remote sensing

MDPI

Article

CONSTDET: Control Semantics-Based Detection for GPS Spoofing Attacks on UAVs

Xiaomin Wei [1,2,3,*], Cong Sun [1,2,3], Minjie Lyu [1], Qipeng Song [1] and Yue Li [1]

[1] School of Cyber Engineering, Xidian University, Xi'an 710071, China
[2] ISN State Key Laboratory, Xidian University, Xi'an 710071, China
[3] Shaanxi Key Laboratory of Network and System Security, Xi'an 710071, China
* Correspondence: xmwei@xidian.edu.cn

Abstract: UAVs are widely used in agriculture, the military, and industry. However, it is easy to perform GPS spoofing attacks on UAVs, which can lead to catastrophic consequences. In this paper, we propose CONSTDET, a control semantics-based detection approach for GPS spoofing attacks of UAVs using machine learning algorithms. Various real experiments are conducted to collect real flight data, on the basis of which CONSTDET is designed as a practical detection framework. To train models for the detection of GPS spoofing attacks, specified flight data types are selected as features based on the control semantics, including the altitude control process and the horizontal position control process, since these data are able to represent the dynamic flight and control processes. Multiple machine learning algorithms are used to train and generate the best classifier for GPS spoofing attacks. CONSTDET is further implemented and deployed on a real UAV to support onboard detection. Experiments and evaluations validate that CONSTDET can effectively detect GPS spoofing attacks and the detection rate can reach 97.70%. The experimental comparison demonstrates that CONSTDET has better performance than existing detection approaches.

Keywords: UAV; GPS spoofing attacks; GPS spoofing detection; control semantics; machine learning

Citation: Wei, X.; Sun, C.; Lyu, M.; Song, Q.; Li, Y. CONSTDET: Control Semantics-Based Detection for GPS Spoofing Attacks on UAVs. *Remote Sens.* **2022**, *14*, 5587. https://doi.org/10.3390/rs14215587

Academic Editor: Gwanggil Jeon

Received: 28 September 2022
Accepted: 2 November 2022
Published: 5 November 2022

1. Introduction

UAVs are widely used in photography, light shows, firefighting, agriculture, the military, and industry. They are playing increasingly important roles in daily life. The accurate control of UAVs is of great importance to reach the designed 3D position so that UAVs can perform the specified task such as aerial photography, crop dusting, fire extinguishing, etc. However, it is easy to perform GPS spoofing attacks on UAVs through software-based global navigation satellite system (GNSS) signal generators [1–3]. GPS spoofing is a well-known threat that can lead to catastrophic consequences such as security problems (e.g., hijacking by attackers) and safety issues (e.g., crashing). It can be implemented with a low-cost apparatus [4–6]. The effects of GPS spoofing have been analyzed and demonstrated on UAVs [4,7–10].

A GPS spoofing attack is when attackers use fake GPS signals to replace the actual GPS signals and guide the UAV to a fake destination. UAVs can also be captured and controlled through GPS spoofing [9,10]. There are a variety of GPS spoofing techniques targeting UAVs [11,12]. An attacker is able to manipulate the true state of a UAV with a GPS spoofing attack, which makes it possible to guide the UAV far from its planned flight path without raising alarms. A scenario of a GPS spoofing attack on a UAV is shown in Figure 1. We can compare the UAV's actual path and the fake flight path. For example, suppose that a UAV is flying to the desired destination following the planned flight path. At some point, the UAV is spoofed with GPS signals from the attacker. The result is that the flight control system generates a current fake position for the UAV location, deviating from the current actual position. It further computes a fake position for the subsequent flight and guides the

UAV to the desired destination. Since the UAV has deviated from its planned flight path due to the GPS spoofing attack, the UAV is flying to the fake destination along the actual flight path as shown in this figure. However, from the UAV ground control station (GCS), the operators can only observe that the UAV is flying to the desired destination along the fake flight path.

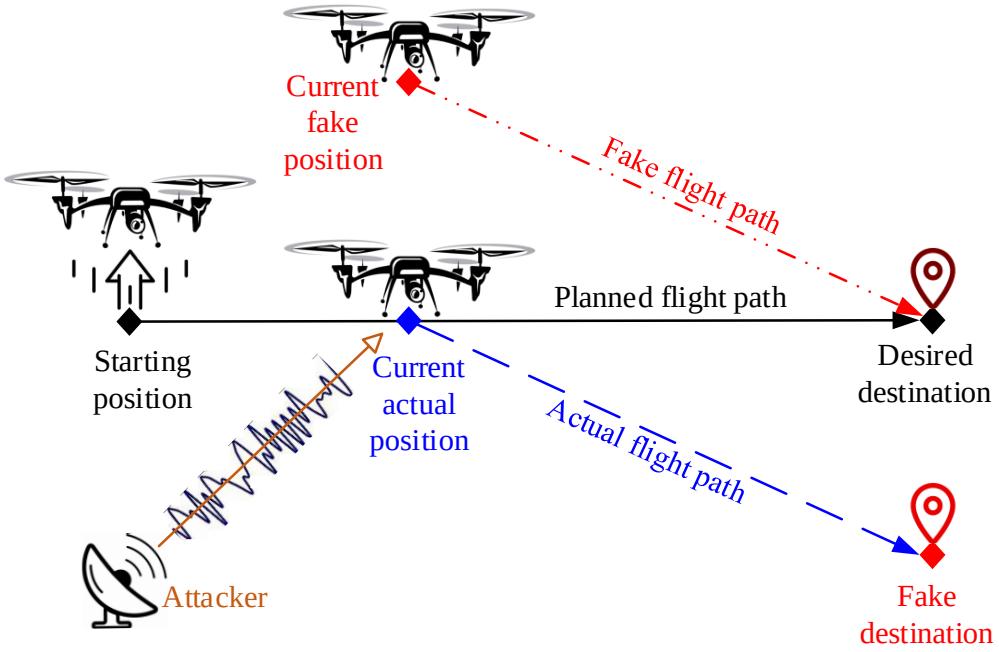

Figure 1. Scenario of GPS spoofing attack on a UAV. A UAV takes off from the starting position and then flies to the desired destination along the planned flight path. At some point, an attacker begins to launch a GPS spoofing attack on this UAV, changing the UAV's location from the current position to the fake position. With sustained attacks, the ground control station shows that this UAV is flying along the fake flight path to reach the desired destination. However, this UAV is flying along the actual flight path to reach the fake destination.

There is no quick, easy, and cheap way to fix the inherent problem of GPS spoofing, especially for an insecure civil GPS signal [4], which is unencrypted, designed with an open standard, and freely accessible to all. This advantage has made civil GPS popular, but at the same time, it means that so many devices, including UAVs and cars, are faced with the challenge of being attacked by GPS spoofing.

To ensure the security and safety of UAVs, there are a variety of studies working on the detection of GPS spoofing attacks. Most detection approaches are based on satellite images, sensor data (including acceleration, angular velocity, video, optical flow, magnetic induction, etc.), GPS signals, and GPS-based positions. Currently, different kinds of machine learning (ML) algorithms are used by many approaches for model training [13–19], so that they can predict the positions for detection or directly determine if a UAV is under a spoofing attack. This is an effective way to detect spoofing attacks. Some methods [20–24] compute the position with sensor data according to the physical significances among the different kinds of sensor data, for example, speed can be obtained from acceleration by integration, distance can be obtained by further integration, and the current position can be obtained based on the distance and yaw. In addition, some methods can detect spoofing attacks based on the cooperation of multiple UAVs [22,25,26]. To determine if there is a GPS spoofing attack, the difference between the predicted position and the GPS-based position is compared with a given threshold value, which is set as a constant value in most detection approaches. Moreover, simulations are used by most existing works to evaluate the performance of their methods [15–19,22–28].

On the whole, ML-based detection approaches have higher detection rates than other approaches. However, there are still some weaknesses in current ML-based detection research into GPS spoofing attacks on UAVs and they are as follows:

- The control semantics are not considered in these detection approaches. However, control theory is the basis of stable flight for UAVs [29,30]. The control semantics include the mechanism and theory about how to control UAVs using flight data. This mechanism is constant for UAVs. The flight data can represent not only this mechanism but also the dynamic flight state. Each kind of UAV data is not independent; therefore, it is necessary to integrate the control semantics into the selection of the training data.
- A majority of existing ML-based detection approaches [15–18] are trained and evaluated using data collected by simulations. The classifiers are built based on the datasets. Simulated data are quite different from real data. For model training, a real or simulated dataset can not only affect the detection accuracy but also reduce the credibility of the method.
- Some detection approaches [14,17,18] only take part of the flight data into consideration. The considered data cannot comprehensively represent the relationships between the position-related flight data. The actual position data follow multiple changing mechanisms since they can be affected by changes in the external environment, different flight modes, changes in the UAV attitude, and so on.

This paper proposes a **con**trol **s**eman**t**ics-based **det**ection approach called **ConstDet**, which is based on the **constant** control semantics to intelligently **detect** attacks using ML algorithms. The UAV flight is controlled by commands from flight control systems, which are implemented according to the control semantics designed for the specified UAV. ConstDet is proposed based on both the horizontal position control semantics and altitude control semantics, making it reasonable to select certain flight data for GPS spoofing detection. ConstDet is designed according to the original data, which are collected through real flights. ConstDet is an intelligent detection approach for GPS spoofing attacks on UAVs using ML algorithms. Given the required flight data about the horizontal position control and altitude control, it directly generates results that show whether the UAV is under a spoofing attack.

In summary, this paper makes the following contributions:

- **Feature selection based on control semantics.** We select key data from UAV control systems by analyzing the control semantics, which contain the real control mechanisms, that is, the horizontal position control process and the altitude control process. Since such flight data can represent both the control process of UAVs and the dynamic flight, these flight data types are selected as features. The horizontal position control system is in charge of the position control, horizontal speed control, and attitude control so we select the actual horizontal position, speed, angle, and angular rate as features. The altitude control system is in charge of the altitude control, vertical speed control, and vertical acceleration control so the actual altitude, vertical speed, and vertical acceleration are selected as features for model training.
- **Intelligent detection framework.** We design an intelligent detection framework for the detection of UAV GPS spoofing attacks. Different kinds of ML models are applied for model training using real flight data. The models intelligently learn the data relationship between the control data obtained from the altitude controller and the horizontal position controller. The trained model (classifier) can be deployed in a UAV to implement the detector for GPS spoofing attacks. The detector analyzes the onboard flight data based on the learned data relationship and then provides an alarm if there is a spoofing attack.
- **Real flight dataset.** We perform a variety of experiments to collect flight data. For intelligent detection methods, the dataset used for model training is one of the key factors for the selection of the ML algorithms and model construction since different datasets can produce different detection models and come to different conclusions.

Thus, it takes us a lot of time and effort to collect real data so that the framework and evaluation can be designed based on real data.

- **Deployment in a real UAV.** To demonstrate the effectiveness of ConstDet, the detector for GPS spoofing attacks is deployed in a real UAV. We implement the online data acquisition on a companion computer so that ConstDet can obtain flight data and detect GPS spoofing attacks in real time.

The remainder of the paper is organized as follows. Section 2 introduces some related works about detection approaches for GPS spoofing attacks on UAVs. Section 3 provides the feature analysis that explains how to select data features for model training based on control semantics and presents the control semantics-based intelligent detection framework. Section 4 describes the experimental setup, feature selection, model selection, and model evaluation. Section 5 reproduces the existing detection method and compares our approach with existing works. Finally, Section 6 concludes this paper.

2. Related Work

According to whether ML algorithms are applied or not, GPS spoofing detection solutions for UAVs are divided into two categories: ML-based detection [13–19] and non-ML-based detection [20–28]. Detailed explanations of these approaches are provided below.

2.1. ML-Based Detection

Most ML-based detection methods [15–19] are evaluated with simulated data.

Xue et al. [13] proposed *DeepSIM*, a satellite imagery matching approach, to detect GPS spoofing attacks of UAVs based on different deep neural network models. DeepSIM achieved detection through a comparison of historical satellite images (collected from Google Earth) at the GPS-based position and real-time aerial images (taken by cameras). Their best model had a detection rate of 94.8% (on-ground detection model) and 89% (onboard detection model) for the detection of GPS spoofing attacks.

Feng et al. [14] proposed an XGBoost-based detection method (called the JSA method) for GPS spoofing attacks on UAVs. The model training was performed with angular velocity, acceleration, and GPS data. To improve the accuracy of the detection, the genetic algorithm was used to tune the parameters of the XGBoost model. To reduce the computation load on UAVs, the model was first generated by off-board training and then deployed to the UAV board for onboard training and predictability. The experiment illustrated that their method was capable of detecting all GPS spoofing attacks according to their dataset. We implement the JSA method for a comparison with our method in Section 5.

Kim et al. [15] proposed a deep learning-based framework to detect sensor spoofing attacks. To learn the nonlinear dynamics, the multi-layer perceptron (MLP) model was trained with the sensor data collected by the software-in-the-loop (SITL) simulation. These data were obtained from the gyroscope, accelerometer, and GPS. Moreover, to make the model training better, generative adversarial networks (GANs) were leveraged to generate a segment of sensor traces for data augmentation. The performance of the proposed techniques was evaluated using the SITL simulation and should be further demonstrated on a real system.

Calvo-Palomino et al. [16] presented a GPS spoofing detection method based on long short-term memory (LSTM). The neural network was trained with GPS Doppler shift measurements in 5 s of granularity. Using low-cost software-defined radio (SDR) receivers embedded in UAVs, UAVs processed the received GPS signals to predict the Doppler characteristics. GPS spoofing attacks were detected with a comparison of the Doppler effect and its predictable pattern originating from the moving GNSS satellites. The effectiveness of the detection function was demonstrated using a simulated GPS spoofing device. More experiments are necessary to further evaluate the detection rate.

Wang et al. [17] also proposed a GPS spoofing attack detection method using LSTM. The model was trained with velocity, acceleration, latitude, and longitude in the x-direction and y-direction of UAVs so that the trained model could predict UAV positions based

on a given flight path. Moreover, UAVs should fly upon the path of a given shape to support the detection of GPS spoofing attacks. GPS spoofing attacks were determined by comparing the predicted position with the position calculated using the GPS signals, and a threshold value was used to measure the difference between the two positions. This method is applicable when UAVs can run smoothly. A MATLAB simulation was leveraged to explain the effectiveness of their method and the detection rate was about 78%.

Panice et al. [18] proposed a detection approach for UAV GPS spoofing attacks based on the support vector machine (SVM) approach with the data from an inertial navigation system (INS). The detection was based on the error data between the GPS-based position and the INS-based position over a period of time. The detection model was trained and evaluated with the simulated data.

Khoei et al. [19] also used machine learning algorithms to train and detect GPS spoofing attacks. Their selected features were the characteristics of the GPS signals including the signal-to-noise ratio, carrier-loop Doppler measurements, prompt quadrature prompt, prompt in-phase prompt, early correlator output, late correlator output, prompt correlator, carrier phase shift, decoded time information, receiver time, pseudo range, Doppler-shift measurements, and satellite number. They were obtained through a simulation using a software-defined radio (SDR) device. In fact, the user could not obtain these data using a widely used GPS receiver.

2.2. Non-ML-Based Detection

Jansen et al. [25] presented Crowd-GPS-Sec, a method of detecting GPS spoofing attacks and localizing GPS spoofers for moving airborne targets such as UAVs or commercial airliners. Crowd-GPS-Sec utilized crowdsourcing to monitor position data derived from GPS data, which were periodically broadcast for air traffic surveillance by UAVs or aircraft. The GPS spoofing detection and localization function were designed based on those data by analyzing the contents and the time of arrival. They evaluated Crowd-GPS-Sec with both the real-world data and the simulated data and the detection rate was about 75%.

Feng et al. [20] proposed a GPS spoofing detection method (called the DATE method) using onboard motion sensors. The angular velocity measured by the gyroscopes was used to compute the coordinate transform matrix. GPS positions were used to compute the acceleration in the geographic coordinate. The matrix and the acceleration were combined to generate the acceleration in the body-fixed coordinate, which was compared with the acceleration measured by the accelerometers to determine if there was a GPS spoofing attack according to a given threshold value. To reduce the errors accumulated by accelerometers, they provided another GPS spoofing detection method called the TECS method [21]. The gyroscope data were utilized to compute the coordinate transform matrix and finally generate the yaw angle. The GPS position was used to compute the angle, which was compared with the yaw angle to detect if GPS spoofing attacks had been performed on UAVs according to a given threshold value. Their ML-based detection approach, the JSA method, was better than the DATE method and TECS method, as seen in the evaluation in [14].

Liang et al. [22] provided a detection solution for UAV GPS spoofing attacks using the positions of multiple UAVs. The ground control station (GCS) received position information from the UAVs and then calculated and sent the positions to specified UAVs. The detection of GPS spoofing attacks was performed based on the calculated position and the original GPS position, which was demonstrated in simulation experiments, and the detection rate reached 98.6% if the number of UAVs was greater than four. If the number of UAVs was less than four, it took 28 s to accomplish detection, in which case the detection rate reached 96.7%.

Meng et al. [23] introduced a GPS spoofing detection approach for UAVs based on linear regression (LR). LR was leveraged to describe and predict the flight trajectory to the destination for UAVs. The predicted longitude and latitude were compared with the longitude and latitude provided by the GPS to determine if there was a GPS spoofing attack based on a given threshold value. The experiment was carried out in the UAV simulation

platform and a dynamic Stackelberg game between a GPS spoofer and a UAV was used to evaluate the performance. The detection rate is an important indicator for demonstrating the effectiveness of a GPS spoofing detection method but it was not evaluated in the above study.

Bada et al. [26] presented a policy-based detection approach for the detection of colluding UAV GPS spoofing attacks in flying ad hoc networks (FANETs). Unlike conventional GPS spoofing attacks, spoofing attacks in FANETs can be performed by several UAVs. The authors added a trust model based on the beta Weibull distribution to overcome the blind trust problem. Simulations were implemented to evaluate their approach and the mean detection probability reached 96.08% for UAVs in FANETs, which was higher than the existing method.

Eldosouky et al. [24] introduced a mathematical framework to protect UAVs against GPS spoofing attacks. They applied system dynamics to describe the UAV motion model to derive the optimal routes that were adopted by UAVs for reaching their destinations, making it possible to obtain the spoofer's optimal imposed locations on the UAVs and predict their traveling routes under attacks. To mitigate the effect of GPS spoofing attacks, they developed a countermeasure mechanism based on the premise of the cooperative localization of multiple nearby UAVs. To improve the defense mechanisms, the interactions between a GPS spoofer and a drone operator were modeled using a dynamic Stackelberg game. The simulation experiments showed that their defense mechanisms were better than other strategy selection techniques in terms of reducing the possibility of capturing UAVs.

Elena et al. [27] applied the Kullback–Leibler divergence to detect GPS spoofing attacks on UAVs. Firstly, the Poisson distribution was used to describe the random variation of parameters, including the altitude, number of visible satellites, GPS speed, angle, latitude, and longitude. Then, the Kullback–Leibler divergence was used to calculate the entropy value. The evaluation was performed through the simulation. However, this method could not distinguish between a higher entropy value produced by abnormal behavior such as an attack and environmental influences.

Barak et al. [28] presented a detection approach for UAV GPS spoofing attacks using frames collected from cameras' video streams and locations obtained from a GPS. They fit the similarity correlation between the frames and the distance between the frame-corresponding GPS-based positions to construct a linear regression function, which could predict the UAV position for a comparison with the GPS-based position. They evaluated this method using the simulation data and real data. However, this method was sensitive to different terrains, ambient light, and altitudes, which shows that it was impractical.

3. Method

3.1. Feature Analysis of Flight Data

Selecting appropriate data is an important step in ML. Specific relevant features can speed up ML processing and improve accuracy [31]. We selected the features for our ML-based approach, ConstDet, through an analysis of the control semantics. The control semantics are the mechanism and theory about how to control UAVs using flight data.

Stable flight is assured by UAV PID control systems [29,30], which contain lots of control data, as shown in Figure 2. A UAV PID control system is constructed by a flight altitude PID control system and a horizontal flight position PID control system. The UAV control system controls a UAV to accomplish the mission, which is composed of the desired altitude and desired horizontal position for the altitude control and horizontal position control, respectively.

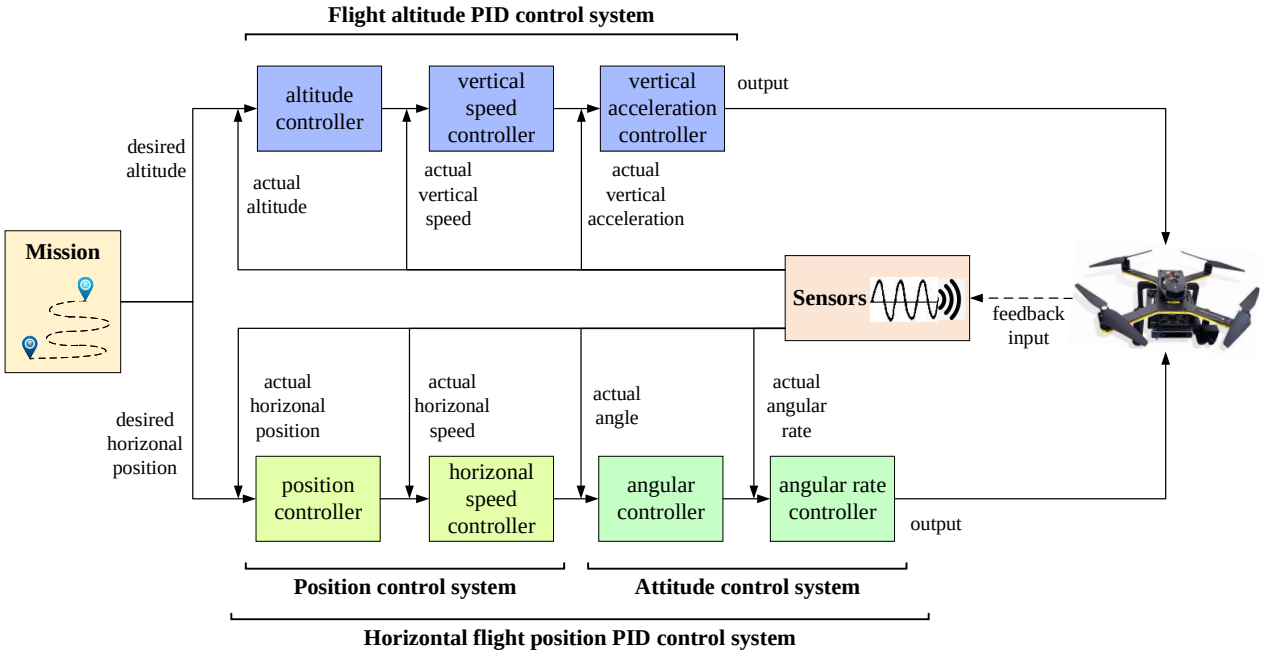

Figure 2. A UAV PID control system.

The flight altitude PID control system is a three-loop cascading controller, which contains an altitude controller, a vertical speed controller, and a vertical acceleration controller. To reach the desired altitude, it computes and generates the controlling value by processing the actual altitude, actual vertical speed, and actual vertical acceleration, which are obtained from the control system.

The horizontal flight position PID control system is a four-loop cascading controller, which is composed of a position control system (including a position controller and a horizontal speed controller) and an attitude control system (including an angular controller and an angular rate controller). To reach the desired horizontal position, the horizontal position controller computes and generates the controlling value by processing the actual horizontal position, actual horizontal speed, actual angle, and actual angular rate, which are derived from the control system.

The control semantics are essentially the same for the various UAVs. The control process depends on the data gathered by the UAV sensors that perceive the state of the UAV. Meanwhile, these data are relevant since they represent the control process and the UAV's dynamic flight, which reflects the control semantics. Therefore, these data can be used to represent the control semantics of UAVs. It is reasonable to select these data types as features for model training. The flight data features of the above UAV PID control theory are summarized in Table 1 including the 12 features used for model training. The explanation and units of the features are provided in the 3rd and 4th columns, respectively. The 3rd, 5th, and 12th features (the altitude, vertical speed, and vertical acceleration) are collected from the flight altitude PID control system and the other features are collected from the horizontal flight position PID control system. The first two features represent the horizontal position. The 6th, 7th, and 8th features denote the actual angle. The 9th, 10th, and 11th features represent the actual angular rate. To train the ML models, these data were obtained from flight logs. Specified log messages that store these feature data are shown in the 5th column.

Table 1. Summary of flight data features for model training.

NO.	Features	Explanation	Units	Log Messages
1	latitude	latitude	degree	GPS
2	longitude	longitude	degree	GPS
3	altitude	altitude	m	GPS
4	horizontal speed	actual horizontal speed	m/s	GPS
5	vertical speed	actual vertical speed	m/s	GPS
6	roll	actual vehicle roll	degree	ATT
7	pitch	actual vehicle pitch	degree	ATT
8	yaw	actual vehicle yaw	degree	ATT
9	roll rate	actual vehicle roll rate	degree/s	RATE
10	pitch rate	actual vehicle pitch rate	degree/s	RATE
11	yaw rate	actual vehicle yaw rate	degree/s	RATE
12	vertical acceleration	actual vehicle vertical acceleration	cm/s/s	RATE

3.2. Control Semantics-Based Intelligent Detection Framework

The framework of our detection approach, ConstDet, is shown in Figure 3. It contains seven steps, as seen below.

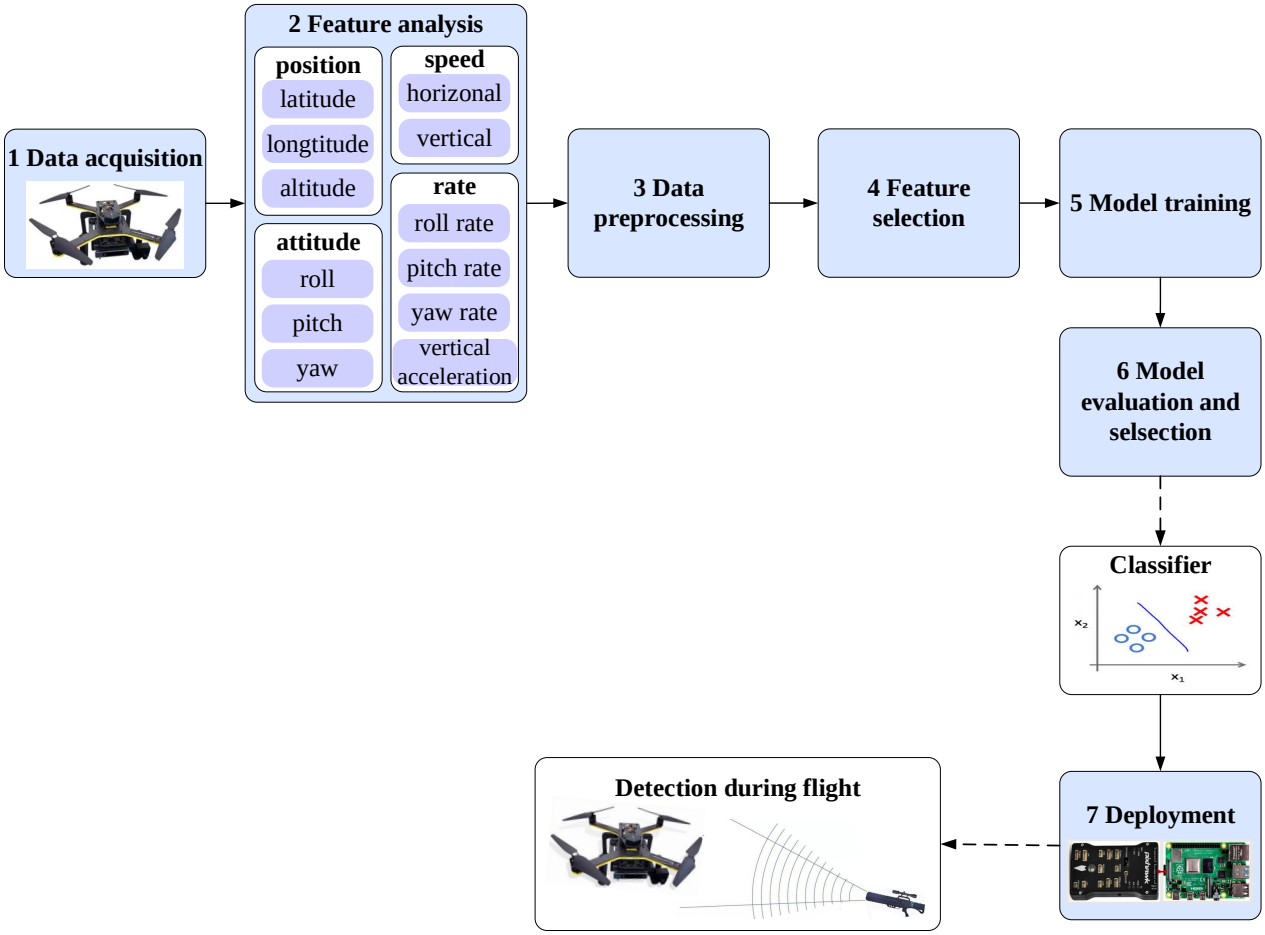

Figure 3. Framework of our detection approach.

(1) **Data acquisition.** To train ML models, we first collected the UAV data from the flight log generated by a real flight. The required data were discussed in the previous section (Section 3.1) and are shown in the 2nd column in Table 1 including the 3D position (latitude, longitude, and altitude), speed (horizontal and vertical speed), attitude (roll, pitch, and yaw), and rate (roll rate, pitch rate, yaw rate, and vertical acceleration). The flight log contained a lot of flight information such as the 3D position, speed, attitude, rate,

and so on. They were generated through specified log message structures [32] so that the original flight data could be recorded in the log. The relationship between the required data and log messages is shown in the 5th column in Table 1 and includes the GPS, ATT, and RATE messages. When ConstDet was implemented on a UAV, the data for detection were obtained from flight controllers in real time.

(2) **Feature analysis.** The feature analysis was carried out in Section 3.1. The selected features are presented in Table 1.

(3) **Data preprocessing.** Firstly, data filtering was carried out. Since the flight log of the attack scenario contained normal data and attack data, we distinguished between these data for the data collection. For example, to obtain the attack dataset, normal data needed to be removed from the log. Additionally, the sensors had different data acquisition frequencies so it was necessary to unify the frequency of these data. Applying the StandardScaler module is an important step in data preprocessing. There was a need to scale the original data in our dataset because the features had diverse proportions and different units, such as degree, meter, m/s, degree/s, and cm/s/s, as shown in the 4th column in Table 1, so the units cm/s/s were converted to m/s/s. When features are scaled, these multidimensional features have similar scales and the accuracy of models can be effectively improved. Finally, we obtained the dataset of the features for model training.

(4) **Feature selection.** Since we were not sure of the representative ability of each data feature to reflect the UAV position, there was a need to measure the importance of these features so that we could discard the lower correlated or even irrelevant features. This can save time and resources for model training and classification. Therefore, we first applied the random forest (RF) model to the dataset based on the information gain theory to compute the feature importance. Then, we chose the most informative features that had the most stable and lowest mean absolute errors (MAE).

(5) **Model training.** Different kinds of ML algorithms were applied for model training using the feature data including support vector machine (SVM), K-nearest neighbor (KNN), RF, gradient boosting decision tree (GBDT), decision tree (DT), multi-layer perceptron (MLP), and extreme gradient boosting (XGBoost). In this paper, various models were used so that we could (1) compare the performance of these models and (2) know which model was suitable for the detection of GPS spoofing attacks, and then the best detector could be built. SVM is suitable for a small sample and nonlinear datasets, as well as high-dimensional pattern recognition problems. However, it is not suitable for multiple classification problems and is sensitive to missing data so it is necessary to select the appropriate kernel function. KNN is simple and easy to understand. It requires no training and no parameter estimation. It is suitable for multi-label problems and has high accuracy. However, its prediction speed is slow and its interpretability is poor. If the sample number of a class is not balanced, accuracy will be affected. RF has low computational overhead and powerful performance in many real-world tasks. For imbalanced datasets, it can balance the errors. However, random forest has demonstrated that it will overfit for certain classification or regression problems with high noise. GBDT has high prediction accuracy. It can deal with nonlinear data and flexibly deal with various types of data including continuous and discrete values. However, it is difficult to train the data in parallel due to the dependency between the weak learners. DT is simple and easy to understand and can handle multiple output problems. However, it is easy to overfit, the generation of decision trees is unstable, and small data changes may lead to different generated decision trees. MLP has a good recognition rate and faster classification speed. However, it may lose the spatial information between pixels and only accept vector inputs. XGBoost can solve both the linear classification and logistic regression problems. XGBoost allows custom loss functions as long as the function supports first- and second-order derivatives. However, the space complexity of the pre-sorting process is too high and it needs to store not only the feature values but also the index of the gradient statistics of the corresponding sample of the feature, which consumes twice the memory. The dataset obtained through the data acquisition, data preprocessing, and feature selection steps was used as the input of the ML

algorithms. The dataset was divided into two parts: 70% of the data were used for training and 30% of the data were used for testing and evaluation.

(6) **Model evaluation and selection.** Various ML algorithms were used in this paper and they had different detection performances. To demonstrate the effectiveness of the trained models and determine which was the best model, we performed a model evaluation. ConstDet evaluated the trained models using several assessment criteria including accuracy, precision, recall, missing, mistake, and the F1-measure. Moreover, the receiver operating curve (ROC) function was used to demonstrate the discriminative potential of the classifiers. Based on the evaluation, the best classifier was selected as the detector of the ConstDet to detect GPS spoofing attacks on UAVs.

(7) **Deployment.** To apply ConstDet on a real UAV, we deployed the classifier on the companion computer of the UAV. Onboard detection was executed using real-time flight data, which were gathered from the flight controller via the MAVLink communication [33]. The original onboard flight data needed to be preprocessed before they were detected by the classifier because some feature data had different units to the training data from the flight log. Moreover, these real-time data also needed to be scaled as with the preprocessing of the training data so that the classifier was suitable for detection with these onboard data. After the deployment, the UAVs could detect GPS spoofing attacks during flight.

The purpose of our approach is for onboard detection of GPS spoofing attacks. We implemented the best classifier in a real UAV. After the detector was deployed in the UAV, it could detect GPS spoofing attacks during flights in real time. The detector was able to read real-time data from the flight controller and then preprocess these data to obtain the same data types and format as the trained data. Using the real-time and preprocessed data, the classifier computed and output the results about whether there were existing GPS spoofing attacks on the UAV.

4. Results

4.1. Experimental Setup

The UAV platform was constructed as shown in Figure 4. This quad-rotor UAV contained a flight controller and a companion computer. The flight controller, a Pixhawk 2.4.8 [34], was deployed using the flight control software ArduPilot 4.0.5 [35]. We only selected some sensors for the experiment, including a GPS receiver, an MPU-6000 (a three-axis accelerometer and a three-axis gyroscope), an HMC5883 (a magnetometer), and an MS5611 (a barometer). The companion computer was a Raspberry Pi 4B with 8G RAM on which Ubuntu 20.0.04 was installed. The Raspberry Pi 4B communicated with the flight controller to obtain the UAV data according to the MAVLink communication protocol [33] so it was able to obtain flight data from the flight controller and then detect whether there was a GPS spoofing attack on the UAV during the flight. The GPS, accelerometer, gyroscope, and magnetometer were connected with the flight controller to perceive the flight state of the UAV. The sampling frequencies of these sensors were set to 5 Hz, 25 Hz, 25 Hz, and 10 Hz, respectively.

We deployed the classifier on the companion computer and implemented the detection function using dronekit-python [36], which allowed us to control the UAV using the Python programming language. The classifier was deployed on the Raspberry Pi 4B using the Python programming language. The control and communication process followed the MAVLink protocol, which is a lightweight messaging protocol between onboard UAV components. MAVLink is the de facto communication protocol for UAVs. It is utilized not only by ArduPilot but also by PX4 [37], Paparazzi [38], and DJI [39].

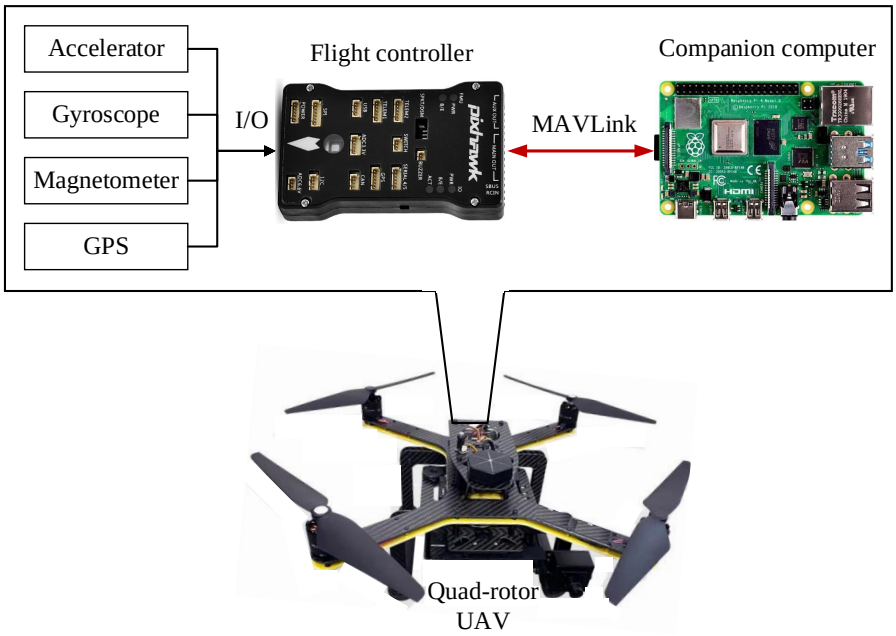

Figure 4. The structure of our UAV

The necessary feature data for detection are presented in Table 2 and correspond to the data from the flight log in Table 1. We used MAVLink messages [40] to transmit data from the flight controller to the companion computer. The third column in Table 2 shows the MAVLink messages through which the detector obtained the flight data. The units of the onboard flight data were different from those in the flight log. Thus, there was a need to ensure that the units of the required data were uniform. For example, since the unit for latitude in Table 1 is different from that in Table 2, the latitude obtained via the GPS_RAW_INT message was divided by 1×10^7 before the classifier used the latitude for the GPS spoofing detection.

Table 2. MAVLink messages for onboard feature data acquisition

NO.	Features	MAVLink Messages	Units
1	latitude	GPS_RAW_INT	degree $\times 1 \times 10^7$
2	longitude	GPS_RAW_INT	degree $\times 1 \times 10^7$
3	altitude	GPS_RAW_INT	mm
4	horizontal speed	GPS_RAW_INT	cm/s
5	vertical speed	Global_Position_INT	cm/s
6	roll	Attitude	rad
7	pitch	Attitude	rad
8	yaw	Attitude	rad
9	roll rate	Attitude	rad/s
10	pitch rate	Attitude	rad/s
11	yaw rate	Attitude	rad/s
12	vertical acceleration	RAW_IMU	cm/s/s

4.2. Data Acquisition

To gather actual flight data for model training, a UAV was used to experiment with normal flights and attacked flights. Different flight paths can affect the sensor data and control process. When a UAV is turning or flying in a curved line, the yaw will change quickly resulting in the desired value being different from the actual value. Although the difference exists, it should be seen as a normal flight. Moreover, when a UAV flies in a straight line as opposed to other flight paths, it is much easier to detect GPS spoofing attacks. Thus, the flight paths designed for the experiments included straight line paths, curved line paths, turning paths, ascending line paths, and descending line paths, as shown

in Figures 5 and 6. For the ascending and descending line paths shown in Figure 6, the UAV flew from a low position to a high position and from a high position to a low position, respectively. In such scenarios, the acceleration and speed in the z-axis change quickly. The actual acceleration and speed values differed from the desired values, which is a normal change for flight data. Thus, the flight experiments could cover various flight scenarios. In our experiments, GPS spoofing attacks were implemented using attack software, which can randomly generate the longitude and latitude to replace the GPS-based position so that the UAV will deviate from the actual position. The random spoofing positions outperformed the linear changing spoofing positions because when the changing pattern of the latter was fixed, it was easy to detect such kinds of spoofing positions.

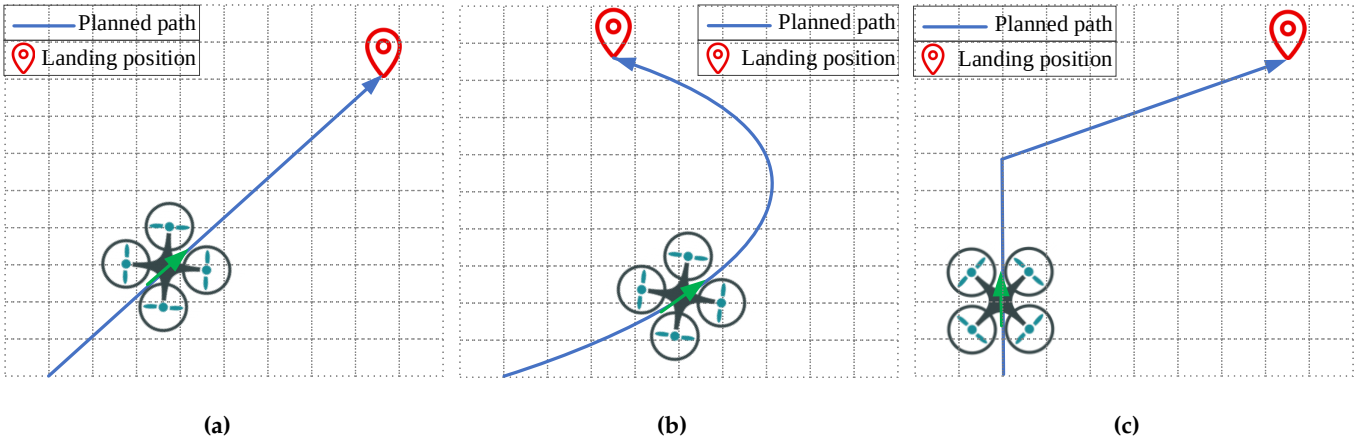

Figure 5. Flight paths. (**a**) Straight line path; (**b**) Curved line path; (**c**) Turning path.

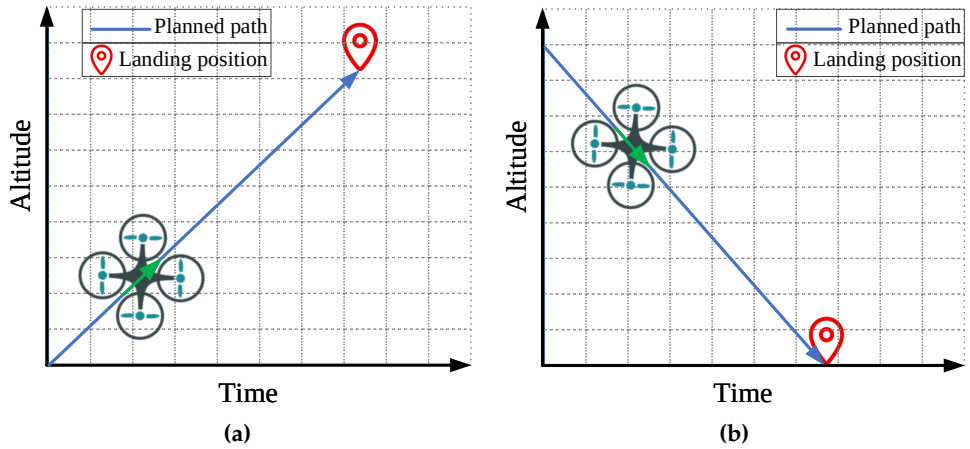

Figure 6. Flight paths. (**a**) Ascending path (from a low position to a high position); (**b**) Descending path (from a high position to a low position).

A picture of our real flight experiment is provided in Figure 7. It is dangerous to collect the attack data since UAVs may fly in a random direction because of random spoofing positions. During our experiments for data collection, the UAV hit a tree several times. To keep the UAV and people safe, we fastened a 50 m nylon cord to the bottom of the UAV.

Figure 7. An actual experimental attack picture: a nylon cord is fastened to the bottom of the UAV.

We experimented with normal and attacked flights. The experimental dataset was extracted from the flight log including the 3D position (latitude, longitude, and altitude), speed (horizontal and vertical speed), attitude (roll, pitch, and yaw), and rate (roll rate, pitch rate, yaw rate, and vertical acceleration). These feature data were stored in the GPS, ATT, and RATE parts of the flight log, as shown in the fifth column in Table 1.

4.3. Data Preprocessing

For the attacked flights, since we needed to ensure that the UAV took off safely in the real flight experiment, a UAV was not attacked at the beginning. After a UAV had taken off and reached more than 4 m in height, attacks were launched to deceive the UAV with the faked position. Thus, the flight log of the attacked flight contained both normal flight data and attacked flight data. We needed to distinguish between the attacked flight data and the normal flight data. To solve this problem, we checked the altitude and compared the differences between the desired values and the actual values for the latitude and longitude so that we could find out the time the GPS spoofing attacks started and the time the GPS spoofing attacks stopped. The attacked data were extracted from the flight log with the start and end times. Moreover, for the original dataset, data preprocessing was performed to unify the sampling frequencies. We set the frequency to 5 Hz for data preprocessing, that is, the interval between two pieces of preprocessed data was 0.2 s. After data preprocessing, the collected dataset was ready for model training and is summarized in Table 3. There were 10,296 pieces of data in total. The normal data contained 4950 pieces and the attacked data contained 5346 pieces.

Table 3. Our UAV dataset for model training.

Data Category	NO. of Data
Normal data	4950
Attacked data	5346
Total	10,296

4.4. Feature Selection

We analyzed the data features based on the control semantics in Section 3.1. There were a total of 12 features, as summarized in the second column in Table 4. We chose these features from the altitude control process and horizontal position control process.

Table 4. All features and their importance scores.

Serial Number	Feature	Importance Score
1	GyrXVar: Variance of Gyroscope X-axis	0.033768529723899314
2	GyrXStd: Standard deviation of Gyroscope X-axis	0.025600626113929277
3	GyrXMean: Mean of Gyroscope X-axis	0.03780320826069406
4	GyrYVar: Variance of Gyroscope Y-axis	0.01755298967944973
5	GyrYStd: Standard deviation of Gyroscope Y-axis	0.026789173200955398
6	GyrYMean: Mean of Gyroscope Y-axis	0.015273964748774242
7	GyrZVar: Variance of Gyroscope Z-axis	0.007971069715234017
8	GyrZStd: Standard deviation of Gyroscope Z-axis	0.00799819035975945
9	GyrZMean: Mean of Gyroscope Z-axis	0.008285023600927378
10	AccXVar: Variance of accelerometer X-axis	0.01273626523457876
11	AccXStd: Standard deviation of accelerometer X-axis	0.005590607777735804
12	AccXMean: Mean of accelerometer X-axis	0.01550091392149543

To select the most significant features, firstly, the RF model was run on the dataset to choose the appropriate number of decision trees. This number was related to the accuracy of the classification. We used the out-of-bag (OOB) error to optimize this number. The OOB error represented the misclassification probability. Thus, we selected the smaller number, which reduced the training time and saved computing resources. This ensured that the OOB error was relatively stable and low and would not seriously fluctuate. In this paper, we computed the OOB error with the varying number of trees from 15 to 600, as shown in Figure 8. It can be observed that the OOB error tended to be stable and close to the minimum value (about 0.012) when the number of trees was 370, which was a parameter for calculating the feature's importance.

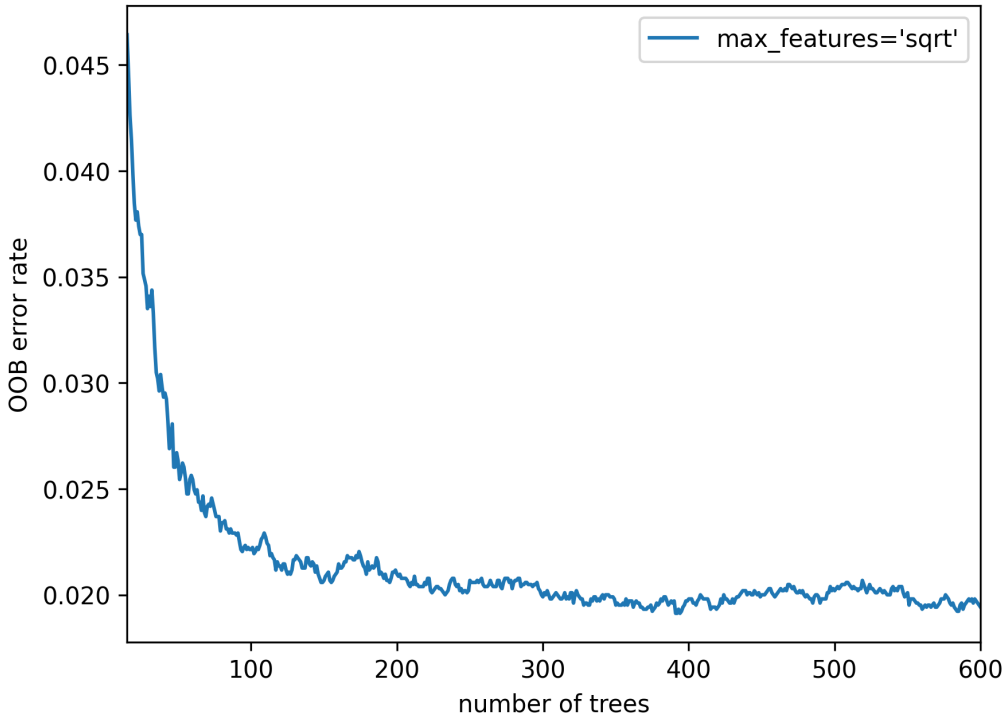

Figure 8. OOB error rate versus the number of trees.

Secondly, we used the RF model with 370 trees to measure the importance of the features. The serial numbers of the features and their corresponding importance scores are provided in Table 4. The importance of each feature was computed according to the information gain it provided [41]. For each node of the tree, the information gain represented the information entropy change of the node from the current state to the proposed state. We sorted these features based on their importance in descending order, as presented in Figure 9. We can see that feature 3 had a higher score than the other features. This feature was the altitude in Table 4. As seen in Figure 9, from feature 2 to feature 7, the importance score decreased steadily, from about 0.1 to 0.07. From feature 6 to feature 9, the importance score basically remained unchanged. From feature 10 to feature 12, the importance score stayed at around 0.02.

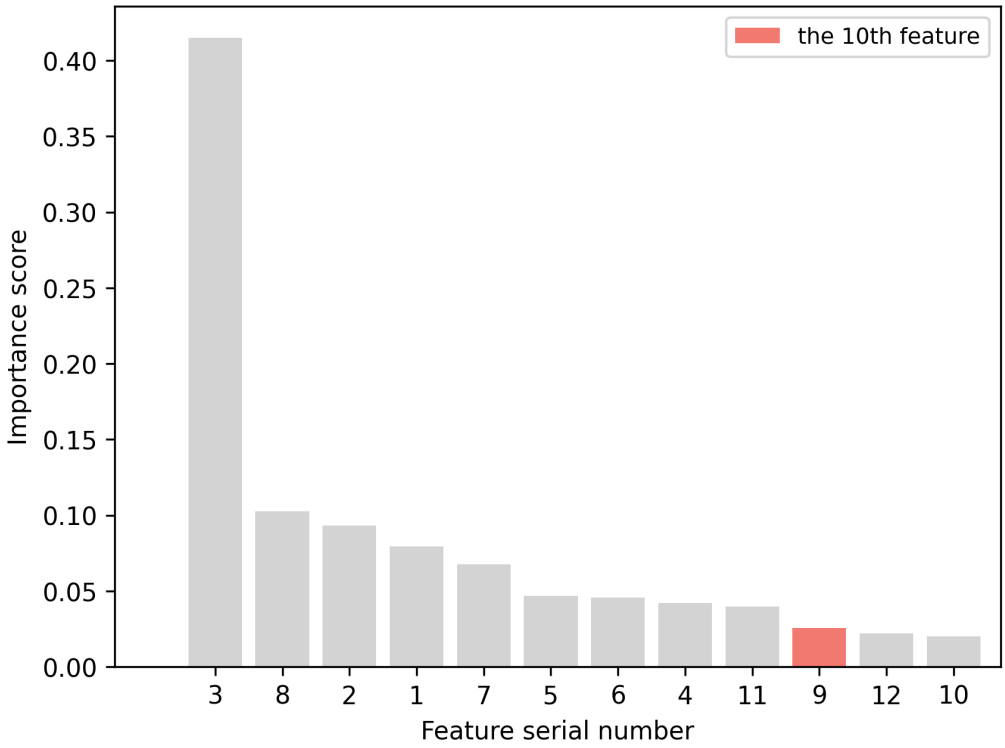

Figure 9. Feature importance sorting.

Finally, the MAE was generated by running the RF model with the varying features from 1 to 12. The MAE is a measure of errors between the original data and predicted data. The most informative features were selected based on the minimum MAE. The number of features and the corresponding MAEs are shown in Figure 10. It can be observed that when the number of features was 11, the MAE had the lowest value. Therefore, we selected 11 features for model training. They had a significant contribution to the representation of the UAV position. The elimination of features improved the efficiency of the classifier and decreased the training time. By querying the importance sequence seen in Figure 9, the 11 selected features were 3, 8, 2, 1, 7, 5, 6, 4, 11, 9, and 12. The names of these features were queried based on the serial numbers in Table 4.

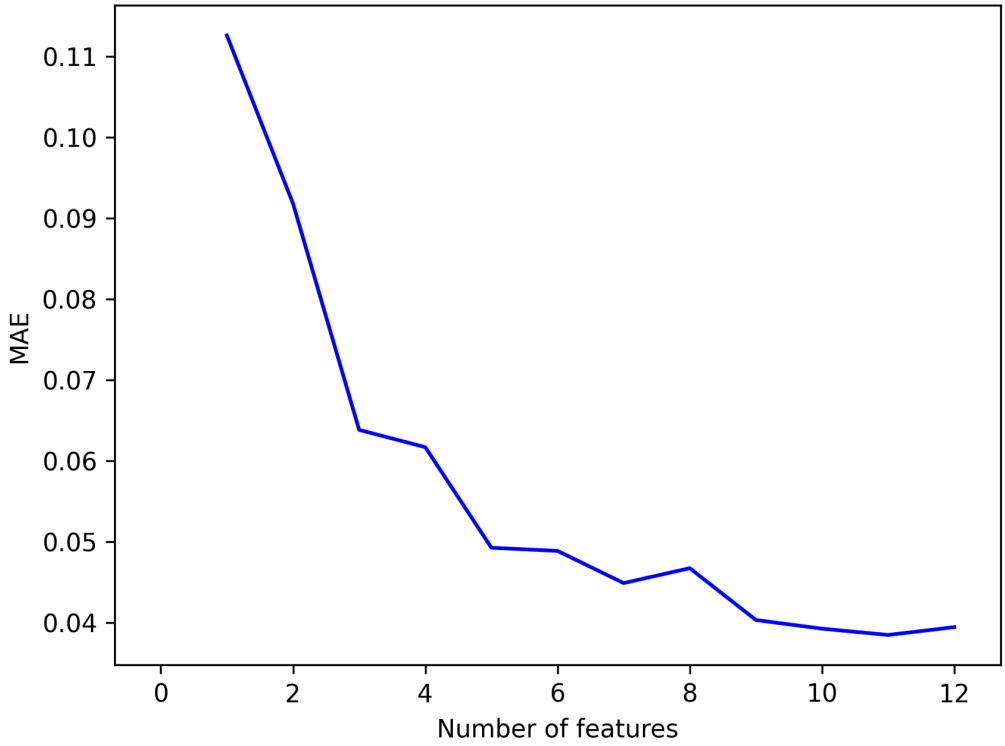

Figure 10. MAE vs. number of features.

4.5. Model Training, Evaluation, and Selection

To train the various ML algorithms, 70% of the data were applied including the SVM, KNN, RF, GBDT, DT, MLP, and XGBoost, and 30% of the data were utilized to evaluate the trained models. We ran these algorithms on the 11 selected features (discussed in Section 4.4) of the dataset.

Receiver operating curves (ROC) are often used to illustrate the validity and classification performance of ML algorithms. An ROC curve is used to express the relationship between the false positive rate (FPR) and the true positive rate (TPR) with a range of assessment scores for the determination of a binary classification. The performance of ConstDet is shown in Figure 11, including the various ML models. The area under the curve (AUC) was calculated for the model evaluation. As shown in the figure, XGBoost was the best model of ConstDet with a value of 0.996887, followed by RF (0.996782), GBDT (0.974714), DT (0.958309), SVM-*rbf* (0.922729), KNN (0.860934), MLP (0.853909), and SVM-*linear* (0.767151).

The classifier needed to compute the score of a sample for the determination of the classification. This score was the threshold used to determine whether the sample was normal or malicious. Different scores can produce different classification performances. In this paper, we computed the threshold value (the score) by maximizing the TPR and minimizing the FPR. The optimal cutoff points for all classifiers in the ROC curve represent this value. They are labeled in Figure 12 in the zoomed-in ROC curve. The FPR, TPR, and corresponding thresholds of the different models are summarized in Table 5. The XGBoost model had the highest TPR (96.82%) and the lowest FPR (1.32%). The RF model had similar TPR and FPR values.

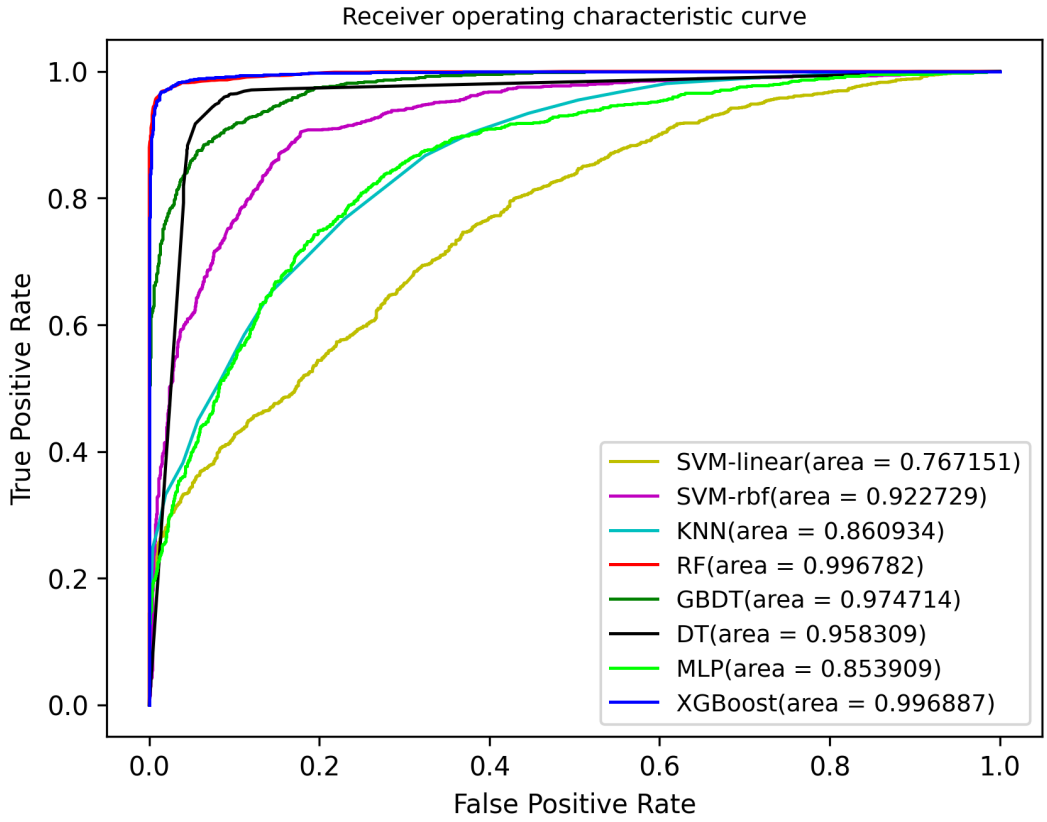

Figure 11. ROC curve.

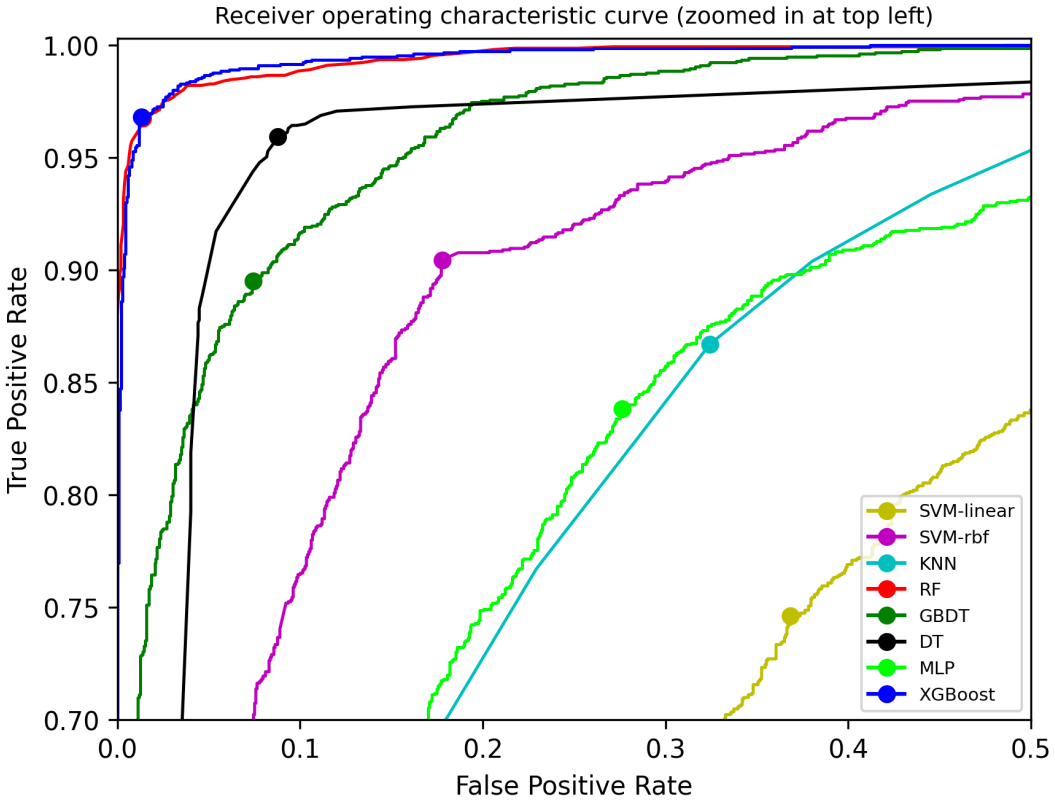

Figure 12. Zoomed-in ROC curve in the top left.

Table 5. Optimal thresholds of each model.

Model	FPR(%)	TPR(%)	Threshold
SVM-*linear*	36.85	74.62	0.458250
SVM-*rbf*	17.80	90.46	0.433721
KNN	32.43	86.70	0.500000
RF	1.38	96.76	0.550000
GBDT	7.45	89.50	0.556963
DT	8.77	95.93	0.400000
MLP	27.62	83.84	0.314078
XGBoost	1.32	96.82	0.670852

For the performance evaluation of ConstDet with the optimal thresholds (in the fourth column in Table 5), the accuracy, precision, recall, missing, mistake, and F1-measure were calculated. The formulae of these measures are presented in Equations (1)–(6), respectively. Explanations are provided below the equations.

$$Accuracy = \frac{\#TP + \#TN}{\#TP + \#TN + \#FP + \#FN} \tag{1}$$

$$Precision = \frac{\#TP}{\#TP + \#FP} \tag{2}$$

$$Recall = \frac{\#TP}{\#TP + \#FN} \tag{3}$$

$$Missing = \frac{\#FN}{\#TP + \#FN} \tag{4}$$

$$Mistake = \frac{\#FP}{\#TN + \#FP} \tag{5}$$

$$F1 = \frac{2Precision * Recall}{Precision + Recall} \tag{6}$$

True positive (*TP*) means that an attack was detected correctly. True negative (*TN*) means that a normal GPS signal was detected correctly, that is, the detector knew that the UAV was flying properly. False negative (*FN*) means that an attack was detected incorrectly as a normal GPS signal. False positive (*FP*) means that the normal GPS signal was detected incorrectly as an attack.

Accuracy means the correct detection rate for GPS spoofing attacks. Precision means the percentage of correctly detected attacks from all the results detected as attacks. Recall means the percentage of correctly detected attacks from all the actual attacks. Missing means the degree of actual attacks detected incorrectly as normal GPS signals from all the actual attacks. Mistake means the percentage of normal GPS signals detected incorrectly as attacks from all the normal GPS signals.

The experimental results are provided in Table 6. The XGBoost model had the best performance. Its accuracy, precision, recall, missing, mistake, and F1-measure had the best performance over the other models. Therefore, the XGBoost classifier was selected as the detector for ConstDet. This means that the detection rate of ConstDet was 97.70%.

Table 6. Experimental results of various ML models with optimal thresholds.

Model	Accuracy (%)	Precision (%)	Recall (%)	Missing (%)	Mistake (%)	F1-Measure (%)
SVM-*linear*	68.95	67.71	74.55	25.45	36.85	70.97
SVM-*rbf*	86.34	84.02	90.33	9.67	17.80	87.06
KNN	77.11	75.10	82.32	17.68	28.28	78.54
RF	97.57	98.70	96.50	3.50	1.32	97.59
GBDT	90.97	92.56	89.44	10.56	7.45	90.97
DT	93.59	91.94	95.80	4.20	8.70	93.83
MLP	78.18	75.86	83.78	16.22	27.62	79.63
XGBoost	97.70	98.70	96.76	3.24	1.32	97.72

The XGBoost is an optimized distributed gradient boosting tool. It provides parallel tree boosting (also known as GBDT, GBM) to solve data classification problems, so it achieved the highest detection rate. The SVM with the linear kernel had the worst performance, as shown in Table 6 because it tried to directly classify the original samples in a linear way. However, the UAV data did not follow a linear distribution so this model had the worst performance. The SVM with the rbf kernel had much better performance. Since it added more characteristics to the samples using the rbf kernel, this allowed the samples to be further classified in a linear way.

4.6. Deployment of the Best Detection Model

The best detection model, the XGBoost model, was deployed on the UAV. The real-time flight data were preprocessed and then used for detection. The data were preprocessed as explained in Section 4.3. Additionally, another data preprocessing step was carried out. Since the data units of some real-time data were different from the data in the flight log, we needed to unify the data units before the onboard detector used the real-time data. For example, the units of horizontal speed, vertical speed, roll, yaw, and pitch were different, which can be seen in Tables 1 and 2.

5. Discussion: Reproduction and Comparison of Existing Detection Methods

5.1. Reproduction and Comparison

Feng et al. [14] applied the XGBoost and SVM algorithms to detect GPS spoofing attacks (called the JSA method). Since their code and dataset were not open source, the JSA method was implemented by us and evaluated using our dataset so that we could compare it with our ConstDet method. The implementation and comparison are explained below.

To implement the JSA method, first, the original data obtained from the flight log were preprocessed. The angular velocity and acceleration were selected as the features. The GPS data, including latitude, longitude, and altitude, were utilized to compute the distance between two positions as a feature. The sampling time was 0.2 s, which was the same value as our ConstDet approach. The preprocessed data of our dataset contained 10,298 items including 4951 normal data and 5347 attacking data. Second, 70 percent of the preprocessed data were used to train the models (XGBoost and SVM) and 30 percent were used for testing. The XGBoost model using the same optimized parameters provided by the JSA method was trained, and these optimized parameters were $learning\_rate = 0.153$, $min\_child\_weight = 0.636$, $max\_depth = 4$, $gamma = 0.123$, and $subsample = 0.5$. The XGBoost model using the default parameters was also trained for a comparison with our ConstDet approach. In addition, the SVM model with the *rbf* kernel and *linear* kernel was also trained using the default parameters. To evaluate the performance of the trained models, the accuracy, precision, recall, missing, mistake, and F1-measure were calculated to support the comparison, as shown in Table 6.

For the JSA method, the XGBoost models with the default parameters or optimized parameters had similar performances, as shown in Table 7. We can see that ConstDet had better performance than the JSA method, as shown in the last three rows in Table 7, even though the JSA method used the optimized model parameters. Moreover, many of the models (such as RF, DT, GBDT, and SVM-*rbf*) in our detection framework in Table 6

also had better performance than the JSA method, as seen in Table 7. We can conclude that the proposed ConstDet approach is better than the JSA method.

Table 7. Comparison of experimental results between the JSA method [14] and our ConstDet approach. We implemented the JSA method. For the JSA method, the SVM-*linear*, SVM-*rbf*, XGBoost (with default parameters), and XGBoost (with optimized parameters provided by the JSA method) models were trained and tested using our UAV dataset. The last row is the performance of our ConstDet approach using the default ML model parameters.

Model	Accuracy (%)	Precision (%)	Recall (%)	Missing (%)	Mistake (%)	F1-Measure (%)
SVM-*linear*	77.51	85.30	68.11	31.89	12.49	75.74
SVM-*rbf*	79.13	87.62	69.30	30.70	10.42	77.39
XGBoost (default parameters)	84.56	86.95	82.42	17.58	13.16	84.63
XGBoost (optimized parameters)	84.66	88.77	80.41	19.59	10.82	84.39
Our ConstDet (default parameters)	97.70	98.70	96.76	3.24	1.32	97.72

Feng et al. proposed the DATE method [20] and the TECS method [21] for the detection of GPS spoofing attacks. These methods were compared with the JSA method in terms of the correctness ratio provided in [14]. The JSA method was much better than the DATE method and TECS method. In this paper, the comparison shows that our ConstDet approach was better than the JSA method. Thus, we can conclude that our ConstDet approach is also better than both the DATE method and the TECS method.

In our experiment, we found that one of the weaknesses of the DATE and TECS methods was that they only considered some flight data and a few kinds of data relationships, which could not sufficiently represent the principle of the position calculation. This means that these methods could not detect flight data samples in various flight scenarios.

5.2. Comparison of Accuracy with Existing Works

ConstDet was compared with prior works that provided detection rates such as the *LSTM-based* detection method of Wang et al. [17], the *Crowd-GPS-Sec* detection method of Jansen et al. [25], the *multi-UAV-based* detection method of Liang et al. [22], and the *DeepSIM* detection method of Xue et al. [13], as shown in Table 8.

Table 8. Comparison of the performance of ConstDet and those of existing works. *LSTM-based*, *DeepSIM*, and ConstDet are ML-based detection approaches. *Crowd-GPS-Sec* and *multi-UAV-based* are non-ML-based detection approaches.

Approach	Reference	Detection Rate (%)	Experimental Data
LSTM-based	Wang et al. [17]	78	simulated data
Crowd-GPS-Sec	Jansen et al. [25]	75	real data and simulated data
multi-UAV-based	Liang et al. [22]	98.6 (>4 UAVs, on-ground detection) 96.7 ($\leq$4 UAVs, on-ground detection)	simulated data
DeepSIM	Xue et al. [13]	94.8 (on-ground detection) 89 (onboard detection)	real data
ConstDet	Ours	97.70	real data

The *multi-UAV-based* approach required several UAVs working together. The *Crowd-GPS-Sec* method needed the GPS data broadcasted by other UAVs or aircraft. These two methods are not suitable for a single-flying UAV.

In general, it is easier to derive a higher detection rate for on-ground detection than for onboard detection since there are many more computing resources on the ground so more complex detection approaches can be applied for spoofing detection. ConstDet can be an onboard or on-ground detection approach. In addition, ConstDet is based on real data, which makes the detection framework more reliable. Moreover, the detection rate of ConstDet is 97.70%, which is higher than most existing methods, as shown in the third

column in the above table. Real data can make the detector reliable and practical. By taking both the detection rate and experimental data (the real data or simulated data) into account, we can conclude that ConstDet is better than existing detection approaches.

6. Conclusions

GPS spoofing attacks are the primary threats to UAV security. It is not difficult to implement GPS spoofing devices and these attacks can lead to catastrophic consequences. Thus, this paper focuses on GPS spoofing attacks on UAVs and proposes a detection approach, ConstDet, based on the control semantics using ML algorithms. The control semantics represent the principles of the UAV control process using flight data. The control process consists of the horizontal position control process and the altitude control process. The flight data are distinctly important in the flight control process and can affect the control ability. The relationship between these flight data must satisfy the control semantics. In other words, these UAV flight data can reflect the control semantics. Thus, flight data features are analyzed and selected based on the control semantics. Features are selected from the flight data produced by the horizontal position control system and altitude control system. To design a practical detector, various experiments are conducted to collect real flight data, supporting the proposal of the intelligent detection approach, ConstDet. Different ML algorithms are trained and evaluated using an actual flight dataset to obtain the best classifier for ConstDet. We implement ConstDet in a real UAV. Through the evaluation, the experiments demonstrate that ConstDet is better than the existing works. Our work shows that the control semantics have a crucial relationship with the flight data, which is effective for the detection of GPS spoofing attacks. One of the reasons for the low detection rates of some existing works is that they only consider part of the flight data, which cannot comprehensively represent the relationship between the position-related flight data.

Our work can be extended in the following directions. First, PID controllers are indispensable components for UAVs and they are critical components for ensuring a stable flight. Thus, our control semantics-based detection approach is suitable for all kinds of UAVs. It can also be extended to the detection of spoofing attacks on other kinds of UAV sensors. Moreover, we plan to test UAVs using actual GPS spoofing signals in an indoor environment to further demonstrate the effectiveness of our approach. It is difficult to conduct such experiments because actual attacks on UAVs are dangerous and it is illegal to produce high-power spoofing signals in an open environment. These are also important reasons why existing works do not test their detection approaches with spoofing attacks on a real flight.

Author Contributions: Conceptualization, X.W.; methodology, X.W. and C.S.; software, X.W.; validation, X.W., C.S., M.L., Q.S. and Y.L.; formal analysis, C.S., M.L., Q.S. and Y.L.; investigation, X.W., M.L., Q.S. and Y.L.; writing—original draft preparation, X.W.; writing—review and editing, X.W., C.S., M.L., Q.S. and Y.L.; visualization, X.W. and C.S.; supervision, C.S.; project administration, X.W.; funding acquisition, X.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the Natural Science Basic Research Program of Shaanxi (No.2021JQ-207). This work was supported by the National Natural Science Foundation of China (No.62272366), the Fundamental Research Funds for the Central Universities (No.XJS211506), and the Natural Science Basic Research Program of Shaanxi (No.2022JQ-621 and 2022JQ-658).

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: https://drive.google.com/drive/folders/1S3LqZCU48lztQxuGzEJN88TOoucJEDYB?usp=sharing accessed on 1 August 2022.

Conflicts of Interest: The authors declare that they have no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

GNSS	global navigation satellite system
GCS	ground control station
ML	machine learning
CONSTDET	**con**trol **s**eman**t**ics-based **det**ection approach
MAE	mean absolute error
MLP	multi-layer perceptron
SITL	software-in-the-loop
GANs	generative adversarial networks
SDR	software-defined radio
LSTM	long short-term memory
SVM	support vector machine
FANETs	flying adhoc networks
KNN	K-nearest neighbor
RF	random forest
GBDT	gradient boosting decision tree
DT	decision tress
XGBoost	extreme gradient boosting
ROC	receiver operating curve
TP	true positive
TN	true negative
TN	false negative
FP	false positive
AUC	area under the curve

References

1. Horton, E.; Ranganathan, P. Development of a GPS spoofing apparatus to attack a DJI Matrice 100 Quadcopter. *J. Glob. Position. Syst.* **2018**, *16*, 1–11. [CrossRef]
2. Seo, S.H.; Lee, B.H.; Im, S.H.; Jee, G.I. Effect of spoofing on unmanned aerial vehicle using counterfeited GPS signal. *J. Position. Navig. Timing* **2015**, *4*, 57–65. [CrossRef]
3. Shepard, D.P.; Bhatti, J.A.; Humphreys, T.E.; Fansler, A.A. Evaluation of smart grid and civilian UAV vulnerability to GPS spoofing attacks. In Proceedings of the 25th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2012), Nashville, TN, USA, 17–21 September 2012; pp. 3591–3605.
4. Humphreys, T. *Statement on the Vulnerability of Civil Unmanned Aerial Vehicles and Other Systems to Civil GPS Spoofing*; The University of Texas at Austin: Austin, TX, USA, 18 July 2012; pp. 1–16.
5. Gaspar, J.; Ferreira, R.; Sebastião, P.; Souto, N. Capture of UAVs through GPS spoofing using low-cost SDR platforms. *Wirel. Pers. Commun.* **2020**, *115*, 2729–2754. [CrossRef]
6. Wang, K.; Chen, S.; Pan, A. Time and position spoofing with open source projects. *Black Hat Eur.* **2015**, *148*, 1–8.
7. Mendes, D.; Ivaki, N.; Madeira, H. Effects of GPS Spoofing on Unmanned Aerial Vehicles. In Proceedings of the 2018 IEEE 23rd Pacific Rim International Symposium on Dependable Computing (PRDC), Taipei, Taiwan, 4–7 December 2018; pp. 155–160. [CrossRef]
8. Arteaga, S.P.; Hernández, L.A.M.; Pérez, G.S.; Orozco, A.L.S.; Villalba, L.J.G. Analysis of the GPS spoofing vulnerability in the drone 3DR solo. *IEEE Access* **2019**, *7*, 51782–51789. [CrossRef]
9. Kerns, A.J.; Shepard, D.P.; Bhatti, J.A.; Humphreys, T.E. Unmanned aircraft capture and control via GPS spoofing. *J. Field Robot.* **2014**, *31*, 617–636. [CrossRef]
10. Gaspar, J.; Ferreira, R.; Sebastião, P.; Souto, N. Capture of UAVs Through GPS Spoofing. In Proceedings of the 2018 Global Wireless Summit (GWS), Chiang Rai, Thailand, 25–28 November 2018; pp. 21–26. [CrossRef]
11. Ding, Y.; Fu, Z. Multi-UAV Cooperative GPS Spoofing Based on YOLO Nano. *J. Cybersecur.* **2021**, *3*, 69. [CrossRef]
12. Guo, Y.; Wu, M.; Tang, K.; Tie, J.; Li, X. Covert spoofing algorithm of UAV based on GPS/INS-integrated navigation. *IEEE Trans. Veh. Technol.* **2019**, *68*, 6557–6564. [CrossRef]
13. Xue, N.; Niu, L.; Hong, X.; Li, Z.; Hoffaeller, L.; Pöpper, C. DeepSIM: GPS Spoofing Detection on UAVs Using Satellite Imagery Matching. In *Annual Computer Security Applications Conference*; Association for Computing Machinery: New York, NY, USA, 2020; pp. 304–319. [CrossRef]
14. Feng, Z.; Guan, N.; Lv, M.; Liu, W.; Deng, Q.; Liu, X.; Yi, W. Efficient drone hijacking detection using two-step GA-XGBoost. *J. Syst. Archit.* **2020**, *103*, 101694. [CrossRef]

15. Kim, K.H.; Nalluri, S.; Kashinath, A.; Wang, Y.; Mohan, S.; Pajic, M.; Li, B. Security Analysis against Spoofing Attacks for Distributed UAVs. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS'16), Vienna, Austria, 24–28 October 2012; Association for Computing Machinery: New York, NY, USA, 2020.
16. Calvo-Palomino, R.; Bhattacharya, A.; Bovet, G.; Giustiniano, D. Short: LSTM-based GNSS Spoofing Detection Using Low-cost Spectrum Sensors. In Proceedings of the 2020 IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM), Cork, Ireland, 31 August–3 September 2020; pp. 273–276.
17. Wang, S.; Wang, J.; Su, C.; Ma, X. Intelligent Detection Algorithm Against UAVs' GPS Spoofing Attack. In Proceedings of the 2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS), Hong Kong, China, 2–4 December 2020; pp. 382–389. [CrossRef]
18. Panice, G.; Luongo, S.; Gigante, G.; Pascarella, D.; Di Benedetto, C.; Vozella, A.; Pescapè, A. A SVM-based detection approach for GPS spoofing attacks to UAV. In Proceedings of the 2017 23rd International Conference on Automation and Computing (ICAC), Huddersfield, UK, 7–8 September 2017; pp. 1–11. [CrossRef]
19. Talaei Khoei, T.; Ismail, S.; Kaabouch, N. Dynamic selection techniques for detecting GPS spoofing attacks on UAVs. *Sensors* **2022**, *22*, 662. [CrossRef] [PubMed]
20. Feng, Z.; Guan, N.; Lv, M.; Liu, W.; Deng, Q.; Liu, X.; Yi, W. Efficient drone hijacking detection using onboard motion sensors. In Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017, Lausanne, Switzerland, 27–31 March 2017; pp. 1414–1419.
21. Feng, Z.; Guan, N.; Lv, M.; Liu, W.; Deng, Q.; Liu, X.; Yi, W. An efficient UAV hijacking detection method using onboard inertial measurement unit. *ACM Trans. Embed. Comput. Syst. (TECS)* **2018**, *17*, 1–19. [CrossRef]
22. Liang, C.; Miao, M.; Ma, J.; Yan, H.; Zhang, Q.; Li, X.; Li, T. Detection of GPS spoofing attack on unmanned aerial vehicle system. In *International Conference on Machine Learning for Cyber Security*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 123–139.
23. Meng, L.; Yang, L.; Ren, S.; Tang, G.; Zhang, L.; Yang, F.; Yang, W. An Approach of Linear Regression-Based UAV GPS Spoofing Detection. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 5517500. [CrossRef]
24. Eldosouky, A.; Ferdowsi, A.; Saad, W. Drones in distress: A game-theoretic countermeasure for protecting UAVs against GPS spoofing. *IEEE Internet Things J.* **2019**, *7*, 2840–2854. [CrossRef]
25. Jansen, K.; Schäfer, M.; Moser, D.; Lenders, V.; Pöpper, C.; Schmitt, J. Crowd-GPS-Sec: Leveraging Crowdsourcing to Detect and Localize GPS Spoofing Attacks. In Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP), Francisco, CA, USA, 20–24 May 2018; pp. 1018–1031. [CrossRef]
26. Bada, M.; Boubiche, D.E.; Lagraa, N.; Kerrache, C.A.; Imran, M.; Shoaib, M. A policy-based solution for the detection of colluding GPS-Spoofing attacks in FANETs. *Transp. Res. Part A Policy Pract.* **2021**, *149*, 300–318. [CrossRef]
27. Basan, E.; Basan, A.; Nekrasov, A.; Fidge, C.; Sushkin, N.; Peskova, O. GPS-spoofing attack detection technology for UAVs based on Kullback–Leibler divergence. *Drones* **2021**, *6*, 8. [CrossRef]
28. Davidovich, B.; Nassi, B.; Elovici, Y. Towards the Detection of GPS Spoofing Attacks against Drones by Analyzing Camera's Video Stream. *Sensors* **2022**, *22*, 2608. [CrossRef] [PubMed]
29. Quan, Q. *Introduction to Multicopter Design and Control*; Springer: Berlin/Heidelberg, Germany, 2017.
30. Mendoza-Mendoza, J.A.; Gonzalez-Villela, V.; Sepulveda-Cervantes, G.; Mendez-Martinez, M.; Sossa-Azuela, H. *Advanced Robotic Vehicles Programming: An Ardupilot and Pixhawk Approach*; Apress: New York, NY, USA, 2020.
31. Kumar, V.; Minz, S. Feature selection: A literature review. *SmartCR* **2014**, *4*, 211–229. [CrossRef]
32. Log Messages. Available online: https://ardupilot.org/copter/docs/logmessages.html (accessed on 8 August 2022).
33. MAVLINK: Micro Air Vehicle Communication Protocol. Available online: https://mavlink.io/ (accessed on 8 August 2022).
34. Pixhawk Overview. Available online: https://ardupilot.org/copter/docs/common-pixhawk-overview.html (accessed on 8 August 2022).
35. Ardupilot. Available online: https://ardupilot.org/ (accessed on 8 August 2022).
36. DroneKit-Python. Available online: https://dronekit-python.readthedocs.io/en/latest/ (accessed on 8 August 2022).
37. PX4 Open Source Autopilot-for Drone Developers. Available online: https://px4.io (accessed on 8 August 2022).
38. Paparazzi—The Free Autopilot. Available online: https://wiki.paparazziuav.org/wiki/Main_Page (accessed on 8 August 2022).
39. DJI Phantom 4 Pro V2.0. Available online: https://www.dji.com/uk/phantom-4-pro-v2?site=brandsite&from=nav (accessed on 8 August 2022).
40. MAVLINK Common Message Set. Available online: https://mavlink.io/en/messages/common.html (accessed on 8 August 2022).
41. Khalid, S.; Khalil, T.; Nasreen, S. A survey of feature selection and feature extraction techniques in machine learning. In Proceedings of the 2014 Science and Information Conference, London, UK, 27–29 August 2014; pp. 372–378. [CrossRef]