices
32. F. van Diggelen, M. Khider. "GNSS Analysis Tools from Google." *InsideGNSS Magazine*, Mar 2018.
33. National Marine Electronics Association. "NMEA 0183--Standard for interfacing marine electronic devices." NMEA, 2002.