

Article

A Machine Learning-Powered Solution for Safe Autonomous Robotic Ground Navigation in Cyber-Contested Environments

Tianjian Wan, Khair Al Shamaileh * and Mustafa Alkhatib

Electrical and Computer Engineering Department, Purdue University Northwest, Hammond, IN 46323, USA; wan159@pnw.edu (T.W.); mostafaalkhatib77@gmail.com (M.A.)

* Correspondence: kalshama@pnw.edu

Featured Application

Detection and classification of cyberattacks targeting autonomous ground vehicles, particularly those with fixed-route navigation, find applications in public transportations.

Abstract

In this article, machine learning (ML) is proposed as a solution to detect and classify false message injection attacks in autonomous ground navigation. First, multiple trajectories are designed and simulated to collect authentic feature samples offered by the odometry and inertial measurement unit (IMU) of an autonomous ground vehicle (UGV). Then, a dataset comprising these samples and other injected samples that simulate two cyberattacks, namely path modification (PM) and velocity drift (VD), is created to train, validate, and benchmark various ML classification models. These include decision tree (DT), k-nearest neighbors (KNN), multi-layer perceptron (MLP), random forest (RF), and support vector machine (SVM). The optimum classification model is experimentally evaluated using a UGV platform, and results suggest that the proposed solution allows the detection of authentic and attacked messages with more than 98% average accuracy and sub-millisecond prediction time. Thus, this solution is ideal for real-time classification, especially in fixed-route applications, e.g., public transportation.

Keywords: autonomous navigation; machine learning (ML); message injection cyberattacks; path modification (PM); unmanned ground vehicles (UGVs); velocity drift (VD)

1. Introduction

The adoption of autonomous unmanned ground vehicles (UGVs) has been witnessing exponential growth recently for their benefits in a multitude of applications, including scientific expeditions, search and rescue missions, exploring hard-to-reach terrains, and the auto industry [1–8]. This growth had been backed by a market size of USD 3.12B in 2025, which is projected to reach USD 79.83B by 2034 [9]. This increase in market size is attributed to the continued demand for automation and rapid advancement in enabling software and hardware technologies (e.g., navigation protocols, sensory equipment). Therefore, state-of-the-art research has dedicated significant attention to promoting autonomous navigation, especially in dynamic environments, with the use of data fusion techniques [10–13], collaborative vehicle-to-vehicle (V2V) networking [14–17], route optimization algorithms [18–21], and most recently, artificial intelligence (AI) [22–25]. Fusion techniques focus on multi-modal sensor integration strategies to provide accurate environmental perception and localization, enabling vehicles to navigate complex landscapes while minimizing collisions.



Academic Editor: DaeEun Kim

Received: 1 July 2026

Revised: 28 July 2026

Accepted: 29 July 2026

Published: 2 August 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

On the other hand, V2V networking facilitates cooperative perception and decentralized coordination to allow real-time data exchange that offers navigation within a network of autonomous vehicles. Furthermore, bio-inspired optimization algorithms use nature-mimicking mechanisms, e.g., swarm intelligence, to address multi-objective navigation challenges that achieve enhanced navigation efficiency and adaptability in highly dynamic environments. Lastly, AI-driven navigation leverages machine learning (ML), deep learning (DL), and reinforcement learning to enable adaptive autonomy via continuous policy optimizations.

Although much attention has been paid to continuously refining the accuracy and trustworthiness of autonomous navigation solutions, fewer research efforts have addressed the associated cybersecurity concerns. These concerns stem from the proneness of UGVs to cyberattacks at the physical layer, e.g., denial-of-service (DoS), location spoofing, and false message injections. Such attacks potentially result in irreversible consequences featuring compromising sensitive data, capturing payloads, damaging properties, or worse, leading to human casualties. For instance, the vulnerabilities of a self-driving vehicle, e.g., Tesla car, was demonstrated in [26,27] by jeopardizing its engine control unit. Moreover, in [28,29], counterfeit satellite signals were subtly broadcasted to drastically alter a vehicle's perceived coordinates, forcing it to deviate from its intended path or enter hazardous zones. The injection of false sensory data or application of adversarial perturbations to deceive perception systems into making incorrect decisions, e.g., emergency braking or failing to stop at a traffic sign, was presented in [30]. In addition, unauthorized steering, acceleration, or total remote takeover via intercepting and maliciously modifying the control instructions sent to a vehicle was showcased in [31,32]. Also, interference, or network flooding, was used to block critical navigation-related communications, leading to connectivity loss or remote disabling of an entire vehicle fleet [33]. Hence, developing cybersecurity solutions to mitigate attacks targeting autonomous UGV navigation is of utmost importance. These solutions must not be invasive to existing standards, while at the same time be compatible with legacy platforms. They must also introduce minimum to no modifications to the readily available hardware (e.g., antennas, cameras, radars).

Recent UGV security approaches exploited proportional-integral observers and residual analysis to identify malicious data injections [34–37]. While these approaches provide low latency, they are frequently constrained by their high sensitivity to environmental noise and abrupt changes in kinematics. To mitigate these constraints, hardware and signal integrity approaches were developed to leverage visual-inertial odometry as a trust anchor or to monitor signal characteristics, e.g., Doppler shifts and angle of arrival [38–41]. However, they often suffer from cumulative sensor drift during extended missions and impose significant computational overhead. Other approaches introduced architectures for systematically reviewing simulated attacks or historical attack records to establish defense guidelines [42–45]. However, because these approaches primarily depend on attack records, they often struggle to adapt to sophisticated, evolving navigation threats. This has led to data-driven approaches, featuring network-based intrusion detection with DL to identify DoS and man-in-the-middle attacks [46–49]. Nevertheless, such approaches often overlook subtle navigational deviations. Temporal state anomaly detection approaches with sequential modeling, e.g., long short-term memory encoding and online learning, were explored to identify sensor tampering and location manipulation [50–53]. Yet, existing detection models struggle to distinguish between physical fluctuations and malicious injections in complex navigation scenarios, such as sharp turns or rapid speed changes. Furthermore, given the challenges of background noise within robotic sensors, maintaining the robustness in distinguishing between authentic navigation and anomalous state transitions remains a critical hurdle. The solution proposed herein presents an approach

to detect and classify malicious injection attacks, specifically designed for UGVs on fixed routes, e.g., public transportations.

Under normal operating conditions, a rigid physical consistency and mathematical coupling exist between wheel-derived odometry and the vehicle's inertial responses. When cyberattacks or anomalies inject erroneous values into the odometry, e.g., fake linear velocities or fabricated steering angles, the untampered inertial measurement unit (IMU) data continue to reflect the vehicle's authentic physical kinematics. By evaluating the instantaneous correlations between these two heterogeneous streams, i.e., odometry and IMU, ML can promptly and efficiently detect the instances where odometry data violate the established physical coupling boundaries. This targeted interlocking strategy enables the reliable isolation of odometry tampering without relying on complex temporal filtering, ensuring a high-precision and sub-millisecond navigation verification. The proposed solution differs from those reported in literature in the following aspects:

- i. Unlike multi-sensor fusion approaches that necessitate added hardware, e.g., antenna arrays or external infrastructure [10,11], the proposed solution leverages existing onboard telemetry to provide reliable detection of malicious injections.
- ii. The proposed solution maintains protocol integrity and compatibility with legacy platforms, allowing for convenient integration with established robotic navigation stacks. In contrast, cooperative approaches or those comprising architectural overhauls require continuous network-wide database synchronization or a fundamental redesign of their core communication logic, both of which pose prohibitive implementation barriers for standalone or legacy systems [17,42].
- iii. The proposed solution is independent of weather conditions as it uses kinematic telemetry data. Hence, it maintains consistent anomaly detection performance as compared to perception-based approaches with visual or LiDAR data, which impacts their precision in degraded conditions, e.g., fog, rain, or low-light [11,13].
- iv. Compared to AI-based temporal and vision-centric approaches, the proposed solution offers a high computational efficiency and low overhead, ensuring sub-millisecond real-time prediction without high-end computing resources [23,25].
- v. The proposed solution utilizes a systematic and lightweight data acquisition and control procedure to enable precise logging of internal sensor states during autonomous navigation. It also facilitates controlled injections of malicious cyberattacks, ensuring rigorous experimentations and reproducibility. On the other hand, other approaches were restricted by third-party datasets [46,48].
- vi. Lastly, all resources developed in this work (i.e., training and validation datasets, classification models) are made publicly accessible to serve as a standardized benchmark for future UGV navigation security research [54].

This article is organized as follows: Section 2 describes the setup for extracting telemetry feature samples (with and without cyberattacks) and creating the training dataset. Section 3 details the development and comparative analysis of multiple AI-based anomaly detection and classification models. Section 4 elaborates the experiments for validating the optimum model in real-time. Finally, conclusions and future work are offered in Section 5.

2. Setup for Extracting Feature Samples and Creating Dataset

In the feature extraction setup elaborated herein, a four-wheeled robot from Clearpath Robotics is exploited, which is equipped with a six-degrees-of-freedom IMU for linear and angular motion sensing, along with wheel encoders for obtaining odometry data [55,56]. Four distinct routes are configured with the high-fidelity Gazebo simulation tool [57]. It enables visual simulations and incorporates complex multi-modal noise characteristics,

e.g., additive white Gaussian noise, into IMU data. It also introduces quantization errors to the encoders, creating a realistic setup for extracting features.

2.1. Preliminaries for Creating Autonomous Navigation

A controller algorithm for mission-driven autonomous navigation, shown in Figure 1, is used within the Robot Operating System 2 (ROS2) platform. It is built as a sequential state machine to execute heterogeneous navigational primitives, including linear displacement, reactive obstacle avoidance, and precise heading transitions. It operates at a 50-Hz sampling rate, i.e., 20-ms per sample, and data acquisition via “Idle”, “Running”, and “Done” phases. During initialization in the *Idle* phase, the controller establishes a communication node with the UGV via a namespace-based subscription mechanism to capture filtered odometry, raw IMU data, and LiDAR scans. To maintain data stability, a three-second buffer, i.e., 150 samples, is enforced before the mission begins, followed by ½-second intervals i.e., 25 samples, between navigation tasks to allow Gazebo and sensor biases to reach steady state and prevent data spikes. Upon transitioning to the *Running* phase, the controller retrieves navigation primitives from a predefined queue and executes the logic for each task. Once all tasks are finalized, the controller enters the *Done* phase to terminate the mission by canceling the control timer, closing the data repository, and shutting down the node.

$\mathcal{M}$: Task sequence defined as $\mathcal{M} = \{task_0, task_1, \dots, task_{n-1}\}$
 f_s : Control frequency (50 Hz), where $\Delta t = 1/f_s$
 N_{init} : Initial stabilization buffer
 N_{gap} : Inter-task transition buffer
 State: Controller state machine $\in \{Idle, Running, Done\}$

```

1: Procedure Path_Control_Execution()
2:   Initialize ROS2 node and establish CSV telemetry stream
3:   subscribe to sensor topics {Odom, LiDAR, IMU}
4:   for each timer interrupt at interval  $\Delta t$  do
5:     if no odom_received then continue           # wait for sensor initialization
6:     if state = Done then
7:        $v_{cmd} \leftarrow [0, 0]$ 
8:     else if state = Idle then
9:        $v_{cmd} \leftarrow [0, 0]$ 
10:      if taskIdx = 0 then  $N_{wait} \leftarrow N_{init}$  else  $N_{wait} \leftarrow N_{gap}$ 
11:      end if
12:      if waitCount <  $N_{wait}$  then waitCount  $\leftarrow$  waitCount + 1
13:      else state  $\leftarrow$  Running, waitCount  $\leftarrow$  0,
14:        execute Reset_Reference_Coordinates( )
15:      end if
16:    else if state = Running then
17:      ( $v, w, is\_finished$ )  $\leftarrow$  Execute_Mission_Logic( $\mathcal{M}[taskIdx]$ )
18:       $v_{cmd} \leftarrow [v, w]$ 
19:      if is_finished then
20:        if taskIdx <  $n - 1$  then state  $\leftarrow$  Idle, taskIdx  $\leftarrow$  taskIdx + 1
21:        else state  $\leftarrow$  Done
22:      end if
23:    end if
24:    Execute Publish_Velocity( $v_{cmd}$ )
25:    Save Telemetry_Log( $v_{cmd}, sensor\_payload, timestamp$ )
26:  end for
27: end Procedure

```

Figure 1. Pseudocode for the mission-driven controller algorithm.

A mission-driven autonomous navigation algorithm, shown in Figure 2, is executed during the *Running* phase, which retrieves navigational primitives from a predefined queue to fulfill mission requirements. It differentiates between three main tasks: “Obstacle”, “Dist”, and “Turn”. Linear navigation is managed with a dual-stage velocity scheduling strategy for *Dist*, i.e., fixed distance, and *Obstacle*, i.e., proximity-based, tasks. The UGV cruises at a constant velocity obtained from a predefined speed library. The controller monitors the

relative displacement and implements a proactive deceleration mechanism as the UGV approaches target, i.e., verified with LiDAR or odometrical displacement.

v_{speed} : Linear or angular velocity from Speed Library $\mathcal{L}$
 K_p, K_d : Proportional (1.2) and derivative (0.6) control gains
 $\epsilon_{dist}, \epsilon_{yaw}$: Convergence thresholds (0.005 rad for orientation)
 d_{decel} : Linear deceleration threshold (2-m)

```

1: Procedure Execute_Mission_Logic(task)
2:   Retrieve target_val and  $v_{speed}$  based on the specific task configuration
3:   if task.type  $\in \{Dist, Obstacle\}$  then
4:      $d_{rem} \leftarrow$  Calculate remaining distance to target pose or obstacle
5:     if  $d_{rem} \leq 0$  then return (0, 0, true)
6:     if  $d_{rem} \leq d_{decel}$  then  $v_{out} \leftarrow \max(0.1, d_{rem} \times 0.5)$ 
7:     else  $v_{out} \leftarrow v_{speed}$ 
8:     end if
9:     return ( $v_{out}$ , 0, false)
10:  else if task.type = Turn then
11:     $error_{yaw} \leftarrow \text{Normalize}(\text{target\_yaw} - \text{current\_yaw})$ 
12:    if  $|error_{yaw}| < \epsilon_{yaw}$  then return (0, 0, true)
13:     $derivative \leftarrow error_{yaw} - \text{pre\_error}$ 
14:     $\omega_{raw} \leftarrow (K_p \times error_{yaw}) + (K_d \times derivative)$ 
15:     $\omega_{out} \leftarrow \text{Clamp}(\omega_{raw}, \text{min} = 0.15, \text{max} = v_{speed})$ 
16:     $\text{pre\_error} \leftarrow error_{yaw}$ 
17:    return (0,  $\omega_{out}$ , false)
18:  end if
19: end Procedure
  
```

Figure 2. Pseudocode for the mission-driven autonomous navigation algorithm.

Since motors are susceptible to stalling at low speeds due to simulated resistance, a clamping mechanism is employed to maintain a minimum velocity until the target tolerance is achieved. As for *Turn* tasks, the controller switches to rotational logic using proportional-derivative strategy to align UGV heading. The proportional component generates the torque to drive the UGV toward target heading, while the derivative component provides active damping to suppress inertial oscillations. By normalizing angular errors within a $[-\pi, \pi]$ manifold and enforcing a stringent 0.005-radian tolerance, the controller achieves sub-degree rotational accuracy. The *Obstacle* task utilizes a centralized scanning window to filter LiDAR clouds, creating a safety buffer that triggers a controlled halt upon proximity detection. Throughout execution, the controller maintains a continuous feedback loop that validates task completion by evaluating the data transmitted from the subscribed sensor nodes, i.e., filtered odometry, IMU, LiDAR. Once the specific convergence criteria are met based on sensor feedback, the controller transitions back to a stabilization state before proceeding to the next mission segment.

2.2. Collection of Authentic Feature Samples

Authentic IMU and odometry samples are collected from Gazebo simulations, which are parameterized to replicate non-ideal sensor noise profiles. These features are detailed in Table 1 and include the timestamp (T_s), task type (T_t), command linear velocity in the x -axis ($Clvx$), command angular velocity in the z -axis ($Cavw$), odometry positions (Opx, Opy, Opz), heading angle, i.e., yaw (Θ), odometry orientations ($Oorx, Oory, Oorz, Oorw$), odometry linear velocity in x -axis ($Olvx$), odometry angular velocity in z -axis ($Oavw$), IMU linear velocities ($Ilvx, Ilvy, Ilvz$), IMU angular velocities ($Iavx, Iavy, Iavz$), and IMU orientations ($Iorx, Iory, Iorz, Iorw$). T_s denotes the cumulative simulation time for temporal sequencing. T_t indicates the designated driving sequence or mission profile. $Clvx$ and $Cavw$ are used to create authentic trajectories. Opx, Opy , and Opz denote the estimated Cartesian coordinates of the vehicle within the global reference frame. Θ is analytically derived from odometry data and is used to facilitate visual inspection. $Oorx, Oory, Oorz$, and $Oorw$ are estimated by the wheel encoders. $Olvx$ and $Oavw$ are also estimated from wheel encoders, which

represent velocity. l_{lvx} , l_{lvy} , and l_{lvz} are measured by the IMU accelerometer onboard in simulations. l_{avx} , l_{avy} , and l_{avz} are measured by the IMU gyroscope onboard in simulations. l_{orx} , l_{ory} , l_{orz} , and l_{orw} denote the vehicle's attitude synthesized via filtering accelerometer and gyroscope measurements.

Table 1. Extracted features for developing the classifiers.

Feature	Representation	Unit	Brief Description
time stamp	T_s	ns	Time elapsed since the start of the simulation
task type	T_t	-	Indicates the designated driving sequence within the tasks
command linear velocity in x-axis	$Clvx$	m/s	Linear velocity operator sends to UGV in x-axis
command linear velocity in z-axis	$Cavw$	rad/s	Angular velocity operator sends to UGV in z-axis
odometry position	Op_x, Op_y, Op_z	m	Position estimated in the simulation world
heading angle (yaw)	Θ	rad	Calculated yaw angle from the odometry quaternion
odometry orientation	$Oor_x, Oor_y, Oor_z, Oor_w$	-	Attitude represented as a normalized quaternion
odometry linear velocity in x-axis	$Olvx$	m/s	Estimated linear velocity derived from wheel odometry
odometry angular velocity in z-axis	$Oavw$	rad/s	Estimated angular velocity derived from wheel odometry
IMU linear velocities	$l_{lvx}, l_{lvy}, l_{lvz}$	m/s ²	Measured by the IMU three-axis accelerometer
IMU angular velocities	$l_{avx}, l_{avy}, l_{avz}$	rad/s	Measured by the IMU three-axis gyroscope
IMU orientations	$l_{orx}, l_{ory}, l_{orz}, l_{orw}$	-	Attitude of the vehicle relative to the inertial reference frame
label	-	-	Represent the type of the data
trajectory label	-	-	Represent the number of the trajectory

The label represents the type of data, i.e., authentic or malicious. The trajectory label represents trajectories 1 through 4. It is worthy to point out that Op_z , l_{lvz} , l_{avx} , l_{avy} , Oor_x , Oor_y , l_{orx} , and l_{ory} are not included in the dataset, as the simulations are in a 2-D plane. Each data point is tabulated and indexed by nanosecond timestamps to ensure absolute temporal alignment between control commands and the UGV response. This allows for the precise characterization of sensor biases and baseline noise, particularly during the *Idle* phase, which serves as a reference for authentic sensor behavior. To enhance the robustness of the dataset and prevent subsequent classifiers from overfitting, a two-tiered generalization is built. First, dynamic velocity profiles are integrated into the control logic to introduce variations in UGV velocity across different mission segments. These variations capture non-linearities in the sensor suite and mechanical latency under different kinetic loads. Second, the spatial diversity of the dataset is expanded by considering four distinct trajectories, i.e., separate routes. This ensures that the learning process captures motion patterns and potential attack signatures, e.g., anomalies between commanded and observed velocity, rather than memorizing coordinates. The resulting dataset maintains a high-frequency temporal consistency, providing a granular and representative foundation for motion analysis and security research.

2.3. Collection of Attack Data

Figure 3 illustrates a two-step scenario for launching malicious message injection cyberattacks. In step 1, the attacker interrupts communications between the operator and access point (AP) to deliberately disrupt their connection while simultaneously logging navigation commands prompted by the operator, as depicted in Figure 3a. This is achieved by injecting radio frequency interference to degrade the signal to interference plus noise ratio (SINR) and force continuous backoff, rendering the operator unable to control the UGV. This can also be achieved by flooding the local data distribution service (DDS) network, exhausting the communication queues of AP control commands or via domain name system

cache poisoning, interrupting the remote command stream during handshake. In step 2, the attacker maliciously modifies logged commands, e.g., velocity, prior to launching them to the target UGV, as demonstrated in Figure 3b. This two-step scenario assumes that the odometry stream transmitted over ROS/ROS2 topics can be tampered with physical- or network-level attacks, while the IMU stream remains authentic due to physical connection with the onboard microcontroller. If both odometry and IMU streams are compromised, the quality of attack detection will be degraded, leading to lower detection rates, higher misdetection rates, and/or higher false alarms. Hence, this single-stream assumption is a possible limitation of the proposed attack scenario, and that defense against multi-hardware stream tampering is beyond the scope of this work.

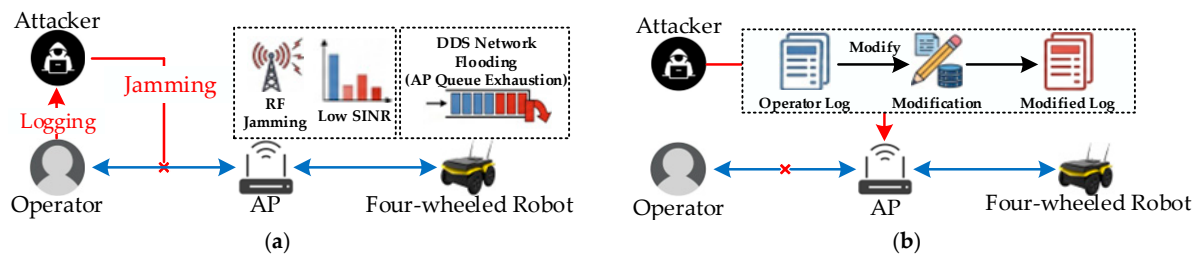


Figure 3. Two-step scenario for launching malicious injection cyberattacks: (a) step 1 and (b) step 2.

Two simulated types of malicious injection attacks are studied in this work: path modification (PM) and velocity drift (VD). PM is concerned with changing the UGV heading by calculating new coordinates selected by the attacker. This is performed recursively (new coordinate is based on previous calculated one) and is realized by altering the orientation and position features. On the other hand, VD is concerned with modifying UGV linear velocity and position features, which eventually causes inaccuracies in the estimated position. Such attacks are generated by tampering sensory data, i.e., wheel odometry, without inducing jamming at the physical layer due to safety and liability concerns.

Both attacks are distributed across trajectories 1–4 and adjusted to provide sufficient data to train the detection and classification models. Figure 4 depicts a pseudocode for generating PM and VD, whereas Table 2 summarizes the distribution of authentic, PM, and VD samples. For each trajectory, the number of authentic samples is around 10,000. The dataset maintains a consistent 2:1:1 ratio for authentic, PM, and VD samples, respectively, i.e., 50% authentic and 50% malicious. This ratio is selected to mitigate class imbalances that often leads to overtraining or undertraining, ensuring classifiers will establish well-defined decision boundaries with no bias towards a specific class. Rare attack instances in practical deployments have no impact on classification performance as the use of balanced datasets in training allows the development of classifiers with optimal decision boundaries for all classes, resulting in high detection rates and low false alarms as will be discussed in detail in Section 3.2. It is paramount to point out that because sequential operation is assumed, PM and VD are independent and do not occur simultaneously.

Table 2. Distribution of authentic and attack samples.

	Authentic Samples	PM Attack Samples	VD Attack Samples	Total Samples/Trajectory
Trajectory 1	9400	4700	4700	18,800
Trajectory 2	11,000	5500	5500	22,000
Trajectory 3	12,600	6300	6300	25,200
Trajectory 4	11,000	5500	5500	22,000
Total samples/type	44,000	22,000	22,000	88,000

$\mathcal{D}$: Authentic ROS2 telemetry dataset read from “auth_data.csv”
 B_{total} : Total attack budget defined as $B_{total} = |\mathcal{D}| \times 0.25$
 $\mathcal{T}$: Target task types defined as $\mathcal{T} = \{Obstacle, Dist\}$
 Θ_{drift} : Minimum PM angle drift bounds ($^{\circ}$), $\Theta_{drift} = [10, 25]$
 λ : PM drift intensity factor $\lambda = 10000$
 V_{scale} : VD velocity scale range $V_{scale} = [0.6, 1.4]$

```

1: Procedure Attack_Generation_UGV()
2:    $B \leftarrow identify\_blocks(\mathcal{D}, \mathcal{T})$ 
3:    $A \leftarrow allocate\_rows(B, B_{total})$ 
4:   for each  $b\_id, G$  in  $B$  do
5:     writerrow(“output.csv”,  $G$ , label  $\leftarrow 0$ )
6:     if  $b\_id \in A$  then
7:        $N\_target \leftarrow A[b\_id]$ 
8:       # --- Generate PM Attack (Label 1) ---
9:        $start \leftarrow rand(buf, |G| - N\_target - buf)$ 
10:       $f\_flux \leftarrow U(0.8, 1.2)$ 
11:       $\Delta\theta \leftarrow (\lambda/N\_target) \times f\_flux$ 
12:       $\Delta\theta \leftarrow clip(\Delta\theta, \Theta_{drift}[0], \Theta_{drift}[1]) \times rand\_sign()$ 
13:       $\delta \leftarrow linspace(0, rad(\Delta\theta), N\_target)$ 
14:      for each  $i$  from 0 to  $N\_target - 1$  do
15:         $m\_atk[“yaw”] \leftarrow (m\_auth[“yaw”] + \delta[i] + \pi) \bmod 2\pi - \pi$ 
16:         $m\_atk[“ori\_z”] \leftarrow sin(m\_atk[“yaw”]/2)$ 
17:         $m\_atk[“ori\_w”] \leftarrow cos(m\_atk[“yaw”]/2)$ 
18:         $m\_atk[“ori\_x”, “ori\_y”] \leftarrow 0.0$ 
19:         $dist \leftarrow calc\_dist(m\_auth[i], m\_auth[i + 1])$ 
20:         $m\_atk[“x”, “y”] \leftarrow m\_prev[“x”, “y”] + dist \times [cos(m\_atk[“yaw”]),$ 
21:           $sin(m\_atk[“yaw”])]$ 
22:        writerrow(“output.csv”,  $m\_atk$ , label  $\leftarrow 1$ )
23:      end for
24:      # --- Generate VD Attack (Label 2) ---
25:       $start \leftarrow rand(buf, |G| - N\_target - buf)$ 
26:       $s \leftarrow U(V\_scale[0], V\_scale[1])$ 
27:      if  $0.95 < s < 1.05$  then  $s \leftarrow s + 0.1$ 
28:      for each  $i$  from 0 to  $N\_target - 1$  do
29:         $m\_atk[“odom\_lin\_x\_v”] \leftarrow m\_auth[“odom\_lin\_x\_v”] \times s$ 
30:         $distMod \leftarrow calc\_dist(m\_auth[i], m\_auth[i + 1]) \times s$ 
31:         $m\_atk[“x”, “y”] \leftarrow m\_prev[“x”, “y”] + distMod \times$ 
32:           $[cos(m\_auth[“yaw”]), sin(m\_auth[“yaw”])]$ 
33:        writerrow(“output.csv”,  $m\_atk$ , label  $\leftarrow 2$ )
34:      end for
35:    end if
36:  end for
37: end Procedure

```

Figure 4. Pseudocode to generate attack samples for PM and VD.

Also, compound attacks induce simultaneous spatial and kinematic outliers. Because compound tampering distorts cross-sensor telemetry, i.e., IMU and odometry, in multiple features at once, the feature vector drifts significantly, immediately triggering a fail-safe strategy. Table 3 elaborates trajectories 1–4, all of which have the same start and destination positions. Travel distances, velocities, and task sequences are varied to contribute to the diversity and inclusivity of the dataset, ensuring a comprehensive evaluation. Table 4 shows the selected range of values for the features in the dataset that lead to PM and VD cyberattacks. Moreover, Figure 5 shows the impact of PM attacks on trajectories 1–4. It is worth noting that demonstration videos for PM and VD can be found in [54].

Table 3. Executed tasks (all authentic) for trajectories 1–4.

Trajectory 1			Trajectory 2			Trajectory 3			Trajectory 4		
Task Type	Distance (m)	Velocity (m/s) *	Task Type	Distance (m)	Velocity (m/s) *	Task Type	Distance (m)	Velocity (m/s) *	Task Type	Distance (m)	Velocity (m/s) *
Obstacle	1.08	0.6	Turn	-	0.7	Turn	-	0.7	Turn	-	0.7
Turn	-	0.7	Obstacle	26.61	3.0	Obstacle	0.84	0.6	Dist	10.00	5.0
Dist	3.50	0.7	Turn	-	0.6	Turn	-	0.7	Turn	-	0.8
Turn	-	0.6	Obstacle	27.01	2.0	Obstacle	0.85	0.7	Dist	30.00	5.0
Obstacle	25.31	4.0	Turn	-	0.8	Turn	-	0.7	Turn	-	0.8
Turn	-	0.8	Dist	5.00	1.2	Obstacle	26.95	4.0	Dist	12.00	4.0
Dist	3.00	0.8	Turn	-	1.5	Turn	-	1.5	Turn	-	1.5
Turn	-	1.5	Obstacle	12.44	0.8	Obstacle	27.71	0.9	Dist	6.00	0.9
Dist	3.00	0.9	Turn	-	0.7	Turn	-	0.7	Turn	-	1.8
Turn	-	0.7	Obstacle	22.68	0.9	Dist	10.00	0.8	Dist	18.00	4.5
Obstacle	7.63	1.5	Turn	-	1.8	Turn	-	1.8	Turn	-	0.7
Turn	-	1.8	Obstacle	7.50	2.5	Obstacle	2.42	2.5	Dist	8.00	0.8
Obstacle	17.46	2.5	Turn	-	1.1	Turn	-	1.1	Turn	-	1.2
Turn	-	1.1	Obstacle	19.75	3.0	Obstacle	17.51	3.0	Dist	11.00	2.5
Obstacle	19.60	3.0	Turn	-	1.2	Turn	-	1.2	Turn	-	1.1
Turn	-	1.2	Obstacle	3.99	2.0	Obstacle	17.29	2.0	Dist	4.00	3.0
Obstacle	3.78	2.0	Turn	-	1.3	Turn	-	1.3	Turn	-	1.3
Turn	-	1.3	Obstacle	3.54	1.0	Obstacle	19.60	1.0	Dist	3.00	2.0
Obstacle	3.58	1.0	Turn	-	0.8	Turn	-	0.8	Turn	-	0.9
Turn	-	0.8	Obstacle	4.66	0.4	Obstacle	3.46	0.4	Dist	4.00	0.4
Obstacle	4.71	0.4	Turn	-	0.9	Turn	-	0.9	Turn	-	1.4
Turn	-	0.9	Obstacle	5.30	1.1	Obstacle	3.14	1.1	Dist	3.00	1.1
Obstacle	5.23	1.1	Turn	-	1.4	Turn	-	0.8	Turn	-	1.1
Turn	-	1.4	Dist	5.00	3.5	Obstacle	4.77	0.4	Obstacle	4.15	0.4
Dist	5.00	3.5	-	-	-	Turn	-	0.9	Turn	-	1.3
-	-	-	-	-	-	Obstacle	5.25	1.1	Obstacle	6.15	1.1
-	-	-	-	-	-	Turn	-	1.4	Turn	-	1.4
						Dist	5.00	3.5	Dist	5.00	3.5

* Velocity for the *Turn* tasks is in (rad/s). Different colors are used to represent the tasks for each of Trajectories 1–4.**Table 4.** Range of feature values for generating PM and VD attacks.

Feature	Description	Value	Unit
Θ : Heading angle, i.e., yaw. Obtained by altering orientation & position	Maximum cumulative drift angle injected into Θ	10 (min) 25 (max)	Degree
<i>Olvx</i> : Odometry linear velocity in <i>x</i> -axis. Obtained by altering velocity & position	Scaling factor or applied to <i>Olvx</i>	0.6 (min) 1.4 (max)	-

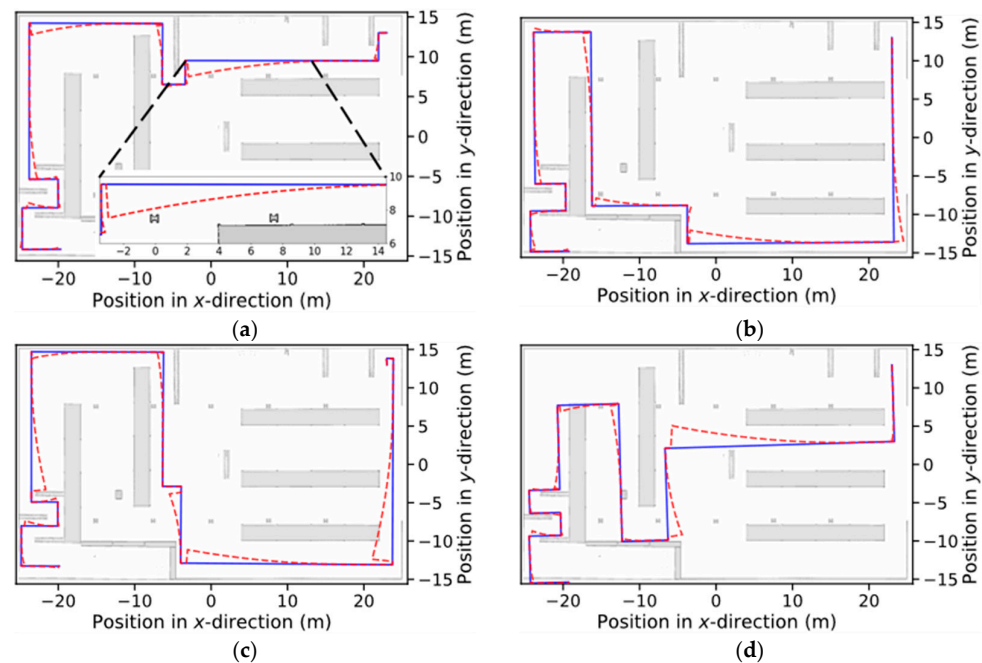


Figure 5. Designed trajectories with and without the presence of PM: (a) trajectory 1, (b) trajectory 2, (c) trajectory 3, and (d) trajectory 4. Solid traces (in blue) represent authentic trajectories, whereas dashed traces (in red) represent counterpart attacked trajectories.

3. Development and Evaluation of Detection and Classification Models

This section details the development of the ML models for detecting and classifying PM and VD cyberattacks. It also benchmarks these models against each other to evaluate their performance. Here, five models are developed using the following classifiers: decision tree (DT), k-nearest neighbors (KNN), multi-layer perceptron (MLP), random forest (RF), and support vector machine (SVM). By exploiting the instantaneous variation between the IMU and odometry data, such models are trained to detect the presence of an attack and classify its type. Subsequent subsections highlight feature analysis and hyperparameter configurations, followed by comprehensive performance analysis addressing classification accuracy, robustness against background noise, and prediction latency.

3.1. Feature Correlation and Importance Analysis

The correlation heatmap of the extracted features, computed using the Spearman correlation algorithm, is illustrated in Figure 6a. It allows for the identification of correlated features to discard those with the least impact on training models, thereby reducing dimensionality that could otherwise affect the training quality and time. A correlation factor $|c| \geq 0.8$ is considered in this work, resulting in the identification of one feature pair as strongly correlated: (*Iorw*, *Oorw*). Therefore, a feature importance analysis, shown in Figure 6b, is carried out with the mean decrease in impurity algorithm. After the relative importance scores within the aforementioned correlated pair are compared, *Iorw* is discarded from the dataset. As a result, the final dataset that is used for training and validating the ML classifiers contains the following features: *Opx*, *Opy*, *Oorz*, *Oorw*, *Olvx*, *Oavw*, *Ilvx*, *Ilvy*, *Iavz*, *Iorz*, and, *trajectory_label* for split; and *label* for target. Both feature correlation and feature importance analyses contribute to (i) the prevention of tree-based classifiers, e.g., RF, from potentially making random or unstable node splits because of correlated features, and (ii) the reduction of feature dimensionality without losing essential kinematic information. This leads to faster training convergence, less memory consumption, and subsequently, more efficient classifiers.

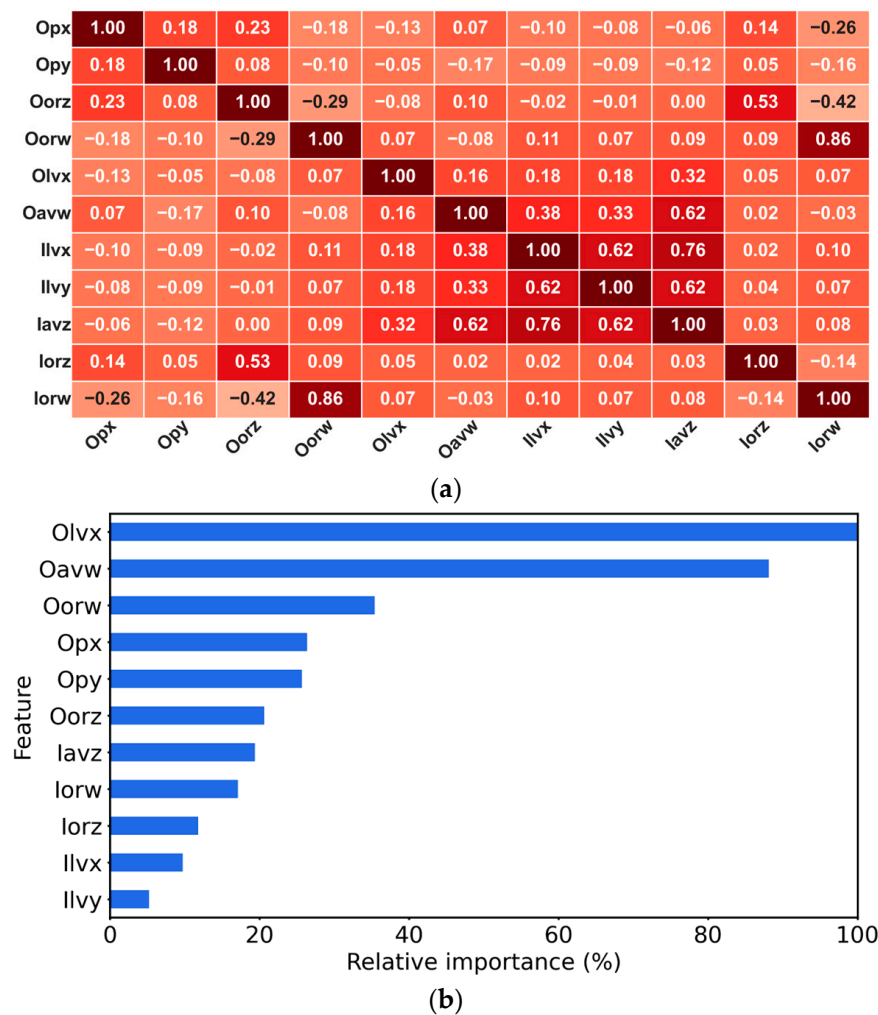


Figure 6. Feature analysis showcasing (a) correlation and (b) importance.

3.2. ML Training and Validation

Upon finalizing the dataset, the ML classifiers are trained to detect and classify PM and VD within a unified framework. Each classifier is optimized using the random search algorithm, which allows for the systematic evaluation of combinations of hyperparameters from a predefined range. During the training phase, each hyperparameter combination is evaluated on 70% of the dataset using 10-fold cross-validation to ensure model robustness and prevent overfitting. Then, the model with the optimum hyperparameter combination is tested on the remaining 30% of the dataset.

Although the samples in the finalized dataset are continuous in time, the introduced fluctuations, sensor noises, and data exchange rates during simulations impose fewer temporal dependencies between such samples. Hence, the random 70–30 split provides a valid feature-space baseline without severely compromising classifier optimization. To assess the impact of temporal correlation on classifier development, real-time validations are performed with a physical ground vehicle platform across routes featuring no temporal, spatial, or otherwise sample-derivation overlap with the finalized dataset, ensuring the solution's true generalization and scalability. Details on such an assessment are presented in Section 4. Table 5 presents the optimized hyperparameters of the models.

Table 5. Optimized hyperparameters for each model.

Classifier	Hyperparameters	Optimized Value
RF	Bootstrap	False
	Criterion	Gini
	Max depth	25
	Min. samples leaf	1
	Min. samples split	10
	No. of estimators	287
	Max. features	Sqrt
KNN	Algorithm	Auto
	Leaf size	30
	Metric	Euclidean
	No. of neighbors	3
	Power	2
	Weight	Distance
MLP	Activation	Relu
	Alpha	0.0002910635913
	Hidden layers	(128, 64)
	Max. iter	173
	Solver	Adam
	Momentum	0.953595
	Initial learning rate	0.0030049873592
	Learning_rate	Constant
	Early stopping	True
DT	Criterion	Gini
	Max. depth	88
	Min. samples leaf	2
	Min. samples split	2
	Max. features	Null
	Splitter	Best
SVM	Svm_C	82.78543388872
	Kernel	rbf
	Shrinking	True
	Probability	False
	Tolerance	1.00×10^{-3}
	Max. iter	Null

The performance of such models is assessed using the validation accuracy (VA), precision (PR), F-score (FS), detection rate (DR), misdetection rate (MDR), false alarm rate (FAR), as well as training time (TT) and prediction time (PT). These are computed as follows:

$$\text{Validation Accuracy (VA)} = \frac{TP + TN}{TP + FP + TN + FN} \quad (1)$$

$$\text{Precision (PR)} = \frac{TP}{TP + FP} \quad (2)$$

$$F - score (FS) = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (3)$$

$$Detection Rate (DR) = \frac{TP}{TP + FN} \quad (4)$$

$$Misdetecation Rate (MDR) = \frac{FN}{TP + FN} \quad (5)$$

$$False Alarm Rate (FAR) = \frac{FP}{FP + TN} \quad (6)$$

where TP , TN , FP , and FN are the true positive, true negative, false positive, and false negative predictions, respectively. PR measures the exactness of detected anomalies, while FS is for evaluating dataset imbalance. DR and MDR evaluate the class-specific detection accuracies and misclassification rates, respectively, whereas FAR indicates the frequency at which authentic labels are falsely flagged as attacks. Table 6 depicts a qualitative analysis of the aforementioned evaluation metrics, along with the computational efficiency measured by the TT and PT . A Windows 11 PC with an Intel i9-13980HX CPU @ 2.20 GHz and 16 GB of DDR5 4800 MHz SODIMM memory is used for training and validation. MLP exhibits the lowest VA in comparison to other models and is associated with the lowest class-specific DR , MDR , and FAR . On the other hand, RF is found to outperform other classifiers, with VA and class-specific DR greater than 98% together with class-specific MDR and FAR less than 2%. Also, its PT is 156.32 ms, obtained via testing 30% of the dataset (i.e., 26,400 samples). This leads to a PT of 5.9 μ s per sample, enabling real-time detection. Figure 7 shows the confusion matrices of the five developed classifiers. Resulting RF and DT matrices showcased in Figure 7a,b, respectively, demonstrate minimum misclassification across all classes, i.e., authentic, PM, VD, suggesting the ability of tree-structured models to identify discrete anomaly signatures. Conversely, Figure 7c depicts the performance of the KNN classifier, which exhibits more misclassifications. For example, it misclassifies 219 PM samples and 455 VD samples as authentic, while falsely flags 185 authentic samples as PM. Lastly, Figure 7d,e show that MLP and SVM have the highest misclassifications and false alarms.

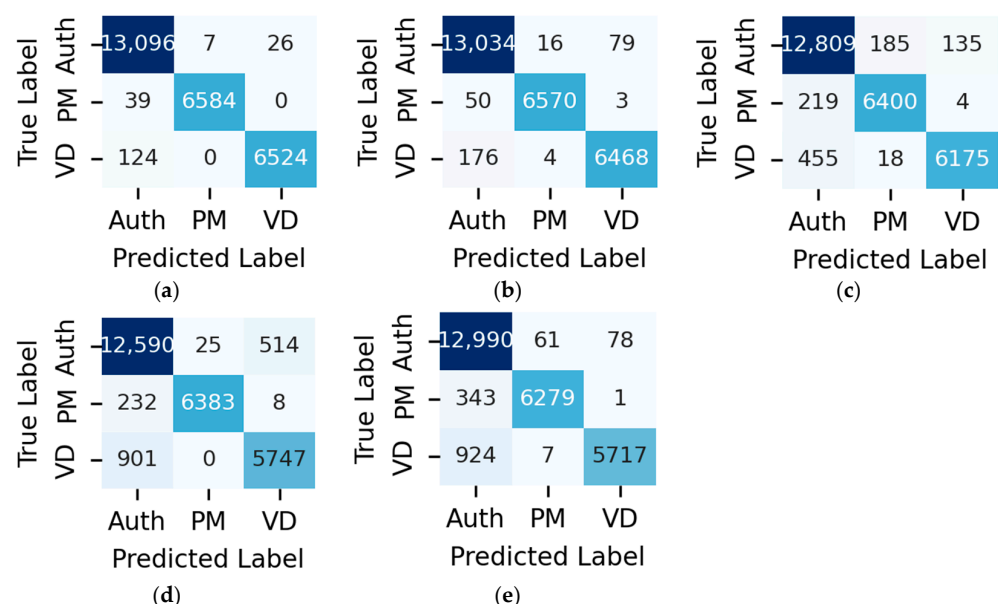


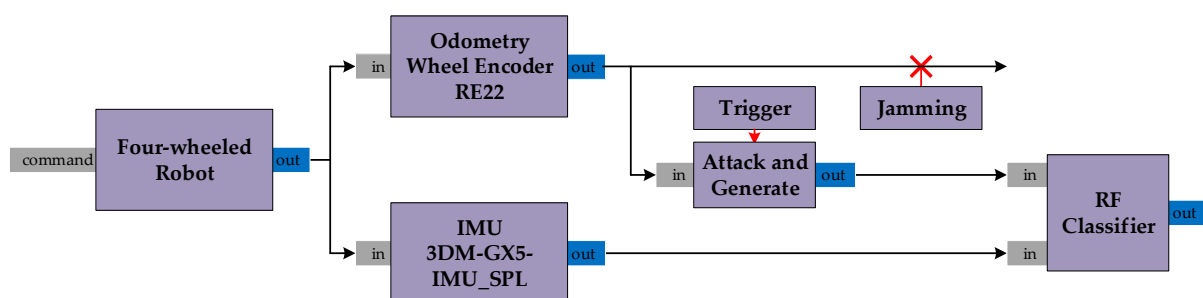
Figure 7. Confusion matrices for the developed classifiers: (a) RF, (b) DT, (c) KNN, (d) MLP, (e) SVM.

Table 6. Evaluation metrics of the developed classifiers.

	Avg. Performance Metrics				Class-Specific DR (%)			Class-Specific MDR (%)			Class-Specific FAR (%)			Time (ms)	
	VA	PR	DR	FS	Auth	PM	VD	Auth	PM	VD	Auth	PM	VD	TT	PT
RF	99.26	99.42	99.1	99.26	99.75	99.41	98.13	0.25	0.59	1.87	1.23	0.04	0.13	2618.32	156.32
KNN	96.15	96.58	95.69	96.13	97.56	96.63	92.89	2.44	3.37	7.11	5.08	1.03	0.7	125.81	414.56
MLP	93.64	94.34	92.91	93.62	95.89	96.38	86.45	4.11	3.62	13.55	8.54	0.13	2.64	171,129.5	40.78
DT	98.76	98.91	98.59	98.75	99.28	99.2	97.29	0.72	0.8	2.71	1.7	0.1	0.42	495.31	2.59
SVM	94.64	96.23	93.25	94.71	98.94	94.81	86	1.06	5.19	14	9.55	0.34	0.4	187,217.9	32,593.01

4. Real-Time Experimentations

To validate operation in real-time, the trained RF classification model is integrated into an experimental setup featuring the four-wheeled platform from Clearpath Robotics (Kitchener, ON, Canada), i.e., Jackal. A simplified flowchart illustrating this setup is shown in Figure 8. The serialized model file occupies 40.6 MB on disk. When loaded into the active ROS2 Python node, the runtime RAM consumption—including Python version 3.12.7 interpreter dependencies and feature processing buffers—remains at 100 MB. The embedded platform computing unit consists of a microcontroller with 16 GB RAM. Therefore, the model requires less than 1% of the system’s memory, leaving ample resources for other navigation stacks. Also, it executes natively on CPU without requiring GPU acceleration frameworks, making it cost-effective and highly reliable for real-time autonomous ground navigation cybersecurity. A malicious odometry node, i.e., Attack and Generate, is created to mimic a man-in-the-middle attack. It uses the PM and VD cyberattack generator illustrated in Figure 4 with a switch, i.e., trigger, to control the launch of the attack and its type. In this setup, authentic samples are collected by the odometry and IMU sensors presented in Figure 9. The malicious node obtains odometry samples and tampers with the data stream to create either PM or VD attacks. Lastly, both IMU samples and tampered odometry samples are fed to the RF classifier for real-time classification. Experimentations are executed considering routes that differ from trajectories 1–4, i.e., turns and velocity variations, and incorporated diverse terrain vibrations as well as noise and attack magnitudes. Table 7 summarizes live classifications of incoming authentic, PM, and VD messages. Since each attack is triggered as standalone, experimental results are presented across three separate runs.

**Figure 8.** A simplified flowchart of this experimental setup.

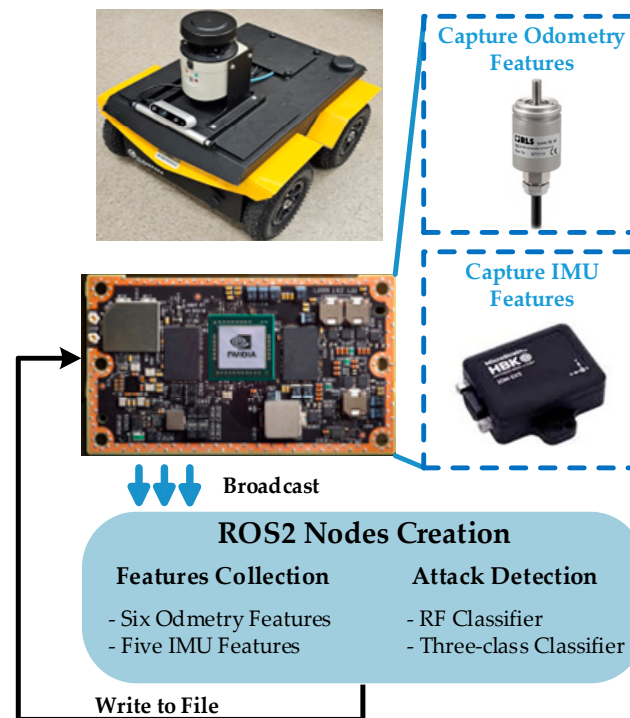


Figure 9. Hardware-software configuration for classifying odometry and IMU features.

Table 7. Live classification experiment.

Total Messages	Messages Type	Accuracy
2421	Authentic	98.48%
4202	Authentic injected with PM	97.12%
3466	Authentic injected with VD	94.86%

“Total Messages” are all incoming messages during testing, and “Accuracy” is computed using (1) to present the percent of samples decided upon in favor of Class 0, i.e., authentic, Class 1, i.e., PM, and Class 2, i.e., VD out of the total messages received. Authentic, PM, and VD messages are predicted with an accuracy of 98.47%, 97.12%, and 94.86%, respectively, averaging an accuracy of 96.82%. The if-else conditional branch evaluation logic of the RF classification model consists of lightweight, i.e., if-else logic, rather than complex matrix operations. This leads to a timely inference within the 50 Hz frame budget per sample and effortless execution with no control lag. Thus, the solution herein enables accurate and real-time classification, especially for edge computing platforms.

5. Conclusions

In this work, a ML solution was proposed for detecting and classifying injection cyberattacks with applications for UGVs. A mission-driven autonomous navigation algorithm was built to collect authentic odometry and IMU samples from four different simulated trajectories. A cyberattack generator was also built to obtain injected PM and VD samples to establish a dataset that contains authentic and malicious samples. The resulting dataset was used to train different ML classification models, and evaluations showed that RF exhibited optimum classification performance with *VA* and class-specific *DR* greater than 98% together with class-specific *MDR* and *FAR* less than 2%. The proposed solution was tested experimentally via a physical UGV platform and demonstrated high accuracy with a sub-millisecond *PT*, facilitating real-time classification. This solution allowed for the accurate detection of two types of injection attacks with no added hardware or physical

infrastructure. It also offered a lightweight setup with no modifications to the existing experimental platform. Future work entails the exploration of this solution in detecting and classifying other types of injection cyberattacks, e.g., ghost injection. Future work also entails the investigation of the inclusion of sample features from other sensors, e.g., LiDAR and camera, to improve detection and classification accuracies. Moreover, future work entails stress-testing the proposed solution across diverse velocity bounds, elevated noise regimes, and multi-tier attack intensities. Lastly, future work entails studying countermeasure strategies, e.g., reinforcement learning, to sideline the impact of cyberattacks targeting UGVs by maintaining safe and trustworthy autonomous navigation in cyber-contested environments.

Author Contributions: Conceptualization, K.A.S.; Methodology, K.A.S.; Software, T.W. and M.A.; Validation, T.W. and M.A.; Formal analysis, T.W.; Investigation, T.W.; Data curation, T.W.; Writing—original draft, T.W.; Writing—review & editing, K.A.S. and M.A.; Visualization, T.W.; Supervision, K.A.S.; Project administration, K.A.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are openly available in [GitHub] at reference number [54].

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

References

1. Zhou, N.; Zhang, G.; Zhu, C.; Dong, X. An unstructured roadless environment navigation map construction method based on remote sensing. *Geo-Spat. Inf. Sci.* **2025**, 1–22. [CrossRef]
2. Liu, Q.; You, X.; Zhang, X.; Zuo, J.; Li, J. Dynamic path planning of autonomous mobile robot in off-road environments using experience replay enhanced distributed proximal policy optimization algorithm. *Geo-Spat. Inf. Sci.* **2023**, 1–17. [CrossRef]
3. Cristóvão, M.P.; Portugal, D.; Carvalho, A.E.; Ferreira, J.F. A LiDAR-Camera-Inertial-GNSS Apparatus for 3D Multimodal Dataset Collection in Woodland Scenarios. *Sensors* **2023**, 23, 6676. [CrossRef] [PubMed]
4. Zhang, S.; Zeng, Q. Online Unmanned Ground Vehicle Path Planning Based on Multi-Attribute Intelligent Reinforcement Learning for Mine Search and Rescue. *Appl. Sci.* **2024**, 14, 9127. [CrossRef]
5. Boyanov, Y.; Petrov, O.; Georgieva, T. A review of ground-based robotic systems for search and rescue. In Proceedings of the 2025 34th Annual Conference of the European Association for Education in Electrical and Information Engineering (EAEIE), Cluj-Napoca, Romania, 18–20 June 2025; pp. 1–7.
6. Ersü, C.; Petlenkov, E.; Janson, K. A Systematic Review of Cutting-Edge Radar Technologies: Applications for Unmanned Ground Vehicles (UGVs). *Sensors* **2024**, 24, 7807. [CrossRef] [PubMed]
7. Trybała, P.; Szrek, J.; Remondino, F.; Kujawa, P.; Wodecki, J.; Blachowski, J.; Zimroz, R. MIN3D Dataset: Multi-sensor 3D Mapping with an Unmanned Ground Vehicle. *PFG–J. Photogramm. Remote Sens. Geoinf. Sci.* **2023**, 91, 425–442. [CrossRef]
8. El Bou, C.M.; Focchi, M.; Chang, M.R.; Camurri, M.; von Ellenrieder, K.D. Smooth Human–Robot Shared Control for Autonomous Orchard Monitoring With UGVs. *IEEE Trans. Autom. Sci. Eng.* **2025**, 22, 13603–13620. [CrossRef]
9. Autonomous Cars Market Size, Share and Industry Analysis, by Type, by Vehicle Type, and Regional Forecast, 2026–2034. Available online: <https://www.fortunebusinessinsights.com/industry-reports/autonomous-cars-market-100141> (accessed on 1 June 2026).
10. Kolar, P.; Benavidez, P.; Jamshidi, M. Survey of Datafusion Techniques for Laser and Vision Based Sensor Integration for Autonomous Navigation. *Sensors* **2020**, 20, 2180. [CrossRef] [PubMed]
11. Yeong, D.J.; Velasco-Hernandez, G.; Barry, J.; Walsh, J. Sensor and Sensor Fusion Technology in Autonomous Vehicles: A Review. *Sensors* **2021**, 21, 2140. [CrossRef] [PubMed]
12. Fayyad, J.; Jaradat, M.A.; Gruyer, D.; Najjaran, H. Deep Learning Sensor Fusion for Autonomous Vehicle Perception and Localization: A Review. *Sensors* **2020**, 20, 4220. [CrossRef] [PubMed]

13. Ilci, V.; Toth, C. High Definition 3D Map Creation Using GNSS/IMU/LiDAR Sensor Integration to Support Autonomous Vehicle Navigation. *Sensors* **2020**, *20*, 899. [[CrossRef](#)] [[PubMed](#)]
14. Ngo, H.; Fang, H.; Wang, H. Cooperative Perception With V2V Communication for Autonomous Vehicles. *IEEE Trans. Veh. Technol.* **2023**, *72*, 11122–11131. [[CrossRef](#)]
15. Yang, H.; Hong, J.; Wei, L.; Gong, X.; Xu, X. Collaborative Accurate Vehicle Positioning Based on Global Navigation Satellite System and Vehicle Network Communication. *Electronics* **2022**, *11*, 3247. [[CrossRef](#)]
16. Wang, Z.; Spasojevic, P.; Schlake, B.W.; Mulay, N.; Zaman, A.F.; Liu, X. Development and Testing of a UWB-Based Vehicle-to-Vehicle (V2V) Ranging System for Self-Propelled Rail Vehicles. *IEEE Trans. Veh. Technol.* **2024**, *73*, 3247–3261. [[CrossRef](#)]
17. Zhu, X.; Li, Z.; Jiang, Y.; Xu, J.; Wang, J.; Bai, X. Real-Time Vehicle-to-Vehicle Communication-Based Network Cooperative Control System Through Distributed Database and Multimodal Perception: Demonstrated in Crossroads. In *Proceedings of Ninth International Congress on Information and Communication Technology (ICICT 2024)*; Lecture Notes in Networks and Systems; Yang, X.-S., Sherratt, R.S., Dey, N., Joshi, A., Eds.; Springer: Singapore, 2024; Volume 1055, pp. 143–154.
18. Tamang, M.T.; Maheriya, D.; Sharif, M.S.; Sutharssan, T. Autonomous Navigation for TurtleBot3 Robots in Gazebo Simulation Environment. In *Proceedings of the 2024 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, Sakhir, Bahrain, 20–21 November 2024; pp. 568–574.
19. Katona, K.; Neamah, H.A.; Korondi, P. Obstacle Avoidance and Path Planning Methods for Autonomous Navigation of Mobile Robot. *Sensors* **2024**, *24*, 3573. [[CrossRef](#)] [[PubMed](#)]
20. Singh, H. Artificial Intelligence and Robotics Transforming Industries with Intelligent Automation Solutions. *SSRN Electron. J.* **2020**. [[CrossRef](#)]
21. Ajeil, F.H.; Ibraheem, I.K.; Azar, A.T.; Humaidi, A.J. Autonomous navigation and obstacle avoidance of an omnidirectional mobile robot using swarm optimization and sensors deployment. *Int. J. Adv. Robot. Syst.* **2020**, *17*, 1729881420929498. [[CrossRef](#)]
22. Taheri, H.; Hosseini, S.R.; Nekoui, M.A. Deep Reinforcement Learning with Enhanced PPO for Safe Mobile Robot Navigation. *arXiv* **2024**, arXiv:2405.16266.
23. Zhu, K.; Zhang, T. Deep reinforcement learning based mobile robot navigation: A review. *Tsinghua Sci. Technol.* **2021**, *26*, 674–691. [[CrossRef](#)]
24. Tiwari, R.; Srinivaas, A.; Velamati, R.K. Adaptive Navigation in Collaborative Robots: A Reinforcement Learning and Sensor Fusion Approach. *Appl. Syst. Innov.* **2025**, *8*, 9. [[CrossRef](#)]
25. Zhu, Y.; Wan Hasan, W.Z.; Harun Ramli, H.R.; Norsahperi, N.M.H.; Mohd Kassim, M.S.; Yao, Y. Deep Reinforcement Learning of Mobile Robot Navigation in Dynamic Environment: A Review. *Sensors* **2025**, *25*, 3394. [[CrossRef](#)] [[PubMed](#)]
26. Winder, D. Tesla Hacked as Electric Cars Targeted in \$1 Million Hacking Spree. Available online: <https://www.forbes.com/sites/daveywinder/2024/01/27/tesla-hacked-as-electric-cars-targeted-in-1-million-hacking-spree/> (accessed on 5 May 2026).
27. Researchers Warn AI ‘Blind Spot’ Could Allow Attackers to Hijack Self-Driving Vehicles. Available online: <https://news.gatech.edu/news/2026/01/27/researchers-warn-ai-blind-spot-could-allow-attackers-hijack-self-driving-vehicles> (accessed on 5 May 2026).
28. Mohammadi, A.; Ahmari, R.; Hemmati, V.; Owusu-Ambrose, F.; Mahmoud, M.N.; Kebria, P.; Homaifar, A. Detection of Multiple Small Biased GPS Spoofing Attacks on Autonomous Vehicles Using Time Series Analysis. *IEEE Open J. Veh. Technol.* **2025**, *6*, 1152–1163. [[CrossRef](#)]
29. Guizzaro, C.; Formaggio, F.; Tomasin, S. GNSS Spoofing Attack Detection By IMU Measurements Through A Neural Network. In *Proceedings of the 2022 10th Workshop on Satellite Navigation Technology (NAVITEC)*, Noordwijk, The Netherlands, 5–7 April 2022; pp. 1–6.
30. Ibrahim, A.D.M.; Hussain, M.; Hong, J.-E. Deep learning adversarial attacks and defenses in autonomous vehicles: A systematic literature review from a safety perspective. *Artif. Intell. Rev.* **2024**, *58*, 28. [[CrossRef](#)]
31. Botta, A.; Rotbei, S.; Zinno, S.; Ventre, G. Cyber security of robots: A comprehensive survey. *Intell. Syst. Appl.* **2023**, *18*, 200237. [[CrossRef](#)]
32. Prasad, A.; Chandra, S. Defending ARP Spoofing-based MitM Attack using Machine Learning and Device Profiling. In *Proceedings of the 2022 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, Greater Noida, India, 4–5 November 2022; pp. 978–982.
33. Avci, İ.; Koca, M. Cybersecurity attack detection model, using machine learning techniques. *Acta Polytech. Hung.* **2023**, *20*, 29–44. [[CrossRef](#)]
34. Pan, D.; Ge, X.; Ding, D.; Han, Q.-L. Simultaneous Cyber Attack Estimation and Radar Spoofing Attack Detection for Connected Automated Vehicles. In *Proceedings of the IECON 2023—49th Annual Conference of the IEEE Industrial Electronics Society*, Singapore, 16–19 October 2023; pp. 1–6.
35. Zhang, K.; Keliris, C.; Parisini, T.; Jiang, B.; Polycarpou, M.M. Passive Attack Detection for a Class of Stealthy Intermittent Integrity Attacks. *IEEE/CAA J. Autom. Sin.* **2023**, *10*, 898–915. [[CrossRef](#)]

36. Hu, M.; Bu, L.; Bian, Y.; Qin, H.; Sun, N.; Cao, D.; Zhong, Z. Hierarchical Cooperative Control of Connected Vehicles: From Heterogeneous Parameters to Heterogeneous Structures. *IEEE/CAA J. Autom. Sin.* **2022**, *9*, 1590–1602. [CrossRef]
37. Xie, M.; Ding, D.; Ge, X.; Han, Q.-L.; Dong, H.; Song, Y. Distributed Platooning Control of Automated Vehicles Subject to Replay Attacks Based on Proportional Integral Observers. *IEEE/CAA J. Autom. Sin.* **2024**, *11*, 1954–1966. [CrossRef]
38. Arafin, M.T.; Kornegay, K. Attack Detection and Countermeasures for Autonomous Navigation. In Proceedings of the 2021 55th Annual Conference on Information Sciences and Systems (CISS), Baltimore, MD, USA, 24–26 March 2021; pp. 1–6.
39. Shen, J.; Won, J.Y.; Chen, Z.; Chen, Q.A. Drift with devil: Security of Multi-Sensor fusion based localization in High-Level autonomous driving under GPS spoofing. In Proceedings of the 29th USENIX Security Symposium (USENIX Security 20), Boston, MA, USA, 12–14 August 2020; pp. 931–948.
40. Liu, S.; Cheng, X.; Yang, H.; Shu, Y.; Weng, X.; Guo, P.; Zeng, K.C.; Wang, G.; Yang, Y. Stars can tell: A robust method to defend against GPS spoofing attacks using off-the-shelf chipset. In Proceedings of the 30th USENIX Security Symposium (USENIX Security 21), Virtual, 11–13 August 2021; pp. 3935–3952.
41. Zhou, Z.; Li, H.; Lu, M. Doppler-Based RAIM for GNSS Spoofing Detection in Vehicular Applications. *IEEE Trans. Veh. Technol.* **2025**, *74*, 10306–10320. [CrossRef]
42. Chattopadhyay, A.; Lam, K.-Y.; Tavva, Y. Autonomous Vehicle: Security by Design. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 7015–7029. [CrossRef]
43. Wang, C.; Tok, Y.C.; Poolat, R.; Chattopadhyay, S.; Elara, M.R. How to secure autonomous mobile robots? An approach with fuzzing, detection and mitigation. *J. Syst. Archit.* **2021**, *112*, 101838. [CrossRef]
44. Cybersecurity and Infrastructure Security Agency (CISA). *Autonomous Ground Vehicle Security Guide: Transportation Systems Sector*; CISA: Washington, DC, USA, 2021. Available online: <https://www.cisa.gov/resources-tools/resources/autonomous-ground-vehicle-security-guide> (accessed on 6 June 2026).
45. Deng, Y.; Zhang, T.; Lou, G.; Zheng, X.; Jin, J.; Han, Q.L. Deep learning-based autonomous driving systems: A survey of attacks and defenses. *IEEE Trans. Ind. Inform.* **2021**, *17*, 7897–7912. [CrossRef]
46. Issa, A.S.A.; Albayrak, Z. DDos attack intrusion detection system based on hybridization of CNN and LSTM. *Acta Polytech. Hung.* **2023**, *20*, 105–123. [CrossRef]
47. Kandasamy, V.; Roseline, A.A. Harnessing advanced hybrid deep learning model for real-time detection and prevention of man-in-the-middle cyber attacks. *Sci. Rep.* **2025**, *15*, 1697. [CrossRef] [PubMed]
48. Satpathy, S.; Swain, P.K.; Mohanty, S.N.; Basa, S.S. Enhancing Security: Federated Learning against Man-In-The-Middle Threats with Gradient Boosting Machines and LSTM. In Proceedings of the 2024 IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Niagara Falls, ON, Canada, 15–18 July 2024; pp. 1–8.
49. Majumder, S.; Deb Barma, M.K.; Saha, A. ARP spoofing detection using machine learning classifiers: An experimental study. *Knowl. Inf. Syst.* **2025**, *67*, 727–766. [CrossRef]
50. Bolboacă, R.; Haller, P.; Kontses, D.; Papageorgiou-Koutoulas, A.; Doulgeris, S.; Zingopis, N.; Samaras, Z. Tampering Detection for Automotive Exhaust Aftertreatment Systems using Long Short-Term Memory Predictive Networks. In Proceedings of the 2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), Genoa, Italy, 6–10 June 2022; pp. 358–367.
51. Dasgupta, S.; Rahman, M.; Islam, M.; Chowdhury, M. A Sensor Fusion-Based GNSS Spoofing Attack Detection Framework for Autonomous Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 23559–23572. [CrossRef]
52. Dimos, A.; Skoutas, D.N.; Nomikos, N.; Skianis, C. A Survey on UxV Swarms and the Role of Artificial Intelligence as a Technological Enabler. *Drones* **2025**, *9*, 700. [CrossRef]
53. Yan, X.; Sarkar, M.; Lartey, B.; Gebru, B.; Homaifar, A.; Karimoddini, A.; Tunstel, E. An Online Learning Framework for Sensor Fault Diagnosis Analysis in Autonomous Cars. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 14467–14479. [CrossRef]
54. Wan, T.J.; AI Shamaileh, K. ML-Auto-UGV-Injection-Detection: A Machine Learning-Powered Solution for Safe Autonomous Robotic Ground Navigation in Cyber-Contested Environments. GitHub. 2026. Available online: <https://github.com/W-w-star/ML-Auto-UGV-injection-detection> (accessed on 21 June 2026).
55. Clearpath Robotics. Jackal UGV—Small Weatherproof Robot. Available online: <https://clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/> (accessed on 15 March 2025).
56. Clearpath Robotics. Inertial Measurement Units. Available online: <https://docs.clearpathrobotics.com/docs/ros/config/yaml/sensors/imu/> (accessed on 15 March 2025).
57. Clearpath Robotics. Manipulation in Gazebo Ignition. Available online: <https://docs.clearpathrobotics.com/docs/ros2humble/ros/tutorials/manipulation/gazebo/> (accessed on 15 March 2025).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.