A Lightweight Weighted Ensemble Approach for Sensor-Based Terrain Classification

Eren Yildirim
Department of Electronics and Communications Engineering, American University of Malta, Malta | Department of Electrical and Electronics Engineering, Bahcesehir University, Turkiye
eren.yildirim@aum.edu.mt (corresponding author)

Yucel Batu Salman
Department of Software Engineering, Bahcesehir University, Turkiye
batu.salman@bau.edu.tr

Received: 24 December 2025 | Revised: 21 January 2026 and 25 January 2026 | Accepted: 28 January 2026

 DOI: https://doi.org/10.48084/etasr.17162

ABSTRACT

The properties of a terrain on which a robot travels have a crucial impact on the design and function of the robot. Therefore, sensing the terrain type using sensors is essential. This study proposes a lightweight method based on time-series data collected by multiple inertial sensors located on the robot. The data in Cartesian coordinates are converted into cylindrical and spherical coordinates and then fused. Next, the average value of each feature over the measurement period is calculated for dimensionality reduction. Prior to classification, data augmentation is applied to the open-source dataset to increase the number of samples. The terrain types used in this study are concrete, grass, pebbles, sand, paving stone, and synthetic running track. Testing the proposed model with a Weighted Ensemble (WE) classifier yielded 97.37% accuracy. The findings showed that using multiple coordinate systems in the reduced feature set increases classification accuracy while maintaining a relatively low computational load.

Keywords-terrain classification; data fusion; Weighted Ensemble (WE) classifier

I. INTRODUCTION

Wheeled mobile robots are increasingly being deployed in diverse outdoor environments, where terrain conditions can vary significantly within short distances [1]. To maintain stable motion and efficient navigation across such terrains, robots must be able to adapt their control strategies in real time [2]. Terrain classification, which is the process of identifying the underlying ground type, plays a central role in enabling this adaptability. By recognizing terrain characteristics, a mobile robot can adjust its traction control [3], wheel slip compensation [4], or motor torque to maintain stability and energy efficiency even under challenging operating conditions [5].

Knowledge of terrain type is therefore crucial for the design of intelligent control and planning systems in mobile robotics. Conventional methods for terrain classification have relied heavily on vision-based methods, where images from onboard cameras are analyzed to infer the type of surface the robot is traversing [6]. Although this approach can achieve high accuracy under controlled conditions, it also comes with significant limitations. Visual methods often depend on adequate lighting, clear visibility, and robust object segmentation to isolate terrain regions from surrounding elements. In real-world outdoor environments, however, lighting variations, shadows, and occlusions can easily degrade the performance of vision-based systems [7]. Therefore, alternative sensing modalities have been explored. Audio-based approaches, which classify terrain types using sound patterns produced by wheel–ground interaction, can be effective in specific contexts but are highly sensitive to environmental noise. Similarly, Light Detection and Ranging (LiDAR)-based methods [8] leverage point cloud data to estimate roughness and texture, but their computational cost and power requirements often make them unsuitable for lightweight or resource-constrained robotic platforms.

Given these limitations, vibration-based terrain classification has emerged as a promising alternative [9]. This method uses inertial sensors such as accelerometers, gyroscopes, and magnetometers to capture vibrations and motion dynamics induced by the interaction between the robot's wheels and the ground surface. A key advantage of this approach is its broad applicability, as most mobile robots already incorporate Inertial Measurement Units (IMUs) for localization and stability control. Furthermore, the data produced by these sensors are one-dimensional time-series signals, which simplify model design and reduce the need for computationally intensive preprocessing.

In this study, we propose a vibration-based terrain classification framework that leverages multidimensional inertial and positional data collected during robot motion. The

dataset includes measurements of translational accelerations in Cartesian coordinates (x, y, z). From these data, the cylindrical and spherical coordinate components are also extracted. Additionally, contextual parameters such as sensor index, speed level, and run index are included [10]. These features collectively capture both the dynamic and geometric aspects of the robot's movement, offering a rich representation of how different terrains influence its kinematic behavior [11]. During training, the data size is increased through data augmentation methods and different fusion versions among Cartesian, spherical and cylindrical coordinates. The main contribution of this study is to enhance classification accuracy while decreasing the overall computation by merging the time-wise averages of motion characteristics in three space domains.

II. METHOD

A. Dataset

We implemented our method on a publicly available dataset introduced by authors in [4]. The dataset includes measurements from an accelerometer, gyroscope, and magnetometer sensors. The measurements were obtained for six different terrain types: concrete, grass, sand, paving stone, pebbles, and synthetic running track. The sensors were mounted on a mobile robot operated under open-loop control with two different Pulse Width Modulation (PWM) duty cycles, 85% and 100%. For each terrain, measurements were repeated seven times, each lasting 4.5 s. To maintain consistency and enable reliable benchmarking with previous studies, we used only the data collected at 100% duty cycle.

The raw dataset contains $N = 253$ measurement files. Each file refers to a measurement made by a certain sensor type with a PWM duty cycle for a terrain type for a single run i, represented by a data file $\mathbf{D}_i$. Each file contains $L = 1{,}800$ sensor readings as time series. The initial dimensionality of each reading is 3, corresponding to Cartesian coordinates (x, y, z) as shown in Figure 1.

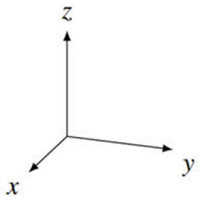

Fig. 1. Example of a Cartesian coordinate system.

The initial structure of a file $\mathbf{D}_i$ is represented by the $L \times 3$ matrix in (1), where $i \in \{1, \cdots, 253\}$:

$$\mathbf{D}_i = \begin{bmatrix} x_1 & y_1 & z_1 \\ \vdots & \vdots & \vdots \\ x_L & y_L & z_L \end{bmatrix} \quad (1)$$

B. Expansion of Coordinate Systems

To expand the feature space representing the sensor readings, each sensor reading vector $\mathbf{r}_k = [x_k, y_k, z_k]$ was converted into spherical and cylindrical coordinate systems. This transformation increases the dimension of each row from 3 to 9.

1) Extraction of Spherical Coordinates

The transformation from Cartesian to spherical coordinates is performed using (2), (3), and (4):

$$\rho_k = \sqrt{x_k^2 + y_k^2 + z_k^2} \quad (2)$$

$$\varphi_k = \tan^{-1}\left(\frac{y_k}{x_k}\right) \quad (3)$$

$$\theta_k = \cos^{-1}\left(\frac{z_k}{\rho_k}\right) \quad (4)$$

where ρ is the radial distance in 3-D space, θ is the polar angle, and φ is the azimuthal angle. This process produces the feature vector $\mathbf{r}_k^{sph} = [\rho_k, \varphi_k, \theta_k]$. An example of a spherical coordinate system is shown in Figure 2.

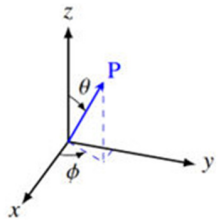

Fig. 2. Example of a spherical coordinate system.

2) Extraction of Cylindrical Coordinates

The transformation from Cartesian to cylindrical coordinates is performed using (5), (6), and (7):

$$\gamma_k = \sqrt{x_k^2 + y_k^2} \quad (5)$$

$$\varphi_k = \tan^{-1}\left(\frac{y_k}{x_k}\right) \quad (6)$$

$$h_k = z_k \quad (7)$$

where γ is the radius of the cylinder, φ is the azimuthal angle, and h is the height. This process gives the feature vector $\mathbf{r}_k^{cyl} = [\gamma_k, \varphi_k, h_k]$. An example of a cylindrical coordinate system is shown in Figure 3.

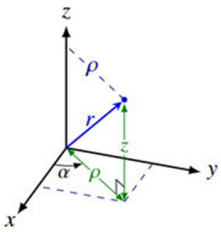

Fig. 3. Example of a cylindrical coordinate system.

The k-th augmented feature vector v_k in (8) is the concatenation of the original and expanded coordinates, where $k \in \{1, \cdots, L\}$:

$$\mathbf{v}_k = \left[\mathbf{r}_k, \mathbf{r}_k^{sph}, \mathbf{r}_k^{cyl}\right] \quad (8)$$

Out of the nine coordinate-based features, the azimuthal angle φ_k is shared by both the cylindrical and spherical coordinates spaces, causing data redundancy. Therefore, we

removed one of them from $\mathbf{v}_k$, resulting with an $L \times 8$ matrix for the i-th measurement file, as shown in (9):

$$\mathbf{D}_i = \begin{bmatrix} x_1 & y_1 & z_1 & \rho_1 & \varphi_1 & \theta_1 & \gamma_1 & h_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_L & y_L & z_L & \rho_L & \varphi_L & \theta_L & \gamma_L & h_L \end{bmatrix} \quad (9)$$

C. Feature Augmentation

In addition to the expanded coordinate system features, three file-specific metadata features were appended to each row: sensor type (S), run index (R), and terrain class (C). The sensor type indicates which sensor was used for that specific file. To convert the categorical data to numerical values, we replaced the sensor names "accelerometer", "gyroscope", and "magnetometer" with 0, 1, and 2 respectively.

The run index (R) ranges from 1 to 7, since each measurement is repeated seven times for each setting. Let $\mathbf{m}_i = [S_i, R_i, C_i]$ be the file-specific metadata vector appended to $\mathbf{v}_k$. The final feature matrix $\mathbf{D}_i$ has dimensions $L \times 11$, as given in (10):

$$\mathbf{D}_i = \begin{bmatrix} x_1 y_1 z_1 \rho_1 \varphi_1 \theta_1 \gamma_1 h_1 S_i R_i C_i \\ \vdots \; \vdots \; \vdots \; \vdots \; \vdots \; \vdots \; \vdots \; \vdots \; \vdots \; \vdots \; \vdots \\ x_L y_L z_L \rho_L \varphi_L \theta_L \gamma_L h_L S_i R_i C_i \end{bmatrix} \quad (10)$$

D. Dimension Reduction and Final Dataset Structure

The matrix $\mathbf{D}_i$ has a size of 1,800 × 11, where 1,800 is the number of readings, or equivalently, the number of time steps, and 11 is the feature length. One of the main contributions of the study is to enable classification in a lightweight form. Thus, final dimension reduction was performed by transforming the $L \times 11$ matrix $\mathbf{D}_i$ into a 1×11 vector $\mathbf{f}_i$. We calculated the average value of each feature over the entire time. Since the last three parameters of $\mathbf{D}_i$ (S_i, R_i, C_i) are file-specific metadata features, there is no need to perform averaging on them. This calculation is given in (11):

$$\bar{\mathbf{v}}_i = \frac{1}{L} \sum_{k=1}^{L} \mathbf{v}_k \quad (11)$$

The final 1×11 feature vector $\mathbf{f}_i$ in (12), representing file i, is obtained by concatenating the averaged features with the file-specific features:

$$\mathbf{f}_i = [\bar{\mathbf{v}}_i, \mathbf{m}_i] \quad (12)$$

The resulting features of this process for each of the $N = 253$ files in the dataset are stacked row-wise to form the final dataset matrix **F,** with dimensions 253 × 11, as given in (13):

$$\mathbf{F} = \begin{pmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_N \end{pmatrix} \quad (13)$$

The entire process explained in this section for a single file is illustrated in Figure 4.

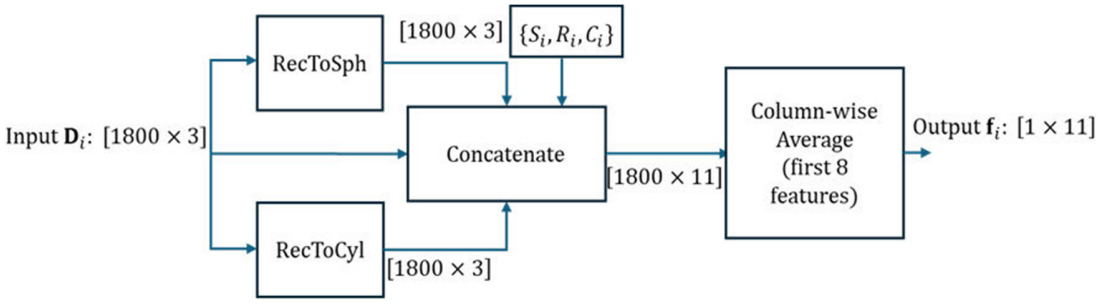

Fig. 4. Flowchart of the proposed model implementation.

III. EXPERIMENTAL RESULTS

A. Weighted Ensemble Learning

In this study, we used the AutoGluon [12] framework for classification tasks. This framework contains several models including but not limited to Random Forest (RF), k-Nearest Neighbor (kNN), and Weighted Ensemble Learning (WEL) [13] methods. We utilized the WEL method for terrain classification. WEL methods such as bagging, boosting, and stacking are employed to enhance performance and reduce variance. AutoGluon WEL collects predictions of lower-level classifiers with identical hyperparameters. It does not take all the models into consideration. Instead, it selects the models with Out-Of-Fold (OOF) predictions from the final layer as the candidates. Finally, it uses a greedy ensemble selection model to combine these predictions in a weighted manner to get the final prediction result. This calculation is shown in (14), where w_i is the weight of the probability outcome of the i-th classifier:

$$p(c|x) = \sum_{i=1}^{M} w_i p_i(c|x) \quad (14)$$

As mentioned in the previous section, we only used measurements with 100% PWM duty cycle, which results in 126 measurements. This refers to 21 samples for each terrain type. Therefore, we applied data augmentation to increase the dataset size for more reliable classification results. For this purpose, we utilized the Synthetic Minority Over-sampling Technique (SMOTE). Its implementation requires two main parameters which are the sampling method and the number of nearest neighbors. In our experiments, we chose "all" as the sampling method, meaning that all samples were resampled. Regarding the number of nearest neighbors, we used 6. As a result of data augmentation, we increased the dataset size to 222 samples, which refers to 37 samples per class.

B. Setup and Evaluation Metrics

The feature extraction stages of the proposed model in this study were implemented using Python 3.11 on the Google Colaboratory (Colab) platform. Dataset modifications and evaluation were carried out using the scikit-learn 1.6.1, pandas 2.2.2, xgboost 3.1.3, and NumPy 2.0.2 libraries, along with their submodules. During the experiments, the default settings of AutoGluon 1.5.0 were used.

The model's performance was evaluated using several metrics such as accuracy, precision, recall, and F1-score to obtain an accurate understanding of its performance. The F1-score combines precision and recall metrics to balance both. The closer the F1-score to 1, the better combination of precision and recall is obtained by the model. The formulas of the metrics are given by (15), (16), (17), and (18). The dataset was randomly divided into two parts for training and testing with a split ratio of 70:30.

Accuracy is the ratio between the number of accurate predictions and all predictions made, and its formula is shown in (15):

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \quad (15)$$

Recall, on the other hand, measures the model's ability by relating the number of correct positive predictions to the total number of actual positive samples:

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \quad (16)$$

Precision measures the ratio of correct positive predictions among all the positive predictions, as given in (17):

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \quad (17)$$

The F1-score is the harmonic mean of precision and recall and is given in (18).

$$\text{F1} - \text{score} = \frac{\text{TP}}{\text{TP+0.5(FP+FN)}} \quad (18)$$

In addition to the above metrics, the Area Under the Receiver Operating Characteristic Curve (ROC-AUC) was also calculated. It enables the assessment of a classifier's ability to differentiate between classes by considering the true positive rate and false positive rate of predicted values. A ROC-AUC value of 1.0 represents a perfect classifier, whereas 0.5 refers to the worst classifier.

C. Results

This section presents the findings of the proposed model for different feature fusion combinations. Table I shows the detailed results for each case of coordinate fusion. When only the original Cartesian coordinates were used, the model achieved an accuracy of 92.11%.

According to the results given in Table I, combining cylindrical coordinates with Cartesian did not improve accuracy, since these coordinates systems share a common feature (z). This overlap does not add any discriminative information but increases computation load. Similarly, the combination of spherical and cylindrical coordinates gave the lowest accuracy. The first reason for this is that they share the azimuthal angle as a common feature. The second reason is that both coordinate systems represent rotational, but not linear, behavior.

TABLE I. PERFORMANCE METRICS FOR DIFFERENT FUSION TYPES

Fusion	Accuracy (%)	Recall (%)	Precision (%)	F1-score	ROC-AUC
CA	92.105	92.105	92.315	0.9206	0.9812
CA + SP	97.368	97.368	97.540	0.9735	0.9935
CA + CY	96.053	96.053	96.334	0.9597	0.9915
SP + CY	96.053	96.053	96.455	0.9601	0.9893
CA + SP + CY	96.053	96.053	96.344	0.9599	0.9897

CA: Cartesian, SP: Spherical, CY: Cylindrical.

The maximum accuracy of 97.37% was achieved using the Cartesian and spherical coordinate combination. This is because the features in this case are distinct from each other. Even though they are mathematically related to each other, they still describe the 3-D points from different geometric perspectives. For instance, the radial and angular components of the spherical system can naturally represent the convex, volumetric clumping of pebbles. On the other hand, the Cartesian system captures the verticality and planar scatter of grass. Using both systems, the model benefits from a hybrid feature set that simplifies non-linear structural patterns into more linearly separable data, enhancing the distinction between granular and vertical textures.

To evaluate the contribution and impact of the proposed model on terrain classification, we performed a benchmark study against existing studies in the literature. For this task, we selected studies reporting results on the same dataset, which are presented in Table II. These studies employed different feature sets and classification methods.

TABLE II. PERFORMANCE COMPARISON WITH STUDIES IN THE LITERATURE

Method	Accuracy (%)
Proposed	97.37
TDF + MLP [14]	92.59
LSTM [14]	74.54
CNN [14]	70.37
TDF + SVM [14]	95.83
FFT + MLP [14]	83.80

Authors in [14] used various combinations of features and classifiers. Firstly, they achieved 92.59% accuracy using a Multilayer Perceptron (MLP) classifier on Time-Domain Features (TDF) manually extracted from the time-series measurement data. The extracted TDFs included mean absolute value, root mean square, number of zero crossings, maximum, minimum, peak-to-peak, number of slope changes, Willison amplitude, and waveform length. The model parameters were: learning rate 0.001, batch size 32, 60 epochs, and the Adam optimizer. Furthermore, Fast Fourier Transform (FFT) features were used with the MLP classifier, achieving 83.80% accuracy. For this experiment, the number of epochs was 20, whereas all other parameters were the same as for TDF+MLP. Using the

same set of TDFs with a Support Vector Machine (SVM) classifier resulted in 95.83% accuracy. In this case, the SVM employed a linear kernel with a regularization parameter of $C = 0.1$.

In the same study, authors also implemented deep learning methods, including Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN) classifiers. For the LSTM network, the architecture consisted of two layers, each with 75 hidden units, followed by a fully connected perceptron layer containing 200 neurons. The training parameters were: learning rate 0.005, batch size 20, NAdam optimizer, and 175 training epochs. This approach achieved 74.54% accuracy. The CNN was trained with a learning rate of 0.001, batch size 32, and 60 epochs, resulting in 70.37% accuracy. It is important to mention that MLP, LSTM, and CNN-based methods require a high computation load. Authors in [14] noted that FFT+MLP had 648,326 hidden trainable parameters, whereas LSTM and CNN had 87,056 and 423,884 parameters, respectively.

The proposed method outperformed all approaches presented in Table II. The primary reason for this improvement is the fusion of coordinate systems. As discussed in the previous results subsection, combining Cartesian and spherical coordinates provides the maximum amount of discriminative information while avoiding redundancy. In addition, the averaging step contributes the high performance by reducing the influence of possible outliers and noisy measurements in the time-series data. In contrast, when CNN and LSTM models are applied directly to raw data, such variations are treated as part of the input signal, which can degrade classification accuracy.

On the other hand, the proposed method achieved 97.37% accuracy with very low computation cost. The training and testing times using Cartesian + Spherical features were 2.39 s and 0.02 s, respectively. The inference times for comparative studies in Table II were not reported by the authors.

Figure 5 displays the confusion matrices for the data fusion types: (a) Cartesian, (b) Cartesian + cylindrical, (c) Cartesian + spherical, (d) spherical + cylindrical, and (e) Cartesian + spherical + cylindrical. Across all approaches, the classifier demonstrates strong performance, with high diagonal density and minimal misclassification. This indicates that the geometric features effectively capture the distinguishing characteristics of the six terrain types: concrete (class 1), grass (class 2), pebbles (class 3), sand (class 4), paving stone (class 5), and synthetic running track (class 6).

Using only Cartesian features (Figure 5(a)) already yields accurate recognition of all classes, although isolated misclassifications occurred, such as grass being predicted as pebbles or concrete being confused with pebbles. The possible reason for these errors is the local similarities in texture or point-density patterns. Adding cylindrical features (Figure 5(b)) improved performance, particularly for sand and synthetic running track. However, a minor confusion appeared between grass and pebbles, suggesting that cylindrical coordinates alone may not sufficiently differentiate surfaces with similar radial variations.

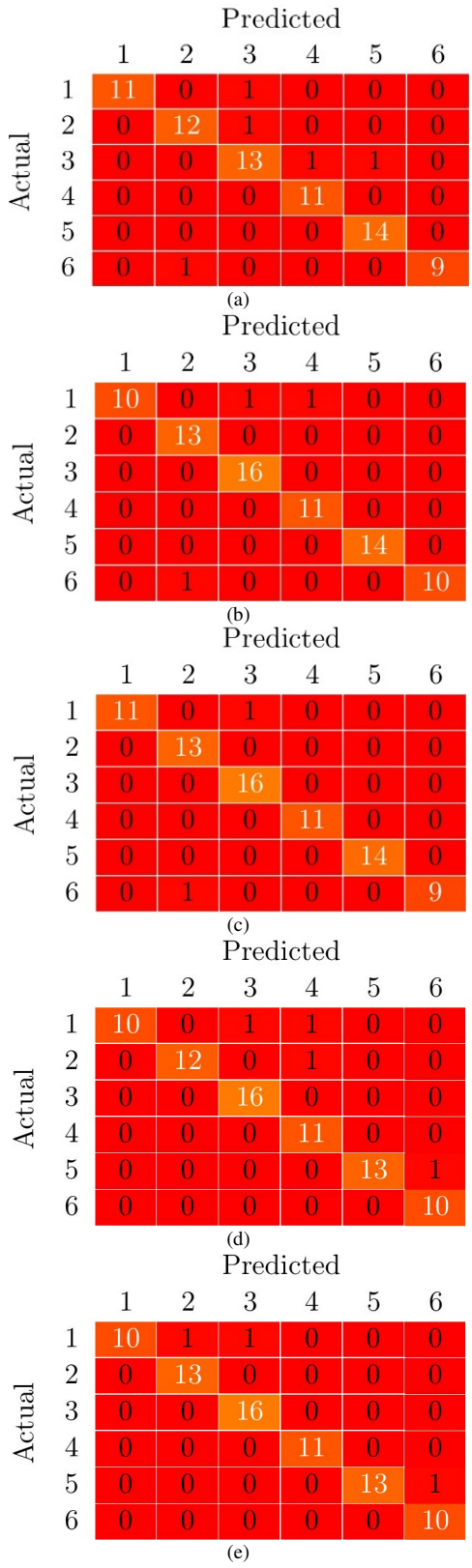

Fig. 5. Confusion matrices for different data fusions: (a) Cartesian, (b) Cartesian + cylindrical, (c) Cartesian + spherical, (d) spherical + cylindrical, (e) Cartesian + spherical + cylindrical.

The Cartesian + spherical fusion (Figure 5(c)) produced the cleanest matrices, with nearly perfect separation across all classes. Spherical features enhance angular and radial structure representation, improving discrimination, especially between grass, pebbles, and concrete, which often share overlapping geometric textures in Cartesian space.

When fusing spherical and cylindrical coordinates (Figure 5(d)), performance remained high, though slightly less consistent than Cartesian-based fusions. A small confusion between paving stone and synthetic running track occurred, potentially due to similarities in surface regularity when absolute positional information is removed.

The full fusion approach (Figure 5(e)) yielded near performance. Misclassifications were nearly eliminated across all six terrain types. This indicates that Cartesian features provide essential spatial anchoring, whereas spherical and cylindrical features contribute complementary geometric descriptors.

IV. CONCLUSION

This study presented a novel, lightweight approach for terrain classification by utilizing time-series data obtained from inertial sensors. The core contribution of this work lies in the feature-level fusion of data transformed into different coordinate systems, namely Cartesian, cylindrical, and spherical, processed through a Weighted Ensemble (WE) classifier.

The experimental results demonstrated that the choice of coordinate combinations is critical for model performance. Fusion of all available coordinate systems resulted in data redundancy, whereas combining Cartesian and spherical coordinates yielded the highest accuracy of 97.37%. This superior performance is attributed to the distinct geometric perspectives provided by these two systems; unlike cylindrical coordinates, which share the z-axis with Cartesian and the azimuthal angle with spherical, the Cartesian–spherical pair maximizes discriminative information without introducing overlapping features.

Furthermore, the proposed model demonstrated significant advantages over existing methods in the literature. It outperformed traditional machine learning approaches and complex deep learning architectures on the same dataset. Most importantly, this high level of accuracy was achieved with a drastically reduced computational load compared to deep learning models, which require training a large number of parameters. Consequently, the proposed method offers an efficient, lightweight, and high-accuracy solution that is ideal for real-time implementation on mobile robots with limited onboard processing power.

Based on the results obtained, future research can be planned for several different aims. One limitation of this study is that the model operates in offline mode. Future work will focus on deploying the algorithm on real-time controllers. The feature averaging over time scheme can be modified so that the average scores update dynamically as the robot travels and collects more data. Validating the inference speed and power consumption on actual robotic hardware is essential to prove its viability for autonomous mobile robots operating in the field.

Although the current model achieves high accuracy using only inertial sensors, a limitation is its reliance on a single modality, which can be risky in dynamic environments. Future studies could investigate fusing this lightweight inertial model with visual data or Light Detection and Ranging (LiDAR) to resolve possible extreme cases where inertial features are similar but visual textures differ.

The experiments can be extended by adding the values obtained by other aggregation methods such as median, minimum, maximum, skewness, or variance to the feature vector. The contribution of these features to classification accuracy must be examined. However, because the size of the final vector is directly proportional to the number of statistical features, adding more features will increase vector size and computational load.

REFERENCES

[1] Z. Zhou, G. Jiao, W. Chen, Y. Lu, and S. Luo, "Wave equivalent plane estimation for wave reconstruction in imaging through water waves," in *2017 IEEE International Conference on Real-time Computing and Robotics*, Okinawa, Japan, 2017, pp. 459–464, https://doi.org/10.1109/RCAR.2017.8311905.

[2] L. Ojeda, J. Borenstein, G. Witus, and R. Karlsen, "Terrain characterization and classification with a mobile robot," *Journal of Field Robotics*, vol. 23, no. 2, pp. 103–122, Feb. 2006, https://doi.org/10.1002/rob.20113.

[3] M. G. H. Nampoothiri, B. Vinayakumar, Y. Sunny, and R. Antony, "Recent developments in terrain identification, classification, parameter estimation for the navigation of autonomous robots," *SN Applied Sciences*, vol. 3, no. 4, Mar. 2021, Art. no. 480, https://doi.org/10.1007/s42452-021-04453-3.

[4] P. Sarcevic *et al.*, "Online Outdoor Terrain Classification Algorithm for Wheeled Mobile Robots Equipped with Inertial and Magnetic Sensors," *Electronics*, vol. 12, no. 15, July 2023, Art. no. 3238, https://doi.org/10.3390/electronics12153238.

[5] C. Weiss, N. Fechner, M. Stark, and A. Zell, "Comparison of Different Approaches to Vibration-based Terrain Classification," in *3rd European Conference on Mobile Robots*, Freiburg, Germany, 2007.

[6] E. Nebot, J. Guivant, and S. Worrall, "Haul truck alignment monitoring and operator warning system," *Journal of Field Robotics*, vol. 23, no. 2, pp. 141–161, 2006, https://doi.org/10.1002/rob.20114.

[7] T. Arafin, A. Hosen, Z. Najdovski, L. Wei, M. Rokonuzzaman, and M. Johnstone, "Advances and Trends in Terrain Classification Methods for Off-Road Perception," *Journal of Field Robotics*, vol. 42, no. 7, pp. 3515–3544, Oct. 2025, https://doi.org/10.1002/rob.22586.

[8] M. K. Villareal and A. F. Tongco, "Remote Sensing Techniques for Classification and Mapping of Sugarcane Growth," *Engineering, Technology & Applied Science Research*, vol. 10, no. 4, pp. 6041–6046, Aug. 2020, https://doi.org/10.48084/etasr.3694.

[9] H. Wang, E. Lu, X. Zhao, and J. Xue, "Vibration and Image Texture Data Fusion-Based Terrain Classification Using WKNN for Tracked Robots," *World Electric Vehicle Journal*, vol. 14, no. 8, Aug. 2023, Art. no. 214, https://doi.org/10.3390/wevj14080214.

[10] D. Csík, Á. Odry, J. Sárosi, and P. Sarcevic, "Inertial sensor-based outdoor terrain classification for wheeled mobile robots," in *2021 IEEE 19th International Symposium on Intelligent Systems and Informatics*, Subotica, Serbia, 2021, pp. 159–164, https://doi.org/10.1109/SISY52375.2021.9582504.

[11] T. Knuth and P. Groves, "IMU Based Context Detection of Changes in the Terrain Topography," in *2023 IEEE/ION Position, Location and Navigation Symposium*, Monterey, CA, USA, 2023, pp. 680–690, https://doi.org/10.1109/PLANS53410.2023.10140086.

[12] N. Erickson *et al.*, "AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data." arXiv, Mar. 13, 2020, https://doi.org/10.48550/arXiv.2003.06505.

[13] M. V. C. Aragão *et al.*, "A practical evaluation of AutoML tools for binary, multiclass, and multilabel classification," *Scientific Reports*, vol. 15, no. 1, May 2025, Art. no. 17682, https://doi.org/10.1038/s41598-025-02149-x.

[14] M. Sátor-Érsek, S. Kajan, and L. Körösi, "Vibration based terrain classification for mobile robot using machine learning methods," *Journal of Electrical Engineering*, vol. 76, no. 5, pp. 420–428, Oct. 2025, https://doi.org/10.2478/jee-2025-0044.