applied sciences

MDPI

Article

Robust-Extended Kalman Filter and Long Short-Term Memory Combination to Enhance the Quality of Single Point Positioning

Truong-Ngoc Tan [1,*], Ali Khenchaf [1], Fabrice Comblet [1], Pierre Franck [2], Jean-Marc Champeyroux [2] and Olivier Reichert [2]

1 Lab-STICC CNRS UMR 6285, ENSTA Bretagne, 2 Rue François Verny, 29806 Brest CEDEX 9, France; ali.khenchaf@ensta-bretagne.fr (A.K.); fabrice.comblet@ensta-bretagne.fr (F.C.)

2 Safran Electronics & Defense, 21 Avenue du Gros Chêne, 95610 Eragny sur Oise, France; pierre.franck@safrangroup.com (P.F.); jean-marc.champeyroux@safrangroup.com (J.-M.C.); olivier.reichert@safrangroup.com (O.R.)

* Correspondence: truongngoctanbn@gmail.com

Received: 22 April 2020; Accepted: 16 June 2020; Published: 24 June 2020

Abstract: In the recent years, multi-constellation and multi-frequency have improved the positioning precision in GNSS applications and significantly expanded the range of applications to new areas and services. However, the use of multiple signals presents advantages as well as disadvantages, since they may contain poor quality signals that negatively impact the position precision. The objective of this study is to improve the Single Point Positioning (SPP) accuracy using multi-GNSS data fusion. We propose the use of robust-Extended Kalman Filter (referred to as robust-EKF hereafter) to eliminate outliers. The robust-EKF used in the present work combines the Extended Kalman Filter with the Iterative ReWeighted Least Squares (IRWLS) and the Receiver Autonomous Integrity Monitoring (RAIM). The weight matrix in IRWLS is defined by the MM Estimation method which is a robust statistics approach for more efficient statistical data analysis with high breaking point. The RAIM algorithm is used to check the accuracy of the protection zone of the user. We apply the robust-EKF method along with the robust combination of GPS, Galileo and GLONASS data from ABMF base station, which significantly improves the position accuracy by about 84% compared to the non-robust data combination. ABMF station is a GNSS reception station managed by Météo-France in Guadeloupe . Thereafter, ABMF will refer to the acronym used to designate this station. Although robust-EKF demonstrates improvement in the position accuracy, its outputs might contain errors that are difficult to estimate. Therefore, an algorithm that can predetermine the error produced by robust-EKF is needed. For this purpose, the long short-term memory (LSTM) method is proposed as an adapted Deep Learning-Based approach. In this paper, LSTM is considered as a de-noising filter and the new method is proposed as a hybrid combination of robust-EKF and LSTM which is denoted rEKF-LSTM. The position precision greatly improves by about 95% compared to the non-robust combination of data from ABMF base station. In order to assess the rEKF-LSTM method, data from other base stations are tested. The position precision is enhanced by about 87%, 77% and 93% using the rEKF-LSTM compared to the non-robust combination of data from three other base stations AJAC, GRAC and LMMF in France, respectively.

Keywords: GNSS; data fusion; MM-Estimation; IRWLS; robust-EKF; LSTM; rEKF-LSTM

1. Introduction

Nowadays, positioning services can greatly improve life quality by covering a wide range of applications such as automatic driving, intelligent transportation, agriculture and so on. In this

context, the Global Navigation Satellite System (GNSS) is an optimal infrastructure to perform accurate positioning. The GNSS-based absolute positioning technologies can be called single point positioning (SPP) technologies and offer multifold advantages, such as the absence of restriction by the inter-station distance, low cost and simple data processing.

GNSS brings more signals to enhance the accuracy of user position. However, a main challenge for multi-constellation GNSS positioning is to determine the reliable satellite measurements from all visible satellites for inferring true positions. The estimation precision depends on the ability of the receiver to get the position in the presence of outlying observations that can affect the data. Multipath and NLoS (Non-Line Of Sight) signals can cause errors in the satellite observations for the receiver moving in harsh environments such as urban canyons, under crowded trees, or inside tunnels. It is thus challenging to achieve the positioning accuracy and reliability in such environments and it is required to adopt robust positioning estimation techniques to prevent the effects of possible wrong or outlier satellite observations on the user position. For this purpose, several algorithms and methods have been proposed to improve the GNSS receiver performance in terms of positioning accuracy. Receiver autonomous integrity monitoring (RAIM) is a commonly used approach to reject outlier observations using pseudo-range measurement redundancy to maintain the positioning accuracy by isolating the contaminated measurements from the good ones [1–3]. However, for user position estimation in urban environments, visible and available satellites are rapidly changing and it is difficult to use the RAIM for satellite selection. Besides the satellite selection, Weighted Least-Squares Estimation (WLSE) is often used for calculating and assigning weights to the GNSS observations [4–9]. The weight model is defined based on the elevation angle and the signal to noise ratio. However, it suffers from the same shortcomings as RAIM since the satellite elevation angle and the signal to noise ratio can be impacted by the multipath and radio interferences, especially in urban canyons. This has been highlighted in previous works [8,9] where we used the WLSE based on the elevation angle and signal to noise ratio. The positioning accuracy can improve, but the estimations remain unstable. In consequence, the robust-Extended Kalman Filter (robust-EKF) [10–18] is proposed in this paper to address this issue. Robust-EKF is the combination of the Extended Kalman Filter with Iterative Reweighted Least Squares algorithm with a weight matrix defined by MM-Estimation, and using the RAIM algorithm to check the protection zone of user.

The robust-EKF is based on robust statistical estimation which is employed to reduce the negative impact of the inaccurate pseudo-range measurements. In the statistical framework, robust estimation algorithms are assessed based on three factors: efficiency, stability and breakdown point. Efficiency is a measure of the performance of an estimator; stability is a measure of how small deviations from the model assumptions affect the performance; whereas the breakdown point is defined as the measure of the maximal fraction of outlier data that the estimator can handle without resulting in an incorrect final estimate. The role of a robust algorithm is to harmonize these three factors. Multiple robust statistical estimation techniques have been discussed including M-Estimation, S-Estimation, MM-Estimation etc. [19]. In this paper, the MM-Estimation proposed by Yohai [20] is used to define the weight matrix for its efficient statistical data analysis with a high breakdown point. Iterative Reweighted Least Squares (IRWLS) [21,22] is used for optimizating results by an iterative method in which each step involves solving a weighted least squares problem. Even though combining MM-Estimation and IRWLS algorithm can eliminate the outlier data, a method to check user's protection zone is necessary. In this case, the RAIM algorithm [23] is proposed to calculate the Horizontal Protection Level (HPL).

Furthermore, although robust-EKF can significantly improve the position accuracy, if the initial state estimate is erroneous, or if the process is incorrectly modelled, the final results are negatively impacted. Therefore, an algorithm which can predetermine the behavior error of the robust-EKF is needed. Recently, "deep learning" is known as one of the powerful methods to apply in many domains such as automatic speech recognition, image recognition, medical image analysis, GNSS, etc. Long Short-Term Memory (LSTM) is an artificial recurrent neural network architecture used in the field of deep learning and known as an effective method to infer predictions based on time series data [24–28].

This paper hence proposes to use LSTM as a denoising filter. Figure 1 summarizes the global flowchart for the position determination using the proposed approach.

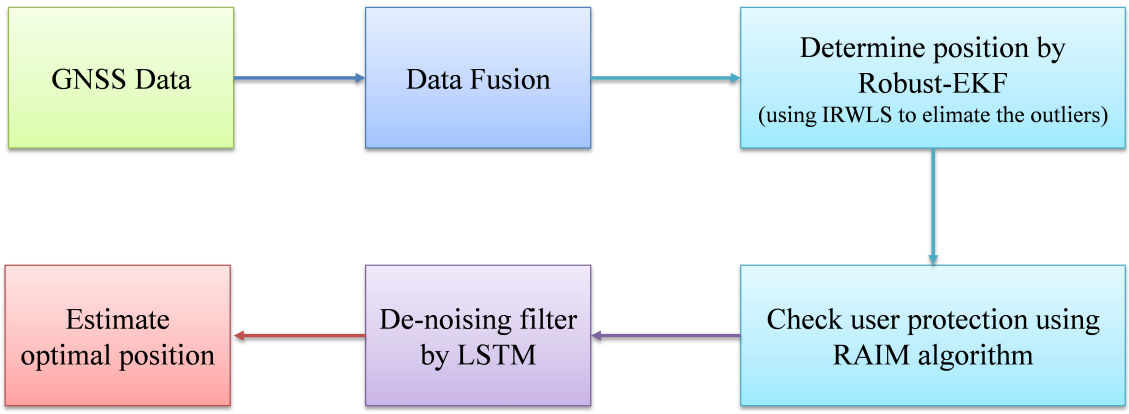

Figure 1. Global flowchart for the position estimation.

The remainder of this paper is organized as follows: Section 2 introduces the single point positioning technology; Section 3 presents the Robust-Extended Kalman Filter; Section 4 shows the experimental results of applying robust-EKF; Section 5 presents the de-noising filter method and experimental results; Section 6 presents the conclusions and perspectives.

2. Single Point Positioning Technology

The Single Point Positioning technology can provide meter-level positioning accuracy for emerging multi-GNSS integration and opens various new prospects. GNSS brings more signals to improve the user position and velocity accuracy. In this paper, observation models of three satellite systems (GPS, Galileo and GLONASS) are combined. For combining these satellites data, two important things are paid attention to: coordinate reference and time reference.

With respect to the coordinate reference, the coordinate systems of GPS, Galileo and GLONASS satellites adopt the broadcast orbits of WGS-84, GTRF and PZ90.11, respectively. Even though their coordinates are different, the disparity is around several centimeters as stated in [29]. As a result, their coordinate systems are considered the same.

As for the time reference, GPS, Galileo and GLONASS systems define their own time scale. These time systems are presented in the RINEX report version 3.03 [30]. GPS time ($GPST$) runs parallel to UTC (Universal Time Coordinated) (1 microsecond difference), but it is a continuous time scale that does not insert any leap seconds. Galileo runs on Galileo System Time (GST) which is nearly identical to GPS time (tens of nanoseconds difference). GLONASS is basically running on UTC or, more precisely, GLONASS time ($GLONASST$) linked to UTC(SU). It is not a continuous time, i.e., it introduces the same leap seconds as UTC. The reported GLONASS time has the same hours as UTC and not UTC+3h as the original GLONASS System Time. Apart from the mild errors in the realization of the different time systems, the relation between them is given by:

$$GLONASST = UTC \quad \text{and} \quad GPST = GST = UTC + \Delta tLS + \tau GPS \tag{1}$$

where ΔtLS is the time difference between $GPST$ and UTC due to leap seconds (1 January 2019, ΔtLS = 18), τ_{GPS} is the fractional time offset between the two systems (GPS and GLONASS).

After synchronization of the coordinate and time systems, the position and velocity estimation models are presented in the following.

Pseudo-range measurements are considered as the sum of the distance to the satellite, the receiver clock error, the ionospheric and tropospheric errors and multi-paths. They are expressed as follows:

$$\rho^{G,E,R} = \sqrt{(x_{G,E,R} - x_u)^2 + (y_{G,E,R} - y_u)^2 + (z_{G,E,R} - z_u)^2} + c\left(\delta t_{G,E,R} - \delta TS_{G,E,R}\right) + Iono_{G,E,R} + Tropo_{G,E,R} + Mulpa_{G,E,R} + \varepsilon_{\rho,G,E,R} \quad (2)$$

where the superscripts/subscripts G, E and R refer to GPS, Galileo and GLONASS satellites respectively; ρ is the measured pseudo-range; c is the speed of light in vacuum; $[x_u, y_u, z_u]$ is the user position; $[x_{G,E,R}, y_{G,E,R}, z_{G,E,R}]$ are positions of GPS, Galileo and GLONASS satellites respectively; δt is the user clock offset; δTS is the satellite clock offset; $Iono$ is the ionospheric delay; $Tropo$ is the tropospheric delay; $Mulpa$ are the multi-paths and ε_ρ is measured noise.

Before the determination of user position and velocity, it is required to correct the GNSS errors that are the satelllites clocks bias, the ionospheric errors, the tropospheric errors, and the multi-paths. The corrections of the satellite clock for GPS and GLONASS are presented in [31] (pp. 88–89), whereas the correction of the satellite clock of GLONASS is discussed in [30] (p. 34). We use the Saastamoinen model to correct the tropospheric errors as in [32] (pp. 135–137). We correct the ionospheric errors using the dual-frequency method in [23] (p. 286). While the multi-paths are estimated by the combination of the code pseudo-ranges and the phase pseudo-ranges in [23] (pp. 290–291). After compensating for satellite clock bias, ionospheric errors, tropospheric errors, and multi-paths from Equation (2), the corrected pseudo-ranges can be written as:

$$\rho_c^{G,E,R} = r^{G,E,R} + c\delta t_{G,E,R} + \varepsilon_{\rho,G,E,R} \quad (3)$$

where ρ_c is the corrected pseudo-range; r is the true range from the satellite to the user:

$$r^{G,E,R} = \sqrt{(x_{G,E,R} - x_u)^2 + (y_{G,E,R} - y_u)^2 + (z_{G,E,R} - z_u)^2} \quad (4)$$

Given the Doppler effect, the pseudo-range rate $\dot{\rho}$ can be computed as [31]:

$$\dot{\rho} = -\frac{Dc}{f} \quad (5)$$

where D is the Doppler effect; c is the speed of light and f is the satellite transmitted frequency.

The velocity can be estimated from the pseudo-range rate, starting by differentiating Equation (3) to obtain:

$$\dot{\rho}_c^{G,E,R} = \dot{r}^{G,E,R} + c\delta\dot{t}_{G,E,R} + \varepsilon_{\dot{\rho},G,E,R} \quad (6)$$

where $\delta\dot{t}$ is the user's clock drift (sec/sec); $\varepsilon_{\dot{\rho}}$ is the error in observation (meters/sec); and $\dot{r}$ is the true range rate expressed as:

$$\dot{r}^{G,E,R} = 1_x\left(v_{x,G,E,R} - v_{x,u}\right) + 1_y\left(v_{y,G,E,R} - v_{y,u}\right) + 1_z\left(v_{z,G,E,R} - v_{z,u}\right) \quad (7)$$

where $\left[v_{x,G,E,R}, v_{y,G,E,R}, v_{z,G,E,R}\right]$ is the satellite's velocity; $\left[v_{x,u}, v_{y,u}, v_{z,u}\right]$ is the true user's velocity; and $\left[1_x, 1_y, 1_z\right]$ is the true line of sight unit vector from the satellite to the user:

$$\left[1_x, 1_y, 1_z\right] = \frac{\left[\left(x_{G,E,R} - x_u\right), \left(y_{G,E,R} - y_u\right), \left(z_{G,E,R} - z_u\right)\right]}{\sqrt{(x_{G,E,R} - x_u)^2 + (y_{G,E,R} - y_u)^2 + (z_{G,E,R} - z_u)^2}} \quad (8)$$

Thus, the combined equations for the corrected pseudo-ranges (Equation (3)) and corrected pseudo-range rates (Equation (6)) can be as expressed as follows:

$$\rho_c^{G,E,R} = r^{G,E,R} + c\delta t_{G,E,R} + \varepsilon_{\rho,G,E,R} \quad \text{and} \quad \dot{\rho}_c^{G,E,R} = \dot{r}^{G,E,R} + c\delta\dot{t}_{G,E,R} + \varepsilon_{\dot{\rho},G,E,R} \quad (9)$$

To simultaneously estimate the user position and velocity, the state vector is defined from Equations (7)–(9) as follows:

$$X = \begin{bmatrix} x_u & v_{x,u} & y_u & v_{y,u} & z_u & v_{z,u} & c\delta t_G & c\delta \dot{t}_G & c\delta t_E & c\delta \dot{t}_E & c\delta t_R & c\delta \dot{t}_R \end{bmatrix}^T \tag{10}$$

This vector will be used in robust extended Kalman filter algorithm detailed in Section 3.

3. Robust Extended Kalman Filter

Extended Kalman Filter (EKF) is an extended application of Kalman filter in solving nonlinear optimal filtering problems. It is used in a wide range of technological fields (radar, electronic vision, communication). However, in hybrid positioning, the large outliers can negatively impact its application to position estimation. According to [17,18], the resulting Kalman filter outliers can be sorted into three types: observation, innovation and structural outliers. In this paper, robust extended Kalman filter is used to handle observation and innovation outliers. Robust-EKF is presented as the combination of the Extended Kalman Filter with the Iterative Reweighted Least Squares algorithm (Section 3.3) with a weight matrix determined by MM-estimation (Section 3.2). Moreover, a combination of Robust-EKF with the RAIM algorithm (Section 3.4) is used to check the accuracy of user zone.

3.1. Robust Extended Kalman Filter Model

The state space representation for the extended Kalman filter model is defined by two equations as follows:

State Equation :

$$X_k = f\left(X_{k-1}\right) + \epsilon_k \tag{11}$$

Observation Equation :

$$Z_k = h\left(X_k\right) + e_k \tag{12}$$

where X_k is the state vector and Z_k is the measurement vector at time t_k; ϵ_k is the vector that conveys the system error sources; e_k is the vector that represents the measurement error sources; $f()$ is the function for state transition and $h()$ is the function for the measurement.

IRWLS works on linear regression to handle the problem of outliers in the data. Therefore, converting a non-linear model into a linear model is necessary to apply the robust-EKF that consists in three steps: Linearizing equations, reforming filter and updating.

***Step 1**: Linearizing equations*

The linearized state in Equation (11) is described as:

$$A_k = \frac{\partial f\left(X_{k-1}\right)}{\partial X} \tag{13}$$

where A_k is the Jacobian of the process model.

The predicted state is determined by:

$$X_k^- = A_k X_{k-1} \tag{14}$$

The relation between the true state X_K and its prediction $\left(\mathrm{X_k^-}\right)$ can be written as:

$$\mathrm{X_k^-} = \mathrm{X_k} - \delta_\mathrm{k} \tag{15}$$

where δ_k is the error between the true state and its prediction.

Linearizing Equation (12) using the first order Taylor series expansion with the predicted state vector $\left(\mathrm{X_k^-}\right)$:

$$Z_k = h\left(\mathrm{X}_\mathrm{k}^-\right) + \left.\frac{\partial h}{\partial X}\right|_{X=X_k^-} \left(X_k - \mathrm{X}_\mathrm{k}^-\right) + e_k \tag{16}$$

By noting: $\left.\frac{\partial h}{\partial X}\right|_{X=X_k^-} = H_k$, Equation (16) can be written as:

$$Z_k = h\left(\mathrm{X}_\mathrm{k}^-\right) + H_k\left(X_k - \mathrm{X}_\mathrm{k}^-\right) + e_k \tag{17}$$

Or

$$Z_k - h\left(\mathrm{X}_\mathrm{k}^-\right) + H_k\mathrm{X}_\mathrm{k}^- = H_k X_k + e_k \tag{18}$$

***Step 2**: Reforming the filter*

This step will reform the filter to a regression equation. To perform this, combining Equations (15) and (18) together in a matrix form yields the following:

$$\begin{bmatrix} X_k^- \\ Z_k - h\left(X_k^-\right) + H_k X_k^- \end{bmatrix} = \begin{bmatrix} I \\ H_k \end{bmatrix} X_k + \begin{bmatrix} -\delta_k \\ e_k \end{bmatrix} \tag{19}$$

where I is the identity matrix.

Equation (19) is expressed in a compact form as:

$$\check{Z}_k = \check{H}_k X_k + \check{e}_k \tag{20}$$

The covariance matrix of $\check{e}_k$ is given by:

$$\check{R}_k = \begin{bmatrix} P_k^- & 0 \\ 0 & R_k \end{bmatrix} = L_k L_k^T \tag{21}$$

where $R_k = E\left(e_k e_k^T\right)$ is the measurement covariance matrix and $P_k^- = E\left[\delta_k \delta_k^T\right]$ is the covariance matrix of the predicted error. P_k^- is determined by:

$$P_k^- = A_k P_{k-1} A_k^T + Q_k \tag{22}$$

where Q_k is the covariance matrix of the process.

L_k in Equation (21) is calculated using Cholesky decomposition. To perform pre-whitening, Equation (20) is multiplyed by L_k^{-1} and by defining: $y_k = L_k^{-1}\check{Z}_k$, $G_k = L_k^{-1}\check{H}_k$, $\zeta_k = L_k^{-1}\check{e}_k$. Equation (20) becomes the regression equation written as:

$$y_k = G_k X_k + \zeta_k \tag{23}$$

***Step 3**: Updating*

This step consists in updating the estimated state and checking the position accuracy using RAIM algorithm presented in section "RAIM algorithm" (Section 3.4). The estimated state $\hat{X}_k$ in Equation (23) is optimized by the IRWLS algorithm presented in section "Iterative Reweighted Least Squares Algorithm" (Section 3.3). The final state estimate $\hat{X}_k$ is given by:

$$\hat{X}_k = \left(G_k^T W_k G_k\right)^{-1} G_k^T W_k y_k \tag{24}$$

where W_k is the diagonal weight matrix at time t_k. This matrix is defined in section "MM Estimation theory" (Section 3.2) and is calculated in section "Iterative Reweighted Least Squares Algorithm" (Section 3.3).

The error covariance matrix is updated as follows:

$$P_k = (I - KH_k)\, P_k^- \tag{25}$$

where K the gain of the filter given by:

$$K = P_k^- H_k^T \left(H_k P_k^- H_k^T + R_k\right)^{-1} \tag{26}$$

3.2. MM Estimation Theory

In regression analysis, using the least squares method would not be enough to handle outliers or extreme observations. We therefore resort to the MM-Estimation which is known as a robust statistics method providing a more efficient analysis of statistical data with high breaking point as demonstrated in [20,22].

The linear equation in this paper is written as follows:

$$y = GX + e \tag{27}$$

where y is the new observation vector after combining system and observation data; G is the combined system and observation matrices corresponding to y; X is the unknown state vector and e is the noise vector corresponding to y.

The robust method is used to bound the influence of outliers on the X state estimates. The goal of the robust estimator is to seek the best fitting criterion to the real observations in order to reduce the effect of abnormal data. Generally, a robust estimator minimizes the cost function $J(r)$ given by:

$$J(r) = \sum_{i=1}^{n} \mu\left(r_i\right) = \sum_{i=1}^{n} \mu\left(y_i - G_i X\right) \tag{28}$$

where y_i is the ith observation, G_i is the transformation vector for the ith observation, and $y_i - G_i X$ is the residual error for the ith observation. $\mu(r)$ is the objective function that gives the contribution of each residual to the cost function $J\left(r\right)$.

The scaled estimation is used to minimize the weights out of the restrained observations dynamically:

$$\min \sum_{i=1}^{n} \mu\left(\frac{r_i}{s}\right) = \sum_{i=1}^{n} \mu\left(\frac{y_i - G_i X}{s}\right) \tag{29}$$

where s is the scale factor.

Let φ be the derivative of the objective function for the unknown X, $\varphi = \mu'|_X$. Differentiating the objective function in Equation (29) and setting the derivative to zero:

$$-\frac{1}{s} \sum_{i=1}^{n} \varphi\left(\frac{y_i - G_i X}{s}\right) G_i = 0 \tag{30}$$

Which is equivalent to:

$$\sum_{i=1}^{n} \varphi\left(\frac{y_i - G_i X}{s}\right) G_i = 0 \tag{31}$$

By introducing a weight $w(\check{r}_{s,i})$ to consider the effects of each scaled residual $\check{r}_{s,i} = \frac{y_i - G_i X}{s}$, Equation (31) can be modified as:

$$\sum_{i=1}^{n} w\left(\check{r}_{s,i}\right) \varphi\left(\check{r}_{s,i}\right) G_i = 0 \tag{32}$$

The solution of Equation (32) is a weighted least squares estimation. Nevertheless, the weights are determined by the scaled residuals, which rely upon the estimated coefficients, which themselves

rely upon the weights. For this reason, the Iterative Re-Weighted Least Squares method is used and presented in section "Iterative Reweighted Least Squares Algorithm" (Section 3.3) to optimize the solution. The weight function plays an important role and some variants are presented in Table 1.

Table 1. Weighting functions [22].

Name	Weight Function
Huber	$w(r_i) = \begin{cases} 1 & \lvert r_i \rvert \leq \alpha \\ \alpha / \lvert r_i \rvert & \lvert r_i \rvert > \alpha \end{cases}$
Bi-Tukey	$w(r_i) = \begin{cases} 1 & \lvert r_i \rvert \leq \alpha \\ 0 & \lvert r_i \rvert \geq \alpha \end{cases}$
Bi-Square	$w(r_i) = \begin{cases} [1 - (\lvert r_i \rvert / \alpha)]^2 & \lvert r_i \rvert \leq \alpha \\ 0 & \lvert r_i \rvert > \alpha \end{cases}$

In Table 1, r_i is the *ith* observation residual. α is a tuning constant to optimize the estimation with a balance between a high breakdown point and efficiency of the estimator [22].

3.3. Iterative Reweighted Least Squares Algorithm

An Iterative Reweighted Least Squares Algorithm (IRWLS) is performed to eliminate the outliers. The steps for the robust estimation procedure are defined as follow:

1. Find an initial estimate $\hat{X}_0$

$$\hat{X}_0 = \left(G_0^T G_0\right)^{-1} G_0^T y \tag{33}$$

2. Estimate the vector for initial residuals of the observations r_0:

$$r_0 = y - G_0 \hat{X}_0 \tag{34}$$

3. Define the initial scale value s_0 [33]

$$s_0 = 1.4826 \,\text{median}\,(|r_0|) \tag{35}$$

where "*median*" stands for the median function computed on the residuals vector r_0.

4. Estimate the initial diagonal weight matrix by MM-Estimation.

$$W_0(i,i) = \begin{cases} 1 & \left| \frac{r_0(i)}{s_0} \right| \leq \alpha \\ 0 & \left| \frac{r_0(i)}{s_0} \right| > \alpha \end{cases} \tag{36}$$

where $i = 1, \ldots, n$; n being the number of observations.

5. While (j is jth iteration)

   (a) Update the value of the matrix G_j at the jth iteration with $\hat{X}_{j-1}$

   (b) Solve the estimated state $\hat{X}_j$ using the weighted least-squared method

$$\hat{X}_j = \left(G_j^T W_{j-1} G_j\right)^{-1} G_j^T W_{j-1} y \tag{37}$$

   (c) Calculate the HPL (in Section 3.4)

   (d) If $\left\| \hat{X}_j - \hat{X}_{j-1} \right\| < 0.001$; break

(e) If $\left\|\hat{X}_j - \hat{X}_{j-1}\right\| > 0.001$; continue

(f) Update the estimated residuals

$$r_j = y - G_j\hat{X}_j \tag{38}$$

(g) Calculate the scale value

$$s_j = 1.4826\,\text{median}\left(|r_j|\right) \tag{39}$$

(h) Recalculate the diagonal weighted matrix using MM-Estimation:

$$W_j(i,i) = \begin{cases} 1 & \left|\frac{r_j(i)}{s_j}\right| \leq \alpha \\ 0 & \left|\frac{r_j(i)}{s_j}\right| > \alpha \end{cases} \tag{40}$$

(i) Go to step (a)

6. End

7. If $HPL \leq HAL$ then the estimated positions are accepted, (HAL: Horizontal Alert Limit)
If not, they are rejected.

In this paper, we use the bi-Tukey weight function in Equations (36) and (40). The HPL is calculated using the RAIM algorithm presented in Section 3.4.

3.4. RAIM Algorithm

In this paper, RAIM algorithm is used to detect positioning errors exceeding the alert limit. One of RAIM's outputs is the HPL, that is defined as the radius of a circle in the horizontal plane, centered at the true position. The Horizontal Alert Limit (HAL) is the maximum allowed HPL, which means that the estimated positions are accepted when $HPL \leq HAL$. Given the linear Equation (27), the estimated state vector $\hat{X}$ is determined by IRWLS as explained in Section 3.3:

$$\hat{X} = \left(G^T W G\right)^{-1} G^T W y \tag{41}$$

where W is the W matrix last obtained matrix in Equation (40).

The state error is calculated as the difference between the estimated state vector $\hat{X}$ and the true state vector X :

$$\begin{aligned} \delta X = \hat{X} - X &= \left(G^T W G\right)^{-1} G^T W y - X = \left(G^T W G\right)^{-1} G^T W (GX + e) - X \\ &= \left(G^T W G\right)^{-1} G^T W e \end{aligned} \tag{42}$$

The estimated residual $\hat{e}$ is determined by computing the difference between the measured observation y and the estimated observation ($G\hat{X}$):

$$\begin{aligned} \hat{e} = y - G\hat{X} &= y - G\left(G^T W G\right)^{-1} G^T W y = \left(I - G\left(G^T W G\right)^{-1} G^T W\right) y = Sy \\ &= \left(I - G\left(G^T W G\right)^{-1} G^T W\right)(GX + e) \\ &= GX + e - G\left(G^T W G\right)^{-1} G^T W G X - G\left(G^T W G\right)^{-1} G^T W e \\ &= GX + e - GX - G\left(G^T W G\right)^{-1} G^T W e = \left(I - G\left(G^T W G\right)^{-1} G^T W\right) e = Se \end{aligned} \tag{43}$$

where:

$$S = I - G\left(G^T W G\right)^{-1} G^T W \tag{44}$$

Note that S is a projector matrix and thus idempotent: $S^2 = S$. The a priori standard deviation of the pseudo-range is calculated as follows:

$$\sigma_0 = \sqrt{\hat{e}^T W \hat{e}} / \sqrt{n - p} \tag{45}$$

where n is the number of observations and p is the number of parameters in the unknown state X.

The position error in the state error (δX) of Equation (42) is defined in the (x, y, z) coordinate system. In practice, a local (e, n, u: east, north, up) coordinate system is more suitable. To transform from (x, y, z) coordinates to (e, n, u) coordinates, the F^T orthogonal transformation matrix is required [23]:

$$F^T = \begin{bmatrix} -\sin\lambda & \cos\lambda & 0 \\ -\sin\varphi\cos\lambda & -\sin\varphi\sin\lambda & \cos\varphi \\ \cos\varphi\cos\lambda & \cos\varphi\sin\lambda & \sin\varphi \end{bmatrix} \tag{46}$$

where φ is the longitude and λ the latitude.

Here, only the parameters of coordinates are considered. For example, the state vector X is defined following Equation (10) with the first, third and fifth positions in relation to the three (x, y, z) coordinates. Therefore, the new $\tilde{G}_0$ matrix is obtained as $\tilde{G}_0 = G(:, 1:2:5)$ and the position errors as $\delta\tilde{X}_0 = \delta X(1:2:5)$. The M matrix related to the errors in easting (dE), northing (dN) and upping (dU) is calculated as follows:

$$\delta X_{ENU} = \begin{bmatrix} \mathrm{dE} \\ \mathrm{dN} \\ \mathrm{dU} \end{bmatrix} = F^T \delta\tilde{X}_0 = F^T \left(\tilde{G}_0^T \tilde{G}_0\right)^{-1} \tilde{G}_0^T e = Me \tag{47}$$

Since the horizontal error related to easting and northing is considered, a matrix $\tilde{M}_0$ can be defined as $\tilde{M}_0 = M(1:2, :)$. In reality, none of the elements in e is equal to zero. However, when serious observation measure error occurs, some elements in e may be extremely larger than the others. Therefore, we only consider the ith observation that has a failure of magnitude β, while we suppose that the other elements are equal to zero. The error vector can thus be re-written as:

$$e = \begin{bmatrix} 0 & \cdots & \beta & \ldots & 0 \end{bmatrix}^T \tag{48}$$

The norm squared of δX_{EN} (errors in easting and northing) assuming this specific choice of e is computed as:

$$\|\delta X_{EN}\|^2 = e^T \tilde{M}_0^T \tilde{M}_0 e = \left(m_{1i}^2 + m_{2i}^2\right) \beta^2 \tag{49}$$

From Equation (43), $\hat{e} = Se$ is recalled. Then $S^T S = S$ and the diagonal entry (i, i) of S is denoted s_{ii} and we have:

$$\|\hat{e}\|^2 = \hat{e}^T \hat{e} = e^T S^T S e = |s_{ii}| \, \beta^2 \tag{50}$$

Combining Equations (49) and (50) gives:

$$\|\delta X_{EN}\|^2 = \frac{m_{1i}^2 + m_{2i}^2}{|s_{ii}|} \|\hat{e}\|^2 \tag{51}$$

Or

$$\|\delta X_{EN}\| = \sqrt{\frac{m_{1i}^2 + m_{2i}^2}{|s_{ii}|}} \|\hat{e}\| = \alpha_i \|\hat{e}\| \tag{52}$$

The obtained equation is a linear function with a straight line through the origin and a slope α_i. The slope α_i of the failure mode axis related to observation i is computed for all $i = 1, \ldots, n$ as:

$$\alpha_i = \sqrt{\frac{m_{1i}^2 + m_{2i}^2}{|s_{ii}|}} \tag{53}$$

The mode axis with the largest α_i is denoted α_{max} and the HPL is defined as:

$$HPL = \alpha_{\max}\sigma_0 \tag{54}$$

where σ_0 is calculated following Equation (45).

4. Experimental Results of Applying Robust-EKF

Numerous investigations on robust-EKF for GNSS data exist in the literature [13–15]. In [13], robust-EKF is used for GEO/IGSO/GPS Raw-PPP/INS data fusion. In [14], it is applied for GPS/Galileo/GLONASS data combination. In [15], it is used for GPS/GLONASS data combination. All these studies assert the significant improvement of the position precision brought by the application of robust-EKF. In this paper, we combine the robust-EKF with the RAIM algorithm to ensure the position precision. The details of this robust-EKF and the experimental results for GNSS data are presented in the sequel in Sections 4.1 and 4.2 respectively.

4.1. Applying Robust-EKF

From Equation (10), we can write the compact user state of the three systems GPS, Galileo and GLONASS as follows:

$$X = \begin{bmatrix} x_u & v_x & y_u & v_y & z_u & v_z & b_G & d_G & b_E & d_E & b_R & d_R \end{bmatrix}^T \tag{55}$$

where $[x_u, y_u, z_u]$ is the user position; $[v_x, v_y, v_z]$ is the user velocity; b_G, b_E and b_R are the errors in range due to the user's clock bias with GPS, Galileo and GLONASS time, respectively; d_G, d_E and d_R are the user's clock drifts with GPS, Galileo and GLONASS time respectively.

We then proceed to the steps of the robust Extended Kalman Filter:

***Step 1**: Linearizing equations*

By linearizing the function $f(X_{k-1})$, the state transition matrix A_k is obtained as a 6×6 block diagonal matrix of the form:

$$A_k = \left.\frac{\partial f}{X}\right|_{X=X_{k-1}} = \begin{bmatrix} A_x & 0 & 0 & 0 & 0 & 0 \\ 0 & A_y & 0 & 0 & 0 & 0 \\ 0 & 0 & A_z & 0 & 0 & 0 \\ 0 & 0 & 0 & A_{b,G} & 0 & 0 \\ 0 & 0 & 0 & 0 & A_{b,E} & 0 \\ 0 & 0 & 0 & 0 & 0 & A_{b,R} \end{bmatrix} \tag{56}$$

where $A_x = A_y = A_z = A_{b,G} = A_{b,E} = A_{b,R} = \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix}$, and ΔT is the time interval between two epochs.

The predicted state is then determined by:

$$X_k^- = A_k X_{k-1} \tag{57}$$

The relation between the true state (X_k) and its prediction (X_k^-) can be written as:

$$X_{\mathrm{k}}^- = \mathrm{X_k} - \delta_{\mathrm{k}} \tag{58}$$

The observation equation is defined as: $Z_k = h\left(X_k\right) + e_k$, where

$$Z_k = \begin{bmatrix} Z_{\rho,G} \\ Z_{\dot{\rho},G} \\ Z_{\rho,E} \\ Z_{\dot{\rho},E} \\ Z_{\rho,R} \\ Z_{\dot{\rho},R} \end{bmatrix} \tag{59}$$

where $Z_{\rho,G}$ and $Z_{\dot{\rho},G}$ are the pseudo range (ρ_G) and the pseudo range rate ($\dot{\rho}_G$) matrices of N_{gps} GPS satellites tracked (dimension ($N_{gps} \times 1$)); $Z_{\rho,E}$ and $Z_{\dot{\rho},E}$ are the pseudo range (ρ_E) and the pseudo range rate ($\dot{\rho}_E$) matrices of N_{gal} Galileo satellites tracked (dimension ($N_{gal} \times 1$)); $Z_{\rho,R}$ and $Z_{\dot{\rho},R}$ are the pseudo range (ρ_R) and the pseudo range rate ($\dot{\rho}_R$) matrices of N_{glo} GLONASS satellites tracked (dimension ($N_{glo} \times 1$)). The ρ_G, $\dot{\rho}_G$, ρ_E, $\dot{\rho}_E$, ρ_R, and $\dot{\rho}_R$ are calculated in Equation (9).

Following step 1 in Section 3.1, the linearized "Observation equation" using the first order Taylor series expansion with the predicted state vector X_k^- is written as:

$$Z_k - h\left(\mathrm{X}_\mathrm{k}^-\right) + H_k \mathrm{X}_\mathrm{k}^- = H_k X_k + e_k \tag{60}$$

where the matrix H_k is determined as the derivative of the function $h(X_k^-)$: $H_k = \left.\frac{\partial h}{\partial X}\right|_{X=X_k^-}$

$$H_k = \begin{bmatrix} H_{\rho,G} & 1 & 0 & 0 & 0 & 0 & 0 \\ H_{\dot{\rho},G} & 0 & 1 & 0 & 0 & 0 & 0 \\ H_{\rho,E} & 0 & 0 & 1 & 0 & 0 & 0 \\ H_{\dot{\rho},E} & 0 & 0 & 0 & 1 & 0 & 0 \\ H_{\rho,R} & 0 & 0 & 0 & 0 & 1 & 0 \\ H_{\dot{\rho},R} & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{61}$$

$H_{\rho,G}$, $H_{\rho,E}$ and $H_{\rho,R}$ are the Jacobians of the pseudo-range measurement model for GPS, Galileo and GLONASS data with the dimensions ($N_{gps} \times 6$), ($N_{gal} \times 6$) and ($N_{glo} \times 6$), respectively; $H_{\dot{\rho},G}$, $H_{\dot{\rho},E}$ and $H_{\dot{\rho},R}$ are the Jacobians of the pseudo-range rate measurement model for GPS, Galileo and GLONASS data with the dimensions ($N_{gps} \times 6$), ($N_{gal} \times 6$) and ($N_{glo} \times 6$), respectively.

$H_{\rho,G}$ is computed as :

$$H_{\rho,G} = \begin{bmatrix} \frac{\partial \rho_{c,G}^1}{\partial x} & 0 & \frac{\partial \rho_{c,G}^1}{\partial y} & 0 & \frac{\partial \rho_{c,G}^1}{\partial z} & 0 \\ \frac{\partial \rho_{c,G}^2}{\partial x} & 0 & \frac{\partial \rho_{c,G}^2}{\partial y} & 0 & \frac{\partial \rho_{c,G}^2}{\partial z} & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial \rho_{c,G}^{Ngps}}{\partial x} & 0 & \frac{\partial \rho_{c,G}^{Ngps}}{\partial y} & 0 & \frac{\partial \rho_{c,G}^{Ngps}}{\partial z} & 0 \end{bmatrix} \tag{62}$$

And $H_{\dot{\rho},G}$ is computed as:

$$H_{\dot{\rho},G} = \begin{bmatrix} 0 & \frac{\partial \rho_{c,G}^1}{\partial x} & 0 & \frac{\partial \rho_{c,G}^1}{\partial y} & 0 & \frac{\partial \rho_{c,G}^1}{\partial z} \\ 0 & \frac{\partial \rho_{c,G}^2}{\partial x} & 0 & \frac{\partial \rho_{c,G}^2}{\partial y} & 0 & \frac{\partial \rho_{c,G}^2}{\partial z} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \frac{\partial \rho_{c,G}^{Ngps}}{\partial x} & 0 & \frac{\partial \rho_{c,G}^{Ngps}}{\partial y} & 0 & \frac{\partial \rho_{c,G}^{Ngps}}{\partial z} \end{bmatrix} \tag{63}$$

where $\frac{\partial \rho^i_{c,G}}{\partial x} = \frac{-\left(x^i_G - x_u\right)}{\sqrt{\left(x^i_G - x_u\right)^2 + \left(y^i_G - y_u\right)^2 + \left(z^i_G - z_u\right)^2}}$; $\frac{\partial \rho^i_{c,G}}{\partial y} = \frac{-\left(y^i_G - y_u\right)}{\sqrt{\left(x^i_G - x_u\right)^2 + \left(y^i_G - y_u\right)^2 + \left(z^i_G - z_u\right)^2}}$; $\frac{\partial \rho^i_{c,G}}{\partial z} = \frac{-\left(z^i_G - z_u\right)}{\sqrt{\left(x^i_G - x_u\right)^2 + \left(y^i_G - y_u\right)^2 + \left(z^i_G - z_u\right)^2}}$, with $i = 1, \ldots, N_{gps}$.

$H_{\rho,E}$, $H_{\rho,R}$ and $H_{\dot{\rho},E}$, $H_{\dot{\rho},R}$ are computed similarly to $H_{\rho,G}$, and $H_{\dot{\rho},G}$ respectively.

Step 2: *Reforming filter*

Following step 1 and step 2 in Section 3.1, from Equations (58) and (60) we have the equation as Equation (19). We compact it as Equation (20), the compact regression equation is as follows:

$$\check{Z}_k = \check{H}_k X_k + \check{e}_k \tag{64}$$

The covariance matrix of $\check{e}_k$ is $\check{R}_k = \begin{bmatrix} P_k^- & 0 \\ 0 & R_k \end{bmatrix} = L_k L_k^T$; where $R_k = E(e_k e_k^T)$ is the measurement covariance matrix and $P_k^- = E\left[\delta_k \delta_k{}^T\right]$ is the covariance matrix of the predicted error, and both matrices are needed to determine L_k.

The determination of R_k is not straightforward and in this work the Exponential model is proposed to estimate it. The Exponential model formulation [23] depends on the elevation angle and is given by:

$$S_l = x_0 + x_1 e^{-\frac{El(l)}{x_2}}\ (m) \tag{65}$$

where l refers to the lth satellite; S_l is the elevation uncertainty (m); x_0 and x_1 are constant; $El(l)$ is the elevation angle of the lth satellite (in *degrees*) and x_2 is a scaled value of the elevation error (in *degrees*). x_0, x_1 and x_2 must be estimated empirically. After performing many tests, we chose the values $x_0 = 0.8$, $x_1 = 0.4$ and $x_2 = 12$. The normalization elevation uncertainty σ_l^2 is given by [7]:

$$\sigma_l^2 = S_l \Big/ \left(\prod_{l=1}^{n_s} S_l\right)^{n_s} \tag{66}$$

n_s being the number of the tracked satellites.

The covariance of the measurement noise of GPS satellite is determined as follows:

$$R_{k,G} = \begin{bmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_{N_{gps}}^2 \end{bmatrix} \tag{67}$$

where N_{gps} is the number of the tracked GPS satellites.

The covariance of the measurement noise of Galileo ($R_{k,E}$) and GLONASS ($R_{k,R}$) satellites are determined the same way as GPS satellite ($R_{k,G}$). Therefore, the covariance of the measurement noise of GPS, Galileo and GLONASS satellites is written as:

$$R_k = \begin{bmatrix} R_{k,G} & 0 & 0 & 0 & 0 & 0 \\ 0 & \gamma R_{k,G} & 0 & 0 & 0 & 0 \\ 0 & 0 & R_{k,E} & 0 & 0 & 0 \\ 0 & 0 & 0 & \gamma R_{k,E} & 0 & 0 \\ 0 & 0 & 0 & 0 & R_{k,R} & 0 \\ 0 & 0 & 0 & 0 & 0 & \gamma R_{k,R} \end{bmatrix} \tag{68}$$

where $\gamma R_{k,G}$, $\gamma R_{k,E}$ and $\gamma R_{k,R}$ are the covariance matrices of the pseudo-range rates of GPS, Galileo and GLONASS, respectively. γ is the covariance pseudo-range rate to the covariance pseudo-range

ratio and is empirically chosen; After performing several tests, the value $\gamma = 0.01$ was chosen in this paper.

The covariance prediction P_k^- is calculated by:

$$P_k^- = A_k P_{k-1} A_k^T + Q_k \tag{69}$$

where Q_k is the process covariance matrix. It plays an important role in the optimal recursive process and is written as:

$$Q_k = \int_{t_{k-1}}^{t_k} A(\tau) \sum A^T(\tau) d\tau \tag{70}$$

where $A(\tau) = diagonal(A_{x,\tau}, A_{y,\tau}, A_{z,\tau}, A_{b,G,\tau}, A_{b,E,\tau}, A_{b,R,\tau})$, with $A_{x,\tau} = A_{y,\tau} = A_{z,\tau} = A_{b,G,\tau} = A_{b,E,\tau} = A_{b,R,\tau} = \begin{bmatrix} 1 & \tau \\ 0 & 1 \end{bmatrix}$. The matrix $\sum$ is the spectral density matrix given by:

$$\sum = \text{diagonal}\left(S_x, S_{vx}, S_y, S_{vy}, S_z, S_{vz}, S_{b,G}, S_{d,G}, S_{b,E}, S_{d,E}, S_{b,R}, S_{d,R}\right) \tag{71}$$

where S_x, S_{vx}; S_y, S_{vy} & S_z, S_{vz} are the power spectral densities of position noise and speed noise in the x-, y- and z-coordinates, respectively; S_b is the power spectral density of the clock bias noise and S_d is the power spectral density of the frequency drift noise. To simply, we consider to the $\sum$ which is unchangeable.

The covariance matrix Q_k in Equation (70) is rewritten as:

$$Q_k = \text{diagonal}\left(Q_x, Q_y, Q_z, Q_{b,G}, Q_{b,E}, Q_{b,R}\right) \tag{72}$$

With

$$Q_x = \begin{bmatrix} S_x \Delta T + S_{vx}(\Delta T)^3/3 & S_{vx}(\Delta T)^2/2 \\ S_{vx}(\Delta T)^2/2 & S_{vx}\Delta T \end{bmatrix} \tag{73}$$

Q_y and Q_z are determined similarly to Q_x; and $Q_{b,G}$ is expressed as:

$$Q_{b,G} = \begin{bmatrix} S_{b,G} \Delta T + S_{d,G}(\Delta T)^3/3 & S_{d,G}(\Delta T)^2/2 \\ S_{d,G}(\Delta T)^2/2 & S_{d,G}\Delta T \end{bmatrix} \tag{74}$$

$Q_{b,E}$ and $Q_{b,R}$ are determined similarly to $Q_{b,G}$.

After the calculation of the covariance matrix R_k and the covariance prediction P_k^-, it is easy to determine the matrix L_k using the Cholesky decomposition. Multiplying Equation (64) by L_k^{-1} and defining: $y_k = L_k^{-1}\check{Z}_k, G_k = L_k^{-1}\check{H}_k, \zeta_k = L_k^{-1}\check{e}_k$, Equation (64) becomes the regression equation written as:

$$y_k = G_k X_k + \zeta_k \tag{75}$$

***Step 3**: Updating*

The estimated state $\hat{X}_K$ in Equation (75) is optimized by the IRWLS algorithm previously detailed in Section 3.3. After testing all weight functions in Section 3.2, we chose the bi-Tukey weight function to eliminate the outliers data and α is set to 1.756 (empirically chosen after many tests) for the IRWLS algorithm. The final estimated state is calculated as Equation (24): $\hat{X}_k = \left(G_k^T W_k G_k\right)^{-1} G_k^T W_k y_k$.

The error covariance matrix is updated as Equation (25): $P_k = (I - KH_k)\, P_k^-$; where K is the filter gain determined in Equation (26) : $K = P_k^- H_k^T \left(H_k P_k^- H_k^T + R_k\right)^{-1}$.

After the state estimation, RAIM algorithm in Section 3.4 is applied to check the position accuracy.

4.2. Experimental Results

To demonstrate the above algorithm, this study uses the GPS, Galileo and GLONASS data from ABMF station (RINEX version 2.11 (GPS and GLONASS) and version 3.02 (Galileo) format). This reference station is located in Guadeloupe. RINEX data and broadcast ephemeris data used for calculation were downloaded from RGP network and CDDIS for day of 1 January 2019. In order to assess the algorithm performance, five scenarios are investigated under MATLAB environment:

- Scenario #1: Navigation solution based on GPS data,
- Scenario #2: Navigation solution based on Galileo data,
- Scenario #3: Navigation solution based on GLONASS data,
- Scenario #4: Navigation solution based on GPS/Galileo/GLONASS data,
- Scenario #5: Navigation solution based on robust-EKF GPS/Galileo/GLONASS data.

Figure 2 illustrates the number visible satellites of GPS (red curve), Galileo (black curve), GLONASS (cyan curve), and combined GPS/Galileo/GLONASS (blue curve) systems over an elevation mask of 10°. It demonstrates that the number Galileo and GLONASS satellites is zero during some epochs. The average number of visible satellites is about 20 satellites when GPS, Galileo, and GLONASS are combined for positioning, which presents an advantage since they bring more signals to the receiver. However, it also presents a disadvantage since they may bring fault signals which can penalize the position estimation accuracy. The results are given Figure 3 which represents the position errors of the five scenarios with respect to time. dE, dN, dU are the errors in the East, North and Vertical components of the user's position estimates.

In Figure 3, the red curve pertains to scenario #1 with GPS data only. The black curve represents scenario #2 with Galileo data only. The cyan curve is related to scenario #3 with GLONASS data only. It can be seen that the position errors of GPS and Galileo data are small and stable whereas those from GLONASS data are large and unstable. The blue curve represents for scenario #4 with combination of GPS, Galileo, and GLONASS data. The position accuracy from the combination improves compared to the use of GLONASS data only. However, the errors still remain unsteady and large in the East axis. In particular, the positions are not determined around the times 22 h, 23 h, and 24 h, because of the large HPL (in Figure 4). Consequently, the robust-EKF method is applied to enhance the accuracy of user position in scenario #5 and the obtained position errors are represented in the green curve. They are more stable and more accurate than the other scenarios.

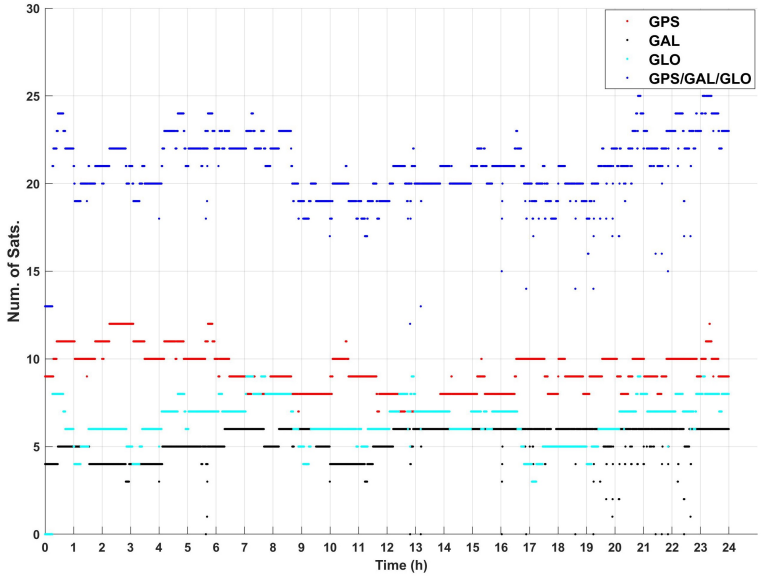

Figure 2. Number of visible satellites.

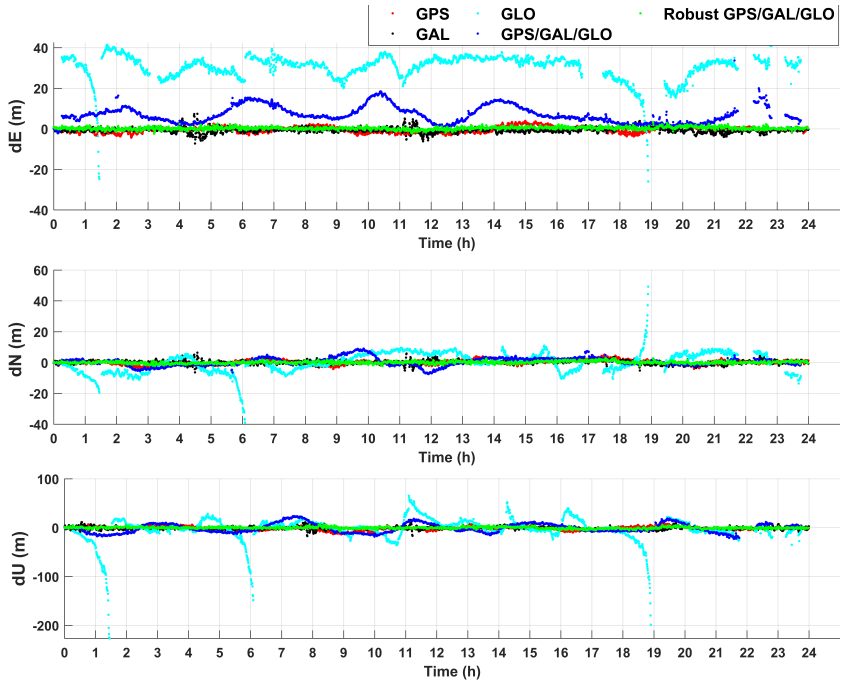

Figure 3. Position errors of the five scenarios at station ABMF.

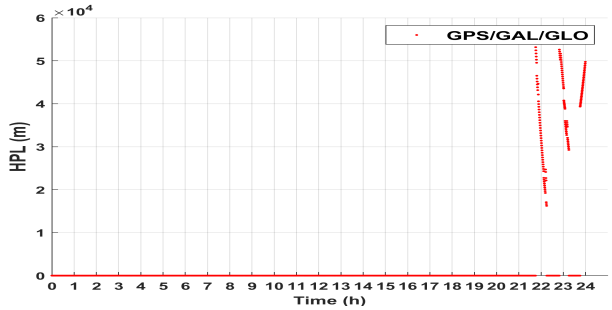

Figure 4. HPLs of combined GPS, Galileo and GLONASS data at station ABMF.

Figure 4 represents the HPLs of combined GPS, Galileo and GLONASS data as a function of time. HPL is represented to check an assured zone of the estimated position. If the HPL is bigger than the HAL, the failure signals are detected and affect the position accuracy. After applying robust-EKF method, HPLs of robust GPS, Galileo and GLONASS data are recalculated in Figure 5 and the large HPLs are reduced.

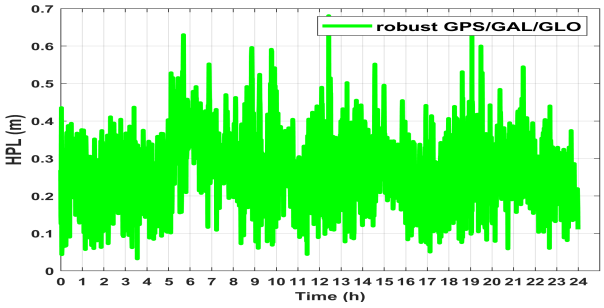

Figure 5. HPLs of robust GPS/GAL/GLO data at station ABMF.

For more insights about the position accuracy obtained for the five scenarios, Table 2 presents a comparison of the corresponding position errors.

Table 2. RMS Errors in Position for the five scenarios at station ABMF in 24 h.

RMS Error (m)	GPS	GAL	GLO	GPS/GAL/GLO	Robust-EKF GPS/GAL/GLO
RMS-E	1.42	1.34	31.00	8.11	0.74
RMS-N	1.46	1.2	6.65	2.74	0.75
RMS-U	3.94	3.08	25.7	9.50	1.82
3D-RMS	4.43	3.56	40.81	12.73	2.1

According to the RMS (Root Mean Square) error values listed in Table 2, the robust combination of GPS, Galileo, and GLONASS data improves the position accuracy by about 53 %, 41%, and 95% compared to the use of GPS data only, Galileo data only, and GLONASS data only, respectively. It is also improved by about 84% compared to the non-robust combination of GPS, Galileo, and GLONASS data.

In the introduction section, it has been stated that the robust-EKF can be better than the Weighted Least-Squares Estimation (WLSE) to improve the position accuracy. To give tangible evidence on that, we compared the two methods and the results are illustrated in Figure 6.

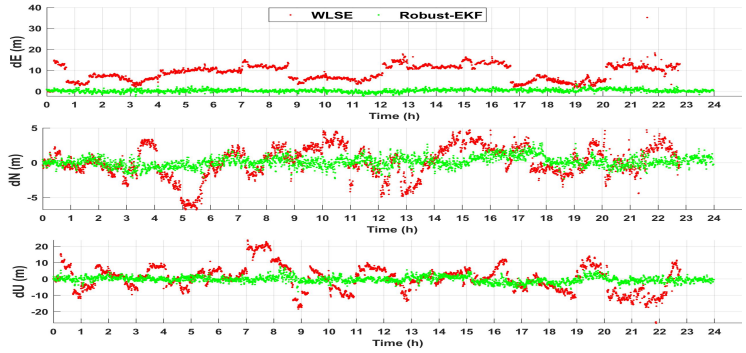

Figure 6. Comparison of position errors between robust-EKF and WLSE.

The Figure 6 presents the position errors obtained from the two methods using the combined GPS, Galileo and GLONASS data. This WLSE method was developed in our previous work in [8], where the weight matrix is defined using the "Exponential model". The position errors of WLSE are wavering; In particular, the positions around 23h-24h are unavailable because of high HPLs. On the other hand, the position errors of robust-EKF are stable and small.

Although the robust-EKF can greatly improve the position accuracy, its outputs might contain errors that cannot be estimated. Therefore, an algorithm that can predetermine the errors of robust-EKF behaviour is needed. For this purpose, a de-noising filter method is presented in Section 5.

5. De-Noising Filter Method and Experimental Results

The recent developments in the “Deep learning” field have allowed the use of its algorithms for GNSS applications [24,25,27,34]. In [34], an intelligent hybrid scheme consisting of an Artificial Neural Network (ANN) and Kalman Filter (KF) has been proposed to improve the accuracies of positional components as well as orientation components in real time. This scheme is able to overcome the limitations of KF. In [25], a “deep learning” algorithm was proposed to denoise the MEMS IMU output signals. The used algorithm is a Long Short Term Memory (LSTM) that was employed to filter the MEMS gyroscope outputs, in which the signals were treated as time series. LSTM is an artificial recurrent neural network (RNN) architecture that is useful for classifying, processing and making predictions based on time series data. Therefore, with a combination of the works presented in [25,34], LSTM is implemented in this paper to learn and compensate for the residual errors of robust-EKF in order to improve the position accuracy. Hence, the rEKF-LSTM hybrid schemes are

proposed as a method capable of learning how the state vector behaves based on the dynamic of the filter. The details of rEKF-LSTM hybrid schemes and the experimental results for GNSS data are detailed in the upcoming Sections 5.1 and 5.2.

5.1. De-Noising Filter Model

Figure 7 presents the LSTM layered architecture. This design demonstrates the flow of a time series X (inputs) with N features of length S through an LSTM layer. The outputs are a time series "h" with H hidden units of length S. In the design, h_k and c_k denote the output (hidden state) and the "cell state" at time step t_k, respectively. The first LSTM block uses the initial state of the network to calculate the first output (h_1) and the "updated cell state" (c_1) at the first time. At the t_k, the block uses the previous state (h_{k-1}, c_{k-1}) to compute the output (h_k) and the "updated cell state" (c_k). A LSTM unit is composed of a cell, an "input gate", an "output gate" and a "forget gate". The basic structure of an LSTM unit is shown in Figure 8 illustrating the flow of data at instant t_k. This diagram highlights the three parts: forget gate, input gate and output gate.

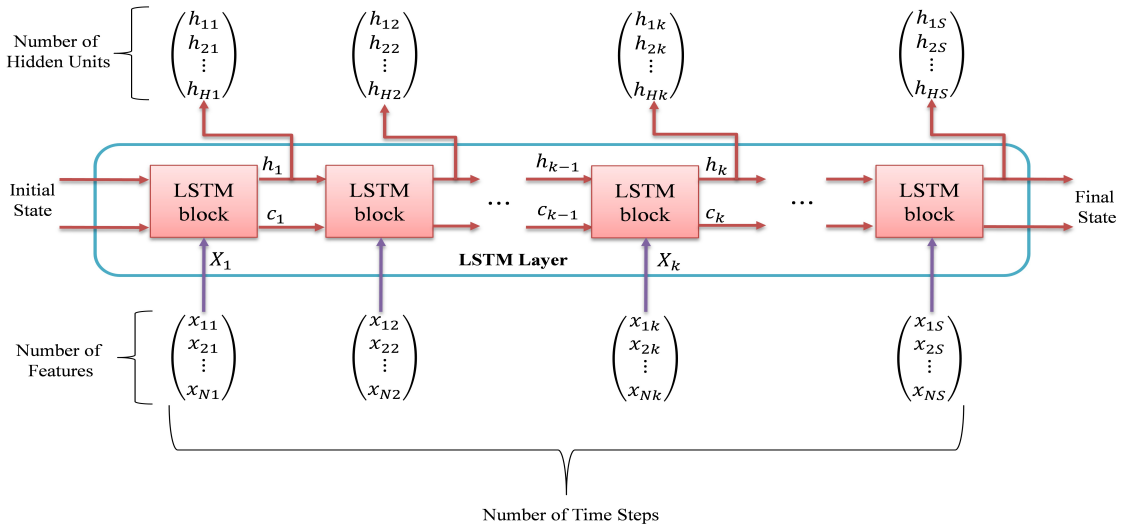

Figure 7. LSTM layered Architecture (source: mathworks/lstm).

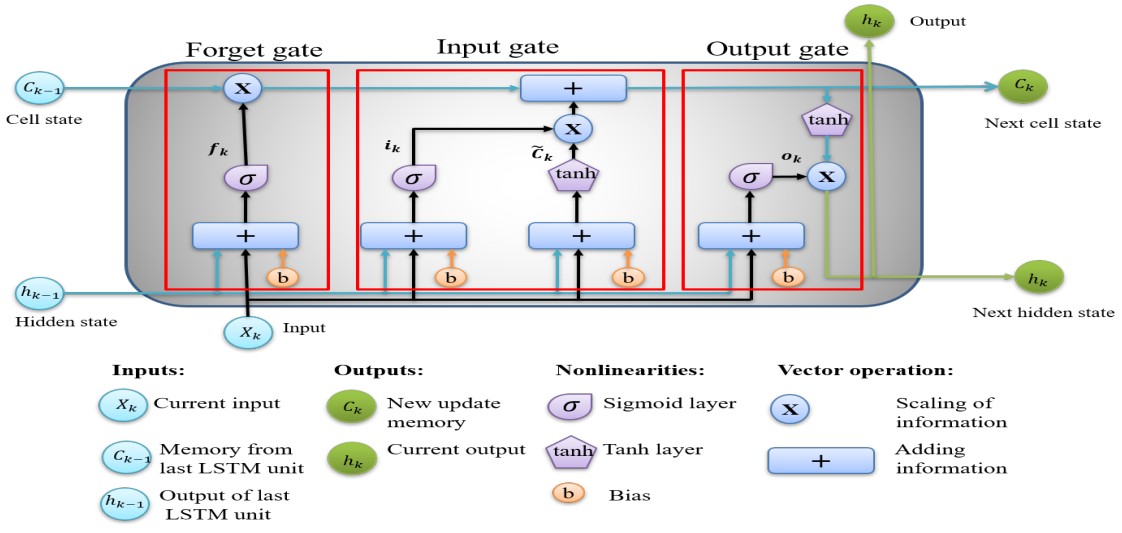

Figure 8. Basic structure of an Long Short Term Memory Unit.

A LSTM layer has the learnable weights and bias: the input weights W_{lstm}, the recurrent weights R_{lstm} and the bias b_{lstm}. The matrices W_{lstm}, R_{lstm}, and b_{lstm} are the union of the input weights, the recurrent weights, and the bias of each part, respectively. These matrices are grouped as follows:

$$\mathrm{W_{lstm}} = \begin{bmatrix} \mathrm{W_f} \\ \mathrm{W_i} \\ \mathrm{W_c} \\ \mathrm{W_o} \end{bmatrix}; \mathrm{R_{lstm}} = \begin{bmatrix} \mathrm{R_f} \\ \mathrm{R_i} \\ \mathrm{R_c} \\ \mathrm{R_o} \end{bmatrix}; \mathrm{b_{lstm}} = \begin{bmatrix} \mathrm{b_f} \\ \mathrm{b_i} \\ \mathrm{b_c} \\ \mathrm{b_o} \end{bmatrix} \tag{76}$$

where f, i, c, and o denote the forget gate, input gate, cell state, and output gate, respectively.

In each LSTM block, the objective is the calculation of the output (hidden state) and the cell state through three parts: forget gate, input gate and output gate. As shown in Figure 8, the "forget gate" is presented in the first part of the LSTM, which is used to decide what information will be kept in the cell state. The decision is made by a sigmoid layer called "forget gate layer". h_{k-1} and x_k are inputs to the sigmoid function, and the output is a value ranging from 0 to 1 for each number in the cell state C_{k-1}. If the output is "1", the information is "completely kept" in the cell state. When the output is "0", the information is "completely cleared". The forget gate's activation vector f_k is written as:

$$f_k = \sigma(W_f x_k + R_f h_{k-1} + b_f) \tag{77}$$

where $\sigma(.)$ is the sigmoid function, h_{k-1} is the hidden state at time t_{k-1}, and x_k is the input vector at time t_k.

The second part is the "input gate", which is employed to decide which new information should enter the previous cell state. This gate is composed of two parts: (1) a "sigmoid" layer to decide what values will be updated. The output values i_k for this layer ranging from 0 to 1. "0" means "not important" and "1" represents "important"; (2) another part is "tanh" layer which creates a vector of new candidate values $\widetilde{C}_k$ between -1 and 1 to help regulate the network. The input gate's activation vector i_k and the new candidate vector $\widetilde{C}_k$ are calculated as follows:

$$i_k = \sigma(W_i x_k + R_i h_{k-1} + b_i) \tag{78}$$

$$\widetilde{C}_k = \tanh\left(W_c x_k + R_c h_{k-1} + b_c\right) \tag{79}$$

where "$tanh$" is the hyperbolic tangent function.

The new cell state C_k is updated from the old cell state C_{k-1} as follows:

$$C_k = f_k * C_{k-1} + i_k * \widetilde{C}_k \tag{80}$$

In the last part, the "output gate" will decide the output. First, a sigmoid layer is used to determine what parts of the cell state will be output. After that, the cell state is passed through a "tanh" function and the ranging of the cell state values is from -1 to 1. Finally, the results are multiplied by the output of the sigmoid gate, and the output parts are decided. The output gate's activation vector and hidden state vector are given by:

$$o_k = \sigma(W_o x_k + R_o h_{k-1} + b_o) \tag{81}$$

$$h_k = o_k * \tanh C_k \tag{82}$$

The above Equations (77)–(82) are the steps to compute the output and the cell state of a LSTM unit. LSTM network can be applied for classification or regression tasks with sequence and time series data. In this paper, we make use of it for the regression task. From Figure 7, we obtain outputs "h" with the dimensions ($H \times S$), while we want have predicted results "Y" with dimesions ($M \times S$); therefore we need a weight matrix with dimension ($M \times H$) to tranforme "h" to "Y". As a result, we have the full architecture of the LSTM network with one LSTM layer for regression as the Figure 9.

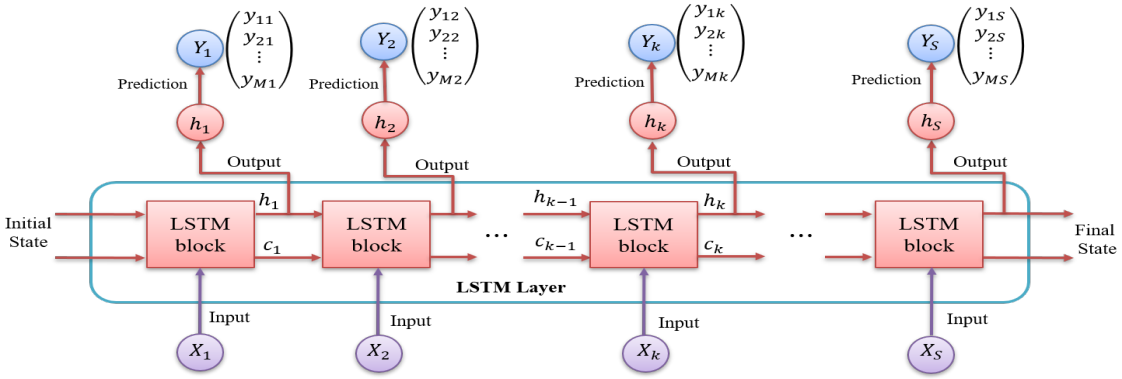

Figure 9. Architecture of LSTM network for regression.

Applying this LSTM network, the inputs data (X) are the positions estimated by robust-EKF in Section 3, and the outputs data (Y) are the errors on the estimated positions.

For training and testing our model, we should divide our data into three distinct sets: training, validation and prediction. For the "training part", Figure 10 presents the rEKF-LSTM training architecture. We determine the estimated position by robust EKF using the combination of GPS, Galileo and GLONASS data. The ENU_{rEKF} is the East, north, and up positions by robust EKF. Before the estimated positions enter the LSTM model, we need to standardize them to have zero mean and unit variance. As a result, the standardized estimated positions are now the training inputs of the LSTM model. The target outputs are the errors of the estimated positions by robust-EKF (δ_{ENU}). The errors in the estimated positions are the difference between the true and the estimated (robust-EFK) positions. Similarly, we standardize the target outputs to have zero mean and unit variance and the standardized target outputs are now the training target outputs of the LSTM model. The model is trained on the training dataset using supervised learning with optimization methods such as: stochastic gradient descent, Adam or RMSProp. Each time, the LSTM model is run with the training inputs and returns the training outputs, which are then compared with the targeted ones. Based on the results of the comparison, the parameters of the model (weights and biases) are adjusted. The training process works until the training errors reach the error threshold and the inferred LSTM model is referred to as the "fitting LSTM model".

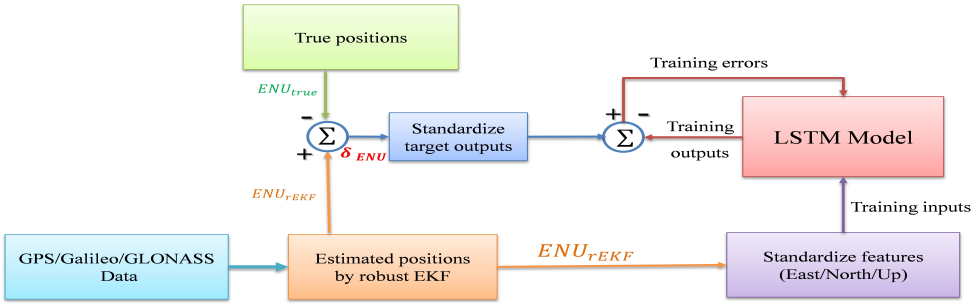

Figure 10. rEKF-LSTM training architecture.

The subsequent part is the "validation part" for which the architecture is illustrated in Figure 11. Firstly, we standardize the estimated east, north, and up positions. After, we use the "fitting LSTM model" from the "training part" to predict the standardized ENU errors. Then, we unstandardize them to obtain the predicted ENU errors. Following that, we compare the predicted ENU errors with the true ENU errors. Based on the comparison, we can see how well the LSTM model is generalizing in the training part. If the model is overfitting or underfitting, it is necessary to go back to the training part to tune the model's hyperparameters (the number of layers, the number of hidden units in each layer etc.). We can return to the training part many times until we get the best LSTM model to ensure confidence on our model performance.

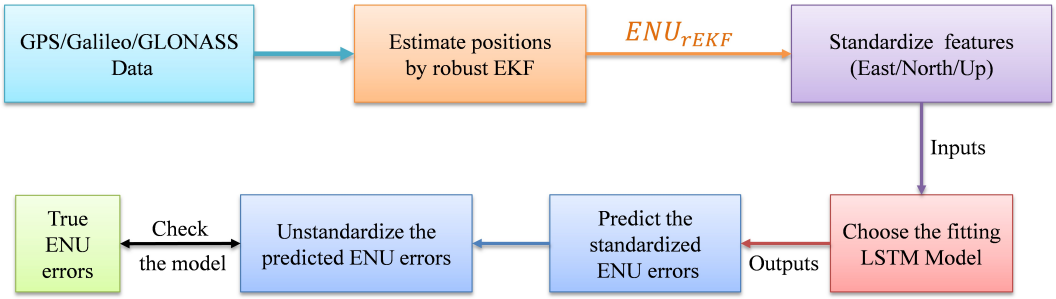

Figure 11. rEKF-LSTM Validation architecture.

After obtaining the best LSTM model, this model is used in the "prediction part". Figure 12 presents the prediction architecture of rEKF-LSTM. First, we standardize the estimated east, north, and up positions similarly to the validation part. After, we use the "best LSTM model" previously validated in the "validation part", to predict the standardization ENU errors. Then, we unstandardize them to obtain the predicted ENU errors. Finally, we obtain the estimated positions by rEKF-LSTM, which are computed as the difference between the estimated positions by robust-EKF and the predicted errors following Equation (83).

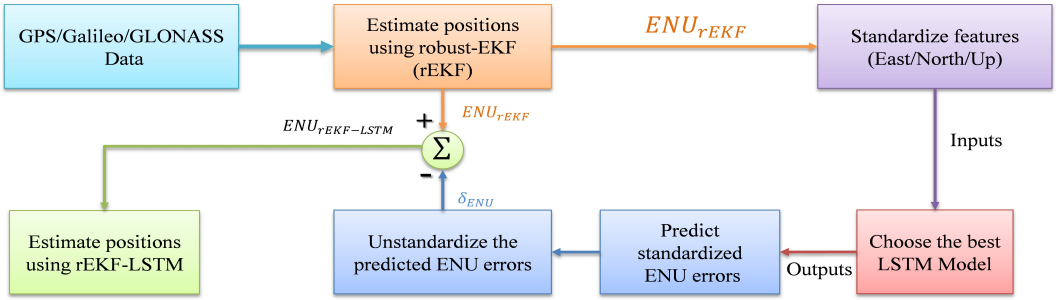

Figure 12. rEKF-LSTM Prediction architecture.

The estimated positions by rEKF-LSTM are written as follows:

$$\mathrm{ENU}_{rEKF-LSTM} = \mathrm{ENU}_{rEKF} - \delta_{ENU} \tag{83}$$

5.2. Experimental Results

In this section we aim at comparing the position estimation performance of the two methods : the robust-EKF and the rEKF-LSTM.

A total of 20 hidden layers were set up for the LSTM in this paper. The LSTM set up to H = 20 hidden layers, N = 3 inputs features (positions), S = 1440 number of timesteps and M = 3 outputs features (errors positions). The data used are still from ABMF station recorded on the 1 January 2019 (as in Section 4.2). Figure 13 represents the position errors with respect to time for the two methods: robust-EKF and rEKF-LSTM. The period of data is from 12 h to 24 h, since data from 0 h to 12 h is used to train the model.

In Figure 13, the green curve represents the position errors for the robust-EKF method, whereas the purple curve represents the errors produced by the rEKF-LSTM method. It can be seen that the position accuracy of rEKF-LSTM method largely improves and remains stable. To have more insights, Table 3 summarizes the RMS position errors of the two methods.

As shown in Table 3, the position accuracy improves by about 74.0 % using rEKF-LSTM compared to the robust EKF. To have a more general view, the five scenarios in Section 4.2 and two scenarios in this section are combined. As a result, six scenarios are considered since the scenario pertaining to robust-EKF is in common. The underlying results are summarized in Table 4.

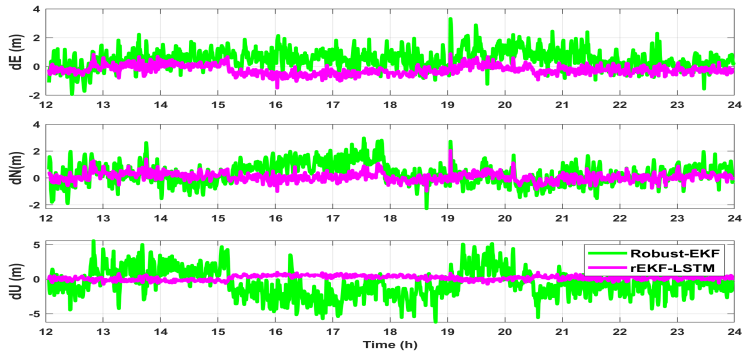

Figure 13. Comparison of position errors between robust-EKF and rEKF-LSTM.

Table 3. RMS Errors in position for the two methods using data from ABMF station in late 12h.

RMS Error (m)	Robust-EKF	rEKF-LSTM
RMS-E	0.82	0.39
RMS-N	0.83	0.32
RMS-U	2.04	0.36
3D-RMS	2.35	0.62

Table 4. RMS Errors in Position for the six scenarios at station ABMF in late 12 h.

RMS Error (m)	GPS	GAL	GLO	GPS/GAL/GLO	Robust-EKF GPS/GAL/GLO	rEKF-LSTM
RMS-E	1.43	1.21	30.71	7.08	0.82	0.39
RMS-N	1.58	1.19	6.18	2.27	0.83	0.32
RMS-U	3.32	3.07	21.10	8.18	2.04	0.36
3D-RMS	3.95	3.50	37.76	11.05	2.35	0.62

As shown in Table 4, the position precision is improved by about 95% using the rEKF-LSTM compared to the non-robust data combination. This result proves that applying LSTM method can greatly improve the position accuracy. In order to assess more thoroughly the rEKF-LSTM method, other base station data are tested and the results are summarized in Table 5.

Table 5. RMS Errors in Position at the three AJAC, GRAC LMMF stations in late 12 h.

Base Station	RMS Error (m)	GPS	GAL	GLO	GPS/GAL/GLO	Robust-EKF GPS/GAL/GLO	rEKF-LSTM
AJAC	**RMS-E**	1.12	1.36	17.76	3.40	0.72	0.73
	RMS-N	1.65	1.25	9.30	4.57	0.88	0.46
	RMS-U	2.52	2.42	3.40	4.75	1.33	0.45
	3D-RMS	3.21	3.04	20.33	7.42	1.75	0.97
GRAC	**RMS-E**	1.14	1.38	17.39	3.42	0.67	0.75
	RMS-N	1.58	1.28	11.87	5.85	0.91	0.54
	RMS-U	2.89	3.02	12.74	4.76	2.29	1.66
	3D-RMS	3.49	3.56	24.6	8.28	2.55	1.89
LMMF	**RMS-E**	1.59	1.07	33.12	6.62	0.68	0.51
	RMS-N	1.36	0.96	11.81	1.82	0.63	0.30
	RMS-U	3.38	2.20	4.32	8.17	1.80	0.47
	3D-RMS	3.96	2.63	35.43	10.67	2.02	0.76

Table 5 gives the RMS errors in position for three other base stations: AJAC, GRAC and LMMF. The position precision is enhanced by about 87%, 77% and 93% using the rEKF-LSTM compared to non-robust data combination from the three base stations AJAC, GRAC and LMMF in France

on 1 January 2019, respectively. This corroborates the conclusion that the rEKF-LSTM method can significantly improve the position precision.

6. Conclusions and Perspectives

In this paper, the SPP technology is applied to determine the user position calculated by GPS, Galileo and GLONASS data fusion. Data fusion may be containing outliers which negatively impact the final results. To address this issue, the MM-Estimation method is used to eliminate the outliers and IRWLS method is used to optimize the estimated results. Although the MM-Estimation and the IRWLS are effective, a method to detect the positioning errors is needed. The RAIM algorithm was used for this purpose to detect positioning errors exceeding the alert limit. As a result, the proposed robust Extended Kalman Filter is built by combining the Extended Kalman Filter, the MM-Estimation, the IRWLS and the RAIM method to improve the position accuracy. Robust GPS, Galileo and GLONASS data fusion using the robust-EKF increases the position precision by about 84.0% compared to non-robust estimation using data of ABMF base station. Furthermore, the position precision is significantly enhanced using the rEKF-LSTM method by about 95% compared to non-robust data combination. To prove the power of rEKF-LSTM, the approach is used for data from three other base stations: AJAC, GRAC and LMMF. The position accuracy is significantly improved by about 87%, 77% and 93% compared to the non-robust data combination, respectively.

In this work, the emphasis is laid on the improvement of the positioning accuracy for a slowly moving receiver. Only data from base stations are used, for which velocities are almost equal to zero and helps enhancing the performance of the LSTM model. A perspective of our work could be assessing the performance and limitations of the method on data from a user in motion and adapting or potentially developing a new method taking into account additional velocity information. While considering data from moving users, sensor noise levels are not constant and the challenge is to estimate an adapted LSTM modeling. Moreover, an additional challenge concerns the GNSS signal blockages. A suggested solution is to combine GNSS and INS (Inertial Navigation System) data. Along these lines, the perspective for this work is to use an adapted rEKF-LSTM model for combining GNSS and INS data from users in motion.

Author Contributions: T.-N.T. proposed the general idea of the method presented and realized its application to real data. A.K., F.C., P.F., J.-M.C. and O.R., suggested the problematic and the field of application, reviewed the idea applied and provided many suggestions. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been performed within the framework of e-PANEMA project. The authors would like to thank the funding organization ADEME.

Acknowledgments: Our warmest thanks go to all the other partners of the e-PANEMA project and also to the people involved in monitoring the project within the ADEME agency. Also our thanks are for anonymous reviewers who have contributed effectively in their comments and suggestions to obtain this version of the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhang, X.; Huang, Z.; Li, R. RAIM analysis in the position domain. In Proceedings of the IEEE/ION Position, Location and Navigation Symposium, Indian Wells, CA, USA, 4–6 May 2010; pp. 53–59, doi:10.1109/PLANS.2010.5507200.
2. Hewitson, S.; Lee, H.K.; Wang, J. Localizability analysis for GPS/Galileo receiver autonomous integrity monitoring. *J. Navig.* **2004**, *57*, 245–259, doi:10.1017/S0373463304002693.
3. Blanch, J.; Ene, A.; Walter, T.; Enge, P. An optimized multiple hypothesis RAIM algorithm for vertical guidance. In Proceedings of the 20th International Technical Meeting of the Satellite Division of the Institute of Navigation, Fort Worth, TX, USA, 25–28 September 2007; Volume 3, pp. 2924–2933.

4. Tay, S.; Marais, J. Weighting models for GPS Pseudorange observations for land transportation in urban canyons. In Proceedings of the 6th European Workshop on GNSS Signals and Signal Processing, Munich, Germany, 5–6 December 2013.
5. Rahemi, N.; Mosavi, M.R.; Abedi, A.A.; Mirzakuchaki, S. Accurate Solution of Navigation Equations in GPS Receivers for Very High Velocities Using Pseudorange Measurements. *Adv. Aerosp. Eng.* **2014**, *2014*, 435891, doi:10.1155/2014/435891.
6. Tabatabaei, A.; Mosavi, M.R.; Khavari, A.; Shahhoseini, H.S. Reliable Urban Canyon Navigation Solution in GPS and GLONASS Integrated Receiver Using Improved Fuzzy Weighted Least-Square Method. *Wirel. Pers. Commun.* **2017**, *94*, 3181–3196, doi:10.1007/s11277-016-3771-1.
7. Li, J.; Wu, M. The improvement of positioning accuracy with weighted least square based on SNR. In Proceedings of the 5th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM), Beijing, China, 24–26 September 2009; pp. 9–12, doi:10.1109/WICOM.2009.5302600.
8. Tan, T.N.; Khenchaf, A.; Comblet, F. Estimating Ambiguity and Using Model Weight To Improve the Positioning Accuracy of a Stand-alone Receiver. In Proceedings of the 13th European Conference on Antennas Propagation (EuCAP 2019), Krakow, Poland, 31 March–5 April 2019; Volume 1.
9. Tan, T.N.; Khenchaf, A.; Comblet, F.; Franck, P.; Champeyroux, J.M.; Reichert, O. GPS/GLONASS Data Fusion and Outlier Elimination to Improve the Position Accuracy of Receiver. In Proceedings of the 2019 IEEE Conference on Antenna Measurements & Applications (CAMA), Bali, Indonesia, 23–25 October 2019; Volume 805, pp. 191–194, doi:10.1109/CAMA47423.2019.8959694.
10. Wang, Y.; Jie, H.; Cheng, L. A Fusion Localization Method based on a Robust Extended Kalman Filter and Track-Quality for Wireless Sensor Networks. *Sensors (Basel)* **2019**, *19*, 3638, doi:10.3390/s19173638.
11. Fang, H.; Haile, M.A.; Wang, Y. Robust Extended Kalman Filtering for Systems with Measurement Outliers. *arXiv* **2019**, arXiv:1904.00335.
12. Perälä, T.; Piché, R. Robust extended Kalman filtering in hybrid positioning applications. In Proceedings of the 4th Workshop on Positioning, Navigation and Communication, WPNC'07—Workshop Proceedings, Hannover, Germany, 22 March 2007; pp. 55–63, doi:10.1109/wpnc.2007.353613.
13. Gao, Z.; Li, Y.; Zhuang, Y.; Yang, H.; Pan, Y.; Zhang, H. Robust kalman filter aided GEO/IGSO/GPS raw-PPP/INS tight integration. *Sensors (Switzerland)* **2019**, *19*, 417, doi:10.3390/s19020417.
14. Wang, J.; Xu, C.; Wang, J. Applications of Robust Kalman Filtering Schemes in GNSS Navigation. In Proceedings of the International Symposium on GPS/GNSS, Tokyo, Japan, 11–14 Novembre 2008.
15. Gaglione, S.; Angrisano, A.; Crocetto, N. Robust Kalman Filter applied to GNSS positioning in harsh environment. In Proceedings of the European Navigation Conference (ENC 2019), Warsaw, Poland, 9–12 April 2019; pp. 1–6, doi:10.1109/EURONAV.2019.8714132.
16. Kumar, P. Robust Kalman Filter Using Robust Cost Function. Master's Thesis, Deparment of Electronics and Communication Engineering, National Institute of Technology, Rourkela, Indian, 2015.
17. Gandhi, M.A.; Mili, L. Robust Kalman Filter based on a generalized maximum-likelihood-type estimator. *IEEE Trans. Signal Process.* **2010**, *58*, 2509–2520, doi:10.1109/TSP.2009.2039731.
18. Zhao, J.; Netto, M.; Mili, L. A Robust Iterated Extended Kalman Filter for Power System Dynamic State Estimation. *IEEE Trans. Power Syst.* **2017**, *32*, 3205–3216, doi:10.1109/TPWRS.2016.2628344.
19. Yuliana, S.; Hasih, P.; Sri Sulistijowati, H.; Twenty, L. M Estimation, S Estimation, and MM Estimation in Robust Regression. *Int. J. Pure Appl. Math.* **2014**, *91*, 349–360.
20. Yohai, Y.J. High Breakdown-point and High Efficiency Robust Estimates for Regression. *Ann. Stat.* **1986**, *14*, 590–606.
21. Huber, P.J. *Robust Statistics*; John Wiley and Sons, Inc.: Hoboken, NJ, USA, 2009.
22. Akram, M.A.; Liu, P.; Wang, Y.; Qian, J. GNSS positioning accuracy enhancement based on robust statistical MM estimation theory for ground vehicles in challenging environments. *Appl. Sci. (Switzerland)* **2018**, *8*, 876, doi:10.3390/app8060876.
23. Strang, G.; Borre, S. *Algorithms for Global Positioning*; Wellesley-Cambridge Press: Wellesley, MA, USA, 2012.
24. Fang, W.; Jiang, J.; Lu, S.; Gong, Y.; Tao, Y.; Tang, Y.; Yan, P.; Luo, H.; Liu, J. A LSTM algorithm estimating pseudo measurements for aiding INS during GNSS signal outages. *Remote Sens.* **2020**, *12*, 256, doi:10.3390/rs12020256.

25. Jiang, C.; Chen, S.; Chen, Y.; Zhang, B.; Feng, Z.; Zhou, H.; Bo, Y. A MEMS IMU de-noising method using long short term memory recurrent neural networks (LSTM-RNN). *Sensors (Switzerland)* **2018**, *18*, 3470, doi:10.3390/s18103470.
26. Li, X.; Wu, X. Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition. In Proceedings of the ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing, Brisbane, Australia, 19–24 April 2015; pp. 4520–4524, doi:10.1109/ICASSP.2015.7178826.
27. Kim, H.U.; Bae, T.S. Deep learning-based GNSS network-based real-time kinematic improvement for autonomous ground vehicle navigation. *J. Sens.* **2019**, *2019*, 3737265, doi:10.1155/2019/3737265.
28. Kim, H.U.; Bae, T.S. Preliminary study of deep learning-based precipitation prediction. *J. Korean Soc. Surv. Geod. Photogramm. Cartogr.* **2017**, *35*, 423–429, doi:10.7848/ksgpc.2017.35.5.423.
29. Nicolini, L.; Caporali, A. Investigation on reference frames and time systems in Multi-GNSS. *Remote Sens.* **2018**, *10*, 80, doi:10.3390/rs10010080.
30. Gurtner, W.; Estey, L. RINEX The Receiver Independent Exchange Format Version 3.03 . In Proceedings of the IGS Workshop 2012, Olsztyn, Poland, 23–27 July 2012; Volume 104, pp. 1–74.
31. Noureldin, A.; Karamat, T.B.; Georgy, J. *Fundamentals of Inertial Navigation, Satellite-Based Positoning and Their Integration*; Springer: Berlin/Heidelberg, Germany, 2013.
32. Hofmann-Wellenhof, B.; Lichtenegger, H.; Wasle, E. *GNSS—Global Navigation Satellite Systems: GPS, GLONASS, Galileo and More*; Springer: Wien, Austria, 2008.
33. Ellis, D.M.; Draper, N.P.; Smith, H. Applied Regression Analysis. *Appl. Stat.* **1968**, *17*, 83, doi:10.2307/2985274.
34. Chiang, K.W.; Chang, H.W.; Li, C.Y.; Huang, Y.W. An Artificial Neural Network Embedded Position and Orientation Determination Algorithm for Low Cost MEMS INS/GPS Integrated Sensors. *Sensors* **2009**, *9*, 2586–2610, doi:10.3390/s90402586.