Blind Spoofing Detection for Multi-Antenna Snapshot Receivers using Machine-Learning Techniques

J. Rossouw van der Merwe*, Ana Nikolikj†, Sebastian Kram*, Ivana Lukčin*, Gorjan Nadzinski†, Alexander Rügamer*, and Wolfgang Felber*

* *Fraunhofer Institute for Integrated Circuits IIS, Nuremberg, Germany*

† *Ss. Cyril and Methodius University Skopje, Macedonia*

BIOGRAPHY

J. Rossouw van der Merwe received his B.Eng (Hons) and M.Eng. degrees in Electronic Engineering from the University of Pretoria, South Africa, in 2014 and 2016, respectively. He joined the Fraunhofer Institute for Integrated Circuits IIS in 2016, where his research focus is directed towards on signal processing methods for robust and resilient GNSS receivers, including interference mitigation, spoofing detection, and array processing.

Ana Nikolikj is student at the Faculty of Electrical Engineering and Information Technologies, Ss Cyril and Methodius University in North Macedonia. Currently, she is finishing her Bachelor's in Computer System Engineering, Automation and Robotics. She plans to focus her study on machine learning and artificial intelligence.

Sebastian Kram received his Master degree in Electrical and Communication Engineering at the Friedrich Alexander University of Erlangen-Nuremberg (FAU), Germany, in 2016. In 2016 he joined the Locating and Communication Systems department at Fraunhofer IIS. Since 2020, he simultaneously has been working in the Navigation Group at the Chair for Information Technology (Communication Electronics) at FAU. His research interests are tracking algorithms, machine learning, and sensor data fusion. He focuses on radio signal-based adaptive cooperative positioning in adverse environments using both model- and data-driven methods.

Ivana Lukčin has been working at the Fraunhofer Institute for Integrated Circuits IIS since 2012. Until April 2017, she was working in the GNSS receiver field with the main emphasis on compressed sensing, snapshot positioning, and multi-path detection. Currently, she is working in the department "Precise Locating and Analytics" supporting the software design and development for different applications, e.g., for hybrid positioning with 5G and GNSS signals within simulations, emulations, and tests. Ivana Lukčin received her MSc. Degree in Mathematics from University of Zagreb, Croatia, in 2012.

Gorjan Nadzinski received his Ph.D. degree from the Faculty of Electrical Engineering and Information Technologies, Ss Cyril and Methodius University in North Macedonia in 2018, and he is currently working there as an assistant professor. His research interests include networked control systems, nonlinear control, robust control, secure communications, and machine learning.

Alexander Rügamer received his Dipl.-Ing. (FH) degree in Electrical Engineering from the University of Applied Sciences Würzburg-Schweinfurt, Germany, in 2007. Since then, he has been working at the Fraunhofer Institute for Integrated Circuits IIS in the Field of GNSS receiver development. He was promoted to Senior Engineer in February 2012. Since April 2013, he is head of a research group dealing with secure GNSS receivers and receivers for special applications. His main research interests focus on GNSS multi-band reception, integrated circuits, and immunity to interference.

Wolfgang Felber received his Dipl.-Ing. degree in Electrical Engineering in 2002 and his doctoral degree Dr.-Ing. in 2006 from Helmut-Schmidt-University of Federal Armed Forces Hamburg, Germany. Since 2014 he is head of the Satellite Based Positioning Systems department of Fraunhofer IIS, division Localisation and Networking in Nuremberg. The main topics in his department are hardware development of satellite navigation receivers for multiple or hybrid precise systems and secure applications. Additionally, since 2016 he is head of the business field localization at Nuremberg, which combines different localization technologies for industrial IoT applications.

ABSTRACT

Spoofing, the transmission of false global navigation satellite system (GNSS) signals, is a problem for a GNSS receiver. Therefore, a spoofing attack should be detected by a receiver to ensure the integrity of the position, velocity, and time (PVT) solution.

Detecting an attack is more difficult for a snapshot receiver, as temporal changes cannot be used as detection metrics. Further, if the spoofing attacker has access to the receiver, then ideal conditions for spoofing can be facilitated. This paper presents a machine learning (ML) approach of detecting a spoofing attack on a multi-antenna snapshot receiver. Blind detection methods are incorporated, as it is assumed that the antenna array could have been tampered with. The ML approaches include logistic regression (LR), K-nearest neighbors (KNN), naïve Bayes (NB), decision tree (DT) and support vector machine (SVM) algorithms. To ensure sufficient variance for training of the models, a spoofing simulation platform is developed and described in the paper. Training and testing is done on both simulated and real world data sets. Preliminary results indicate good classification, when training on the simulated data and validating on the real recorded data. Several of the ML methods have a classification f1-score exceeding 99 %. Even simple ML methods, like LR, KNN and NB, show good performance, indicating that the selected features are already adequately separating the spoofing and real data. This paper represents the first adaption of ML methods to snapshot based spoofing detection.

KEYWORDS

Spoofing detection, global navigation satellite system (GNSS), antenna array, snapshot processing, blind processing, anti-tamper, machine learning (ML), logistic regression (LR), K-nearest neighbors (KNN), naïve Bayes (NB), decision tree (DT) and support vector machine (SVM)

INTRODUCTION

Spoofing is the falsified transmission of global navigation satellite system (GNSS) signals [1–3]. It misleads a GNSS receiver to a wrong position, velocity, and time (PVT) solution. Therefore, it is a significant threat to the reliability and security of GNSS receivers. Spoofing is illegal and is associated with criminal, terrorist, and military activities. A spoofer may, for example, allow cheating with mobile position-based games (like *Pokémon-Go!*) [4], avoiding GNSS based automatic toll collection (ATC) [5], illegal fishing in protected maritime areas [6], vehicular theft [7], valuable goods theft [8], or disrupt a GNSS-synchronized electricity grid [9]. Therefore, it is essential to detect, mitigate, and remove spoofers for GNSS safety. If the spoofing attacker has direct access to the victim's GNSS receiver (including spoofing own equipment), then he can facilitate the ideal circumstances for spoofing, for example, by forcing a cold start of the receiver or electromagnetically isolate the receiver's GNSS antenna [10, 11]. Therefore, tampering increases the difficulty of detecting a spoofing attack significantly. This paper presents machine learning (ML) results to improve blind spoofing detection with snapshot-based GNSS receivers. Blind spoofing detection is the determination of a spoofing attack without sufficient knowledge of the receiver, for example if the receiver is calibrated or tampered with.

Snapshot processing, also known as cloud-GNSS [12] or server-based processing [13, 14], determines a PVT solution based only on a couple of milliseconds of data. Since only a snapshot of data is used, conventional processing techniques (e.g., signal tracking, symbol decoding, or navigation message extraction) are not possible. The snapshot PVT solution relies entirely on the acquisition results and external ephemeris information. However, the snapshot is typically processed remotely (hence, the terms cloud-GNSS and server-based) through software-defined radio (SDR) techniques. Therefore, the processing can be adapted as needed and use recursive processing, leading to flexible architectures [15]. Server-based processing can be applied to low size, weight, and power (SWAP) devices [16, 17], which cannot afford a full GNSS receiver for positioning. Applications such as remote sensing and animal tracking can benefit from this technique [18]. Mobile devices can also profit from this approach [19]. Another application for server-based processing is the verification of a receiver's position using encrypted GNSS signals [20, 21].

There are many spoofing detection techniques [2, 10, 22, 23]. However, many of them monitor the tracking– and PVT changes over time, and are consequently not applicable to snapshot-based receivers. Therefore, there are significant limitations with snapshot-based receivers against spoofing attacks. Still, one possibility is to use an array of antennas with spatial processing capabilities to determine the direction of the spoofing signals [24, 25]. It requires a snapshot front-end with multiple calibrated antennas and multiple synchronized channels for recording. However, if the spoofing attacker can manipulate the hardware, then an additional risk of calibration loss is perceived. Blind array processing counters this limitation through a relative comparison between the received signals [26].

Blind array processing with a snapshot receiver to detect spoofing signals was previously investigated [26, 27]. This paper extends the research by combining the previously identified detection metrics along with other snapshot processing outputs through ML. In the context of ML, a classification problem is set up using the detection metrics as features to classify whether the signals are spoofed or not. Supervised learning algorithms like logistic regression (LR), K-nearest neighbors (KNN), naïve Bayes (NB), decision tree (DT), and support vector machine (SVM) divide the feature space into areas corresponding to each of the classes by learning from given data sets. An exhaustive grid-search obtains the optimal combination of hyper-parameters for the ML algorithms. Then, the performance of the algorithms is compared with previously unseen data.

ML techniques have been used previously for spoofing detection [28–31]. However, these studies focused on conventional receivers, which observes the tracking outputs that change over time. This paper applies ML to snapshot-based receivers. Furthermore, special attention is paid to investigate if the algorithms can generalize over different data sets. Especially, using simulated

data for training and recorded data for testing is investigated in detail. Furthermore, the performance of different ML algorithms is compared.

The paper is structured as follows: the *Receiver Architecture* shows the receiver architecture and provides necessary foundations to snapshot processing and previous spoofing detection techniques. Next, the *Simulation Environment* with the various modules is presented. The *Machine Learning Approaches* discuss the ML approaches taken in the study. The *Experimental Setup* and *Results* present the evaluation and outcomes. Finally, the *Conclusion* is drawn in the last section.

RECEIVER ARCHITECTURE

The system is composed of three main sections. First, a multi-antenna snapshot receiver acquires the satellite vehicle (SV) signals spatially incoherent. From the output, the antenna steering vectors (ASVs) are estimated and beamforming for spatially coherent acquisition is applied. The improved acquisition results determine the PVT solution for the snapshot. Lastly, ML methods extract features from the ASV estimation, blind acquisition and PVT calculation processing blocks, and use these for spoofing detection. Figure 1 shows the receiver architecture, from the snapshot input to the final spoofing detection. Subsequent sections discuss the system components in more detail.

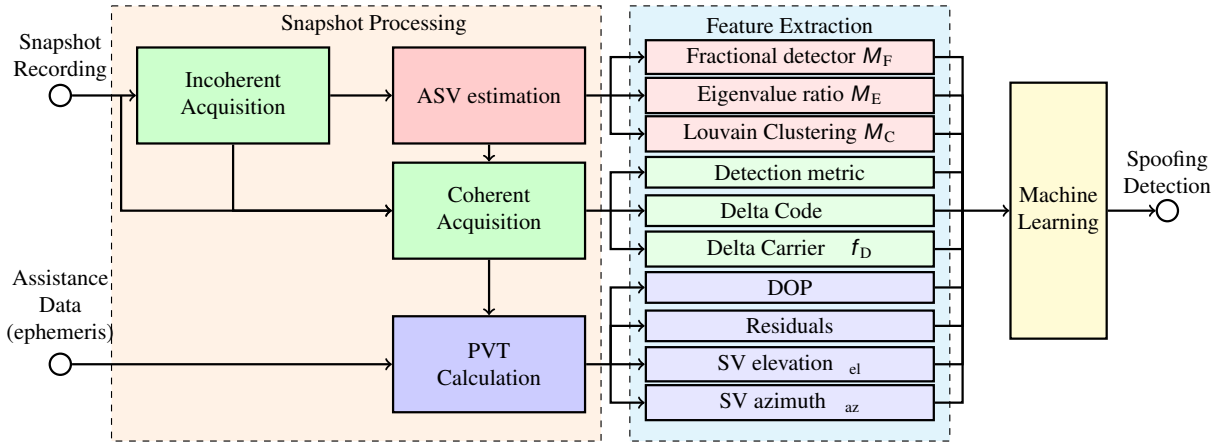


Figure 1: Acquisition methodology

Snapshot Acquisition

A blind acquisition technique, which firstly implements incoherent spatial integration to find the correct correlation peak, secondly estimates the ASV, and thirdly does coherent spatial integration, has already been developed [32]. This method is considered blind, as it is independent of direction-of-arrival (DOA) estimation. Consequently, no calibration or information about the array configuration or orientation is required.

The incoherent acquisition correlates each antenna signal with an SV's pseudo-random noise (PRN) code separately then combines the output incoherently. The peak within the combined correlation function relates to the estimated code phase $^{(I)}$ and carrier Doppler $f_D^{(I)}$. The peak values are extracted and used to estimate the ASV. The ASVs determine three spoofing detection metrics, which are discussed in a later section. The ASVs coherently combine the antenna signals (i.e., beamforming), before the next correlation with the same PRN, for the second acquisition. It results in a coherent code phase $^{(C)}$ and carrier Doppler $f_D^{(C)}$ estimations. The delta code and delta carrier f_D between the two acquisitions are extracted as features for later ML processing (see Figure 1):

$$= \quad ^{(I)} \quad ^{(C)} \tag{1}$$

$$f_D = f_D^{(I)} \quad f_D^{(D)} \tag{2}$$

The acquisition detection metric is also extracted as a feature. The current implementation uses the peak-to-next-peak detector, also referred to as a ratio detection (RD) [33]. A limitation of this method is that the estimation of the array steering vector is based upon the results of incoherent acquisition. These values may contain multi-path or cross-correlation components from other signals, which can obscure the estimation process. In turn, the erroneous ASV may form a beam that does not sufficiently suppress these unwanted signal components.

Snapshot Positioning

The PVT algorithm uses the coherent code phase $^{(C)}$ and carrier Doppler $f_D^{(C)}$ estimations together with the ephemeris information for each SV. The PVT calculation uses a classical least mean squares (LMS) approach [34]. A snapshot receiver cannot decode the navigation message, due to the limited data, and does not have a time of transmit (ToT) as a reference. Therefore, ambiguities exist, and the receiver time should also be estimated. As such, a minimum of five SVs is required for the PVT algorithm. The residuals of the LMS optimization for the PVT solution is used as a feature. However, if only five SVs are visible, then there are no residuals available.

Once a PVT solution is computed, the receiver and SV positions are known [35]. The elevation angle $_{el}$ and azimuth angle $_{az}$ for each SV relative to the receiver can then be determined. Further, the dilution of precision (DOP) for the constellation can be calculated.

Snapshot-Based Spoofing Detection

Spoofing detection with an array of antennas has already been proven to be successful and reliable [25, 36–38]. However, this requires a calibrated array with known receiver phase offsets and antenna orientation. Further, these methods are DOA based and estimate the direction to each satellite. It is a limitation as it requires calibration of the antenna array, which increases the cost of the receiver. Further, tampering with the array counteracts calibration. Blind array processing with a snapshot receiver to detect spoofing signals was previously investigated [27]. In this study, three detection metrics, which consider a constellation of satellites, are derived. These metrics include a fraction of detection approach M_F, where the fraction of satellites pairs, which are detected as spoofed are determined; a maximum to mean eigenvalue ratio detector M_E; and a Louvain clustering approach M_C. These three metrics rely on a correlation matrix for the ASVs of each acquired satellite's signal. This subsection summarizes the three metrics.

The correlation matrix $\mathbf{C}_{\text{norm}}$, on which all three metrics are based, is required. First, all the estimated ASV $\hat{\mathbf{a}}_n$ are stacked in a matrix $\hat{\mathbf{A}}$:

$$\hat{\mathbf{A}} = [\hat{\mathbf{a}}_1; \hat{\mathbf{a}}_2; \dots; \hat{\mathbf{a}}_{N_{SV}}] \; ; \tag{3}$$

where $\hat{\mathbf{a}}_n$ the N_{el} sized column vector for the array steering vector for the n-th SV. The constellation consists of a total of N_{SV} satellites. $\hat{\mathbf{A}}$ is a N_{el} N_{SV} matrix. The matrix is correlated to get a non-normalized correlation matrix $\mathbf{C}$:

$$\mathbf{C} = \hat{\mathbf{A}}^H \quad \hat{\mathbf{A}} \; ; \tag{4}$$

where $(\,)^H$ is the Hermitian transpose of the signal, and " " is the matrix multiplication. Each element in the matrix is equivalent to the dot product between each pair of steering vectors. $\mathbf{C}$ is an N_{SV} N_{SV} Hermitian matrix. The magnitude values of the auto-correlations need to be determined to normalize the matrix:

$$\mathbf{c} = \sqrt{\text{diag}(\mathbf{C})} \; ; \tag{5}$$

where $\mathbf{c}$ is a column vector containing the magnitude values, and diag( ) takes the diagonal of a matrix. The normalized correlation matrix $\mathbf{C}_{\text{norm}}$ is calculated as:

$$\mathbf{C}_{\text{norm}} = \langle f\mathbf{C}g \quad \mathbf{c} \quad \mathbf{c}^T \rangle^{\circ -1} \; ; \tag{6}$$

where $\langle f\, g$ takes the real component of the correlation, is the Hadamard product, and $(\,)^{\circ -1}$ is the Hadamard inverse. $\mathbf{C}_{\text{norm}}$ is also a Hermitian matrix. Each element of this matrix has the form:

$$C_{\text{norm}}(n;m) = \frac{\langle \hat{\mathbf{a}}_n^* \quad \hat{\mathbf{a}}_m}{k\hat{\mathbf{a}}_n k\, k\hat{\mathbf{a}}_m k} \tag{7}$$

where $k\mathbf{a}_n k$ is the magnitude of the n-th ASV.

The first metric considers the fraction of detections in a detection matrix $\mathbf{D}$. This matrix depends on the correlation matrix $\mathbf{C}_{\text{norm}}$:

$$D(n;m) = \begin{cases} 1; & \text{if } C_{\text{norm}}(n;m) \quad _f \\ 0; & \text{otherwise} \end{cases} \; : \tag{8}$$

where $_f$ is the threshold value for the single detections. The threshold is selected as [27]:

$$_f = \cos(15^\circ) \quad 0{:}966 \tag{9}$$

Note that the diagonal of the detection matrix $\mathbf{D}$ is always unitary, due to the correction process, and needs to be removed. The fraction of detections metric M_F is:

$$M_F = \frac{\sum_{n;m} D(n;m) \quad N_{SV}}{N_{SV}(N_{SV} \quad 1)} \; : \tag{10}$$

The second metric uses the Eigenvalue decomposition:

$$\mathbf{C}_{\text{norm}} = \mathbf{Q}_{\text{c}} \quad {}_{\text{c}}\mathbf{Q}_{\text{c}}^{-1}\,; \tag{11}$$

where $\mathbf{C}_{\text{norm}}$ is the matrix to be decomposed, $\mathbf{Q}_{\text{c}}$ is a matrix where each column is a unique Eigenvector, and $_{\text{c}}$ is a diagonal matrix containing the Eigenvalues. The Eigenvalues are in descending order. The ratio of the maximum Eigenvalue to the mean of the Eigenvalues is used as a detection metric [39]:

$$M_{\text{E}} = \frac{k \quad {}_{1}k}{\frac{1}{N_{\text{SV}}} \sum_{i} k \quad {}_{i}k} = \frac{k \quad {}_{\text{c}}(1;1)k}{\frac{1}{N_{\text{SV}}} k\text{diag}(\quad {}_{\text{c}})k}\,; \tag{12}$$

where $_{1}$ is the largest Eigenvalue, $_{i}$ is the i-th Eigenvalue from the matrix, and N_{SV} is the number of Eigenvalues.

The third metric uses clustering methods. The Louvain clustering algorithm is an iterative algorithm that optimizes the modularity of a data set [40, 41]. The data set consists of several nodes and edges (links) between these nodes. Highly connected nodes are grouped into communities by this algorithm. The modularity of the data set is an indication of the connectedness of the nodes. Hence, high modularity indicates that the nodes can be grouped efficiently into communities, whereas low modularity indicates that the data is highly random and not easily grouped into communities. This algorithm requires a matrix which contains the weight of the edges between each of the nodes. Higher values in this matrix correspond to stronger links between the nodes. Further, this algorithm requires a vector $\mathbf{v}$ which contains a list of the different communities.

Maximizing the modularity of the data set achieves useful clustering. The modularity Q is defined as:

$$Q = \frac{1}{2m} \sum_{i;j} \quad C_{\text{norm}}(i;j) \quad \frac{k(i)k(j)}{2m} \qquad (v(i);v(j))\,; \tag{13}$$

where m is the total connectedness between all of the nodes:

$$m = \frac{1}{2} \sum_{i;j} C_{\text{norm}}(i;j)\,; \tag{14}$$

and k_i is the connectedness of the i-th node to the other nodes and is defined as:

$$k(i) = \sum_{j} C_{\text{norm}}(i;j)\,: \tag{15}$$

Lastly, $(\ )$ is a modified Dirac function and is defined as:

$$(v(i);v(j)) = \begin{array}{ll} 1; & \text{if } v(i) = v(j) \\ 0; & \text{otherwise} \end{array}\,: \tag{16}$$

At the start of the clustering process, all of the nodes are in separate communities. The initial modularity Q for the system is calculated. Systematically, each node is moved to different communities. The community, which results in the most significant increase of system modularity Q for each node, is selected as the new community for that node. When all nodes are adjusted to the optimal community, a new system is formed by grouping nodes from the same community. Thereby, implicitly reducing the dimensions of the matrix $\hat{\mathbf{A}}$ in each iteration. After that, the process is repeated until no improvement of the modularity Q is achievable. As the algorithm iterates, similar communities are clustered together to form new and more populous communities. The clustering metric is simply the modularity:

$$M_{\text{C}} = Q \tag{17}$$

These three metrics are applied to the correlation matrix $\mathbf{C}_{\text{norm}}$, which is derived from the ASV matrix $\hat{\mathbb{A}}$. It determines how similar the system of SVs are. However, the approach could also use the unit vector to each SV to determine how similar the SV orientations are. Therefore, the metrics based on the ASVs ($M_{\text{F}}^{(\text{AV})}$, $M_{\text{E}}^{(\text{AV})}$, and $M_{\text{C}}^{(\text{AV})}$), and the SVs ($M_{\text{F}}^{(\text{SV})}$, $M_{\text{E}}^{(\text{SV})}$, and $M_{\text{C}}^{(\text{SV})}$), are used for features for the machine learning.

Machine Learning Techniques

The ML is the new module introduced in this paper. The motivation of ML is to combine the detection metric with other receiver outputs to improve the detection capabilities optimally using more information. The current objective is to develop a method for detecting straightforward spoofing attacks and to establish a solid foundation. However, once the framework is in place, more complex problems are solvable, like a partially spoofed constellation, severe multi-path detection, or spoofing attack type classification. The information analysis and ML approach are described in more detail in a later section.

SIMULATION ENVIRONMENT

This section describes the simulation environment and models. A limitation of previous work [27] was limited scenarios where evaluated. It results in a sparse representation of the scenarios and with issues with ML model training. A snapshot receiver for different scenarios is simulated and used for the training of the ML methods in later sections to address this issue. Further, a high degree of statistical variance in the simulation is required to avoid biases during ML training or risk of over-fitting to the data set. It also makes the problem more generic as it decouples the training from a specific receiver and scenario. Lastly, the simulation model warrants more detail, as the assumptions made in simulations directly impact the results. It emphasizes the need for clarity in the simulation procedure.

Random Variables

In the simulation environment, several different random variables (RVs) introduce statistical variance [42]. Most commonly uniform distributed ($U(a;b)$) and Gaussian distributed ($G(\ ;\ )$ RVs are used:

$$U(a;b): \quad f_u(x) = \begin{cases} \frac{1}{b-a} & \text{if } a \quad x \quad b \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

$$G(\ ;\ ): \quad f_g(x) = \frac{1}{\sqrt{2}} e^{-\frac{1}{2}(\frac{x-}{})^2} \tag{19}$$

where $f_u(x)$ and $f_g(x)$ are the associated probability density functions (PDFs) for these distributions. A chi-squared RV ($^2(k)$) is a compounded distribution. It is the sum of k squared Gaussian RVs, and has a PDF of $f\ (x;k)$:

$$^2(k) = \sum_k G(0;1)^2: \quad f\ (x;k) = \begin{cases} \frac{x^{k=2-1}e^{-x=2}}{2^{k=2}\ (k=2)} & \text{if } 0 \quad x \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

Lastly, a Rayleigh RV ($R(a)$) is the square-root of a chi-squared RV of order $k = 2$, and has a PDF of $f_r(x)$:

$$R(a) = a \sqrt{^2(2)}: \quad f_r(x) = \begin{cases} \frac{x}{^2} e^{\frac{-x^2}{2\ ^2}} & \text{if } 0 \quad x \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

Scenario Selection

Each scenario requires a satellite constellation for the transmitted signals and a receiver position. A satellite constellation, using the Keplerian orbital parameters from the Galileo almanac [43], is generated. The time of receive (ToR) is a uniform distributed RV, starting with the almanac publishing date and ending 10 days later. The period relates to the time it takes the Galileo constellation to be at the same locations relative to earth at the same time of day.

A receiver position depends on three RVs. The longitude $_{\text{long}}$ is a uniform distributed RV:

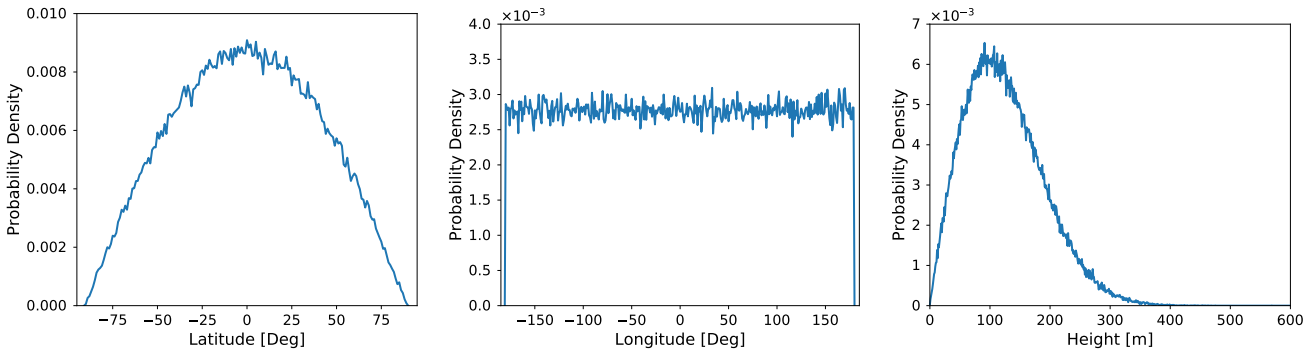
$$_{\text{long}} \quad U(\ 180^\circ; 180^\circ) \tag{22}$$

The latitude $_{\text{lat}}$ is a transformed from a uniform distributed RV:

$$_{\text{lat}} = \frac{180^\circ}{}\cos^{-1}(x_{\text{lat}}) \quad 90^\circ \text{ and } x_{\text{lat}} \quad U(\ 1;1) \tag{23}$$

As a result, the receiver has an equal probability of being on any location on the earth's surface[1]. The altitude a_{alt} is simulated as Rayleigh distribution with a scale parameter of 100 m: $a_{\text{alt}} \quad R(100\ \text{m})$. The receiver position and SV position determine the elevation angle $_{\text{el}}$ and azimuth angle $_{\text{az}}$ for each satellite. Fig 2 shows the simulated position PDFs. The receiver specific properties are determined in the next section.

[1] Assuming the earth is spherical. Considering the earth as an ellipsoid would improve this model, but it is adequate for the current implementation.



(a) Latitude (b) Longitude (c) Altitude

Figure 2: Simulated position statistics

Receiver Selection

The simulated receiver has an array of antennas. For GNSS, a uniform circular array (UCA) or a uniform circular array with a center element (UCA-CE) is preferred, as these arrays allow two-dimensional (2-D) beamforming and a stable phase center. The three-dimensional (3-D) position vector of each antenna element $\mathbf{d}_n$ is determined relative to the center of the array. For a UCA consisting of N_{el} elements, the n-th element position vector $\mathbf{d}_n(r)$ is:

$$\mathbf{d}_n(r) = \quad r \sin \quad 2 \quad \frac{n}{N_{\text{el}}} \quad ; r \cos \quad 2 \quad \frac{n}{N_{\text{el}}} \quad ; 0 \quad ; n \; 2 \; f0; \; : : : \; ; N_{\text{el}} \quad 1g \tag{24}$$

A UCA-CE has the same structure, but an extra element as added at the origin. The position vector $\mathbf{d}_n(r)$ for a UCA-CE is:

$$\mathbf{d}_n(r) = \begin{matrix} (& \text{D} & & & \text{E} \\ & r \sin \quad 2 \quad \frac{n}{N_{\text{el}}-1} & ; r \cos \quad 2 \quad \frac{n}{N_{\text{el}}-1} & ; 0 & \text{if } 0 \quad n < N_{\text{el}} \quad 1 \\ & h0; \; 0; \; 0i & & & \text{if } n = N_{\text{el}} \quad 1 \end{matrix} \tag{25}$$

For the simulation, UCAs with three to six elements and UCA-CEs with four to seven elements are selected. Figure 3 displays the different array types and the associated likelihood of being selected during the scenario simulation. Most commonly, four-element and six-element UCAs are selected, with 35 % and 45 % probability, respectively.

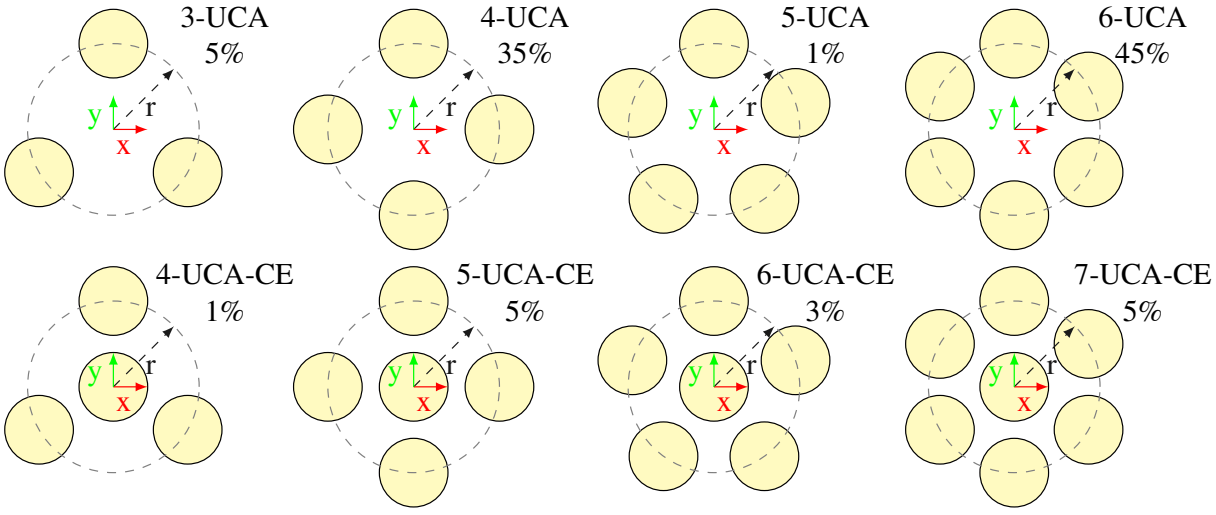


Figure 3: UCA and UCA-CE arrays and the percentage of occurrence in simulation

The radius of the outer ring r is, on average, set to a half-wavelength spacing for the center of the GNSS L1 band (=2 = 95.1 mm), as this is a popular selection for GNSS antenna arrays. A uniform RV is introduced to account for different radius selections used for antenna array manufacturers:

$$r = r + r_u = \frac{}{2} + r_u; \; r_u \; U(\quad 10 \text{ mm}; 10 \text{ mm}) \tag{26}$$

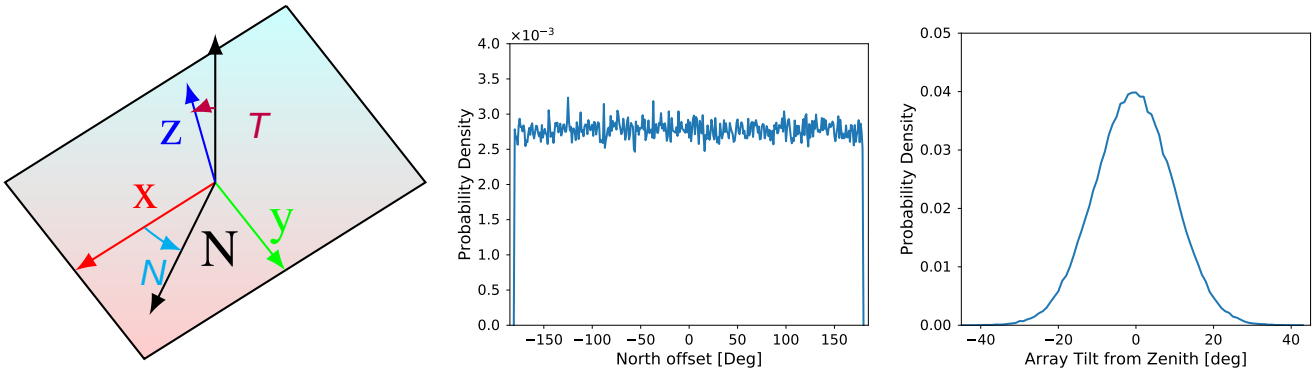
Each antenna element has an additional position uncertainty of 10 mm per element. It represents manufacturing and mounting tolerances and errors, which may move the phase center of the antenna. The resultant position for the n-th antenna element is consequently randomized:

$$\mathbf{d}_n = \mathbf{d}_n \quad \frac{}{2} \quad + \{d_x;\ d_y;\ d_z\};\ \{d_x; d_y; d_z\} \quad G(0;\ 10\ \text{mm}) \tag{27}$$

The array orientation is determined by the north offset $_N$, and the array tilt as an angle from the zenith $_T$. The north offset is the angle between the reference antenna and the geographic north. The angle relates to the difference between the angle of arrival (AOA) and the DOA for a received signal. It is modeled as a uniform distributed RV, $_N$ $U(\ 180; 180)$ [deg]. It represents an antenna array that can be mounted on a mobile platform. The array tilt angle is modeled as a zero-mean Gaussian RV $_T$ $G(0; 10°)$. It represents a vehicular application over some uneven ground. The orientation can be added to the position vector by using appropriate rotation matrices (e.g., degree rotation around the x-axis uses the rotation matrix $\mathbf{R}_x(\ )$):

$$\hat{\mathbf{d}}_n = \mathbf{d}_n \quad \mathbf{R}_z(\ _N) \quad \mathbf{R}_x(\ _T) \tag{28}$$

Figure 4 shows the diagram and the generated PDFs for the orientations.



(a) Orientation diagram

(b) Array north offset $_N$

(c) Array tilt from zenith $_T$

Figure 4: Array orientation

Once the antenna positions $\hat{\mathbf{d}}_n$, and the SV's position are known ($_{\text{el}}$ and $_{\text{az}}$), the array signal model can be generated. The received signal model is:

$$\mathbf{y}(t) = \mathbf{C}_{\text{cp}} \quad \mathbf{A} \quad \mathbf{x}(t) + \mathbf{n}(t) \tag{29}$$

$\mathbf{y}(t)$ is the received signal and is a N_{el} element column vector, and $\mathbf{n}(t)$ is a noise vector of the same size consisting of additive white Gaussian noise (AWGN). $\mathbf{x}(t)$ is the received signal from each SV and is an N_{SV} element column vector. The array matrix $\mathbf{A}$ is the complex gain for each signal at each antenna element and is a N_{el} N_{SV} matrix. The coupling matrix $\mathbf{C}_{\text{cp}}$ is a N_{el} N_{el} square matrix and represents the coupling between the antenna elements.

Each column of the array matrix $\mathbf{A}$ is the ASV to a specific SV. The ASVs are estimated by a receiver for beamforming purposes and forms part of the spoofing detection. In the non-spoofing case, all of the ASVs relate to the respective SVs. However, in the spoofed case, all of the ASVs are the same. The theoretical model for a calibrated array matrix $\mathbf{A}$ is [44]:

$$A(n; m) = e^{-j\mathbf{k}(\ _{\text{az}}(m);\ _{\text{el}}(m))\cdot\hat{\mathbf{d}}_n} \tag{30}$$

where $\mathbf{k}(\ _{\text{az}};\ _{\text{el}})$ is the wave vector and is a function of the wavelength and the DOA for each SV:

$$\mathbf{k}(\ _{\text{az}};\ _{\text{el}}) = \frac{2}{} \{\cos\ _{\text{az}} \cos\ _{\text{el}};\ \sin\ _{\text{az}} \cos\ _{\text{el}};\ \sin\ _{\text{el}}\} \tag{31}$$

The simulated matrix $\hat{\mathbf{A}}$ includes additional receiver effects. It assumes that the receiver switches on, then takes a snapshot, and finally switches off again. With each new power-cycle, the digital down-converters (DDCs) of the radio-frequency front-end (RFFE) have a relative phase-offset, and as only limited data is recorded, no calibration is possible. Further, a low-cost receiver is likely not calibrated (i.e., RFFE phase offsets and antenna phase offsets) as this dramatically increases costs. Lastly, temporal changes to the receiver, like temperature changes and aging, may also induce new offsets. Therefore, considering all these effects, each antenna element is simulated to have an arbitrary phase offset e_n $U(0; 2\ )$. This phase-offset e_n is constant throughout the snapshot and is the same for every signal received from the same antenna.

The antenna also contributes to the received signal power. A typical GNSS antenna forms a main lobe towards the zenith [45]. It suppresses low elevation signals to reduce multi-path effects. A patch-like antenna is assumed for the simulation with a sinc-function-based antenna pattern:

$$G_{\text{ant}}(\ _{\text{el}}) = G_{\text{ant,main}} + 10\log_{10} \quad \text{sinc}^2 \quad \frac{90 \quad _{\text{el}}}{2B} \tag{32}$$

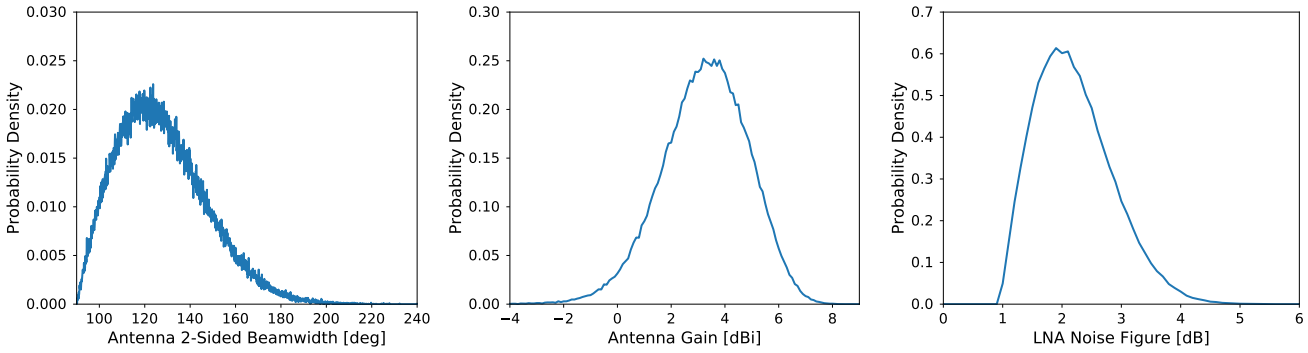
$G_{\text{ant}}(\ _{\text{el}})$ is the total antenna gain, $G_{\text{ant,main}}$ is the gain of the main lobe, and B is the two-sided -3 dB antenna beamwidth. The beamwidth is constant in an antenna array, but differs for each scenario according to a chi-squared RV:

$$B = 2 \quad (45^\circ + 15^\circ \quad b_x)\ ;\ b_x \qquad ^2(2) \tag{33}$$

The gain of the main lobe $G_{\text{ant,main}}$ depends on the efficiency of the antenna:

$$G_{\text{ant,main}} = 10\log_{10} \quad \frac{2}{1 \quad \cos(B=2)} \quad ; \qquad G(0{:}6; 0{:}14) \tag{34}$$

The efficiency relates to the design, composition, and manufacturing quality of the antenna. External factors, like the impact of the mounting structure or the use of a radome, also affect efficiency. Figure 5 shows the PDFs of the antenna properties.



(a) Antenna Beamwidth (b) Antenna Gain (c) LNA Noise figure

Figure 5: Simulated position statistics

The simulated matrix $\hat{A}$ considers these effects:

$$\hat{A}(n;m) = G_{\text{ant}}(\ _{\text{el}}(m)) \quad \mathrm{e}^{-\mathrm{j}\mathbf{k}(\ _{\text{az}}(m);\ _{\text{el}}(m))\cdot\hat{\mathbf{d}}_n + e_n} \tag{35}$$

The coupling matrix $\mathbf{C}_{\text{cp}}$ for an ideal system is an identity matrix, i.e., no coupling. For this simulation, the coupling matrix is defined as follows for a UCA:

$$C_{\text{cp}}(n;m) = \begin{cases} 1 & \text{if } n = m \\ 0{:}1 \quad e^{jc_x} & \text{if } jn \quad mj = 1; c_x \quad U(0;2\ ) \\ 0{:}01 \quad e^{jc_x} & \text{if } jn \quad mj = 1; c_x \quad U(0;2\ ) \\ 0 & \text{otherwise} \end{cases} \tag{36}$$

Notice that the nearest antenna has 10 dB isolation with a random phase.

Each antenna has a low-noise amplifier (LNA). The LNA amplifies the signal before the receiver can further process it. As the LNA is the first amplifier in the receiver it has the most significant influence to the received noise power ^2_n. The noise power of the receiver depends on the LNA's noise factor N_{F}[2], the receiver temperature T, the receiver bandwidth BW, and Boltzmann's constant k:

$$^2_n = N_{\text{F}} \quad T \quad \text{BW} \quad k \tag{37}$$

The noise factor N_{F} is modeled as:

$$N_{\text{F}} = 10^{\frac{1+x_n}{10}};\ x_n \qquad R(1) \tag{38}$$

The receiver temperature is simulated as a Gaussian RV T $G(18^\circ\text{C}; 7^\circ\text{C})$. The receiver bandwidth BW is selected to be 10.125, 20.25, 40.5, and 81 MHz, with 25 %, 25 %, 40 % and 10 % likelihoods, respectively.

[2]The noise figure is the decibel representation of the noise factor.

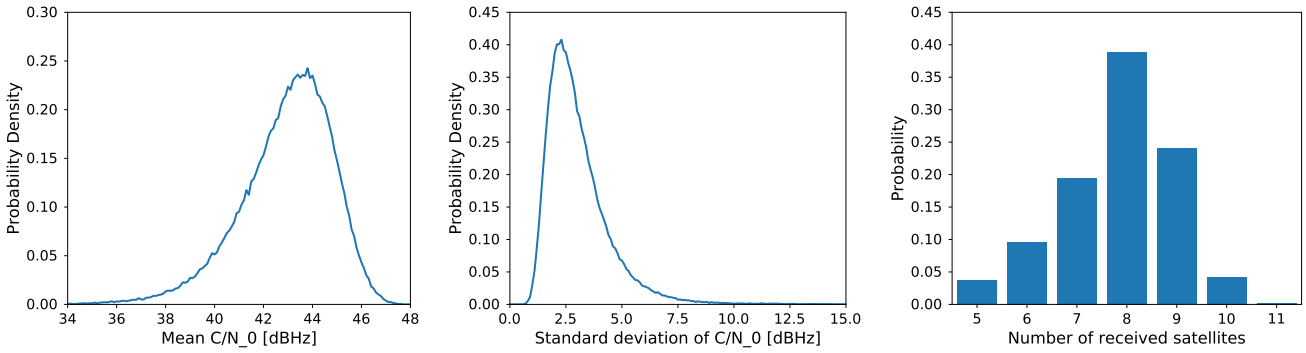
Path-Loss Effects

The constellation information, along with the receiver platform information, determines the path-loss of the signals between the SV and the receiver [46]. Further, combining the path-loss with the noise power determines the theoretical carrier-to-noise density ratio (C/N_0) for a scenario. The path-loss L_{path} is determined with Frii's free space loss equation [47]:

$$L_{\text{path}}[\text{dB}] = P_r \quad P_t + G_t + G_r + 20 \log \frac{}{4 \, d_s} \tag{39}$$

P_t and P_r are the transmitted and received signal powers, G_t and G_r are the transmit and receive antenna gains, is the wavelength of the signal, and d_s is the separation distance between the transmitter and the receiver. It is a simplistic model, as it does not include multi-path, shadowing, scintillation, or atmospheric effects. It can be improved to use more effective models and is considered a future expansion of the simulation platform.

Figure 6 shows the summarized statistics of the C/N_0, along with the number of visible SVs for the simulation. An elevation mask of 5 ° filters the number of visible satellites.



(a) Mean C/N_0 (b) Standard deviation of C/N_0 (c) Number of visible satellites

Figure 6: Simulated position statistics

Processing Simulation

The core of snapshot processing is the acquisition of the SVs. In the acquisition processing, the received signals are correlated with each SV's PRN code. The correlation, even if efficient parallel search methods are employed, is highly processing intensive. The full signal is not generated to reduce simulation duration significantly. Only a single chip of the PRN code for each SV and antenna is generated. As an example of the efficiency, it results in a 4096-factor reduction of the generated signal for Galileo E1B/C. The simulation primarily focuses on Galileo E1B/C, which uses a composite binary offset carrier (CBOC) modulation. The CBOC consists out of two binary offset carrier (BOC) modulations. A BOCc(1,1) and a BOCc(6,1). Due to bandwidth limitations applied to snapshot processing, often only the BOCc(1,1) component is available. Therefore, only the BOCc(1,1) is used in the simulations.

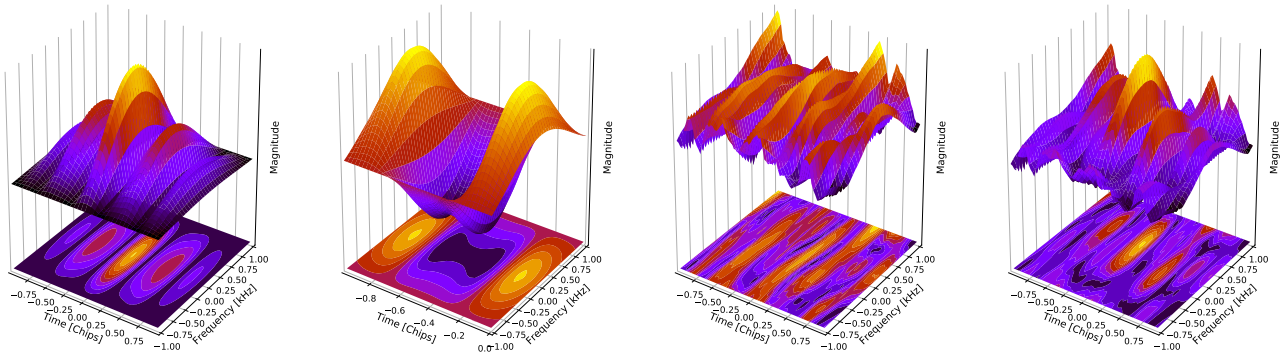
The auto-correlation function (ACF) is the output of the correlation. Figure 7 shows the simulated ACF. Here a BOCc(1,1) signal is simulated, with a coherent integration time $_{\text{int}}$ of 1 ms and a theoretical C/N_0 of 40 dBHz. The ACF can be generated by correlating the single chip. On the contrary, noise generation is more complicated. A gird of cells is generated, each an AWGN RVs, which represent different time offsets (chips) and frequency offsets (carrier Dopplers). In an actual receiver, the acquisition processing causes each neighboring cells to be correlated. The noise grid is filtered with a 2-D mask (See Figure 7b), to simulate this behavior. In the time dimension, the mask represents a single chip (including the bandwidth limitation of the receiver fronted), in the frequency dimension the mask represents the Doppler response:

$$h(t; f) = \begin{array}{ll} x_{\text{Chip}}^{(\text{BL})}(t) \quad \text{sinc}\,(f \;_{\text{int}}) & \text{if } jtj <= \;_{\text{int}};\; jfj <= \frac{2}{\text{int}} \\ 0 & \text{otherwise} \end{array} \tag{40}$$

where $_{\text{int}}$ is the coherent integration time, and $x_{\text{Chip}}^{(\text{BL})}(t)$ is the band-limited modulated chip derived as:

$$x_{\text{Chip}}^{(\text{BL})}(t) = x_{\text{Chip}}(t) \quad \text{sinc} \quad t\frac{\text{BW}}{f_s} \tag{41}$$

where BW is the analog bandwidth of the receiver, f_s is the temporal sample-rate of the generated grid, and ( ) ( ) denotes the convolution function (i.e., filter).



(a) BOC signal ACF (b) Noise correlator function (c) Correlated noise signal (d) Noisy BOC ACF

Figure 7: Simulated position statistics

For each scenario, N_{SV} N_{el} N_{inco} grids are generated. The specific SV's C/N_0 determines the noise power for the associated blocks, and the array steering matrix $\mathbb{A}$ determines the complex magnitude for each antenna element and SV. The N_{inco} number of incoherent epochs depends on the snapshot length and integration time $_{int}$.

For incoherent acquisition, the N_{el} N_{inco} grids are added together incoherently. For coherent acquisition, N_{el} sets of grids are combined through beamforming using the ASV. After that, the N_{inco} beamformed signals are combined incoherently (i.e., temporal incoherent acquisition). The processed girds are used to find the peak, which determines the estimated code phases ($^{(I)}$ and $^{(C)}$) and carrier Doppler ($f_D^{(I)}$ and $f_D^{(C)}$). It allows the acquisition errors to be simulated and to propagate down the receiver chain to the PVT. Therefore, it facilitates accurate receiver modeling.

MACHINE LEARNING APPROACHES

Apart from the model-based approach that the derived metrics [27] are based on, they can also be understood as features in a ML sense. Therefore, the recorded data set is interpreted as list of feature vectors $\boldsymbol{x_i}$ containing the snapshot observations aligned with a binary class labels y 2 (Spoofed; not Spoofed). Data sets are then used to train classifiers to decide between the labels based on observed feature vectors. The data imply a decision rule, dividing the feature space into regions corresponding to each of the classes. The ML pipeline is depicted in Fig. 8. Suitable subsets of the data are selected for testing, training and validation.

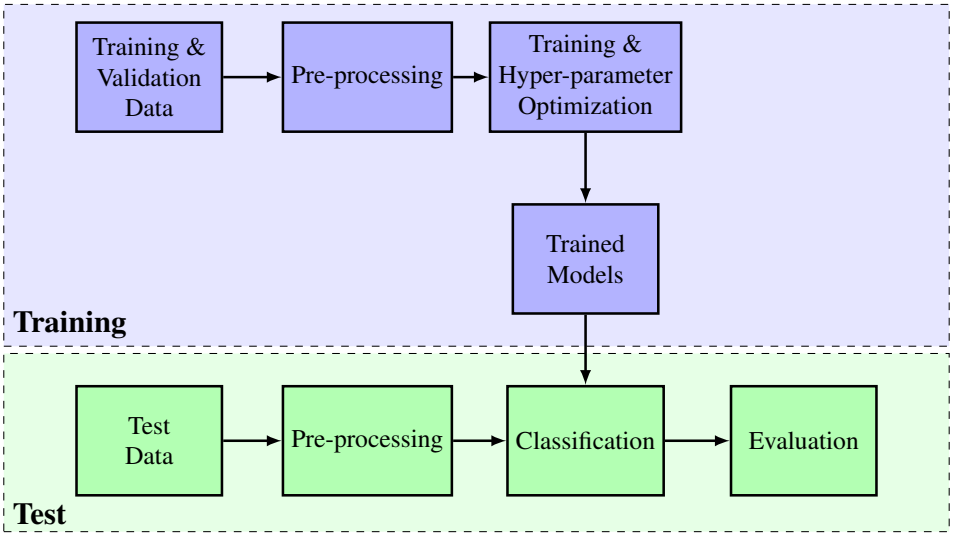


Figure 8: The ML pipeline.

The specific setups used for validation is presented in the experimental setup section.

First, the data are pre-processed by removing incomplete data and scaling the features to uniform minima and maxima. Next, the data set is divided into 70 % training and 30 % test data. For hyper-paramter optimization [48], further division on train and validation sets is made based on stratified k-fold cross-validation [49], where the number of folds is chosen to be $k = 5$. Hereby,

the test data set is split into k consecutive folds where each fold is then used once as a validation while the k 1 remaining folds form the training set. There are many ML models [50], but for this paper LR, KNN, NB, DT and SVM are considered.

LR is a statistical model in which the probabilities describing the possible outcomes of a single trial are modeled using a logistic function [51, 52]. Typically a sigmoid function is used. The sigmoid function is an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1. A threshold on the sigmoid function decides the class to which the feature vector belongs to. LR is straightforward to implement, interpret and efficient to train. Also it does not require too many computational resources and data preparation. The main limitation is the assumption of linearity between the dependent variables and the independent variables or features. Removing features which are uncorrelated to the output variable does not improve the algorithm. Further, removing correlated attributes have the same effect. Therefore, feature engineering plays an important role in regards to the performance.

A KNN classifier is a simple algorithm that stores all available samples in memory and assigns a new feature vector $\mathbf{x}_i$ to the class the majority of its nearest neighbors belong to [53, 54]. The closest neighbors are the values which are most similar to the new feature set, and are determined using a distance function. For example, the Euclidean distance between two feature sets is often used:

$$D_{i;j} = k\mathbf{x}_i \quad \mathbf{x}_j k^2 \tag{42}$$

The nearest neighbor is the feature set $\mathbf{x}_j$, which minimizes the distance $D_{i;j}$. In the simplest case (i.e., $k = 1$), the class of the nearest neighbor determines the class of the new feature set. If more neighbors are considered, a voting process selects the class from the k nearest neighbors. The advantage of such a memory-based approach is that the classifier immediately adapts as new training data are introduced, as all the observations are kept permanently. Furthermore, to optimize the classification performance a suitable value of the the number of neighbors k must be determined. Since KNN performs on-the-spot learning, a well-tuned k can model complex decision spaces having arbitrarily complicated decision boundaries. However, the comparison of the new feature set with the entire data set for each prediction results in significant processing requirements. Consequently, KNN classifiers do not meet the performance requirements for many end-user applications, especially with large data sets. However, they are often used as a baseline classifier, due to the simplicity an uncompressed representation of the whole dataset.

NB classifiers are based on Bayes' theorem with the assumption of statistical independence of the features [55]. Bayes' theorem determines the probability of the l-th class to be true for a given feature set $\mathbf{x}$:

$$p(C_l j\mathbf{x}) = \frac{p(C_l)p(\mathbf{x}jC_l)}{p(\mathbf{x})} \tag{43}$$

where $p(\ )$ is the probability of an event, and $p(AjB)$ is the conditional probability of A given that B is true. A NB model is straightforward to build, with no complicated iterative parameter estimation (if any), which makes it particularly attractive for large data sets or highly dimensional data. It provides straightforward probabilistic prediction and is simple to interpret. However, the underlying simplified statistical model (usually a Gaussian distribution of the features within each class) does not hold for most real-world problems that have more complex feature-domain distributions. It comes at the cost of only being able to capture much simpler mappings between the input variables and the output class (hence, it is called "naïve"). Therefore, NB often cannot compete with more complicated ML techniques in terms of accuracy.

DT classifiers use the training data to learn simple decision rules from the features in the form of a "tree" of consecutive, one-dimensional conditions [56]. Within the tree structure, leaves represent classification results, non-leaf nodes represent conditions for single features, and branches represent conjunctions of features that lead to the classifications. At the non-leaf nodes, a selection measure is used to choose the rule that properly splits between the different classes. Since DT classifiers are defined by a series of conditional statements, their popularity is exactly due to the ability to handle complex problems by providing understandable classification rules, which are easier to interpret and implement even on simple hardware architectures. Furthermore, by choosing single features for each of the node, the classifier implicitly prioritizes the most important features, simplifying feature-engineering. However, decision trees tend to over-fit to data, especially for large numbers of nodes.

SVM classifiers create a decision hyper-planes to separate a set of observations [57]. The hyper-planes are defined by the set of closest observations for each class, called the support vectors. Since the support vectors are a subset of the training points, less computational complexity is required for training and using the model. Over-fitting is uncommon due to the generalized separator. The simplest is the linear support vector machine (L-SVM), and can be described as:

$$y_i \quad !^T\mathbf{x}_i + b \quad 1 \quad _i ; \quad _i \quad 0 \tag{44}$$

where $\mathbf{x}_i$ is the feature vector, $!$ is the weight vector, b is the bias, y_i is the i-th hyper-plane, and $_i$ is the slack variable used for handling non-separable data. The weights $!$ and bias b of the model is optimized through:

$$\min_{\{!;b\}} \frac{1}{2}k!k^2 + C \sum_{i=1}^{N} \quad _i \tag{45}$$

where N_p is the number of hyper-planes, and C is the capacity constant, also referred to as the penalty parameter of the error term. The maximum distance between data points of both classes provides some reinforcement so that future data points can be classified with more confidence. While a SVM is a linear classifier, it can applied to non-linear problems by the use of so-called "kernel trick", allowing SVMs application to non-linear problems. As an example, a radial basis function (RBF) kernel k_{RBF} is defined as:

$$k_{\mathrm{RBF}}(\mathbf{x}_i, \mathbf{x}_j) = \quad (\mathbf{x}_i) \quad (\mathbf{x}_j) = \mathrm{e}^{- \|\mathbf{x}_i - \mathbf{x}_j\|^2} \tag{46}$$

where the scaling function of the kernel. The resultant model is:

$$y_i \quad !^T \quad (\mathbf{x}_i) + b \quad 1 \quad _i ; \quad _i \quad 0 \tag{47}$$

These functions project the data into a space with higher dimensions to separate the data more effectively. SVM is useful in high dimensional spaces. Even when the input data are non-separable it generates accurate classification results due to its robustness. SVM classifiers are also commonly used for spoofing estimation in GNSS [28–31].

The f1-score is a balanced quantitative measure of quality and determines the performance of classification methods [58]. It distinguishes the correct classification within different classes and represents a harmonic mean of precision and recall. Therefore, this score takes both false positives (precision) and false negatives (recall) into account. It is defined as:

$$f_1 = \frac{2}{\mathrm{precision}^{-1} + \mathrm{recall}^{-1}} = \frac{P(H_0|H_0)}{P(H_0|H_0) + \frac{1}{2}\left(P(H_0|H_1) + P(H_1|H_0)\right)} \tag{48}$$

where $P(\,|\,)$ is the conditional probability, $P(H_0|H_0)$ is a true positive for spoofing detection, $P(H_0|H_1)$ is a false positive, and $P(H_1|H_0)$ is a false negative.

EXPERIMENTAL SETUP

The proposed ML pipeline is applied on two different data sets[3]. The first data-set contains data obtained from the simulation environment described in earlier sections. The simulated data is primarily used for training and consists of multiple scenarios and antenna arrays to generalize the models. The simulated data set consists of 200 000 data points, of which 70 % are simulated GNSS-only scenarios, and 30 % are spoofed scenarios. The simulation environment only generates Galileo E1B/C signals.

The second data set consists of measurement data recorded in four scenarios: two real GNSS signals from open-sky recordings using antenna arrays, and two laboratory spoofing attacks using a Spirent GSS9000 radio-frequency constellation simulator (RFCS). For the open-sky recordings four- and six-element UCAs were used. The data for all four scenarios are recorded with a six-channel receiver, which consisted of two synchronized three-channel Flexiband RFFEs [59]. In the case where only four channels where used, the extra two channels are simply discarded. The RFFE used either a sample-rate of 10.125 MHz or 40.5 MHz, at an analog-to-digital converter (ADC) quantization word-length of 8 bit and 4 bit complex, respectively. One spoofing attack uses a radio-frequency (RF) splitter, which provides every RFFE channel the same signal. Another spoofing attack is a "tin can" attack [10, 11], where a tin can is placed over an antenna array. Inside the tin can is a radiating antenna, which transmits the spoofing signal. Each recording is one minute long and evaluated in post-processing. From the recorded data, snapshots are extracted. Snapshot sizes between 2 ms to 20 ms are extracted, and either GPS L1 C/A signals or Galileo E1B signals are used. Table 1 shows a summary of the recorded data set, the applied processing, and references to other evaluations.

Table 1: Summary of the recorded data-set

Scenario No	Type	Input	Number of elements N_{el}	Sample-rate [MHz]	Word-length [bits]	GNSS Signals	Number of Snapshots	Published results
# 1	GNSS	6-UCA	6	10.125	8	L1CA	59 812	[26, 27, 32]
# 2	GNSS	4-UCA	4	40.5	4	E1B+L1CA	19 994	[11]
# 3	Spoof	Splitter	6	10.125	8	L1CA	60 000	[26, 27, 32]
# 4	Spoof	4-UCA+ Tin-Can	4	40.5	4	E1B+L1CA	20 000	[11]

For the ML approaches three groups of studies are performed:

1. **Scenario #1**: Training and validation on the same data set (i.e., either only the simulated or recorded data is used).

[3]The data-sets are available at `https://www.iis.fraunhofer.de/positioning`

2. **Scenario #2**: Training on one data set and then validation on another (e.g., train on the simulated data and then validate on the recorded data).

3. **Scenario #3**: Training on one data set and then validation on another, but restricting the simulated data set to only use four- and six-element UCA arrays (these are the only arrays that are also present in the real data set).

Through cross validation between the various data sets, the robustness and validity of the ML models can be assessed and the simulation environment can be verified. Table 2 shows the distribution of the data sets for the different scenarios.

Table 2: Combinations of data sets

Scenario No	**Scenario Name**	**Training data set**	**Testing data set**	**Number of training samples**	**Number of of testing samples**
#1	Only individual data sets	simulated	simulated	140 000	60 00
		recorded	recorded	112 000	48 000
#2	Using two different data sets	simulated	recorded	140 000	112 000
		recorded	simulated	112 000	200 000
#3	Antenna-specific data sets	simulated	recorded	117 000	200 000
		recorded	simulated	112 000	117 000
		simulated (4,6)	simulated (3,5,7)	117 000	32 000

Optimal hyper-parameters were obtained using a cross-validated grid search method, optimizing the models in terms of classification performance on the validation data. The f1-score assess the classification performance, because it is a balanced quantitative measure of quality.

In the grid search, first,the stratified k-fold and the number of folds are chosen for the cross-validation scheme [49]. A set of the suitable hyper-parameters and aligned parameter search spaces, listed in Table 3 is chosen for each of the models: For the KNN model only the optimal number of neighbors k are determined. The DT grid search requires the split criterion and the maximum depth of the tree ("maxdepth"). Lastly, the SVM model is used with linear and RBF kernels. For each kernel the penalty parameter of the error term C, and the radius of influence of samples selected as support vectors are determined.

For each cross-fold, all possible parameter combinations are tested on the corresponding cross-fold validation set to obtain the parameters producing the highest classification score (i.e., f1-score). The best parameter combinations are then used to train on the whole training set and generate the final model used in evaluation.

Table 3: Grid search setup

Model	**Parameter**	**Selection grid**
KNN	k neighbors	$2\ f2;3;\ \ ;10g$
DT	criterion	$2\ f$gini; entropy$g$
	maxdepth	$2\ f1;\ \ ;25g$
SVM	kernel	$2\ f$linear; RBF$;g$
	C	$2\ f10^{-4};10^{-4};\ \ ;10^{4}g$
		$2\ f10^{-4};10^{-4};\ \ ;10^{4}g$

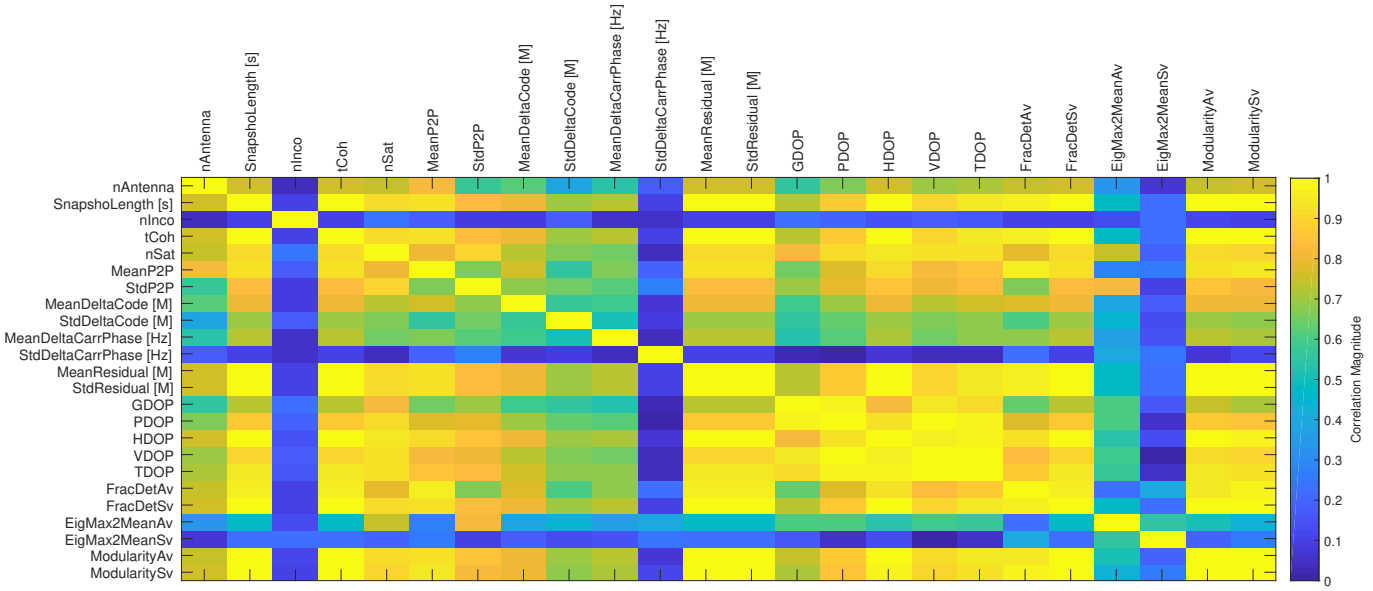
For evaluation, the test data were pre-processed in the same manner as the training and validation data. The trained models were then used to perform classification. Finally, the classification results were compared with the true labels and evaluated with the f1-score.

RESULTS

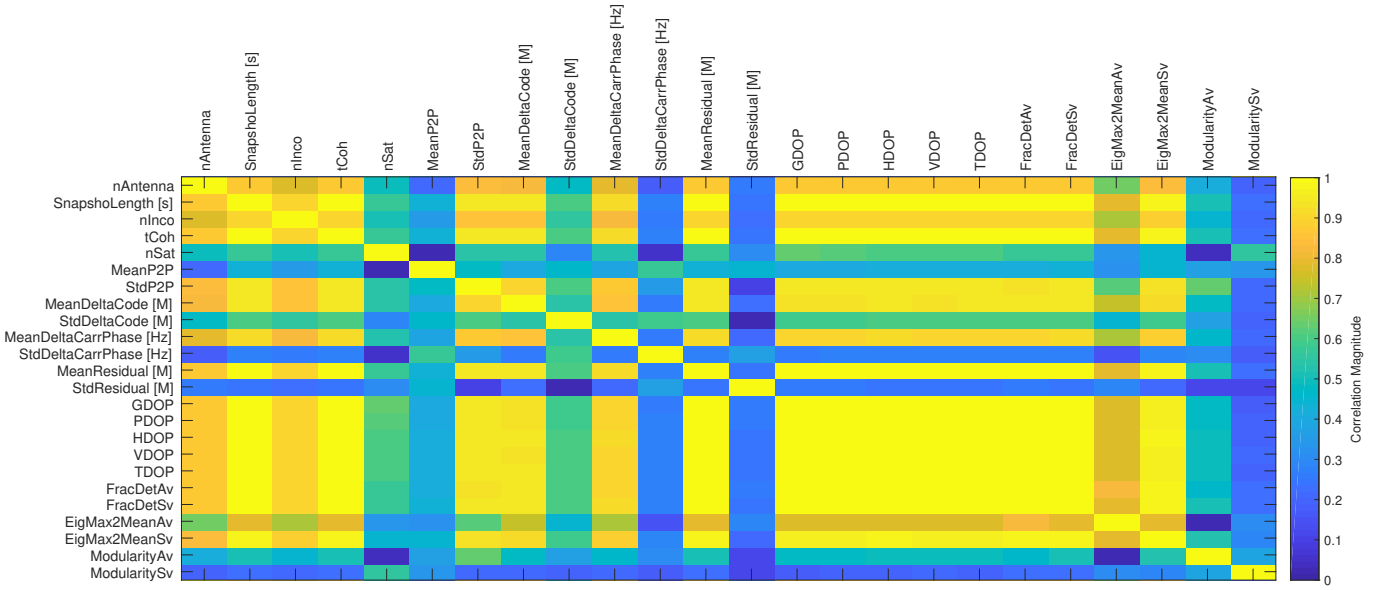
This section presents the results from training and validating the ML models.

Feature Analysis

Figure 9 shows the correlation matrix of the two data sets for all features.



(a) Recorded data set



(b) Simulation data set

Figure 9: Feature correlation matrices for the two data sets

Optimization Results

The optimal values as determined by the grid-search is shown in Table 4. There is a strong coherency for the optimal parameters between the simulated data for all antenna arrays and only the four and six element antenna arrays. However, the recordings have significantly different optimal values.

Performance Results

Table 5 shows the f1-score for the different methods on the independent test set.

High classification scores are achieved for scenario #1, where training and testing are both from the same data sets. In the case of real recordings, perfect detection is achieved by most of the classifiers and the error is slightly higher for NB. This indicates that the classes are non-overlapping in feature space. In the case of simulation data, the almost perfect score means that the data are similarly separable for all classifiers. However, the slight degradation of performance is an indication that the data is less sparse. These results prove the feasibility of the proposed data sets and features for the implied ML task. Significant and exploitable difference exists in both data sets; hence, the features are discriminative.

An important question is whether the trained classifiers can be used to generalize over data sets. Specifically, the use of recorded

Table 4: Optimal parameters from the grid search for training data.

Model	Parameter	Simulated(all)	Simulated data(4,6)	Real Recordings
KNN	k	5	3	2
DT	criterion	entropy	entropy	gini
	maxdepth	22	18	2
Linear	C	1	1	10^{-4}
RBF	C	100	10	10^{-4}
SVM		0.1	scale	0.1
LR	C	100	10	10^{-4}

Table 5: f1 scores for the ML models

Scenario	Data set		f1 score [%]					
No.	Train	Eval	LR	KNN	NB	DT	L-SVM	RBF-SVM
#1	recorded	recorded	**100.0**	**100.0**	**99.93**	**100.0**	**100.0**	**100.0**
#1	simulated	simulated	**100.0**	**99.99**	96.37	**99.94**	**100.0**	**99.99**
#2	simulated	recorded	**99.66**	**99.66**	**99.45**	*89.58*	**98.71**	**98.39**
#2	recorded	simulated	*89.82*	*83.44*	*0.0*	*73.02*	*86.56*	*84.73*
#3	simulated(4,6)	recorded	**96.28**	**99.10**	*0.0*	92.11	95.96	92.42
#3	recorded	simulated(4,6)	*89.78*	*83.37*	*0.0*	*73.02*	*86.65*	*84.58*
#3	simulated(4,6)	simulated(3,5,7)	**99.86**	**99.87**	*0.0*	**99.58**	**99.72**	**99.93**

*Values in **blue** are above 99 % and values in *red* are below 90 %.

data for the evaluation of classifiers trained on simulated data is of practical importance. Training on the simulated data and evaluation on the recorded data for scenario #2 yielded high results for most of the classifiers. The highest overall success rate of 99.66 % was achieved by LR and KNN, followed by NB. Over-fitting is common with decision trees simply due to the nature of their training. Lower depth makes the model faster but not as accurate, whereas higher depths gives accuracy on validation data but risks over-fitting. These results verify that the proposed simulation environment can be used to generate data for real-world applications, as the classifiers (apart from DT) are able to generalize between the data-sets achieving an almost perfect f1-score. Therefore, the results indicate that the simulated data can enable reliable spoofing classification. In contrast, training on the recorded data in scenario #2, yielded significantly poorer results, implying that the real recordings do not generalize well to the simulated data set. The recorded data consists of limited antenna configurations and satellite constellations. Therefore, it confirms the original hypothesis that the recorded data is too sparse for ML and cannot be used to accurately train ML models. Also notable is the decrease of the f1-score to 0.0 for NB. Analysis of the confusion matrices showed that for this case, the whole data set was predicted as “not spoofed”. This could be attributed to a highly multi-modal distribution of the classes within the feature-domain that cannot be modelled properly by Gaussian distributions. Consequently, the estimated distributions overlap heavily and the classifier decides for the class with the higher prior.

In scenario #3, when using simulated data selected corresponding to the antenna setups in the real recordings, the performance slightly worse than with using the whole data set (expect for DT). This indicates that, while the overall modelling in the simulation environment is suitable, the specific antenna influence might be different under realistic conditions. This is backed up by the fact that there is no performance increase for training with real recordings and testing with simulated data for the corresponding antenna configurations. This results is unexpected and needs to be studied further and clarified in the future.

In terms of classifier comparison, both LR and L-SVM generated comparatively high classification results for all evaluations. They are in line with KNN, but the linear models are much more efficient in terms of computational requirements for prediction as discussed earlier. DT, as discussed earlier over-fits to specific data-sets and shows worse generalization results. While NB produced high f1-scores for training with the whole simulated and testing with the whole real data set, it is heavily biased toward “not spoofed” for training with real data and the antenna-specific evaluations.

CONCLUSION

This paper presents an investigation on different ML approaches to detect spoofing signals for a multi-antenna snapshot receiver. Specifically, to address short-comings with currently available recorded data being too sparse for reliable ML classification, the use of simulated data for training before applying the ML models in real-world application is investigated. First, a simulation environment for multi-antenna snapshot receivers is created and described in detail. Second, the simulation environment is used for training of the ML algorithms. Third, the real recorded data is used to validate the ML approaches and the simulation environment. Additional cross-validation and training is also done, to ensure reliable results.

Training and validating on the same data sets showed the best results, as expected. Training on the simulated data and then validating on the recorded data had a small decrease in performance. However, several algorithms still achieved a better than 99 % f1-score. It indicates that the simulation environment can be used to accurately classify the data. Training on the recorded data and validating on the simulation data had significantly poorer results, with all f1-scores below 90 % This highlights risks that the recorded data is sparse, and prone to over-fitting. Either more data can be recorded—which will require significant data gathering activities and associated time investment—to overcome this problem, or the simulation environment could be used to circumvent the effort. Preliminary results show that the good performance was achieved with LR, KNN and L-SVM. Both NB and DT indicated some issues with training, and often had significantly lower performance than other algorithms. Finally, the RBF kernel SVM was not completed by the time of writing of the paper and no accurate conclusions of this method can be made.

Since the simulation platform is validated through the process, more advanced studies are proposed for future research. For example, the evaluation of a mixed real and spoofed scenarios, or to differentiate between spatially coherent signals caused by multi-path effects and spatially coherent spoofed signals. Optimization of ML methods are also proposed for future research.

REFERENCES

[1] J. A. Volpe, "Vulnerability assessment of the transport infrastructure relying on the global positioning system," *U.S. DoT*, 2001.

[2] A. Jafarnia-Jahromi, A. Broumandan, J. Nielsen, and G. Lachapelle, "GPS vulnerability to spoofing threats and a review of antispoofing techniques," *International Journal of Navigation and Observation*, vol. 2012, pp. 1–16, 2012.

[3] C4ADS, *Above us only stars: Exposing GPS spoofing in Russia and Syria*, 2019.

[4] Ajinkya, "How to play Pokemon Go without moving on Android," 2019. https://devsjournal.com/how-to-play-pokemon-go-without-moving-no-root-required.html.

[5] Spy Blog, "Road pricing GPS signal jammers? What about cheap GPS position spoofing devices?," 2007. https://spyblog.org.uk/ssl/spyblog/2007/08/09/road-pricing-gps-signal-jammers-what-about-cheap-gps-position-spoofing-devices.html.

[6] N. Ungerleider, "Spoofed satellite feeds trouble Google's global fishing watch," 2014. https://www.fastcompany.com/3038863/spoofed-satellite-feeds-trouble-googles-global-fishing-watch.

[7] R. Mit, "GPS spoofing mystery affirms need for protection," 2019. https://www.wardsauto.com/industry-voices/gps-spoofing-mystery-affirms-need-protection.

[8] Associated Press, "Porch pirates beware: Amazon and new jersey police are planting fake packages with gps chips to catch thieves," 2018. https://www.businessinsider.com/amazon-police-team-up-fake-packages-gps-catch-porch-pirates-2018-12.

[9] T. Yasmin Mina, S. Bhamidipati, and G. Xingxin Gao, "Gps spoofing detection for the power grid network using a multireceiver hierarchical framework architecture," *NAVIGATION*, vol. 66, no. 4, pp. 857–875, 2019.

[10] M. L. Psiaki and T. E. Humphreys, "GNSS spoofing and detection," *Proceedings of the IEEE*, vol. 104, pp. 1258–1270, June 2016.

[11] J. R. Van der Merwe, A. Rügamer, A. Popugaev, X. Zubizarreta, and W. Felber, "Cooperative spoofing attack detection using multiple antennas and a snapshot receiver," in *Proceedings of the 32nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2019)*, September 2019.

[12] V. Lucas-Sabola, G. Seco-Granados, J. A. López-Salcedo, J. A. Garcia-Molina, and M. Crisci, "Cloud GNSS receivers: New advanced applications made possible," in *2016 International Conference on Localization and GNSS (ICL-GNSS)*, pp. 1–6, June 2016.

[13] I. Fernández-Hernández and K. Borre, "Snapshot positioning without initial information," *GPS Solutions*, vol. 20, pp. 605–616, Oct 2016.

[14] S. V. Shafran, E. A. Gizatulova, and I. A. Kudryavtsev, "Snapshot technology in GNSS receivers," in *2018 25th Saint Petersburg International Conference on Integrated Navigation Systems (ICINS)*, pp. 1–3, May 2018.

[15] T. Pany, *Navigation Signal Processing for GNSS Software Receivers*. GNSS technology and applications series, Artech House, 2010.

[16] T. N. Dinh and V. La The, "A novel design of low power consumption GPS positioning solution based on snapshot technique," in *2017 International Conference on Advanced Technologies for Communications (ATC)*, pp. 285–290, Oct 2017.

[17] B. Wales, L. Tarazona, and M. Bavaro, "Snapshot positioning for low-power miniaturised spaceborne GNSS receivers," in *2010 5th ESA Workshop on Satellite Navigation Technologies and European Workshop on GNSS Signals and Signal Processing (NAVITEC)*, pp. 1–6, Dec 2010.

[18] D. Rosenfeld and E. Duchovny, "Off-board positioning using an efficient GNSS SNAP processing algorithm," in *Proceedings of the 23rd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2010)*, pp. 1088–1093, 2010.

[19] A. Rügamer, D. Rubino, X. Zubizarreta, W. Felber, J. Wendel, and D. Pfaffelhuber, "Spoofing resistant UAVs," in *Proceedings of the 31st International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2018)*, 2018.

[20] D. Rubino, A. Rügamer, I. Lukčin, S. Taschke, M. Stahl, and W. Felber, "Galileo PRS snapshot receiver with serverside positioning and time verification," in *Proceedings of DGON POSNAV 2016*, 2016.

[21] A. Rügamer, D. Rubino, I. Lukčin, S. Taschke, M. Stahl, and W. Felber, "Secure Position and Time Information by Server Side PRS Snapshot Processing," in *Proceedings of the 29th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2016)*, pp. 3002–3017, September 2016.

[22] C. Günther, "A survey of spoofing and counter-measures," *Navigation: Journal of the Institute of Navigation*, vol. 61, no. 3, pp. 159–177, 2014.

[23] A. Rügamer and D. Kowalewski, "Jamming and Spoofing of GNSS Signals - An Underestimated Risk?!," in *Proceedings, FIG Working Week 2015, May 17 - 21, 2015, Sofia, Bulgaria*, 2015.

[24] P. Y Montgomery, T. E Humphreys, and B. M. Ledvina, "Receiver-autonomous spoofing detection: Experimental results of a multi-antenna receiver defense against a portable civil gps spoofer," in *Proceedings of the 2009 International Technical Meeting of The Institute of Navigation*, vol. 1, pp. 124–130, 01 2009.

[25] A. Konovaltsev, M. Cuntz, C. Haettich, and M. Meurer, "Autonomous Spoofing Detection and Mitigation in a GNSS Receiver with an Adaptive Antenna Array," in *Proceedings of the 26th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2013)*, pp. 2937 – 2948, 2013.

[26] J. R. van der Merwe, A. Rügamer, A. Fernández-Dans Goicoechea, and W. Felber, "Blind spoofing detection using a multi-antenna snapshot receiver," in *2019 International Conference on Localization and GNSS (ICL-GNSS)*, June 2019.

[27] J. R. Van der Merwe, A. Rügamer, and W. Felber, "Blind spoofing GNSS constellation detection using a multi-antenna snapshot," *Sensors*, vol. 19, p. 5439, 12 2019.

[28] G. Panice, S. Luongo, G. Gigante, D. Pascarella, C. Di Benedetto, A. Vozella, and A. Pescapè, "A svm-based detection approach for gps spoofing attacks to uav," in *2017 23rd International Conference on Automation and Computing (ICAC)*, pp. 1–11, 2017.

[29] S. Semanjski, A. Muls, I. Semanjski, and W. De Wilde, "Use and validation of supervised machine learning approach for detection of GNSS signal spoofing," in *2019 International Conference on Localization and GNSS (ICL-GNSS)*, pp. 1 – 6, June 2019.

[30] M. R. Manesh, J. Kenney, W. C. Hu, V. K. Devabhaktuni, and N. Kaabouch, "Detection of gps spoofing attacks on unmanned aerial systems," in *2019 16th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pp. 1–6, 2019.

[31] S. Semanjski, I. Semanjski, W. De Wilde, and A. Muls, "Use of supervised machine learning for gnss signal spoofing detection with validation on real-world meaconing and spoofing data—part i," *Sensors*, vol. 20, p. 1171, Feb 2020.

[32] J. R. van der Merwe, A. Fernández-Dans Goicoechea, A. Rügamer, X. Zubizarreta, D. Rubino, and W. Felber, "Multi-antenna snapshot receiver," in *2019 European Navigation Conference (ENC)*, April 2019.

[33] B. C. Geiger, C. Vogel, and M. Soudan, "Comparison between ratio detection and threshold comparison for GNSS acquisition," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 48, pp. 1772–1779, April 2012.

[34] F. van Diggelen, *A-GPS: Assisted GPS, GNSS, and SBAS*. Artech House, 2009.

[35] E. D. Kaplan and C. J. Hegarty, *Understanding GPS: Principles and Applications*. Artech House mobile communications series, Artech House, 2 ed., 2006.

[36] M. Appel, A. Konovaltsev, and M. Meurer, "Joint antenna array attitude tracking and spoofing detection based on phase difference measurements," in *Proceedings of the 29th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2016) September 12 - 16, 2016*, pp. 3018 – 3026, 2016.

[37] E. Pérez Marcos, A. Konovaltsev, S. Caizzone, M. Cuntz, K. Yinusa, W. Elmarissi, and M. Meurer, "Interference and spoofing detection for GNSS maritime applications using direction of arrival and conformal antenna array," in *Proceedings of the 31st International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2018)*, pp. 2907 – 2922, 2018.

[38] A. Broumandan, A. Jafarnia-Jahromi, S. Daneshmand, and G. Lachapelle, "Overview of spatial processing approaches for GNSS structural interference detection and mitigation," *Proceedings of the IEEE*, vol. 104, pp. 1246–1257, June 2016.

[39] A. Kortun, M. Sellathurai, T. Ratnarajah, and C. Zhong, "Distribution of the ratio of the largest eigenvalue to the trace of complex Wishart matrices," *IEEE Transactions on Signal Processing*, vol. 60, pp. 5527–5532, Oct 2012.

[40] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, p. P10008, oct 2008.

[41] J. M. Pujol, V. Erramilli, and P. Rodriguez, "Divide and conquer: Partitioning online social networks," *CoRR*, vol. abs/0905.4918, 2009.

[42] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*, vol. 1 of *Prentice-Hall Signal processing series*. Prentice Hall, 1993.

[43] European Global Navigation Satellite Systems Agency (GSA), "Galileo almanac," 2020.

[44] E. Tuncer and B. Friedlander, *Classical and Modern Direction-of-Arrival Estimation*. Burlington: Elsevier, 2009.

[45] GPS World, *2018 GNSS Antenna Survey*, 2018.

[46] T. S. Rappaport, *Wireless Communications: Principles and Practice*. Electrical engineering, Prentice Hall, 1996.

[47] F. Neri, *Introduction to Electronic Defense Systems*. SciTech, 2006.

[48] Y. Bengio, "Gradient-based optimization of hyperparameters," *Neural Computation*, vol. 12, no. 8, pp. 1889–1900, 2000.

[49] J. D. Rodriguez, A. Perez, and J. A. Lozano, "Sensitivity analysis of k-fold cross validation in prediction error estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 3, pp. 569–575, 2010.

[50] E. Alpaydin, *Introduction to Machine Learning*. Adaptive Computation and Machine Learning series, MIT Press, 2014.

[51] S. H. Walker and D. B. Duncan, "Estimation of the probability of an event as a function of several independent variables," *Biometrika*, vol. 54, no. 1/2, pp. 167–179, 1967.

[52] R. Langari, Liang Wang, and J. Yen, "Radial basis function networks, regression weights, and the expectation-maximization algorithm," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 27, no. 5, pp. 613–623, 1997.

[53] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[54] Wen-Jyi Hwang and Kuo-Wei Wen, "Fast kNN classification algorithm based on partial distance search," *Electronics Letters*, vol. 34, no. 21, pp. 2062–2063, 1998.

[55] P. Domingos and M. Pazzani, "On the optimality of the simple Bayesian classifier under zero-one loss," *Machine Learning*, vol. 29, no. 2, pp. 1573 – 0565, 1997.

[56] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 1573 – 0565, 1986.

[57] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 1573 – 0565, 1995.

[58] H. Huang, H. Xu, X. Wang, and W. Silamu, "Maximum f1-score discriminative training criterion for automatic mispronunciation detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 4, pp. 787–797, 2015.

[59] A. Rügamer, F. Förster, M. Stahl, and G. Rohmer, "A flexible and portable multiband GNSS front-end system," in *Proceedings of the 25th International Technical Meeting of the Satellite Division of the Institute of Navigation, ION GNSS 2012, September 17-21, 2012, Nashville, Tennessee, USA*, September 2012.