Open Source Robotics-Research Article

INTERNATIONAL JOURNAL OF ADVANCED ROBOTIC SYSTEMS

International Journal of Advanced Robotic Systems
May-June 2020: 1–8
© The Author(s) 2020
DOI: 10.1177/1729881420904526
journals.sagepub.com/home/arx
SAGE

Real-time GNSS precise positioning: RTKLIB for ROS

António Ferreira[1], Bruno Matias[1], José Almeida[1,2] and Eduardo Silva[1,2]

Abstract

The global navigation satellite system (GNSS) constitutes an effective and affordable solution to the outdoor positioning problem. When combined with precise positioning techniques, such as the real time kinematic (RTK), centimeter-level positioning accuracy becomes a reality. Such performance is suitable for a whole new range of demanding applications, including high-accuracy field robotics operations. The RTKRCV, part of the RTKLIB package, is one of the most popular open-source solutions for real-time GNSS precise positioning. Yet the lack of integration with the robot operating system (ROS), constitutes a limitation on its adoption by the robotics community. This article addresses this limitation, reporting a new implementation which brings the RTKRCV capabilities into ROS. New features, including ROS publishing and control over a ROS service, were introduced seamlessly, to ensure full compatibility with all original options. Additionally, a new observation synchronization scheme improves solution consistency, particularly relevant for the moving-baseline positioning mode. Real application examples are presented to demonstrate the advantages of our *rtkcrv_ros* package. For community benefit, the software was released as an open-source package.

Keywords

RTKLIB, ROS, RTKRCV, robotics, real-time precise positioning, GNSS, GPS, open source, rtkrcv_ros

Date received: 31 March 2019; accepted: 29 November 2019

Topic Area: Field Robotics
Topic Editor: Yangquan Chen
Associate Editor: Francisco Rico

Introduction

For a mobile robot to be effective, accurate knowledge about its localization is mandatory. Autonomous localization is a perception-driven process, usually addressed through probabilistic data fusion techniques, which combine information from several sensors, to derive an updated estimate of the robot's position and orientation. In this context, outdoor solutions benefit from an important resource, denied to indoor applications, which is the global navigation satellite system (GNSS).

GNSS is a ubiquitous resource, enabling any platform, equipped with a receiver, and under clear sky view, to access periodic global position references. This information is crucial to the localization process, as global updates are essential to bound estimation errors associated with dead reckoning strategies. Nowadays, budget multi-frequency/multi-constellation receivers are accessible, allowing the application of differential processing methods, namely the real time kinematic (RTK) technique, to achieve positioning with centimeter-level accuracy.

[1] Institute for Systems and Computer Engineering, Technology and Science (INESC TEC), Porto, Portugal
[2] Department of Electrical Engineering, ISEP–School of Engineering, Polytechnic Institute of Porto, Porto, Portugal

Corresponding author:
António Ferreira, Institute for Systems and Computer Engineering, Technology and Science (INESC TEC), Rua Dr. António Bernardino de Almeida 431, 4249-015 Porto, Portugal.
Email: ajbf@inesctec.pt

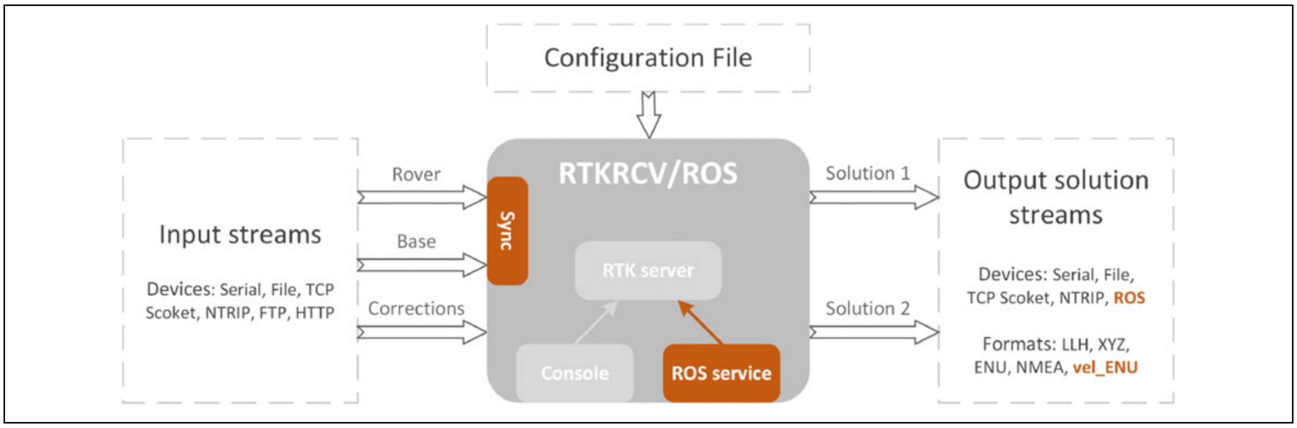


Figure 1. Illustration of the RTKRCV data flow, where raw observations are processed to compute a positioning solution. Results are accessible through two output streams. Highlighted are some of the newly developed features, such as a ROS service to control the RTK server, the new output device to publish output streams over ROS, the velocity solution format (vel_ENU) and the new input observation synchronization method (illustrated by the Sync block).

Despite most receiver's ability to perform RTK internally, transferring differential GNSS processing to an external computational unit is most times advantageous, offering the user more control and monitoring capabilities, while opening the possibility for more complex positioning networks to be developed.

The RTKLIB[1] is one of the most popular and versatile software packages for differential GNSS processing. It consists of an open-source portable library, written in the C programming language, plus a set of standalone programs for real-time and post-processing precise positioning. In the field robotics context, the RTKRCV application assumes particular relevance, offering several real-time precise position modalities, while supporting the most common raw data formats, for wide-range compatibility with most receivers on the market.

Unfortunately, out-of-the-box integration with the robot operating system (ROS)[2] is not available. Nowadays, the ROS middleware is broadly disseminated, serving as development base in the robotics community, hence a ROS translation of the RTKRCV code makes perfect sense.

This article presents the new *rtkrcv_ros* package, a ROS integration of the RTKRCV program, with clear improvements over previous implementations,[3,4] including

- the capability to publish all output stream formats into ROS;
- the possibility to use all previous output devices alongside the ROS publishing functionality;
- the full compatibility with the original configuration file;
- the inclusion of a new velocity output format;
- the possibility of controlling the precise positioning process, by means of a dedicated ROS service; and
- the implementation of a new observation synchronization scheme, to enhance the solution consistency of the moving-baseline positioning mode.

The *rtkrcv_ros* package maintains the configuration flexibility of the original RTKRCV program, essential to the development of complex GNSS positioning systems. Some practical applications, involving single and multiple antenna systems are demonstrated here. Considering the potential interest for the robotics community, the *rtkrcv_ros* package was made freely available.[5]

This article is organized as follows. The next section provides an overview of the RTKRCV and analyses the limitations of previous packages developed for its integration in ROS. The third section presents the new *rtkrcv_ros* package and details the most important features. A demonstration of the *rtkrcv_ros* package capabilities is given in the fourth section, with several examples of single and multiple antenna GNSS systems that can developed based on the *rtkrcv_ros* package. A conclusion is provided in the last section.

Related work

The RTKRCV is a stand-alone command line application, included in the RTKLIB package,[1] for real-time GNSS precise positioning. Regardless of the selected positioning method, the RTKRCV program follows the processing pipeline illustrated in Figure 1, where raw input observations are processed to retrieve a positioning solution.

In the current stable release (RTKLIB 2.4.2), two output streams are available to communicate positioning solutions, each one following one of the four supported output formats: geodetic position (LLH), Earth-Centered Earth-Fixed (XYZ) position, local East-North-Up (ENU) baseline position, and NMEA 0183 sentences (see Figure 1). Output solution streams are configured independently and forwarded through one of the following devices: serial port, TCP socket, NTRIP server or stored in a local file.

All parameters, related with processing options and input/output streams, are stored in a configuration file and loaded at program start.

During execution, the RTKRCV offers a command line console, also reachable over a telnet connection, for process monitoring and control. The monitoring commands display status information on the screen, which are useful for human supervision of the precise positioning process. On the other hand, a small set of commands (start, stop, restart, shutdown, load) allow basic control over the execution of the positioning algorithm—the RTK server thread.

Previous ROS integrations

To our knowledge, two open-source RTKLIB wrappers are available in the form of ROS packages: the *rtklibros* package[4] and the *gnss* package.[3]

The *rtklibros*[4,6] is based on an older RTKRCV version (RTKLIB 2.4.1). It expects raw GNSS observations to be received over a TCP socket, providing positioning solutions in the ENU and LLH output formats exclusively. Configurations can either be loaded from the standard configuration file or from an YAML file. This last option makes use of the ROS parameter server, to store and manage the configuration options, however no substantial benefit is achieved, as on the fly configuration changes still require the RTK server to be restarted. Nonetheless, a ROS service dedicated to the RTK server restart operation, facilitates this procedure.

On the other hand, the *gnss* package[3] implements a bridge between a running RTKRCV instance and the ROS environment. This package serves the unique purpose of translating the ENU output stream, received over a specific TCP socket, into a ROS message.

Both packages offer incomplete integration of the RTKRCV features, particularly regarding the supported output solution formats and input devices. Also, simultaneous execution of several RTKRCV instances is not straightforward, as both packages define specific device configurations for the input data, which conflict if several nodes are executed. Also, conflicting hard-coded ROS identifiers, such as ROS node and ROS topic names, forbid the simultaneous execution of the same node within the global ROS namespace.

Moreover, the use of outdated RTKLIB code, as in the *rtklibros* package, should be avoided, mainly due to hard-coded definitions which require regular revision. Hence, keeping the source code updated is an essential condition to achieve consistent solutions. For that, the latest stable release must be used, while keeping track of all subsequent patches.

The new *rtkrcv_ros* package

We share the major objective behind previous integration packages—transpose the RTKRCV's positioning solutions into the ROS environment. This is accomplished by adopting the ROS publish/subscribe mechanism. However, our *rtkrcv_ros* package not only supports all four original output formats but also offers a new stream (vel_ENU), to communicate velocity information (see Figure 1).

Additionally, the *rtkrcv_ros* package does not compromise on the versatility offered by the original RTKRCV program. In fact, from the user perspective, this package behaves as the RTKRCV, with all options set through the configuration file and no restrictions in terms of output combinations. This allows previous RTKLIB users to easily adapt to this new package. Moreover, effort has been taken to ensure that newly introduced features remain compatible with all previous options.

Some extra features were added, such as a new observation synchronization method to improve the consistency of the moving-baseline solutions. Finally, a new ROS service offers an alternative to the command line console, for easy control over the RTK server.

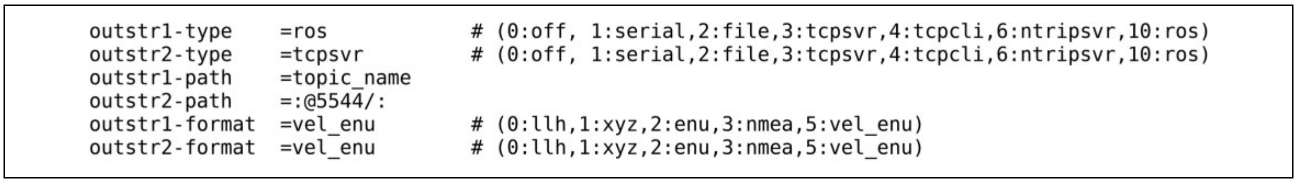
ROS publishing

The original RTKRCV offers two configurable output solution streams, as illustrated in Figure 1. In our implementation, the ROS publishing functionality was seamlessly integrated, behaving as any other device in the original RTKRCV. Therefore, all output formats are eligible to be published into ROS, with all necessary settings made through a configuration file.

Setting a ROS output stream. Our package adopts the same configuration file structure used by the original RTKRCV program, with the exception of some additional arguments related with the new developed features. In the configuration file, a ROS output stream is set by assigning the keyword *ros* to variables *outstr1-type* and/or *outstr2-type*. ROS topic names can also be changed, by modifying variables *outstr1-path* and/or *outstr2-path*, as shown in Figure 2. The ability to specify ROS topic names constitutes a key differentiating aspect of our implementation, since, as opposed to enforcing hard-coded identifiers, as other available implementations do,[3,4] the assignment of custom topic names opens the possibility for publishing multiple topics and running several RTKRCV instances, without resorting to the remap functionality.

ROS messages. In the ROS environment, nodes communicate with each other using ROS topics, which exchange ROS messages. A ROS message defines the structure of the transmitted data package, composed of a set of typed data fields, established at the programming phase. Hence, several ROS messages were necessary to accommodate the data of the different output formats conveniently.

Table 1 presents the association between each output format and the corresponding ROS message. For the position and velocity output streams, since they all share the



```
outstr1-type    =ros           # (0:off, 1:serial,2:file,3:tcpsvr,4:tcpcli,6:ntripsvr,10:ros)
outstr2-type    =tcpsvr        # (0:off, 1:serial,2:file,3:tcpsvr,4:tcpcli,6:ntripsvr,10:ros)
outstr1-path    =topic_name
outstr2-path    =:@5544/:
outstr1-format  =vel_enu       # (0:llh,1:xyz,2:enu,3:nmea,5:vel_enu)
outstr2-format  =vel_enu       # (0:llh,1:xyz,2:enu,3:nmea,5:vel_enu)
```

Figure 2. Partial screenshot of the RTKRCV's configuration file, demonstrating how two output solution streams are configured. Both are set to output the new velocity format. The first output is assigned to a ROS topic publisher, with topic name defined by variable *outstr1-path*. The second is assigned to a TCP socket server, already available in the original RTKRCV implementation.

Table 1. ROS messages used to communicate each output format.

Output format	ROS message
Geodetic position	states_hms.msg
ECEF position	or
ENU position	states_tow.msg
Velocity	
NMEA 0183	nmea.msg
	nmeaGPGGA.msg
	nmeaGPGSA.msg
	nmeaGPGSV.msg
	nmeaGPRMC.msg

same field structure, only two ROS messages were defined. They differ on the timestamp format selected for the output streams – variable *out-timeform* in the configuration file. Accordingly, message *states_hms* is used for timestamps in the form of hours–minutes–seconds. Otherwise, for timestamps expressed in terms of seconds of the week, message *states_tow* is automatically selected.

Outputs following the NMEA 0183 specification are composed of several sentences (GPGGA, GPGSA, GPGSV, and GPRMC), each one with unique field structure. Therefore, custom ROS messages were defined for each case. All individual NMEA sentences are clustered together inside the *nmea* ROS message (see Table 1).

ROS service control commands

Our implementation provides an additional interface, based on the ROS service mechanism, for receiving control commands. This provides an additional interface for other ROS applications to easily interact with the RTK server. At execution time, each *rtkrcv_ros* node advertises a service, labeled after the node's name plus the suffix *_cmd*. The ROS service accepts string commands from Table 2, answering back with the feedback text message generated by the RTKRCV plus a Boolean flag, which becomes true if the operation succeeds.

New velocity output format

In the current stable RTKRCV release (2.4.2), besides NMEA sentences, there are three output formats to communicate position information. The provided output

Table 2. Commands accepted by the ROS service.

Command	Description
start	Starts the RTK server
stop	Stops the RTK server execution
restart	Restarts the RTK server
load *file_path*	Loads a configuration file located at *file_path*

formats do not reflect the evolution of the RTKRCV, which by now supports the estimation of receiver dynamics, by extending the internal Kalman Filter's state vector, to include velocity and acceleration states. Accessing this information becomes desirable, especially for the velocity states, since a direct observation, derived from the Doppler frequency measurement, ensures an accurate estimate.

The most recent 2.4.3 beta release introduces a new output format, based on the solution status stream, logged to file in previous releases. Despite making velocity and acceleration states accessible in real time, the stream lacks uncertainty information, relevant to the development of downstream data fusion processes.

Our implementation fills this gap by establishing a dedicated output format to communicate velocity estimates, together with the corresponding uncertainty, extracted from the covariance matrix of the RTKRCV's Kalman Filter.

Similarly to the original streams, additional status information complements states and uncertainty data, as defined in Table 3. Velocity is given in the local ENU reference frame. The uncertainty is represented by 6 standard deviation (SD) values (σ_{ve}, σ_{vn}, σ_{vu}, σ_{ven}, σ_{vnu}, σ_{veu}), through which the 3×3 covariance submatrix P^v, related to the velocity states, is reconstructed as follows:

$$P^v = \begin{bmatrix} {\sigma_{ve}}^2 & {\sigma_{ven}}^2 & {\sigma_{veu}}^2 \\ {\sigma_{ven}}^2 & {\sigma_{vn}}^2 & {\sigma_{vnu}}^2 \\ {\sigma_{veu}}^2 & {\sigma_{vnu}}^2 & {\sigma_{vu}}^2 \end{bmatrix} \tag{1}$$

The new velocity stream is selected, in the configuration file (Figure 2), by setting variables *outstr1-format* and/or *outstr2-format* equal to *vel_enu*. The user must also enable the estimation of rover dynamics, by setting variable *pos1-dynamics* to *on*, otherwise velocity states will not be estimated by the Kalman Filter and the velocity output will not

Table 3. Message format adopted for the new velocity output stream.

Content	Description
1–Time	Epoch time of the solution.
2–Velocity	Three decimal values corresponding to the velocity in the local ENU reference frame.
3–Quality	Flag indicating the solution quality: 1 Fixed, 2 Float, 4 DGPS, 5 Single
4–Number of satellites	Number of valid satellites used to compute the current solution
5–Standard deviations	Six decimal standard deviations values (σ_{ve}, σ_{vn}, σ_{vu}, σ_{ven}, σ_{vnu}, σ_{veu}), through which the 3×3 covariance matrix, related with velocity states, can be reconstructed
6–Age	Time difference between the receiver and the base station observations
7–Ratio	Current ratio factor used for ambiguity validation

be generated. This stream can be forwarded to all existing output devices, including the new ROS topic publishers.

Observation synchronization

In the RTKRCV, the *misc-svrcycle* configuration option defines the time period at which incoming messages are checked. In that loop, rover messages assume high priority over the base station stream, triggering the positioning computation as soon as rover message arrive. If the base observation is delayed with respect to the rover's, rover messages are associated with older base measurements, as the algorithm does not wait for the arrival of the corresponding base message. While this approach is perfectly acceptable for fixed base stations, it can deeply impact performance in the moving-baseline processing mode.

The moving-baseline RTK mode is a differential GNSS positioning technique, suitable for applications where the base station is allowed to move. Instead of relying on a fixed position, broadcast by the reference station, the algorithm applies the single-point positioning to the base station observations to track its position. This solution is then used in the RTK calculation. Despite the poor global positioning performance, the moving-baseline method provides an accurate relative position measurement between the rover and the base station.

To ensure consistency of the measured baseline, the computed base position should reflect its current state, which implies that base messages must be combined with rover observations from the same epoch. Instead of enforcing this, the original RTKRCV compensates for the base station delay, computing a correction to the base station position, by integrating the measured base velocity over the delay interval. This operation does not guarantee a consistent solution, especially for long observation delays and platforms subjected to fast varying dynamics.

We address this issue by implementing two routines, associated with previously existing options in the configuration file, namely the *pos2-maxage* and *pos2-syncsol* options.

The *pos2-maxage* option establishes a maximum admissible delay between the last base station observation and the most recent rover observation. However, this functionality had not been previously developed for the moving-baseline mode. Our implementation fills this gap, skipping the positioning routine whenever the defined threshold is exceeded. This delay threshold should be adjusted according to accuracy requirements, robot dynamics, and observation data rates. Lower values contribute to increase the solution's accuracy but may force some epochs to be rejected due to lack of recent base station observations.

For high-accuracy applications, with rover and base streams at the same data rate, enforcing exact observation synchronization would be possible and preferable. In the reported implementation, the user may now force the algorithm to establish synchronized observation pairs, by waiting for the corresponding base observation. This feature is associated with option *pos2-syncsol*, which despite being available in the configuration file, had not been implemented for the original RTKRCV. This option is now effective for all positioning modes.

Applications

The new *rtkrcv_ros* package was developed to maintain the same configuration flexibility offered by the original RTKRCV method. This opens the possibility to develop complex GNSS systems, composed of multiple GNSS receivers, running several *rtkrcv_ros* instances, publishing various output solutions, and making use of all available devices.

This section presents three application examples, which make use of the *rtkrcv_ros* package. The first example demonstrates the RTK technique applied to a mobile robot equipped with a single antenna. The second example consists on a dual antenna GNSS system, assembled onboard an autonomous underwater vehicle (AUV), for position, velocity, heading, and pitch determination. Finally, a triple antenna GNSS system for position and full attitude determination is introduced. The configuration files for the three cases are supplied within the *rtkrcv_ros* package.[5]

Single antenna application

From all the positioning algorithms offered by the RTKRCV, the RTK technique is the most advantageous, both in terms of convergence time and in terms of positioning accuracy, capable of reaching centimeter accuracy almost instantaneously, when using multi-frequency receivers. Such performance is achieved by tracking the carrier wave of GNSS signals, a process that requires the combination of observations from a close range base station. To

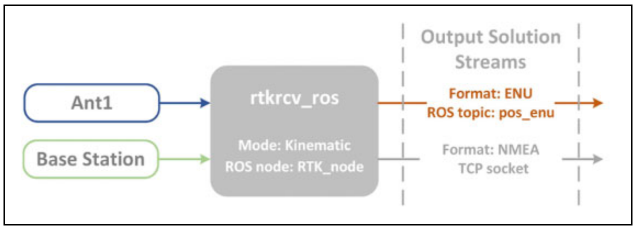


Figure 3. Application of the *rtkrcv_ros* package to compute an RTK positioning solution.

perform RTK positioning of a moving target, the RTKRCV must be configured in the kinematic processing mode and receive the raw data streams from the GNSS receiver onboard (the rover) and the base station, as illustrated in Figure 3.

In the *rtkrcv_ros*, output streams remain independent of each other. This allows the combination of different devices to communicate information simultaneously. For the case illustrated in Figure 3, the first output stream publishes, in the *pos_enu* topic, the robot's position in the ENU reference frame. The second output is fully independent of the first one so, for the sake of this demonstration, the NMEA output format is forwarded through a TCP socket.

Dual antenna setup

Carrying a dual antenna GNSS system onboard provides a practical way to instantly measure the pose in five degrees of freedom, including the heading angle.[7] This arrangement is most valuable in situations where heading is hard to measure, particularly in marine applications, where visual references are scarce and North-seeking techniques are not effective due to continuous motion. Therefore, for the dual antenna example, let's consider the case of an autonomous underwater vehicle (AUV) whose GNSS system is represented in Figure 4.

The possibility of changing ROS node and topic names, allows the user to choose unique identifiers to avoid resource name conflicts. By doing so, as depicted in Figure 4, several *rtkrcv_ros* nodes and various ROS topics can run simultaneously in the same machine and under the global ROS namespace.

For redundancy reasons, and to increase the chance of obtaining an accurate positioning solution, the streams of both antennas are processed in Kinematic mode. This provides a pair of position and velocity measurements from each antenna, which are published in ROS to be fused by the localization algorithm. Additionally, a third *rtkrcv_ros* node, configured in moving-baseline mode, measures the relative position between antennas onboard. The result in the form of a 3-D vector, decomposed along directions East-North-Up ($\vec{v}b = [vb_E, vb_N, vb_U]$), is published into ROS using the ENU output format.

Assuming that the antenna baseline is aligned with the AUV's x axis, the calculation of heading ψ and pitch θ

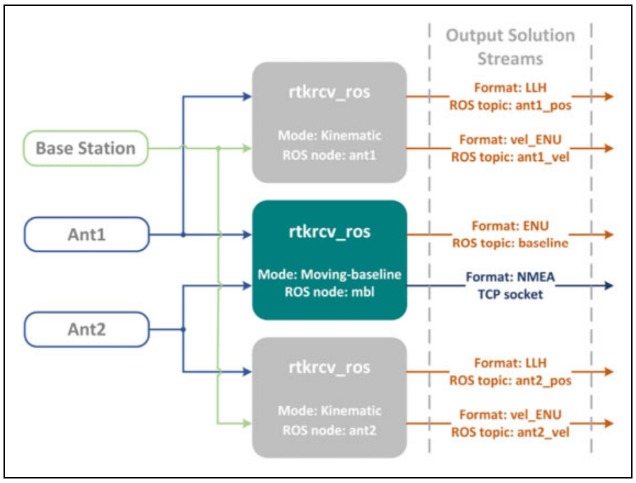


Figure 4. Architecture of the dual antenna GNSS system developed for the AUV. Two *rtkrcv_ros* instances compute global positioning solutions, by combining observations from each antenna with the fixed base station. Another *rtkrcv_ros* process combines both antennas, in moving-baseline mode, to obtain a relative position measurement.

angles can be executed through simple trigonometric expressions, following the direct method from,[8] giving:

$$\psi = \tan^{-1}\left(\frac{vb_N}{vb_E}\right) \tag{2}$$

$$\theta = \tan^{-1}\left(\frac{vb_U}{\sqrt{vb_E^2 + vb_N^2}}\right) \tag{3}$$

The ENU output format contains 6-SD parameters, which allow the reconstruction of the covariance matrix associated with the 3-D baseline measurement. The SD, σ_ψ, associated with the computed heading angle is obtained by projecting the relative position SDs as follows:

$$\sigma_\psi = \nabla\psi . \begin{bmatrix} \sigma vb_E \\ \sigma vb_N \end{bmatrix} . \nabla\psi^T \tag{4}$$

where $\nabla\psi$ indicates the Jacobian of equation (2) with respect to vb_E and vb_N.

Similarly, the SD associated with the pitch angle ($\sigma\theta$) is obtained as follows:

$$\sigma_\theta = \nabla\theta . \begin{bmatrix} \sigma vb_E \\ \sigma vb_N \\ \sigma vb_U \end{bmatrix} . \nabla\theta^T \tag{5}$$

where $\nabla\theta$ indicates the Jacobian of equation (3) with respect to vb_E, vb_N, and vb_U.

Observation synchronization. In the moving-baseline mode, the activation of the new observation synchronization method, by setting option *pos2-syncsol=on*, is recommended to ensure consistent baseline measurements. The

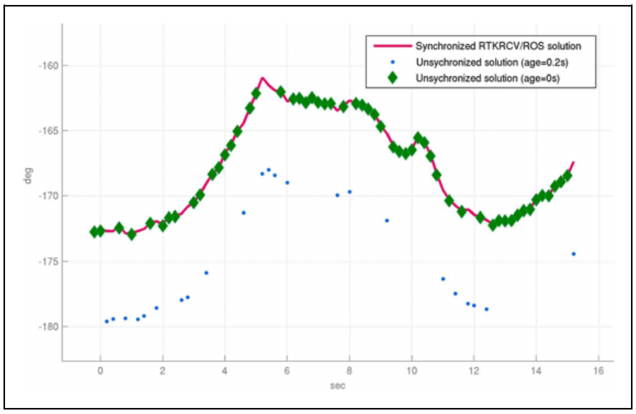


Figure 5. Impact of the base station measurement delay on the computed heading angle. Solutions resulting from delayed base station observations (age=0.2) do not agree with results obtained when rover and base observations are synchronized.

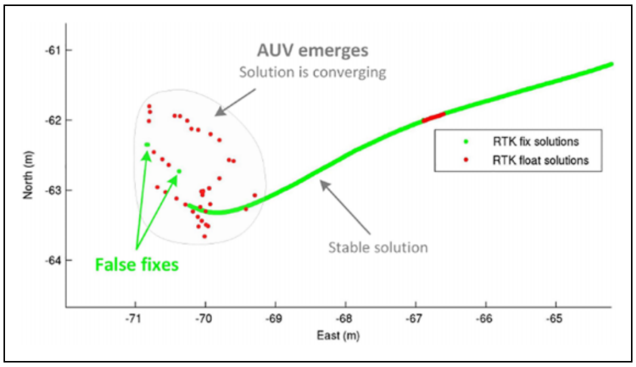


Figure 6. Typical output of the RTKRCV in kinematic mode when the AUV returns to the surface. The RTK process can take several minutes to converge, with false fixes occurring frequently. After convergence, either fix and float solutions remain consistent.

effects of combining observations from different epochs is evaluated here for the AUV's case.

The configuration on the AUV is particularly challenging due to the short baseline of 70 cm between antennas onboard. For the moving-baseline process, with the new synchronization mechanism inactive, base station observations are often combined with no delay (age = 0 s) or delayed by one epoch (age = 0.2 s; 200 ms at 5 Hz data rate). Despite the AUV's slow cruising speed of around 1 m/s, the combination of delayed base station observations causes considerable positioning error, with noticeable degradation on the computed heading angle, as illustrated in Figure 5.

A notorious degradation is visible when outdated base measurements are considered. The impact on the computed heading angle exceeds 5°, which surpasses by far the admissible error for this particular application. Figure 5 also demonstrates the benefit of the new synchronization mechanism, on enforcing perfect synchronization between the two observation streams, to achieve consistent solutions.

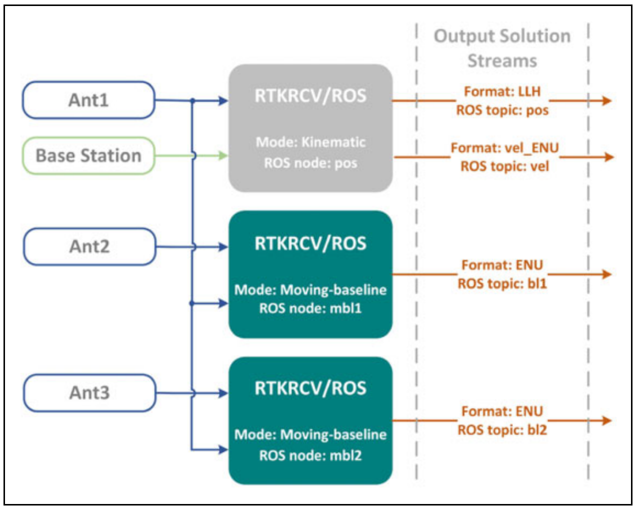


Figure 7. Triple antenna GNSS system for position velocity and attitude determination. One kinematic RTK process is used to compute a global positioning solution, by combining one antenna onboard with the fixed base station. Two relative baseline measurements are obtained between antennas onboard. Several outputs are used to communicate the solutions, including the new velocity format.

ROS service control commands. The new ROS service, built into the *rtkrcv_ros* package, allows external ROS applications to control the execution of the positioning algorithm. This feature comes in handy to mitigate inconsistent solutions and false fixes, which usually occur when the vehicle returns to the surface after long dive periods. As demonstrated in Figure 6, the RTK algorithm takes some time to converge after reacquiring GNSS observations, producing several outliers in the meanwhile. Outliers, and especially false fixes, are difficult to identify in real time, causing significant impact on the localization estimate.

This behavior is not restricted to underwater vehicles only. All applications, where GNSS signals can be lost for some time, are susceptible. It may be the case of a robot traveling through a tunnel, or entering an indoor environment.

The most effective way found to avoid false fixes, improve positioning precision, and decrease convergence time altogether consists on stopping the positioning process when the vehicle dives, restarting the execution as soon as the platform reaches the surface. Resorting to the ROS service, this task was automated by an external application that controls the RTK_server based on the estimated vehicle depth.

Triple antenna example

This last example demonstrates how several *rtkrcv_ros* nodes can be combined to measure position, velocity, and orientation, for a platform equipped with a triple antenna GNSS system. As depicted in Figure 7, a single *rtkrcv_ros* node, performing RTK between one onboard antenna and a fixed base station, is sufficient to simultaneously provide a global positioning solution and a relative velocity measurement with respect to the base station. When requesting the new velocity output format, as in Figure 7, the estimation of rover dynamics must be activated by setting configuration option *pos1-dynamics=on*. Similarly to the AUV's case, redundant solutions could be computed by repeating the same process for the other antennas.

The two additional *rtkrcv_ros* nodes represented in Figure 7, run in the moving-baseline RTK mode, to implement two relative baselines between the three antennas onboard. If the three antennas are carefully placed with respect to the body reference frame, such that the two established baselines form an orthogonal configuration, the three Euler angles defining the platform's attitude can be computed trough the direct method, as previously demonstrated in.[9] For other geometric arrangements, the least square method can be applied.[8,10]

Conclusion

In the context of mobile robotics, the ability to self-localize constitutes an essential competence. For outdoor agents, periodic global positioning references, obtained from the GNSS system, reveal extremely advantageous, especially when coupled with precise positioning techniques for improved accuracy. The reported implementation facilitates the integration of real-time precise positioning routines, by translating the RTKRCV application to the ROS environment. The new *rtkrcv_ros* package expands the information retrieved from the internal estimation algorithm and offers slight improvements to enhance accuracy.

The *rtkrcv_ros* package's main features include the possibility to publish all output solutions, including the new velocity output format, through ROS topics. The control over the core positioning algorithm via a ROS service, as an alternative to the original command line console. A new observation synchronization strategy was also developed to ensure the consistency of the moving-baseline positioning mode.

The package offers full configuration flexibility, maintaining all original options. This enables the design of complex positioning networks to target demanding robotic applications. In the quest to increase robustness, attain long-term operations and improve accuracy and autonomy of outdoor robots, multiple antenna GNSS systems can be easily implemented to retrieve additional localization references, such as full attitude and 3-D relative velocity.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work is financed by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, within project UIDB/50014/2020.

ORCID iD

António Ferreira https://orcid.org/0000-0002-6091-1549

References

1. Takasu T and Yasuda A. Development of the low-cost RTK-GPS receiver with an open source program package RTKLIB. In: *International Symposium on GPS/GNSS*, Jeju, Korea, 4–6 November 2009.
2. Quigley M, Gerkey BP, Conley K, et al. ROS: an open-source robot operating system. In: *ICRA Workshop on Open Source Software*, Kobe, Japan, 17 May 2009.
3. Fraunhofer's GNSS package. http://wiki.ros.org/gnss (accessed 31 March 2019).
4. ETHZ's rtklibros package. https://github.com/ethz-asl/rtklibros (accessed 31 March 2019).
5. The rtkrcv_ros package. https://github.com/ajbfinesc/rtkrcv_ros (accessed 31 March 2019).
6. Grieneisen D. Real time kinematic GPS for micro aerial vehicles. Master Thesis, Eidgenössische Technische Hochschule Zürich (ETHz), 2012.
7. Gade K. The seven ways to find heading. *J Navig* 2016; 69(5): 955–970.
8. Dai Z, Knedlik S, and Loffeld O. A MATLAB toolbox for attitude determination with GPS multi-antenna systems. *GPS Solut* 2008; 13: 241–248.
9. Almeida J, Ferreira A, Matias B, et al. ¡VAMOS! underwater mining machine navigation system. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain, 1–5 October 2018, pp. 1520–1526.
10. Lu G. Development of a GPS multi-antenna system for attitude determination. PhD Thesis, University of Calgary, 1995.