2022 NDIA MICHIGAN CHAPTER
GROUND VEHICLE SYSTEMS ENGINEERING
AND TECHNOLOGY SYMPOSIUM
CYBERSECURITY OF GROUND SYSTEMS TECHNICAL SESSION
AUGUST 16-18, 2022 - NOVI, MICHIGAN

DETECTION AND MITIGATION OF ERRONEOUS AND MALICIOUS DATA IN VEHICLE SENSOR NETWORKS

Ryan McBee, Jonathan Wolford, Abe Garza

Southwest Research Institute, San Antonio, TX

ABSTRACT

This paper describes research into the applicability of anomaly detection algorithms using machine learning and time-magnitude thresholding to determine when an autonomous vehicle sensor network has been subjected to a cyber-attack or sensor error. While the research community has been active in autonomous vehicle vulnerability exploitation, there are often no well-established solutions to address these threats. In order to better address the lag, it is necessary to develop generalizable solutions which can be applied broadly across a variety of vehicle sensors. The current measured results achieved for time-magnitude thresholding during this research shows a promising aptitude for anomaly detection on direct sensor data in autonomous vehicle platforms. The results of this research can lead to a solution that fully addresses concerns of cyber-security and information assurance in autonomous vehicles.

Citation: R. McBee, J. Wolford, A. Garza, "Detection and Mitigation of Erroneous and Malicious Data in Vehicle Sensor Networks," In *Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium* (GVSETS), NDIA, Novi, MI, Aug. 16-18, 2022

1. INTRODUCTION

Introduction of erroneous data into modern sensor fusion algorithms can occur through take-over vulnerabilities and lower tech issues such as damaged or obstructed sensors. These vulnerabilities include GPS spoofing, which can be used in off-road and wrong-way attacks with high success rates [1]. Damaged or obstructed sensors can also introduce anomalous sensor data leading to unintended results. Detection and mitigation of erroneous data in real-time on vehicle sensor networks may flag issues with sensors that would be difficult or impossible to detect from access to a single sensor output.

While the research community has demonstrated take-over vulnerabilities with high-success rates, there is a gap in the availability of technologies to mitigate these threats. The purpose of this research was to evaluate the applicability of anomaly detection algorithms using machine learning and time-magnitude thresholding to determine when an autonomous vehicle sensor network has been subjected to a cyber-attack or sensor error. Notably, this focuses on operating on data directly from the sensors in front of any sensor fusion or other system. The research focuses on a single measurement, velocity, obtained from a variety of sensors commonly used in autonomous vehicles such as GPS, wheel speed, and LIDAR. However, these results are generalizable to other sensor networks of interest in ground vehicle research and development.

2. BACKGROUND

Previously, researchers at SwRI demonstrated the ability to remotely exploit automated vehicle sensors such as GPS [2]. SwRI also exploited automated vehicle sensors by the manipulation of object classification algorithms [3]. Figure 1 shows examples of both exploits performed by SwRI. While the research community has been active in autonomous vehicle vulnerability exploitation, there are often no well-established solutions to address these threats. In order to better address the lag, it is necessary to develop generalizable solutions which can be applied broadly across a variety of vehicle sensors.

In order to simplify the process of evaluating multiple algorithms, this study was limited to vehicle velocity measurements. Velocity has the benefit that it can be obtained from multiple sensors, including camera, GPS, wheel speed, and LIDAR. Additionally, the data is often of a high consistency. An array of anomalies was investigated including those related to cyber-attacks and lower tech issues.

Detection of anomalous data has been used with limited success by filtering out object location data from camera and LIDAR sensors. This may be accomplished by flagging data from composite location measurements which exceed a tolerance threshold [4].

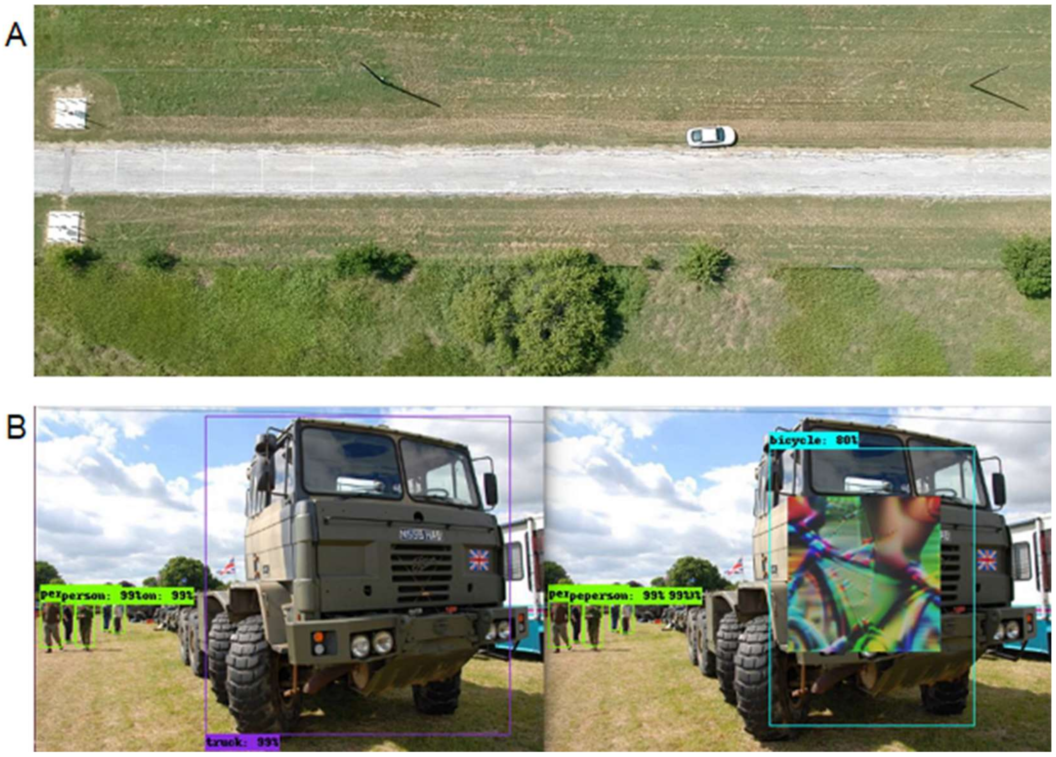

Figure 1. Examples of sensor manipulation include A) An offset applied mid-route forces an automated vehicle off the road, and B) an object classification algorithm classified as a truck as a bicycle

Combination approaches involving convolutional neural networks (CNNs) and Kalman filters have been studied which incorporate in-vehicle speed, GPS speed and in-vehicle acceleration data [5]. However, combination techniques like this are often limited to single sensor measurements per physical property and are therefore more suited for non-autonomous vehicles. Incorporation of techniques which leverage the same property being measured by multiple sensors is more suitable for autonomous vehicles and complex connected sensor networks.

These anomaly detection algorithms, which feed on values being measured by multiple sensors embedded in a complex sensor network, can be used to flag issues that would be difficult to detect from access to a single sensor output.

3. TECHNICAL APPROACH

To fulfill the objective of the research, the technical approach was divided into five main tasks: data collection, velocity data extraction, anomaly insertion, detection, and testing and evaluation. **Error! Reference source not found.** below shows the function diagram of the approach described in this paper.

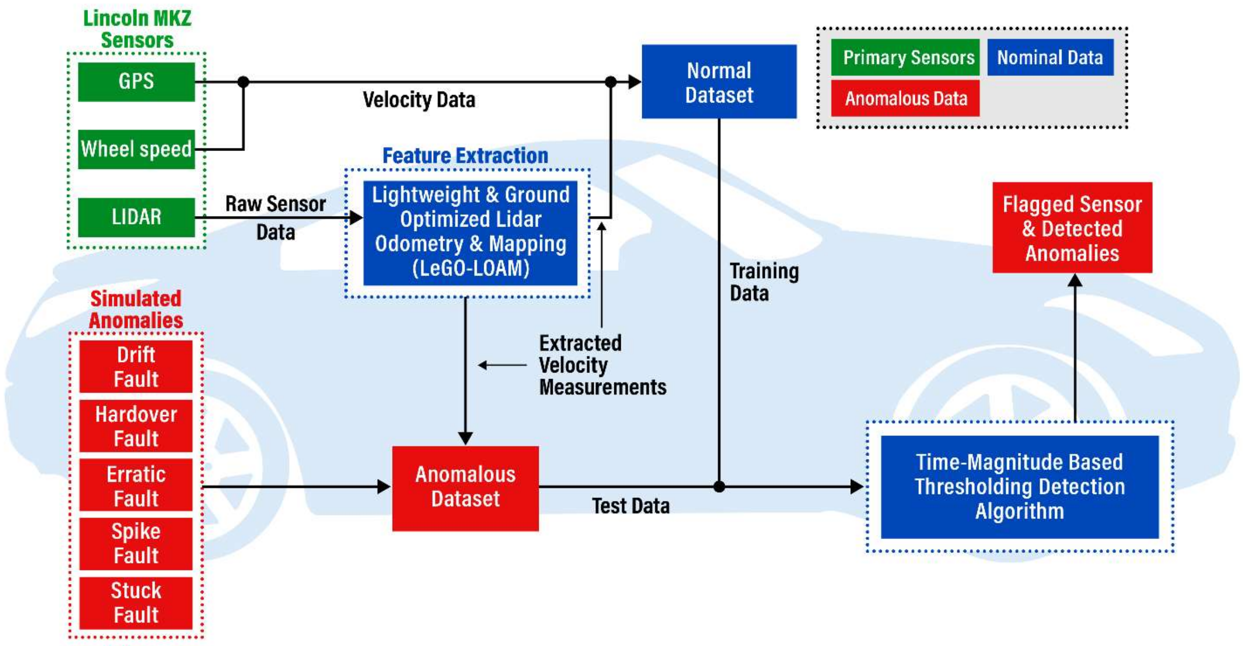

Figure 2. Functional Diagram

3.1. Data Collection

New and pre-recorded datasets from an autonomous vehicle platform were utilized. This vehicle system had raw sensor feeds recording one LIDAR sensor, a wheel encoder, and GPS. Camera data was also recorded, but ultimately not used due to the available resources for extracting useable velocity data. The data was recorded in the Robot Operation System (ROS) bag format and captured in several environments, including a private test track, public roads, and highways in differing amounts of traffic and speed to provide diversity in data. Data was also captured at residential intersections including traffic lights and stop signs.

Collection of data focused on targeted vehicle velocities including 10 MPH, 25

MPH, and 45 MPH, which are described as slow, medium, and fast, respectively.

3.2. Data Processing and Feature Extraction

For algorithm development, researchers collected a dataset of various sensors that generate velocity measurements as well as additional sensors which could be used to supplement the velocity data. Raw velocity from the GPS and wheel speed sensors were directly used. To generate a velocity measurement from the lidar, the Lightweight and Ground Optimized Lidar Odometry and Mapping (LeGO-LOAM) algorithm was used. During analysis, the extracted LIDAR velocity data was found to contain large amounts of noise particularly when the data set contained non-static objects such as other moving cars in the scene. These objects were difficult to filter solely using the LeGO-LOAM algorithm. An internally developed tool that uses machine learning to remove cars and other non-static objects from the lidar point cloud was used to help filter data and remove noise. Additionally, data from the areas where the vehicles moved faster, such as highway, was found to be significantly noisier for the LIDAR, so our data collection was focused on residential areas that had fewer non-static objects and lower overall noise.

Additional data from the cameras, Inertial Measurement Unit (IMU), throttle, brake, and steering wheel direction were also collected for potential future use.

3.3. Anomaly Insertion

There was a lack of data available publicly for the task of detecting anomalies in autonomous vehicle sensors. Thus, it was deemed necessary to define and implement an anomaly generator that could be used to both train and test the models created during this research. The researchers implemented a series of faults discussed in prior research such as drift and spike faults and common cyber security attacks such as replay attacks that would behave similar to what a malicious party would likely use in an attack [6]. Multiple types of faults/attacks were implemented, and each were applied to the collected data. The simulated faults and attacks are described in the following text.

Drift fault is where a slight acceleration is applied to the sensor data, thereby causing a growing offset, or “drift”, that gradually builds up. Linear and exponential drifts were both used as inserted anomalies. An example is shown in Figure 2.

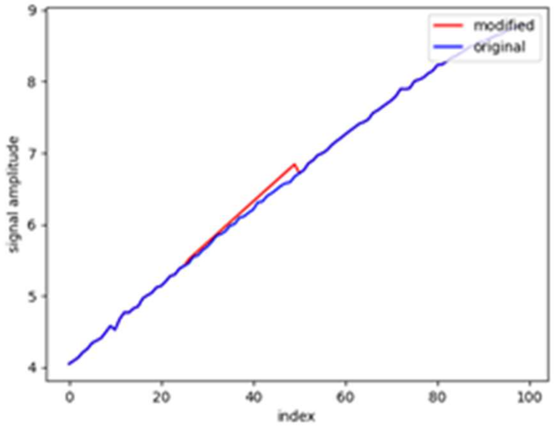

Figure 2. An Example of a Drift Fault

Spike faults add positive momentary pulses to the signal that imitate a sensor shorting. An example is shown in Figure 4.

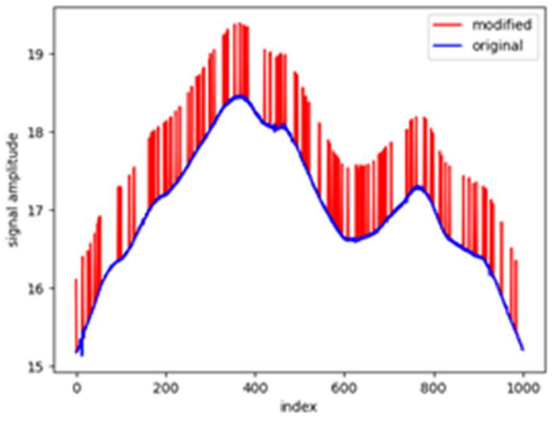

Figure 3. An Example of a Spike Fault

Hardover faults imitate whenever a sensor saturates and temporarily reports the maximum sensor reading. An example is shown in Figure 4.

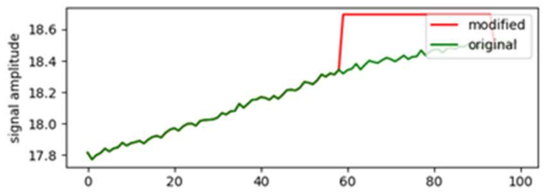

Figure 4. An Example of a Hardover Fault

Erratic faults add a normally distributed noise profile to the overall signal. An example is shown in Figure 5.

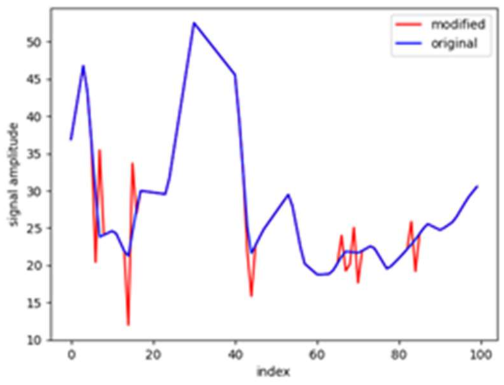

Figure 5. An Example of an Erratic Fault

Stuck faults imitate a sensor getting jammed at a fixed value temporarily. An example is shown in Figure 6.

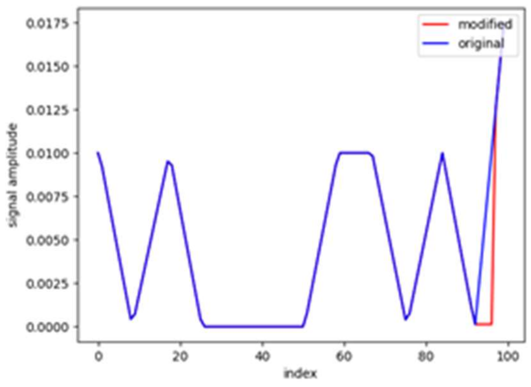

Figure 6. An Example of a Stuck Fault

Simulated attacks included replay and offset attacks. A replay attack is where communications are intercepted/recorded and fraudulently resent to misdirect the receiver. The replay attacks were implemented by replaying data recorded from separate data collections where various parameters such as location and traffic conditions differed from the test dataset. An offset attack is a replay attack where the intercepted data is simply delayed by some amount of time. Fully and partially blocked sensors were also simulated in the anomalous datasets.

3.4. Detection

Multiple algorithms were explored in this research for anomaly detection capabilities; time-magnitude based thresholding, Temporal Autoencoder (TAE), and Density-Based Spatial Clustering of Applications with Noise (DBSCAN).

The data had to be time aligned to be usable with the data driven anomaly detection approaches. All the different sensors used would come in at different times which, in its raw form, is difficult to use. To handle this, the data was resampled at 10 Hz and linear interpolation was used to approximate the sensor values at the resampled points. 10 Hz was chosen because that was the data rate of the slowest sensor, the GPS. This rate was also found to be a good trade-off between data fidelity and processing speed for the different detection algorithms.

The first and simplest approach for a data driven multi-sensor anomaly detection algorithm was a time-magnitude based threshold algorithm. This algorithm works by taking a predefined window of time-aligned data and finding the absolute difference in velocity between two different sensors. The algorithm would then count the total number of data points where the difference in values exceed a pre-defined threshold. The algorithm counts the number of points that exceed the difference threshold and flags the data window as an anomaly if the total number exceed some anomaly count threshold. These parameters for length of time window, difference threshold, and total points threshold are tunable and can be configured to create multiple detection algorithms tuned to detect different kinds of faults.

The final time-magnitude detection algorithm utilized multiple layered thresholding algorithms in combination to create a robust anomaly detection algorithm. Each algorithm was tuned individually for different sets of anomalies. A high difference threshold and low anomaly count threshold was used to detect transient and erratic faults while a lower difference threshold with a larger anomaly count threshold was used to find lower frequency anomalies such as sensor drift.

To determine the values for these thresholds, the data collected for this project was analyzed and the parameters tuned to minimize false positives or false negatives. The final time-magnitude detection algorithm comprised of three different time-magnitude algorithms.

Another approach that was briefly explored was determining if a Temporal Autoencoder (TAE) could learn and rebuild the nominal velocity signal when fed multiple velocity topics with anomalies present. This TAE follows the scheme shown in Figure 7.

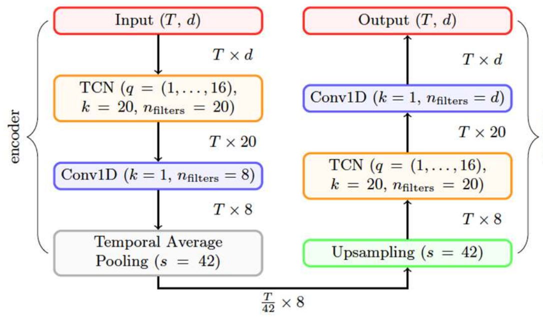
Input (T, d)
$T \times d$
TCN $(q = (1,\dots,16), k = 20, n_{\text{filters}} = 20)$
$T \times 20$
Conv1D $(k = 1, n_{\text{filters}} = 8)$
$T \times 8$
Temporal Average Pooling $(s = 42)$
encoder
$\frac{T}{42} \times 8$
Output (T, d)
$T \times d$
Conv1D $(k = 1, n_{\text{filters}} = d)$
$T \times 20$
TCN $(q = (1,\dots,16), k = 20, n_{\text{filters}} = 20)$
$T \times 8$
Upsampling $(s = 42)$
decoder

Figure 7. TCN-AE Architecture [7]

The goal for this architecture was that by reconstructing a 'nominal' signal from multiple sensor readings, where one or more of the sensors may be hijacked, then a system could determine if a sensor is faulty or malicious by computing the difference between the nominal generated signal value and the various input topics. If a sensor reading is outside of the pre-determined threshold, it would then be flagged.

The last machine learning focused algorithm explored was DBSCAN, which is a modern clustering algorithm useful for anomaly detection as it can be trained on data representing only normal sensor traffic [8]. DBSCAN groups together high-density areas in the data by assigning clusters to them. Data that exists outside of those clusters are considered anomalies. Figure 8 offers a visual explanation of DBSCAN.

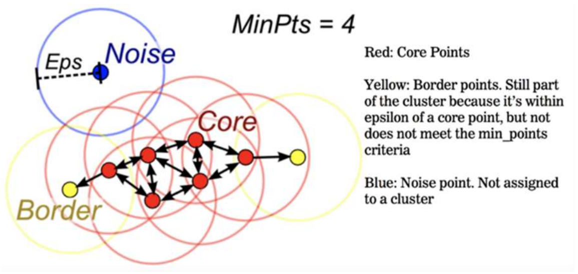
MinPts = 4
Eps
Noise
Core
Border
Red: Core Points
Yellow: Border points. Still part of the cluster because it's within epsilon of a core point, but not does not meet the min_points criteria
Blue: Noise point. Not assigned to a cluster

Figure 8. Visual Representation of DBSCAN [9]

3.5. Testing and Evaluation

Testing was implemented by adding anomalies to a single data point at a time, grabbing a fixed time window of ten seconds of data and applying the algorithm to evaluate its ability to detect the anomaly. The algorithm was applied to one sensor at a time to evaluate its performance using datasets containing collected data from the target sensor. The algorithm was also tested with no anomalies on each sensor to represent the nominal operating conditions. This provides a baseline performance metric for the detection algorithm.

The following metrics were evaluated during testing for each sensor and fault type:

- Accuracy – the measure of correct predictions over all predictions.

- Precision – the measure of correctly detected anomalies out of all anomaly detections.
- Recall – the measure of correctly detected anomalies out of all predictions.
- F1 – the harmonic mean, or average, of the precision and recall metrics.

These metrics are calculated from the number of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). These are defined in the context of this research as follows:

- TP – An anomalous data point is correctly predicted to be an anomaly.
- TN – A non-anomalous data point is correctly predicted to **not** be an anomaly.
- FP – An anomalous data point is incorrectly predicted to **not** be an anomaly.
- FN – A non-anomalous data point is incorrectly predicted to be an anomaly.

The formulas for each of the metrics are listed below:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+F} \quad (1)$$

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

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (4)$$

All algorithms, data, and anomalies were implemented using the Python programming language.

4. RESULTS

Of the four approaches explored, the time-magnitude approach yielded the best results. The thresholding approach was able to detect most of the simulated faults. It was also able to overcome deficiencies in the data set such as noise. The majority of the noise in the data resulted from the velocity extracted from the LIDAR which would sporadically change its value. While this can be fixed by having a more robust LIDAR velocity extraction algorithm, that effort is outside the scope of this research.

The three-machine learning based anomaly detection approaches were more difficult to implement due to the noise in the data set. The current data set is generally insufficient to adequately train the machine learning algorithms as the algorithms would learn poorly and not generalize the inputs well.

Therefore, only the results pertaining to the time-magnitude threshold anomaly detection algorithm are shown in Table and Table. These tables show the accuracy, precision, recall, and F1 measurements for each sensor and fault type tested. The measured results are split into two tables for readability.

Table 1. Time-Magnitude Thresholding Measured Accuracy and Precision Results

Fault/Attack	Accuracy			Precision		
	LIDAR	GPS	Wheelspeed	LIDAR	GPS	Wheelspeed
No Fault	90.88%	100.00%	100.00%	N/A	N/A	N/A
Linear Drift	94.22%	84.48%	81.59%	86.00%	86.42%	87.50%
Exponential Drift	93.75%	96.53%	92.36%	82.42%	93.48%	86.90%
Spike	87.32%	94.09%	93.02%	83.71%	95.48%	62.66%
Hardover	93.94%	95.78%	96.44%	83.68%	89.39%	91.87%
Erratic	66.67%	69.06%	65.75%	42.58%	86.15%	57.14%
Stuck	72.60%	74.58%	71.53%	67.48%	85.02%	79.14%
Replay	94.14%	95.60%	96.34%	89.62%	92.05%	91.40%
Offset	91.96%	91.96%	94.76%	81.00%	84.75%	89.32%
Partial Block	90.91%	96.85%	96.50%	78.67%	91.25%	89.87%
Full Block	91.80%	96.48%	96.88%	82.93%	88.46%	90.12%
TOTAL	**88.02%**	**90.49%**	**89.56%**	**77.81%**	**89.25%**	**82.59%**

Table 2. Time-Magnitude Thresholding Measured Recall and F1 Score Results

Fault/Attack	Recall			F1		
	LIDAR	GPS	Wheelspeed	LIDAR	GPS	Wheelspeed
No Fault	N/A	N/A	N/A	N/A	N/A	N/A
Linear Drift	97.73%	68.63%	48.28%	91.49%	76.50%	62.22%
Exponential Drift	97.40%	95.56%	86.90%	89.29%	94.51%	86.90%
Spike	78.09%	85.92%	55.93%	80.80%	90.44%	59.10%
Hardover	99.67%	99.24%	98.71%	90.98%	94.06%	95.17%
Erratic	10.44%	8.55%	2.95%	16.77%	15.56%	5.61%
Stuck	33.69%	27.29%	22.12%	44.94%	41.31%	34.58%
Replay	95.00%	94.19%	97.70%	92.23%	93.10%	94.44%
Offset	95.29%	95.24%	95.83%	87.57%	89.69%	92.46%
Partial Block	85.51%	97.33%	97.26%	81.94%	94.10%	93.42%
Full Block	90.67%	100.00%	100.00%	86.62%	93.88%	94.81%
TOTAL	**78.35%**	**77.20%**	**70.57%**	**76.26%**	**78.32%**	**71.87%**

The target thresholds for all four measurements were 95% or greater. GPS and wheel speed sensors achieved the target **Accuracy** thresholds for the majority of the different faults and attacks that were tested; furthermore, all three sensors achieved the **Recall** target thresholds on the majority of the faults and attacks tested. The time-magnitude threshold anomaly detection algorithm worked best on the GPS sensor data and had generally good results in terms of accuracy and recall. Most results show at least 90% and only a few resulted in less than 85%.

The algorithm had poorer results in terms of precision which in turn lowered the F1 score. The algorithm particularly had issues when encountering erratic and stuck faults on all sensors. Out of all the sensors, LIDAR experienced the poorest performance. This was expected as the velocity data from the LIDAR contained noise despite the filters.

Overall, the Precision and Recall scores show that false negatives and false positives were generally low with a high true positive detection rate. The results are generally close to reaching the target thresholds that were set prior to the research.

Cleaner velocity data would vastly improve the results when detecting anomalies on the LIDAR data. This can be improved either through better filtering of problematic data points in the raw LIDAR sensor data, or through use of an alternative algorithm. Further improvements of the algorithm can be made through more advanced threshold adjustments or layering. In addition, a less noisy and more extensive data set will improve the useability of the machine learning algorithms. The machine learning algorithms combined with the time magnitude thresholding will likely provide a more complete, robust solution for detecting anomalies in an autonomous vehicle sensor network.

5. FUTURE DEVELOPMENT

Future work on this research includes refinement of the time-magnitude thresholding anomaly detection algorithm and the data set, testing on an autonomous vehicle platform, fully implementing and evaluating the machine-learning based anomaly detection solutions and expanding the sensor and data types present in the anomaly detection system. Figure 9 shows the autonomous vehicle platform used during this research.

Refinement of the thresholding algorithm and data set can improve the anomaly detection metrics enough to reach the target 95% threshold. This can also improve its response when encountering the most problematic anomalies, erratic and stuck faults. Moving this system from a pure Python implementation to an autonomous vehicle platform will provide better validation of the anomaly detection algorithms and showcase its capabilities.

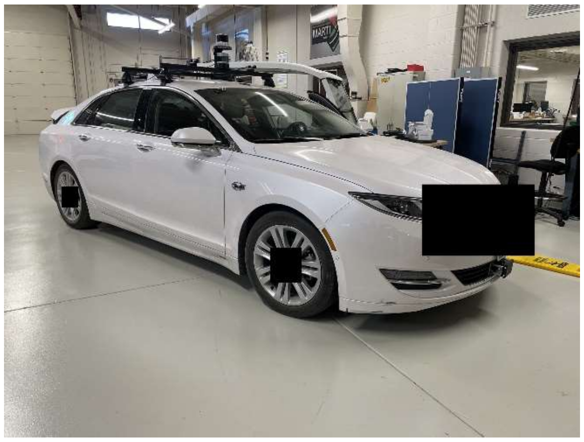

Figure 9. Autonomous Vehicle Platform

Full implementation of a machine learning anomaly detection algorithm alongside the thresholding algorithm can provide a more complete and robust solution that reaches all target thresholds and minimize any false positives which can be problematic for any anomaly detection solution. This can be accomplished by further improvement of the data set by additional data collection and noise reduction. Expansion of the data set to include other sensors such as camera and radar alongside other data types such as object tracking and localization, can further diversify the data for more comprehensive development of an autonomous vehicle sensor network anomaly detection solution.

6. CONCLUSION

The current results achieved during this research shows a promising aptitude for anomaly detection on direct sensor data in autonomous vehicle platforms. The measured results do not quite reach the target thresholds, but with further improvements, this system can reach those thresholds and incorporate a larger variety of data and sensors. The results of this research can lead to a solution that addresses concerns of cyber-security and information assurance in autonomous vehicles.

7. REFERENCES

[1]J. Shen, J. Y. Won, Z. Chen, en Q. A. Chen, “Drift with Devil: Security of Multi-Sensor Fusion based Localization in High-Level Autonomous Driving under GPS Spoofing (Extended Version)”, CoRR, vol abs/2006.10318, 2020.

[2]V. Murray, “Legal Over-the-Air Spoofing of GPS and the Resulting Effects on Autonomous Vehicles”, Available: http://i.blackhat.com/USA-19/Wednesday/us-19-Murray-Legal-GNSS-Spoofing-And-Its-Effects-On-Autonomous-Vehicles-wp.pdf [Accessed 3 March 2022]

[3]D. Chambers, H. Garza, “Physically Realizable Adversarial Example for Convolutional Object Detection Algorithms,” Proc. SPIE 10988, Automatic Target Recognition XXIX, 109880R (14 May 2019); doi: 10.1117/12.2520166

[4]S. Lambermont, J. Kim, J. Wei, G. Bhatia, “Operation-Security System for an Automated Vehicle”, Aptiv Technlogies Limited, St. Michael, BB 2020

[5]F. van Wyk, Y. Wang, A. Khojandi, N. Masoud, “Real-Time Sensor Anomaly Detection and Identification in Automated Vehicles” *IEEE Trans. Intelligent Transportation Systems*, vol. 21, pp. 1264-1276, March 2020

[6] L. Biddle and S. Fallah, "A Novel Fault Detection, Identifcation and Prediction Approach for Autonomous Vehicle Controllers Using SVM," 5 April 2021. [Online]. Available: https://link.springer.com/article/10.1007/s42154-021-00138-0. [Accessed 3 March 2022].

[7]W. K. a. T. B. M. Thill, "Time Series Encodings with Temporal Convolutional Networks," 2020. [Online]. Available: https://doi.org/10.1007/978-3-030-63710-1_13. [Accessed 3 March 2022].

[8] Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996, August). A density-based algorithm for discovering clusters in large spatial databases with noise. In Kdd (Vol. 96, No. 34, pp. 226-231).

[9] E. Lutins, *DBSCAN: What is it? When to use it? How to use it.*, Medium, Sep. 5, 2017. Available: https://medium.com/@elutins/dbscan-what-is-it-when-to-use-it-how-to-use-it-8bd506293818