Malicious Data Detection in TurtleBot

Adishree Sane#1, Sharvari Doijode #2, Shubham Melvin Felix#3, Uddesh Shyam Kshirsagar #4

1 adsane@syr.edu 2 ssdoijod@syr.edu

3 smfelix@syr.edu 4 ukshirsa@syr.edu

Abstract— **The increasing adoption of robots in smart home environments raises questions about safety, security, and reliability, especially for platforms like TurtleBot-a smart sweeper bot-that rely heavily on sensor data for navigation, such as LIDAR, IMU, and cameras, to interact with the environment. In this perspective, TurtleBot is vulnerable to cyber-attacks like data tampering or even injection; the interconnected networks of IoT increase these risks since disruptions in the functioning of the robot can be achieved by data injection.**

This project aims to develop an anomaly detection system, which will detect unusual or malicious changes within the sensor data of the autonomous robot TurtleBot3 Burger. This system focuses on LIDAR and IMU sensor data that are used for its basic movements and interactions in space. The system spots abnormalities using machine learning models such as Isolation Forest, Autoencoders, and Random Forest. The simulation setup in Gazebo involves the TurtleBot3 Burger, collecting IMU and LIDAR data with ROSbag, and then preprocessing the data. For training the system, features are extracted, data is synchronized, and models are evaluated. The results show that the system can successfully detect abnormal sensor data, ensuring the robot operates safely in the simulation.

I. Introduction

With the growing integration of self-sufficient robots into smart home environments, large security concerns have emerged, specifically around sensor information manipulation. As these robots rely closely on sensors to navigate and engage with their surroundings, ensuring the integrity of this facts is paramount. Malicious sports such as information injection, sensor spoofing, and tampering present a extreme hazard, potentially inflicting robots to malfunction or maybe behave unpredictably. These vulnerabilities may be exploited through cyber attackers, leading to compromised operations or even malicious control over the robotic. The want for strong security features is mainly vital in environments where robots perform critical tasks like cleansing, tracking, and helping, all of which rely on correct environmental sensing.

TurtleBot, a cellular robot designed for self-sustaining sweeping duties, is prepared with an array of sensors together with LIDAR, IMU, and cameras. These sensors allow the robotic to map its surroundings, stumble on obstacles, and navigate autonomously inside smart homes. However, the robot's heavy reliance on sensor records makes it vulnerable to various kinds of cyber-attacks that focus on its sensory inputs. If an attacker manipulates the sensor information, the robot can be misled into making wrong decisions, such as colliding with obstacles, failing to hit upon changes within the environment, or even altering its supposed route. Given the superiority of such dangers in IoT and IoRT environments, protecting the integrity of sensor information becomes a crucial a part of securing those robot systems.

This assignment specializes in the improvement of an Anomaly Detection System (ADS) designed to guard TurtleBot from malicious records manipulation. By detecting unusual patterns inside the sensor facts, the ADS objectives to identify potential threats together with spoofed LIDAR readings or injected IMU information. The system will appoint advanced device studying techniques to analyse the sensor information in actual-time and flag any inconsistencies or outliers that could imply tampering or manipulation. A key objective is to make sure that the system remains both powerful and green, able to figure out anomalies without overly complicating the robot's operational workflow. This ADS is designed to paintings with TurtleBot's present software and hardware additives at the same time as being flexible enough to scale to different IoT devices.

To gain this, the ambiguity detection system may be advanced and examined in a managed software-based totally surroundings the use of ROS 2 (Robot Operating System) and Gazebo simulations. This setup removes the want for bodily hardware throughout the development and testing stages, supplying the ability to simulate a huge variety of assaults and sensor disasters in a secure, repeatable manner. The experimental environment will allow to manipulate over variables, allowing iterative trying out and optimization of the detection model. With ROS and Gazebo, the development procedure can carefully mimic real-global situations, allowing the ADS to be evaluated below various situations and assault vectors, making sure its robustness earlier than deployment in actual-international environments. This technique will offer a strong foundation for securing self-sustaining robots in clever domestic settings, ultimately contributing to the wider discipline of IoT protection.

II. Literature Review

The safety and reliability of anomaly detection in sensor data, especially from LIDAR and IMU technologies, is important for autonomous robots. This would include highly detailed 3[D mapping from LIDAR and the tracking of motion and orientation with IMUs. Such sensors in combination enable robots to efficiently work in challenging environments and execute complex tasks, but they are also starting to open challenges with anomaly or malicious activity detection in the data.

Various researchers have investigated techniques for anomaly detection in sensor data. [1] presented a review of machine learning and deep learning techniques on anomaly detection in IoT data, giving a deep view of how the current methodologies are being used in the identification of abnormalities in data streams. Their review categorizes the different approaches based on the type of data, learning modes, and evaluation criteria, and highlights areas where future research is needed to overcome existing challenges in anomaly detection. The paper further discussed Anomaly detection learning modes which can be classified into three types based on the availability of labels in the datasets used to create baselines[5], including supervised, semi-supervised, and unsupervised anomaly detection, as shown in Fig 1.

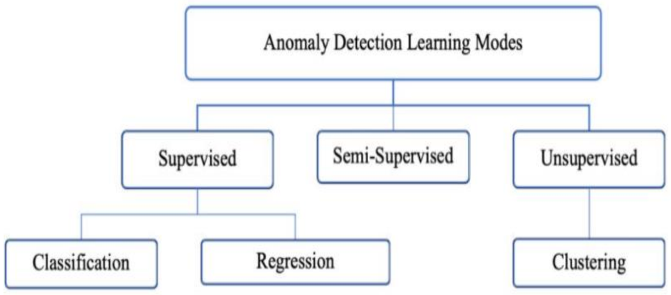


Fig 1. IoT anomaly detection learning modes

In robotics, [2] Patel et al. (2024) targeted on figuring out the security vulnerabilities inherent inside the TurtleBot3 robotic system. Their look at investigates the safety structure of those robots and highlights the potential dangers related to their substantial deployment in diverse environments. Specifically, the researchers emphasize the significance of know-how how cyber threats like records injection, sensor spoofing, and unauthorized get admission can compromise the functionality of robots. TurtleBot3, being a cellular robotic prepared with sensors including LIDAR, IMU, and cameras, faces several security challenges, particularly given the increasing integration of these robots into clever home systems and different IoT environments. Patel et al.'s work underscores the necessity of developing effective anomaly detection systems that may become aware of and mitigate those dangers in real-time. By addressing these vulnerabilities, their research aims to enhance the resilience of robot systems, making sure that they can continue to perform reliably and securely in actual-world packages. This look at serves as a foundation for building extra secure robots and is particularly relevant as the use of self-sufficient robots in public and private areas keeps developing.

[3]Chalapathy and Chawla (2019) presented an in-depth survey at the utility of deep mastering techniques in anomaly detection. Their evaluation protected diverse deep mastering models, consisting of autoencoders, convolutional neural networks (CNNs), and recurrent neural networks (RNNs), and tested how each play across one-of-a-kind domain names, from industrial automation to healthcare. One key takeaway from their work is the growing role of deep studying in improving anomaly detection systems, mainly in complicated environments in which traditional methods can also battle. The have a look at suggests that deep learning fashions, via getting to know complicated styles from big datasets, are higher prepared to locate subtle or previously unseen anomalies in sensor statistics. This insight is particularly precious for robotics, because it paves the way for integrating deep learning-primarily based anomaly detection into autonomous systems, increasing their protection and reliability. By leveraging the power of deep learning, researchers can increase greater state-of-the-art structures which could adapt to a extensive variety of operational eventualities, ensuring that robots can perform obligations accurately without being compromised by using outside threats or inner malfunctions.

[4]He et al. (2024) proposed a unique deep studying-primarily based approach, called TSAN, to detect sensor anomalies in cell robots. Their paintings build on preceding studies by introducing a new methodology for sensor anomaly detection that is designed for the demanding situations faced by means of mobile robots in dynamic environments. The TSAN model makes use of superior deep gaining knowledge of techniques to analyse sensor records from more than one sources, which include LIDAR, IMU, and cameras, and perceive inconsistencies or errors that might suggest problems consisting of sensor glide, calibration mistakes, or external interference.[4]He et al. Verified that their technique outperforms conventional methods in detecting sensor anomalies, imparting a extra reliable answer for retaining the accuracy and safety of robotic structures. These paintings no longer best advances the field of anomaly detection in robotics but additionally highlights the growing significance of leveraging deep mastering to address the complexities of real-time information processing in mobile robots. Their approach presents a vast step forward within the improvement of smarter, more resilient robots capable of functioning accurately in ever-changing environments.

The findings from this research together emphasize the growing importance of integrating LIDAR and IMU facts in the improvement of anomaly detection systems for robotic platforms. As self-reliant robots come to be greater integrated into dynamic and unpredictable environments, the need for greater strong detection strategies has grown. The fusion of LIDAR, IMU, and different sensor information permits the advent of a greater comprehensive information of the robotics' surroundings and movements, that's critical for identifying capacity anomalies that might compromise its operation. Researchers are continuing to explore how superior machine mastering and deep learning techniques can be implemented to enhance the accuracy and reliability of these structures. By combining the strengths of these methodologies, they aim to expand robots that aren't only capable of appearing complicated responsibilities autonomously however also resilient to errors and outside threats. This study is pushing the boundaries of what's possible inside the area of robotics, making sure that self-reliant robots can competently and efficaciously operate in real-time, dynamic conditions without being at risk of malicious attacks or sensor disasters.

III. Problem analysis

IoT has revolutionized connectivity inside independent systems, which includes TurtleBot, by allowing seamless statistics gathering and sharing between sensors and other components in real-time. This more suitable connectivity performs a critical role in duties which include navigation, where TurtleBot is based on sensor statistics to understand its environment, make choices, and navigate autonomously. By integrating sensors like IMU (Inertial Measurement Unit) and LIDAR (Light Detection and Ranging), the robot can appropriately map its surroundings, detect obstacles, and adjust its movements consequently. However, this interconnectivity, even as enhancing the robot's capabilities, additionally introduces full-size protection risks. With the upward thrust of IoT, there are developing issues approximately malicious actors who can make the most those vulnerabilities by manipulating the sensor facts, leading to faulty robot behaviour or, within the worst instances, accidents. These threats may be especially dangerous in real-world programs wherein robots perform duties autonomously without direct human supervision. As IoT maintains to develop, the significance of securing these structures becomes greater critical to making sure the safety and reliability of self-sustaining robots.

For TurtleBot, sensors like IMU and LIDAR are essential to its navigation and environmental consciousness. The IMU sensor facilitates the robot apprehend its orientation and motion, even as the LIDAR sensor affords special records approximately the distances and barriers in its environment. These sensors paintings collectively to permit TurtleBot to move safely and accurately via dynamic environments. However, any tampering with the records provided by means of those sensors could have serious consequences. If the facts from the IMU or LIDAR is manipulated—whether through sensor spoofing, records injection, or other malicious activities—the robot could make incorrect selections, together with veering off route, colliding with boundaries, or even failing to detect adjustments in its environment. Given the crucial role that sensor information plays in making sure secure operation, preserving the integrity of this statistics is of utmost importance. Ensuring records integrity way that the robot can depend upon correct, unaltered sensor inputs, which without delay contributes to both its functionality and general protection in appearing self-sustaining obligations.

Considering these safety worries, the development of the Anomaly Detection System (ADS) is important to guard TurtleBot from ability facts manipulation and make sure steady operation. The ADS is designed to continuously reveal the sensor data for any deviations from regular behaviour, which can suggest malicious interference, sensor faults, or environmental adjustments that affect the sensor readings. By continuously staring at the sensor information streams, the ADS can come across patterns that deviate from the expected variety of values, thereby identifying anomalies in actual time. This proactive method lets in the gadget to flag any suspicious or faulty records, which can otherwise cause incorrect decisions and hazardous robot behaviour. In addition to detecting manipulation or faults, the ADS is also equipped to differentiate among environmental changes and real sensor disasters, ensuring that the robot can adapt to dynamic conditions without compromising protection. This machine now not most effective identifies capacity security threats but additionally responds to them in actual time, minimizing risks and ensuring the robotic can retain to characteristic smoothly and securely in its targeted surroundings.

With the implementation of the ADS, TurtleBot can perform securely by constantly monitoring and responding to anomalous facts. This real-time anomaly detection presents an introduced layer of safety, ensuring that the robot stays resilient to external threats and internal malfunctions. By detecting deviations from the anticipated sensor facts, the ADS facilitates the robot make secure choices even in the presence of potentially risky changes to the information. Whether it’s malicious manipulation, sensor faults, or changes in the environment, the ADS lets in TurtleBot to respond as it should be, reducing the risk of injuries or gadget disasters. In turn, this device affords the important protection for TurtleBot to function autonomously with confidence, knowing that any deviations from normal conduct might be detected and addressed right away. This answer no longer best protects the robotics' operational integrity but additionally ensures its safe integration into IoT and IoRT ecosystems, where protection concerns are paramount.

IV. Methodology

The section below elaborates on the methodology adopted in building and evaluating an Anomaly Detection System for detecting malicious sensor data from IMU and LIDAR sources in TurtleBot. It outlines the procedure in steps: data collection, preprocessing, feature extraction, model training, evaluation, and deployment.

1. Creating a Simulated environment in ubuntu jellyfish 22.04 LTS

- Install ROS 2 Packages: Update your system and install the required TurtleBot3 and simulation packages with:

```
sudo apt update
sudo apt install ros-humble-turtlebot3 ros-humble-turtlebot3-simulations
```

- Set TurtleBot3 Model: Set the TurtleBot3 model to burger by running export TURTLEBOT3_MODEL=burger to ensure the simulation runs with the correct model.
- Launch the Simulation: Start the Gazebo simulation environment by launching the TurtleBot3 world with the command:

```
ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py
```

2. Data Collection and Preprocessing

The data was collected from TurtleBot in a simulated environment using ROS 2 and Gazebo. The main sensor data acquired includes:

- IMU Data: Measurement of the orientation, angular velocity, and linear acceleration of the robot. This was captured with the /imu topic node in ROS 2.
- LIDAR Data: Distance measurement of the surroundings of the robot using the /scan topic node.
- Velocity data: the robot's linear and angular velocities recorded from /cmd_vel topic node.

This data was captured using ROSbag and then exported into CSV files for easier manipulation. The IMU data had some data points' anomalies injected into it manually, whereas the LIDAR had labels marking a reading as normal or anomalous.

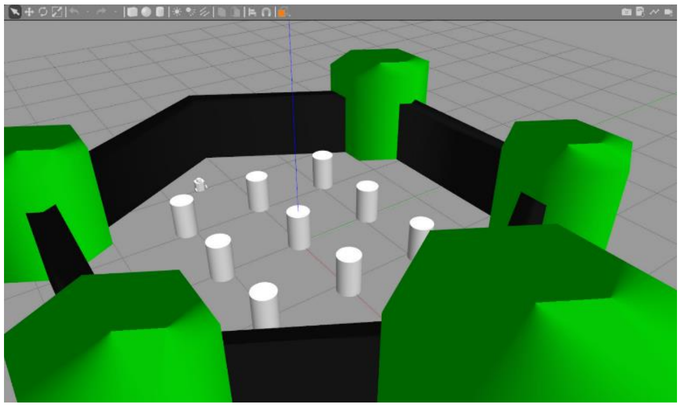
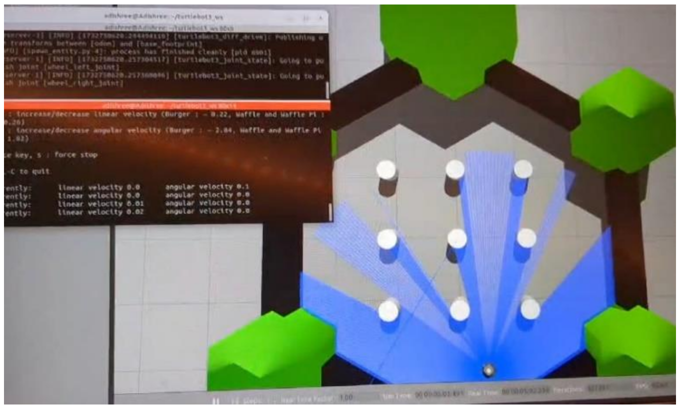


Fig 2. Gazebo simulation of turtlebot3 burger model



```
adishree@Adishree: ~/turtlebot3_ws
adishree@Adishree: ~/turtlebot3_ws 80x9
rserver-1] [INFO] [1732758620.244494119] [turtlebot3_diff_drive]: Publishing o
 transforms between [odom] and [base_footprint]
NFO] [spawn_entity.py-4]: process has finished cleanly [pid 6901]
server-1] [INFO] [1732758620.257304517] [turtlebot3_joint_state]: Going to pu
sh joint [wheel_left_joint]
server-1] [INFO] [1732758620.257360046] [turtlebot3_joint_state]: Going to pu
sh joint [wheel_right_joint]

 : increase/decrease linear velocity (Burger : ~ 0.22, Waffle and Waffle Pi :
0.26)
 : increase/decrease angular velocity (Burger : ~ 2.84, Waffle and Waffle Pi
 1.82)

e key, s : force stop

-C to quit

rently:      linear velocity 0.0      angular velocity 0.1
ently:       linear velocity 0.0      angular velocity 0.0
ently:       linear velocity 0.01     angular velocity 0.0
ently:       linear velocity 0.02     angular velocity 0.0
```

Fig 3 Gazebo simulation of LIDAR navigation

3. Feature Extraction

Feature extraction was done for both IMU and LIDAR data to convert raw sensor data into a format suitable for machine learning models.

IMU Features:

The dataset you've shared contains a range of sensor readings typically associated with a robotic system like a TurtleBot. Each entry represents a set of measurements across different time instances. The features provided are related to the robot's motion, orientation, and acceleration, and can be categorized as follows:

a) **Linear Velocity (cmd_vel_x, cmd_vel_y, cmd_vel_z)**
These represent the linear velocity components in the x, y, and z directions (usually in meters per second).
   - **cmd_vel_x**: Forward/backward motion.
   - **cmd_vel_y**: Lateral (sideways) motion.
   - **cmd_vel_z**: Vertical motion (often unused in typical TurtleBot configurations).

   Anomalies: Extremely high velocity in the x-direction at entry 3 (likely a command error or rapid motion).

b) **Angular Velocity (cmd_ang_x, cmd_ang_y, cmd_ang_z)**
These values indicate the rotational velocity around the x, y, and z axes (usually in radians per second).
   - **cmd_ang_x**: Rotation around the x-axis (roll).
   - **cmd_ang_y**: Rotation around the y-axis (pitch).
   - **cmd_ang_z**: Rotation around the z-axis (yaw).

   Anomalies: Extreme values at entry 5 (angular velocity around the x-axis, angular rotation around z-axis, etc.).

c) **Orientation (Quaternion) (orient_x, orient_y, orient_z, orient_w)**
Quaternions are used to represent orientation in 3D space to avoid singularities that can occur with Euler angles (roll, pitch, yaw). The components are x, y, z, w where w is the scalar part.

Anomalies: Entry 4 has an unusual quaternion indicating possible drift or calculation errors.

d) IMU (Inertial Measurement Unit)

- **Angular Velocity** (imu_ang_x, imu_ang_y, imu_ang_z):

  These represent the angular velocity measurements from the IMU in the x, y, and z axes (in radians per second).
  Anomalies: Unusually high angular velocity values (e.g., entry 5).

- **Acceleration** (imu_accel_x, imu_accel_y, imu_accel_z):

  These values represent linear acceleration measurements in the x, y, and z axes (in meters per second squared).
  Anomalies: Large spikes in acceleration (e.g., entry 3) and an unusually high vertical acceleration (entry 4).

- **Magnitude of Angular Velocity** (imu_ang_magnitude):

  Represents the combined magnitude of the angular velocity vector.
  Anomalies: Extremely high magnitude at entry 5 (likely an outlier or sudden spike).

- **Magnitude of Acceleration** (imu_accel_magnitude):

  Represents the combined magnitude of the linear acceleration vector.
  Anomalies: Large spikes in magnitude, especially at entries 3 and 4.

- **Delta Angles** (delta_roll, delta_pitch, delta_yaw):

  These represent the change in roll, pitch, and yaw over time (typically in radians). They can indicate the rate of change of orientation.
  Anomalies: Sharp changes in the roll, pitch, and yaw at entries 4 and 5, suggesting abrupt rotations.

e) Key Anomalies (added externally):

Explanation of some Anomalous Entry

- **Entry 3 (Spike in Velocity and Acceleration):**
  cmd_vel_x: A linear velocity of 10.0 (unusually high for normal operation).
  imu_accel_x, imu_accel_y: Extreme values like 50.0 and -50.0 indicate abnormal acceleration.
  This could represent a rapid unintended movement or a sensor error.
- **Entry 4 (Quaternion Drift and High Acceleration):**
  orient_x, orient_y, orient_z, orient_w: Values like [1.0, 1.0, 1.0, 0.0] are invalid because they violate the unit quaternion norm.
  imu_accel_z: A value of 50.0 indicates an unusual vertical acceleration, possibly due to a system malfunction.
- **Entry 5 (Extreme Angular Motion):**
  cmd_ang_x: A value of 10.0 for angular velocity is far beyond normal operation.
  imu_ang_x, imu_ang_y, imu_ang_z: Values like 20.0 indicate abnormally high angular velocities, which could signify rapid spinning or sensor miscalibration.
- **Entry 4 (Extreme Orientation Values):**
  roll, pitch, yaw: Values like 10.0 indicate extreme orientation changes, which could represent a sudden tilt or system instability.
- **Entry 5 (Sharp Changes in Orientation):**
  delta_roll, delta_pitch, delta_yaw: Sharp changes of 5.0 or -5.0 represent abrupt motions or system shock, which are unusual under normal operation.

LIDAR Features:
The features in the LiDAR dataset describe the robot's movement and the environment:

- **_cmd_vel_linear**: Robot's forward/backward speed. Important for correlating motion with LiDAR data.
- **_cmd_vel_angular**: Robot's rotational speed. Helps understand robot's turns and orientation.
- **_scan_ranges**: Raw LiDAR distance data. Critical for detecting obstacles and environment mapping.
- **_min_range**: Closest object distance. Indicates collision risk.
- **_max_range**: Furthest detectable distance. Shows open space or sensor limits.
- **_mean_range**: Average distance in scan. Indicates general environment density.
- **_variance_range**: Spread of scan distances. High variance means more complex surroundings.
- **_median_range**: Middle distance in scan. Less affected by outliers, gives robust summary.
- **_percentage_of_4**: Percentage of distances within 4 meters. Shows how crowded the environment is.
- Most important features: **_scan_ranges**, **_min_range**, **_max_range**, and **_percentage_of_4** for detecting obstacles and mapping the environment.

Testing anomalies:

- **Row 1**: The range values are mostly (infinite). This could indicate a sensor malfunction or a scenario where no obstacles are detected, which is unusual.
- **Row 2**: The values are very low, and the presence of 10 values could indicate the sensor is detecting very far distances or malfunctioning.
- **Row 3**: Normal but could be considered anomalous if the pattern significantly deviates from the rest of the normal data.
- **Row 4**: This row has **100%** 4 values, which could signify an extreme case of sensor malfunction or a scenario where nothing is detected at all (such as the sensor being obstructed or turned off).
- **Row 5**: The presence of 4 and a sudden increase or decrease in range values might suggest an anomaly, depending on your dataset's context.

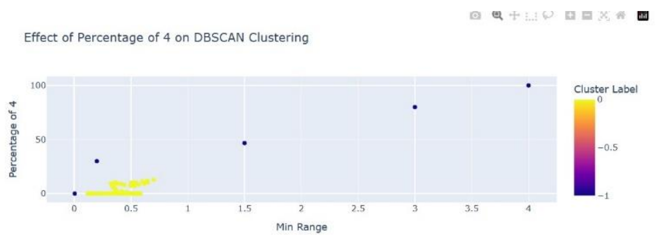
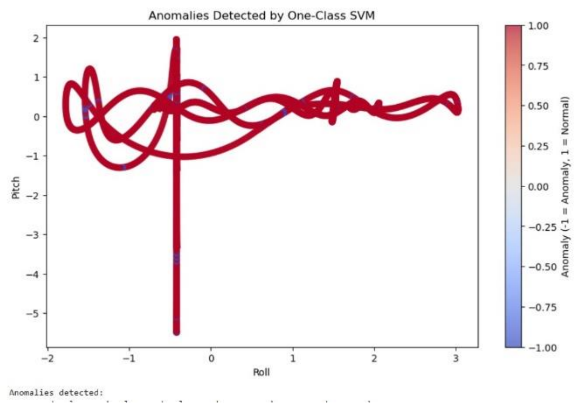


Fig.4 Anomaly detection of LIDAR with DBSCAN

4.Model Selection and Training

The following are some of the anomaly detection models that were used to detect anomalies in sensor data.

- Unsupervised Models: As labelled data was not always available, unsupervised models were used to detect anomalies.
- Isolation Forest: It is a tree-based model, which isolates anomalies by partitioning the data. It is efficient for high-dimensional data such as IMU and LIDAR features.
- Autoencoders: These are neural network-based methods for anomaly detection; the reconstruction error is considered an indication of an anomaly.
- Supervised Models: In cases where labelled data was available, supervised models like Random Forest and Support Vector Machines (SVM) were used to classify the data as normal or anomalous.



```
Anomalies detected:
```

Fig.5 Anomaly detection of IMU with one class svm

```
[63 rows x 26 columns]
```

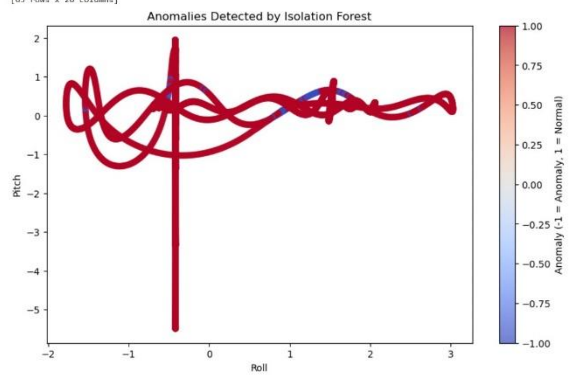


Fig.6 Anomaly detection of IMU with Isolation Forest

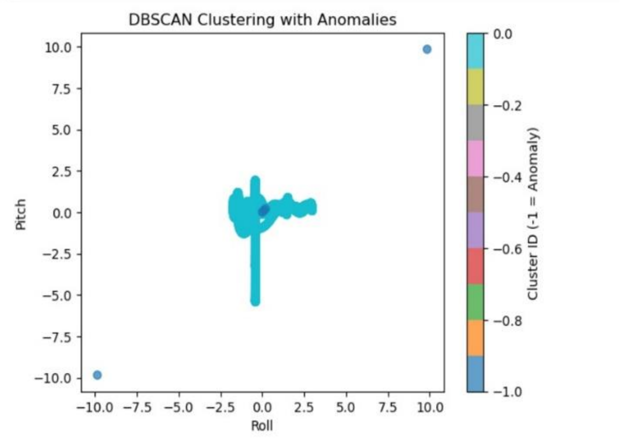


Fig.7 Anomaly detection of IMU with DBSCAN

5.Model evaluation

Standard metrics were used to evaluate the models' performance. Precision, Recall, and F1-Score in case of classification-based models, such as Random Forest. Reconstruction Error -unsupervised models, such as Autoencoders, consider high reconstruction error as indicative of an anomaly. Cross-validation and confusion matrices were used to evaluate model generalization and identify false positives and false negatives.

6. Deployment
After training and evaluating the models, the final anomaly detection system was deployed as a ROS 2 node. This node constantly listens for incoming IMU and LIDAR data, extracts feature from it and uses the trained model to detect anomalies in real-time.

Real-Time Monitoring: The result of anomaly detection is being published to a new ROS topic that is monitored while the operation of TurtleBot is performed.

Visualization: Tools such as RViz were used to visualize real-time detected anomalies for immediate feedback and understanding of how the robot interacts with its environment.

V. Observations

The integration of IMU and LIDAR data significantly enhanced anomaly detection by providing complementary information about the robot's movement and its environment, crucial in IoT ecosystems. In IoT environments, where multiple devices generate diverse sensor data, feature engineering—especially statistical measures like variance and mean—was key in identifying unusual sensor readings and ensuring the system's robustness in dynamic, interconnected settings.

VI. Conclusion

This project successfully developed an Anomaly Detection System (ADS) for TurtleBot, designed to hit upon malicious facts manipulation in real-time sensor readings from the robotics' IMU and LIDAR sensors. The main intention turned to ensure the robot’s secure and stable operation through identifying atypical styles in sensor facts that might indicate capability cyber-attacks, sensor malfunctions, or different anomalies. In this system, we utilized each unsupervised and supervised machine gaining knowledge of models to address the project of anomaly detection in dynamic environments in which the robotic operates autonomously. The models were educated to understand deviations from ordinary sensor conduct, for this reason enhancing TurtleBot's resilience towards statistics manipulation and ensuring that it can retain functioning successfully even under doubtlessly harmful situations.

The assignment was carried out the usage of ROS 2 (Robot Operating System) and Gazebo to facilitate statistics series, preprocessing, feature extraction, and version education. ROS 2 provided the important framework for interfacing with TurtleBot’s sensors, while Gazebo enabled realistic simulation of the robotic's environment, bearing in mind the collection of various sensor facts. Data preprocessing worried cleaning, synchronizing, and extracting relevant functions from the IMU and LIDAR sensors, such as orientation, angular speed, distance measurements, and velocity, which were then used to teach the anomaly detection models. These features have been essential in identifying ability irregularities in the statistics.

We skilled numerous anomaly detection fashions to make sure comprehensive insurance of various sorts of anomalies. These protected Isolation Forest, Autoencoders, and Random Forest. Isolation Forest and Autoencoders were used for unsupervised gaining knowledge of locating outliers in unlabeled information, even as Random Forest was employed for supervised anomaly detection, utilizing labeled datasets. Each of these fashions became evaluated based totally on its potential to pick out anomalies which include facts injections or sensor malfunctions. After rigorous testing, the fashions established first rate performance, with Isolation Forest and Autoencoders specifically excelling in detecting subtle anomalies, at the same time as Random Forest showed sturdy results in classifying acknowledged anomalies with high accuracy.

Finally, those educated models have been deployed as real-time ROS nodes to continuously reveal the sensor statistics from TurtleBot at some point of operation. The actual-time deployment ensured that any detected anomalies had been immediately flagged, allowing for set off corrective movements or in addition analysis. The ADS proved to be particularly powerful, imparting a further layer of security and making sure TurtleBot navigates and carries out tasks adequately, even within the presence of potential records manipulation or sensor errors. The mission correctly confirmed how systems gaining knowledge of models can be integrated into robotics to decorate security and operational reliability in actual-international IoT environments.

VII. Outlook

1. Real-World IoT/IoRT Deployment: Refining the system in dynamic, real-world IoT and IoRT environments will improve its ability to handle unpredictable data and sensor conditions, ensuring reliable operation in smart homes and industrial applications.
2. Multi-Sensor Fusion: Integrating additional sensors (e.g., cameras, GPS) in IoT and IoRT systems will provide richer data for anomaly detection, boosting reliability and decision-making.
3. Adaptive Detection: Developing models that adapt to evolving data patterns in IoT and IoRT environments will ensure continuous, effective anomaly detection in dynamic settings.

References

[1] Al-amri, R., Murugesan, R. K., Man, M., Abdulateef, A. F., Al-Sharafi, M. A., & Alkahtani, A. A. (2021). A review of machine learning and deep learning techniques for anomaly detection in IoT data. Applied Sciences, 11(12), 5320.

[2] Patel, Y., H Rughani, P., & Kumar Maiti, T. (2024). An examination of the security architecture and vulnerability exploitation of the turtlebot3 robotic system. International Journal of Computing and Digital Systems, 16(1), 1593-1602.

[3] Chalapathy, R., & Chawla, S. (2019). Deep learning for anomaly detection: A survey. arXiv preprint arXiv:1901.03407.

[4] He, Z., Chen, Y., Zhao, Y., Zhang, D., Liu, A., & Zhang, H. (2024). TSAN: A New Deep Learning-Based Detection Method for Sensor Anomaly in Mobile Robots. IEEE Transactions on Industrial Electronics.

[5] Goldstein, M.; Uchida, S. A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data. PLoS ONE 2016,11, e0152173