# Simulation-Generated Data for Fault Detection and Diagnosis in Mobile Robots

Abdalmotalib Alghoul[1], Alessandro Freddi[1], Andrea Monteriù[1]

***Abstract*— Fault detection and diagnosis is a crucial task for modern mobile robots, as it permits their correct functioning with a positive impact on availability, autonomy and safety. Despite several approaches of fault detection and diagnosis of sensor faults in mobile robots, there exists a lack of scientific studies addressing the low-cost LiDAR faults which are commonly adopted in indoor robots. This paper is a preliminary attempt to fill this research gap by proposing a data-driven approach for developing a diagnostic module for mobile robots, focusing on the specific case study of the Stretch robot.**

**Several LiDAR faults are first modeled, then simulated in ROS and Gazebo environments to generate a high-quality dataset. The dataset is used for training a machine learning model, which can diagnose such faults as well as distinguish them from other typical faults, such as IMU pose drift.**

**Our proposed model shows high detection and isolation accuracies across the injected fault scenarios, thus paving the way for the deployment on a real robotic system for further evaluation and analysis.**

## I. INTRODUCTION

Faults in mobile robots can lead to operational inaccuracies, degenerate into failures, increase costs, and eventually pose risks to people and the environment if not detected early during operation [1].

Accurate navigation is critical for robot movements, as it allows the robot to achieve the planned path and perform the required task at the desired position. When robot sensors produce excessive noise or drifting measurements, the control and navigation unit plans trajectories based on inaccurate sensory data, thus leading to navigation errors, repeated pose corrections, and possible task failures such as robot falls or collisions, with an overall increase of the task execution time [2], [3], [4].

Several diagnostic approaches can be found in the literature applied to mobile robots. Among them, Machine Learning (ML) first, and Deep Learning (DL) then are becoming more common, due to their ability in reversing the cause-effect fault-symptom relationship starting from sensory data. Feature engineering, model choice and training methods become critical in the data-driven workflow, as they permit to obtain a diagnostic model which can provide high performance metrics while generalizing to new data [5]. This is especially challenging in robotic applications, where the environment is complex, the sensor noise level is high, and the computational capabilities may be limited [4].

One common problem of data-driven fault diagnosis is that the availability and quality of data is critical. In real case applications, indeed, there is a noticeable unbalance between fault-free class data and faulty ones, as faulty data are much more difficult to obtain. An effective solution consists in using synthetic data originated from a simulated model; this solution, however, does not only require a good model of the kinematics and dynamics of the robotic system, but also of the fault to be diagnosed [6]. In this paper, we focus our attention on Light Detection and Ranging (LiDAR) sensor faults. The contributions of the paper are the following.

- We model the 2D LiDAR faults, together with the specific mobile robot, in ROS and Gazebo; such faults include dropout, Gaussian noise, beam Loss, and radial offset.
- We simulate the modeled fault to obtain a high-quality synthetic dataset for training a machine learning diagnostic classifier, which can diagnose such faults, as well as distinguish them from other faults common in mobile robots (such as the pose drift fault in Inertial Measurement Units (IMUs)).
- We publicly release the multi-fault model, together with the feature extraction code, with the aim of helping reproducibility and facilitating future works to expand our simulation model by integrating additional faults.

The rest of the article is structured as follows. Section II presents a study of related works in robot sensor Fault Detection and Diagnosis (FDD). Section III introduces the proposed fault model and defines the injected fault scenarios. Section IV describes the simulation setup used for data generation. Section V demonstrates the feature engineering process and dataset preparation for training the machine learning model. Section VI provides a discussion of the results, while Section VII concludes the paper.

## II. RELATED WORK

Indoor mobile robots rely on low-cost 2D LiDAR and IMU for localization and path planning. Therefore, FDD of these sensors is critical to ensuring reliable autonomous operation. In recent years, many fault detection approaches have been studied, and they are commonly divided into model-based and data-driven.

As for the model-based approaches, the kinematics and dynamics equations of the system play an essential role in designing the diagnostic model [3], [4]. Model-based Fault Detection (FD) methods generate residuals by comparing the predicted output of the models with the actual sensor output. Stavrou et al. [7] proposed a model-based approach to detect and isolate actuator faults in differential-drive service mobile robots operating in an indoor environment. They

[1]Department of Information Engineering, Università Politecnica delle Marche, Ancona, Italy, {a.alghoul}@pm.univpm.it, {a.freddi, a.monteriu}@univpm.it

used residual generation from kinematic models. Yan et al. [8] presented a fault detection algorithm that addresses Gaussian mixture noise in an EKF-based LiDAR/IMU localization system. Khalastchi and Kalech [1] surveyed model-based residual generation and data-driven methods for fault detection in multi-robot systems. Fault detection by using residuals is commonly straightforward, where fault isolation and possible identification typically require more complex approaches, such as bank of observers [9] and disturbance observers [10]. The performance of model-based FDD is highly dependent on the accuracy of the models and the available low-cost sensors, which are subject to noise, drift, and varying environmental conditions [11].

Data-driven approaches, in contrast, learn fault patterns directly from sensor data without pre-designed models of the robot hardware or environment. Surveys of machine learning techniques for mobile robot control have reported the rapidly growing use of data-driven methods for classification and their potential to improve robot performance [12]. El Mawas et al. [13] used decision trees and random forests for fault detection and isolation in multi-robot cooperative localization. Their model detects localization data bias, gyroscope drift, wheel encoder drift, absence of LiDAR reflection information, and beam intensity bias. Jiang [14] proposed fault classification and diagnosis method for indoor substation inspection based on Support Vector Machine (SVM). The authors reported improved performance using an SVM-based model. Jeong and Lee [15] proposed a method based on Convolutional Neural Networks (CNN) to detect LiDAR scan faults to improve the accuracy of Simultaneous Localization and Mapping (SLAM) in dynamic environments. Yin et al. [16] employed feature extraction and a logistic regression model to enhance the accuracy of failure detection in 3D LiDAR-based localization. Brito et al. [17] presented a deep learning model that determines the pose, type, and dimensions of 2D primitives under high and low noise conditions. Guo et al. [18] proposed a hybrid algorithm that combines model-based and deep learning methods to reduce the inertial information noise from low-cost sensors. Miao et al. [11] used a channel-wise CNN with feature augmentation and time-window-sliding segmentation for FDD in wheeled mobile robots. Matos et al. [19] developed a fault-injection framework in CARLA/Autoware for autonomous vehicles, targeting outdoor AV scenarios and various fault types.

However, while El Mawas et al. addressed LiDAR, IMU, and other sensors, albeit with different fault types, these studies focused on faults different from the ones presented in this paper, namely beam-level faults in low-cost 2D LiDAR (Dropout, Gaussian noise, Beam Loss, and Radial Offset) combined with IMU-induced pose drift in indoor mobile robots.

## III. Proposed Fault Injection

The proposed fault injection framework simulates five independent sensor faults by modifying the relevant ROS topics during robot navigation. Four faults (Gaussian noise, beam dropout, beam loss, and radial offset) are injected into the LiDAR scan beams published on the `/scan` topic. The fifth fault simulates an IMU acceleration bias along the $x-$axis, that induces odometry pose drift; this bias is injected directly into the `Odom` $\rightarrow$ `base_link` transform in the TF tree.

### A. Gaussian Noise

Gaussian noise is added to the scan measurements (2000 beams ) received from the `/scan` topic. The faulty readings are then republished to the AMCL node. The Gaussian noise is injected according to the following equations:

$$r_{i,GAU} = r_i + n_i \tag{1}$$

$$n_i \sim \mathcal{N}(0, \sigma^2) \tag{2}$$

where $r_{i,GAU}$ is the noisy range value, $r_i$ is the original clean range value, $n_i$ is the additive Gaussian noise sample, and $\sigma$ is the standard deviation of the noise.

### B. Beam Dropout

To simulate beam dropout, individual beam measurements are replaced by a large value ($r_\infty$) with probability $p$:

$$r_{i,DRP} = \begin{cases} r_\infty & \text{with probability } p \\ r_i & \text{with probability } 1-p \end{cases} \tag{3}$$

where $r_{i,DRP}$ is the final (possibly dropped) range value, $r_i$ is original clean range value, $p$ is the dropout probability, and $r_\infty$ is the dropped measurement.

Later in the data preparation stage, the $r_\infty$ values were replaced by outrange values (i.e. 20 m) to avoid computational issues during the feature extraction.

### C. Beam Loss

The Beam loss fault is injected by replacing a contiguous block of $k$ consecutive beams with an out of range value ($r_\infty$) exceeding the sensor's `range_max`:

$$r_{i,LOS} = \begin{cases} r_\infty & \text{if } s \le i < e \\ r_i & \text{otherwise} \end{cases} \tag{4}$$

where $r_\infty$ is the replacement value (e.g., 20.0 m), $s, e$ are start and end indices of the block, and $i = e-s$ is the number of affected beams.

Later in the data preparation stage, the $r_\infty$ values were replaced by outrange values (i.e. 20 m) to avoid computational issues during the feature extraction.

### D. Radial Offset

The LiDAR radial range offset is injected along each beam measurement, considering the beam angle to produce a realistic directional offset:

$$r_{i,OFF} = r_i + d_0 \cos(\theta_i) \tag{5}$$

where $r_i$ is the original range value, $r_{i,OFF}$ is the final modified range value, $d_0$ is the distance offset, and $\theta_i$ is the beam angle.

### E. IMU-Induced Pose Drift

In the Hello Robot Stretch platform, which is used as testing platform as described in Section IV, the IMU readings are not integrated into the default odometry estimation. As a consequence, the pose drift induced by IMU acceleration bias along the $x-$axis is injected directly into the `odom→base_link` transform (TF):

$$\Delta x_{\text{drift}}(t) = \frac{1}{2} a_x t^2 \tag{6}$$

The drifted coordinates are then calculated as:

$$x' = x + \Delta x_{\text{drift}}(t) \cdot \cos(\psi) \tag{7}$$

$$y' = y + \Delta x_{\text{drift}}(t) \cdot \sin(\psi) \tag{8}$$

where $a_x$ is the acceleration bias (m/s²), $t$ is the elapsed time (s), and $\psi$ is the original yaw angle

## IV. Experimental Setup

The mobile platform considered in this study is the Stretch V2, a mobile manipulation robot developed by Hello Robotics [20]. It features a wheeled mobile base, a lightweight telescoping arm, and a dexterous gripper with an integrated camera. The robot has a well-supported digital twin in Gazebo Simulator, which was used to implement the the fault injection framework using the Robot Operating System (ROS Noetic) with Gazebo v.11 and the Hello Stretch Navigation package as shown in Figure 1. In detail, 4 dedicated Python injector nodes (one node jointly handling Gaussian noise and radial offset) were developed together with an additional autonomous navigation node.

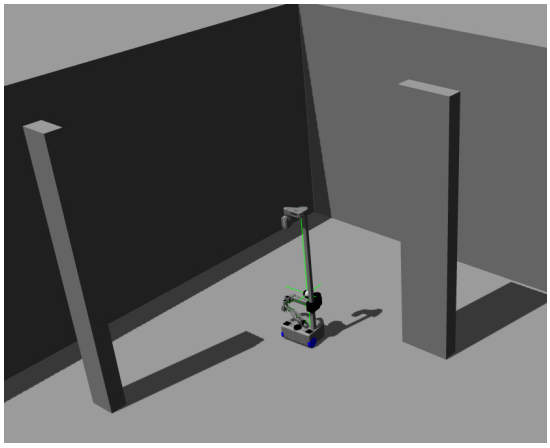

Fig. 1: Hello Stretch Robot v2 in the Gazebo Simulation Environment.

The simulation procedure adopted for generating the data is as follows.

1) The navigation node is launched first, enabling the robot to perform recursive goal-directed navigation toward three randomly selected destinations inside a $6\,\text{m} \times 8\,\text{m}$ room containing three static obstacles.
2) One fault injection node is activated (only one per trial), which subscribes to either the LiDAR scan topic (`/scan`) or the TF tree (the navigation package publishes odometry exclusively via TF). Upon receiving a message, the node injects the corresponding fault according to the parameters defined in YAML configuration files and immediately republishes the corrupted data. In the case of LiDAR faults, the injection is conducted on 2000 beams, which are subsequently downsampled to 350 beams during the saving process before being stored in the final dataset.
3) The AMCL node processes these faulty measurements and supplies updated pose estimates to the navigation stack, which then generates the control commands.
4) The data are logged and stored into a database, which is later used for training the diagnostic ML model.

To ensure reproducible fault conditions, the following scenarios were adopted for each fault type:

**LiDAR anomalous gaussian noise,** one fault scenario with a noise standard deviation $\sigma = 0.056\ m$;
**LiDAR radial offset,** one fault scenario with an offset $d_0 = 20\ cm$;
**LiDAR dropout,** four different fault scenarios as described in Table I;
**LiDAR beam loss,** three different fault scenarios as described in Table II;
**IMU pose drift,** one fault scenario with an acceleration bias $a_x = 8.7 \cdot 10^{-5}\ m/s^2$.

| Scenario | Mode | Interval (s) | Duration (s) | Prob. |
|---|---|---|---|---|
| Low dropout | periodic | 1.0 | 5.0 | 0.07 |
| Medium dropout | periodic | 4.0 | 7.0 | 0.10 |
| High dropout | periodic | 9.0 | 10.0 | 0.10 |
| Full dropout | periodic | 10.0 | 5.0 | 1.00 |

TABLE I: Dropout scenarios used in the simulations

| Scenario | Percentage (%) |
|---|---|
| Low Beam Loss | 2.0 |
| Medium Beam Loss | 4.0 |
| High Beam Loss | 6.0 |

TABLE II: Beam loss values used in the simulations

Each simulation run had a duration of approximately 250 seconds, during which a single fault was injected.

## V. Feature Engineering and Dataset Preparation

Different simulation runs were conducted to collect a comprehensive dataset including:

- LiDAR scan ranges per beam at an angular resolution of $1.028°$;
- AMCL pose estimations of the robot ($x$, $y$ and $\psi$);
- TF poses;
- Particle Filter statistics;
- commanded angular velocities for the two wheels.

The main dataset contains 165 052 samples and was cleaned by removing NaN and Inf values. From the cleaned raw data, a total of 303 statistical, geometric, residual, and

fault-sensitive features were initially extracted by using a sliding windows of 6 s duration (60 samples with 50 % overlap). This window length was chosen to enable online FDD while fully capturing the temporal signatures of all injected faults (Gaussian noise, beam dropout, beam loss, radial offset, and IMU-induced TF-pose drift).

The features were then further reduced by removing constant features and highly correlated features, namely with a Pearson correlation threshold exceeding 0.95. Finally, the 50 most discriminative features were selected using XGBoost feature-importance ranking, ordered by their contribution to multi-fault classification. These 303 extracted features can be grouped into the following four categories:

- **LiDAR beam statistics and geometry** (spatial, distributional, normalized, and centroid-based descriptors, particularly effective for dropout, beam loss, and radial offset faults)
- **Pose, localization, and uncertainty measures** (TF odometry kinematics, AMCL estimates, particle filter statistics, covariance trace, and confidence indicators) ;
- **Sensor disagreement, residuals, and drift dynamics** (TF–AMCL pose differences, Euclidean distances, drift rates, cumulative bias, and CUSUM statistics)
- **Command-response consistency and advanced temporal diagnostics** (velocity/yaw residuals, correlations, autocorrelation, slopes)

Representative examples from each category are given below, while the complete mathematical definitions of the 49 selected features are provided in the public repository accompanying this paper.

For LiDAR beam statistics and geometry, the normalized mean of all beam ranges is

$$\text{lidar\_normalized\_mean} = \frac{1}{N}\sum_{i=1}^{N}\frac{r_i - \bar{r}}{\sigma_r + \epsilon}, \tag{9}$$

where $r_i$ is the range value of beam $i$, $\bar{r}$ is the mean range, $\sigma_r$ is the standard deviation of the ranges, and $\epsilon$ is a small constant.

For pose, localization, and uncertainty measures, the mean translational speed derived from TF odometry is

$$\text{tf\_speed\_mean\_expert} = \frac{1}{N}\sum_{i=1}^{N}\sqrt{(\Delta x_i)^2 + (\Delta y_i)^2}. \tag{10}$$

For sensor disagreement, residuals, and drift, the mean Euclidean distance between TF and AMCL poses is

$$\text{tf\_amcl\_dist\_mean} = \frac{1}{N}\sum_{i=1}^{N}\sqrt{(x_{\text{tf},i} - x_{\text{amcl},i})^2 + (y_{\text{tf},i} - y_{\text{amcl},i})^2}. \tag{11}$$

For command-response consistency and temporal diagnostics, the mean residual between commanded and observed yaw change is

$$\text{tf\_cmd\_yaw\_residual\_mean} = \frac{1}{N}\sum_{i=1}^{N}(\Delta\psi_{\text{tf},i} - \Delta\psi_{\text{cmd},i}). \tag{12}$$

To interpret model decisions, SHAP (SHapley Additive exPlanations) values were computed on the selected features. Figure 2 shows the SHAP feature importance ranking for the dataset, computed on 1000 stratified test samples of the XGBoost classifier, defined in Section VI. The global SHAP analysis reveals that LiDAR-related features dominate the model decisions. The two most influential features are `lidar_normalized_entropy` and `lidar_roughness` (both with mean absolute SHAP = 0.742), followed by `lidar_pca_var_exp_pc2` (0.458), `particle_uncertainty_mean_expert` (0.409), and `tf_speed_mean_expert` (0.237).

Per-class SHAP analysis further confirms that `lidar_normalized_entropy` and `lidar_roughness` are consistently among the top contributors across all fault classes, particularly for beam-related faults. High entropy strongly signals Gaussian Noise due to increased randomness in range distribution, while roughness effectively captures abrupt discontinuities caused by Beam_Loss and localized anomalies from Dropout and radial offset. The features `lidar_pca_var_exp_pc2` and `particle_uncertainty_mean_expert` are highly discriminative for Dropout, radial offset, and IMU-induced drift, as they quantify variance in beam structure and degradation in localization confidence, respectively. The kinematic feature `tf_speed_mean_expert` contributes strongly to detecting motion inconsistencies under pose drift. Overall, the SHAP analysis confirms that the proposed causal feature engineering pipeline successfully prioritizes physically meaningful indicators of sensor degradation and localization uncertainty.

The complete fault injection framework, feature extraction pipeline, trained models, and datasets generated in this study are publicly available at: `https://github.com/Abdalmotalib-Alghoul/Mobile-Robot-Multi-Fault-Detection`

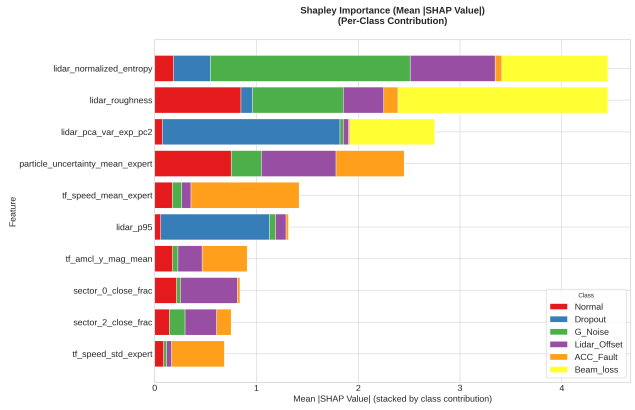


Fig. 2: SHAP feature importance (mean absolute SHAP values) for the multi-fault classifier.

## VI. Results

This section evaluates the performance of the proposed multi-fault classification approach. Four classifiers were compared: XGBoost, Optimized Ensemble (AdaBoost), Artificial Neural Network (ANN), and SVM with cubic kernel. Evaluation was performed on six classes: Normal, Dropout,

Gaussian Noise (G_Noise), LiDAR Offset (radial offset), ACC_Fault (IMU-induced pose drift), and Beam_Loss.

The training was performed using a 5-fold nested cross-validation with Optuna hyperparameter optimization (40 trials per fold) and early stopping. For each fold, the final model was trained on 90 % of the training data, with early stopping monitored on the remaining 10 %. Performance was assessed on an external hold-out test set (20 % of the total data). The primary metric computed are the macro-averaged F1-score, supplemented by per-class precision, recall, F1-score, overall accuracy, and confusion matrices. Class weights were applied to address the imbalance and improve recall in the Normal class (2.0×).

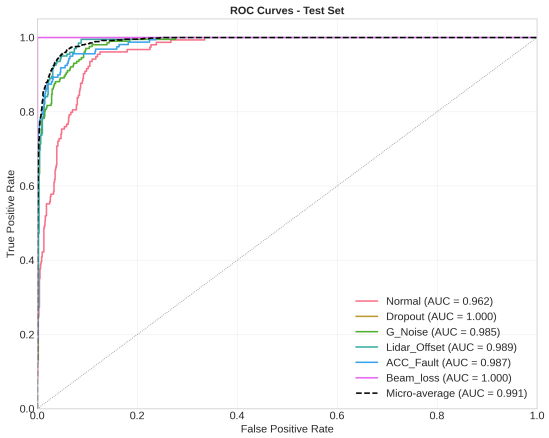


Fig. 3: ROC curves for the test set.

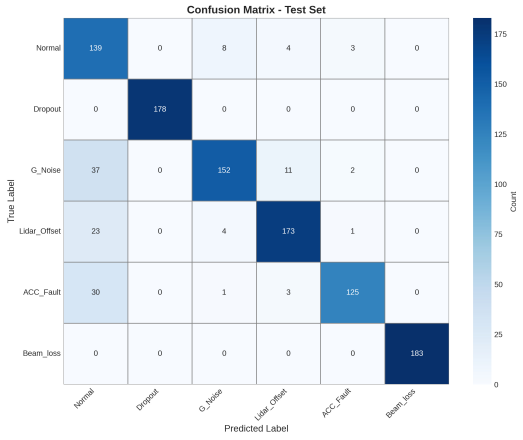


Fig. 4: Confusion matrix for the test set.

### A. Best Performing Classifier - XGBoost

The data set contained 4307 training samples (3876 for hyperparameter optimization and cross-validation, 431 for early stopping) and 1077 test samples. Using the top 49 features, the optimized XGBoost model (Normal-class weight multiplier 2.0) achieved a macro-averaged F1-score of 89 % on the external test set. The ROC curve and the confusion matrix for the test set are shown in Figures 3 and 4 respectively, while the quantitative metrics are reported in Table III.

| Class | Precision | Recall | F1-score |
|---|---|---|---|
| Normal | 0.69 | 0.81 | 0.74 |
| Dropout | 1.00 | 1.00 | 1.00 |
| G_Noise | 0.86 | 0.83 | 0.85 |
| Lidar_Offset | 0.88 | 0.90 | 0.89 |
| ACC_Fault | 0.93 | 0.80 | 0.86 |
| Beam_Loss | 1.00 | 1.00 | 1.00 |
| **Macro Avg** | **0.89** | **0.89** | **0.89** |

TABLE III: Classification metrics.

### B. Classifiers Comparison and Sensitivity to LiDAR Offset Magnitude

Table IV compares the XGBoost model against Optimizable Ensemble (AdaBoost), SVM (cubic kernel), and ANN on identical feature sets. Additional datasets were also created using the same procedure described in Section IV, but this time injecting separately a LiDAR radial offset of $d_0 = 10\ cm$, $d_0 = 5\ cm$, and $d_0 = 2.5\ cm$. These additional datasets were used to provide a preliminary evaluation of the FDD module's performance under fault magnitudes different from those used in the main dataset. Table V compares the performance of XGBoost with the different magnitudes of the radial offset.

| Offset | Model | Test Macro F1 (%) |
|---|---|---|
| 20 cm | Proposed XGBoost | 89 |
| 20 cm | Ensemble (AdaBoost) | 88.95 |
| 20 cm | SVM (Cubic) | 86.82 |
| 20 cm | ANN | 81.99 |

TABLE IV: Performance and sensitivity comparison.

| Class | 20 cm | 10 cm | 5 cm | 2.5 cm |
|---|---|---|---|---|
| Normal | 0.74 | 0.66 | 0.60 | 0.54 |
| Dropout | 1.00 | 1.00 | 1.00 | 1.00 |
| G_Noise | 0.85 | 0.77 | 0.79 | 0.81 |
| Lidar_Offset | 0.89 | 0.67 | 0.68 | 0.47 |
| ACC_Fault | 0.86 | 0.88 | 0.78 | 0.80 |
| Beam_Loss | 1.00 | 1.00 | 1.00 | 1.00 |
| **Macro Avg** | **0.89** | **0.83** | **0.81** | **0.77** |

TABLE V: F1-Scores for different radial offset magnitudes.

### C. Discussion

The experimental results demonstrate that the proposed fault injection framework and causal feature engineering pipeline enable effective multi-fault classification in a simulated ROS/Gazebo-based mobile robot navigation system. On the 20 cm radial offset dataset, the optimized XGBoost classifier achieved a macro-averaged F1-score of 89 %. The classification performance was perfect for Dropout and Beam_Loss faults (F1-score = 1.00 in both cases). Decreasing the magnitude of the radial offset from 20 cm to 2.5 cm worsened the separability of the classes, leading to a 77 % macro-average F1-score. The main source of error remained partial signature overlap among the Normal, Gaussian Noise, and LiDAR Offset classes, as visible in the ROC curves (Figure 3) and confusion matrix (Figure 4).

SHAP analysis confirmed that the most influential features were physically meaningful and directly related to the injected faults. LiDAR beam degradation indicators (`lidar_normalized_entropy`, `lidar_roughness`, and `lidar_pca_var_exp_pc2`) and localization uncertainty measures (`particle_uncertainty_mean_expert`) consistently ranked highest (Figure 2). These findings align well with the expected physical effects of the injected sensor faults. In direct comparison with benchmark classifiers on identical feature sets, the proposed XGBoost model performed competitively, achieving results on par with the Optimized Ensemble (AdaBoost) while outperforming SVM and ANN.

All experiments were performed in simulation and have not yet been validated on physical hardware; as such, several limitations remain. Real-world sensor noise, dynamic obstacles, and environmental changes may affect performance. To validate on the physical robot, we will first characterize the baseline noise of the real LiDAR and IMU under nominal operation, then replicate the injected fault magnitudes on the physical hardware to assess whether the classifier generalizes to real sensor corruption.Moreover, the current study injected only single faults per navigation run; in future works, we will add selected realistic compound classes (e.g., LiDAR radial offset with IMU drift). Additionally, the fault magnitudes used in the simulations were limited to specific discrete values. An extended validation involving variable fault magnitudes together with sensitivity analysis could provide a better understanding of the robustness of the proposed workflow. Finally, considering the availability of reliable models for mobile robots, the addition of physics-informed (model-based) could provide a boost to performance. All these aspects are currently under investigation.

## VII. Conclusion

This paper presented a reproducible framework for fault detection and diagnosis (FDD) in low-cost sensor-based mobile robot navigation. Five realistic faults, Gaussian noise, beam dropout, beam loss, radial offset, and IMU-induced pose drift were systematically injected into a ROS Noetic/Gazebo simulation of the Hello Stretch robot. A sliding-window feature engineering pipeline extracted 303 statistical, geometric, residual, and drift-sensitive features, which were subsequently reduced to the 49 most discriminative ones via XGBoost feature importance ranking.

The optimized XGBoost classifier achieved a macro-averaged F1-score of 89%, attaining perfect classification (F1 = 1.00) for the beam dropout and beam loss faults. SHAP interpretability analysis verified that the selected features are physically meaningful and consistent with the expected manifestations of the injected sensor faults. Performance evaluation was conducted on a simulation-generated dataset using 5-fold nested cross-validation and Optuna hyperparameter optimization. Comparative analysis against strong baselines (AdaBoost, SVM, and ANN) confirmed the superior performance of the proposed XGBoost model.

Overall, this study contributes to an interpretable, and practical methodology to the field of FDD in autonomous robotics, supporting the development of safer mobile robots.


## Acknowledgment

The scripts used in this work as well as the paper writing language were enhanced with the assistance of AI tools.


## References


[1] E. Khalastchi and M. Kalech, "Fault detection and diagnosis in multi-robot systems: A survey," *Sensors*, vol. 19, no. 18, p. 4019, 2019.

[2] P. Yan, Z. Li, F. Huang, W. Wen, and L.-T. Hsu, "Fault detection algorithm for gaussian mixture noises: An application in lidar/imu integrated localization systems," *NAVIGATION: Journal of the Institute of Navigation*, vol. 72, no. 1, 2025.

[3] B. Patle, A. Pandey, D. Parhi, A. Jagadeesh, *et al.*, "A review: On path planning strategies for navigation of mobile robot," *Defence Technology*, vol. 15, no. 4, pp. 582–606, 2019.

[4] G. Jombo, E. Zhang, and N. Lu, "Sensor fault detection and diagnosis: Methods and challenges," in *IFAC Proceedings*, 2024.

[5] Y. Zhang *et al.*, "Fault types and diagnostic methods of manipulator robots: A review," *Sensors*, vol. 25, no. 6, p. 1716, 2025.

[6] A. Khan, H. Hwang, and H. S. Kim, "Synthetic data augmentation and deep learning for the fault diagnosis of rotating machines," *Mathematics*, vol. 9, no. 18, 2021.

[7] D. Stavrou, D. G. Eliades, C. G. Panayiotou, and M. M. Polycarpou, "Fault detection for service mobile robots using model-based method," *Autonomous Robots*, vol. 40, no. 2, pp. 383–394, Feb. 2016.

[8] P. Yan, Z. Li, F. Huang, W. Wen, and L.-T. Hsu, "Fault detection algorithm for gaussian mixture noises: An application in lidar/imu integrated localization systems," *NAVIGATION: Journal of the Institute of Navigation*, vol. 72, no. 1, 2025.

[9] A. Baldini, R. Felicetti, A. Freddi, S. Longhi, and A. Monteriù, "Hexarotor fault tolerant control using a bank of disturbance observers," in *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2022, pp. 608–616.

[10] ——, "Disturbance observer based fault tolerant control for tilted hexarotors," in *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2021, pp. 20–27.

[11] Z. Miao, F. Zhou, X. Yuan, Y. Xia, and K. Chen, "Multi-heterogeneous sensor data fusion method via convolutional neural network for fault diagnosis of wheeled mobile robot," *Applied Soft Computing*, vol. 129, p. 109554, 2022.

[12] M. Rybczak, N. Popowniak, and A. Lazarowska, "A survey of machine learning approaches for mobile robot control," *Robotics*, vol. 13, no. 1, p. 12, 2024.

[13] Z. El Mawas, C. Cappelle, and M. E. B. El Najjar, "Decision tree based diagnosis for hybrid model-based/data-driven fault detection and exclusion of a decentralized multi-vehicle cooperative localization system," in *IFAC-PapersOnLine*, vol. 56, no. 2, 2023, pp. 7740–7745.

[14] Y. Jiang, "Fault diagnosis technology of substation indoor inspection robot based on svm," in *2021 7th International Symposium on Mechatronics and Industrial Informatics (ISMII)*, 2021, pp. 115–118.

[15] H. Jeong and H. Lee, "Cnn-based fault detection of scan matching for accurate slam in dynamic environments," *Sensors*, vol. 23, no. 6, p. 2940, 2023.

[16] H. Yin, L. Tang, X. Ding, Y. Wang, and R. Xiong, "A failure detection method for 3d lidar based localization," in *2019 Chinese Automation Congress (CAC)*, 2019, pp. 1234–1239.

[17] A. Brito, P. Sousa, A. Couto, G. Leão, L. P. Reis, and A. Sousa, "Using deep learning for 2d primitive perception with a noisy robotic lidar," in *2024 7th Iberian Robotics Conference (ROBOT)*, 2024.

[18] F. Guo *et al.*, "Model-based deep learning for low-cost imu dead-reckoning of wheeled mobile robot," *IEEE Transactions on Industrial Electronics*, vol. 71, no. 7, pp. 7531–7541, 2024.

[19] F. Matos, J. Durães, and J. Cunha, "Simulating the effects of sensor failures on autonomous vehicles for safety evaluation," *Informatics*, vol. 12, no. 3, 2025.

[20] V. Ranganeni, V. Nguyen, H. Evans, J. Evans, J. Mehu, S. Olatunji, W. Rogers, A. Edsinger, C. Kemp, and M. Cakmak, "Robots for humanity: In-home deployment of stretch re2," in *Companion Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, New York, NY, USA, 2024, p. 1299–1301.