GNSS Spoofing Detection Based on Opportunistic Position Information

Wenjie Liu, *Graduate Student Member, IEEE,* and Panos Papadimitratos, *Fellow, IEEE*

***Abstract*—The limited or no protection for civilian Global Navigation Satellite System (GNSS) signals makes spoofing attacks relatively easy. With modern mobile devices often featuring network interfaces, state-of-the-art signals of opportunity (SOP) schemes can provide accurate network positions in replacement of GNSS. The use of onboard inertial sensors can also assist in the absence of GNSS, possibly in the presence of jammers. The combination of SOP and inertial sensors has received limited attention, yet it shows strong results on fully custom-built platforms. We do not seek to improve such special-purpose schemes. Rather, we focus on countering GNSS attacks, notably detecting them, with emphasis on deployment with consumer-grade platforms, notably smartphones, that provide *off-the-shelf opportunistic information* (i.e., network position and inertial sensor data). Our Position-based Attack Detection Scheme (PADS) is a probabilistic framework that uses regression and uncertainty analysis for positions. The regression optimization problem is a weighted mean square error of polynomial fitting, with constraints that the fitted positions satisfy the device velocity and acceleration. Then, uncertainty is modeled by a Gaussian process, which provides more flexibility to analyze how sure or unsure we are about position estimations. In the detection process, we combine all uncertainty information with the position estimations into a fused test statistic, which is the input utilized by an anomaly detector based on outlier ensembles. The evaluation shows that the PADS outperforms a set of baseline methods that rely on SOP or inertial sensor-based or statistical tests, achieving up to 3 times the true positive rate at a low false positive rate.**

***Index Terms*—GNSS attack detection, secure localization, opportunistic information**

I. Introduction

Global Navigation Satellite Systems (GNSSs) face a wide range of attack threats, with spoofing being particularly concerning, as it allows adversaries to manipulate the GNSS position and time. Real-world incidents, such as disrupting sensor fusion algorithms to cause crashes in autonomous vehicles [1] and misnavigation of luxury yachts [2], highlight the increasing threat of GNSS attacks [3]. The rising sophistication and accessibility of GNSS spoofing technology further intensifies these concerns [4]. In response, various standalone strategies have been proposed to counter and identify attacks, ranging from the implementation of cryptographic protocols [5]–[7] to signal-level mechanisms [8]–[11]. Cryptographic solutions require extensive updates to satellites and receivers, which are cost-prohibitive and hard to implement at scale. Signal-level solutions, relying on characteristics such as angle of arrival (AoA), may require specialized hardware (e.g., antenna arrays) and may be ineffective in challenging environments, such as urban canyons with significant multipath [8], [9]. Moreover, many consumer-grade devices do not integrate GNSS receiver with strong anti-jamming/spoofing features or do not expose the necessary low-level signal data (e.g., phase, correlations) through standard operating system APIs [12].

Modern consumer-grade mobile platforms, such as smartphones and autonomous vehicles, offer a promising opportunity for GNSS spoofing detection. These devices are equipped with opportunistic information sources, including connectivity (i.e., Wi-Fi, cellular, Bluetooth signals) and onboard sensors such as inertial measurement units (IMUs). This combination of data sources enables robust attack detection by cross-checking GNSS positions with alternative positioning methods. However, leveraging these data sources for spoofing detection is nontrivial due to their inherent limitations: network-based positions often have larger fluctuation errors than GNSS and IMU, while IMU-based systems suffer from cumulative error [1], [13], [14]. Therefore, effectively integrating these information sources into a unified framework for spoofing detection remains an ongoing topic that requires further investigation.

An increasingly explored approach involves leveraging signals of opportunity (SOP) and IMU for GNSS-denied environments or spoofing detection [15]–[23]. For instance, [20], [23] utilize SOP, such as Wi-Fi or cellular signals, to localize devices or detect deviations from GNSS-provided positions, typically employing threshold-based binary decision models. [15], [16] incorporate IMU data for trajectory analysis to identify inconsistencies indicative of spoofing. [17]–[19], [22] use both SOP and IMU with sophisticated custom hardware platforms to present an accurate alternative positioning result, although they do not look for the detection per se. While these methods have shown promise for localization in GNSS-denied environments, they are not readily applicable on consumer-grade platforms, e.g., smartphones. Hence, we direct efforts towards the detection of GNSS attacks with the use of all available inputs (network positions and onboard sensor data) and a hardware-agnostic design.

We propose a *Position-based Attack Detection Scheme (PADS)* that integrates network-based positioning results and velocity, acceleration, and orientation from the onboard sensors, termed *off-the-shelf opportunistic information*, within a

W. Liu and P. Papadimitratos are with the Networked Systems Security Group, KTH Royal Institute of Technology, 114 28 Stockholm, Sweden.

Corresponding author: Wenjie Liu (e-mail: *wenjieli@kth.se*).

This work was supported in part by the SSF SURPRISE cybersecurity project, the Security Link strategic research center, and the China Scholarship Council. The computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council through grant agreement no. 2022-06725. We would also like to acknowledge the work of the organizers of Jammertest 2024.

probabilistic detection framework for widely-used consumer-grade mobile devices. The key idea involves accounting for various types of noisy position sources with different update rates, the movement of the GNSS receiver, and designing a test statistic. The construction of the test statistic involves two steps, which we refer to as a combo of model-based and data-driven techniques. First, we establish a closed-form solution to describe the relationship between motion and position data, resulting in a motion-constrained regression problem. This connects short-term estimation via the receiver's motion with long-term estimation through network-based positions, effectively smoothing positions with motion model constraints. Second, we employ Gaussian process regression [24] to model the uncertainty inherent in the smoothed positions. Then, we calculate a weighted sum of both positions and uncertainties into a unified Gaussian function for the ensemble-based anomaly detection [25], [26]. Crucially, PADS is designed as a software-based detection layer that utilizes opportunistic information commonly accessible in modern platforms, making it deployable without requiring specialized hardware or low-level signal access.

Building on our earlier work [27], we incorporate a learning-based detector into the decision-making part. Furthermore, we apply our PADS to an existing dataset [1] in enhanced network simulations of terrestrial infrastructures with real-world Wi-Fi and cellular layouts, and a newly collected dataset from consumer-grade Android phones under real GNSS attacks. Additionally, we present improved theoretical and experimental analyses to further assess the effectiveness of the scheme.

The main contribution of this work is a **probabilistic framework for GNSS attack detection**: We combine network-based positions and onboard sensors within a probabilistic framework for GNSS position attack detection. Our *Position-based Attack Detection Scheme (PADS)* can fuse position, velocity, acceleration, and orientation data from various sources with different accuracies and update rates. It provides a robust and interpretable detection outcome, as well as a recovered position using benign GNSS, networks, and onboard sensors.

In PADS, we contribute a formulation of a **motion-constrained regression problem for position smoothing** that combines short-term trajectory smoothing via IMU with long-term stabilization from network-based positions. This fusion reduces position noise and prevents IMU drift. A Gaussian process regression is then employed to quantify positional uncertainty. They are computationally efficient, with polynomial-time complexity, and mathematical proofs are presented. In addition, we design **anomaly detection via a weighted test statistic** that incorporates position trajectories and their uncertainties, enabling unsupervised anomaly detection. This hyperparameter-free approach uses ensemble methods to detect spoofing attacks and is also extensible and compatible with signal-level detection methods by incorporating signal properties, such as Doppler shift.

We also contribute a **comprehensive evaluation on consumer-grade platforms** for GNSS attack detection. First, we evaluate with the help of a simulated autonomous driving platform. Second, we experiment with various Android smartphones, including different brands, prices, and chips (from Exynos, MTK, Qualcomm, and Google Tensor). GNSS attack strategies include a variety of meaconing and spoofing, incurring gradual deviation or position jumping, notably in a real-world setting, with data collected in Jammertest 2024.

The rest of the paper is organized as follows: Sec. II provides background and reviews related work on GNSS attacks, detection, and network-based positioning. Sec. III presents our system model and adversary. Sec. IV and V detail the problem formulation and the proposed PADS. Sec. VI discusses evaluation and comparison with baselines. Finally, Sec. VII concludes the paper.

II. Related Work and Background

A. GNSS Attack and Detection

GNSS spoofing attacks typically craft fraudulent signals with precise power and format as per GNSS protocols. Before spoofing, the attacker may first employ jamming to deliberately disrupt GNSS signals, causing the victim to lose the GNSS signal lock [19]. Alternatively, with more sophisticated strategies, the attacker may gradually amplify the spoofing signal, eventually tricking the victim to follow it [1]. Meaconing, the easiest method of spoofing signal generation, involves retransmitting rightful satellite signals from a different area. When it comes to authenticated GNSS signals, relay or replay attacks [28] can transmit satellite signals using low-complexity setups to deceive the victim into trusting the information. Another more sophisticated modification, known as selective delay [8], rebroadcasts separate satellite signals, allowing for modification of the position solution according to the attack scenario. Distance-decreasing (DD) attacks [29] provide additional options for adversaries, employing Early Detection and Late Commit to relay the GNSS signal, thereby making the relayed one appear to arrive earlier than it would have.

Standalone detections often focus on analyzing the physical characteristics of GNSS signals, e.g., Doppler effect, AoA, signal-to-noise ratio (SNR), and received signal strength (RSS) [8], [9], [11], [30], [31]. Recent advances include leveraging signal quality monitoring [32] or machine learning on signal features [33]. These methods can be effective against attacks that cause signal distortions, but mostly can not provide an alternative positioning. Moreover, access to the necessary low-level measurements (e.g., precise phase, AoA) is often limited by hardware and standard APIs [12]. For example, AoA typically requires specialized multi-antenna hardware. Furthermore, multipath in urban areas can also cause signal anomalies. Sophisticated attackers can also employ strategies such as slowly varying spoofing [1], [14] or imitating AoA that minimizes abrupt changes, making detection based solely on signals difficult.

Increasingly, low-end GNSS receivers are integrated into mobile platforms that also feature onboard sensors and network interfaces. This presents opportunities for leveraging the sensors and networks to enhance attack detection. For instance, [34] focuses on unmanned aerial vehicles (UAVs) to fuse GNSS with IMU data. Then, they use relative distance information obtained from RSS data to detect spoofing, with alternative positions based on multi-agent systems enhancing

navigation robustness under spoofing conditions. SOP from terrestrial network infrastructures can assess GNSS-provided positions [20], [35]. This involves assuming adequate network scanning and the availability of base station (BS) or access point (AP) positions, using RSS or time-of-arrival (TOA) as the distance measure between the mobile platform and the station to estimate position for checking GNSS. Additionally, in [15], an extended Kalman filter (EKF) integrates GNSS and IMU data, with Receiver Autonomous Integrity Monitoring (RAIM) to assist with spoofing detection. Combined metric-based approach [36] integrates multiple detection features such as autocorrelation distortion, RSS, pseudorange, carrier phase difference, and AoA to enhance detection.

Without considering the detection of GNSS attack, [17]–[19], [22] fuse SOP with IMU to provide position and navigation with great accuracy in GNSS-denied environments. The work considers and experiments with different grades of IMUs, types of devices, cellular clock errors, pseudorange measurement models, unknown transmitter locations, etc. However, most of these APIs or information are still unavailable on consumer-grade platforms, e.g., smartphones.

B. Network-based Positioning

In addition to the conventional dependence on GNSS for positioning, the network infrastructures, such as Wi-Fi, cellular, Bluetooth, and eLoran [37], can also provide alternatives or backups for accurate localization. They play an important role in scenarios where GNSS signals may be limited or unavailable [19]. Our focus is not to incorporate these positioning techniques into our framework, but to use off-the-shelf network-based positions to enhance the robustness and reliability of the detection process.

Fingerprinting methods [38]–[40] are commonly used where databases of pre-collected fingerprints (RSS, magnetic field values, channel state information, or even visual information) are compiled. After that, deterministic or probabilistic fingerprint-matching algorithms are used for localization. They provide supplementary information for positioning or offering validations of GNSS attacks. However, fingerprint database construction is time-consuming, especially for wide-open outdoor environments. Hence, fingerprint-based positioning is often limited to a small area.

Range-based methods [41], [42] make use of various inputs such as RSS, propagation time, or AoA to derive pseudo-ranges, which are then utilized for multilateration. Recent advances in network-GNSS hybrid positioning [43], [44] rely on the ranging information (e.g., TOA of 5G millimeter-wave) or localization results to provide additional observations in the EKF framework.

III. System Model

We consider a mobile GNSS-enabled platform that provides computational power, heterogeneous network infrastructures and diverse sensors. At time t, the actual platform position, denoted as $\mathbf{p}_c(t) \in \mathbb{R}^2$, needs to be estimated based on positioning. $\mathbf{p}_0(t)$ represents the GNSS position at time t. Wi-Fi and/or cellular networks can provide positions, $\mathbf{p}_m(t)$,

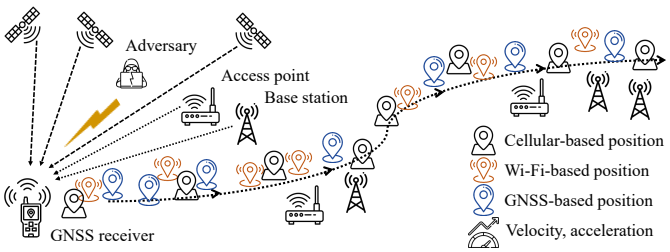

Fig. 1. Illustration of position information from GNSS and network infrastructures, motion information from onboard sensors, and external adversary.

based on network-based positioning algorithms (e.g., [42], [45], [46]), where $m = 1, 2, ..., M$ and M signifies the number of network interfaces; t spans from 1 to N seconds, with N as the total number of time indexes. IMUs provide multi-axis acceleration measurements, while velocity is possibly obtained from wheel speed sensors (e.g., in autonomous vehicles), denoted as velocity, $\mathbf{v}(t)$, acceleration, $\mathbf{a}(t)$, and orientation, $\boldsymbol{\omega}(t)$. Continuous network connectivity is not required, as some infrastructure might be temporarily inaccessible for reasons independent of the mobile device itself. We assume that positioning errors of benign GNSS, Wi-Fi, and cellular networks are zero-mean Gaussian random variables [47]–[49]. As the mobile platform moves along a path in benign conditions (as illustrated in Fig. 1), the GNSS-derived position aligns with the opportunistic position information.

Adversary: Spoofed, relayed, or replayed GNSS satellite signals manipulate the mobile platform into falsely estimating the position. We assume that the attacker can observe the victim position and has access to software-defined radios (SDRs) equipped with GNSS spoofing capabilities to falsify the signals. As GNSS signals are low power, the attack can compel the victim to lose the lock on legitimate signals and then acquire the lock on adversarial signals. We remain neutral regarding the exact details of the attack, and do not restrict the type of attacker as long as it achieves the malicious alteration of position. In other words, as the GNSS solution includes position and time, detecting alterations solely in time, without corresponding changes in position, falls beyond the scope of our investigation. The attacker can skillfully design the victim's spoofed positions, considering its actual path. Certain trajectory designs, such as path drift [50] and gradual deviation [1], remain almost undetectable for a period following the initiation of the attack (e.g., Kalman filter-based detection).

The attacker considered here operates exclusively within the GNSS realm. We assume the opportunistic information sources (network-based position obtained via platform services, onboard sensor data) remain unaffected by the GNSS spoofing attack itself. This assumption is grounded in the practical separation of these systems (e.g., network signals are often authenticated or encrypted, and IMU sensors cannot be affected unless the device itself is compromised). Therefore, the scope of this work focuses on detecting GNSS attacks using trustworthy opportunistic data. Wi-Fi spoofing, cellular base station simulation, and physical sensor manipulation are considered beyond this work. However, we assume that the adversary can extend periods of unavailability for network-

based positioning by interfering with wireless networks. Furthermore, it is assumed that the attacker does not exert physical control over the victim, thereby preventing manipulation of the process for deriving position information from various network interfaces and onboard sensors. In short, the adversarial actions are confined to GNSS spoofing, while wireless networks may experience interference. As a result, during a spoofing attack, the GNSS-derived position should diverge from the actual position and not match the opportunistic position information.

IV. Problem Formulation

Our objective is to assess the consistency between the GNSS-derived position and opportunistic position information from $\{\mathbf{p}_m(t), \mathbf{v}(t), \mathbf{a}(t), \boldsymbol{\omega}(t)\}$ to determine whether the current GNSS position is indicative of an attack. By evaluating the probability of a GNSS position attack, we seek to maximize the true positive rate of detection. Additionally, we aim for a detection scheme that remains reliable even if certain types of opportunistic information are unavailable.

For the detection of GNSS position attacks at a given time t and with M network interfaces, we use data $\{\mathbf{p}_m(i), \mathbf{v}(i), \mathbf{a}(i), \boldsymbol{\omega}(i)\}$ for $0 < i < t$ and $m \in \{0, 1, ..., M\}$ to determine whether $\mathbf{p}_0(t)$ is subject to an attack. Two corresponding hypotheses are presented as follows:

- $\mathcal{H}_0$: $\mathbf{p}_0(t)$ is not under attack;
- $\mathcal{H}_1$: $\mathbf{p}_0(t)$ is subject to attack.

Then, the decision made at the time t is denoted as $\hat{\mathcal{H}}(t) \in \{\mathcal{H}_0, \mathcal{H}_1\}$. The true positive is expressed as $\hat{\mathcal{H}}(t) = \mathcal{H}_1$ under attack ($\mathcal{H}_1$), and the false positive is $\hat{\mathcal{H}}(t) = \mathcal{H}_1$ under $\mathcal{H}_0$. The true positive rate for $0 < t \leq N$ is

$$R_{\text{TP}}(\hat{\mathcal{H}}(t)) = \mathbb{P}[\hat{\mathcal{H}}(t) = \mathcal{H}_1 | \mathcal{H}_1]. \tag{1}$$

The false positive rate is

$$R_{\text{FP}}(\hat{\mathcal{H}}(t)) = \mathbb{P}[\hat{\mathcal{H}}(t) = \mathcal{H}_1 | \mathcal{H}_0]. \tag{2}$$

We define the *detection time delay*, ΔT, as the interval between the moment the alarm is raised and the start of the attack:

$$\Delta T = \min\left\{t \;\middle|\; \mathbb{I}\{\hat{\mathcal{H}}(t) = \mathcal{H}_1 | \mathcal{H}_1\} = 1\right\} - \min\left\{t \;\middle|\; \mathbb{I}\{\hat{\mathcal{H}}(t) = \mathcal{H}_0 | \mathcal{H}_1\} = 1\right\} \tag{3}$$

where indicator function $\mathbb{I}\{A|B\}$ equals to 1 when condition A is satisfied given condition B.

The problem is to: (a) maximize R_{TP} when fixing R_{FP}, (b) minimize ΔT, and (c) provide a probability of being under GNSS position attack, along with a recovered position that fuses opportunistic information to replace GNSS position when $\hat{\mathcal{H}}(t) = \mathcal{H}_1$.

V. Proposed Scheme

As Fig. 2 shows, PADS detects attacks on GNSS positions by using information about network positions and movements, i.e., velocity, acceleration, and orientation. The input data is from GNSS, Wi-Fi, cellular, and onboard sensors. A rolling window takes a fixed-length series of positions, which will be interpolated to generate a smoothed trajectory. Subsequently, confidence intervals for these $M+1$ position series are calculated. The confidence intervals construct a fused test statistic to determine whether the current GNSS position reflects an attack, and if so, trigger an alarm. The overall process is shown as Algorithm 1.

Rolling Window (①) collects real-time positions of the platform from GNSS, Wi-Fi, and cellular sources (with $M+1$ categories available, and we consider $M = 2$ here), in addition to velocity, acceleration, and orientation data from onboard sensors. These positions are organized in order of their timestamps. As a result, the filters implement rolling window techniques, taking fixed-length data at each detection time t rather than utilizing the entire series.

Confidence Interval (②) is constructed by combining both motion and statistical models. The motion part (Sec. V-B1) uses a local polynomial regression algorithm with movement constraints to fit the position data. It exploits both the short- and long-term characteristics of data: while onboard sensors provide high short-term accuracy, they are unable to maintain stable positional accuracy over time; to overcome this limitation, we integrate their data with less frequent but periodic positioning updates from terrestrial networks. We use regression to fit the positions and minimize the fitting error while adhering to movement constraints, ensuring the resulting fit conforms to velocity and acceleration. The statistical part (Sec. V-B2) is a Gaussian process, focusing on confidence intervals represented as a probability distribution, indicating the uncertainty of positions. We assume that the benign position series of $\mathbf{p}_m(t)$ follows a Gaussian process, with a mean already determined through the motion part. To compute the variance, Gaussian process regression uses a predefined covariance function and differences from fitted results. This allows for a better idea of the variability in position data.

Decision-Making (③) builds a test statistic by using the mean and variance of confidence intervals. Then, anomaly detection relies on this test statistic across multiple sources and over the time domain. To integrate the confidence intervals in the time domain, we calculate a weighted sum over time in Sec. V-C1, with the weights normalized to ensure their summation to 1. The weighted sum is still Gaussian, with its mean and variance determined as a linear combination of the means and variances of the individual confidence intervals. To fuse the GNSS-derived position with M opportunistic position sources, we multiply $M+1$ distributions for time t. Then, we create a fused test statistic based on it and apply an anomaly detector (Loda [25]) in Sec. V-C2.

A. Rolling Window for Detection

We combine screening and detection: instead of analyzing the entire time series at each detection step for every time slot t, we select a fixed-size series by implementing a rolling window mechanism with a specific window size. This ensures that only recent network-based positions and motion data are used for evaluating potential attacks on the current GNSS.

1) Coordinate Format: Coordinates $\mathbf{p}_m(t) \in \mathbb{R}^2, m = 0, 1, ..., M$ are formatted according to the World Geodetic System (WGS) Latitude, Longitude, Altitude standard. Data from

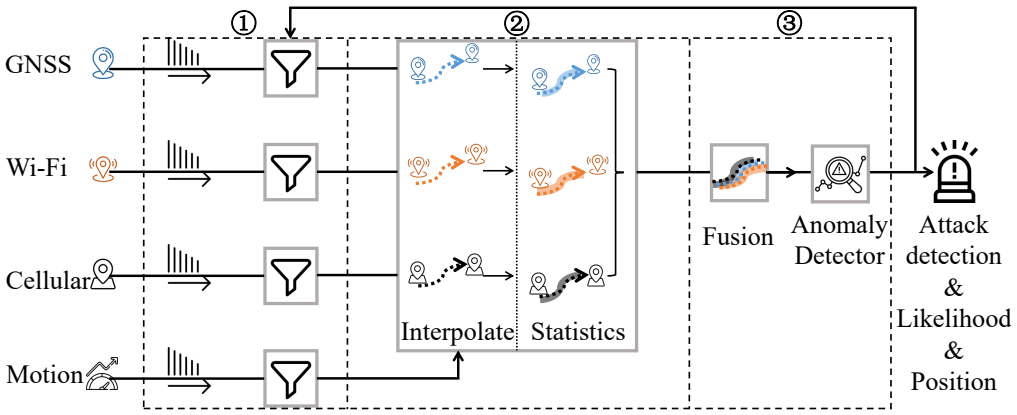

Fig. 2. PADS overview. Inputs: GNSS position ($\mathbf{p}_0$), network positions ($\mathbf{p}_{1..M}$), and motion data ($\mathbf{v}, \mathbf{a}, \boldsymbol{\omega}$). Components: Rolling Window (①) selects recent data; Confidence Interval (②) estimates smoothed position $\hat{\mathbf{p}}_m$ and uncertainty $\hat{\boldsymbol{\Sigma}}_m$ for each source m using motion-constrained regression and Gaussian processes; Decision-Making (③) fuses these intervals into a test statistic ($\Lambda_{1:M}$), computes an anomaly score ($f_{\mu,\sigma}$), makes a detection decision ($\hat{\mathcal{H}}$), and provides a recovered position (μ). Outputs: $\hat{\mathcal{H}}, f_{\mu,\sigma}, \mu$ for each time t. It fuses information across multiple (available) sources and over the time domain.

Algorithm 1 PADS overall operation with two alternative positioning sources and onboard sensors

Input $\mathbf{p}_0(t), \mathbf{p}_1(t), \mathbf{p}_2(t), \mathbf{v}(t), \mathbf{a}(t), \boldsymbol{\omega}(t)$
Parameter w
Output $\hat{\mathcal{H}}(t), f_{\mu,\sigma}(t), \mu(t)$

initialize S ▷ Sequence of positions and motion data
$t \leftarrow 0$
while $t < N$ **do**
  $t \leftarrow t + 1$
  ensure $length(S) = w$ ▷ Rolling window
  $CI \triangleq \{\mathcal{N}(\hat{\mathbf{p}}_m(t), \hat{\boldsymbol{\Sigma}}_m(t))\}_{m=0..M} \leftarrow$ Algorithm 2 ▷ Construct confidence intervals
  $\hat{\mathcal{H}}(t), f_{\mu,\sigma}(t), \mu(t) \leftarrow$ Algorithm 3 ▷ Detect attack using confidence intervals
  if $\hat{\mathcal{H}}(t) = \mathcal{H}_1$ **then**
    $S \leftarrow \{S;\ \mathbf{p}_1(t), \mathbf{p}_2(t), \mathbf{v}(t), \mathbf{a}(t), \boldsymbol{\omega}(t)\}$
  else
    $S \leftarrow \{S;\ \mathbf{p}_0(t), \mathbf{p}_1(t), \mathbf{p}_2(t), \mathbf{v}(t), \mathbf{a}(t), \boldsymbol{\omega}(t)\}$
  end if ▷ Append data at t to the sequence
end while

Algorithm 2 Construct confidence intervals of positions

Input S
Output CI

$i \leftarrow 0$
while $i < w$ **do**
  $i \leftarrow i + 1$
  $\mathbf{W} \leftarrow$ (5) ▷ Compute the curve-fitting parameter
  $\hat{\mathbf{p}}_m(t - w + i) \leftarrow$ (4) ▷ Smoothen positions
  $\mathbf{x}_m(t - w + i) \leftarrow$ (8) ▷ Residuals after smoothing
end while
$\hat{\mathbf{x}}_m(t) \leftarrow$ (9) ▷ Estimate uncertainty at t using residuals
$CI \leftarrow$ (11) ▷ Combine smoothed position and uncertainty

onboard sensors $\mathbf{v}(t), \mathbf{a}(t) \in \mathbb{R}^3$ adheres to a local coordinate system in the same units as $\mathbf{p}_m(t)$, and $\boldsymbol{\omega}(t) \in \mathbb{R}^3$ comprises roll (ϕ), pitch (θ), and yaw (ψ) angles from gyroscope and magnetometer, which represent the orientations with respect to the local coordinates and WGS.

2) Window Size w: The length of data series S is a parameter: $S = \{\mathbf{p}_m(i), \mathbf{v}(i), \mathbf{a}(i), \boldsymbol{\omega}(i)\}$ for $t - w < i < t$. Numerous approaches are available to determine a good rolling window size. One example is to use a "trial and error" strategy that minimizes the mean squared error (MSE) of positioning. It is a small-scale experiment with a range of window sizes and evaluating their performance on the validation set before detection, which is shown in our experiment results. Upon selecting an appropriate window size w, we can then move forward with processing S.

3) Processing S: At each t, the filtering process receives detection feedback regarding the current GNSS position, indicating whether it is potentially under attack. In the event of an alarm, the filtering process constructs S for $t + 1$ by incorporating data from sources excluding GNSS, denoted as $\mathbf{p}_m(t), \mathbf{v}(t), \mathbf{a}(t), \boldsymbol{\omega}(t), m \in 1, 2, ..., M$. Conversely, if no attack is detected, the filtering updates S using information from all available sources, denoted as $\mathbf{p}_m(t), \mathbf{v}(t), \mathbf{a}(t), \boldsymbol{\omega}(t), m \in 0, 1, ..., M$.

B. Constructing Confidence Intervals

This process involves two main steps: first, a model-based approach using motion-constrained regression to obtain smoothed positions, depicted in the dotted lines of Fig. 3, and second, a data-driven approach using Gaussian processes to model position uncertainty, as depicted in the shaded areas of Fig. 3. The process is described in Algorithm 2.

1) Motion-Assisted Fitting: We use local polynomial regression for its flexibility in interpolating and predicting positions, based on discrete $\mathbf{p}_m(t)$ position points and motion data $\mathbf{v}(t), \mathbf{a}(t), \boldsymbol{\omega}(t)$. Crucially, we incorporate this with motion constraints derived from onboard sensors. It allows smoothing of noisy position measurements while ensuring the resulting position is physically plausible (adhering to velocity and acceleration limits), and the constrained optimization provides robustness against outliers compared to unconstrained fits or filters that might fuse spoofed data or IMU noise into

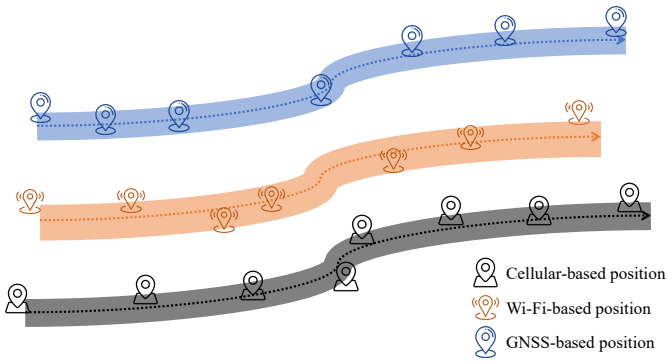

Fig. 3. Local polynomial regression is used to estimate traces (tiled view), and the Gaussian process is used to model the residual part of estimated positions (shaded areas).

updates. Local polynomial regression involves fitting a Taylor expansion at a given point of a function through weighted least squares [51]. Thus, for a polynomial with degree n, at a given time t, the estimator $\hat{\mathbf{p}}_m(t)$ is represented as

$$\hat{\mathbf{p}}_m(t) = \mathbf{W}\mathbf{t} \tag{4}$$

where $\mathbf{W} \in \mathbb{R}^{2\times(n+1)}$ denotes the polynomial coefficients of Taylor expansion that need to be determined, and $\mathbf{t}$ represents a $(n+1)$ dimensional vector, $[\mathbf{t}]_i = t^{i-1}$.

To determine $\mathbf{W}$, we introduce an optimization problem and present a theorem for the estimation process, ensuring both computational efficiency and reliability of position predictions. $\mathbf{W}$ at the mth position and time t is from

$$\begin{aligned} \min_{\mathbf{W}} \quad & f_{\mathcal{P}}(\mathbf{W}) \\ \text{s.t.} \quad & |\hat{\mathbf{p}}_m(t) - \tilde{\mathbf{p}}_m(t)| \le \boldsymbol{\epsilon}_t \end{aligned} \tag{5}$$

where $\tilde{\mathbf{p}}_m(t)$ is the mth position based on motion data (to constrain $\hat{\mathbf{p}}_m(t)$), and $\boldsymbol{\epsilon}_t \in \mathbb{R}^2$ represents a small tolerance. $\tilde{\mathbf{p}}_m(t)$ ensures the physical feasibility of a short-term movement (onboard sensors), and $\mathbf{p}_m(t)$ ensures long-term (network-based positions) anti-spoofing considerations. The objective function for regression in (5) should minimize the weighted squared error between the fitted polynomial $\hat{\mathbf{p}}_m(t') = \mathbf{W}\mathbf{t}'$ and the observed positions $\mathbf{p}_m(t')$, defined as

$$f_{\mathcal{P}}(\mathbf{W}) = \sum_{t'=t-w}^{t} [\mathbf{W}\mathbf{t}' - \mathbf{p}_m(t')]^{\mathrm{T}} K_{\mathrm{loc}}(t'-t)[\mathbf{W}\mathbf{t}' - \mathbf{p}_m(t')] \tag{6}$$

where $K_{\mathrm{loc}}(x) = \exp(-\kappa x^2)$ is a kernel function assigning weights that help emphasize the contribution of recent data points while down-weighting the influence of more distant points, κ is a kernel parameter, and $\mathbf{p}_m(t')$ are data points.

To provide $\tilde{\mathbf{p}}_m(t)$ for (5) using motion data, it is essential to standardize the coordinate systems of onboard sensors. $\mathbf{R}$ represents the rotation matrix responsible for converting the local coordinate system to WGS coordinates [27]:

$$\mathbf{R}(t) = \mathbf{R}_{\psi}(t)\mathbf{R}_{\theta}(t)\mathbf{R}_{\phi}(t).$$

The state of the mobile platform is $\big(\mathbf{p}_m(t), \mathbf{v}(t)\big)$, so

$$\begin{aligned} \tilde{\mathbf{p}}_m(t) = \mathbf{p}_m(t-\Delta t) + \mathbf{R}(t-\Delta t)\mathbf{v}(t-\Delta t)\Delta t \\ + \frac{1}{2}\mathbf{R}(t-\Delta t)\mathbf{a}(t-\Delta t)(\Delta t)^2 \end{aligned} \tag{7}$$

and $\tilde{\mathbf{p}}_m(0)$ are initialized by the first GNSS and network positions. Moreover, if the onboard sensor does not furnish velocity information, $\mathbf{v}(t)$ can be substituted by $\mathbf{v}(t-\Delta t) + \int_{t-\Delta t}^{t} \mathbf{a}(t)\mathrm{d}t$, and $\mathbf{v}(t-\Delta t)$ is from the checked GNSS. Similarly, if acceleration information is unavailable from the onboard sensor, $\mathbf{a}(t)$ is assumed to be zero, indicating uniform motion over Δt. Then, $\tilde{\mathbf{p}}_m(t)$ in the constraint provides a rough movement range for the smoothed position $\hat{\mathbf{p}}_m(t)$.

Theorem 1. *The estimator $\hat{\mathbf{p}}_m(t)$ in (4) can estimate $\mathbf{p}_m(t)$ within polynomial time.*

Proof. See Appendix A. □

This guarantees that (5) is convex and solvable in polynomial time, making the estimation suitable for real-time applications. Updating the position estimations from $t = 1$ to N, the $\hat{\mathbf{p}}_m(t)$ values should closely resemble the dotted lines depicted in Fig. 3, illustrating the smoothed positions. These lines represent the estimated trajectory of the mobile platform from a tiled view to enhance visualization.

2) Modeling Uncertainty: After obtaining the smoothed positions $\hat{\mathbf{p}}_m(t)$ from regression, we model the remaining uncertainty. While (6) provides a measure of fit, we employ Gaussian processes for a more principled and flexible approach to modeling uncertainty. Gaussian processes [24] offer a non-parametric, data-driven method to estimate the distribution of the position residuals. They can capture temporal correlations in the uncertainty via kernel functions, providing a better uncertainty representation than just assuming independent noise. Denote the residuals of the estimated positions at m, t as

$$\mathbf{x}_m(t) = \hat{\mathbf{p}}_m(t) - \mathbf{p}_m(t). \tag{8}$$

Then, in the absence of GNSS attack-induced deviations, $\{\mathbf{x}_m(i); i \in (0,t)\}$ are zero-mean Gaussian random variables with unknown standard deviations $\boldsymbol{\sigma}_m(i)$. A covariance function $K(\mathbf{x}_m(t), \mathbf{x}_m(t')) = \frac{1}{2}\mathbb{E}[(\mathbf{x}_m(t) - \mathbf{x}_m(t'))^2]$ is selected to characterize the interrelation of two residuals, $\mathbf{x}_m(t)$ and $\mathbf{x}_m(t')$, at time t and t'. Commonly used kernels include linear, polynomial, and squared exponential covariance functions, and the best model and hyperparameters can generally be selected from cross-validation [24]. Subsequently, a linear unbiased estimator can estimate the residual, $\mathbf{x}_m(t)$:

$$\hat{\mathbf{x}}_m(t) = \sum_{i=t-w}^{t-1} \lambda_i \mathbf{x}_m(i) \tag{9}$$

where $\sum_{i=t-w}^{t-1} \lambda_i = 1$. Gaussian process regression calculates λ_i, minimizing the variance of the estimation error:

$$\begin{aligned} \min_{\boldsymbol{\lambda}} \quad & \mathbb{V}[\hat{\mathbf{x}}_m(t) - \mathbf{x}_m(t)] \\ \text{s.t.} \quad & \textstyle\sum_{i=t-w}^{t-1} \lambda_i = 1 \end{aligned} \tag{10}$$

which can be solved using the Lagrangian method.

Algorithm 3 Decision based on confidence intervals from opportunistic position information

Input *CI*
Parameter γ
Output $\hat{\mathcal{H}}(t), f_{\mu,\sigma}(t), \mu(t)$

$\Lambda_{1:M}(\mathbf{p}_0(t)) \leftarrow$ (14) ▷ Fuse confidence intervals
$\mu(t) \leftarrow$ (16) ▷ Fused alternative position
$f_{\mu,\sigma}(t) \leftarrow$ (17) ▷ Compute anomaly score
if $f_{\mu,\sigma}(t) \geq \gamma$ **then**
    $\hat{\mathcal{H}}(t) \leftarrow \mathcal{H}_1$ ▷ Positive as score exceeds a threshold
else
    $\hat{\mathcal{H}}(t) \leftarrow \mathcal{H}_0$ ▷ Negative otherwise
end if

Theorem 2. *Given a covariance function, $\hat{\mathbf{x}}_m(t)$ in (9) can estimate $\mathbf{p}_m(t)$ uncertainty in polynomial time.*

Proof. See Appendix B. □

As the prediction yields a distribution for each time t, the confidence intervals $\mathcal{I}_m(t)$ indicate the uncertainty of the estimation. Since GNSS and network-based positions are subject to observational noise, the confidence intervals conform to a Gaussian distribution at each time t for each information source in a benign environment. Consequently, its mean $\hat{\mathbf{p}}_m(t)$ and standard deviation $\hat{\boldsymbol{\sigma}}_m(t)$ of $\hat{\mathbf{x}}_m(t)$ characterize the confidence interval:

$$\mathcal{I}_m(t) \sim \mathcal{N}(\hat{\mathbf{p}}_m(t), \hat{\boldsymbol{\Sigma}}_m(t)), m = 0, 1, ..., M \tag{11}$$

where $\hat{\boldsymbol{\Sigma}}_m(t) = \mathrm{diag}([\hat{\boldsymbol{\sigma}}_m(t)]^2) \in \mathbb{R}^{2\times 2}$ is a diagonal matrix and the square is Hadamard power. An illustration in Fig. 3 shows lines connecting the individual position pins, i.e., $\hat{\mathbf{p}}_m(t)$, while the shaded areas are the uncertainties, $\hat{\boldsymbol{\Sigma}}_m(t)$.

C. Decision-Making Using the Intervals

Having obtained Gaussian confidence intervals $\mathcal{I}_m(t) \sim \mathcal{N}(\hat{\mathbf{p}}_m(t), \hat{\boldsymbol{\Sigma}}_m(t))$, decision-making fuses this information from all position sources (i.e., GNSS, Wi-Fi, and cellular-based positions) into a single test statistic. It then utilizes an anomaly detector for GNSS position attacks. We have two perspectives in the context of test statistic construction. First, the temporal perspective assesses the historical behavior of positions over time to capture patterns and anomalies. Second, the categorical perspective groups different sources of positions.

1) Fusing Intervals: To process the data, S, along with its associated confidence intervals, $\mathcal{I}_m(t) = \hat{\mathbf{p}}_m(t) + \hat{\mathbf{x}}_m(t)$, which are derived from Algorithm 2, we fuse these confidence intervals. It involves aggregating the weighted confidence intervals across t with weights denoted as $K(m,t)$, which is a kernel function to ensure that $K(m,t)$ from $t-w$ to t sum to 1. Then, the temporal fusion is

$$Z(m,t) \triangleq \sum_{t'=t-w}^{t} K(m,t')\mathcal{I}_m(t') \tag{12}$$

and denote its probability density function (PDF) as $f_{Z(m,t)}(\mathbf{p})$. Thus, the mth test statistic for $\mathcal{H}_0$ is

$$\Lambda_m(\mathbf{p}_0(t))\,|\mathcal{H}_0 = f_{Z(m,t)}(\mathbf{p}_0(t)). \tag{13}$$

where $\mathbf{p}_0(t)$ is GNSS position. For M sources of positions, the fused test statistic is

$$\Lambda_{1:M}(\mathbf{p}_0(t)) = \prod_{m=0}^{M} \Lambda_m(\mathbf{p}_0(t))\,|\mathcal{H}_0\,. \tag{14}$$

To simplify the calculation, we observe that $\Lambda_{1:M}(\mathbf{p}_0(t))$ is proportional to a Gaussian PDF.

Theorem 3. *$\Lambda_{1:M}(x) = \frac{S}{\sigma(t)}\varphi\left(\frac{x-\mu(t)}{\sigma(t)}\right)$, where S is a constant scaling factor,*

$$\sigma(t) = \left(\sum_{m=0}^{M}\left(\sum_{t'=t-w}^{t}\left[K(m,t')\right]^2\left[\hat{\boldsymbol{\sigma}}_m(t')\right]^2\right)^{-1}\right)^{-\frac{1}{2}} \tag{15}$$

$$\mu(t) = \sigma^2(t)\sum_{m=0}^{M}\frac{\sum_{t'=t-w}^{t} K(m,t')\,\hat{\mathbf{p}}_m(t')}{\sum_{t'=t-w}^{t}\left[K(m,t')\right]^2\left[\hat{\boldsymbol{\sigma}}_m(t')\right]^2}\,. \tag{16}$$

Proof. See Appendix C. □

Note that in the fused mean $\mu(t)$, each source m's contribution is inversely weighted by its estimated uncertainty $\hat{\boldsymbol{\Sigma}}_m(t')$ (derived from the Gaussian process). This means that sources estimated to be less certain (higher variance) are naturally down-weighted in the test statistic.

2) Decision-Making: After constructing the function in (14), we apply Loda [25] to generate an anomaly score $f_{\mu,\sigma}(t)$, i.e., the probability of GNSS being under attack, shown as Algorithm 3. It is unsupervised, requiring no labeled attack data for training; lightweight and efficient, based on an ensemble of simple histograms on projections, making it suitable for resource-constrained platforms; robust due to its ensemble nature; and hyperparameter-free, simplifying deployment. The detector works by using random projections of the input and then comparing their histograms to find differences.

Input data consists of $\sigma(t)$ from (15) and $\mu(t) - \mathbf{p}_0(t)$ from (16) (other possible information includes antenna gain, dilution of precision, etc.). For the training phase, the algorithm learns the benign behavior of the data by constructing a set of models, i.e., representations, using a subset of the available benign data. This can be summarized by the following steps: (i) projection: it projects benign data $(\sigma(t), \mu(t) - \mathbf{p}_0(t))$ onto a lower-dimensional space using k random projection vectors, $\{\mathbf{v}_i\}_{i=1}^k$, to get the projected ones, $\{\tilde{x}_i\}_{i=1}^k$, where $\tilde{x}_i = (\sigma(t), \mu(t) - \mathbf{p}_0(t))\mathbf{v}_i$, (ii) histogram: it calculates histogram, $\mathbf{h}_i$, of each projected value, $\tilde{x}_i$, from ith projection vector, and (iii) representations: the projection vectors, $\{\mathbf{v}_i\}_{i=1}^k$, and histograms, $\{\mathbf{h}_i\}_{i=1}^k$, are the benign patterns.

For the testing (detecting) phase, the algorithm uses the representations constructed during the training phase to detect anomalies: (i) projection: this step is the same as in the training phase to get the projected $\{\tilde{z}_i\}_{i=1}^k$, and (ii) comparison: the projected testing vectors are compared with the trained

TABLE I
DATASETS FOR EXPERIMENTS.

	Ground Truth	GNSS	Network	Onboard Sensor
Dataset A	1 Hz	1 Hz	1 Hz & 1 Hz	200 Hz
Dataset B	1 Hz	1 Hz	0.1 – 0.3 Hz	100 Hz

histograms $\{\mathbf{h}_i\}_{i=1}^k$ to compute the anomaly score, $f_{\mu,\sigma}(t)$, by using the frequency of $\{\tilde{z}_i\}_{i=1}^k$ in the distribution of $\{\mathbf{h}_i\}_{i=1}^k$:

$$f_{\mu,\sigma}(t) = -\frac{1}{k}\sum_{i=1}^{k} \log \mathbb{P}[\tilde{z}_i]. \tag{17}$$

Note that all hyperparameters are determined automatically, as presented in [25]; thus, the detector is termed hyperparameter-free. The anomaly score, $f_{\mu,\sigma}(t)$, reflects the degree of abnormality relative to the learned benign distribution, which is empirically chosen based on the benign training data and environment. When $f_{\mu,\sigma}(t) \geq \gamma$, the decision is to raise an alarm that the GNSS position is attacked ($\mathcal{H}_1$). Even if it may be a false alarm, our provided recovered position from (16) is close to the actual position because it is a secure fusion of GNSS, network positions, and onboard sensors that removes spoofed GNSS positions in S. Therefore, this will not endanger the operation of one system that relies on this GNSS position.

D. Computational Complexity

The complexity of the proposed PADS framework in Algorithm 1 consists of solving the convex optimization problem (5), Gaussian process regression (10), and Loda (17). First, (5) implies computations for forming and solving a quadratic program. The cost to form the objective part is $\mathcal{O}(w(n+1)^2)$ and the constraint part is $\mathcal{O}(n)$, where n is the polynomial degree, usually taking a value of $1-3$. The cost to solve it depends on the (analytic, numerical) method, and it is approximately $\mathcal{O}((n+1)^3)$. Considering n is small, the total cost is $\mathcal{O}(w)$.

Second, for the construction of uncertainty using a Gaussian process, the main cost lies in solving the linear system derived from the Lagrangian to find the weights λ_i. This typically involves inverting a $w \times w$ covariance matrix, leading to a complexity of $\mathcal{O}(w^3)$. Third, fusing intervals requires calculations with complexity linear in M (number of sources) and w. Loda detector requires projecting the input data onto k random vectors. The complexity per detection is $\mathcal{O}(M \times w + k)$.

As a result, the total complexity of detecting $\mathbf{p}_0$ is $\mathcal{O}(w + w^3 + M \times w + k)$. Hence, the complexity is dominated by the Gaussian process component ($\mathcal{O}(w^3)$), but remains polynomial. Given $w = 15-30$, PADS is computationally feasible. Our following experiments on mobile platforms also confirmed efficient, real-time processing at typical GNSS rates.

VI. EXPERIMENT RESULTS

A. Experiment Setup

We have two datasets (Table I and illustrated in Fig. 4): (i) Dataset A [1], comprising 6 GNSS traces from outdoor urban environments and simulated attacks, and (ii) Dataset B is collected in a real GNSS-attack environment, Jammertest 2024, in Bleik village, Norway [52]. GNSS receivers are mounted on vehicles with speeds ranging from 0 to 90 km/h. The error for benign GNSS positions is mostly within 3–10 meters for autonomous vehicles and smartphones in the test.

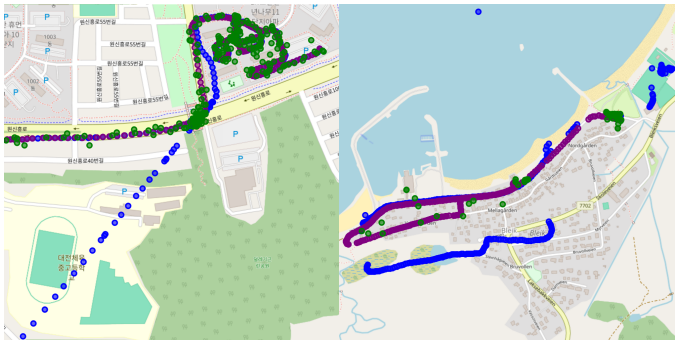

Fig. 4. Ground truth (purple), network (green), and GNSS (blue) positions of two traces from Dataset A (left) and B (right). Note that "network position" refers to both Wi-Fi and cellular positions in Dataset A, and positions from Android Network Location Provider in Dataset B.

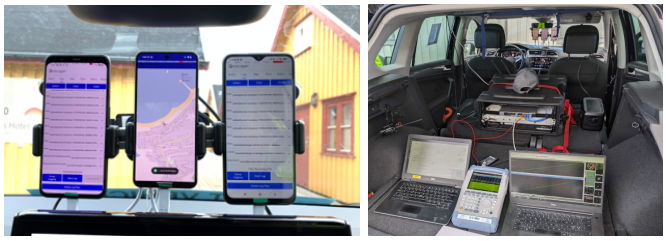

Fig. 5. The placement of the mounted phones in a vehicle (car windshield).

1) Dataset A: It includes GNSS positions, IMU data, and ground truth positions from a simulated Apollo autonomous driving platform [1]. In the same context, opportunistic position data is synthesized using a custom-made network simulator. The simulator generates the RSS from the seven nearest BSs and APs, whose positions are from open databases [53] and [54]. To accurately model the signal propagation, the simulation parameters for the free-space path loss model are derived from Long-Term Evolution (LTE) TR36.814 [55] and 802.11n 2.4 GHz. The transmit power of a BS is set at 20 dBm, while it is 15 dBm for an AP. Additionally, Gaussian noise with a variance level of 3 dB is added to the received power for network interfaces. The positioning algorithm based on AP and BS separately employs weighted nonlinear least squares [42], and the resulting positions are subject to estimation error with variances of 33 or 9 meters, as per [43], [56]. The unavailability probability, U_m, for $\mathbf{p}_m(t)$ is set at 0.2 per t for all $m = 1, 2, ..., M$, following a binomial distribution. The spoofing attacks are provided in [1]. Its strategy consists of (i) vulnerability profiling, where the attacker performs a constant spoofing to GNSS with a small deviation, and (ii) aggressive spoofing, where the attacker makes the deviation grow exponentially after the receiver accepts the spoofed position.

2) Dataset B: GNSS receivers are embedded in Android smartphones, which are Samsung Galaxy S9, Redmi 9, Google Pixel 4 XL, and Google Pixel 8, shown in Fig. 5, and two u-blox receivers as reference. The ground truth positions are provided by precise kinematic positioning results from u-blox ZED-F9P receivers using benign constellations and a nearby GNSS reference station[1]. The (opportunistic) network positions, GNSS positions, and IMU data are collected with GNSSLogger. The Android `LocationManager` provides access to multiple types of location services. It obtains 1 Hz updates of GNSS positions, while network positioning results are not at a fixed frequency and may update less frequently than GNSS, typically 3 to 10 seconds between updates, depending on the environment. The `SensorManager` stably delivers linear acceleration, gyroscope, magnetometer, and orientation at 100 Hz. Note that the u-blox receiver uses only benign satellite constellations and an RTK reference station as ground truth. The results presented later evaluate the detection capabilities of PADS using Android smartphone standard GNSS/Network/Sensors APIs while they were subjected to spoofing attacks. The u-blox by itself is not a baseline method. The GNSS position attacks involved both synchronous and asynchronous spoofing [52]: stationary spoofing of small/large position jumps, SBAS spoofing, simulated driving, flying spoofing, as well as jamming. The attack equipment includes cigarette-type jammers, handheld jammers, permanently installed jammers, BladeRF x115 mobile SDR spoofers, and USRP X300 SDRs. Simulated GNSS signals corresponding to 5 predefined traces were transmitted using Skydel and USRP X300 with an amplifier, including paths with small deviations or jumping to another distant place[2].

Since the attack gradually deviates GNSS positions from the ground truth positions, we define a lower bound of attack deviation, $\boldsymbol{\delta}_{\text{d}}$, for the ground truth detection results. Two types of positions are close when there is no attack. If the distance between GNSS and ground truth is larger than $\boldsymbol{\delta}_{\text{d}} = 10$ meters, we classify this GNSS position as the result of an attack, as our ground truth detection.

B. Baseline Methods

We consider the following four methods from related work as baselines, including SOP, IMU-based, and both, which can be implemented with the collected opportunistic information.

1) Signals of Opportunity (SOP, Baseline 1): [20] uses the broadcast signals from the BSs and APs to validate GNSS. It calculates weights $\mathbf{w} = [w_1, w_2, ..., w_J]$ based on RSS, where J is the number of BSs/APs, and the estimated mobile platform position is the weighted centroid $\hat{\mathbf{p}}_{\text{c}} = \frac{\mathbf{w} \cdot \mathbf{p}_{\text{bs}}}{|\mathbf{w}|}$, where $\mathbf{p}_{\text{bs}} \in \mathbb{R}^{J \times 2}$ is the concatenated coordinate of all BSs and APs. If the distance of $\hat{\mathbf{p}}_{\text{c}}$ and the GNSS-provided position is higher than a threshold, its outcome is "under attack".

[1]ZED-F9P logs UBX files, and then we use RTKLIB tool to convert them into RINEX files. The reference station with the code name "ANDE00NOR" is located at Andøya island and logs RINEX files. The RINEX files contain multiple constellations (Global Positioning System (GPS), GLONASS, Galileo, and BeiDou) plus SBAS and QZSS satellites, with two frequencies, so we use L1+L2, kinematic positioning mode, and benign constellations that exclude GPS and Galileo in the RTKLIB post-processing tool.

[2]https://github.com/NPRA/jammertest-plan/blob/main/Testcatalog.pdf

2) Kalman Filter (KF, Baseline 2): [57] fuses IMU and GNSS measurements, then we adapt it for IMU-based spoofing detection [15], [16]. The filter estimates the position of the mobile platform based on GNSS position and IMU. It minimizes the error of observation and motion to recursively get the mean and covariance matrix of the estimated position $\hat{\mathbf{p}}_{\text{c}}(t)$. If the residual between $\hat{\mathbf{p}}_{\text{c}}(t)$ and GNSS position is larger than a threshold, this method detects it as an attack.

3) Particle Filter (PF, Baseline 3): Similar to Kalman filter, we use a simple particle filter for IMU-based spoofing detection, which is based on the Markov Monte Carlo method [58]. It generates particles uniformly around the initial position and then calculates the error between particles and position measurements. Then, the estimated position is a weighted sum of the particles based on errors. In the detection phase, a distance threshold-based detector is used for classification.

4) Combined Metrics (GLRT, Baseline 4): It follows the generalized likelihood ratio test (GLRT) framework in [36]. Step 1 involves calculating detection metric $\log \Lambda_m = -\frac{1}{2}||\mathbf{p}_0(t) - \tilde{\mathbf{p}}_m(t)||^2_{\boldsymbol{\Sigma}_m(t)^{-1}}$ for m. Step 2 assumes these detection metrics are statistically independent, and a likelihood ratio function is used to combine them: $\log \Lambda_{1:M} = \sum_{m=1}^{M} \log \Lambda_m$. Step 3 tests $\log \Lambda_{1:M}$ whether it is zero mean (i.e., under $\mathcal{H}_0$) or non-zero (i.e., an attack, under $\mathcal{H}_1$).

Comparison includes three metrics: true positive rate, R_{TP}, detection time delay, ΔT, and the absolute error of recovered position, μ. To perform a fair comparison between PADS and the baseline methods, we assess three cases and PADS variants: (i) exclusive utilization of network-based positioning results (removing the constraints of (5), termed PADS-N), (ii) sole reliance on onboard sensors (removing network positions in (5), termed PADS-O), and (iii) combined usage of network positions and onboard sensors (as (5), termed PADS-A).

C. Evaluation: True Positive Rate

We investigate R_{TP} at different R_{FP} to plot ROC curves[3]. Our rolling window size is empirically set to $w = 15$ for GNSS-provided and network-based positions, and we choose $\kappa = 1$ in the kernel function K_{loc}.

PADS-N shares the same network conditions as Baseline 1. As shown in Fig. 6, PADS-N exhibits a modest improvement, at most 33%, when $R_{\text{FP}} < 15\%$. In Dataset A, it achieves R_{TP} of 81–96% when R_{FP} is 5–15%, compared to 63–87% for Baseline 1. In Dataset B, it achieves R_{TP} of 36–54% when R_{FP} is 5–15%, similar to 33–57% for Baseline 1. When $R_{\text{FP}} > 10\%$, PADS-N has at most 20% performance gain. Furthermore, the performance with Dataset A is better than that with B because network positions are much sparser and noisier in Dataset B. In general, as R_{FP} increases, R_{TP} tends to increase and converge for both PADS-N and SOP, both methods detecting the attacks well and thus resisting gradual deviation or position jumping spoofing.

PADS-O, KF (Baseline 2), and PF (Baseline 3) use GNSS position, incorporating motion data from onboard sensors.

[3]All the work here is at the level of single-position detection, i.e., detecting each GNSS position based on historical measurements of the trace. We did not attempt to investigate whether a trace is under attack based on the spoofing detections of the positions of the entire trace.

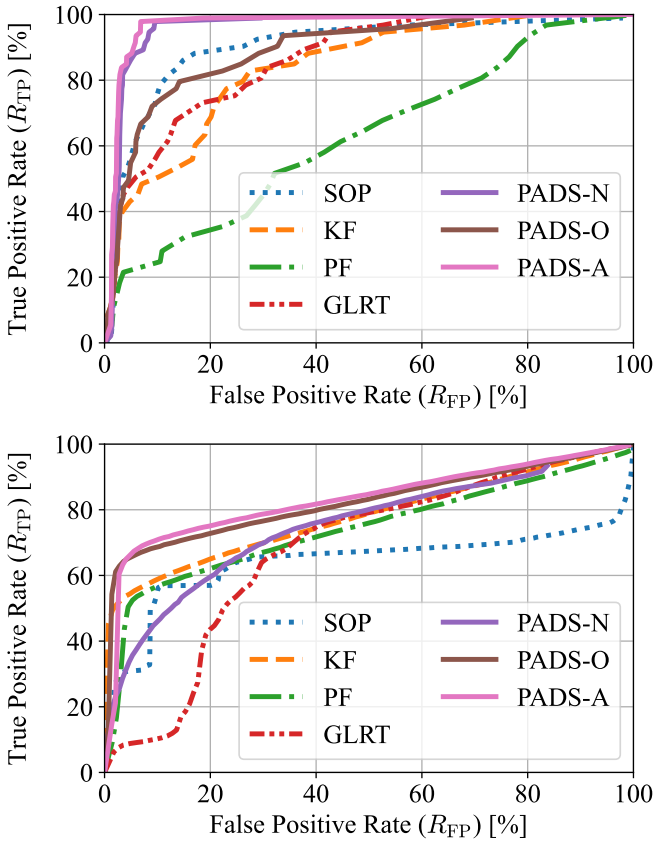

Fig. 6. ROC curves for Dataset A (upper) and B (lower), i.e., R_{TP} versus R_{FP}.

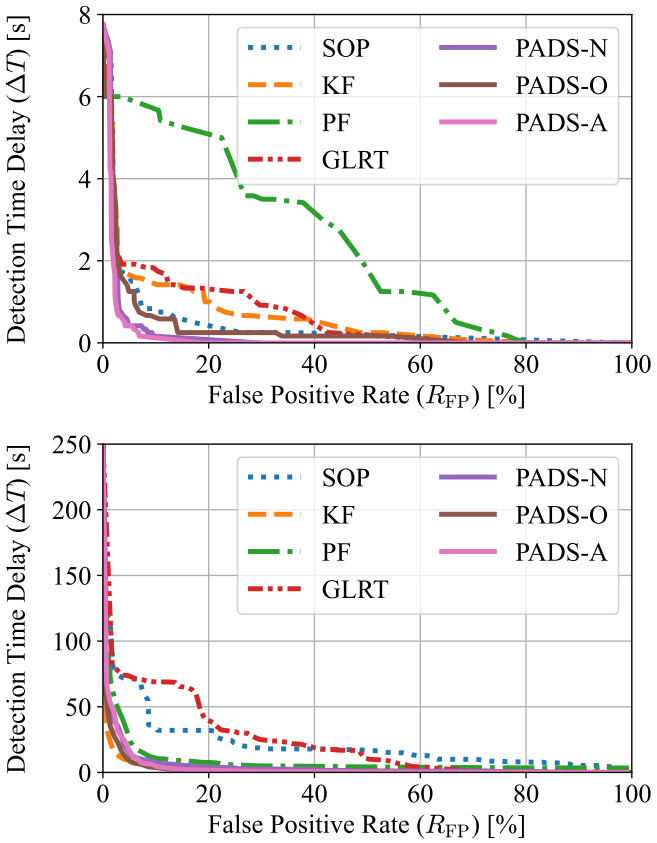

Fig. 7. ΔT versus R_{FP} for Dataset A (upper) and B (lower).

Fig. 6 illustrates R_{TP} as a function of R_{FP}. PADS-O and Baseline 2 maintain relatively consistent and similar trends of R_{TP}. However, in both Dataset A and B, Baseline 2 and Baseline 3 indicate relatively low R_{TP}, even at higher R_{FP}. PADS-O outperforms them with at most a 44% R_{TP} gain in Dataset A and 23% in B, when R_{FP} is 5–15%. This is because the spoofed positions will influence the filters, while the proposed scheme detects and screens the spoofed position simultaneously. Furthermore, the regression in PADS-O can not only deal with Gaussian noise but also general zero mean noise, according to the least squares assumptions. Whenever the noise is not zero mean, PADS-O detects it as an anomaly caused by spoofing.

Compared to PADS-A, GLRT (Baseline 4) struggles due to the absence of rolling window and motion-constrained regression for position data, hindering the effective fusion of heterogeneous data (i.e., positions, velocity, and acceleration). When utilizing all available opportunistic information sources, PADS-A consistently surpasses Baseline 4 across R_{FP}.

When comparing PADS-N and PADS-A, incorporating IMU results in a performance improvement of up to 7% with Dataset A and 32% with Dataset B. IMU data is particularly helpful when attack-induced position deviation grows fast. In cases of subtle deviation changes, the effectiveness is relatively low. PADS-A achieves higher R_{TP} compared to PADS-N and PADS-O, highlighting the performance gain resulting from the fusion of network-based positioning results and onboard sensor data. Considering PADS-O and other factors such as cost, sensor availability, and system complexity, the filters—Baseline 2 and Baseline 3—prove to be less effective compared to PADS-N but are relatively easy to acquire. We also observe that Baseline 2 and Baseline 3 perform much better in Dataset B than in A. This is because Dataset B is collected in a village without too many network infrastructures and the network positions are very sparse, so network positions can not produce high performance gain compared to onboard sensors.

D. Evaluation: Detection Time Delay

The detection time delay, ΔT, represents the duration between the initiation of an attack and its detection. Given the stealthy nature of spoofing attacks in GNSS traces, where deviations from the actual position evolve, analyzing the time delay in detecting such attacks is important. Our focus in this context is on measuring ΔT, independent of computation delays. This metric reflects how fast the schemes can identify attacks, i.e., the sensitivity of the detection schemes. It is also influenced by factors such as the rolling window size, which will be investigated later.

Fig. 7 presents ΔT as a function of R_{FP} for all three cases, respectively. PADS-N exhibits performance curves with similar shapes to Baseline 1 but mostly lower ΔT. We have discovered that network-derived positions are inherently noisy, and PADS-N possesses the capability to effectively smoothen out this noise, resulting in a more accurate estimation of

TABLE II
ABSOLUTE ERROR EVALUATION OF RECOVERED POSITION ACCURACY OVER DIFFERENT METHODS AND DATASETS.

Methods	Dataset A				Dataset B			
	Mean	Median	Best 20%	Worst 20%	Mean	Median	Best 20%	Worst 20%
PADS-N	2.45	1.57	0.48	3.77	243.82	30.90	8.48	260.36
PADS-O	11.43	2.02	0.59	5.00	276.61	35.87	9.32	350.17
PADS-A	2.92	1.87	0.83	4.22	260.61	35.62	9.31	345.05
SOP	5.87	4.80	2.42	8.85	544.91	298.14	283.60	655.61
KF	15.50	6.01	1.55	13.28	280.91	36.53	9.57	369.00
PF	64.11	24.11	6.92	81.96	266.64	48.52	9.00	275.06
GLRT	15.43	6.02	1.14	13.28	280.90	36.52	9.51	369.00
GNSS	30.21	22.33	13.74	41.41	370.53	169.93	17.34	479.69

the actual position. The improved accuracy, in turn, allows it to detect spoofing and anomalies more quickly. Furthermore, PADS-N demonstrates a more significant reduction in ΔT as R_{FP} increases to 15%, indicating a better trade-off between false alarms and detection sensitivity.

PADS-O, KF (Baseline 2), and PF (Baseline 3) consistently exhibit ΔT exceeding at most 5–12 seconds when R_{FP} is 5–15%, due to the necessity of updating posterior distributions over sufficiently large time and accumulating deviation than the detection threshold. Moreover, the filtering process is influenced by the spoofed positions fed into the filter. In contrast, PADS-O outperforms both Baseline 2 and Baseline 3 by data fitting with motion information constraints and excluding spoofed positions before regression.

PADS-A showcases lower ΔT compared to GLRT (Baseline 4) and the aforementioned filter-based schemes at equivalent R_{FP}. It consistently exhibits faster detection times across different datasets and R_{FP}. Additionally, PADS-A demonstrates a more significant reduction in ΔT as R_{FP} increases, highlighting its success in keeping a balance between minimizing false alarms and reducing delay. Baseline 4 also performs well, delivering competitive ΔT compared to other schemes, especially at lower $R_{\text{FP}} < 5\%$.

When considering three cases in conjunction with others, it is clear that better detection schemes and more precise network-based position information contribute to lower ΔT. A high R_{FP} also leads to a low ΔT. Cases that give accurate opportunistic information for making decisions are more likely to catch spoofing attacks quickly. Conversely, methods that fail to integrate this information tend to miss detections, resulting in longer ΔT. The use of the detection for high R_{FP} is at the discretion of the method user, and we did not investigate how to use several successive alarms under some high R_{FP}.

E. Evaluation: Recovered Position Accuracy

The recovered position is defined as the mean of the confidence intervals in (16). We consider all spoofed positions and calculate the absolute error between the actual and the recovered position. Table II showcases the worst 20% (i.e., 20% of the errors are higher than this), best 20% (i.e., 20% of the errors are lower than this particular), median, and mean error of the distribution. The last row, GNSS, is the result of positioning ($\mathbf{p}_0(t)$) without any network signals and onboard sensors, representing the raw spoofing distances. Baseline 1 has a relatively high worst 20% error compared to its mean error, indicating a big fluctuation in its performance. Notably, PADS-N demonstrates significantly lower error values than Baseline 1, particularly in terms of the worst 20% error with both datasets. While PADS-O has a much lower error, Baseline 3 also has a relatively good performance for the best 20% error. Regarding the raw GNSS error during attacks, the average distance is 30.21 meters in Dataset A and 370.53 meters in Dataset B.

When considering the performance difference between Dataset A and B, all four types of errors are much higher with Dataset B than A. This is because the autonomous vehicle has much more accurate onboard sensors and GNSS positioning than the smartphones in the experiments. Also, autonomous vehicles are used in the outdoor urban environment, whereas smartphones are used in the village environment. As a result, the former has higher-quality network positions than the latter. Specifically, for the ratio of mean to median error, Dataset B is much larger than A, which means that many spoofed positions are not recovered, resulting in the mean being much larger than the median. This can also be confirmed in the detection accuracy of R_{FP} and the worst 20% error. This is due to some long-range absence of network positions in Dataset B, so these spoofed positions are mostly not detected and recovered. We also find the errors of filters and PADS-O are more similar in Dataset B than A, which means smartphone IMU contributes a lot in this experiment setting.

PADS has the lowest mean error, ranging from 19% to 98% of the mean absolute error (MAE) observed in other methods, indicating better overall accuracy in estimation. This notable accuracy is due to the effective combination of various opportunistic information sources. While absolute error is a critical metric, it is also important to consider other related factors. They include consistency, reflecting whether the position result is stable as dataset size or categories of opportunistic information increase; variability, which denotes the uncertainty size associated with the position result; and specific application requirements, such as the types of sensors available on the mobile platform.

TABLE III
R_{TP} UNDER DIFFERENT w AND κ VERSUS R_{FP}.

R_{FP}	w							κ						
	5	10	15	20	25	30	35	0	0.5	1.0	1.5	2.0	2.5	3.0
5%	86.6%	90.3%	88.2%	89.0%	91.1%	91.3%	89.2%	55.4%	89.7%	88.2%	86.0%	85.5%	85.4%	86.0%
10%	98.9%	98.3%	98.7%	97.8%	96.8%	98.7%	97.8%	73.7%	98.9%	98.9%	97.8%	97.8%	97.8%	97.8%
15%	98.9%	98.9%	98.9%	98.9%	98.9%	99.8%	100%	84.9%	98.9%	98.9%	98.9%	98.9%	98.9%	98.9%
20%	98.9%	100%	100%	100%	100%	100%	100%	91.4%	100%	100%	99.5%	100%	100%	99.5%
25%	100%	100%	100%	100%	100%	100%	100%	91.4%	100%	100%	100%	100%	100%	100%

F. Effect of Rolling Window

We obtain some insights from the detection performance for different choices of window size w and kernel parameter κ in Sec. V-A. In this experiment, we use Dataset A, and the settings are the same as the previous PADS-A, and R_{FP} is 5–25%, but w ranges from 5 to 35 samples and κ ranges from 0 to 3.0. The metrics for performance evaluation are the true positive rate, R_{TP}, shown in Table III.

Determining the appropriate w is a trade-off, as smaller windows offer computational efficiency but potentially sacrifice detection accuracy. Conversely, larger window sizes may lead to the processing of unnecessary historical data, resulting in slower and less accurate detection. Moreover, assigning weights to the data samples, which are controlled by κ in the kernel function, is also important to make better use of the historical data.

R_{TP} in Table III reveals some rough "optimal" values for w and κ, indicating that further increasing w or κ does not significantly benefit detection performance. A small value of w or κ leads to lower accuracy, whereas increasing them will also increase R_{TP} until they reach a sufficient size. This is because an oversized rolling window includes unnecessary data and gives too much weight, diminishing accuracy. R_{TP} increases as w increases, but the performance is stable after $w \geq 10$. R_{TP} increases as κ increases, up until $\kappa = 1.0$. Beyond this point, R_{TP} starts to decrease again, as a value of $\kappa \geq 1.5$ assigns much higher weights to historical data, which can obscure the current information. In terms of computational complexity, the w directly influences the complexity, i.e., the bigger w comes with the higher complexity, as explained in Sec. V-B. However, the change of κ will not impact the complexity.

G. Discussion

PADS demonstrates much better performance compared to baseline methods in Sec. VI due to its fusion of opportunistic information over time and consideration of position correlations. In most cases, by integrating data from all available sources, PADS-A naturally surpasses both PADS-N and PADS-O variants in R_{TP} and ΔT. Regarding the recovered position accuracy, PADS-A is in the middle of PADS-N and PADS-O. This highlights the difference between positioning and spoofing detection; even if the MAE of the positioning algorithm is excellent, it may not be good at detecting position spoofing. Hence, each has its own focus, and designing a detection-specific algorithm is important. Moreover, PADS-A mitigates the impact of accumulated errors from onboard sensors and one-time errors from network-based positioning, addressing both long-term and short-term inaccuracies. By incorporating uncertainty modeling (in Sec. V-B2), we get an improved R_{TP} compared to PADS without uncertainty.

Given that the proposed PADS can operate at a software layer and does not rely on low-level hardware, its compatibility with other signal-level or cryptographic anti-attack methods is seamless. Incorporating this algorithm into existing consumer-grade devices is easy, making it convenient for deployment alongside other security measures to provide a multi-layer defense. With positions and motion information as its only inputs, both commonly available in consumer-grade mobile platforms, the algorithm can opportunistically validate GNSS positions whenever network-based positions are accessible, providing a versatile and easily implementable solution for GNSS attack detection. Also, the computational complexity of PADS is adjustable, depending on the number of projection vectors in the lightweight anomaly detector [25] and the chosen window size in Algorithm 1.

VII. Conclusion

This paper presents an algorithmic framework for constructing confidence intervals for positions from opportunistic information and integrates them to estimate the probability of GNSS spoofing. It leverages both the motion dynamics of the mobile platform and the statistical characteristics of the positions. We employ a local polynomial regression technique with motion constraints, which is mathematically demonstrated to be convex for estimating and smoothing the position. Then, using Gaussian process regression, we capture the uncertainties inherent in position prediction and combine them into a fused test statistic for an unsupervised anomaly detector. The evaluation, based on both simulated and real-world attack datasets collected on common mobile platforms, shows significant improvements, including up to a 54% increase in the true positive rate. Specifically, when the false positive rate is between 5% and 10%, we have a 7–48% gain in the true positive rate.

Appendix A
Proof of Theorem 1

Recall the problem formulation (5):

$$\begin{aligned} \min_{\mathbf{W}} \quad & \sum_{t'=t-w}^{t} [\mathbf{W}\mathbf{t}' - \mathbf{p}_m(t')]^{\mathrm{T}} K_{\text{loc}}(t'-t)[\mathbf{W}\mathbf{t}' - \mathbf{p}_m(t')] \\ \text{s.t.} \quad & |\hat{\mathbf{p}}_m(t) - \tilde{\mathbf{p}}_m(t)| \leq \boldsymbol{\epsilon}_t \end{aligned}$$

where $\hat{\mathbf{p}}_m(t) = \mathbf{W}\mathbf{t}$ and $\mathbf{t}$ is the vector $[1, t, t^2, ..., t^n]^{\mathrm{T}}$. $\mathbf{W}$ contains the coefficients we need to find. The kernel $K_{\text{loc}}(t' - t)$ assigns weights, and $\tilde{\mathbf{p}}_m(t)$ is the motion-derived position constraint.

To check for convexity, we take the second derivative of the objective function with respect to $\mathbf{W}$:

$$\nabla^2 f_{\mathcal{P}}(\mathbf{W}) = 2 \sum_{t=t'-w}^{t'} K_{\text{loc}}(t - t') \cdot (\mathbf{t}' \cdot \mathbf{t}'^{\mathrm{T}})^{\mathrm{T}} \otimes \mathbb{I}$$

which is a positive definite matrix, as $K_{\text{loc}}(t - t') > 0$ always holds, and the matrix $(\mathbf{t}' \cdot \mathbf{t}'^{\mathrm{T}})^{\mathrm{T}} \otimes \mathbb{I}$ is positive semidefinite. Thus, the objective function is convex. The constraints in (5) are equivalent to

$$\begin{cases} \mathbf{W}\mathbf{t} - \tilde{\mathbf{p}}_m(t) \leq \boldsymbol{\epsilon}_t \\ \mathbf{W}\mathbf{t} - \tilde{\mathbf{p}}_m(t) \geq -\boldsymbol{\epsilon}_t \end{cases}, \forall t$$

which are the absolute difference between the fitted position $\hat{\mathbf{p}}_m(t) = \mathbf{W}\mathbf{t}$ and the motion-derived position $\tilde{\mathbf{p}}_m(t)$, so these are linear inequality constraints on $\mathbf{W}$. The set of points satisfying a system of linear inequalities forms a convex set. Hence, (5) is a convex optimization problem. It is solvable using Lagrange multipliers or the practical algorithms like interior-point methods; thus, the estimator $\hat{\mathbf{p}}_m(t)$ can estimate $\mathbf{p}_m(t)$ in polynomial time.

Appendix B
Proof of Theorem 2

Ordinary Gaussian process regression uses a linear unbiased estimator for $\mathbf{x}_m(t)$. We can use Lagrange multipliers to extract the λ_i parameters from the optimization problem.

$$\begin{aligned} L(\boldsymbol{\lambda}, \mu) &= \mathbb{V}[\hat{\mathbf{x}}_m(t) - \mathbf{x}_m(t)] + \mu(\sum_{i=t-w}^{t-1} \lambda_i - 1) \\ &= \mathbb{E}[\sum_{i=t-w}^{t-1} \lambda_i \mathbf{x}_m(i) - \mathbf{x}_m(t)]^2 + \mu(\sum_{i=t-w}^{t-1} \lambda_i - 1) \\ &= \sum_{i=t-w}^{t-1} \lambda_i \mathbb{E}[\mathbf{x}_m(i) - \mathbf{x}_m(t)]^2 \\ &\quad - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j \mathbb{E}[\mathbf{x}_m(i) - \mathbf{x}_m(j)]^2 + \mu(\sum_{i=t-w}^{t-1} \lambda_i - 1) \end{aligned}$$

where $\mathbb{E}[\mathbf{x}_m(i) - \mathbf{x}_m(t)]^2$ and $\mathbb{E}[\mathbf{x}_m(i) - \mathbf{x}_m(j)]^2$ are calculated from a fixed covariance function $K(\mathbf{x}_m(t), \mathbf{x}_m(t'))$. Then, we take the partial derivatives of $L(\boldsymbol{\lambda}, \mu)$ and set them to 0:

$$\frac{\partial L(\boldsymbol{\lambda}, \mu)}{\partial \boldsymbol{\lambda}} = 0 \tag{18}$$

$$\frac{\partial L(\boldsymbol{\lambda}, \mu)}{\partial \mu} = 0 \tag{19}$$

obtaining a system of $w + 1$ linear equations in the $w + 1$ unknowns ($\boldsymbol{\lambda}$ and μ). There exist several algorithms for solving it, such as Gaussian elimination. The computational complexity is dominated by the inversion or decomposition of the $(w + 1) \times (w + 1)$ matrix, which takes approximately $\mathcal{O}(w^3)$ arithmetic operations.

Appendix C
Proof of Theorem 3

Assuming independence among the random variables $\mathcal{I}_m(t)$, we consider the time slots from $t - w$ to t, where $K(m, t)$ is determined by a kernel function to ensure that $K(m, t)$ from $t - w$ to t sum to 1. Then, we use the moment-generating function:

$$M_{\mathcal{I}_m(t)}(s) = \mathbb{E}[e^{s\mathcal{I}_m(t)}]. \tag{20}$$

Recall that the weighted integral of $\mathcal{I}_m(t)$ from $t - w$ to t is

$$Z(m, t) = \int_{t-w}^{t} K(m, t')\mathcal{I}_m(t')\mathrm{d}t' \tag{21}$$

practically utilized in discrete form:

$$Z(m, t) = \sum_{t'=t-w}^{t} K(m, t')\mathcal{I}_m(t'). \tag{22}$$

Its moment-generating function is

$$\begin{aligned} M_{Z(m,t)}(s) &= \mathbb{E}\left[e^{sZ(m,t)}\right] \\ &= \mathbb{E}\left[e^{s\sum_{t'=t-w}^{t} K\left(m,t'\right)\mathcal{I}_m(t')}\right] \\ &= \prod_{t'=t-w}^{t} \mathbb{E}\left[e^{sK\left(m,t'\right)\mathcal{I}_m(t')}\right] \\ &= \prod_{t'=t-w}^{t} M_{\mathcal{I}_m(t')}\left(K\left(m, t'\right)s\right). \end{aligned}$$

The moment-generating function of a Normal distribution, $\mathcal{N}(\mu, \sigma^2)$, is given by $\exp(s\mu + \frac{1}{2}\sigma^2 s^2)$. Thus,

$$\begin{aligned} M_{Z(m,t)}(s) &= \prod_{t'=t-w}^{t} e^{K\left(m,t'\right)\hat{\mathbf{p}}_m(t')s + \frac{1}{2}\left[K\left(m,t'\right)\right]^2[\hat{\boldsymbol{\sigma}}_m(t')]^2 s^2} \\ &= e^{\sum_{t'=t-w}^{t} K\left(m,t'\right)\hat{\mathbf{p}}_m(t')s + \frac{1}{2}\sum_{t'=t-w}^{t}\left[K\left(m,t'\right)\right]^2[\hat{\boldsymbol{\sigma}}_m(t')]^2 s^2}. \end{aligned}$$

$Z(m, t)$ follows a normal distribution, $\mathcal{N}(\sum_{t'=t-w}^{t} K\left(m, t'\right)\hat{\mathbf{p}}_m(t'), \sum_{t'=t-w}^{t}\left[K\left(m, t'\right)\right]^2 \hat{\boldsymbol{\Sigma}}_m\left(t'\right))$, which means we compute the distribution $Z(m, t)$ by taking the weighted mean of distributions $\mathcal{I}_m(t)$.

Let $x \triangleq \mathbf{p}_0(t)$, $\mu_{(m)} \triangleq \mu\left(Z\left(m, t\right)\right)$, $\sigma_{(m)} \triangleq \sigma\left(Z\left(m, t\right)\right)$, and $\mu_{(0,1,...,M)}, \sigma_{(0,1,...,M)}$ refer to the parameters of the multiplied Gaussian functions. The multiplication of the first two Gaussian functions, $\Lambda_0\left(\mathbf{p}_0\left(t\right)\right) \times \Lambda_1\left(\mathbf{p}_0\left(t\right)\right)|\mathcal{H}_0$, is

$$\begin{aligned} &\prod_{m=0}^{1} \frac{1}{\sigma\left(Z\left(m, t\right)\right)} \varphi\left(\frac{x - \mu\left(Z\left(m, t\right)\right)}{\sigma\left(Z\left(m, t\right)\right)}\right) \\ =& \prod_{m=0}^{1} \frac{1}{\sigma_{(m)}} \varphi\left(\frac{x - \mu_{(m)}}{\sigma_{(m)}}\right) \\ =& \frac{1}{2\pi\sigma_{(0)}^2\sigma_{(1)}^2} \exp\left[-\frac{(x - \mu_{(0)})^2}{2\sigma_{(0)}^2} - \frac{(x - \mu_{(1)})^2}{2\sigma_{(1)}^2}\right] \\ =& \frac{S_{(0,1)}}{\sqrt{2\pi}\sigma_{(0,1)}} \exp\left[-\frac{(x - \mu_{(0,1)})^2}{2\sigma_{(0,1)}^2}\right] \end{aligned}$$

where $\varphi(\cdot)$ represents the standard normal PDF, $S_{(0,1)}$ is a constant scaling value, $\frac{1}{\sigma^2_{(0,1)}} = \frac{1}{\sigma^2_{(0)}} + \frac{1}{\sigma^2_{(1)}}$ and $\frac{\mu_{(0,1)}}{\sigma^2_{(0,1)}} = \frac{\mu_{(0)}}{\sigma^2_{(0)}} + \frac{\mu_{(1)}}{\sigma^2_{(1)}}$. Similarly, for mth Gaussian function, we have

$$\frac{1}{\sigma^2_{(0,1,\ldots,m)}} = \frac{1}{\sigma^2_{(0,1,\ldots,m-1)}} + \frac{1}{\sigma^2_{(m)}}$$
$$\frac{\mu_{(0,1,\ldots,m)}}{\sigma^2_{(0,1,\ldots,m)}} = \frac{\mu_{(0,1,\ldots,m-1)}}{\sigma^2_{(0,1,\ldots,m-1)}} + \frac{\mu_{(m)}}{\sigma^2_{(m)}}$$

This process can be extended to multiply $M+1$ Gaussian likelihoods. By mathematical induction, it follows that

$$\frac{1}{\sigma^2} \triangleq \frac{1}{\sigma^2_{(0,1,\ldots,M)}} = \frac{1}{\sigma^2_{(0)}} + \frac{1}{\sigma^2_{(1)}} + \ldots + \frac{1}{\sigma^2_{(M)}}$$
$$\frac{\mu}{\sigma^2} \triangleq \frac{\mu_{(0,1,\ldots,M)}}{\sigma^2_{(0,1,\ldots,M)}} = \frac{\mu_{(0)}}{\sigma^2_{(0)}} + \frac{\mu_{(1)}}{\sigma^2_{(1)}} + \ldots + \frac{\mu_{(M)}}{\sigma^2_{(M)}}$$

which is equivalent to (15) and (16). Note that the scaling value

$$S = \frac{(2\pi)^{-\frac{M}{2}} \sigma e^{(\mu^2/\sigma^2 - \sum_{i=0}^{M} \mu(Z(m,t))^2/\sigma(Z(m,t))^2)/2}}{\prod_{m=0}^{M} \sigma\left(Z\left(m,t\right)\right)^2} \quad (23)$$

and it will not change the optimum. Thus, it is feasible to exclude this part from the calculation.

REFERENCES

[1] J. Shen, J. Y. Won, Z. Chen, and Q. A. Chen, "Drift devil: Security multi-sensor fusion based localization high-level autonomous driving GPS spoofing," in *Proc. 29th USENIX Security*, virtual event, Aug. 2020.

[2] M. L. Psiaki, T. E. Humphreys, and B. Stauffer, "Attackers can spoof navigation signals without our knowledge. here's how fight back GPS lies," *IEEE Spectrum*, vol. 53, no. 8, pp. 26–53, 2016.

[3] D. Goodin, "GPS interference caused FAA reroute Texas air traffic. experts stumped," *Ars Technica*, 2022. [Online]. Available: https://arstechnica.com/information-technology/2022/10/cause-is-unknown...

[4] M. Psiaki and T. Humphreys, *Civilian GNSS Spoofing, Detection, Recovery*. Wiley-IEEE Press, 2021, pp. 655–680.

[5] K. Wesson, M. Rothlisberger, and T. Humphreys, "Practical cryptographic civil GPS signal authentication," *J. Inst. Navigation*, vol. 59, no. 3, pp. 177–193, 2012.

[6] I. Fernández-Hernández, V. Rijmen, G. Seco-Granados, J. Simon, I. Rodríguez, and J. D. Calle, "A navigation message authentication proposal Galileo open service," *J. Inst. Navigation*, vol. 63, no. 1, pp. 85–102, 2016.

[7] J. M. Anderson, K. L. Carroll, N. P. DeVilbiss, J. T. Gillis, J. C. Hinks, B. W. O'Hanlon, J. J. Rushanan, L. Scott, and R. A. Yazdi, "Chips-message robust authentication (Chimera) GPS civilian signals," in *Proc. 30th ION GNSS+*, Portland, OR, USA, Sep. 2017.

[8] P. Papadimitratos and A. Jovanovic, "GNSS-based positioning: Attacks countermeasures," in *Proc. IEEE Mil. Commun. Conf.*, San Diego, CA, USA, Nov. 2008.

[9] A. Broumandan, A. Jafarnia-Jahromi, V. Dehghanian, J. Nielsen, and G. Lachapelle, "GNSS spoofing detection handheld receivers based signal spatial correlation," in *Proc. IEEE/ION PLANS*, Myrtle Beach, SC, USA, Apr. 2012.

[10] K. Zhang, R. A. Tuhin, and P. Papadimitratos, "Detection exclusion RAIM algorithm spoofing/replaying attacks," in *Proc. Int. Symp. GNSS*, Kyoto, Japan, Nov. 2015.

[11] S. Liu, X. Cheng, H. Yang, Y. Shu, X. Weng, P. Guo, K. C. Zeng, G. Wang, and Y. Yang, "Stars can tell: Robust method defend GPS spoofing attacks using off-the-shelf chipset," in *Proc. 30th USENIX Security*, virtual event, Aug. 2021.

[12] S. Barbeau, "GPSTest database," *Google Sheets*, 2025. [Online]. Available: https://docs.google.com/spreadsheets/d/1jXtRCoEnnFNWj6_oFlVWflsf-b0jkfZpyhN-BXsv7uo/

[13] G. Oligeri, S. Sciancalepore, O. A. Ibrahim, and R. Di Pietro, "Drive me not: GPS spoofing detection cellular network," in *Proc. 12th ACM WiSec*, Miami, FL, USA, May 2019.

[14] Y. Gao and G. Li, "A slowly varying spoofing algorithm avoiding tightly-coupled GNSS/IMU multiple anti-spoofing techniques," *IEEE Trans. Veh. Technol.*, vol. 71, no. 8, pp. 8864–8876, 2022.

[15] S. Khanafseh, N. Roshan, S. Langel, F.-C. Chan, M. Joerger, and B. Pervan, "GPS spoofing detection using RAIM INS coupling," in *Proc. IEEE/ION PLANS*, Monterey, CA, USA, May 2014.

[16] M. Ceccato, F. Formaggio, N. Laurenti, and S. Tomasin, "Generalized likelihood ratio test GNSS spoofing detection devices IMU," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 3496–3509, 2021.

[17] J. J. Morales and Z. M. Kassas, "Tightly coupled inertial navigation system signals opportunity aiding," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 57, no. 3, pp. 1930–1948, 2021.

[18] M. Maaref and Z. M. Kassas, "Autonomous integrity monitoring vehicular navigation cellular signals opportunity IMU," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 5586–5601, 2021.

[19] Z. M. Kassas, J. Khalife, A. A. Abdallah, and C. Lee, "I am not afraid GPS jammer: Resilient navigation signals opportunity GPS-denied environments," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 37, no. 7, pp. 4–19, 2022.

[20] G. Oligeri, S. Sciancalepore, O. A. Ibrahim, and R. Di Pietro, "GPS spoofing detection crowd-sourced information connected vehicles," *Comput. Netw.*, vol. 216, p. 109230, 2022.

[21] Y. Gao and G. Li, "A GNSS instrumentation covert directional spoofing algorithm UAV equipped tightly-coupled GNSS/IMU," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–13, 2023.

[22] Z. M. Kassas, N. Khairallah, J. J. Khalife, C. Lee, J. Jurado, S. Wachtel, J. Duede, Z. Hoeffner, T. Hulsey, R. Quirarte *et al.*, "Aircraft navigation GNSS-denied environments radio SLAM terrestrial signals opportunity," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 10, pp. 14 164–14 182, 2024.

[23] L. Bai, C. Sun, A. G. Dempster, H. Zhao, and W. Feng, "GNSS spoofing detection mitigation single 5G base station aiding," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 60, no. 4, pp. 4601–4620, 2024.

[24] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes Machine Learning*. The MIT Press, Nov. 2005.

[25] T. Pevnỳ, "Loda: Lightweight on-line detector anomalies," *Machine Learning*, vol. 102, pp. 275–304, 2016.

[26] Y. Zhao, Z. Nasrullah, and Z. Li, "PyOD: Python toolbox scalable outlier detection," *J. Mach. Learn. Res.*, vol. 20, no. 96, pp. 1–7, 2019.

[27] W. Liu and P. Papadimitratos, "Probabilistic detection GNSS spoofing using opportunistic information," in *Proc. IEEE/ION PLANS*, Monterey, CA, USA, Apr. 2023.

[28] M. Lenhart, M. Spanghero, and P. Papadimitratos, "Distributed mobile message level relaying/replaying GNSS signals," in *Proc. ION ITM*, Long Beach, CA, USA, Jan. 2022.

[29] K. Zhang and P. Papadimitratos, "On effects distance-decreasing attacks cryptographically protected GNSS signals," in *Proc. ION ITM*, Reston, VA, USA, Jan. 2019.

[30] W. Zhou, Z. Lv, X. Deng, and Y. Ke, "A new induced GNSS spoofing detection method based on weighted second-order central moment," *IEEE Sens. J.*, vol. 22, no. 12, pp. 12 064–12 078, 2022.

[31] M. Spanghero, F. Geib, R. Panier, and P. Papadimitratos, "GNSS jammer localization and identification with airborne commercial GNSS receivers," *IEEE Trans. Inf. Forensics Secur.*, vol. 20, pp. 3550–3565, 2025.

[32] W. Zhou, Z. Lv, G. Li, B. Jiao, and W. Wu, "Detection spoofing attacks global navigation satellite systems using Kolmogorov-Smirnov test-based signal quality monitoring method," *IEEE Sens. J.*, vol. 24, no. 7, pp. 10 474–10 490, 2024.

[33] A. Iqbal, M. N. Aman, and B. Sikdar, "A deep learning based induced GNSS spoof detection framework," *IEEE Trans. Mach. Learn. Commun. Netw.*, vol. 2, pp. 457–478, 2024.

[34] G. Michieletto, F. Formaggio, A. Cenedese, and S. Tomasin, "Robust localization secure navigation UAV formations GNSS spoofing attack," *IEEE Trans. Autom. Sci. Eng.*, 2022.

[35] L. Bai, C. Sun, A. G. Dempster, W. Feng, and C. Zhuang, "Robust GNSS spoofing detection UE maneuver GNSS-5G mmWave hybrid positioning system," *IEEE Sens. J.*, vol. 24, no. 13, pp. 21 237–21 253, 2024.

[36] F. Rothmaier, Y.-H. Chen, S. Lo, and T. Walter, "A framework GNSS spoofing detection combinations metrics," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 57, no. 6, pp. 3633–3647, 2021.

[37] D. Goward, "GPS backup demonstration projects explained," *GPS World*, 2022. [Online]. Available: https://www.gpsworld.com/gps-backup-demonstration-projects-explained/

[38] Q. D. Vo and P. De, "A survey fingerprint-based outdoor localization," *IEEE Commun. Surv. Tutor.*, vol. 18, no. 1, pp. 491–506, 2015.

[39] Z. Xing and J. Chen, "Constructing indoor region-based radio map without location labels," *arXiv preprint arXiv:2308.16759*, 2023.

[40] S. Huai, X. Liu, Y. Jiang, Y. Dai, X. Wang, and Q. Hu, "Multifeature-based outdoor fingerprint localization accuracy enhancement cellular network," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–15, 2023.

[41] C. Laoudias, A. Moreira, S. Kim, S. Lee, L. Wirola, and C. Fischione, "A survey enabling technologies network localization, tracking, navigation," *IEEE Commun. Surv. Tutor.*, vol. 20, no. 4, pp. 3607–3644, 2018.

[42] Mozilla, "Ichnaea," https://github.com/mozilla/ichnaea/blob/main/ichnaea/api/locate/mac.py, 2023.

[43] L. Bai, C. Sun, A. G. Dempster, H. Zhao, J. W. Cheong, and W. Feng, "GNSS-5G hybrid positioning based multi-rate measurements fusion proactive measurement uncertainty prediction," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–15, 2022.

[44] W. Song, H. Ding, L. Zhang, L. Chen, W. Guo, W. Lin, and S. Qi, "Performance verification GNSS/5G tightly coupled fusion positioning urban occluded environments smartphone," *GPS Solutions*, vol. 29, no. 1, p. 40, 2025.

[45] B. Mager, P. Lundrigan, and N. Patwari, "Fingerprint-based device-free localization performance changing environments," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 11, pp. 2429–2438, 2015.

[46] K. Shamaei and Z. M. Kassas, "Receiver design time arrival estimation opportunistic localization 5G signals," *IEEE Trans. Wirel. Commun.*, vol. 20, no. 7, pp. 4716–4731, 2021.

[47] S. Miura, L.-T. Hsu, F. Chen, and S. Kamijo, "GPS error correction pseudorange evaluation using three-dimensional maps," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 6, pp. 3104–3115, 2015.

[48] K. Jansen, N. O. Tippenhauer, and C. Pöpper, "Multi-receiver GPS spoofing detection: Error models realization," in *Proc. 32nd ACSAC*, Los Angeles, CA, USA, Dec. 2016.

[49] A. Schwaighofer, M. Grigoraş, V. Tresp, and C. Hoffmann, "GPPS: Gaussian process positioning system cellular networks," in *Proc. 16th NIPS*, Whistler, British Columbia, Canada, Dec. 2003.

[50] S. Narain, A. Ranganathan, and G. Noubir, "Security GPS/INS based on-road location tracking systems," in *Proc. IEEE S&P*, San Francisco, CA, USA, May 2019.

[51] J. Fan, *Local Polynomial Modelling Its Applications: Monographs Statistics Applied Probability 66*. Routledge, 1996.

[52] Jammertest, "The world's largest open jamming spoofing test," *Jammertest*, 2024. [Online]. Available: https://jammertest.no/about-2/

[53] UnwiredLabs, "The world's largest open database cell towers," *OpenCelliD*, 2023. [Online]. Available: https://opencellid.org/

[54] Bobzilla, Arkasha, and Uhtu, "Wigle.net. all networks. found everyone." *WiGLE*, 2023. [Online]. Available: https://wigle.net/

[55] "Evolved universal terrestrial radio access (E-UTRA); further advancements for E-UTRA physical layer aspects," 3GPP, Tech. Rep. TR 36.814 (Release 9), Mar. 2017. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2493

[56] S. Kuutti, S. Fallah, K. Katsaros, M. Dianati, F. Mccullough, and A. Mouzakitis, "A survey state-of-the-art localization techniques their potentials autonomous vehicle applications," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 829–846, 2018.

[57] M. Kok, J. D. Hol, and T. B. Schön, "Using inertial sensors position orientation estimation," *arXiv preprint arXiv:1704.06053*, 2018.

[58] A. Domi, "Particle filter," https://github.com/ArmandoDomi/Particle_Filter/, 2020.