Evaluation of Classifier Architectures and Ultrasound Features for Robotic Wall-Following Navigation

Burra Shanthakumari[1], Namoju Vinja[2], Polusani Susritha[2], Sinduri Srivishnu Shashipreetham[2], Yata Saiteja[2]

Department of Computer Science and Engineering (Data Science), Vaagdevi College of Engineering, Warangal, Telangana, India

*Corresponding Email: shanthak27@gmail.com

ABSTRACT

Wall-following navigation is an essential capability for autonomous robots in structured environments such as warehouses, factories, and homes. This research enhances both the accuracy and reliability of wall-following by evaluating and optimizing machine learning classifiers built on ultrasound sensor features. Traditional approaches—rule-based algorithms coupled with basic ultrasonic sensors and fixed thresholds—often lack the precision, adaptability, and real-time performance required for effective navigation in dynamic settings. Our approach leverages advanced machine learning techniques to process ultrasound data, extracting features such as distance measurements and echo patterns to deliver high-quality inputs for classification. We implement and compare two classifiers—Decision Tree (DTC) and K-Nearest Neighbors (KNN)—using predefined movement categories ("Slight-Left-Turn," "Move-Forward," "Sharp-Right-Turn," and "Slight-Right-Turn") to govern the robot's actions. Extensive testing and validation, assessed via confusion matrices, demonstrate that these classifiers significantly improve obstacle detection, wall-following accuracy, and robustness across varied environments. By identifying the optimal classifier architecture, this system aims to optimize autonomous navigation and strengthen real-time decision-making in robotic applications.

Keywords: Autonomous navigation, wall following robots, Ultrasound sensor features, Supervised learning, KNN classifier, DTC model.

1. INTRODUCTION

Robotic wall-following navigation has undergone significant evolution over the past decades, driven by advancements in sensors, algorithms, and computational power. In the early 2000s, robots primarily relied on simple rule-based systems and basic proximity sensors for wall-following tasks. For instance, early systems used fixed ultrasonic thresholds to determine distance and adjust navigation accordingly. However, these methods often struggled with dynamic obstacles and varying environmental conditions. By the mid-2010s, more sophisticated techniques began to emerge, including adaptive algorithms and improved sensor technologies. The introduction of machine learning into robotics around 2015 marked a turning point, as it allowed for more nuanced interpretations of sensor data and better handling of complex navigation scenarios. Recent years have seen rapid advancements in machine learning and sensor technologies, particularly with the integration of deep learning models and advanced ultrasound sensors. For example, in 2020, researchers demonstrated that machine learning classifiers significantly improve navigation accuracy and adaptability in dynamic environments by effectively processing ultrasound data. This period also witnessed the deployment of robots in diverse applications, such as autonomous delivery systems and industrial automation, showcasing the practical benefits of enhanced navigation systems. These advancements underscore the importance of evaluating and optimizing classifier architectures to leverage the full potential of modern ultrasonic sensing capabilities.

The motivation behind enhancing robotic wall-following navigation stems from the need for autonomous robots to operate efficiently and safely in a variety of environments. Traditional navigation

doi: 10.48047/ijiee.2025.15.4.16

systems, such as those relying on rule-based algorithms and fixed ultrasonic thresholds, often fall short in dynamic settings where obstacles and environmental conditions can change rapidly. For example, rule-based systems struggle with detecting and responding to unforeseen obstacles or varying wall textures, leading to suboptimal navigation performance and increased risk of collision.

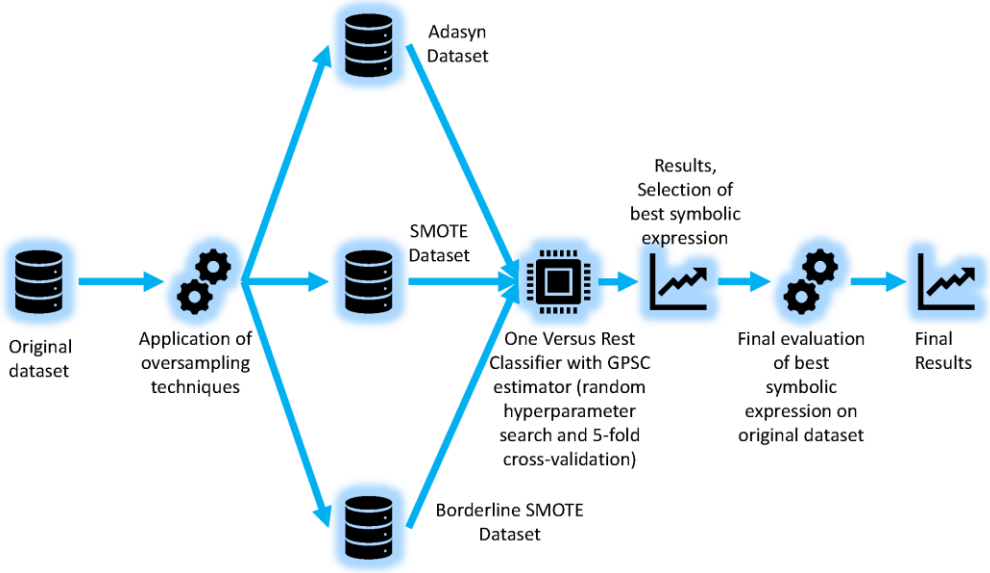

Figure 1: Common working flow.

Machine learning techniques offer a promising solution by providing a more flexible and adaptive approach to processing sensor data. These techniques can learn from vast amounts of data, enabling robots to recognize complex patterns and make more informed decisions in real-time. This adaptability is crucial for applications where precision and reliability are paramount, such as in autonomous warehouse robots or home assistance devices. By leveraging advanced classifier architectures and ultrasound features, the proposed approach aims to overcome the limitations of traditional methods, resulting in more accurate, reliable, and versatile wall-following navigation.

2. LITERATURE SURVEY

Varma, N., Poornima, V., Aivek, and Ravikumar Pandi. [1] proposed an intelligent wall-following control system for differential drive mobile robots. They utilized various machine learning models to improve the accuracy of wall-following, target tracking, and obstacle avoidance. Their study leveraged a dataset from SCITOS G5 mobile robots, incorporating 24 ultrasound sensors to capture detailed navigation data. The research demonstrated significant advancements in wall-following algorithms by comparing the performance of different machine learning models and showing improved accuracy compared to previous methods. Freire, Ananda L., et al. [2] conducted a study on short-term memory mechanisms in neural network learning for robot navigation tasks. They focused on utilizing neural network designs, such as Multi-Layer Perceptron (MLP) and Elman Recurrent Network, to handle non-linearly separable problems in wall-following tasks. Their work involved using a dataset of 24 ultrasound sensor readings to develop models that navigate more effectively, highlighting the effectiveness of neural networks in solving complex navigation problems.

Juang, Chia-Feng, Ying-Han Chen, and Yue-Hua Jhan. [3] presented a wall-following control method for a hexapod robot using a data-driven fuzzy controller learned through differential evolution. Their approach aimed to enhance navigation accuracy by incorporating fuzzy logic and evolutionary algorithms. They utilized ultrasound sensor data to train the controller and demonstrated its

doi: 10.48047/ijiee.2025.15.4.16

effectiveness in improving the wall-following capabilities of hexapod robots. Wall-Following Robot Navigation Data Dataset. [4] provided a comprehensive dataset for wall-following robot navigation, including readings from 24 ultrasound sensors. This dataset, captured using SCITOS G5 mobile robots, was used to evaluate various control models. It includes three formats: full 24 sensors, 4 sensors, and 2 sensors per record. The dataset was instrumental in comparing the performance of different machine learning models, including Decision Tree and Gradient Boost Classifier, achieving high accuracies and contributing to advancements in robot navigation research. Dash, Tirtharaj, et al. [5] proposed a neural network-based approach for controlling wall-following robots. Their research utilized various neural network models, including Shallow Neural Networks, to navigate robots effectively. The study emphasized the importance of using accurate training and testing procedures to avoid overfitting and improve model performance on unseen data. They evaluated their models on the dataset provided by [4], achieving significant improvements in navigation accuracy.

Dash, Tirtharaj, Tanistha Nayak, and Rakesh Ranjan Swain. [6] explored controlling wall-following robot navigation using gravitational search and feed-forward neural networks. Their approach aimed to enhance control accuracy by integrating advanced optimization techniques and neural network models. The study highlighted the challenges of model generalization and overfitting, emphasizing the need for robust testing methods to ensure reliable performance. Chen, Yen-Lun, et al. [7] investigated classification-based learning for wall-following robots using particle swarm optimization. Their work focused on optimizing classification algorithms to improve navigation accuracy. They demonstrated that particle swarm optimization enhance the performance of various machine learning models, providing insights into effective control strategies for autonomous robots. Osunmakinde, Isaac O., Chika O. Yinka-Banjo, and Antoine Bagula. [8] examined the use of Bayesian Networks and k-NN models for autonomous robot behavior development. Their research contributed to understanding how different probabilistic and instance-based models be applied to wall-following tasks. They showed that Bayesian Networks effectively improve navigation accuracy and behavior prediction in robots.

Dash, Tirtharaj. [9] proposed an automatic navigation method for wall-following robots using Adaptive Resonance Theory (ART) of type-1. Their study focused on improving navigation accuracy by applying ART models to wall-following tasks. They provided detailed performance evaluations and comparisons with other models, highlighting ART's effectiveness in handling complex navigation scenarios. Karakuş, Mücella Özbay, and E. R. Orhan. [10] presented a probabilistic neural network approach for robot navigation tasks. Their research aimed to enhance navigation control by applying probabilistic methods to handle sensor data effectively. They demonstrated that probabilistic neural networks achieve high accuracy in predicting robot movements, contributing to the development of more reliable navigation systems. LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. [11] provided a comprehensive overview of deep learning techniques. Their seminal work highlighted the advancements in neural network models and their applications across various domains, including robotics. They emphasized the potential of deep learning for improving robot navigation and control by leveraging large datasets and complex neural network architectures. Kato, Nei, et al. [12] discussed the vision for deep learning in heterogeneous network traffic control. Their paper addressed the challenges and future directions of applying deep learning techniques to complex control problems. They proposed that deep learning significantly enhance performance in managing diverse network traffic, which is relevant to control systems in robotics.

Wolpert, David H., and William G. Macready. [13] introduced the No Free Lunch Theorems for optimization. Their work provided a theoretical foundation for understanding the limitations of machine learning algorithms in solving different problems. They emphasized that no single algorithm can universally outperform others, a principle relevant to evaluating various models for wall-following

doi: 10.48047/ijiee.2025.15.4.16

robot navigation. SCITOS G5, Embedded PC and Operating System, Version, FC12-2011-01-17. [14] provided technical documentation for the SCITOS G5 mobile robot platform. This document detailed the hardware and software specifications of the SCITOS G5, which was used for collecting the sensor data in several studies, including [4]. The information was crucial for understanding the capabilities and limitations of the robot platform used in the research. Christian Martin, SCITOS G5 – A Mobile Platform for Research and Industrial. [15] described the SCITOS G5 mobile robot platform, highlighting its applications in research and industry. The document provided insights into the robot's design and functionality, emphasizing its role in data collection for wall-following research and its potential applications in various fields.

Karystinos, George N., and Dimitrios A. Pados. [16] discussed issues related to overfitting, generalization, and training set expansion. Their work highlighted the importance of using separate test sets for evaluating model performance and avoiding overfitting. This principle was applied in several studies, including those evaluating wall-following control models. F. Chollet. [17] provided the Keras deep learning library, offering a user-friendly interface for building and training neural networks. The library was used in the research to implement various deep learning models for robot navigation tasks, facilitating the development and evaluation of complex neural network architectures. Pedregosa, Fabian, et al. [18] introduced Scikit-learn, a popular machine learning library in Python. The library was utilized in the studies to implement and evaluate various machine learning models for wall-following robot navigation. It provided essential tools and algorithms for data analysis and model evaluation. Xu, Qing-Song, and Yi-Zeng Liang. [19] explored Monte Carlo cross-validation techniques. Their work provided a robust method for evaluating machine learning models, including those used in robot navigation research. Monte Carlo cross-validation was employed to ensure reliable performance assessments and to mitigate overfitting issues. Zeiler, Matthew D. [20] proposed ADADELTA, an adaptive learning rate method for training neural networks. ADADELTA was used to optimize the training of deep learning models in the research, improving their performance on navigation tasks by dynamically adjusting learning rates.

3. PROPOSED SYSTEM

This section is about the proposed methodology as shown in Fig.2. It consists of Data uploading, data preprocessing, ML model training, model predication, performance evaluation.

- **Dataset Uploading** The importing of necessary libraries and loading the dataset from a CSV file. The data is read into a pandas DataFrame, providing an initial overview of the dataset’s structure and contents. Key attributes of the dataset are examined, including data types and basic statistics, to understand its characteristics.
- **Data Preprocessing** Data preprocessing involves several crucial steps to prepare the dataset for machine learning models. It first checks for and handles any missing or duplicate values. Following this, label encoding is applied to the categorical 'Class' column to convert it into numerical format. Feature scaling is performed using StandardScaler to standardize the dataset, which helps in improving the performance and convergence of machine learning algorithms. The dataset is then split into training and testing subsets to evaluate model performance.
- **ML Model Training** Two machine learning models are trained: K-Nearest Neighbors (KNN) and Decision Tree Classifier. For each model, It first checks if a pre-trained model already exists. If not, it trains the model on the training data and saves it for future use. The KNN classifier and Decision Tree Classifier are configured with their respective parameters, and the models are trained on the scaled training data.

doi: 10.48047/ijiee.2025.15.4.16

— **Model Prediction on New Test Data** After training, the models are used to make predictions on new test data. The test data is scaled using the same scaler applied to the training data, ensuring consistency in preprocessing. Predictions are made for each row of the test dataset, and the results are added to the DataFrame for further analysis.

— **Performance Evaluation** The performance of the KNN and Decision Tree Classifiers is evaluated using various metrics including accuracy, precision, recall, and F1 score. A confusion matrix is also generated to visualize the performance of each model. The metrics are calculated and displayed for both classifiers, allowing a comparison of their effectiveness.

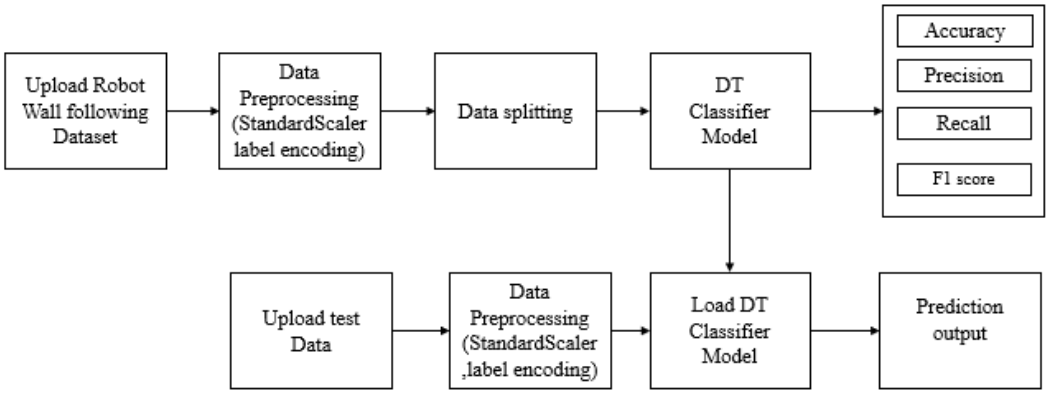

Figure 2: Block Diagram

3.1 Data Preprocessing

Data preprocessing is a fundamental step in preparing data for machine learning models. Initially, the dataset is inspected for any missing or duplicated values, which can affect the model's performance. Missing values are typically handled by imputation or removal, while duplicates are eliminated to ensure data quality. Next, the categorical target variable 'Class' is encoded into numerical values using LabelEncoder. This conversion is necessary because most machine learning algorithms require numerical input. The feature set is then standardized using StandardScaler. Scaling transforms the features to have a mean of zero and a standard deviation of one, which is crucial for algorithms sensitive to feature scaling, such as K-Nearest Neighbors and Decision Trees. The data is split into training and testing sets to evaluate the model’s performance. This split ensures that the model is trained on one subset of the data and tested on a separate, unseen subset. The splitting process helps in assessing how well the model generalizes to new, unseen data.

3.2 Model Building and Training

KNN Classifier

K-Nearest Neighbors (KNN) is a simple, non-parametric machine learning algorithm used for classification and regression tasks. It works by finding the k closest data points to a given test point in the feature space and classifying or predicting the output based on the majority class (classification) or average of values (regression) of those neighbors. The key parameters are the number of neighbors (k) and the distance metric, typically Euclidean distance.

How It Works?

1. Choose the number of neighbors, k.

doi: 10.48047/ijiee.2025.15.4.16

2. Calculate the distance (Euclidean, Manhattan, etc.) between the test point and all training data points.
3. Sort all training points by distance.
4. Select the k nearest neighbors.
5. For classification, predict the class based on majority voting from the k nearest neighbors.
6. For regression, predict the output as the average value of the k neighbors.
7. A distance metric like Euclidean is used to measure the proximity of data points.

Architecture of KNN

1. **Input Data**: A dataset with labeled examples.
2. **Distance Measure**: Measures the proximity between data points using metrics like Euclidean, Manhattan, or Minkowski distance.
3. **Selection of K**: Choose the number of neighbors (k) based on cross-validation.
4. **Prediction**: Once the nearest neighbors are identified, the algorithm predicts the class or value.
5. **Model Output**: Outputs the majority class (classification) or the average of the neighbors (regression).

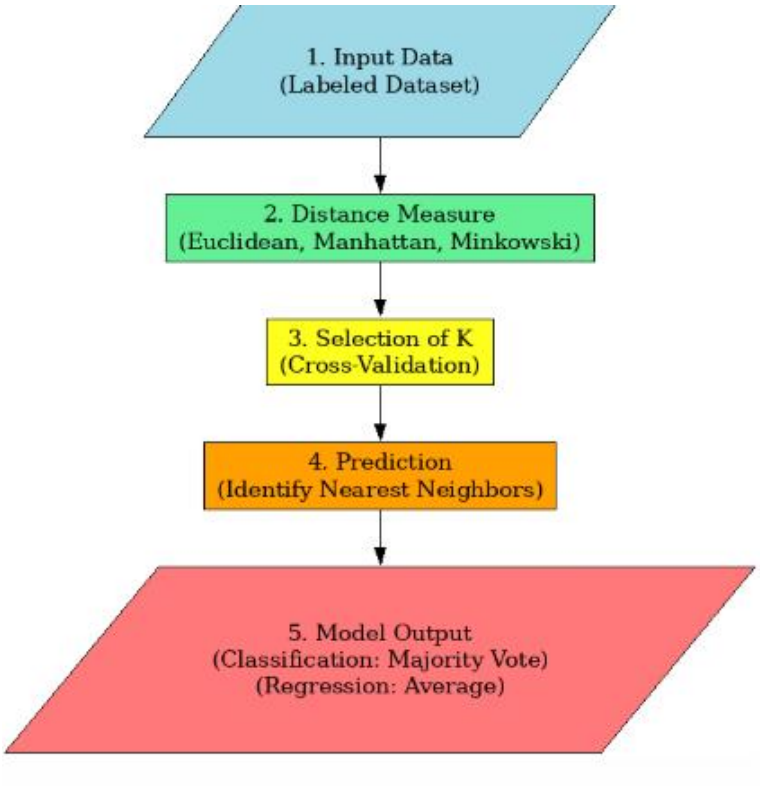

Figure 3: KNN working Flow

DTC model

A Decision Tree Classifier (DTC) is a supervised machine learning algorithm used for classification and regression tasks. It works by recursively splitting the dataset based on feature values that result in the greatest information gain (in classification) or minimize variance (in regression). The tree is

doi: 10.48047/ijiee.2025.15.4.16

composed of nodes representing decisions or tests on features and branches representing the outcome of the tests. The leaf nodes represent the final output or class label.

How It Works?

1. The dataset is recursively split into subsets based on feature values.
2. At each step, the algorithm selects the feature that best splits the data (based on criteria like Gini impurity, information gain, or variance reduction).
3. This process continues until the data is homogeneous or a stopping condition (e.g., maximum depth, minimum samples per leaf) is reached.
4. Each path from the root to a leaf represents a decision rule that classifies the data into different categories.
5. For prediction, the input data traverses the tree from the root to a leaf node, where the output is a class label (classification) or a continuous value (regression).

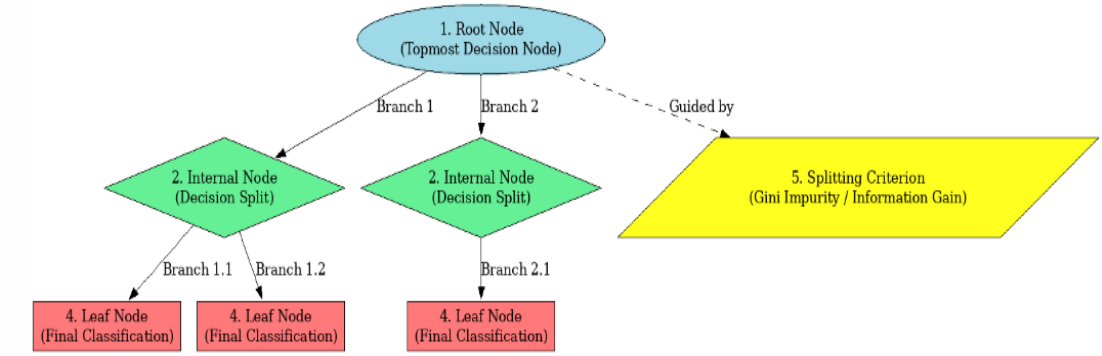

Figure 4: Working flow for DTC

Architecture of DTC

1. **Root Node**: The topmost decision node where the dataset is first split based on a chosen feature.
2. **Internal Nodes**: Decision points where the data is split based on a condition.
3. **Branches**: Represent the outcome of a decision or test.
4. **Leaf Nodes**: Terminal nodes that provide the final classification or prediction.
5. **Splitting Criterion**: Functions like Gini impurity or information gain that guide how the data is split.

4. RESULTS AND DISCUSSION

The dataset comprises ultrasound sensor readings organized by various angles and associated with specific classes. Each sensor (US1 through US24) captures data at discrete angles ranging from -180° to +180°. This angular distribution provides a comprehensive spatial coverage, allowing the system to capture reflections from obstacles in all directions around the robot.

Class 1: Obstacles in Front

In this class, images are characterized by strong reflections from obstacles directly in front of the robot. Sensors positioned at angles close to 0° (US11, US12, and US13) exhibit higher intensity readings,

doi: 10.48047/ijiee.2025.15.4.16

indicating a closer proximity to objects directly ahead. These readings help the system understand and interpret obstacles directly in the robot's path, ensuring accurate navigation and collision avoidance.

Class 2: Obstacles to the Right

For obstacles located to the right of the robot, sensors at positive angles (US15 through US24) will show elevated readings. This class encompasses images where reflections from the right-hand side are significant, with higher intensity values observed from sensors at angles such as 90° and 105°. The data helps the robot detect and navigate around objects situated on its right side, adjusting its movement trajectory accordingly.

Class 3: Obstacles to the Left

Conversely, for obstacles on the left side of the robot, sensors positioned at negative angles (US1 through US12) provide the key data. Stronger reflections from these sensors indicate the presence of obstacles on the left. Sensors at angles such as -90° and -105° are particularly useful in this context, helping the robot to detect and avoid objects on its left side, thereby ensuring smooth wall-following and navigation.

Class 4: No Obstacle

In this class, sensor readings across all angles exhibit low intensity, signifying that no significant obstacles are detected in the vicinity. The uniformity and lower values of the sensor readings across all angles (both positive and negative) suggest an open space with no nearby objects. This data is crucial for the robot to recognize and confirm clear paths, avoiding unnecessary adjustments when no obstacles are present.

Class 5: Complex Scenarios

This class has complex or irregular obstacle arrangements where multiple sensors show varied reflections. These scenarios involve objects at multiple angles or distances, creating a more intricate detection pattern. The dataset for this class helps in training the model to handle diverse and non-standard obstacle configurations, enhancing the robot's ability to navigate through more challenging environments.

9.3 Results Description

Figure 5(a) shows that the confusion matrix of the KNN classifier with True values and Error values and Figure 5(b) shows that the confusion matrix of the DTC model which have more true values that the KNN classifier. Figure 6 demonstrate that the sample predictions on test data using proposed DCT model as it has superior performance metrics.

doi: 10.48047/ijiee.2025.15.4.16

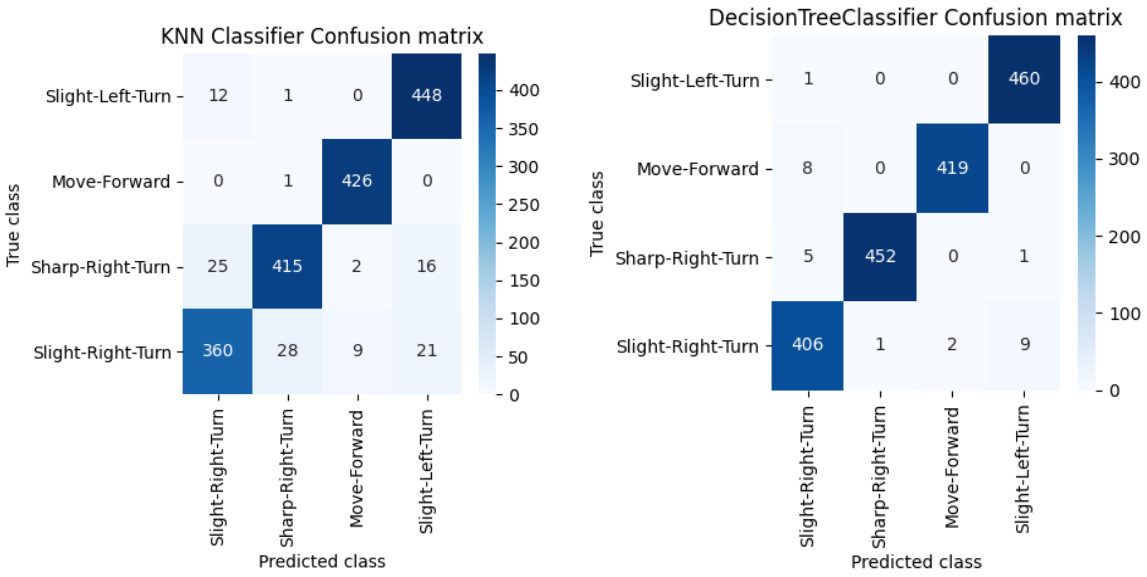

Figure 5: Confusion matrix obtained using (a) KNN classifier. (b) DTC model.

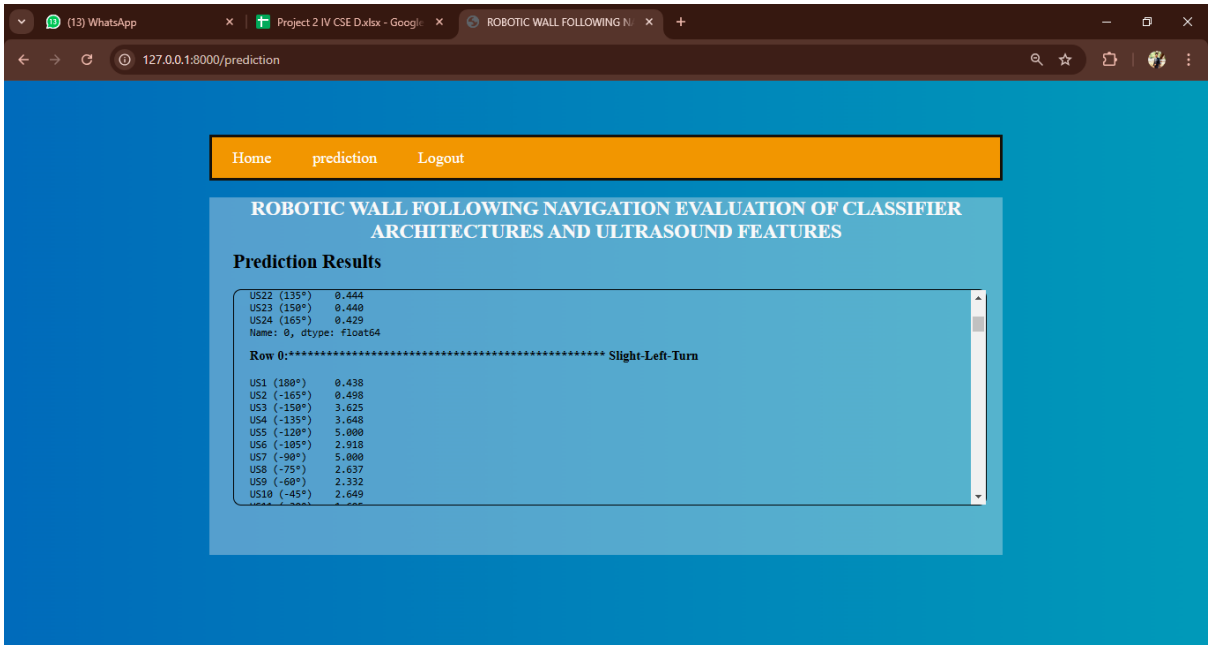

Figure 6: Sample prediction son test data.

The comparative analysis between the existing K-Nearest Neighbors (KNN) algorithm and the proposed Decision Tree Classifier (DTC) reveals that the DTC significantly outperforms KNN in terms of all evaluation metrics. KNN achieves an accuracy of 93.48%, with balanced precision (93.45%), recall (93.42%), and F-score (93.40%), indicating reliable classification when properly tuned. However, the DTC surpasses these results with an accuracy of 98.47%, precision of 98.46%, recall of 98.43%, and F-score of 98.44%, demonstrating superior capability in identifying true cases with minimal errors.

Table 1. Comparative analysis of KNN and DTC models.

Metric	KNN Classifier	DTC model

doi: 10.48047/ijiee.2025.15.4.16

Accuracy	93.48%	98.47%
Precision	93.45%	98.46%
Recall	93.42%	98.43%
F-Score	93.40%	98.44%

5. CONCLUSION

This research successfully implements machine learning models—K-Nearest Neighbors (KNN) and Decision Tree Classifier (DTC)—to classify obstacle scenarios based on ultrasound sensor readings. The dataset, consisting of 24 sensor readings covering a 360° field, enables accurate obstacle detection and navigation decisions for robotic applications. Data preprocessing, including handling missing values, label encoding, and feature scaling, ensures model robustness. Performance evaluation demonstrates that the DTC outperforms KNN with an accuracy of 98.47%, compared to 93.48% for KNN. The decision tree's superior performance is attributed to its ability to effectively capture complex decision boundaries. The implemented GUI allows users to interact with the system for real-time predictions, enhancing practical usability.

REFERENCES

[1] Varma, N., Poornima, V., Aivek, and Ravikumar Pandi. "Intelligent wall following control of differential drive mobile robot along with target tracking and obstacle avoidance." Intelligent Computing, Instrumentation and Control Technologies (ICICICT), International Conference on. IEEE, 2017.

[2] Freire, Ananda L., et al. "Short-term memory mechanisms in neural network learning of robot navigation tasks: A case study." Robotics Symposium (LARS), 6th Latin American. IEEE, 2009.

[3] Juang, Chia-Feng, Ying-Han Chen, and Yue-Hua Jhan. "Wall-Following Control of a Hexapod Robot Using a Data-Driven Fuzzy Controller Learned Through Differential Evolution." IEEE Trans. Industrial Electronics 62.1 (2015): 611-619.

[4] Wall-Following Robot Navigation Data Data Set, Machine Learning Repository, University of California, Irvine (UCI). https://archive.ics.uci.edu/ml/datasets/Wall-Following+Robot+Navigation+Data

[5] Dash, Tirtharaj, et al. "Neural network approach to control wall-following robot navigation." Advanced Communication Control and Computing Technologies (ICACCCT), International Conference on. IEEE, 2014.

[6] Dash, Tirtharaj, Tanistha Nayak, and Rakesh Ranjan Swain. "Controlling wall following robot navigation based on gravitational search and feed forward neural network." Proceedings of the 2nd International Conference on Perception and Machine Intelligence. ACM, 2015.

[7] Chen, Yen-Lun, et al. "Classification-based learning by particle swarm optimization for wall-following robot navigation." Neurocomputing 113 (2013): 27-35.

[8] Osunmakinde, Isaac O., Chika O. Yinka-Banjo, and Antoine Bagula. "Investigating the Use of Bayesian Network and k-NN Models to Develop Behaviours for Autonomous Robots." Mobile Intelligent Autonomous Systems 34 (2012): 751-764.

[9] Dash, Tirtharaj. "Automatic navigation of wall following mobile robot using adaptive resonance theory of type-1." Biologically Inspired Cognitive Architectures 12 (2015): 1-8.

doi: 10.48047/ijiee.2025.15.4.16

[10] Karakuş, Mücella Özbay, and E. R. Orhan. "Learning of robot navigation tasks by probabilistic neural network." Learning (2013).
[11] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." Nature 521.7553 (2015): 436.
[12] Kato, Nei, et al. "The deep learning vision for heterogeneous network traffic control: Proposal, challenges, and future perspective." IEEE Wireless Communications 24.3 (2017): 146-153.
[13] Wolpert, David H., and William G. Macready. "No free lunch theorems for optimization." IEEE Transactions on Evolutionary Computation 1.1 (1997): 67-82.
[14] SCITOS G5, Embedded PC and Operating System, Version, FC12-2011-01-17, MetraLabs GmbH. http://www.metralabs-service.com/downloads/SCITOS/recovery/Fedora12/SCITOS_G5-EmbeddedPC_and_OS_Fedora12.pdf
[15] Christian Martin, SCITOS G5 – A mobile platform for research and industrial, MetraLabs GmbH, Mobile Robotics. http://download.ros.org/data/Events/CoTeSys-ROS-School/metralabs.pdf
[16] Karystinos, George N., and Dimitrios A. Pados. "On overfitting, generalization, and randomly expanded training sets." IEEE Transactions on Neural Networks 11.5 (2000): 1050-1057.
[17] Chollet, F. "Deep Learning for Humans." Available: https://github.com/fchollet/keras
[18] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." Journal of Machine Learning Research 12.Oct (2011): 2825-2830.
[19] Xu, Qing-Song, and Yi-Zeng Liang. "Monte Carlo cross validation." Chemometrics and Intelligent Laboratory Systems 56.1 (2001): 1-11.
[20] Zeiler, Matthew D. "ADADELTA: an adaptive learning rate method." arXiv preprint arXiv:1212.5701 (2012).

doi: 10.48047/ijiee.2025.15.4.16