IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received September 23, 2021, accepted October 19, 2021, date of publication November 4, 2021,
date of current version November 15, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3125496

OpenPosLib: A Library to Achieve Centimetric Geo-Spatial Positioning on a Budget

LIONEL METONGNON[1,2], SÉBASTIEN STREBELLE[1], FABIEN DUCHENE[1], AXEL LEGAY[1], AND RAMIN SADRE[1]

[1]ICTEAM, Université catholique de Louvain, 1348 Louvain-La-Neuve, Belgium
[2]IFRI, Université d'Abomey-Calavi, Abomey-Calavi, Benin

Corresponding author: Lionel Metongnon (lionel.metongnon@uclouvain.be)

This work was supported in part by Proximus.

ABSTRACT Positioning systems can be found everywhere, from the navigation system of a car to the smart watch that tracks the running performances of its wearer. With the generalization of positioning systems, new use cases have begun to emerge that require or could benefit from increased accuracy. While the technology has been detailed in the literature for several years, the deployment of positioning techniques at the centimeter level has proved challenging. In this paper we propose OpenPosLib, an open source library that aims to fill the gap between all the components needed to achieve a centimetric accuracy and the user-facing application. Our objective is to remove most of the complexity needed to obtain centimetric accuracy from the developer so as to enable end-users to reap the benefits of more applications that leverage centimetric accuracy. Our results show that when coupled with inexpensive hardware, OpenPosLib enables users to get centimetric precision on a budget.

INDEX TERMS Satellite navigation systems, global positioning system, position measurement.

I. INTRODUCTION AND MOTIVATION

Since its inception with the *Navy Navigation Satellite System* (Transit) [1], satellite navigation has become an integral part of our life. With the advent of the Global Positioning System (GPS) [2], what used to be a military technology aimed to guide ballistic missiles became available to the general public. Nowadays, positioning systems can be found everywhere, from the navigation system of a car to the smart watch that tracks the running performances of its wearer. Because of the increasing reliance on positioning systems, several countries started to develop their own, such as the Global Navigation Satellite System (GLONASS), Galileo and BeiDou Navigation Satellite System (BDS). These multiple positioning systems now come under the generic term of Global Navigation Satellite System (GNSS). With the generalization of positioning systems, new use cases began to emerge: Augmented Reality games such as *Pokemon Go*, Geocaching [3], sport tracking [4], precision agriculture, etc. This also resulted in new requirements in terms of accuracy. For instance, augmented reality games may benefit from increased accuracy, whereas land surveying requires it.

The associate editor coordinating the review of this manuscript and approving it for publication was Di He.

With the typical GPS-enabled smartphone typically being accurate to within 4.9 m under ideal conditions [5], new techniques such as Real-time kinematic positioning (RTK) were developed to increase the accuracy of a measurement up to the centimetric level. While the technology has been detailed in the literature for several years, the deployment of positioning techniques at the centimeter level has proved challenging, even in professional environments such as precision agriculture [6]. Professional land surveying equipment leveraging these technologies has also reached the market, but its cost and lack of transparency puts it out of reach of the average user or application developer.

To enable end-users and application developers to reap the benefits of centimetric positioning, we propose OpenPosLib, an open source library that aims to fill the gap between all the components needed to achieve centimetric accuracy and the user-facing application. By bundling and integrating all the necessary technologies to achieve centimeter level accuracy, OpenPosLib is able to present the developer with a simple and intuitive interface to obtain centimetric measurements easily.

In this paper, we aim to answer two questions:

Q1 Is it possible to attain centimetric precision by using only a smartphone and OpenPosLib?

Q2 Is it possible to attain centimetric precision by using OpenPosLib and an external GNSS receiver?

To answer these questions, we first detail the relevant background knowledge in Section II. In Section III we present the general architecture needed to achieve centimetric accuracy and the challenges entailed. Section IV details OpenPosLib's architecture and functionalities. To validate the effectiveness of OpenPosLib, we compare it with existing professional products in Section V. Our results shows that the answer to the first research question is *no*, while the answer to the second question is *yes*. By taking measurements in several locations and comparing the results we demonstrate that, when coupled with inexpensive hardware, OpenPosLib enables users to get centimetric precision on a budget.

II. BACKGROUND

The need for positioning technologies has always been a requirement of human activities. When on the move, we need to identify our positions and correct or adjust the directions in which we are heading. Many techniques have been used to determine these geographic positions, from celestial navigation, map and compass, sextant and chronometer up to radar navigation. These technologies had a lot of inaccuracies, resulting in loss of fuel and time and sometimes accidents. In the last few years, new technologies enabling more accurate positioning have emerged. In this section we introduce the relevant positioning technologies used in this project.

A. GLOBAL NAVIGATION SATELLITE SYSTEM (GNSS)

Global Navigation Satellite System (GNSS) is the standard generic term referring to the use of geodetic technologies through satellites to provide accurate positioning information [7]. A GNSS consists of a collection of satellites, called a constellation, providing radio signals from space that transmit timing and positioning data to GNSS receivers. The Global Positioning System (GPS) is the first constellation that emerged in 1978; it belongs to the US and was followed by GLONASS (Russia) in 1982, BDS (China) in 2000, and Galileo (Europe) in 2016. Locally, the Quasi-Zenith Satellite System (QZSS) and Indian Regional Navigation Satellite System (IRNSS) were launched by Japan and India, respectively, in 2018 to cover their regions.

Every constellation has an optimized number of satellites at a certain altitude, inclination and rotation period. GPS is composed of 31 satellites at an orbit of 20,200 km, 55° inclination and a period 11 hours and 58 minutes. GLONASS has 24 satellites at an orbit of 19,130 km, 64.8° inclination and a period of 11 hours and 16 minutes. Galileo is composed of 28 satellites at an orbit of 23,222 km with 56.0° inclination and a period of 14 hours 4 minutes 42 seconds. BDS is composed of 35 satellites at an orbit of 21,150 km inclined at 55.5° and with a period of 12 hours 53 minutes. Each satellite broadcasts a 1500-bit-long navigation message containing information on the satellite's clock, orbital parameters, health status, etc. The message is modulated with a Pseudo Random Noise (PRN) code that is different for each satellite. When such a message is received by a GNSS receiver the latter can determine its distance (range) to the satellite by calculating the difference between the time of reception and the clock information found in the message. To determine its position on Earth, the receiver would need messages from three satellites and a precise clock synchronized to the satellites' clocks. Fortunately, the need for such a clock can be avoided by using the signals of four satellites instead of three (Fig. 1).

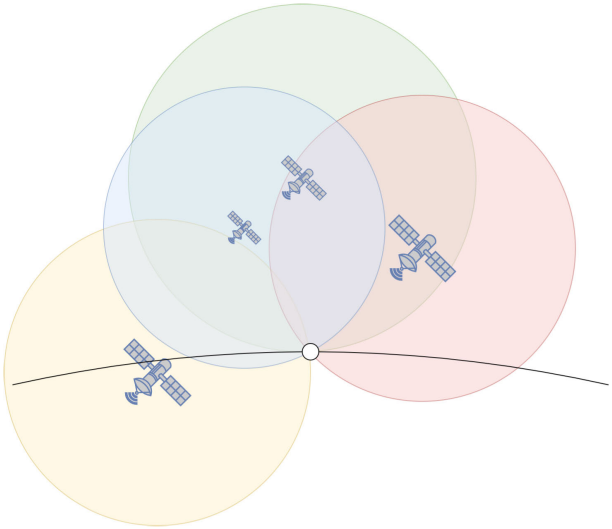


FIGURE 1. Determine position using satellites.

Since the satellite signal travels from space to Earth, many factors induce position estimation errors. The code-phase method yields an accuracy of 3 to 5 meters, which is limited by the following error sources [8], [9]:

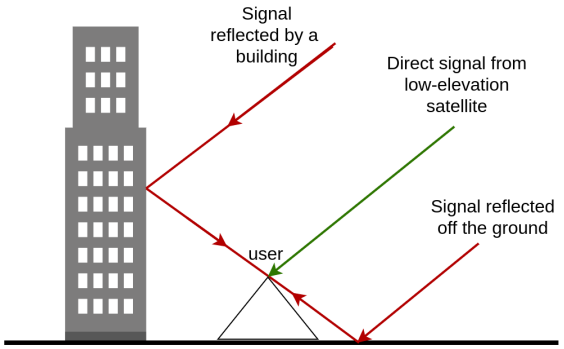
- timing errors related to clock errors on the satellite (±2m) or the GNSS receiver and inter-system biases;
- signal propagation errors due to atmospheric delay, especially inside the ionosphere(>5m) and the troposphere (±0.5m), and multipath errors (±1m) due to refraction on the ground;
- system errors composed of satellite orbital errors (±2.5m) and receiver noise (±0.3m).

To reduce the error, many GNSS receivers use dual frequency measurements. Since atmospheric conditions have different impacts depending on signal frequency, nowadays, satellites use up to three different carrier frequencies to send their information. By comparing the delays of different carriers coming from the same satellite, the receiver can deduce the medium errors and correct the position.

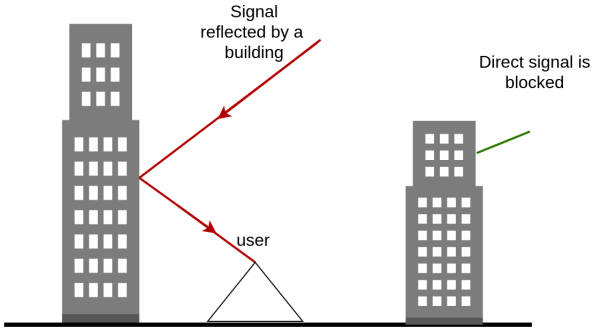
B. DIFFICULT ENVIRONMENT FOR GNSS TECHNOLOGIES

The requirement for geographic position computation is to have at least four satellite signals. However, those signals needs to be strong enough for the GNSS receiver to compute an accurate pseudorange and then its position. A strong signal is correlated to the Line of Sight (LOS) between the GNSS receiver and the satellites [10]. In an urban environment,

due to the presence of tall buildings, the sky observed by the GNSS receivers is narrowed, which limits the number of satellites in the LOS. Having access to more constellations can increase the number of satellites in the LOS. However, due to the buildings, GNSS receivers also observe multipath interferences, which increases the error margin of the computed pseudoranges and carrier phases [11], [12]. Two different scenarios are shown in Fig. 2. In the first scenario (a), the LOS signal and its reflection from multipath interference are received. In the second scenario (b), the Non-line of Sight (NLOS) signal alone is received by the GNSS.



(a) Multipath interference



(b) NLOS Reception

FIGURE 2. Multipath and NLOS interference [12].

Several solutions exist to address the above issues, such as specific antenna design, receiver design, weighting model, signal processing, image processing, consistency checking, mapping-aided and statistical approaches [13]. Such approaches can have a huge impact on the hardware's performance and cost. Another way to reduce GNSS errors is to use Real-time kinematic (RTK), which is detailed in Section II-D.

C. COORDINATE REFERENCE SYSTEM (CRS)

In order to express a position calculated from satellite communications, some unit of measurements is required. Geographical coordinates are typically given in *latitude*, *longitude* and *altitude*. These coordinates describe a position relative to an *ellipsoid* model of the earth [14], where the planet is considered to be a smooth sphere flattened at the poles. The latitude and longitude are the angles formed between the position and the two semi-major axes of the ellipsoid, the equator

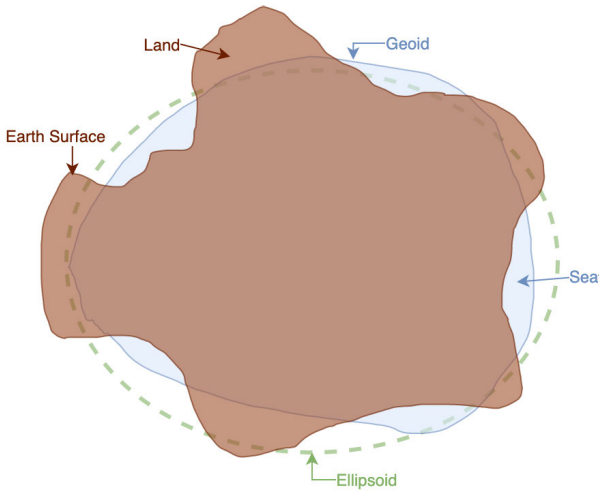


FIGURE 3. Differences between ellipsoid and geoid models of Earth.

and the prime meridian, respectively. The ellipsoidal altitude is the height of the position above the ellipsoid surface. Most usages, however, define the altitude as the orthometric height, or the elevation above the geoid model of the earth [15]. This model takes the influence of gravity and Earth's rotation into account to give a better representation of the shape of the earth, and is equivalent to the mean sea level (the shape of the ocean surface without meteorological influences). The orthometric height is therefore known as the altitude above mean sea level. The difference between an ellipsoid and geoid model of the earth can be seen in Fig. 3.

Several ellipsoid models might be used to define *Geodetic Datums*, reference frames to measure positions on Earth. The datum most used by GNSS receivers is the World Geodetic System (1984) (WGS 84) [16] standard used as a coordinate system by the GPS constellation and most mapping applications around the globe. Because Earth is not a perfect ellipsoid, some datums are made specifically for a certain region to get a more accurate representation of local positions [17].

D. REAL-TIME KINEMATIC (RTK)

RTK [18] was developed in the mid-1990s to provide millimeter-level accuracy and real-time positioning by eliminating errors caused by receiver errors (multipath, accuracy, etc.). Under a clear-sky location, the timing errors, propagation errors and orbital errors are canceled when differential processing is used. The satellite clock error is constant for every device, while the multipath and noise errors are related to the GNSS receivers. Instead of using the PRN pseudorange measurements used for GNSS receivers, RTK technology relies on carrier-phase measurements, bringing the accuracy from 1 meter to 5 millimeters [19]. For that, it uses a special receiver called a *Base Station*. A base station is a fixed receiver with a well-known position. The base station generates a replica of the satellite signal and tries to synchronize it with the incoming signal [20]. Carrier-phase ambiguity resolution is the main source of errors and fixing these ambiguities brings the position accuracy to the level

of a few millimeters. RTK technology is used to compute the position of the fixed base station and then transmits correction information by radio frequency to other GNSS receivers [21]–[24]. In this case, the moving GNSS receivers are called rovers.

Since the information is sent by radio waves, a drawback of this approach is that the distance between the rover and the base station needs to be between 10 and 20 kilometers [25]. The distance limitation is due to natural biases since the rover and base station will not get the same information from the satellites (different atmospheric conditions). This limitation is resolved by using multiple interconnected base stations. Typically, these base station networks are composed by 5 to 10 base stations per 10, 000 kilometers^2. These base stations can be independently owned but they contribute to achieving better accuracy together.

The data processing for a Network RTK (NRTK) involves three steps:

1) The phase ambiguity resolution can be fixed by using differential measurements between different base stations. In [19], the authors presented the Carrier Phase Corrections (CPC) of the NRTK explained by Equation (1), as shown at the bottom of the page.
2) Estimation of the ionospheric refraction to compute the Medium-Scale Travelling Ionospheric Disturbances (MSTID) since, depending on the distance between the base station and rover, biases appear at the same time as the base station and rover's correlation disappears.
3) A set of observations is computed and sent to the rover.

Several techniques are available for the last steps [26]:

- A Virtual Reference Station (VRS) can be created near the rover after all the observations of the base stations around it haven been computed. A new set of observations is generated for this VRS.
- In the Master Auxiliary Corrections (MAX) approach, up to six base stations near the rover send their observations, with the closest one acting as the master and the others as auxiliaries. The rover computes the margin of error and corrects its position.
- etc.

Base station observations are sent to the rover using the Radio Technical Commission for Maritime (RTCM) protocol. RTCM is a binary protocol used as the international standard for data transmission. Since the distances between base stations and the rover are increasing, the rover needs more powerful equipment to exchange information. A better method of sending base station observations is then Networked Transport of RTCM via Internet Protocol (NTRIP). In this paper, an NRTK of Continuously Operating Reference Stations (CORS) using NTRIP communication is called a CORS network.

E. NETWORKED TRANSPORT OF RTCM VIA INTERNET PROTOCOL (NTRIP)

NTRIP is a Transmission Control Protocol (TCP) network protocol for RTCM exchanges through the Internet. Apart from solving the distance problem with NRTK infrastructures, NTRIP allows the apparition of new GNSS receivers like Smartphones or computers using the Internet to contact the base stations. With the NRTK the cost of radio frequency licenses also disappears. NTRIP is composed of three major parts [27]:

1) several NTRIP servers that define an ID, called a ''mountpoint,'' for every NTRIP source (base station). Those mountpoints allow the NTRIP client to select its base station and often the technology used for the correction (VRS, MAX, etc.).
2) a NTRIP caster, which is essentially an http server allowing the NTRIP clients and servers to exchange information. By default, its response consists of a source-table with the records of all the available NTRIP servers, the RTCM format used, etc.
3) NTRIP client software that is used by the GNSS receiver to exchange information with the NTRIP caster.

The GNSS receiver starts by getting its approximate position, then uses its NTRIP client to initiate a connection to a NTRIP caster, sends a RTCM containing its position and gets observations from the caster. The GNSS receiver uses the observations to correct its position, and compute its accuracy.

$$CPC^j(t) = \rho_A^j(t) - \lambda\phi_A^j(t) - \lambda N_A^j(t) - c\delta^j(t) - c\delta_A(t) = -I_A^j(t) - T_A^j(t) - E_A^j(t) \quad (1)$$

where:

$\rho_A^j(t) = \sqrt{(X_A - X^j)^2 + (Y_A - Y^j)^2 + (Z_A - Z^j)^2}$ =geometrical range calculated with the known coordinates of the base and rover;
$\lambda\phi_A^j(t)$ =carrier phase multiplied by wavelength;
λN =phase ambiguity multiplied by wavelength, fixed by the control center;
$c\delta^j(t)$ =satellite clock error;
$c\delta_A(t)$ =receiver clock error, estimated for all CORSs by the control center;
$I_A^j(t)$ =ionospheric error;
$T_A^j(t)$ =tropospheric error;
$E_A^j(t)$ =ephemerids error;

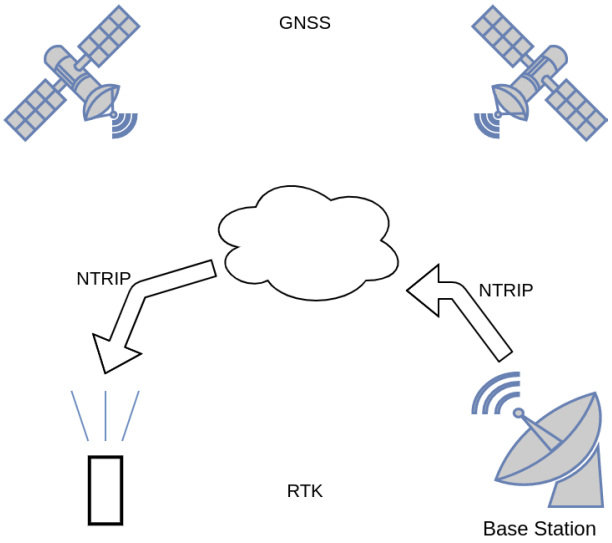


FIGURE 4. Precise positioning using NTRIP.

III. SCIENTIFIC QUESTIONS

In this section we present how the different technologies introduced in Section II could interact with each other to answer the questions presented in Section I and what is needed on the application side to achieve centimetric accuracy.

A. CENTIMETRIC POSITION ON AN END-USER APPLICATION

To access positioning services, the end-user application needs to access GNSS information through a receiver. Nowadays, smartphones come with built-in chips allowing their applications to benefit from geolocation. These chips are used by applications such as Google Maps to provide end-users with directions. However, precision of only 4.9 meters can be achieved. In order for the application on the smartphone to have access to precise positioning, it needs to access RTK data. Because there is not always a base station nearby (10 to 20 kilometers), NTRIP is a good way for smartphones to access those corrective data through the Internet as shown in Fig. 4. The CORS network shares the data related to its position (and its view of the satellites) with the rover using the NTRIP protocol. This allows the rover to correct the ambiguity of its position and to achieve centimetric accuracy using RTK computation.

To get centimetric measurements on a smartphone, two solutions are available: using the built-in GNSS chip or an external receiver.

1) USING THE SMARTPHONE'S RECEIVER

In using the smartphone's built-in chip, the first difficulty is the lack of a dedicated RTK chip. RTKLIB,[1] a powerful library for RTK corrections, is used extensively, especially with desktop applications. Fig. 5 illustrates the data flow to

[1]http://www.rtklib.com/

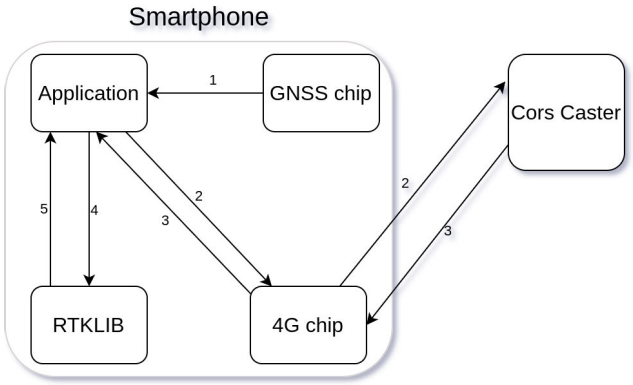


FIGURE 5. Data flow of centimetric positioning with a smartphone.

get precise positioning information on a smartphone using RTKLIB. In Step 1, the smartphone's GNSS chip gets the satellites' views and sends them to the application. When a position is determined, in Step 2 the application sends its position to a CORS caster through the 4G chip. The CORS then sends the appropriate satellites views back to the application in Step 3. In Step 4, the application sends both the CORS caster's and its GNSS views to the RTK software (RTKLIB). After performing the corrections, RTKLIB sends the precise position back to the application in Step 5.

We used a Google-provided[2] equation to compute important information such as the PseudoRange (PRN) that are required for information exchange between RTKLIB and Android's GNSS Application Programming Interface (API).

$$Pseudorange = (Signal_{AT} - Signal_{TT}) * C \tag{2}$$

This is shown in Equation (2), where `Signal_AT` is the arrival time of the signal in nanoseconds, `Signal_TT` the transmission time of the signal in nanoseconds and C the speed of light (299792458 m/s).

$$\begin{aligned} Signal_{AT} = (time - fullBias - (\lfloor -fullBias/WeekSec * 1^{-9} \rfloor \\ * WeekSec * 1^{9}) - timeOffset - bias) * 1^{-9} \end{aligned} \tag{3}$$

The equation to compute the arrival time of the signal is shown in Equation (3), where `time` is the hardware clock of the receiver in nanoseconds, `fullBias` is the difference between the hardware clock and the true GPS Time since 0000Z, January 6^{th}, 1980, in nanoseconds, `timeOffset` is the time offset at which the measurement was taken in nanoseconds, `bias` is the clock's sub-nanosecond bias and `WeekSec` is the duration of a week in seconds (604800).

$$Signal_{TT} = receivedSvTime * 1^{-9} \tag{4}$$

The equation to compute the transmission time of the signal is shown in Equation (4), where `receivedSvTime` is the received satellite time of the signal in nanoseconds.

While OpenPosLib supports the configuration described in Fig. 5, our measurements show that when only the built-in

[2]https://drive.google.com/file/d/1jwrVWkdWhlO14b0uJ1 × 64Z97w5w5tzaf/view

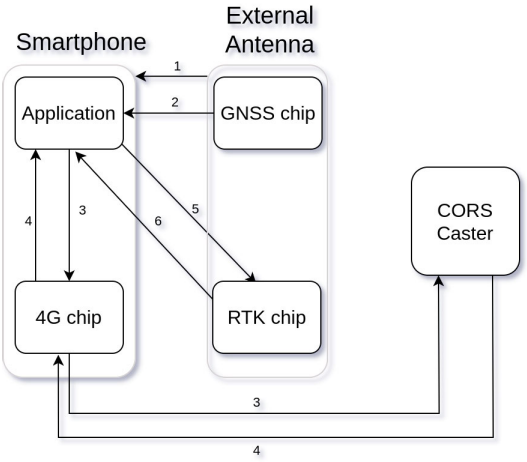


FIGURE 6. Data flow of centimetric positioning with a smartphone coupled with an external antenna.

GNSS receivers were used, our smartphones (Huawei P10, OnePlus 7) were unable to reach a centimetric precision. As explained in [28], the small size of the GNSS antenna inside the smartphone and the lack of multipath rejection capabilities cause signal reflection that results in incoherent PRN and carrier-phase values. As shown in [28], it is possible to achieve centimetric precision using the internal GNSS receiver; however this requires the use of a choke ring and data post-processing. To answer the first scientific question: *Q1. is it possible to achieve centimetric precision by using only a smartphone and* OpenPosLib*?*, we can answer that even with the addition of RTKLIB, Smartphones are not yet able to compute a reliable real-time centimetric position on their own.

2) USING AN EXTERNAL RECEIVER

The other solution, in which an external receiver is used, is shown in Fig. 6. In this figure, the first step consists of connecting the smartphone to the external antenna through a USB serial connection. The position data are sent to the application in Step 2. The application sends its position through the 4G chip to a CORS caster in Step 3 and gets its views in Step 4. In Step 5, the satellite views coming from the CORS are sent to the RTK chip of the external receiver, where the precise position is computed and sent back to the smartphone application in Step 6.

In this configuration, our smartphones are able to get centimetric measurements using OpenPosLib. Section V contains the answer to the second scientific question: *Q2. is it possible to achieve centimetric precision by using* OpenPosLib *and an external GNSS receiver?* through an evaluation and comparison with an existing professional product.

B. EXISTING PRODUCTS

Several commercial products have already been designed to meet the need for centimetric positioning. For instance, Leica makes smart antennas to provide its customer with precise position [29] and business applications. While this kind of solution works, one of the drawbacks is that they usually are expensive and cannot be customized to fit the client's needs. The software comes as a black-box and does not let you understand the operations or check the correctness of the results. In the same way, Trimble provides integrated GNSS systems to register equipment position. They provided a limited set of software for cadastral field surveying, site vision and topographic surveying that does not offer the versatility needed by some businesses.

IV. OpenPosLib's APPROACH

A. MOTIVATION

Even with the correct hardware, obtaining centimetric accuracy can be a challenge for an application developer. First, the developer needs to get access to one of the most important components of the positioning technology: the GNSS receivers. Nowadays, Android provides an API to access a smartphone GNSS receiver and then a possibility for RTK corrections. However, as mentioned in Section III-A1, our findings show that using the smartphone's internal receiver does not provide centimetric precision. To reach this target, the developer thus needs to acquire an external receiver with the required performance for precise positioning. Only a handful of chipsets (ublox, trimble, tersus, etc.) provide the required performance.

Once the external receiver is acquired, the developer needs to connect it to the smartphone via either Bluetooth (when available) or a serial connection. To use a serial connection over the USB port of the smartphone, the developer needs to find and add an external driver library to convert USB to Serial. One example is UsbSerial.[3] This library supports six different chipsets through the UsbSerialDevice class to configure and exchange information between Android applications and the receiver.

In order to achieve centimetric precision, the developer also needs RTK corrections. After getting access to a local CORS network, the developer needs to create a client for one of the network's NTRIP servers and receive RTK corrections using the NTRIP protocol.

Some chipsets can compute RTK corrections and output precise locations when given CORS views. For other chipsets, the developer needs to combine the receiver's imprecise output with the CORS views and compute the corrections her/himself. The open-source library RTKLIB[4] can help to perform the RTK computations needed to get the precise location.

Depending on the level of precision needed, the data need to be converted based on the continent derivation as explained in Section II-C. Some open-source libraries to make these conversion, such as Proj4J,[5] Geopackage-Java[6] or Geotools[7]

[3]https://github.com/felHR85/UsbSerial.git
[4]http://www.rtklib.com
[5]https://github.com/Proj4J/proj4j
[6]https://github.com/ngageoint/geopackage-java
[7]https://github.com/geotools/geotools

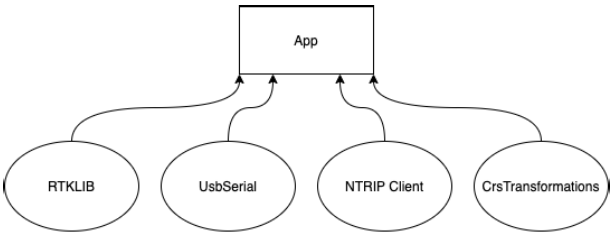


FIGURE 7. Normal application design using NRTK solution.

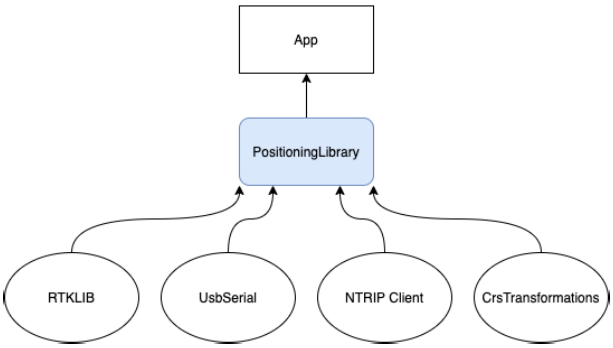


FIGURE 8. Libraries encapsulated by OpenPosLib.

exist. CrsTransformations[8] is another library that uses the inputs from six other libraries, to increase the total number of compatible Coordinate Reference Systems (CRSs) and the overall accuracy of the conversion. It does so by taking the average of the six libraries's values and returning this value to the application.

In the end, the software developer will have an application connected to a lot of different libraries that s/he will have to understand, manage and, more importantly, maintain. These dependencies are illustrated in Fig. 7, where the application is connected to four libraries.

B. OpenPosLib *FUNCTIONALITIES*

The goal of OpenPosLib is to provide an open-source library for Android applications abstracting all RTK and GNSS knowledge and providing a simple API to get precise and accurate positioning information. Software developers can then focus only on the business part of their system and rely on OpenPosLib for the positioning feature. Written in Kotlin, OpenPosLib provides a convenient way to use and combine other libraries presented in Section IV-A and as shown in Fig. 8. Using any of these libraries requires proper knowledge, so our library encapsulates their complexity within a simple API to make the developer's work easier.

Since different types of GNSS receiver exist, our library can be interfaced with both a smartphone and external antenna with or without RTK chips. OpenPosLib also makes it easy for the developer to switch between different execution modes for testing and future work.

When an external receiver is used, OpenPosLib needs to exchange information with different types of antenna

[8]https://github.com/TomasJohansson/crsTransformations

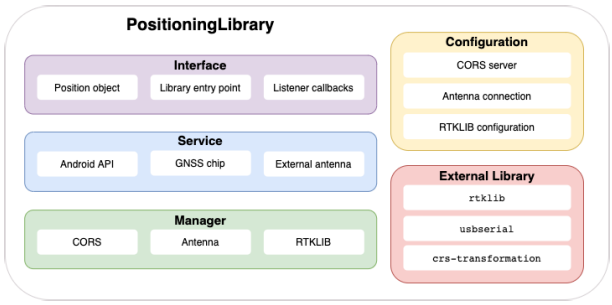


FIGURE 9. OpenPosLib architecture.

from different vendors. For versatility purposes, OpenPosLib exchanges information with external receivers through National Marine Electronics Association (NMEA) messages. Because the NMEA is a well-defined standard, OpenPosLib is not tied to specific antenna proprietary protocol but can be connected to any of them independently. OpenPosLib needs the *gga* messages for time-, position-, and fix-related data, *gst* messages for position error statistics and *dtm* messages for datum reference information. Internally, OpenPosLib is composed of five main components as shown in Fig. 9. The main component is the interface with the library entry point providing the position object through listener callbacks. The positioning service can be performed in many ways with/without RTK and using an internal/external antenna. For more flexibility, we provide an easily configurable API where the developer can set the mode s/he wants to use and provide the appropriate objects required by that mode. Those configuration objects are provided with default values that cover most cases. The developer can then easily navigate between the positioning service mode without much efforts and without significant changes in her/his code base if they update their material. For easy integration, we opted for callback functions so that developers can get precise positions in a straightforward way inside their applications.

Fig. 10 shows an example use case of OpenPosLib. First the user defines callback functions through an implementation of the PositioningListener. The user then configures the library in Internal mode using the device's internal GNSS receiver and starts the service. The library sends position updates through the callback to the user, until the user stops the service. The user then configures the library in External Corrected mode, using a connected external antenna and NTRIP to make RTK corrections, and starts the service again. This time an error occurs, stopping the library, and the user gets the error message through the callback. Through this use case, the library user has to use only a few API functions, while the internal functioning of the positioning is managed by the library. The user has access at the end to the Position object containing the position's timestamp, latitude, longitude and altitude, and finally the horizontal and vertical accuracy. The horizontal accuracy is computed from the standard deviation of the latitude and longitude through the formula $acc_{hz} = \sqrt{stddev_{lat}{}^2 + stddev_{lon}{}^2}$.

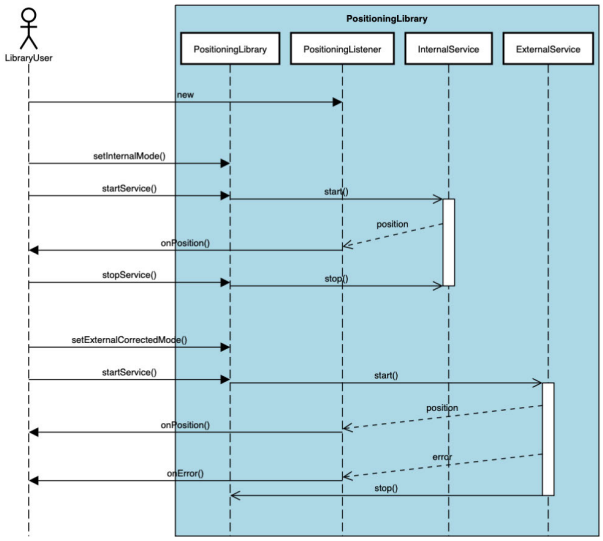


FIGURE 10. OpenPosLib example usage.

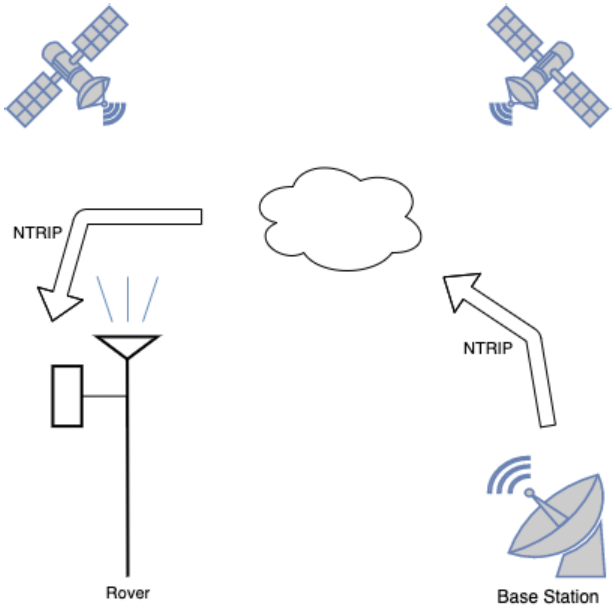


FIGURE 11. Setup of the experiment with NTRIP.

V. EVALUATION

To evaluate OpenPosLib's performances, we tested the correctness of the positions, their accuracy and the convergence time of the positioning data. We measured some coordinates with different antennas in the same environment using the same CORS server. We took measurements in an open space and inside an urban canyon to assert their impact on the antenna and the library's performance. In the following experiments, we focus on the u-blox family coupled with our library. We used two types of antenna:

- a u-blox GNSS Multiband antenna ANN-MB-00 (IP67)
- a calibrated Survey GNSS Multiband antenna (IP67) with GPS L1/L2, GLONASS G1/G2, COMPASS B1/B2/B3 and Galileo E1/E5b/E6.

Both antennas were connected through the u-blox ZED-F9P high precision GNSS modules, which allow up to 1 cm of precision using NTRIP corrections or a base-rover configuration. We then used our solution in those two use-cases. As a control device for the experiments to check the data accuracy of OpenPosLib, we used a commercial Leica antenna:

- Leica GG04 plus smart antenna (IP68) with RTK & multi-frequency GPS, GLONASS, Galileo, BeiDou, QZSS, SBAS.

A. OpenPosLib *USING NTRIP CORRECTIONS*

In this scenario, we used OpenPosLib in *External mode* connected to the RTK calibrated suveyor kit from Ardusimple[9] composed of the u-blox ZED-F9P and the calibrated Survey GNSS Multiband antenna coupled with the Belgian CORS network *Walcors*. The configuration of the antenna is as follows:

- NMEA GGA, GST & DTM activated;
- the baud rate is 38, 400;
- data bits is 8;
- stop bit is 1;
- parity is 0;
- flow control is 0;
- antenna height is 1.80 m.

The CORS network we used had access to four RTK technologies, namely: Flächen-Korrektur Parameter - Area Correction Parameters (FKP), VRS, MAX and NEAR, where each one provided different mountpoints based on the number of constellations used. We had mountpoints with access to GPS only, GPS with GLONASS and, finally, GPS, GLONASS and Galileo together. In our experiment, shown in Fig. 11, we used the MAX technology with access to GPS with GLONASS named MAX17GG and GPS, GLONASS and Galileo named IMAX.

B. OpenPosLib *USING BASE-ROVER CORRECTIONS*

In this scenario, we configured a mobile base station with the antenna ANN-MB-00 using Ardusimple setup.[10] The base station and rover were equipped with a Long Range (LR) radio module based on the Digi XBee SX series and an antenna able to send RTK corrections between 10 and 14 kilometers in both point-to-point and point-to-multipoint mode. This setup was tested in an open space and inside an urban canyon as shown in Fig. 12. The base station was configured as follow:

- The PVT and SVIN messages needed to be activated to monitor the base station status.
- The base station mode was TMODE3 with a minimum of 2.5 m for 3D standard deviation errors and a minimum of 60 seconds before fixing the base station position.

[9]https://www.ardusimple.com/product/rtk-calibrated-surveyor-kit/

[10]https://www.ardusimple.com/configuration-files/

TABLE 1. Position accuracy and convergence time.

Location	Hardware/Software	Position	Vertical Accuracy	Horizontal Accuracy	First Position Time	Convergence time
Open space with no building	Leica Zeno	50.668447 4.621918	2 cm	2 cm	32 seconds	52 seconds
	OpenPosLib	50.668446 4.621918	1.42 cm	1.42 cm	18 seconds	55 seconds
Open space with building in 10 Meters radius	Leica Zeno	50.668526 4.622937	2 cm	2 cm	22 seconds	44 seconds
	OpenPosLib	50.668526 4.622938	1.42 cm	1.42 cm	28 seconds	64 seconds
Urban Canyon building in 3 meters radius	Leica Zeno	50.66912 4.621551	6 cm	4 cm	26 seconds	237 seconds
	OpenPosLib	50.669125 4.621530⋆	5 cm	5 cm	28 seconds	225 seconds
NGI reference point 40A13C1	Leica Zeno	50.675931 4.62275	2 cm	2 cm	31 seconds	59 seconds
50.675931 4.622750	OpenPosLib	50.675931 4.622750	1.42 cm	1.42 cm	17 seconds	26 seconds
NGI reference point 40T106	Leica Zeno	50.662872 4.631626	2 cm	3 cm	24 seconds	46 seconds
50.662872 4.631626	OpenPosLib	50.662872 4.631626	1.42 cm	1.42 cm	33 seconds	45 seconds

- The base station sent the 1077, 1087, 1097, 1230 and 1005 RTCM messages to the rover.

The rover antenna configuration was the same as in Section V-A without the connection to Walcors.

C. RESULTS

The results of our experiments are shown in Table 1. In an open space (represented by the first two lines), the results for OpenPosLib and the commercial product we compared it against (a Leica Zeno Mobile managing a Leica antenna) are quite similar. OpenPosLib gives a better accuracy at the cost of a little more time (3∼20 seconds) than Leica Mobile. The convergence time increases with proximity to buildings. These first measurements prove that in an open space setting, both OpenPosLib and the Leica Zeno report similar results.

These tests were done using both NTRIP and our local base station with a distance of 10 meters between the base station and the rover. Because both setups provided similar results in terms of positions and convergence time, we chose to present only the NTRIP measurements in Table 1.

In the challenging case described in Section II-B and called a "Urban Canyon," where the receiver was located between two 3-story buildings approximately 7 meters apart, a significant disparity between OpenPosLib and the Leica Zeno appeared as shown in the third line of Table 1. Those results were obtained by starting OpenPosLib and the Leica Zeno inside the urban canyon. When we investigated the discrepancy, we realized that despite a reported horizontal accuracy of 4 and 5 cm, respectively, both OpenPosLib and the Leica Zeno were actually reporting an erroneous position. The errors in the positions reported by the Leica Zeno and OpenPosLib were actually greater than 2 and 3 meters (measured using a measuring tape), respectively. Upon further inspection, we attributed this inaccuracy to the quality disparity of the GNSS antenna. However, it is interesting to note that OpenPosLib provides a better reported horizontal accuracy and a shorter convergence time than Leica Zeno

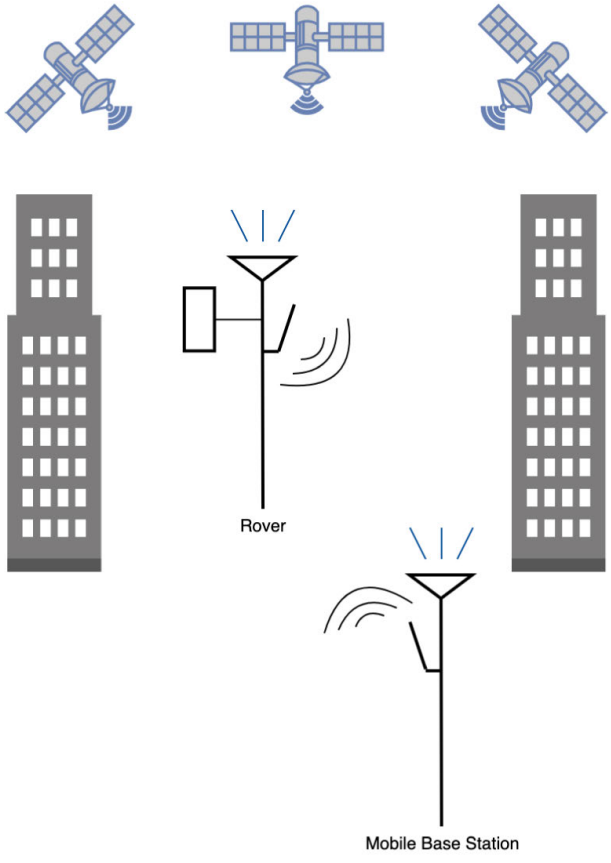


FIGURE 12. Setup of the experiment with base-rover corrections.

Mobile even with erroneous data provided by the antenna. To achieve a reliable positioning inside the urban canyon, our solution was to enter the urban canyon from an open space and thus with an already corrected position using RTK corrections. This approach worked for both the Leica Zeno and OpenPosLib and provided similar results. We did not add those measurements to Table 1 due to the initial positioning being done in an open space and in a different position from the measured position. Because of the difference in methodology, the convergence time would not have been comparable with the other results. This test was also done using both NTRIP and our local base station (located outside of the canyon), yielding similar results.

Finally, because the urban canyon scenario demonstrated that despite a good reported accuracy, erroneous measurements were still possible, we decided to compare the performances of OpenPosLib and the Leica Zeno by using officially known geographical positions provided by Belgium's National Geographic Institute (IGN/NGI). These measurements, done in an open space, are shown in the last two lines of Table 1 and demonstrate that in less challenging situations, both OpenPosLib and the Leica Zeno report accurate results.

VI. CONCLUSION

With this paper we aimed to enable application developers to reap the benefits of multiple positioning technologies and easily get measurements that are accurate to the centimeter. We also aimed to answer two questions:

- Q1 Is it possible to achieve centimetric precision by using only a smartphone and OpenPosLib?
- Q2 Is it possible to achieve centimetric precision by using OpenPosLib and an external GNSS receiver?

To do so, we presented OpenPosLib, an open-source library that enables application developers to get access to centimetric measurements easily by interfacing with our simple API. We used OpenPosLib to answer the two questions with "no" and "yes", respectively. While smartphone GNSS receivers are still too prone to multipath interferences, our study shows that by using OpenPosLib in conjunction with an external GNSS receiver, application developers can easily get access to centimetric accuracy. To verify our findings, we compared OpenPosLib with an existing commercial product and showed that it could get the same accuracy for a fraction of the cost and with an open-source code. The source code of OpenPosLib and a sample application interfacing with OpenPosLib to access measurements are available at https://github.com/SebStreb/OpenPosLib.

REFERENCES

[1] O. L. Sentman, "Navy navigation satellite system (transit)," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 2, no. 7, pp. 25–26, Jul. 1987.
[2] J. G. McNeff, "The global positioning system," *IEEE Trans. Microw. Theory Techn.*, vol. 50, no. 3, pp. 645–652, Mar. 2002.
[3] K. O'Hara, "Understanding geocaching practices and motivations," in *Proc. Conf. Hum. Factors Comput. Syst. (CHI)*. New York, NY, USA: Association for Computing Machinery, 2008, pp. 1177–1186.
[4] R. Cortés, X. Bonnaire, O. Marin, and P. Sens, "Sport trackers and big data: Studying user traces to identify opportunities and challenges," INRIA, Paris, France, Res. Rep. RR-8636, Nov. 2014.
[5] F. V. Diggelen and P. Enge, "The world's first GPS MOOC and worldwide laboratory using smartphones," *Proc. 28th Int. Tech. Meeting Satell. Division Inst. Navigat. (ION GNSS+)*, 2015, pp. 361–369.
[6] M. Pini, G. Marucco, G. Falco, M. Nicola, and W. De Wilde, "Experimental testbed and methodology for the assessment of RTK GNSS receivers used in precision agriculture," *IEEE Access*, vol. 8, pp. 14690–14703, 2020.
[7] P. D. Groves, "Principles of GNSS, inertial, and multisensor integrated navigation systems, 2nd edition [book review]," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 30, no. 2, pp. 26–27, Feb. 2015.
[8] M. Karaim, M. Elsheikh, and A. Noureldin, "GNSS error sources," in *Multifunctional Operation and Application of GPS*, R. B. Rustamov and A. M. Hashimov, Eds. Rijeka, Croatia: InTech, Apr. 2018. [Online]. Available: https://www.intechopen.com/chapters/60049, doi: 10.5772/intechopen.75493.
[9] (2021). *GPS Tutorial*. Accessed: Apr. 5, 2021. [Online]. Available: https://www.trimble.com/gps_tutorial/
[10] P. Xie and M. G. Petovello, "Measuring GNSS multipath distributions in urban canyon environments," *IEEE Trans. Instrum. Meas.*, vol. 64, no. 2, pp. 366–377, Feb. 2015.
[11] P. D. Groves, "Shadow matching: A new GNSS positioning technique for urban canyons," *J. Navigat.*, vol. 64, no. 3, pp. 417–430, 2011.
[12] P. D. Groves, Z. Jiang, L. Wang, and M. K. Ziebart, "Intelligent urban positioning using multi-constellation GNSS with 3D mapping and NLOS signal detection," in *Proc. 25th Int. Tech. Meeting Satell. Division Inst. Navigat. (ION GNSS)*, 2012, pp. 458–472.
[13] N. Zhu, J. Marais, D. Bétaille, and M. Berbineau, "GNSS position integrity in urban environments: A review of literature," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 9, pp. 2762–2778, Sep. 2018.
[14] *The Earth as an Ellipsoid*. Accessed: Jun. 3, 2021. [Online]. Available: http://www.manifold.net/doc/mfd9/the_earth_as_an_ellipsoid.htm
[15] *The Difference Between Ellipsoidal and Orthometric Elevations?* Accessed: Jun. 3, 2021. [Online]. Available: https://support.virtualsurveyor.com/en/support/solutions/articles/1000261349-the-difference-between-ellipsoidal-and-orthometric-elevations
[16] *Department of Defense World Geodetic System 1984, Its Definition and Relationships With Local Geodetic Systems*, DMA WGS 84 Develop. Committee, Defense Mapping Agency, 1991. [Online]. Available: https://www.scirp.org/journal/CTA.aspx?paperID=217
[17] *Latitude and Longitude are Not Enough*. Accessed: Jun. 3, 2021. [Online]. Available: http://www.manifold.net/doc/mfd9/index.htm#latitude_and_longitude_are_not_enough.htm
[18] R. B. Langley, "RTK GPS," *GPS World*, vol. 9, no. 9, pp. 70–76, 1998.
[19] A. Cina, P. Dabove, A. M. Manzino, and M. Piras, "Network real time kinematic (NRTK) positioning—Description, architectures and performances," in *Satellite Positioning—Methods, Models and Applications*, S. Jin, Ed. Rijeka, Croatia: InTech, Mar. 2015. [Online]. Available: https://www.intechopen.com/chapters/47449, doi: 10.5772/59083.
[20] J. M. O. I. Llop, D. Moreno-Salinas, and J. Sánchez, "Full real-time positioning and attitude system based on GNSS-RTK technology," *Sustainability*, vol. 12, no. 23, p. 9796, Nov. 2020.
[21] X. Chen, H. Landau, and U. Vollath, "New tools for network RTK integrity monitoring," in *Proc. 16th Int. Tech. Meeting Satell. Division Inst. Navigat. (ION GPS/GNSS)*, 2003, pp. 1355–1360.
[22] G. Lachapelle and P. Alves, "Multiple reference station approach: Overview and current research," *J. Global Positioning Syst.*, vol. 1, no. 2, pp. 133–136, Dec. 2002.
[23] C. Rizos, "Network RTK research and implementation—A geodetic perspective," *Positioning*, vol. 1, no. 2, pp. 144–150, 2009.
[24] G. Dardanelli, A. Maltese, C. Pipitone, A. Pisciotta, and M. L. Brutto, "NRTK, PPP or static, that is the question. Testing different positioning solutions for GNSS survey," *Remote Sens.*, vol. 13, no. 7, p. 1406, Apr. 2021.
[25] *Introduction to Network RTK*. Accessed: Apr. 7, 2021. [Online]. Available: http://www.wasoft.de/e/iagwg451/intro/introduction.html
[26] M. S. Garrido, E. Giménez, M. C. D. Lacy, and A. J. Gil, "Testing precise positioning using RTK and NRTK corrections provided by MAC and VRS approaches in SE Spain," *J. Spatial Sci.*, vol. 56, no. 2, pp. 169–184, Dec. 2011.
[27] G. Weber, D. Dettmering, H. Gebhard, and R. Kalafus, "Networked transport of RTCM via internet protocol (NTRIP). IP-streaming for real-time GNSS applications," in *Proc. 18th Int. Tech. Meeting Satell. Division Inst. Navigat. (ION GNSS)*, 2005, pp. 2243–2247.

[28] H. Sharma, M. Bochkati, and T. Pany, "Influence of the multipath mitigation on precise positioning with smartphone raw GNSS measurements," in *Proc. ISGNSS*, Oct. 2019. [Online]. Available: http://ipnt.or.kr/2019proc
[29] *Smart Antennas*. Accessed: Apr. 5, 2021. [Online]. Available: https://leica-geosystems.com/products/gnss-systems/smart-antennas

LIONEL METONGNON received the Ph.D. degree in network security from UCLouvain, in 2020, under Prof. Ramin Sadre and Prof. Eugene C. Ezin. Since completing his thesis, he has been working at UCLouvain as a Postdoctoral Researcher on projects related to geolocation and blockchain in health care under Prof. Axel Legay's supervision. His research interests include network attack detection and mitigation, the Internet of Things, and privacy.

SÉBASTIEN STREBELLE was born in Brussels, Belgium, in 1995. He received the master's degree in computer science from UCLouvain, Louvain-la-Neuve, Belgium, in 2019. He has been a Teaching Assistant at the Louvain's School of Engineering, UCLouvain. His research interests include the security of connected vehicles and teaching of computer science.

FABIEN DUCHENE received the Ph.D. degree in computer networking from UCLouvain, in 2019. He completed his Ph.D. under the supervision of Olivier Bonaventure. He is currently a Postdoctoral Researcher in cyber security at UCLouvain working under the supervision of Axel Legay. His research interests include automatic detection of bugs and malware, penetration testing, and networking security.

AXEL LEGAY received the Ph.D. degree from ULiege. He is currently a Professor of cyber security and software engineering at UCLouvain. Prior to that, he was a Team Leader at INRIA. He was a Postdoctoral Researcher at Carnegie Mellon University under the supervision of Ed. Clarke (Turing Award, 2007). He has written more than 350 articles related to security and validation, has participated in more than 50 academic projects on those topics, and has collaborative ties with various industry key players in the area.

RAMIN SADRE is currently a Professor of security and performance of networked systems at UCLouvain. Before that, he was an Assistant Professor at Aalborg University and a Postdoctoral Researcher at the University of Twente. His research interests include the design and security of the Internet of Things, and industrial control systems, in particular performance aspects and detecting intrusions.