pyrtklib: An Open-Source Package for Tightly Coupled Deep Learning and GNSS Integration for Positioning in Urban Canyons

Runzhi Hu, Penghui Xu, *Graduate Student Member, IEEE*, Yihan Zhong, and Weisong Wen, *Member, IEEE*

***Abstract*— Global Navigation Satellite Systems (GNSS) are crucial for intelligent transportation systems (ITS), providing essential positioning capabilities globally. However, in urban canyons, the GNSS performance could significantly degraded due to the blockage of direct GNSS signals. The pseudorange measurements are largely affected and the conventional model of weighting observations is not suitable in urban canyons. This paper addresses these challenges by integrating Artificial Intelligence (AI), specifically deep learning, into GNSS positioning process to enhance positioning accuracy. Traditional methods have primarily focused on pseudorange correction due to the absence of ground truth for weight estimation. In response, we propose an innovative indirect training approach using deep learning to optimize both pseudorange bias and weight estimation, aiming to minimize the positioning errors. To support this integration, we developed pyrtklib, a Python binding for the open-source RTKLIB tool, bridging the gap between traditional GNSS algorithms, typically developed in Fortran or C, and modern Python-based AI frameworks. Comparative analyses demonstrate that our method surpasses established tools like goGPS and RTKLIB in positioning accuracy, marking a significant advancement in the field. The source code of tightly coupled deep learning and GNSS integration, along with pyrtklib, is available on GitHub at https://github.com/ebhrz/TDL-GNSS and https://github.com/IPNL-POLYU/pyrtklib.**

***Index Terms*— Artificial intelligence, deep learning, GNSS, RTKLIB.**

I. Introduction

WITH the rapid growth of computing speed and power, artificial intelligence (AI), epitomized by deep learning (DL), is now practical for everyday use. Due to its excellent non-linear fitting capability, deep learning has proven effective in numerous fields, including computer vision (CV) and natural language process (NLP). In modern intelligent transportation systems (ITS), deep learning also demonstrates potential in areas such as traffic control [1], autonomous driving (AD) [2], and human behavior analyze [3]. Global positioning, commonly known as global navigation satellite system (GNSS) positioning, is crucial for perception and decision-making within ITS [4], [5], [6]. Consequently, there is a pressing need within the ITS community to integrate deep learning techniques into global positioning strategies.

Currently, GNSS positioning accuracy can achieve centimeter-level precision under open skies. However, in urban canyons, the performance dramatically declines as GNSS signals are diffracted, reflected, and even obstructed by high-rise buildings [4], [7]. To mitigate the adverse effects of non-line-of-sight (NLOS) and multipath interference, two strategies are implemented: directly correcting measurements to improve accuracy and downweighing measurements from low-quality signals in the weighted least squares (WLS) solution process [8]. Traditional methods use physical models and empirical formulas to model biases and weights, providing high interpretability but often struggling in complex and dynamically changing environments. In contrast, data-driven deep learning approaches can potentially model biases and weights more effectively, provided that the training data is of high quality.

Previous approaches in deep learning-aided GNSS have primarily focused on correcting pseudorange bias, with few researchers exploring weight estimation. Unlike pseudorange bias, the ground truth for weight cannot be directly derived from ground truth positions. From a deep learning perspective, one solution is to treat both bias and weight as parameters and to propagate errors back from the positioning outcomes. However, this method requires a deep integration of deep learning with GNSS, highlighting a significant gap between these two technologies.

Deep learning frameworks and applications have overwhelmingly adopted Python due to its simplicity, flexibility, and the vast ecosystem of libraries and tools available, such as TensorFlow [9] and PyTorch [10]. This accessibility and ease of use facilitate rapid prototyping and deployment of complex models, making Python the language of choice for most new developments in deep learning and AI. Meanwhile, GNSS software historically leans on programming languages like Fortran and C [11], [12], [13]. Though these languages offer high performance and control over hardware interaction, which are critical for the real-time processing demands and precision required in satellite navigation, these languages are not equipped with good compatibility with Python.

Received 27 August 2024; revised 20 January 2025; accepted 5 March 2025. This work was supported in part by the Research Centre for Data Sciences and Artificial Intelligence (RCDSAI) and in part by the Meituan Academy of Robotics Shenzhen (H-ZGHQ). The Associate Editor for this article was Z. Xiao. *(Corresponding author: Weisong Wen.)*

The authors are with the Department of Aeronautical and Aviation Engineering, The Hong Kong Polytechnic University, Hong Kong (e-mail: welson.wen@polyu.edu.hk).

Digital Object Identifier 10.1109/TITS.2025.3552691

This divergence in technological stacks presents a significant challenge for integrating cutting-edge AI methodologies, like deep learning, directly into traditional GNSS software systems. Bridging this gap requires either extending these systems to interface with Python-based tools or developing new capabilities within the GNSS software to support advanced machine learning techniques directly in C or Fortran, both of which entail substantial development and potential refactoring of existing codebases.

To fill this gap, we make the Python binding, named **pyrtklib**, for the most popular open-source GNSS library, RTKLIB [12]. **pyrtklib** provides access to the full functionalities of RTKLIB, combining the speed of C with the convenience of Python. Based on this, we introduce a tightly coupled deep learning approach for weight estimation and bias correction in single point positioning, which has been validated on our datasets. The following contributions of this paper are presented:

1) This paper developes a network to predict pseudorange biases and integrates these predictions into the correction of GNSS pseudorange measurements, tightly coupling them within the least squares process.
2) This paper has designed a network specifically to predict weights for each measurement, which are then utilized in the weighted least squares process.
3) This paper presents a network that simultaneously predicts both weights and biases for each measurement, applying these predictions in the weighted least squares process to improve accuracy.
4) This paper has open-sourced a Python package named **pyrtklib**, a Python binding for RTKLIB. Using meta-programming techniques, we automatically translate RTKLIB's header files into Python binding code via pycparser [14] and pybind11 [15], maintaining the integrity of RTKLIB's constants and functions. The package can be accessed at https://github.com/IPNL-POLYU/pyrtklib.
5) Building on **pyrtklib**, this paper proposes a sub-system that integrates deep learning into the GNSS positioning process. This innovative framework is engineered for training and predicting weights and biases within the least squares solving process, available at https://github.com/ebhrz/TDL-GNSS.

II. Related Works

A. GNSS Tools

There are several existing GNSS tools as shown in Table I. Bernese GNSS [11] and MuSNAT [13] are two popular commercial GNSS software, however, their closed-source nature limits their usability for algorithm development. NavSU, MAAST, and goGPS [16] are open-source software programs written in Matlab. Despite their open-source status, a paid Matlab license is still required for their use. Laika and gnss_lib_py both are Python libraries, but they only provide basic GNSS functions and lack the support for real-time kinematic (RTK) and precise point positioning (PPP). RTKLIB is a comprehensive open-source GNSS tool that enjoys widespread use not only within the GNSS community but also in the robotics sector [17], [18], [19], [20]. However, its C-based architecture poses integration challenges with Python. Additionally, while the above tools utilize empirical formulas to predict variance and control weights, they lack the capability to correct pseudorange biases. To address this need, we developed **pyrtklib**. This library, written in C++ and integrated into Python, combines the efficiency of C with the ease of use of Python.

B. Deep Learning in GNSS

In our recent review paper [21], we categorize the application of deep learning in GNSS as follows:

1) **Improved pseudorange measurement**
2) **Measurement status prediction**
3) **Positioning level information**
4) **Measurement error prediction**

The approach outlined in 1) aims to enhance the correlator and discriminator at the receiver level to improve signal quality control [22], [23], [24]. These methods are highly integrated, forming a super tightly coupled relationship between deep learning and GNSS. However, upgrading the receiver hardware is challenging to scale rapidly. The objective of 2) is to utilize deep learning to classify NLOS and multipath signals in advance, thus eliminating or mitigating their adverse effects [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38]. The goal of 3) is to predict the final positioning error and apply corrections [39], [40], [41], [42], [43], [44]. These strategies represent preprocess and postprocess applications that are loosely coupled with deep learning in GNSS. Lastly, the algorithms discussed in 4) have a direct impact on the measurements and are considered tightly coupled [31], [37], [45], [46].

Super tightly coupled methods require support from either hardware or software-based receivers. In contrast, the labels for loosely coupled methods are easily accessible when the ground truth position is known, allowing the training process to be decoupled from the positioning process. However, for tightly coupled methods, measurement correction is intricately linked to positioning, making it impractical to separate training and positioning processes. In such scenarios, it is advantageous for both positioning and training to be conducted in the same programming language. To this end, our framework built on **pyrtklib** is specifically designed to support tightly coupled deep learning and GNSS approaches. We anticipate that our framework will facilitate the development of tightly coupled algorithms.

III. Tightly Coupled Deep Learning Framework For GNSS

A. GNSS Position Principle

A standard GNSS model can be formulated as:

$$p = r + c\Delta t + I + T + \epsilon \quad (1)$$

Here, p represents the pseudorange measured by the receiver, c is the speed of light, and I and T signify the ionospheric and tropospheric delays, respectively. These delays are

TABLE I
COMPARISON OF EXISTING GNSS TOOLS

Tool	Language	Open-source	Weight Prediction	Bias Correction	Weight+Bias Prediction
RTKLIB	C	Y	Y	N	N
Bernese GNSS	Fortran	N	Y	N	N
MuSNAT	Matlab/C++	N	Y	N	N
NavSU	Matlab	Y	Y	N	N
MAAST	Matlab	Y	Y	N	N
goGPS	Matlab	Y	Y	N	N
Laika	Python	Y	Y	N	N
gnss_lib_py	Python	Y	Y	N	N
pyrtklib	Python	Y	Y	Y	Y

typically estimated using atmospheric models in single point positioning. Δt denotes the receiver's time bias and ϵ represents Gaussian noise. The variable r denotes the distance between the satellite and the receiver, which is formulated as follows:

$$r = \sqrt{(x^{(s)} - x_r)^2 + (y^{(s)} - y_r)^2 + (z^{(s)} - z_r)^2} \tag{2}$$

where $x^{(s)}$, $y^{(s)}$, $z^{(s)}$ and x_r, y_r, z_r are the coordinates of the satellite and the receiver in Earth-centered, Earth-fixed (ECEF) coordinate system respectively. The superscript (s) represents the index of the satellite. When there are n pseudorange measurements, the observation function is:

$$\widetilde{\mathbf{Z}} = h(\mathbf{y}) = \begin{bmatrix} r_1 + c\Delta t + I_1 + T_1 \\ r_2 + c\Delta t + I_2 + T_2 \\ \cdots \\ r_n + c\Delta t + I_n + T_n \end{bmatrix} \tag{3}$$

where $\widetilde{\mathbf{Z}} = [p_1, p_2 \ldots, p_n]^T$ and $\mathbf{y} = (x_r, y_r, z_r, \Delta t)$, which are the measurements and state respectively. The first-order approximation of the function can be written as:

$$h(\mathbf{y} + \Delta\mathbf{y}) \approx h(\mathbf{y}) + \mathbf{H}\Delta\mathbf{y} \tag{4}$$

$\mathbf{H}$ is the Jacobian matrix:

$$\mathbf{H} = \begin{bmatrix} \frac{x^{(1)} - x_1}{r_1} & \frac{y^{(1)} - y_1}{r_1} & \frac{z^{(1)} - z_1}{r_1} & c \\ \frac{x^{(2)} - x_1}{r_2} & \frac{y^{(2)} - y_1}{r_2} & \frac{z^{(2)} - z_1}{r_2} & c \\ \cdots & \cdots & \cdots & \cdots \\ \frac{x^{(n)} - x_1}{r_n} & \frac{y^{(n)} - y_1}{r_n} & \frac{z^{(n)} - z_1}{r_n} & c \end{bmatrix} \tag{5}$$

The Gussian-Newton-based non-linear weighted least squares (WLS) is employed to solve the unknown state $\mathbf{y}$ by iteration as follows:

$$\Delta\mathbf{y} = \left(\mathbf{H}^\mathbf{T}\mathbf{W}\mathbf{H}\right)^{-1} \mathbf{H}^\mathbf{T}\mathbf{W}\left(\widetilde{\mathbf{Z}} - h(\mathbf{y}_i)\right) \tag{6}$$

$$\mathbf{y}_{i+1} = \mathbf{y}_i + \Delta\mathbf{y} \tag{7}$$

where $\mathbf{W}$ represents the weight matrix, which is a diagonal matrix determined by the variance of the pseudorange measurement error. For details on the computation of the weight matrix in conventional methods, see Equations (32) and (34). $\mathbf{y}_0$ denotes the initial guess of the state, typically set to set $(0, 0, 0, 0)$. The iteration process is halted once $||\Delta y||$ falls below a predefined threshold, at which point the final $\mathbf{y}$ represents the determined position.

Note that the WLS positioning process requires two inputs, $\mathbf{W}$ and $\widetilde{\mathbf{Z}}$, We simplify the expression using the following equation:

$$\mathbf{y} = f_{WLS}(\mathbf{W}, \widetilde{\mathbf{Z}}) \tag{8}$$

Although the description above represents an ideal scenario, various factors such as NLOS and multipath effects, imprecise ephemerides, or receiver errors can introduce biases. Consequently, the model can be reformulated as follows:

$$p = r + c\Delta t + I + T + b + \epsilon \tag{9}$$

where b is the unmodeled bias. To achieve more accurate positioning results despite these biases, two strategies are employed. The first strategy involves directly correcting the pseudorange measurements, while the second strategy entails down-weighting the unhealthy measurements. In the following subsection, we will introduce a tightly coupled deep learning and GNSS framework designed for bias correction and weight prediction.

B. Tightly Coupled Deep Learning/GNSS Framework

As illustrated in equation (8), achieving optimal positioning results relies on accurately predicting the weights $\mathbf{W}$ or obtaining improved measurements $\widetilde{\mathbf{Z}}$. By utilizing the ground truth position and employing the mean square error (MSE) as the loss function, we can optimize both the measurements and weights using the following equations:

$$L(\mathbf{y}_{gt}, \mathbf{y}) = \frac{1}{2}(\mathbf{y}_{gt} - \mathbf{y})^2 \tag{10}$$

$$\frac{\partial L}{\partial \widetilde{\mathbf{Z}}} = \frac{\partial L}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \widetilde{\mathbf{Z}}} \tag{11}$$

$$\frac{\partial L}{\partial \mathbf{W}} = \frac{\partial L}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{W}} \tag{12}$$

Equation (10) defines the loss function. Equations (11) and (12) demonstrate the gradients of $\widetilde{\mathbf{Z}}$ and $\mathbf{W}$, respectively, calculated using the chain rule under the specified conditions of the loss function. Should $\mathbf{W}$ and $\widetilde{\mathbf{Z}}$ be derived from a neural network, these gradients are then propagated backward through the backpropagation process.

In this demonstration, three key features are selected as inputs:

- **Carrier-to-noise density (C/N0)**: C/N0 is an essential parameter that quantifies the quality of the received signal. It is defined as the ratio of the carrier power to the noise power per unit bandwidth and is typically expressed in decibels-hertz (dB-Hz).

- **Elevation Angle**: The elevation angle is the vertical angle measured from the receiver's horizon to the line of sight of a satellite. This measurement indicates the satellite's position relative to the receiver's location on Earth. In urban environments, satellites with higher elevation angles are often line-of-sight (LOS) satellites and are less likely to be obstructed.
- **Residuals from Ordinary Least Squares Solution**: Initially, the position is calculated using an ordinary least squares (OLS) solution. The residuals from each measurement are then analyzed. This feature aids in identifying potentially problematic or unhealthy measurements, thereby enhancing the reliability of the positioning data.

These three features are compiled into a vector $\mathbf{x} = [C/N0, Elevation, Residual]$ for the network input.

1) Pseudorange Bias Correction Network: The detailed structure of the bias network is depicted in Figure 1a. This network comprises a straightforward four-layer architecture, including:

- An input layer, which has a configuration of 3×1, corresponding to the three input features.
- Two hidden layers, sized 64×1 and 128×1 respectively, designed to progressively refine the feature representations.
- An output layer, configured as 1×1, which outputs the predicted bias.

The rectified linear unit (ReLU) is employed as the activation function throughout the network to introduce non-linearity, enhancing the model's capability to learn complex patterns. The final output represents the desired bias, which is used to predict the pseudorange bias as follows:

$$\mathbf{b} = f_{nn,b}(\mathbf{X}; \Theta) \tag{13}$$

where $\mathbf{X}$ represents the batch input, defined as $\mathbf{X} = [\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}]$ and $\mathbf{b} = [b_1, b_2, \ldots, b_n]$ denotes the corresponding batch output. Θ symbolizes the set of parameters within the neural network. Using these definitions, the corrected measurements and resultant positions can be expressed as follows:

$$\hat{\tilde{\mathbf{Z}}} = \tilde{\mathbf{Z}} - \mathbf{b} \tag{14}$$

$$\hat{\mathbf{y}} = f_{WLS}(\mathbf{W}, \hat{\tilde{\mathbf{Z}}}) = f_{WLS}(\mathbf{W}, \tilde{\mathbf{Z}} - f_{nn,b}(\mathbf{X}; \Theta)) \tag{15}$$

The training process can be formulated as follows:

$$\Theta = \underset{\Theta}{\operatorname{argmin}} \sum_{i=1}^{n} L(\mathbf{y}_{gt}, f_{WLS}(\mathbf{W}, \tilde{\mathbf{Z}} - f_{nn,b}(\mathbf{X}_i; \Theta))) \tag{16}$$

$$\frac{\partial L}{\partial \Theta} = \frac{\partial L}{\partial \hat{\mathbf{y}}} \frac{\partial \hat{\mathbf{y}}}{\partial \hat{\tilde{\mathbf{Z}}}} \frac{\partial \hat{\tilde{\mathbf{Z}}}}{\partial \mathbf{b}} \frac{\partial \mathbf{b}}{\partial \Theta} \tag{17}$$

$$\Theta_{n+1} = \Theta_n - \eta \frac{\partial L}{\partial \Theta} \tag{18}$$

Equation (16) illustrates that the training objective is to identify the optimal network parameters, $\Delta\Theta$, that minimize the loss function. Equation (17) details the gradient transfer process, with η representing the learning rate.

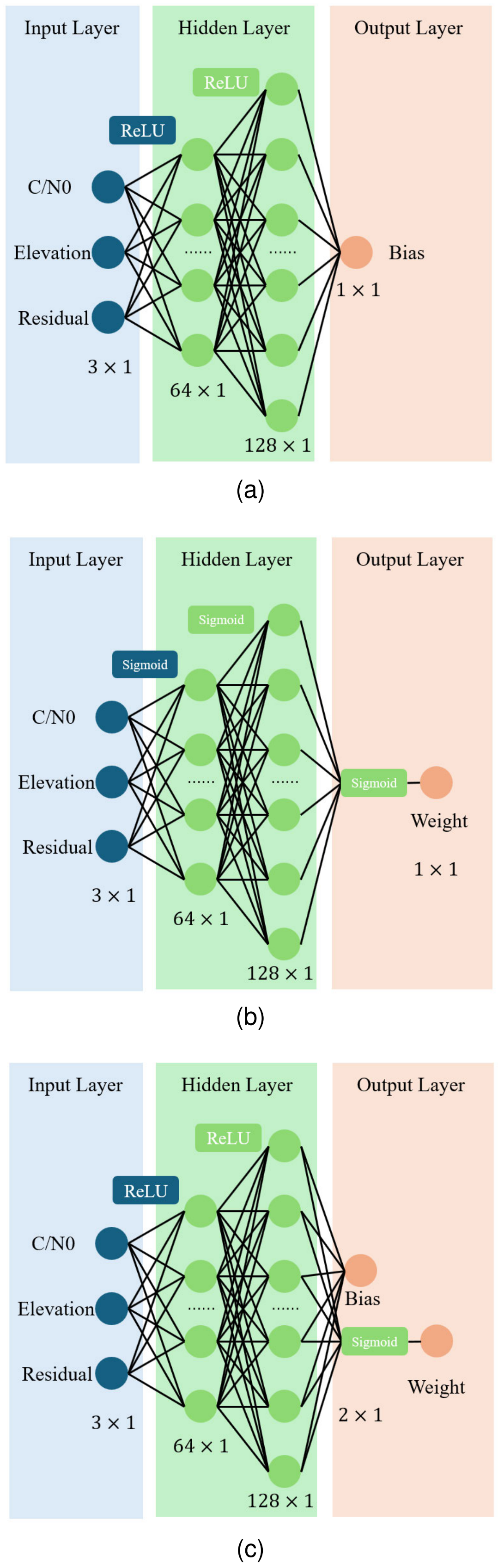


Fig. 1. The detailed structure of the bias network and weight network.

2) Weights Prediction Network: The detailed structure of the weight network is depicted in Figure 1b. While similar to the bias network, this network employs a sigmoid activation function instead of ReLU. The network is designed to predict the weights as follows:

$$\hat{\mathbf{w}} = f_{nn,w}(\mathbf{X}; \Theta) \tag{19}$$

$$\hat{\mathbf{W}} = diag(\hat{\mathbf{w}}) \tag{20}$$

where $\hat{\mathbf{w}} = [w_1, w_2, \ldots, w_n]$ represents the vector of predicted weights in the batch output. $\hat{\mathbf{W}}$ is a diagonal matrix composed of the elements from $\hat{\mathbf{w}}$. The position process is then formulated as follows:

$$\hat{\mathbf{y}} = f_{WLS}(\hat{\mathbf{W}}, \tilde{\mathbf{Z}}) = f_{WLS}(f_{nn,w}(\mathbf{X}; \Theta), \tilde{\mathbf{Z}}) \tag{21}$$

The training process is formulated as follows:

$$\Theta = \underset{\Theta}{\operatorname{argmin}} \sum_{i=1}^{n} L(\mathbf{y}_{gt}, f_{WLS}(f_{nn,w}(\mathbf{X}_i; \Theta), \tilde{\mathbf{Z}})) \tag{22}$$

$$\frac{\partial L}{\partial \Theta} = \frac{\partial L}{\partial \hat{\mathbf{y}}} \frac{\partial \hat{\mathbf{y}}}{\partial \hat{\mathbf{W}}} \frac{\hat{\mathbf{W}}}{\partial \Theta} \tag{23}$$

$$\Theta_{n+1} = \Theta_n - \eta \frac{\partial L}{\partial \Theta} \tag{24}$$

Equation (23) demonstrates the gradient transfer process within the tightly coupled deep learning and GNSS framework, tracing the path from the position loss function to the network parameters through the weights.

3) Bias Correction and Weights Prediction Network: The previously described frameworks focus exclusively on either bias or weight prediction, yet it is possible to predict both simultaneously. The architecture of this dual-prediction network is illustrated in Figure 1c. In this network, ReLU serves as the activation function for the initial two layers. The output layer features two outputs: one for bias and another for weight. To ensure that the weight values range from zero to one, a sigmoid function is applied specifically to the weight output. This design allows the bias and weight predictions to share network parameters, effectively extracting information from the input. The predicted bias and weight are denoted as follows:

$$(\mathbf{b}, \hat{\mathbf{W}}) = f_{nn,bw}(\mathbf{X}; \Theta) \tag{25}$$

$$\mathbf{b} = f^{b}_{nn,bw}(\mathbf{X}; \Theta) \tag{26}$$

$$\hat{\mathbf{W}} = f^{w}_{nn,bw}(\mathbf{X}; \Theta) \tag{27}$$

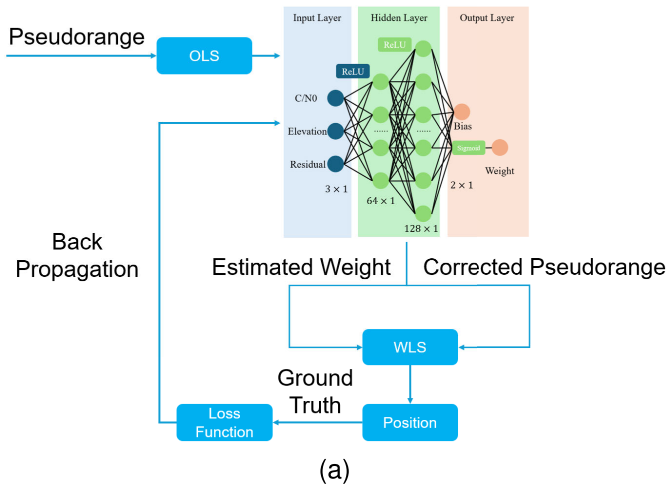
The superscript b and w represent the bias output and the weight output, respectively. With these outputs defined, the positioning process can be formulated as follows:

$$\begin{aligned}\hat{\mathbf{y}} &= f_{WLS}(\hat{\mathbf{W}}, \tilde{\mathbf{Z}} - \mathbf{b}) \\ &= f_{WLS}(f^{w}_{nn,bw}(\mathbf{X}; \Theta), \tilde{\mathbf{Z}} - f^{b}_{nn,bw}(\mathbf{X}; \Theta))\end{aligned} \tag{28}$$

The training process is delineated as follows:

$$\Theta = \underset{\Theta}{\operatorname{argmin}} \sum_{i=1}^{n} L(\mathbf{y}_{gt}, f_{WLS}(\hat{\mathbf{W}}, \tilde{\mathbf{Z}} - \mathbf{b})) \tag{29}$$

$$\frac{\partial L}{\partial \Theta} = \frac{\partial L}{\partial \mathbf{b}} \frac{\partial b}{\partial \Theta} + \frac{\partial L}{\partial \mathbf{W}} \frac{\partial W}{\partial \Theta} \tag{30}$$



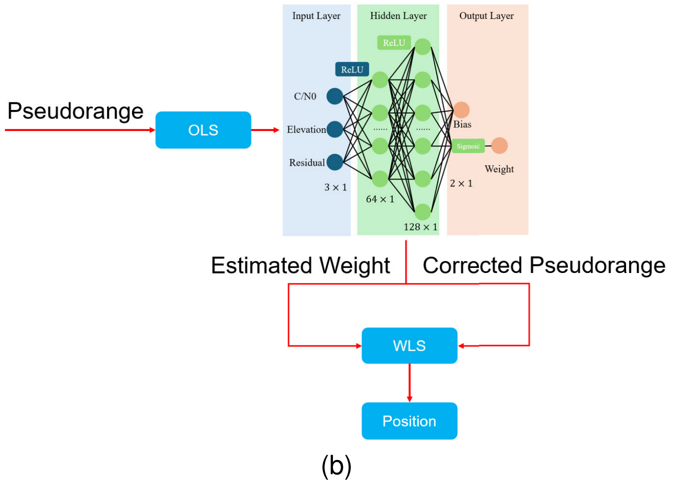


Fig. 2. The overview of our training and prediction process.

$$\Theta_{n+1} = \Theta_n - \eta \frac{\partial L}{\partial \Theta} \tag{31}$$

Equation (30) encapsulates how the gradients are derived from both the biases and weights, effectively linking the loss function to the network parameters Θ through tightly integrated feedback loops.

IV. Experiment

In the preceding section, we detailed the training and prediction processes for our tightly coupled deep learning GNSS positioning framework. In this section, we will evaluate our approach and compare its performance against other tools. To facilitate a concise discussion, we will use the abbreviations TDL-B, TDL-W, and TDL-BW to refer to the bias correction network, weight prediction network, and combined bias correction and weight prediction network, respectively.

A. Framework Overview

Figures 2a and 2b illustrate the training and prediction processes, respectively. In the training process, pseudoranges are processed using an OLS method to provide an initial solution. Subsequently, extracted features, including C/N0,

TABLE II
THE DETAIL OF THE DATASETS

Dataset	Date	DoU	Epoch	Samples	Usage
KLT1	2021.06.10	Light	203	4676	testing
KLT2	2021.06.10	Light	209	4914	testing
KLT3	2021.06.10	Light	404	8857	training
Whampoa	2021.07.14	Deep	1205	12926	testing

TABLE III
THE DETAIL OF THE SENSORS

Sensor	Output	Frequency(Hz)	Other
SPAN-CPT	coordinate	100	/
Ublox F9P	pseudorange	1	GPS L1, BeiDou B1, Galileo E1, GLONASS G1

elevation angle, and least squares residuals, are fed into the network. The network then predicts the pseudorange errors that need correction and estimates weights for each satellite. A WLS calculation is performed to determine the position. The loss function calculates the error between the ground truth and the estimated position, and backpropagation is used to optimize the network. The prediction process mirrors the training process but concludes at the WLS step, outputting the final position.

B. Experiment Setup

Four datasets are utilized for evaluation. Three of these datasets were collected in the urban areas of Hong Kong's Kowloon Tong (KLT), and one dataset was gathered in the Whampoa area of Hong Kong. KLT is characterized as a light urban area, whereas Whampoa is considered a deep urban area with an approximate 2D positioning error of 18 meters. The specific details of each dataset are provided in Table II, where DoU represents the degree of urbanization. KLT3 was designated for training, while the remaining datasets were used for testing.

A Ublox-F9P receiver was utilized to receive and decode GNSS signals at a frequency of 1Hz. Additionally, a NovAtel SPAN-CPT system [47], providing a Real-Time Kinematic (RTK) GNSS/INS integrated solution, was used to generate centimeter-level ground truth data at 100Hz. Further details are provided in Table III. These setups are consistent with those used in our previously open-sourced UrbanNav datasets [48].

Details of the training process for the three networks are outlined in Table IV. While most parameters are consistent across the networks, the number of training epochs varies. Specifically, the TDL-BW model utilizes fewer epochs to prevent overfitting, which is a risk due to its complex nature. The training loss curves for each network are illustrated in Figure 3.

The training process involves a critical detail regarding the initial state $\mathbf{y}_0$, which should not be set as $(0, 0, 0, 0)$. As depicted in Equations (22), (16), and (29), the process consists of a two-step optimization. Initially, the position state is solved using WLS optimization, and this solution

TABLE IV
TRAINING DETAILS

Component	Specification
System	Ubuntu 20.04
CPU	AMD 5900x
Graphics Card	Nvidia 4090
Memory	128 Gigabytes
Loss Function	Mean Square Error (MSE)
Epoch	500 for TDL-B and TDL-W, 100 for TDL-BW
Optimizer	Adam [49]
Learing Rate	0.001

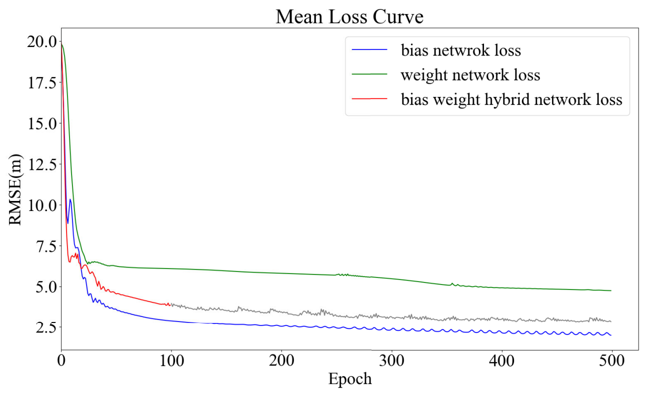


Fig. 3. The loss curves illustrate the training process for the bias and weight networks. The blue curve represents the mean position loss of the bias network, while the green curve is associated with the weight network. The red curve depicts the loss curve for the bias-weight hybrid network, terminating at the 100th epoch. The continuous gray curve highlights a potential risk of overfitting within the network.

is then tightly coupled to the network. The gradient of the weight in Equation (6) largely depends on the $\mathbf{H}$ matrix, which is derived from the current position solution. Per Equations (6) and (7), the solution accumulates iteratively. In early iterations, significant changes in the solution result in drastic alterations to the $\mathbf{H}$ matrix, leading to unstable weight gradients. This instability can cause the optimization to fall into local minima and fail to converge. To mitigate this issue, the initial state is derived from the solution of an OLS calculation, which is nearly converged. Consequently, the $\mathbf{H}$ matrix remains relatively stable, enhancing the reliability of the training process.

C. Experiment Result

1) Competing Methods: In this section, we present our results and compare them with those obtained using RTKLIB and goGPS. Specifically, in RTKLIB, the weights assigned to each measurement are primarily derived from the elevation angles:

$$\sigma^2(\theta) = a^2 + \frac{b^2}{sin^2\theta} \tag{32}$$

$$w = \frac{1}{\sigma^2(\theta)} \tag{33}$$

TABLE V
2D Mean Error on Testing Datasets

Dataset	TDL-BW (m)	TDL-B (m)	TDL-W (m)	goGPS (m)	RTKLIB (m)
KLT1	1.84	2.24	2.57	1.88	2.44
KLT2	1.86	2.35	2.89	2.66	4.48
Whampoa	10.94	17.81	16.11	13.95	20.89

TABLE VI
3D Mean Error on Testing Datasets

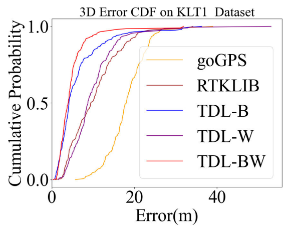
Dataset	TDL-BW (m)	TDL-B (m)	TDL-W (m)	goGPS (m)	RTKLIB (m)
KLT1	4.72	5.30	9.92	18.14	10.31
KLT2	3.92	5.89	7.75	15.44	10.94
Whampoa	28.31	49.45	42.49	50.55	60.62

In RTKLIB, the weight assigned to each measurement depends on the elevation angle, denoted by θ. The coefficients a and b, known as super parameters, are typically set at 0.3. Meanwhile, in goGPS [16], weights are computed based on both the C/N0 and the elevation angle, as follows:

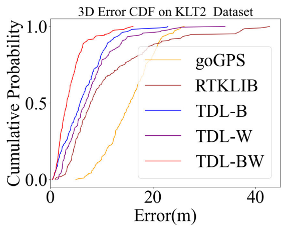
$$k_1(s) = -\frac{s - s_1}{a}, k_2(s) = \frac{s - s_1}{s_0 - s_1}$$
$$w = \begin{cases} \frac{1}{\sin^2\theta}\left(10^{k_1(S)}\left(\left(\frac{A}{10^{k_1(s_0)}} - 1\right)k_2(S) + 1\right)\right), S < s_1 \\ 1, C/N0 \geq s_1 \end{cases} \tag{34}$$

In this formula, S represents the C/N0, and θ denotes the elevation angle. The parameters A, a, s_0, and s_1 are super parameters and are typically set to 30, 20, 10, and 50, respectively. These values are crucial for determining the weights based on the quality and position of the satellite signals.

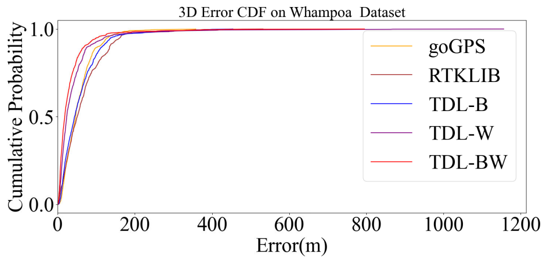
2) *Results and Analysis:* The 2D and 3D positioning MSE errors are presented in Table V and Table VI, respectively. Additionally, a cumulative distribution function (CDF) plot of the 3D errors, a boxplot of the 3D errors, and a detailed view of these errors are depicted in Figure 4, Figure 5 and Figure 6. The corresponding trajectories are illustrated in Figure 7. In the 2D error comparison, the TDL-B and TDL-W models perform worse than goGPS; however, their 3D error results are significantly better than those of both goGPS and RTKLIB. This discrepancy arises because the training process primarily utilizes the 3D error to calculate the loss function, leading to a focused analysis on 3D errors. According to Table VI, CDF Figure 4 and boxplot Figure 5, it is evident that the TDL-BW consistently outperforms others in terms of both accuracy and reliability. Among the three CDF graphs, TDL-BW demonstrates significant advantages, primarily characterized by its rapid convergence to a cumulative probability of 1, indicating superior performance in error minimization compared to other methods. The red curve of TDL-BW swiftly approaches a cumulative probability near 1, suggesting that the majority of measurement errors are confined within this lower error threshold. This performance



(a)



(b)



(c)

Fig. 4. The CDF for 3D error of the compared methods on the three datasets.

not only shows a smooth and consistent upward trend without abrupt jumps or fluctuations but also underscores its stability across various measurement conditions. Specifically in boxplot, TDL-BW exhibits the lowest mean errors and standard deviations, indicating a high level of precision and stability, which is crucial for applications requiring rigorous spatial accuracy. For instance, within the KLT1 dataset, TDL-BW achieved a mean error of 4.72 meters with a standard deviation of 4.26 meters, signifying minimal deviation in error measurement across samples. Similarly, in the KLT2 dataset, it maintained a mean error of 3.92 meters and an even lower standard deviation of 2.99 meters, further affirming its superior performance. Conversely, the weight strategies used by goGPS and RTKLIB demonstrated significant variability, particularly in the Whampoa dataset where deep urban challenging conditions likely exacerbated their performance issues; goGPS and RTKLIB recorded mean errors of 54.42 meters and 64.58 meters, respectively, coupled with high standard deviations exceeding 40 meters. These results highlight the variability and potential limitations of goGPS and RTKLIB in such conditions, suggesting a more suitable application in less demanding scenarios.

V. Discussion

In the section, we focus on two typical outliers of TDL-BW in the two representative scenarios, Case A and Case B, depicted in Figure 8. The errors in the two cases are 31.89 meters and 1061.09 meters respectively. These scenarios highlight the challenges posed by LOS and NLOS conditions on satellite signal reception and the consequent effects on positioning accuracy.

In case A, as illustrated in Figure 8a and detailed in Table VII, the TDL-BW network outputs show a clear differentiation between LOS and NLOS signals, as indicated

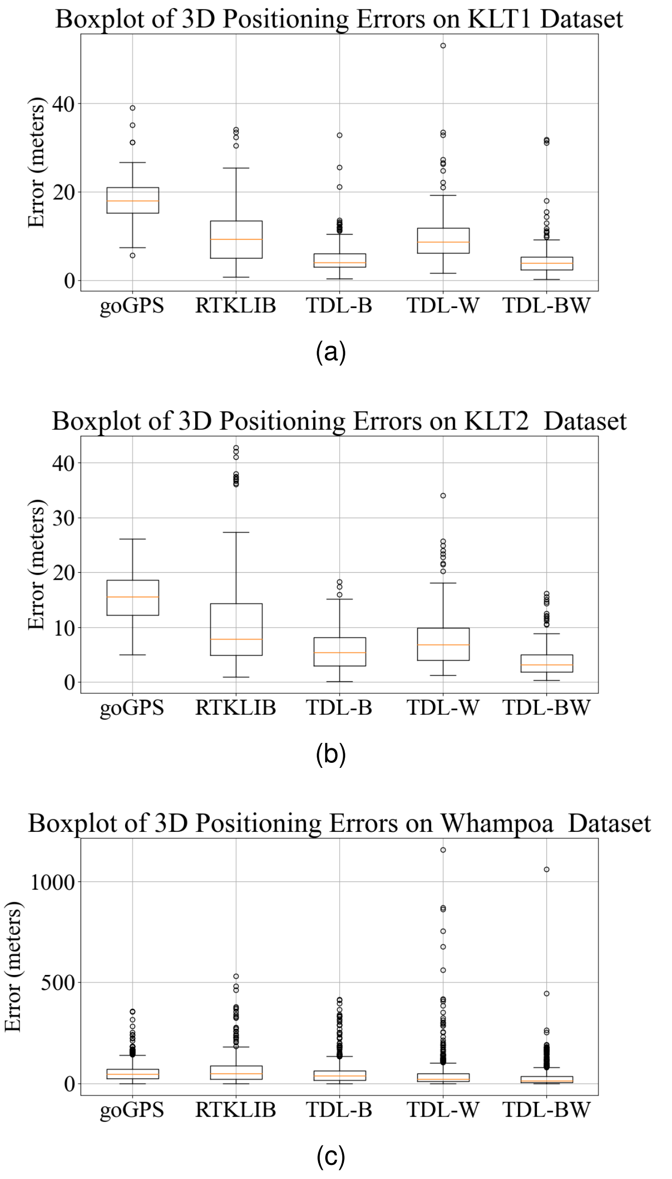


Fig. 5. The boxplot for 3D error of the compared methods on the three datasets.

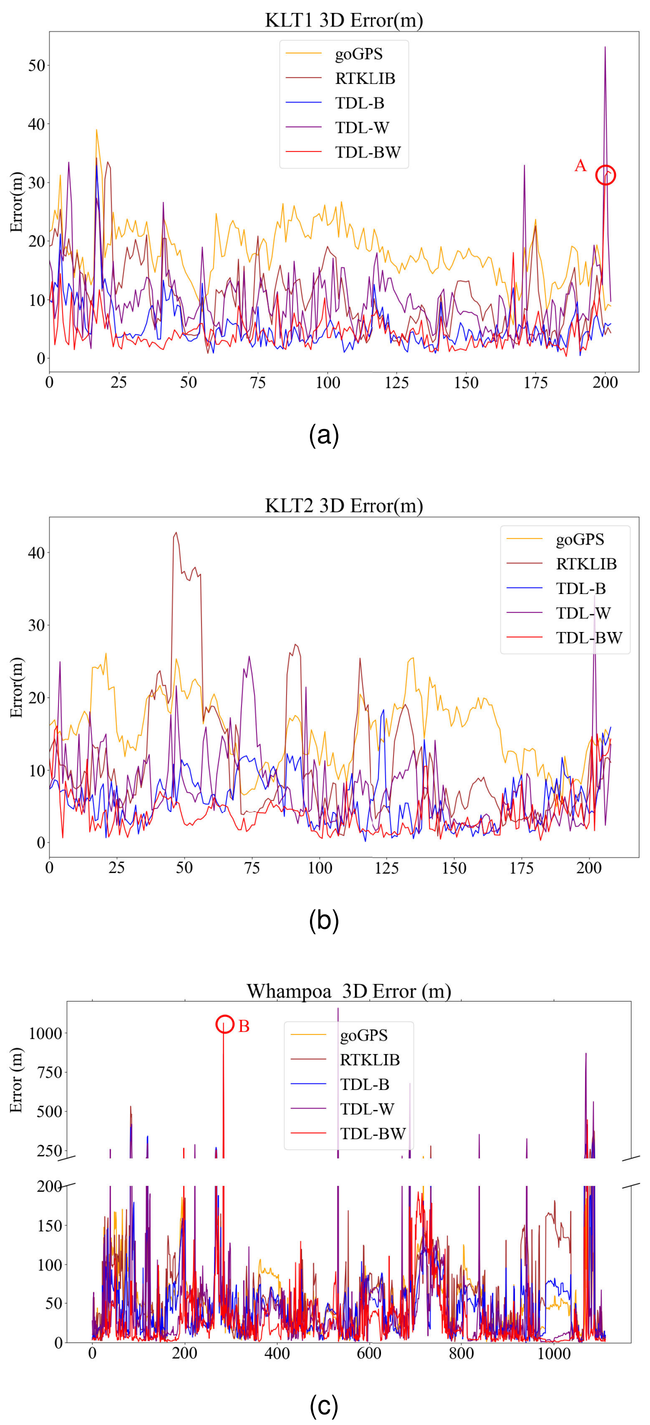


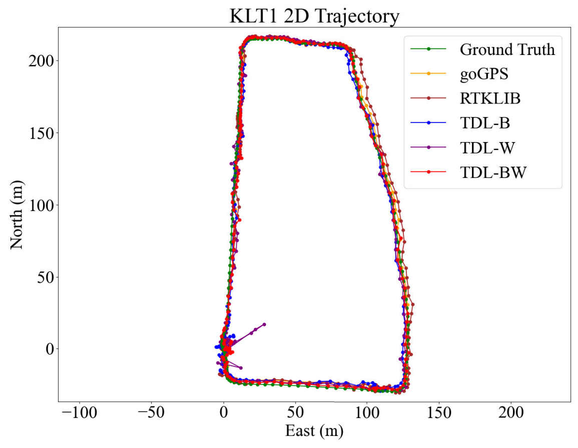
Fig. 6. The detailed 3D error of the compared methods on the three datasets.

TABLE VII
THE PREDICTED WEIGHT AND BIAS IN CASE A

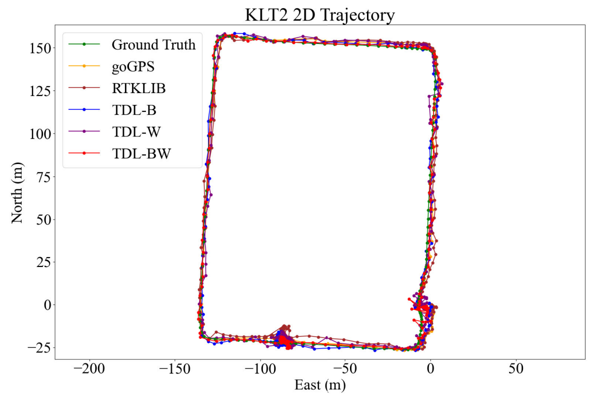
PRN	weight	bias	PRN	weight	bias
G07	0.62	4.61	G03	0.00	5.44
G01	0.36	3.12	G14	0.00	3.22
G30	0.14	3.48	G17	0.00	7.81
C11	0.12	2.27	G22	0.00	5.77
G21	0.01	5.88	C13	0.00	6.00
C07	0.00	4.72	C08	0.00	2.25
C23	0.00	6.83	C25	0.00	2.19
G28	0.00	4.41			

by the color-coded PRN entries (green for LOS and red for NLOS). This scenario, characterized by a tree canopy, predominantly exhibits NLOS conditions, impacting the weight distribution among satellites. Notably, satellites labeled as LOS (G07 and G01) receive non-zero weights, affirming the network's capability to identify viable signals amidst obstructions. However, the network misclassifies satellite G22, positioned near the edge of a building possibly affected by diffraction, assigning it no weight. A critical observation here is the excessive elimination of NLOS satellites, which, while reducing noise from obstructed signals, also minimizes redundancy in the available data for accurate positioning, potentially compromising the robustness of the positioning solution.

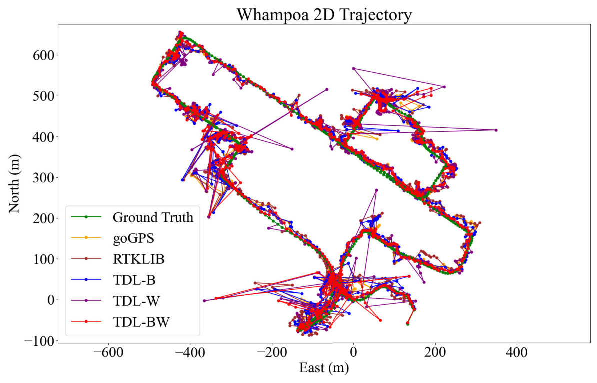
Presented in Figure 8b and Table VIII for case B, this scenario demonstrates a similar pattern where the network



(a)



(b)

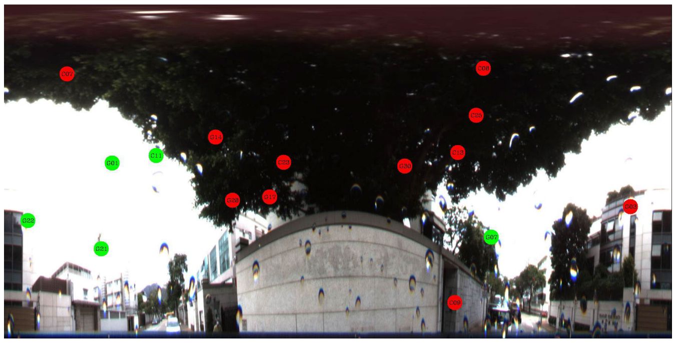


(c)

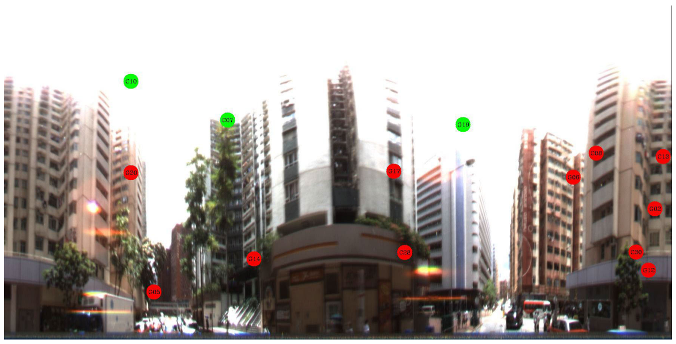
Fig. 7. The trajectory and ground truth in East-North-Up (ENU) frame of the compared methods on the three datasets.

TABLE VIII
THE PREDICTED WEIGHT AND BIAS IN CASE B

PRN	weight	bias	PRN	weight	bias
C10	1.00	0.40	G12	0.00	9.38
G19	0.96	1.90	G14	0.00	8.32
G06	0.89	3.46	G20	0.00	13.99
G17	0.65	1.86	C08	0.00	2.32
C07	0.21	0.22	C13	0.00	6.19
G02	0.00	14.72	C28	0.00	20.50
G05	0.00	11.85	C30	0.00	7.43



(a)



(b)

Fig. 8. The scenarios of case A and B.

effectively identifies and assigns higher weights to LOS satellites (C10 and G19). It also appropriately categorizes C07, which, despite potential obstructions, is deemed reliable. However, akin to Case A, the network's stringent filtering leads to the exclusion of numerous NLOS satellites, manifesting in an overly sparse dataset that might detract from the accuracy of the resultant positioning due to insufficient satellite coverage and geometry.

Both cases underscore the TDL-BW network's proficiency in distinguishing between LOS and NLOS satellites and its consequential decision-making concerning weight assignments. While this ability is advantageous for enhancing signal quality by excluding NLOS influences, it also raises concerns regarding the adequacy of satellite data for reliable positioning. The elimination of too many satellites, particularly under dense canopy or urban settings where NLOS conditions are prevalent, could severely limit the system's operational effectiveness by reducing the geometric diversity necessary for optimal positioning. The less robust performance of TDL-W in several situations is also due to the failure to allocate weights to enough measurements. In contrast, the results from TDL-B in the two cases are 5.52 meters and 223.50 meters, which are much better than TDL-BW and TDL-W. Therefore, due to the unreliable results from TDL-W and TDL-BW, where only a few measurements are weighted, it is advisable to switch to TDL-B.

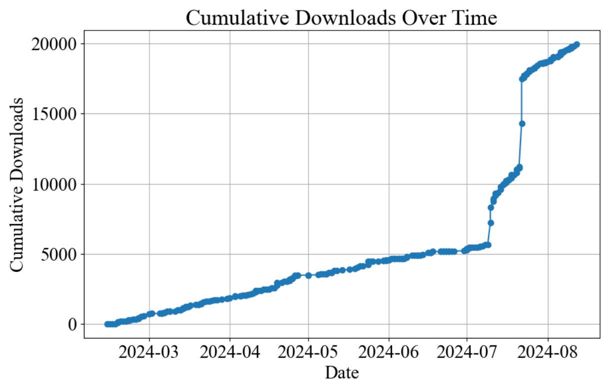


Fig. 9. The cumulative download number for **pyrtklib** over the past 6 months.

VI. Conclusion

In this paper, we introduced **pyrtklib**, a Python binding for the widely-used GNSS library, RTKLIB. Utilizing **pyrtklib**, we developed a tightly coupled deep learning aided GNSS SPP method that predicts weights and biases for each satellite, thereby enhancing positioning performance. Our methods were compared against RTKLIB and goGPS. The results demonstrate that TDL-BW, which simultaneously predicts both weights and biases, outperforms the others. This network effectively differentiates between line-of-sight (LOS) and non-line-of-sight (NLOS) satellites, assigning appropriate weights and biases accordingly. Both **pyrtklib** and the deep learning subsystem are available as open-source resources at https://github.com/IPNL-POLYU/pyrtklib and https://github.com/ebhrz/TDL-GNSS. As shown in Figure 9, bfpyrtklib has been downloaded approximately 20,000 times over the past six months.

The network structure employed in this demonstration is relatively straightforward, and the feature set used is limited to the current scope of our work. However, our framework is designed to seamlessly incorporate a broader range of deep learning models, such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), transformers and other advanced architectures, to enhance GNSS algorithms. Moving forward, we plan to expand the network architecture to account for spatial and temporal variations in GNSS data, and to integrate multi-modal inputs such as images, point clouds, and maps. Additionally, our approach will include more comprehensive techniques for predicting biases and weights in code measurements, further enriching the effectiveness of the GNSS positioning process. The value of pyrtklib lies in its ability to provide a flexible platform for researchers to easily experiment with and integrate various deep learning models, accelerating the development of innovative solutions in the field. Our ultimate aim is to bridge the gap between AI and GNSS, fostering the evolution of both fields through richer, more effective applications.

Acknowledgment

The authors would like to thank RCDSAI and the Metituan Academy of Robotics Shenzhen (H-ZGHQ) for their support. They would also like to thank their colleagues Ivan Ng, Liyuan Zhang, and Dr. Yingying Wang.

References

[1] K.-F. Chu, A. Y. S. Lam, and V. O. K. Li, “Traffic signal control using end-to-end off-policy deep reinforcement learning,” *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 7184–7195, Jul. 2022.

[2] Z. Zhu and H. Zhao, “A survey of deep RL and IL for autonomous driving policy learning,” *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 14043–14065, Sep. 2022.

[3] B. Li et al., “A new unsupervised deep learning algorithm for fine-grained detection of driver distraction,” *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 10, pp. 19272–19284, Oct. 2022.

[4] G. Zhang, H.-F. Ng, W. Wen, and L.-T. Hsu, “3D mapping database aided GNSS based collaborative positioning using factor graph optimization,” *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 10, pp. 6175–6187, Oct. 2021.

[5] L. Heng, T. Walter, P. Enge, and G. X. Gao, “GNSS multipath and jamming mitigation using high-mask-angle antennas and multiple constellations,” *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 741–750, Apr. 2015.

[6] Q. Chen, Q. Zhang, and X. Niu, “Estimate the pitch and heading mounting angles of the IMU for land vehicular GNSS/INS integrated system,” *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 10, pp. 6503–6515, Oct. 2021.

[7] H. Ng, G. Zhang, Y. Luo, and L. Hsu, “Urban positioning: 3D mapping-aided GNSS using dual-frequency pseudorange measurements from smartphones,” *Navigation*, vol. 68, no. 4, pp. 727–749, Dec. 2021.

[8] P. D. Groves, “Principles of GNSS, inertial, and multisensor integrated navigation systems, 2nd edition [Book review],” *IEEE Aerosp. Electron. Syst. Mag.*, vol. 30, no. 2, pp. 26–27, Feb. 2015.

[9] M. Abadi et al., “TensorFlow: A system for large-scale machine learning,” in *Proc. 12th USENIX Symp. Operating Syst. Design Implement. (OSDI)*, 2016, pp. 265–283.

[10] A. Paszke et al., “PyTorch: An imperative style, high-performance deep learning library,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–12.

[11] R. Dach, S. Lutz, P. Walser, and P. Fridez, “Bernese GNSS software version 5.2,” Astronomical Inst., Univ. Bern, Bern, Switzerland, Tech. Rep., 2015, doi: 10.7892/boris.72297.

[12] T. Takasu and A. Yasuda, “Development of the low-cost RTK-GPS receiver with an open source program package RTKLIB,” in *Proc. Int. Symp. GPS/GNSS*, Seogwipo-si, Republic of Korea, vol. 1, 2009, pp. 1–6.

[13] T. Pany et al., “The multi-sensor navigation analysis tool (MuSNAT)—Architecture, LiDAR, GPU/CPU GNSS signal processing,” in *Proc. 32nd Int. Tech. Meeting Satell. Division Inst. Navig. (ION GNSS+)*, 2019, pp. 4087–4115.

[14] E. Bendersky. (2022). *Complete C99 Parser in Pure Python*. [Online]. Available: https://github.com/eliben/pycparser

[15] W. Jakob. (2016). *pybind11—Seamless Operability Between C++11 and Python*. [Online]. Available: https://github.com/pybind/pybind11

[16] A. M. Herrera, H. F. Suhandri, E. Realini, M. Reguzzoni, and M. C. de Lacy, “goGPS: Open-source MATLAB software,” *GPS Solutions*, vol. 20, no. 3, pp. 595–603, 2016.

[17] J. Yin, A. Li, T. Li, W. Yu, and D. Zou, “M2DGR: A multi-sensor and multi-scenario SLAM dataset for ground robots,” *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 2266–2273, Apr. 2022.

[18] S. Cao, X. Lu, and S. Shen, “GVINS: Tightly coupled GNSS–visual–inertial fusion for smooth and consistent state estimation,” *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2004–2021, Aug. 2022.

[19] T. Li, L. Pei, Y. Xiang, W. Yu, and T.-K. Truong, “P^3-VINS: Tightly-coupled PPP/INS/visual SLAM based on optimization approach,” *IEEE Robot. Automat. Lett.*, vol. 7, no. 3, pp. 7021–7027, Jul. 2022.

[20] C. Kilic, N. Ohi, Y. Gu, and J. N. Gross, “Slip-based autonomous ZUPT through Gaussian process to improve planetary rover localization,” *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 4782–4789, Jul. 2021.

[21] P. Xu, G. Zhang, B. Yang, and L.-T. Hsu, “Machine learning in GNSS multipath/NLOS mitigation: Review and benchmark,” *IEEE Aerosp. Electron. Syst. Mag.*, vol. 39, no. 9, pp. 26–44, Sep. 2024.

[22] P. Borhani-Darian, H. Li, P. Wu, and P. Closas, “Deep learning of GNSS acquisition,” *Sensors*, vol. 23, no. 3, p. 1566, Feb. 2023.

[23] H. Li, P. Borhani-Darian, P. Wu, and P. Closas, “Deep neural network correlators for GNSS multipath mitigation,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 59, no. 2, pp. 1249–1259, Apr. 2023.

[24] M. Orabi, J. Khalife, A. A. Abdallah, Z. M. Kassas, and S. S. Saab, "A machine learning approach for GPS code phase estimation in multipath environments," in *Proc. IEEE/ION Position, Location Navig. Symp. (PLANS)*, Apr. 2020, pp. 1224–1229.

[25] R. R. Yakkati, B. Pardhasaradhi, J. Zhou, and L. R. Cenkeramaddi, "A machine learning based GNSS signal classification," in *Proc. IEEE Int. Symp. Smart Electron. Syst. (iSES)*, Dec. 2022, pp. 532–535.

[26] A. Guillard, P. Thevenon, C. Milner, and C. Macabiau, "Benefits of CNN-based multipath detection for robust GNSS positioning," in *Proc. 36th Int. Tech. Meeting Satell. Division Inst. Navig. (ION GNSS+)*, 2023, pp. 283–297.

[27] A. Blais, N. Couellan, and E. Munin, "A novel image representation of GNSS correlation for deep learning multipath detection," *Array*, vol. 14, Jul. 2022, Art. no. 100167.

[28] R. Zawislak et al., "GNSS multipath detection aided by unsupervised domain adaptation," in *Proc. 35th Int. Tech. Meeting Satell. Division Inst. Navig. (ION GNSS+)*, 2022, pp. 2127–2137.

[29] C. Jiang et al., "Convolutional neural networks based GNSS signal classification using correlator-level measurements," *Int. Arch. Photogramm., Remote Sens. Spatial Inf. Sci.*, vol. 46, pp. 61–66, Apr. 2022.

[30] A. Guillard, P. Thevenon, and C. Milner, "Using convolutional neural networks to detect GNSS multipath," *Frontiers Robot. AI*, vol. 10, May 2023, Art. no. 1106439.

[31] G. Zhang, P. Xu, H. Xu, and L.-T. Hsu, "Prediction on the urban GNSS measurement uncertainty based on deep learning networks with long short-term memory," *IEEE Sensors J.*, vol. 21, no. 18, pp. 20563–20577, Sep. 2021.

[32] T. Suzuki and Y. Amano, "NLOS multipath classification of GNSS signal correlation output using machine learning," *Sensors*, vol. 21, no. 7, p. 2503, Apr. 2021.

[33] E. Munin, A. Blais, and N. Couellan, "Convolutional neural network for multipath detection in GNSS receivers," in *Proc. Int. Conf. Artif. Intell. Data Anal. Air Transp. (AIDA-AT)*, Feb. 2020, pp. 1–10.

[34] T. Suzuki, K. Kusama, and Y. Amano, "NLOS multipath detection using convolutional neural network," in *Proc. 33rd Int. Tech. Meeting Satell. Division Inst. Navig. (ION GNSS+)*, 2020, pp. 2989–3000.

[35] S. J. Cho, B. Seong Kim, T. S. Kim, and S.-H. Kong, "Enhancing GNSS performance and detection of road crossing in urban area using deep learning," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Oct. 2019, pp. 2115–2120.

[36] Y. Quan, L. Lau, G. W. Roberts, X. Meng, and C. Zhang, "Convolutional neural network based multipath detection method for static and kinematic GPS high precision positioning," *Remote Sens.*, vol. 10, no. 12, p. 2052, Dec. 2018.

[37] S. Cho, H.-W. Seok, and S.-H. Kong, "MPCNet: GNSS multipath error compensation network via multi-task learning," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2023, pp. 1–6.

[38] H. Zhang, Z. Wang, and H. Vallery, "Learning-based NLOS detection and uncertainty prediction of GNSS observations with transformer-enhanced LSTM network," in *Proc. IEEE 26th Int. Conf. Intell. Transp. Syst. (ITSC)*, Sep. 2023, pp. 910–917.

[39] A. Mohanty and G. Gao, "Learning GNSS positioning corrections for smartphones using graph convolution neural networks," *NAVIGATION: J. Inst. Navig.*, vol. 70, no. 4, 2023, Art. no. navi.622.

[40] A. V. Kanhere, S. Gupta, A. Shetty, and G. Gao, "Improving GNSS positioning using neural-network-based corrections," *NAVIGATION: J. Inst. Navig.*, vol. 69, no. 4, 2022, Art. no. navi.548.

[41] Y. Tao et al., "Real-time multipath mitigation in multi-GNSS short baseline positioning via CNN-LSTM method," *Math. Problems Eng.*, vol. 2021, no. 1, 2021, Art. no. 6573230.

[42] A. Siemuri, K. Selvan, H. Kuusniemi, P. Välisuo, and M. S. Elmusrati, "Improving precision GNSS positioning and navigation accuracy on smartphones using machine learning," in *Proc. 34th Int. Tech. Meeting Satell. Division Inst. Navig. (ION GNSS+)*, 2021, pp. 3081–3093.

[43] F. van Diggelen, "End game for urban GNSS: Google's use of 3D building models," *Inside GNSS*, vol. 16, no. 2, pp. 42–49, 2021.

[44] P. Xu, G. Zhang, Y. Zhong, B. Yang, and L.-T. Hsu, "A framework for graphical GNSS multipath and NLOS mitigation," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 9, pp. 12176–12186, Sep. 2024.

[45] R. Hu, W. Wen, and L.-T. Hsu, "Fisheye camera aided GNSS NLOS detection and learning-based pseudorange bias correction for intelligent vehicles in urban canyons," in *Proc. IEEE 26th Int. Conf. Intell. Transp. Syst. (ITSC)*, Sep. 2023, pp. 6088–6095.

[46] M. Maaref, L. Garin, and P. McBurney, "Leveraging machine learning to mitigate multipath in a GNSS pure L5 receiver," in *Proc. 34th Int. Tech. Meeting Satell. Division Inst. Navig. (ION GNSS+)*, 2021, pp. 3740–3748.

[47] S. Kennedy, J. Hamilton, and H. Martell, "Architecture and system performance of SPAN-NovAtel's GPS/INS solution," in *Proc. IEEE/ION PLANS*, Apr. 2006, pp. 266–274.

[48] L.-T. Hsu et al., "UrbanNav: An open-sourced multisensory dataset for benchmarking positioning algorithms designed for urban areas," in *Proc. 34th Int. Tech. Meeting Satell. Division Inst. Navig. (ION GNSS+)*, 2021, pp. 226–256.

[49] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

Runzhi Hu was born in Leshan, Sichuan, China. He received the B.S. and master's degrees in mechanical engineering and computer science from China Agricultural University. He is currently pursuing the Ph.D. degree with The Hong Kong Polytechnic University. His research interests include HD maps, multi-sensor fusion, SLAM, and GNSS positioning in urban canyons.

Penghui Xu (Graduate Student Member, IEEE) received the B.S. degree from South China Agricultural University, China, in 2015, and the M.Sc. degree in mechanical engineering from The Hong Kong Polytechnic University (PolyU), Hong Kong, in 2017, where he is currently pursuing the Ph.D. degree. After the M.Sc. degree, he mainly works in machine learning algorithm development. His research interests include machine learning, GNSS urban localization, and multi-sensor integration for positioning.

Yihan Zhong received the bachelor's degree in process equipment and control engineering from Guangxi University, Nanning, China, in 2020, and the master's degree from The Hong Kong Polytechnic University (PolyU), Hong Kong, in 2022, where he is currently pursuing the Ph.D. degree with the Department of Aeronautical and Aviation Engineering (AAE). His research interests include collaborative positioning and low-cost localization.

Weisong Wen (Member, IEEE) received the B.E. degree in mechanical engineering from Beijing Information Science and Technology University (BISTU), Beijing, China, in 2015, the M.E. degree in mechanical engineering from China Agricultural University, Beijing, in 2017, and the Ph.D. degree in mechanical engineering from The Hong Kong Polytechnic University (PolyU), Hong Kong, in 2020. He was a Visiting Ph.D. Student with the Faculty of Engineering, University of California, Berkeley (UC Berkeley), Berkeley, CA, USA, in 2018. Before joining PolyU as an Assistant Professor, in 2023, he has been a Research Assistant Professor with the Department of Aeronautical and Aviation Engineering (AAE), PolyU, since 2021. He has published 30 SCI papers and 40 conference papers in the field of GNSS (ION GNSS+) and navigation for robotic systems (IEEE ICRA and IEEE ITSC), such as autonomous driving vehicles. He won the Innovation Award from TechConnect 2021, the Best Presentation Award from the Institute of Navigation (ION) in 2020, and the First Prize in the Hong Kong Section in the Qianhai-Guangdong-Macao Youth Innovation and Entrepreneurship Competition in 2019 based on his research achievements in 3-D LiDAR-aided GNSS positioning for robotics navigation in urban canyons. The developed 3-D LiDAR-aided GNSS positioning method has been reported by top magazines, such as inside GNSS, and has attracted industry recognition with remarkable knowledge transfer.