remote sensing

MDPI

Article

Sparse Decomposition-Based Anti-Spoofing Framework for GNSS Receiver: Spoofing Detection, Classification, and Position Recovery

Yuxin He [1,2], Xuebin Zhuang [3,*] and Bing Xu [2,4]

[1] The State Key Laboratory of Satellite Navigation System and Equipment Technology, Shijiazhuang 050010, China; yuxin16.he@connect.polyu.hk
[2] Department of Aeronautical and Aviation Engineering, The Hong Kong Polytechnic University, Hong Kong, China; pbing.xu@polyu.edu.hk
[3] School of Systems Science and Engineering, Sun Yat-sen University, Guangzhou 510220, China
[4] The Hong Kong Polytechnic University Shenzhen Research Institute, Shenzhen 518000, China
* Correspondence: zhuangxb@mail.sysu.edu.cn

Abstract: Achieving reliable navigation is critical for GNSS receivers subject to spoofing attacks. Utilizing the inherent sparsity and inconsistency of spoofing signals, this paper proposes an anti-spoofing framework for GNSS receivers to detect, classify, and recover positions from spoofing attacks without additional devices. A sparse decomposition algorithm with non-negative constraints limited by signal power magnitudes is proposed to achieve accurate spoofing detections while extracting key features of the received signals. In the classification stage, these features continuously refine each channel of the receiver's code tracking loop, ensuring that it tracks either the authentic or counterfeit signal components. Moreover, by leveraging the inherent inconsistency of spoofing properties, we incorporate the Hausdorff distance to determine the most overlapped position sets, distinguishing genuine trajectories and mitigating spoofing effects. Experiments on the TEXBAT dataset show that the proposed algorithm detects 98% of spoofing attacks, ensuring stable position recovery with an average RMSE of 6.32 m across various time periods.

Keywords: GNSS receiver; spoofing attack; sparse decomposition; position recovery

Academic Editor: Gino Dardanelli

Received: 23 June 2025
Revised: 27 July 2025
Accepted: 31 July 2025
Published: 4 August 2025

Citation: He, Y.; Zhuang, X.; Xu, B. Sparse Decomposition-Based Anti-Spoofing Framework for GNSS Receiver: Spoofing Detection, Classification, and Position Recovery. *Remote Sens.* **2025**, *17*, 2703. https://doi.org/10.3390/rs17152703

1. Introduction

Global Navigation Satellite Systems (GNSS) offer worldwide coverage along with precise, reliable, and real-time position, velocity, and timing (PVT) services [1]. At the same time, such systems are inherently vulnerable due to the fragility of the signals [2], which primarily stems from two factors: (i) the limited power of GNSS signals on the ground, and (ii) the open structure of these signals. Their resulting fragility makes GNSS signals susceptible to instances of satellite spoofing interference, including spoofing attacks on portable GNSS receivers, unmanned aerial vehicles, and ships [3,4]. Unfortunately, most commercial GNSS receivers are not equipped with effective countermeasures against such spoofing attacks [5].

The main challenge faced by most commercial GNSS receivers in handling spoofing interference is that when under attack, their tracking loops cannot resist external drag-off from spoofing signal, resulting in continuous generation of fake position results. This problem is further aggravated during hybrid jamming–spoofing attacks [6]. In such scenarios, the attacker first transmits high-power jamming signals to prevent the receiver from acquiring authentic satellites. When acquisition is blocked, the attacker switches to

high-power spoofing transmissions, which the receiver may mistakenly lock onto during startup. These increasingly sophisticated spoofing strategies highlight the need for robust countermeasures capable of detecting spoofing interference and mitigating its impact on receiver position solutions.

Currently, the majority of anti-spoofing algorithms for GNSS receivers concentrate on spoofing detection [7–9]. Spoofing classification was accomplished by [10] through spoofing and authentic parameter estimation. However, merely detecting or classifying spoofing signals is not enough; the more challenging and crucial task lies in eliminating the impact of spoofing signals on receivers that have already been spoofed and ensuring the generation of authentic PVT results.

To address these issues, methods based on multiple antennas that involve controlling the antenna steering patterns have been proposed to generate nulls and eliminate spoofing signals [11,12]. Furthermore, sensor fusion-based anti-spoofing interference techniques were propose in [13,14]. Using a fifth-generation (5G) signal base station has also proven helpful in spoofing mitigation [15]. However, the above approaches rely on adding extra devices to increase information redundancy, which not only increases costs but also limits the application scenarios of anti-spoofing algorithms for GNSS receivers [16]. The design of an anti-spoofing framework that relies on the observed data of only a single GNSS receiver to achieve reliable position results in generic scenarios remains an open problem worthy of further investigation.

To avoid excessive utilization of extra devices, Receiver Autonomous Integrity Measurement (RAIM) [17] provides a statistical reliability testing method which assumes a scenario in which a single visible satellite is spoofed. While effective, RAIM cannot discern spoofing when most channels are subject to spoofing, which is often the case during spoofing attacks. Lately, algorithms proposed by in [18,19] have employed vector tracking loops to estimate the magnitude, propagation delay, and carrier phase of spoofing attacks. However, this approach relies on prior assumptions such as classifying signals with higher amplitudes and longer delays as inauthentic, which may not be valid in common matched-power and intermediate spoofing scenarios. To address this limitation, the design of a Maximum Likelihood Estimation (MLE) principle with multiple correlator arrays was presented in [20] to estimate and then mitigate GNSS spoofing attacks. In [5], the authors introduced transitioning between multipath estimation of the delay lock loop and coupled amplitude delay lock loop for continuously tracking of authentic signal components. Nevertheless, the methods proposed in [5,20] both assume that the receiver operates normally during tracking before spoofing begins. This assumption is often unrealistic, since in practice it is difficult for a receiver to determine exactly when a spoofing attack begins.

In addition, many of the above methods [5,17–20] focus on identifying and mitigating attacks based on self-defined prior knowledge or assumptions during the tracking stage. These spoof mitigation strategies cannot track all contributing components simultaneously to achieve both authentic and fake pseudo-range measurements for each satellite. As a result, they miss opportunities to exploit redundant information deep within during the navigation phase of position computation, making them more difficult to apply in more general scenarios that go beyond prior assumptions.

Motivated by the above discussion, we propose a novel anti-spoofing framework for solving the spoofing detection, classification, and position recovery problem of GNSS receivers under frequency-locked spoofing attacks. The salient feature of our algorithm is that it continuously tracks all the received signals' contributing components and further exploits the intrinsic inconsistency of the spoofing signals, thereby eliminating the need for additional limitations such as added extra devices, prior assumptions, and specific

initial states. Compared with the existing literature, the main contributions of this work are highlighted as follows:

(1) Unlike methods the proposed in [7–9], which only focus on spoofing detection, we devise a sparse decomposition algorithm with non-negative constraints limited by received signal power magnitudes, which not only achieves accurate spoofing detection but also simultaneously extracts key features of the received signal's contributing components, achieving reliable spoofing classification.
(2) Distinct from the methods introduced in [5,17–20], we adopt Advanced Iterative Hard Thresholding (AIHT) to integrate the key features extracted from our sparse decomposition method into Auxiliary Peak Tracking (APT), enabling separate tracking of spoofing and authentic components of each satellite to derive the true and spoofed pseudo-range measurements of each satellite. In this way, the intrinsic inconsistency of the spoofing signals can be further exploited without any extra devices and prior assumptions.
(3) By leveraging the inherent inconsistency of spoofing properties, we incorporate the Hausdorff distance to determine the most overlapped position sets to identify genuine position trajectories in general scenarios. Compared with the methods proposed in [10,18,20], this mitigates the impacts of spoofing in position recovery without specific initial state limitations.
(4) The efficacy and advantage of the proposed anti-spoofing framework are fully illustrated by extensive experiments conducted on the public TEXBAT dataset, showing that our algorithm detects 98% of spoofing attacks and guarantees stable position recovery with an average RMSE of 6.32 m across various time periods.

The rest of this paper is organized as follows: the spoofing signal model and analysis of the spoofed receiver correlators are provided in Section 2; Section 3 presents the design process of the sparse decomposition-based spoofing detection, classification, and position recovery framework; Section 4 shows the experimental results conducted with the TEXBAT dataset; Finally, Section 6 concludes the paper.

2. Problem Background

2.1. Spoofing Signal Model

After down-conversion from radio frequency (RF), the received signals in each channel of the receiver are composed of base-band GNSS signals and thermal noise. Without loss of generality, the GPS L1 signal is taken as the representative of different types of satellite signals in the following sections. The structure of a typical down-converted GPS L1 signal in the ith channel is described as follows:

$$s_{\mathrm{ai}}(mT) = \sqrt{P_{\mathrm{ai}}} C_{\mathrm{i}}(mT - \tau_{\mathrm{ai}}) D_{\mathrm{i}}(mT) e^{j(2\pi f_{\mathrm{i}} mT + \theta_{\mathrm{ai}})} \tag{1}$$

where the subscript label $*_a$ usually refers to parameters of an authentic satellite signal, $*_i$ refers to the parameters of the ith channel in the receiver, P_{ai} is the power of the received authentic signal, m is the discrete sample index of the tracking start time, T is the sampling period defined in a software receiver, τ_{ai} and f_{i} respectively denote the code phase and Doppler frequency, θ_{ai} is the carrier phase parameter, and C_{i} and D_{i} represent the C/A spreading code and the navigation message, respectively.

Similarly, the frequency-locked spoofing attack [10] of $s_{\mathrm{ai}}(mT)$, which is carried out by intermediate spoofers and replicate authentic Doppler frequency f_{i}, is formulated as follows:

$$s_{\mathrm{si}}(mT) = \sqrt{P_{\mathrm{si}}} C_{\mathrm{si}}(mT - \tau_{\mathrm{si}}) D_{\mathrm{i}}(mT) e^{j(2\pi f_{\mathrm{i}} mT + \theta_{\mathrm{si}})} \tag{2}$$

where the subscript label $*_s$ means a counterfeit signal, while P_{si}, τ_{si} and θ_{si} respectively represent the power, code, and carrier parameters of the spoofing signal $s_{\mathrm{si}}(mT)$ in the ith channel. According to Equations (1) and (2), when spoofing attack exists, the received signal $s_{\mathrm{i}}(mT)$ in the ith channel is modeled as

$$s_{\mathrm{i}}(mT) = s_{\mathrm{ai}}(mT) + s_{\mathrm{si}}(mT) + n(mT), \tag{3}$$

where $n(mT)$ is assumed to be Additive White Gaussian Noise (AWGN) combined with loop noise. To prevent the navigation message $D_{\mathrm{i}}(mT)$ from changing during two consecutive integration periods, the coherent integration period T_0 is maintained at less than half the bit duration. This ensures a constant navigation bit, and is omitted in the subsequent discussions in this article.

2.2. Analysis of the Spoofed Receiver Correlators

In the tracking process, the local replicas are generated by GNSS receivers to continuously achieve carrier and code wipe-off, and the locally constructed replica in the ith channel is provided as follows:

$$r(mT, \hat{\tau}_{\mathrm{i}}) = C_{\mathrm{i}}(mT - \hat{\tau}_{\mathrm{i}})e^{j(2\pi f_{\mathrm{i}} mT + \hat{\theta}_{\mathrm{i}})} \tag{4}$$

where $\hat{\tau}_{\mathrm{i}}$ and $\hat{\theta}_{\mathrm{i}}$ represent the estimated code and carrier parameters in time, respectively. By leveraging the local replica $r(mT, \hat{\tau}_{\mathrm{i}})$ in Equation (4), the correlator outcome in the ith channel $y(\tau_{\mathrm{i}})$ is

$$\begin{aligned} y(\tau_{\mathrm{i}}) =& \frac{1}{N_{\mathrm{c}}} \sum_{m}^{m+N_{\mathrm{c}}-1} s_{\mathrm{i}}(mT) r(mT, \hat{\tau}_{\mathrm{i}})^* \\ =& \sqrt{P_{\mathrm{ai}}} R(\Delta\tau_{\mathrm{ai}}) e^{j\Delta\theta_{\mathrm{A}}} + \sqrt{P_{\mathrm{si}}} R(\Delta\tau_{\mathrm{si}}) e^{j\Delta\theta_{\mathrm{S}}} + \tilde{\eta} \\ =& y_{\mathrm{A}}(\tau_{\mathrm{i}}) + y_{\mathrm{S}}(\tau_{\mathrm{i}}) + \tilde{\eta} \end{aligned} \tag{5}$$

with

$$\Delta\tau_{\mathrm{si}} = \tau_{\mathrm{si}} - \hat{\tau}_{\mathrm{i}},\ \Delta\tau_{\mathrm{ai}} = \tau_{\mathrm{ai}} - \hat{\tau}_{\mathrm{i}},\ \Delta\theta_{\mathrm{si}} = \theta_{\mathrm{si}} - \hat{\theta}_{\mathrm{i}},\ \Delta\theta_{\mathrm{ai}} = \theta_{\mathrm{ai}} - \hat{\theta}_{\mathrm{i}},$$

where $N_{\mathrm{c}} = f_{\mathrm{s}} T_0$ is the number of samples of the coherent integration period T_0 and $f_{\mathrm{s}} = \frac{1}{T_0}$ is the sampling frequency. In addition, $(\cdot)^*$ denotes the complex conjugate operator, $R(\cdot)$ denotes the auto-correlation function (ACF) depicted as an isosceles triangle, $\tilde{\eta}$ is the combination of different noise components, including the thermal noise in real signals and counterfeit signals, and $y_{\mathrm{A}}(\tau_{\mathrm{i}})$ and $y_{\mathrm{S}}(\tau_{\mathrm{i}})$ respectively denote the authentic signal correlation component and spoofing signal correlation component.

It is worth noting that τ_{i} represents discrete time indexes, which is related to the estimated parameter $\hat{\tau}_{\mathrm{i}}$. When τ_{i} takes different values, a sequence $\boldsymbol{r}(mT)$ of local replica elements with incremental time delays D is constructed, forming a post-correlation vector $\boldsymbol{y} = [y(\tau_{\mathrm{i}} + D) \cdots y(\tau_{\mathrm{i}} + ND)]^T$. Here, N is the number of correlator taps, which can be rewritten as follows:

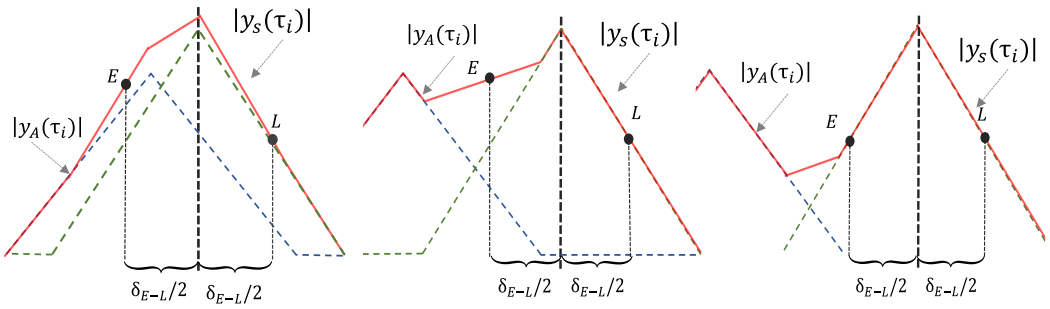
$$\boldsymbol{y} = \boldsymbol{y}_{\mathrm{A}} + \boldsymbol{y}_{\mathrm{S}} + \tilde{\boldsymbol{\eta}} \tag{6}$$

where $\tilde{\boldsymbol{\eta}}$ is the noise vector of $\tilde{\eta}$ in different taps, while $\boldsymbol{y}_{\mathrm{A}} = [y_{\mathrm{A}}(\tau_{\mathrm{i}} + D) \cdots y_{\mathrm{A}}(\tau_{\mathrm{i}} + ND)]^T$ and $\boldsymbol{y}_{\mathrm{S}} = [y_{\mathrm{S}}(\tau_{\mathrm{i}} + D) \cdots y_{\mathrm{S}}(\tau_{\mathrm{i}} + ND)]^T$ respectively represent the ACF envelops of the real and counterfeit signal components of $\boldsymbol{y}$.

As shown in Equation (6), the postcorrelation vector $\boldsymbol{y}$ is composed of two main components when the satellite signal is spoofed, namely, an authentic signal correlation component $\boldsymbol{y}_{\mathrm{A}}$ and a spoofing signal correlation component $\boldsymbol{y}_{\mathrm{S}}$. The same conclusion can also be drown in any correlator outcome elements $y(\tau_{\mathrm{i}})$:

$$y(\tau_i) = y_A(\tau_i) + y_S(\tau_i) + \tilde{\eta}. \tag{7}$$

Figure 1 shows how $y(\tau_i)$ is distorted by the spoofing component $y_S(\tau_i)$, altering its normal triangular shape. This figure also details the drag-off process in delay-locked loops (DLLs) at various stages. As the code phase gap $\Delta\tau_{as} = \tau_{ai} - \tau_{si}$ between $y_A(\tau_i)$ and $y_S(\tau_i)$ widens, the DLLs are increasingly misled into locking onto $y_S(\tau_i)$. When the drag-off process is complete, $\boldsymbol{y}_S$ gains control of the DLLs, indicating a successful spoofing attack. In this study we focus on frequency-locked attacks, which presents a more sophisticated challenge for detection and mitigation efforts than the unlocked attacks due to decoupling of the carrier phase changes from code phase shifts [21].



(a) $\Delta\tau_{as}<1$ **(b)** $1<\Delta\tau_{as}<2$ **(c)** $2<\Delta\tau_{as}$

Figure 1. Different stages of spoofing signals attempting to take control of DLLs, where $\boldsymbol{y}$, $\boldsymbol{y}_A$, and $\boldsymbol{y}_S$ are depicted in the red, blue, and green triangles, respectively.

3. Methodology

To recover generic positions of the GNSS receivers under spoofing attacks, the following design procedure is provided and utilized, as shown in Figure 2:

i In the detection phase, we leverage the sparse nature of the spoofed ACF and apply the AIHT algorithm with an additional non-negative constraint to enhance the accuracy of spoofing interference detection;

ii During the classification stage, we introduce an AIHT-based APT method that tracks both authentic and counterfeit components of the same satellite using dual channels, termed a channel pair. Using code phase gap estimations from the modified AIHT, this method allows for continuous adjustments to the channel pairs.

iii Finally, by employing different selection schemes, we obtain various position results; we then apply Hausdorff distance to identify the most consistent result, where the greatest number of candidate position sets overlap. This result is considered to be the true position, and the channels within the corresponding selection scheme are recognized for tracking authentic components.

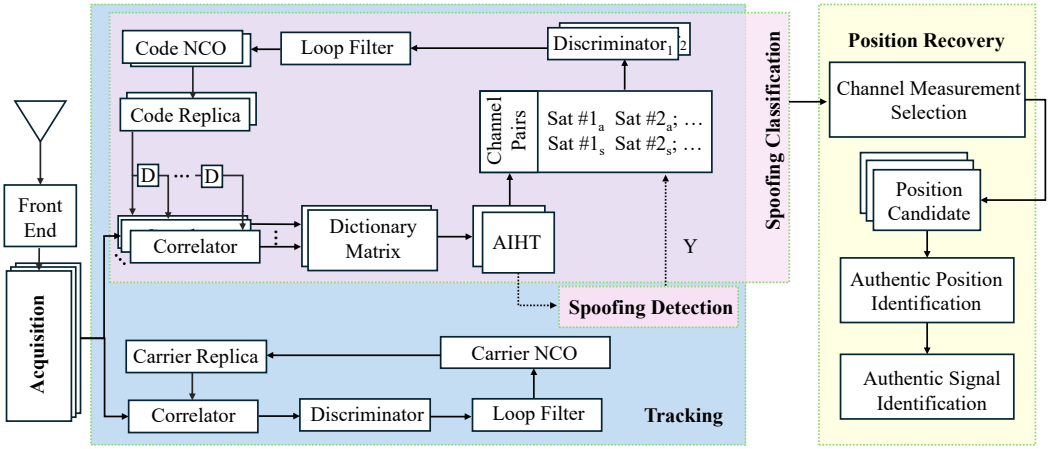


Figure 2. Flowchart of the proposed spoofing detection, classification, and position recovery algorithm.

3.1. Sparse Decomposition-Based Spoofing Detection

According to findings in [22], the spoofed postcorrelation vector outcome $\boldsymbol{y}$ inherently possesses a sparse representation in the original domain, which is provided as follows:

$$\underbrace{\begin{bmatrix} y(\tau_1) \\ \vdots \\ y(\tau_n) \end{bmatrix}}_{\boldsymbol{y}} = \underbrace{\begin{bmatrix} q_{11} & \cdots & q_{1n} \\ \vdots & \ddots & \vdots \\ q_{n1} & \cdots & q_{nn} \end{bmatrix}}_{\boldsymbol{Q}} \underbrace{\begin{bmatrix} \beta_1 \\ \vdots \\ \beta_n \end{bmatrix}}_{\boldsymbol{\beta}} + \boldsymbol{e} \tag{8}$$

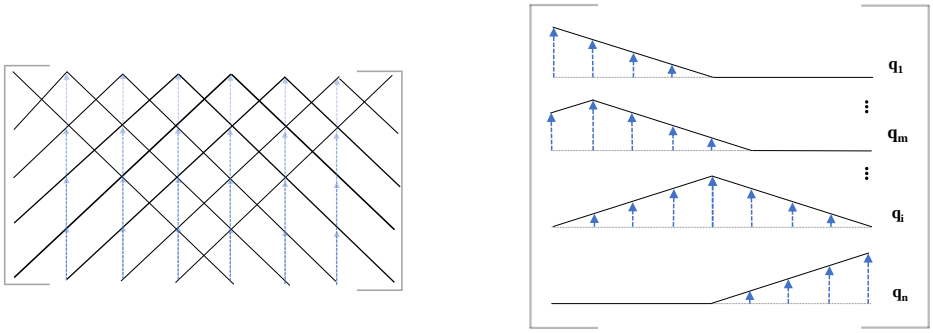
where $\boldsymbol{Q}$ is a high-resolution dictionary, $\boldsymbol{\beta}$ is a sparse vector, and $\boldsymbol{e} \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$ represents the observation error due to loop noise and hardware limitations, with variance σ^2.

The underlying assumption of this sparse representation of $\boldsymbol{y}$ is that $\boldsymbol{y}$ exhibits sparsity in the ACF dictionary domain, particularly under spoofing attacks. In the context of compressed sensing, a vector $x \in \mathbb{R}^n$ is considered sparse if only a small number of its components are significantly nonzero, i.e., $\|x\|_0 \ll n$, where $\|x\|_0$ denotes the number of nonzero elements. Under spoofing conditions, the ACF profile exhibits two dominant correlation peaks, one from the authentic signal and one from the spoofed signal; all other components are negligible, and primarily arise from thermal noise or weak multipath effects. This results in the sparse representation in (8), where the associated sparse coefficient vector $\boldsymbol{\beta}$ has only a few nonzero elements corresponding to those dominant code phases, satisfying $\|\beta\|_0 \ll n$.

As introduced by [10], the high-resolution dictionary $\boldsymbol{Q}$ depicted in Figure 3 is composed of N m-sequence ACF triangle vectors with different code delays, which is modeled as follows:

$$\boldsymbol{Q} = \left[\boldsymbol{R}(0))^T \cdots \boldsymbol{R}((N-1)D))^T \right]^T \tag{9}$$

where $\boldsymbol{R}(\tau_i) = [R(\tau_i) \cdots R(\tau_i - (N-1)D)]$ is the ACF triangle vector of m-sequence at code phase τ_i.



(**a**) Dictionary matrix. (**b**) Correlation triangles of different code phases.

Figure 3. Dictionary matrix $\boldsymbol{Q}$ construction.

Given $\boldsymbol{y}$ aggregates ACF triangles of each signal component with distinct amplitude and code phases, and understanding the role of the dictionary $\boldsymbol{Q}$, the operation $\boldsymbol{Q} * \boldsymbol{\beta}$ aims to reconstruct optimally $\boldsymbol{y}$. The specific sparse vector $\boldsymbol{\beta}$ that minimizes the error $\boldsymbol{e}$ reveals the amplitude, count, and phase differences of the contributing signal components, for which the detailed explanations are as follows:

- *Amplitude:* The relative amplitude of each component corresponds to the element value in $\boldsymbol{\beta}$. Elements below T_{th} are disregarded.
- *Count:* When the target receiver works normally, $\boldsymbol{y}$ exhibits a single peak and $\hat{\boldsymbol{\beta}}$ has an element exceeding T_{th}. During spoofing, $\boldsymbol{y}$ displays superimposed peaks from $\boldsymbol{y}_{\text{A}}$ and $\boldsymbol{y}_{\text{S}}$ such that two elements in $\hat{\boldsymbol{\beta}}$ exceed T_{th}.
- *Code Phase:* With N set, the non-zero indices in $\boldsymbol{\beta}$ identify the code phase of the peaks of $\boldsymbol{y}_{\text{A}}$ and $\boldsymbol{y}_{\text{S}}$, denoted as τ_{i} and τ_{j}, respectively, enabling precise mapping of contributing components' code phases in constructing $\boldsymbol{y}$.

T_{th} is a normalized threshold within the range $(0, 1]$, and is applied after scaling the sparse vector $\hat{\boldsymbol{\beta}}$ by its maximum value. Because the non-zero elements of $\hat{\boldsymbol{\beta}}$, representing power magnitudes, are inherently positive, T_{th} must be strictly greater than zero. Collectively, the detection is determined by the number of elements in the normalized $\hat{\boldsymbol{\beta}}$ exceed T_{th}. If this count is less than two, the receiver is considered not spoofed; otherwise, it is considered to be under spoofing attacks. Therefore, achieving an accurate estimation of $\boldsymbol{\beta}$ is crucial for both spoofing detection and classification.

3.2. Advanced IHT Algorithm

In order to achieve the sparsest possible representation of $\boldsymbol{y}$ and find the best match of $\boldsymbol{\beta}$, Equation (8) is modified as follows [23]:

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \|\boldsymbol{y} - \boldsymbol{Q}\boldsymbol{\beta}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{\beta}\|_0 \leq \text{K} \tag{10}$$

where $\hat{\boldsymbol{\beta}}$ represents the estimated sparse vector obtained through minimization of the cost function. In addition, the ℓ_0 norm denotes the number of non-zero components in $\boldsymbol{\beta}$, while K represents the prior coefficient level.

The IHT algorithm [24] decomposes sparse signals by retaining only the K largest absolute coefficients per iteration, setting the rest to zero. In this way, all elements in the vector $\hat{\boldsymbol{\beta}}$ should be zero or positive regardless of spoofing. However, as shown in Figure 4a, loop noise and hardware limitations can distort $\boldsymbol{y}$ from its ideal shape, leading the IHT algorithm to overfit by mistaking noise for signal components. This results in a sparse vector with negative or excessive non-zero elements.

Furthermore, as shown in Figure 4a, the IHT algorithm reconstructs the envelope $\hat{\boldsymbol{y}}_0 = \boldsymbol{Q}\hat{\boldsymbol{\beta}}$, closely matching the distorted outcome $\boldsymbol{y}$ affected by thermal noise, including several negative elements. Referring to the procedure outlined by [10], if negative elements are discarded, the two largest positive elements (red bars in Figure 4a) would be interpreted as the code phases τ_i and τ_j of the two main signal components; however, as shown by the green line and red bars in Figure 4a, neglecting the negative elements not only introduces significant error into the reconstructed result $\hat{\boldsymbol{y}}$, it also causes τ_i and τ_j to be inaccurate.

Thus, an Advanced IHT (AIHT) is designed to ensure non-negative output estimation:

$$\hat{\boldsymbol{\beta}}^{+} = \underset{\boldsymbol{\beta}^{+}}{\operatorname{argmin}} \left\| \boldsymbol{y} - \boldsymbol{Q}\boldsymbol{\beta}^{+} \right\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{\beta}\|_0 \leq \mathrm{K} \tag{11}$$

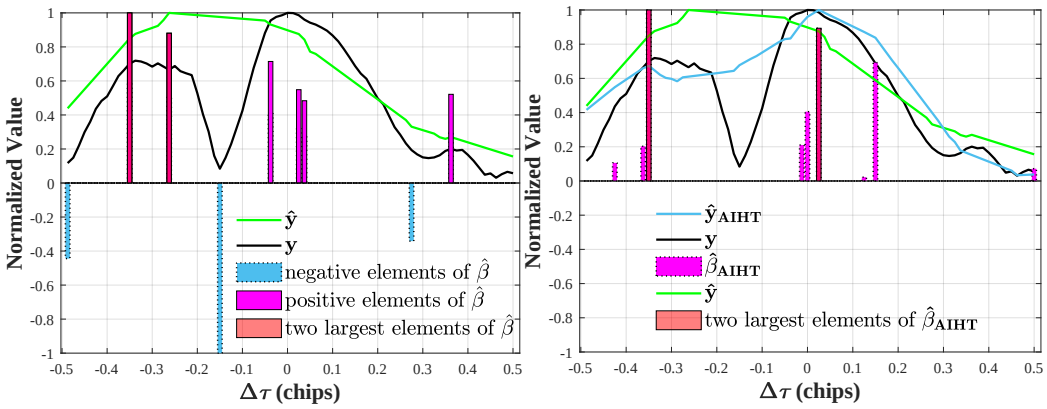
where $\boldsymbol{\beta}^{+}$ is the sparse vector with non-negative elements and $\hat{\boldsymbol{\beta}^{+}}$ is the estimated result of $\boldsymbol{\beta}^{+}$. To solve the optimization problem shown in Equation (11), the following iterative algorithm is adopted:

$$\boldsymbol{\beta}^{n+1} = H_\mathrm{K}^{+}\left(\boldsymbol{\beta}^{n} + \boldsymbol{Q}^{H}(\boldsymbol{y} - \boldsymbol{Q}\boldsymbol{\beta}^{n})\right) \tag{12}$$

where $\boldsymbol{\beta}^{n+1}$ represents the estimated sparse vector $\boldsymbol{\beta}^{+}$ in nth iteration and we have

$$H_\mathrm{K}^{+}(\beta_\mathrm{i}) = \begin{cases} 0, & \text{if } \beta_\mathrm{i} < \lambda_\mathrm{K}^{0.5}(\boldsymbol{\beta}^{+}), \\ \beta_\mathrm{i}, & \text{if } \beta_\mathrm{i} \geqslant \lambda_\mathrm{K}^{0.5}(\boldsymbol{\beta}^{+}), \end{cases} \tag{13}$$

where $\lambda_\mathrm{K}^{0.5}(\boldsymbol{\beta}^{+})$ is set to the Kth largest value of $\boldsymbol{\beta}^{n} + \boldsymbol{Q}^{H}(\boldsymbol{y} - \boldsymbol{Q}\boldsymbol{\beta}^{n})$. If less than K values are positive, then we define $\lambda_\mathrm{K}^{0.5}(\boldsymbol{\beta}^{+})$ to be the smallest value of the positive coefficients.



(**a**) Conventional IHT algorithm, where $\hat{\boldsymbol{y}} = \boldsymbol{Q} * [\text{positive elements of } \hat{\boldsymbol{\beta}}]$.

(**b**) Proposed AIHT algorithm, where $\hat{\boldsymbol{y}}_{\mathrm{AIHT}} = \boldsymbol{Q} * \hat{\boldsymbol{\beta}}_{\mathrm{AIHT}}$.

Figure 4. Comparison of reconstruction performance between the conventional IHT algorithm and the proposed AIHT algorithm.

Next, we explain theoretically why the proposed AIHT is less likely to include false estimation of τ_i and τ_j due to noise $\boldsymbol{e}$. Suppose that the estimated support $\hat{\boldsymbol{S}}$ of $\boldsymbol{\beta}^{+}$ is defined as

$$\hat{\boldsymbol{S}} = \{i \mid \beta_i \neq 0\} \tag{14}$$

and that $\boldsymbol{S}$ is the real support of $\boldsymbol{\beta}$. For any atom $\boldsymbol{q}_j$ within $\boldsymbol{Q}$ that is not in the true support $\boldsymbol{S}$, the project z_j reduces to

$$z_j = \boldsymbol{q}_j^{\top}\boldsymbol{y} = \boldsymbol{q}_j^{\top}(\boldsymbol{Q}\boldsymbol{\beta} + \boldsymbol{e}) = \boldsymbol{q}_j^{\top}\boldsymbol{e} \sim \mathcal{N}(0, \sigma^2\|\boldsymbol{q}_j\|^2), \tag{15}$$

i.e., the projection is approximately Gaussian, since $\boldsymbol{q}_j$ is approximately orthogonal to the dictionary of real support $\boldsymbol{Q}_S$ under low-coherence assumptions [25].

In classical IHT, support selection is based on the top K largest magnitudes $|z_j|$, meaning that large negative projections due to noise can be incorrectly included. In contrast, AIHT excludes negative projections by design and selects only the top K positive ones. Therefore, the probability of a false atom being included in the support is strictly lower under AIHT. The probabilities of false support estimation under the classic IHT and AIHT methods can be compared by examining the tail of the Gaussian distribution:

$$\mathbb{P}_{\text{IHT}}(\text{false support}) = \mathbb{P}(|z_j| > T_{\text{th}}) = 2Q\left(\frac{T_{\text{th}}}{\sigma\|\mathbf{q}_j\|}\right) \tag{16}$$

while the probability of it exceeding T_{th} in the positive direction (as used in AIHT) is

$$\mathbb{P}_{\text{AIHT}}(\text{false support}) = \mathbb{P}(z_j > T_{\text{th}}) = Q\left(\frac{T_{\text{th}}}{\sigma\|\mathbf{q}_j\|}\right). \tag{17}$$

Here, $Q(x)$ is the Gaussian tail function, defined as $Q(x) = \frac{1}{\sqrt{2\pi}}\int_x^\infty e^{-t^2/2}dt$, which provides the probability of a standard normal random variable exceeding x. Clearly, $\mathbb{P}_{\text{AIHT}} < \mathbb{P}_{\text{IHT}}$, meaning that AIHT is less likely to include false atoms due to noise.

In addition, we can consider the reconstruction error bound. Because AIHT always chooses the top non-negative K components and the signal amplitude is also non-negative, the correct support is preserved throughout the iterations. Therefore, the error bound from IHT carries over to AIHT as follows [25]:

$$\|\boldsymbol{y} - \boldsymbol{Q}\boldsymbol{\beta}^\star\| \leq \frac{\lambda_{\text{K}}^{0.5}(\boldsymbol{\beta}^\star)}{\alpha(\boldsymbol{Q})} \tag{18}$$

where the positive constant $\alpha(\boldsymbol{Q})$ denotes the condition number of $\boldsymbol{Q}$ and $\boldsymbol{\beta}^\star$ denotes the vector $\boldsymbol{\beta}^+$ at any fixed point.

As depicted by the green line in Figure 4a, negative elements overlooked by the IHT algorithm result in substantial deviations in the reconstructed $\hat{\boldsymbol{y}}$ and estimated τ_{A} and τ_{S} within a satellite. Conversely, by incorporating a non-negative constraint of $\boldsymbol{\beta}^+$, compared to $\boldsymbol{\beta}$ in Equation (10), as indicated by the blue line and red bars in Figure 4b, our AIHT algorithm not only significantly reduces the reconstruction error of $\hat{\boldsymbol{y}}$ but also precisely determines τ_{i} and τ_{j}, which aligns more accurately with the signal components $\boldsymbol{y}_{\text{A}}$ and $\boldsymbol{y}_{\text{S}}$. Thus, AIHT provides enhanced accuracy in the detection and classification of spoofing signals.

3.3. Spoofing Classification via AIHT-Based APT Algorithm

The primary objective of spoofing classification is to ensure a stable APT process for each satellite. To achieve this, our AIHT algorithm accurately extracts the respective code phases β_{i} and β_{j} of both the spoofing and genuine signal components from the combination of postcorrelation peaks $\boldsymbol{y}$. Therefore, an AIHT-based APT algorithm is designed to estimate code phases τ_{i} and τ_{j} while separately achieving steady tracking of each satellite's signal components $\boldsymbol{y}_{\text{A}}$ and $\boldsymbol{y}_{\text{S}}$. Let us assume i $>$ j and that a spoofing signal is successfully detected. Then, the iterative correction steps of the proposed AIHT-based APT algorithm are as follows:

(1) *Initial Correction*: Since it remains unclear which of the two elements corresponds to $\boldsymbol{y}_{\text{A}}$, we begin by recovering the correlation peaks $\hat{\boldsymbol{y}}_{\text{i}}$ and $\hat{\boldsymbol{y}}_{\text{j}}$, which represent two

distinct components of the overall received signal $\hat{\boldsymbol{y}}$ from the pth spoofed satellite. This is done using the coefficients β_i and β_j from $\hat{\boldsymbol{\beta}}^+$:

$$\begin{cases} \hat{\boldsymbol{y}}_i = \boldsymbol{Q}\hat{\boldsymbol{\beta}}_i^+ \\ \hat{\boldsymbol{y}}_j = \boldsymbol{Q}\hat{\boldsymbol{\beta}}_j^+ \end{cases} \tag{19}$$

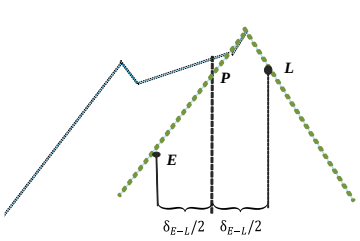
where $\hat{\boldsymbol{\beta}}_i^+ = [\underbrace{0\cdots0}_{1\times(i-1)}, \beta_i, \underbrace{0\cdots0}_{1\times(r-i)}]$ and $\hat{\boldsymbol{\beta}}_j^+ = [\underbrace{0\cdots0}_{1\times(j-1)}, \beta_j, \underbrace{0\cdots0}_{1\times(r-j)}]$ denote that the matrices only contain the non-zero element at the estimated code phase β_i and β_j, indicating the specific peaks extracted from the overall signal.

(2) *Channel Allocation for Tracking*: Two separate channels, referred to as a channel pair, are allocated to track the pth spoofed satellite. For a GNSS receiver tracking P satellites, a total of P channel pairs, comprising 2P independent digital channels, are required:

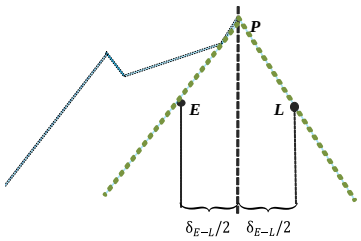
$$\{[C_{1,1}, C_{1,2}], \cdots, [C_{P,1}, C_{P,2}]\} \tag{20}$$

where $[C_{p,1}, C_{p,2}]$ represents a channel pair tracking different components of a single satellite for $p = 1, 2, \cdots, P$. Each channel within a pair is dedicated to one of the two corrected correlation peaks, $\hat{\boldsymbol{y}}_i$ and $\hat{\boldsymbol{y}}_j$, ensuring simultaneous tracking of both signal components. Specifically, $C_{p,1}$ tracks $\hat{\boldsymbol{y}}_i$, associated with $\boldsymbol{\beta}_i$, while $C_{p,2}$ tracks $\hat{\boldsymbol{y}}_j$, associated with $\boldsymbol{\beta}_j$.

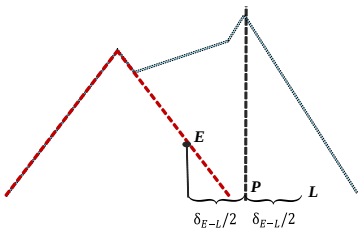
(3) *Continuous Update and Tracking*: AIHT-based correction is continuously repeated to update coefficients β_i and β_j in each channel, guaranteeing continuous correlation peak corrections and consequently steady tracking across all tracking pairs.



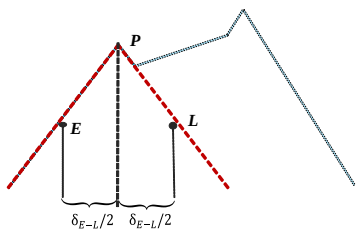
(**a**) Before tracking peak $\hat{\boldsymbol{y}}_j$



(**b**) After tracking peak $\hat{\boldsymbol{y}}_j$



(**c**) Before tracking peak $\hat{\boldsymbol{y}}_i$



(**d**) After tracking peak $\hat{\boldsymbol{y}}_i$

Figure 5. The early (E), prompt (P) and late (L) values selected by the DLL; here, δ_{E-L} means the early–late separation.

Based on the aforementioned iterative correction steps, as shown in Figure 5a,b, $\hat{\boldsymbol{y}}_j$ represents the green correlation triangle, which the DLL progressively tracks. Similarly, in Figure 5c,d, $\hat{\boldsymbol{y}}_i$ represents the red correlation triangle, and the second channel tracks it after correction. Thus, two separate channels within a pair track the genuine and spoofing peak components of a spoofed satellite signal. This simultaneous tracking enables the possibility of exploiting the inherent inconsistencies in spoofing signals, providing an essential foundation for designing position recovery methods.

3.4. Position Recovery

In the position recovery stage, by evaluating the position sets generated by different channel selection schemes, we identify specific channels that only track the authentic component $\boldsymbol{y}_{\mathrm{A}}$ of each satellite. The results can be extended to guarantee an accurate position solution.

First, one channel is selected from each channel pair $[C_{\mathrm{p},1}, C_{\mathrm{p},2}]$ which has completed tracking. In total, $\mathrm{P}-1$ channels are chosen from all P channel pairs. Therefore, the channel selection scheme $G_g = [C_{1,x}, C_{2,x}, \cdots, C_{\mathrm{P}\cdot 2^{\mathrm{P}\text{-}1},x}]$, where $x = 1$ or 2, has a total of $\boldsymbol{C}_{\mathrm{P}}^{\mathrm{P}-1}(\boldsymbol{A}_2^1)^{\mathrm{P}\text{-}1} = \mathrm{P}\cdot 2^{\mathrm{P}\text{-}1}$ possibilities. Then, each channel selection scheme G_g produces different pseudo-ranges for each satellite. Different channel selection schemes result in varying pseudo-ranges for each satellite. Finally, during the receiver's PVT estimation stage, the Extended Kalman Filter (EKF) is applied to integrate these pseudo-ranges, yielding varying position results.

Because the receiver calculates position periodically, each G_g produces different position trajectories over time. These trajectories, representing discrete position solutions, form unique sets. In Figure 6, each set is labeled S_i, where $i = 1, 2, \ldots, \mathrm{P}\cdot 2^{\mathrm{P}-1}$ corresponds to a particular channel selection scheme; thus, there are $\mathrm{P}\cdot 2^{\mathrm{P}-1}$ distinct sets, with each set representing a specific selection scheme. Among all these selection schemes, there exist three different scenarios:

(1) *Authentic selection schemes* (P sets): Composed of channels that exclusively track genuine signals; when $\mathrm{P} > 4$, these sets tend to produce consistent position results.
(2) *Fake selection schemes* (P sets): Made up of channels that only track spoofed signals; these sets exhibit varied position outcomes due to the spoofing signals' inability to continuously generate drag-off phases while simultaneously ensuring a uniform position across all satellites.
(3) *Mixed selection schemes* ($\mathrm{P}\cdot(2^{\mathrm{P}-1}-2)$ sets): Consisting of channels tracking both genuine and spoofed signals, the position results from these sets also vary significantly.

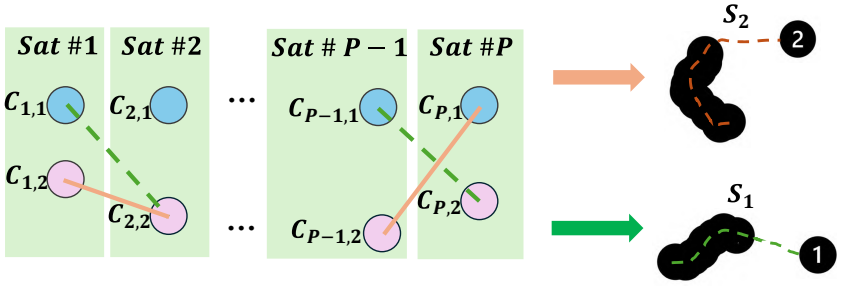


Figure 6. Formation of distinct position sets S_i based on different channel selection schemes.

Collectively, there are P overlapping position sets S_i, corresponding to authentic position solutions generated by authentic selection schemes. The remaining position sets produced by fake and mixed schemes are scattered, with relatively large distances between them.

In order to range the distance and compare the similarity between two sets between different position sets, the Hausdorff distance d_{H} [26] is adopted. Let the bth and cth channel selection scheme corresponds to position set S_b and S_c, respectively, where $b, c = 1, 2, \cdots, \mathrm{P}\cdot 2^{\mathrm{P}-1}$ and $b \neq c$. Then, the Hausdorff distance between S_b and S_c is defined as

$$d_{\mathrm{H}}(S_b, S_c) = max\{h(S_b, S_c), h(S_c, S_b)\},$$

while the shortest distance $h(S_b, S_c)$ from S_b to S_c is

$$h(S_b, S_c) = \max\{\min\{d(p_\mathrm{b}, p_\mathrm{c}) | p_\mathrm{b} \in S_b\}, p_\mathrm{c} \in S_c\},$$

where p_b and p_c are arbitrary points in S_b and S_c, respectively, and $d(p_\mathrm{b}, p_\mathrm{c})$ represents the distance between any two points in S_b and S_c.

To evaluate the similarity between two different output result sets, we normalize the Hausdorff distance between any two candidate sets S_b and S_c by dividing it by the maximum pairwise distance:

$$\tilde{d}_\mathrm{H}(S_b, S_c) = \frac{d_\mathrm{H}(S_b, S_c)}{\max_{b \neq c} d_\mathrm{H}(S_b, S_c)}. \tag{21}$$

A normalized threshold D_{thre} is then applied to determine valid overlaps. If $\tilde{d}_\mathrm{H}(S_b, S_c)$ between S_b and S_c is less than D_{thre}, then sets S_b and S_c are considered to overlap approximately. By calculating the Hausdorff distance d_H between each pair of sets, a distance matrix F can be constructed as

$$F = \begin{bmatrix} \tilde{d}_\mathrm{H}(1,1) & \cdots & \tilde{d}_\mathrm{H}(1, \mathrm{P} \cdot 2^{\mathrm{p}-1}) \\ \tilde{d}_\mathrm{H}(2,1) & \cdots & \tilde{d}_\mathrm{H}(2, \mathrm{P} \cdot 2^{\mathrm{p}-1}) \\ \vdots & \ddots & \vdots \\ \tilde{d}_\mathrm{H}(\mathrm{P} \cdot 2^{\mathrm{p}-1}, 1) & \cdots & \tilde{d}_\mathrm{H}(\mathrm{P} \cdot 2^{\mathrm{p}-1}, \mathrm{P} \cdot 2^{\mathrm{p}-1}) \end{bmatrix}. \tag{22}$$

As shown in Figure 7, the distance matrix F is symmetric, and each row (or column) of matrix F corresponds to the normalized distance $\tilde{d}_\mathrm{H}$ between the current set of positions and other sets of positions. By identifying the gth row (or column)

$$F(g,:) = \left[\tilde{d}_\mathrm{H}(g,1), \cdots, \tilde{d}_\mathrm{H}(g, \mathrm{P} \cdot 2^{\mathrm{P}-1})\right] \quad \text{or} \quad F(:,g) = \left[\tilde{d}_\mathrm{H}(1,g), \cdots, \tilde{d}_\mathrm{H}(\mathrm{P} \cdot 2^{\mathrm{P}-1}, g)\right]$$

with the maximum number of overlapping sets, all channels included in the gth selection scheme G_g are able to track the true signal, thereby obtaining the correct position solution S_g for the receiver.

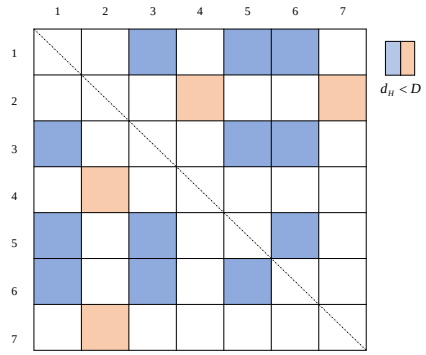


Figure 7. Distance matrix F with color-coded elements: blue for values below D_{thre} and yellow for values above.

3.5. System Overview

Collectively, the pseudocode of implementation for the proposed spoofing detection, classification and position recovery algorithm is summarized in Algorithm 1. Under different channel selection schemes, the receiver's operating state can be analyzed as follows:

- If our algorithm is initiated before spoofing attacks, it confirms that the target receiver is not under spoof, and no further steps are implemented unless the outcome of the AIHT algorithm suggests an opposite decision.
- If our algorithm is run after spoofing attacks, the receiver switches to alarm mode to perform classification and position recovery. The authentic position solution S_g is identified by selecting the gth row (or column) of the distance matrix F with the highest number of overlapping sets.

Thus, from the process in Algorithm 1, it can be found that there is no limitation in the start time when applying our algorithm.

In the following, we further analyze the computational complexity in Algorithm 1 to illustrate the practicality of our results. Recall that the signal data length J and P satellites can be tracked, the computational complexity of the tracking loops is $O(\mathrm{P} \times \mathrm{J})$, and the AIHT's complexity mirrors that of the conventional IHT algorithm, which is $O(\mathrm{K} \times \mathrm{N})$. Consequently, the cumulative complexity of the proposed AIHT algorithm before position recovery amounts to $O(\mathrm{P} \times \mathrm{K} \times \mathrm{N} \times \mathrm{J})$. In the position recovery process, assuming that the EKF operation of the PVT estimation has a computational complexity of $O(\mathrm{E})$, the iterative process is conducted $(\mathrm{P} \times (2\mathrm{P} - 1))$ times in various channel selections, and a pair of tracking loops needs to be calculated for each satellite. Thus, the computational complexity post-spoofing detection is $O(\mathrm{P} \times \mathrm{K} \times \mathrm{N} \times \mathrm{J}) + O(\mathrm{P} \times (2\mathrm{P} - 1) \times \mathrm{E})$, which can be run by most commercial GNSS receivers.

Algorithm 1 The proposed spoofing detection, classification and position recovery algorithm.

Input: Satellite number P, correlation outputs $\boldsymbol{y}_{\mathrm{p}}$, prior coefficient level K, reconstruction error ε, threshold T_{th} and D_{thre}.
Output: Receiver state, selection scheme G_g of $(F(g,:))$ or column $(F(:,g))$ in F.

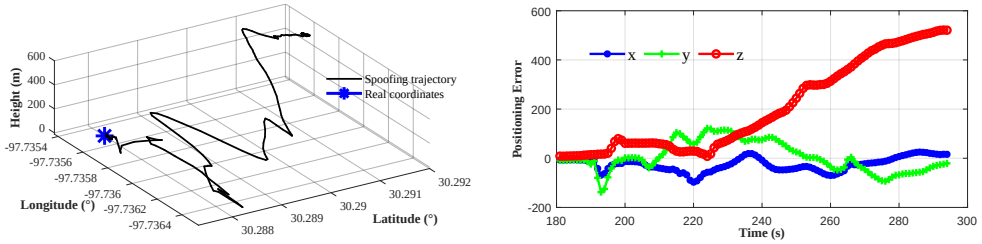
Initializing 2P digital channels
while $t < t_{\mathrm{end}}$ **do**
  if No spoofing signals are detected **then**
    for $\mathrm{p} = 1, 2, \cdots, \mathrm{P}$ **do**
      Initializing dictionary $\boldsymbol{Q}$
      while $\left(\left\| \boldsymbol{y}_{\mathrm{p}} - \boldsymbol{Q}\boldsymbol{\beta}^{+} \right\|_2^2 > \varepsilon \right)$ **do**
        $\hat{\boldsymbol{\beta}}^{+} = \underset{\boldsymbol{\beta}^{+}}{\operatorname{argmin}} \|\boldsymbol{y} - \boldsymbol{Q}\boldsymbol{\beta}^{+}\|_2^2, \quad s.t. \quad (\|\boldsymbol{\beta}\|_0 \leq \mathrm{K})$
      end while
      if $\nexists \beta_{\mathrm{i}}$ which $\beta_i \geq T_{\mathrm{th}}$ **then**
        No existence of spoofing.
      else if $\exists \beta_{\mathrm{i}}, \beta_{\mathrm{j}}$ which $\beta_{\mathrm{i}}, \beta_{\mathrm{j}} \geq T_{\mathrm{th}}$ **then**
        Receiver under spoofing attack
      end if
    end for
  else
    if $\mathrm{P} > 4$ **then**
      Activating position recovery
      for $\mathrm{p} = 1, 2, \cdots, \mathrm{P}$ **do**
        Obtaining corrected $\hat{\boldsymbol{y}}_{\mathrm{p_i}}$, $\hat{\boldsymbol{y}}_{\mathrm{p_j}}$ by AIHT-based APT method
        Tracking $\hat{\boldsymbol{y}}_{\mathrm{p_i}}$ and $\hat{\boldsymbol{y}}_{\mathrm{p_j}}$ within channel pair $[\mathrm{C_{p,1}}$ and $\mathrm{C_{p,2}}]$
      end for
      Form selection scheme G_g from different channel pairs $G_r = [C_{1,x}, C_{2,x}, \cdots, C_{\mathrm{P}\cdot 2^{\mathrm{P}-1},x}], x \in \{1, 2\}$
    end if
  end if
end while
for $r = 1, 2, \cdots, \mathrm{P} \cdot 2^{\mathrm{P}-1}$ **do**
  Form position set S_r under scheme G_r
  Calculate $F(r,:)$ or $F(:,r)$
end for
Form the matrix F
Identify vector $F(g,:)$ or $F(:,g)$ with each normalized elements smaller than D_{thre}.

4. Experimental Results

In this section, we consider a GNSS receiver which suffers from spoofing attacks. The proposed anti-spoofing algorithm in Algorithm 1 is deployed to achieve spoofing detection, classification, and position recovery. The public TEXBAT dataset provided by the University of Texas at Austin containing GPS L1 base-band spoofing data [21,27] is utilized in the experiments. The dataset is postprocessed using FGI-GSRx, which is an open-source GPS software-defined receiver (SDR) developed by the Finnish Geospatial Research Institute (FGI) [28]. Extensive experiments are conducted to fully illustrate the efficiency and advantages of each stage of the proposed spoofing algorithm.

4.1. Performance Analysis of AIHT Algorithm

In this subsection, scenario 4 in TEXBAT is utilized to evaluate the proposed AIHT detection algorithm's performance. As shown in Figure 8, the spoofing attack begins to distort the channel at the 190-s mark.

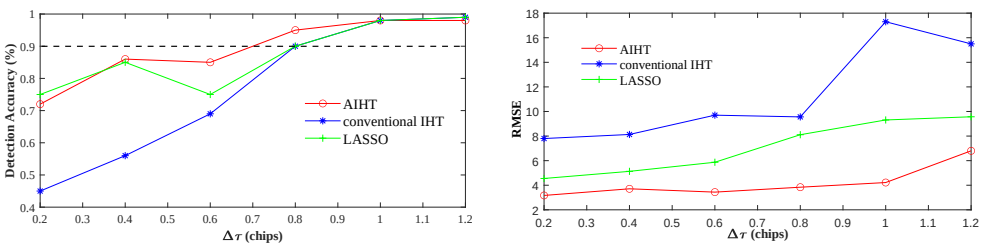


(**a**) position errors caused by spoofing signals. (**b**) Receiver trajectory under spoofing attack.

Figure 8. Position results of a GNSS receiver under spoofing interference in scenario 4 of the TEXBAT dataset.

Moreover, to illustrate the efficiency and advantage of our algorithm, we further analyze it in comparison with some related methods, including a traditional IHT algorithm [25] and a Least Absolute Shrinkage and Selection Operator (LASSO)-based spoofing detection algorithm [10].

Figure 9a compares the detection accuracy of our AIHT algorithm against the conventional IHT and LASSO methods. To achieve 90% accuracy, AIHT can detect spoofing for delays as small as $\Delta\tau > 0.71$ chips, whereas both IHT and LASSO require $\Delta\tau > 0.8$ chips under the same criterion. Figure 9b displays the root mean square error (RMSE) between $\boldsymbol{y}$ and the reconstructed result $\hat{\boldsymbol{y}}$ under different algorithms. Collectively, the proposed AIHT algorithm not only maintains a satisfying detection rate in a small code phase gap between the authentic and spoofing peak components, it also has the smallest reconstruction error, implying outstanding ability to recognize the corresponding code phases $\hat{\boldsymbol{\beta}}_{\mathrm{i}}^{+}$ and $\hat{\boldsymbol{\beta}}_{\mathrm{j}}^{+}$ of different signal components.



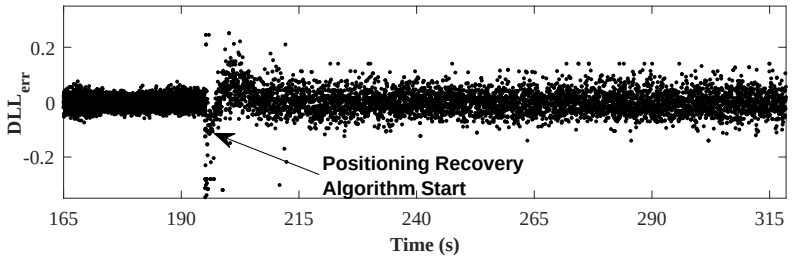
(**a**) Spoofing interference detection accuracy. (**b**) Signal reconstruction results.

Figure 9. Comparison of spoofing detection accuracy and reconstruction RMSE across algorithms for various code phase intervals.

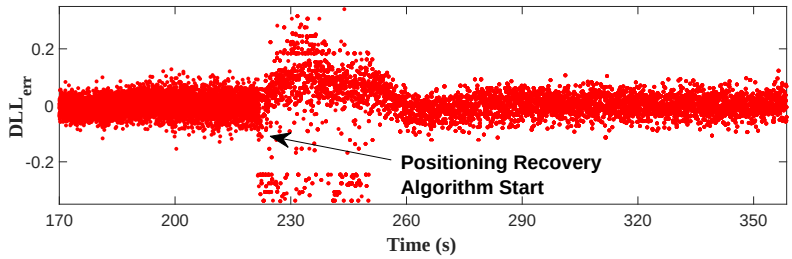
4.2. Robustness Analysis of DLL Under AIHT-Based APT Algorithm

In this section, the fluctuation of the DLL is analysed over different time periods to evaluate the stability of DLL loop calibration under our AIHT-based APT algorithm. Again, we use scenario 4 in the TEXBAT dataset.

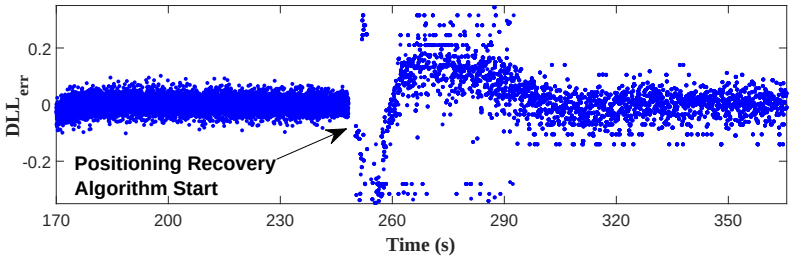
As shown in Figure 10, during the time periods of 195–215 s, 220–250 s and 250–280 s, the oscillation amplitude $\mathrm{DLL_{err}}$ decreases and the DLL loop tends to stabilize after tracking and alignment with the target signal component $\hat{\boldsymbol{y}}_{\mathrm{i}}$ or $\hat{\boldsymbol{y}}_{\mathrm{j}}$. Upon detection of spoofing signals, AIHT-based APT algorithm begins. Figure 11 depicts the actual DLL correction process of the proposed AIHT-based APT algorithm for different values of $\Delta\tau_{\mathrm{ij}}$. The tracking process in channels $C_{\mathrm{p},1}$ and $C_{\mathrm{q},1}$, illustrated in Figure 11a,c, respectively, gradually locks onto $\hat{\boldsymbol{y}}_{\mathrm{i}}$, while the tracking process in channels $C_{\mathrm{p},2}$ and $C_{\mathrm{q},2}$, detailed in Figure 11b,d, respectively, gradually locks onto $\hat{\boldsymbol{y}}_{\mathrm{j}}$, $p \neq q$.



(**a**) Discriminator oscillation from 195 s to 210 s.



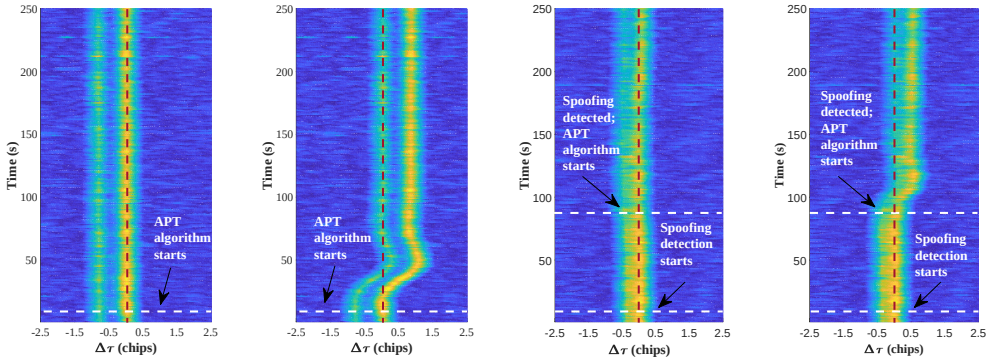
(**b**) Discriminator oscillation from 220 s to 250 s.



(**c**) Discriminator oscillation during 250 to 280 s.

Figure 10. The oscillation of DLLs induced by APT at different time intervals.

The tracking adjustments depicted in Figure 11 correlate with changes in the discriminator of the DLL, as shown in Figure 10. When the initial correlation peak is not the one the channel is designated to lock onto, $\mathrm{DLL_{err}}$ undergoes significant adjustments to shift from initially aligning with $\hat{\boldsymbol{y}}$ to the calibrated components $\hat{\boldsymbol{y}}_{\mathrm{i}}$ or $\hat{\boldsymbol{y}}_{\mathrm{j}}$. Upon successful re-locking onto the correct peak, $\mathrm{DLL_{err}}$ diminishes and approaches zero.

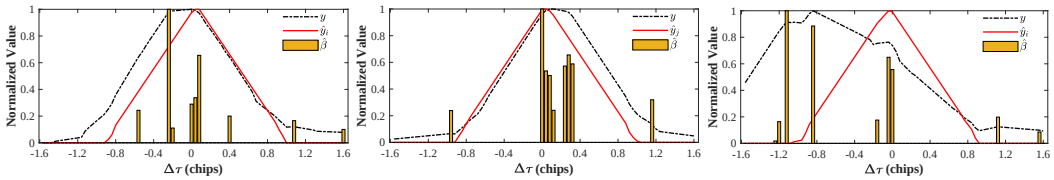


(**a**) $\hat{y}_{\mathrm{i}}$ tracked by $C_{\mathrm{p},1}$ (**b**) $\hat{y}_{\mathrm{j}}$ tracked by $C_{\mathrm{p},2}$ (**c**) $\hat{y}_{\mathrm{i}}$ tracked by $C_{\mathrm{q},1}$ (**d**) $\hat{y}_{\mathrm{j}}$ tracked by $C_{\mathrm{q},2}$

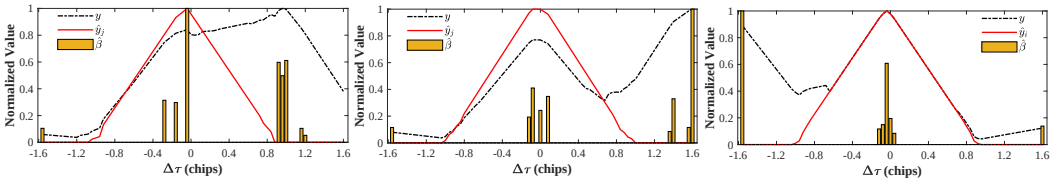
Figure 11. Tracking status of the channel pair $\left[C_{\mathrm{p},1}, C_{\mathrm{p},2}\right]$ when code interval $\Delta\tau_{\mathrm{ij}} > 1$ (**a**,**b**) and $\left[C_{\mathrm{q},1}, C_{\mathrm{q},2}\right]$ when code interval $\Delta\tau_{\mathrm{ij}} < 1$ (**c**,**d**).

As shown in Figure 12, different signal components of the channel pair $([C_{p,1}, C_{p,2}])$ are stably tracked and calibrated, with the DLL loop aligned precisely to the assigned correlation peak $(\hat{y}_i)$ or $(\hat{y}_j)$. Figure 12a,b illustrates the channel pair when the code phase difference $(\Delta\tau_{ij})$ is less than 1.

Figure 12c,d shows the channel pair for $(1.5 > \Delta\tau_{ij} \geq 1)$, while Figure 12e,f depicts the channel pair for $(\Delta\tau_{ij} \geq 1.5)$. These figures emphasize the efficacy of AIHT-driven code phase estimations in achieving independent channel adjustments within a channel pair, ensuring stable AIHT-based APT tracking and minimizing fluctuations in subsequent position recovery.



(**a**) Tracking status of $\hat{y}_{\mathrm{i}}$ when $\Delta\tau_{\mathrm{ij}} < 1$ (**b**) Tracking status of $\hat{y}_{\mathrm{j}}$ when $\Delta\tau_{\mathrm{ij}} < 1$ (**c**) Tracking status of $\hat{y}_{\mathrm{i}}$ when $1.5 > \Delta\tau_{\mathrm{ij}} \geq 1$



(**d**) Tracking status of $\hat{y}_{\mathrm{j}}$ when $1.5 > \Delta\tau_{\mathrm{ij}} \geq 1$ (**e**) Tracking status of $\hat{y}_{\mathrm{i}}$ when $\Delta\tau_{\mathrm{ij}} \geq 1.5$ (**f**) Tracking status of $\hat{y}_{\mathrm{j}}$ when $\Delta\tau_{\mathrm{ij}} \geq 1.5$

Figure 12. Tracking status of the calibrated $\hat{y}_{\mathrm{i}}$ and $\hat{y}_j$ after correction, with different code delays $\Delta\tau_{\mathrm{ij}}$ between channels.

4.3. Evaluation of the Receiver's Position Recovery Results

In this section, the position recovery performance of our algorithm is fully evaluated under the scenario 4 in TEXBAT, where the spoofing attack is initiated at 190 s and gradually introduces a 600 m erroneous position offset in the earth-centered earth-fixed (ECEF) coordinates. The true coordinates of the receiver in the WGS84 coordinate system are known to be latitude $30°17'15.068''N$, longitude $97°44'08.642''E$, and altitude 170 m.

Figure 13 illustrates the different position results obtained by proposed spoofed measurement mitigation algorithm discussed in Section 3.4 for different channel selections

among all channel pairs, from the 300 s to the 350 s. Because the receiver in scenario 4 can receive $P = 7$ satellites, there are a total of $C_P^{P-1}(A_2^1)^{P-1} = 448$ position sets for different channel results.

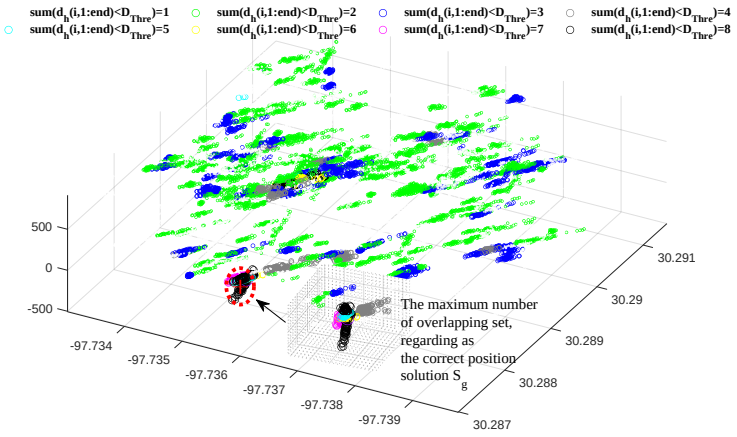
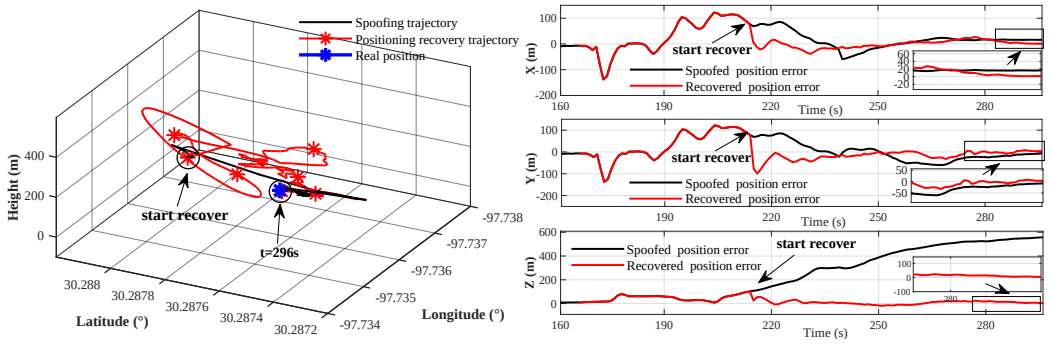


Figure 13. Different position results for different channel selections.

As shown in Equation (22), d_{H} is calculated between each pair of the 448 position sets, resulting in a 448 × 448 matrix F. Setting the threshold $T_{\mathrm{th}} = 50\%$, $D_{\mathrm{Thre}} = 8\%$, the search for the rows $F(g,:)$ (or columns $F(:,g)$) with the most overlaps in matrix F allows for the selection of eight sets of approximately overlapping position trajectories, namely, $\sum_{1<i<448} d_{\mathrm{H}}(\mathrm{i},g) < D_{Thre}$, which are regarded as the true position solution S_g of the target receiver. Based on these approximately overlapping position sets and the corresponding channel selections, channel identification for tracking the true component in $[C_{\mathrm{p},1}, C_{\mathrm{p},2}]$ can be achieved.

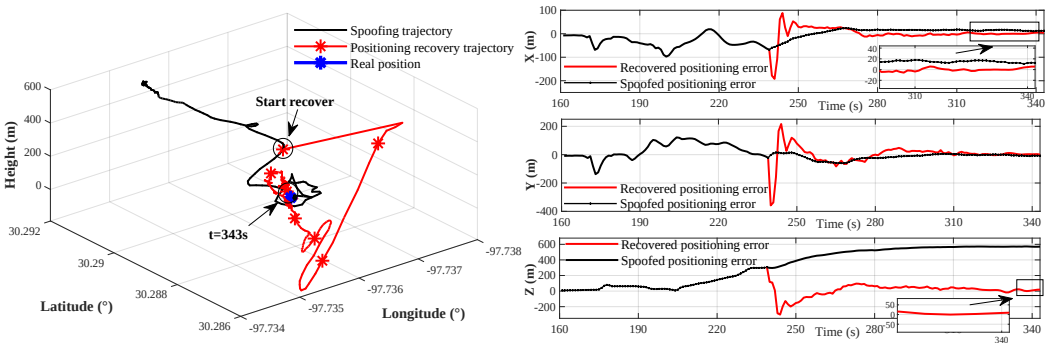
Figures 14–16 illustrate the identification results of the position recovery trajectories and dimensional position error of the position recovery algorithm in different time periods.



(**a**) Position recovery trajectory (**b**) Position error

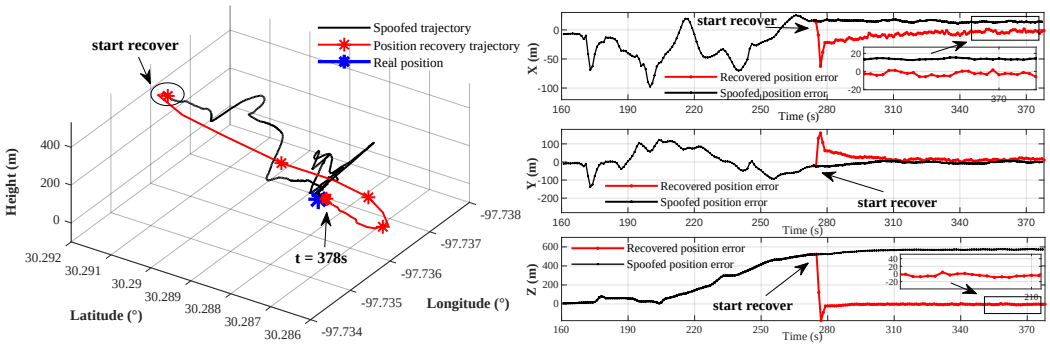
Figure 14. Three-dimensional trajectory and position error of spoofed and recovered navigation solutions from 160 s to 296 s.

Figure 14 shows that when the proposed spoofing countermeasure is applied before the spoofing attack begins, the recovered position error (red line) overlaps with the spoofed position error in the early stages before detection, as compared to Figure 8; afterwards, the proposed sensitive AIHT algorithm enables early detection and recovery, stabilizing the receiver's trajectory and preventing significant deviations in the position results. Figures 15 and 16 display the trajectories and position errors from 245 s to 343 s when $\Delta\tau_{\mathrm{ij}} \leq 1$ and from 275 s to 378 s when $\Delta\tau_{\mathrm{ij}} > 1$, respectively. These figures demonstrate that significant position resolution deviations are corrected by the position recovery algorithm, effectively realigning the position to the true values in different time periods.



(**a**) Position recovery trajectory (**b**) Position error

Figure 15. Three-dimensional trajectory and position error of spoofed and recovered navigation solutions from 245 s to 343 s.



(**a**) Position recovery trajectory (**b**) Position error

Figure 16. Three-dimensional trajectory and position error of spoofed and recovered navigation solutions from 275 s to 378 s.

To evaluate the stability of our position recovery algorithm, we compute the RMSE over different time periods in both static (scenario 4) and dynamic (scenario 6) environments. Table 1 lists the RMSE values in the X, Y, and Z dimensions of the ECEF coordinate system during various phases of recovery. As shown in Figures 14b–16b, the algorithm initially shows fluctuations for the first 80 s after deployment, which then stabilize until the end. The RMSEs for the time periods before and after 80 s from deployment are calculated and presented in Table 1. Scenario 6 shows a larger RMSE than scenario 4 during the first 80 s because the spoofing attack shifts the receiver farther away; in later stages, the RMSE decreases but remains slightly higher than in scenario 4, which is due to the stronger spoofing power causing the receiver to track the spoofed signals more persistently. We further analyze performance across different spoofing delays $\Delta\tau_{ij}$. As indicated in Table 1, our method performs effectively for both $\Delta\tau_{ij} \leq 1$ and $\Delta\tau_{ij} > 1$, with satisfyingly low RMSE during the stabilized recovery stages from 80 s to the end of the recovery algorithm. It is worth mention that the higher Y-axis RMSE under $\Delta\tau_{ij} > 1$ in scenario 4 results from unstable code phase estimates during the spoofing-dominant phase. This instability is primarily caused by loop noise, which particularly affects satellites contributing along the Y-axis.

Table 1. RMSE (m) under our position recovery algorithm across different time periods after deployment. The values represent the RMSE for two different scenarios: scenario 4 (first value) and scenario 6 (second value).

$\Delta\tau_{ij}$ (chip)	Direction	RMSE (m)	
		0 s–80 s	80 s–end
non-spoofing	X	14.4, 159.1	4.9, 9.3
	Y	25.6, 18.1	7.0, 8.2
	Z	17.5, 49.8	6.1, 8.3
$\Delta\tau_{ij} \leq 1$	X	37.8, 349.4	3.7, 9.4
	Y	33.4, 76.6	5.3, 10.1
	Z	71.6, 222.4	6.4, 9.8
$\Delta\tau_{ij} > 1$	X	13.1, 527.1	3.2, 7.4
	Y	34.7, 82.3	14.1, 11.9
	Z	62.4, 187.8	6.2, 10.1

5. Discussion

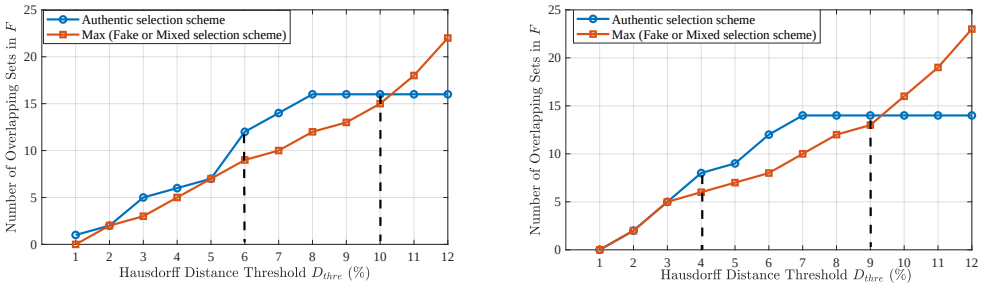
The two main objectives of this article are to achieve accurate detection of spoofing interference and to ensure stable position recovery results under different spoofing conditions. To achieve the first objective, as discussed in Section 4.1, spoofing detection accuracy is evaluated by comparing the RMSE between $\boldsymbol{y}$ and the $\hat{\boldsymbol{y}}$ reconstructed by the proposed AIHT-based detection method and other algorithms, for which we use publicly available datasets across several different scenarios. For the second objective, as outlined in Section 4.2, the effectiveness of the position recovery algorithm is indirectly evaluated by analyzing the tracking performance of each channel pair during the classification phase. Because smaller DLL discriminator fluctuations indicate a more stable AIHT-based APT algorithm, Figures 10–13 show that the discriminator quickly regains stability during loop correction, validating the potential for successful position recovery.

The experiments in Section 4.3 visually demonstrate the effectiveness and stability of the proposed position recovery algorithm. Figures 14–16 and Table 1 show that the proposed spoofing detection, classification, and position recovery algorithm effectively mitigates the impact of spoofing interference across various scenarios both before and during spoofing attacks on the receiver while maintaining stable recovery results.

To validate the sensitivity of D_{thre}, we vary its value and check whether the proposed positioning recovery method still correctly selects the authentic scheme with the maximum number of overlapping sets in matrix F. For Scenario 4, D_{thre} values between 6% and 10% are effective, as shown in Figure 17a,b. When D_{thre} exceeds 10%, the number of overlaps from the authentic selection scheme no longer increases, as all relevant sets are already included; meanwhile, overlaps from fake or mixed schemes continue to grow. Therefore, if D_{thre} becomes too large, the recovery algorithm may mistakenly treats spoofed channels as authentic, leading to positioning errors. Similarly, for scenario 6, D_{thre} values between 4% and 9% are effective. Based on both cases, we set $D_{thre} = 8\%$ in our implementation.

Compared to prior approaches such as [5,20], our method introduces fewer assumptions and demonstrates better adaptability under challenging spoofing conditions. The CADLL-based method in [5] requires a stable initialization from a preceding MEDLL stage. If spoofing has already started before the receiver begins tracking, the loop fails to converge and the system may diverge. The MLE-based estimator in [20] assumes that the spoofing delay exceeds 0.75 chips. When this condition is not met, the estimation matrix becomes ill-conditioned, often resulting in severe estimation errors. In contrast, our method neither depends on prior authentic tracking states nor requires a minimum

spoofing delay. Through sparse decomposition and iterative recovery, it can resolve closely spaced correlation peaks and remain effective even when the spoofing delay is small.



(**a**) Scenario 4 (**b**) Scenario 6

Figure 17. Overlapping set counts under varying normalized D_{thre} values for the two different scenarios.

Importantly, we acknowledge a critical limitation of the proposed framework in that it requires at least five visible satellites (i.e., $p > 4$) in order to construct a valid position recovery scheme. In urban canyon environments, this condition may not always be met. To handle such scenarios, a promising approach is to integrate signals from other systems, such as 5G localization or Low Earth Orbit (LEO)-based navigation. A recent study [15] has shown that 5G signals can enhance spoofing resistance when GNSS signals are degraded. Moreover, we are developing a custom-designed navigation signal for LEO dedicated positioning [2] to ensure robust performance under limited satellite visibility. Hybrid GNSS-LEO based anti-spoofing strategies will be explored in our future work.

6. Conclusions

This paper has investigated the issue of reliable and secure navigation for GNSS receivers subject to spoofing attacks. We propose a sparse decomposition algorithm with non-negative constraints limited by received signal power magnitudes, which not only achieve accurate spoofing detection but also extracts key features of the received signal's contributing components. During the spoofing classification process, these features are utilized to continuously refine receiver's code tracking loop in order to resist drag-off by the spoofed signal, ensuring that the contributing components of each spoofed satellite are tracked separately within channel pairs. Moreover, leveraging the inherent inconsistency of spoofing properties, we incorporate the Hausdorff distance to identify the most overlapped position sets, enabling the determination of genuine position trajectories and effectively mitigating the impacts of spoofing. The key advantage of our anti-spoofing framework is its ability to continuously track all contributing components of received signals and further exploit the inherent inconsistency of spoofing signals, which eliminates the need for extra devices or specific initial conditions. The efficiency and advantages of the proposed anti-spoofing framework are fully illustrated through extensive experimental studies conducted on the public TEXBAT dataset.

Author Contributions: Conceptualization, Y.H.; formal analysis, B.X.; methodology, Y.H.; project administration, X.Z.; resources, X.Z.; software, Y.H.; supervision, X.Z.; validation, B.X.; writing—original draft, Y.H. and X.Z.; writing—review and editing, B.X. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Open Fund from the State Key Laboratory of Satellite Navigation System and Equipment Technology (Grant No. CEPNT2022A05).

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

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

References

1. Kaplan, E.D.; Hegarty, C. *Understanding GPS/GNSS: Principles and Applications*; Artech House: Norwood, MA, USA, 2017.
2. He, Y.; Zhuang, X.; Hou, Y.; Wu, L. Robust blind space-time adaptive processing for measurement error mitigation in GNSS receivers. *IET Commun.* **2023**, *17*, 1021–1036. [CrossRef]
3. Bhatti, J.; Humphreys, T.E. Hostile control of ships via false GPS signals: Demonstration and detection. *Navig. J. Inst. Navig.* **2017**, *64*, 51–66. [CrossRef]
4. Guo, Y.; Wu, M.; Tang, K.; Tie, J.; Li, X. Covert spoofing algorithm of UAV based on GPS/INS-integrated navigation. *IEEE Trans. Veh. Technol.* **2019**, *68*, 6557–6564. [CrossRef]
5. Wang, Y.; Kou, Y.; Huang, Z.; Zhao, Y. GNSS spoofing maximum-likelihood estimation switching between MEDLL and CADLL. *GPS Solut.* **2023**, *27*, 148. [CrossRef]
6. Psiaki, M.L.; Humphreys, T.E. GNSS spoofing and detection. *Proc. IEEE* **2016**, *104*, 1258–1270. [CrossRef]
7. Zhou, Z.; Li, H.; Chen, Z.; Lu, M. Velocity Consistency Checking based GNSS Spoofing Detection Method for Vehicles. *IEEE Trans. Veh. Technol.* **2023**. [CrossRef]
8. Fang, J.; Yue, J.; Xu, B.; Hsu, L.T. A post-correlation graphical way for continuous GNSS spoofing detection. *Measurement* **2023**, *216*, 112974. [CrossRef]
9. Sun, C.; Cheong, J.W.; Dempster, A.G.; Zhao, H.; Bai, L.; Feng, W. Robust spoofing detection for GNSS instrumentation using Q-channel signal quality monitoring metric. *IEEE Trans. Instrum. Meas.* **2021**, *70*, 8504115. [CrossRef]
10. Schmidt, E.; Gatsis, N.; Akopian, D. A GPS spoofing detection and classification correlator-based technique using the LASSO. *IEEE Trans. Aerosp. Electron. Syst.* **2020**, *56*, 4224–4237. [CrossRef]
11. Wang, H.; Li, H.; Zhong, M.; Lu, M. A Space-Time-Ambiguity Decomposition Method for DOA Estimation Enhancing Anti-Spoofing Via Rotating Dual Antennas. *IEEE Trans. Aerosp. Electron. Syst.* **2024**, *60*, 7643–7662. [CrossRef]
12. He, L.; Li, H.; Lu, M. Dual-antenna GNSS spoofing detection method based on Doppler frequency difference of arrival. *Gps Solut.* **2019**, *23*, 78. [CrossRef]
13. Shang, X.; Sun, F.; Liu, B.; Zhang, L.; Cui, J. GNSS Spoofing Mitigation With a Multicorrelator Estimator in the Tightly Coupled INS/GNSS Integration. *IEEE Trans. Instrum. Meas.* **2022**, *72*, 2529013. [CrossRef]
14. Kujur, B.; Khanafseh, S.; Pervan, B. Optimal INS Monitor for GNSS Spoofer Tracking Error Detection. *Navig. J. Inst. Navig.* **2024**, *71*, navi.629. [CrossRef]
15. Bai, L.; Sun, C.; Dempster, A.G.; Zhao, H.; Feng, W. GNSS Spoofing Detection and Mitigation With a Single 5G Base Station Aiding. *IEEE Trans. Aerosp. Electron. Syst.* **2024**, *60*, 4601–4620. [CrossRef]
16. Kujur, B.; Khanafseh, S.; Pervan, B. Detecting GNSS spoofing of ADS-B equipped aircraft using INS. In Proceedings of the 2020 IEEE/ION Position, Location and Navigation Symposium (PLANS), Portland, OR, USA, 20–23 April 2020; pp. 548–554. [CrossRef]
17. Kuusniemi, H.; Blanch, J.; Chen, Y.H.; Lo, S.; Innac, A.; Ferrara, G.; Honkala, S.; Bhuiyan, M.Z.H.; Thombre, S.; Söderholm, S.; et al. Feasibility of fault exclusion related to advanced RAIM for GNSS spoofing detection. In Proceedings of the 30th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2017), Portland, OR, USA, 25–29 September 2017; pp. 2359–2370.
18. Zhou, W.; Lv, Z.; Wu, W.; Shang, X.; Ke, Y. Anti-spoofing technique based on vector tracking loop. *IEEE Trans. Instrum. Meas.* **2023**, *72*, 8504516. [CrossRef]
19. Xu, B.; Jia, Q.; Hsu, L.T. Vector tracking loop-based GNSS NLOS detection and correction: Algorithm design and performance analysis. *IEEE Trans. Instrum. Meas.* **2019**, *69*, 4604–4619. [CrossRef]
20. Shang, X.; Sun, F.; Zhang, L.; Cui, J.; Zhang, Y. Detection and mitigation of GNSS spoofing via the pseudorange difference between epochs in a multicorrelator receiver. *Gps Solut.* **2022**, *26*, 37. [CrossRef]
21. Humphreys, T.E.; Bhatti, J.A.; Shepard, D.; Wesson, K. *The Texas Spoofing Test Battery: Toward a Standard for Evaluating GPS Signal Authentication Techniques*; The University of Texas: Austin, TX, USA, 2012.
22. Mohimani, H.; Babaie-Zadeh, M.; Jutten, C. A fast approach for overcomplete sparse decomposition based on smoothed $\mathcal{L}_0$ norm. *IEEE Trans. Signal Process.* **2008**, *57*, 289–301. [CrossRef]
23. Blanchard, J.D.; Cermak, M.; Hanle, D.; Jing, Y. Greedy algorithms for joint sparse recovery. *IEEE Trans. Signal Process.* **2014**, *62*, 1694–1704. [CrossRef]
24. Foucart, S.; Lecué, G. An IHT algorithm for sparse recovery from subexponential measurements. *IEEE Signal Process. Lett.* **2017**, *24*, 1280–1283. [CrossRef]
25. Blumensath, T.; Davies, M.E. Iterative hard thresholding for compressed sensing. *Appl. Comput. Harmon. Anal.* **2009**, *27*, 265–274. [CrossRef]

26. Huttenlocher, D.P.; Klanderman, G.A.; Rucklidge, W.J. Comparing images using the Hausdorff distance. *IEEE Trans. Pattern Anal. Mach. Intell.* **1993**, *15*, 850–863. [CrossRef]
27. Humphreys, T. *TEXBAT Data Sets 7 and 8*; The University of Texas: Austin, TX, USA, 2016.
28. Söderholm, S.; Bhuiyan, M.Z.H.; Thombre, S.; Ruotsalainen, L.; Kuusniemi, H. A multi-GNSS software-defined receiver: Design, implementation, and performance benefits. *Ann. Telecommun.* **2016**, *71*, 399–410. [CrossRef]