EECE-5644 Machine Learning & Pattern Recognition 2026-SPRG

Multi-Modal Terrain Classification and Sim-to-Real Transfer on a Quadruped Robot

Luke Stevenson, M.S. (2026)

April 18, 2026

Department of Electrical and Computer Engineering
Northeastern University

ABSTRACT

When a robot is in the field, knowing what terrain a robot is actively traversing can provide the information to make necessary adjustments in real time. This research develops a multi-modal terrain classification system for a 12 DoF quadruped robot, utilizing data from an IMU, a time-of-flight depth camera, and an RGB camera. Data collection involved the robot walking over five terrains; grass/dirt, concrete/pavement, tile, wood, and carpet, which were segmented into 4 second windows. The feature vectors built from this data consist of 48 statistical, spectral, and texture features extracted from each of these windows. Classifiers k-Nearest Neighbors and Random Forest were trained on simulation data (TartanGround, 14,112 samples, 2 classes) and a real-world dataset (1,089 samples, 5 classes). Training resulted in the fused-sensor Random Forest model providing a 99.08% test accuracy on real world data, however direct sim-to-real training achieved only 57.78% revealing a significant domain gap. Further analysis of the learning curve demonstrates that 94% accuracy can be attained with only 76 training samples, displaying practical deployment of this system.

Contents

I Introduction

Understanding environments is a fundamental capability robots need to ensure stable and efficient operation, especially in quadruped robots that must have precise motion to achieve movement. Different terrains a robot traverses can have extremely dissimilar surfaces - higher or lower friction, smooth or uneven texture, etc - which results in the same gait plan becoming more or less effective depending on terrain type. Training a terrain detection classifier assists with this issue by providing the information to change gait parameters based on the identified terrain.

The robot used for this research, Jango, is a custom 12 degree of freedom quadruped robot with servos used for actuation, on-board sensors (IMU, ToF Camera, RGB Camera), and ROS2 code developed (by me, prior to this research) for simulation, control, and data collection. The robot utilizes a fixed gait, no matter the circumstance, with parameters such as cycle time, step height, and stride length being hard coded. Jango has no active environmental awareness, such that the robot operates the same no matter the context. This project addresses that fault by developing a model that classifies terrain types from real-world multi-modal sensor data, providing the necessary step to build adaptive gait control.

Prior work in terrain classification for robots rely heavily on simulated environments, proprietary hardware, and other non-generalizable methods. The TartanGround dataset [1] is a large-scale simulated dataset for legged robots with IMU, depth map, and RGB image modalities from 63 environments, providing an accessible starting point for baseline model training. Recent research has developed terrain adaptive movement in industrial [2] and construction [3] settings, however the sim-to-real transfer of this work is still unoptimized in the general case. This project aims to close that gap by training, evaluating and analyzing models based on both the TartanGround simulation data and the real-world data, displaying the dependencies and quantifying the transfer gap.

The specific contributions of this work are:

- Trained terrain classification model on a custom quadrupedal robot
- 48-feature multi-modal feature development (IMU + depth + RGB)
- Quantitative sim-to-real transfer analysis with root cause identification
- Sensor modality ablation revealing discrepancies between simulation and real-world data
- Learning curve analysis to define the sufficient amount of data required for each terrain

The report is structured as follows: Section II details the simulation and real-world datasets, the data collection method, and the feature engineering procedure. Section III reports the classification algorithms, data preprocessing, and code architecture. Section IV provides the results of the baseline, cross-dataset, ablation and learning curve experiments. Section V discusses the causes of sim-to-real gap, as well as analyzing the models against each other and their limitations. Section VI concludes the report with future work and key takeaways from this project.

II Data Description

A TartanGround Simulation Dataset

The TartanGround dataset [1] is a large scale simulation dataset built for robot perception and navigation, providing multi-modal sensor data from legged (ANYmal), omnidirectional, and differential drive robots in 63 generated environments. In this research, 10 environments were chosen as they were generated by a quadrupedal robot and matched the desired terrain targets concrete/pavement (Downtown, ModularNeighborhood, NordicHarbor, ModernCityDowntown) and grass/dirt (ForestEnv, Gascola, GreatMarsh, SeasonalForestAutumn, SeasonalForestSpring, SeasonalForestWinter). Where TartanGround falls short for this project is with indoor environments, with neither carpet, wood or tile terrains being traversed by a quadrupedal robot in this dataset. From the selected simulated trajectories, sensor data was extracted with a 4-second sliding window with a 50% overlap, yielding 14,112 total samples (8,631 grass/dirt, 5,481 pavement). The feature vectors include gyroscope and gravity-removed accelerometer at 10 Hz, body velocity, depth images (640×640), and RGB images. It is important to note that the body velocity extracted from each window is ground-truth which is not available on Jango, this will be addressed in the discussion of the domain gap in Section V.

B Real-World Dataset

The real-world data was collected using a custom 12 degree of freedom quadrupedal robot named Jango. The robot is ROS2 based publishing sensor data, joint commands, joint states, and odometry topics making the system easy to record and analyze. Jango has three main sensors; IMU publishing at 50 Hz with a prebiased correction, a time-of-flight depth camera publishing PointCloud2 data at 12 Hz, and an RGB camera publishing compressed JPEG images at 20 Hz. The data collection process was automated with a ROS2 trial recording node (`record_terrain_trial.py`) that performs the following procedure:

1. Stand for 3 s for calibration and sensor sync
2. Sensor and ROS2 topic sync check, start rosbag recording
3. Walk forward at 0.35 m/s for set time
4. Stand and stop recording

To ensure the data was valid, real time sensor synchronization was monitored across all three sensors, resulting in a median skew of 36 ms. Along with this, post-trial validation verified IMU magnitude and rate, depth point count, and sensor sync quality to catch any faulty trials. In the real-world data collection process, 42 trials were recorded across all five terrain classes, summarized in Table 1.

Table 1: Real-world data collection statistics.

Class	Windows	Trials
Carpet	199	15
Grass/Dirt	234	3
Pavement	306	8
Tile	260	4
Wood	90	12
Total	**1,089**	**42**

The carpet and wood data was gathered in an apartment setting requiring shorter walks (10–30 seconds) and more trials, while the tile, grass/dirt, and pavement data were eligible for longer walks (120–240 seconds), only needing a few trials each. A limitation presented in the data collection process is surface diversity: the carpet and wood data were collected at the same location, making the data less generalizable than desired. Along with this, the robot's trajectory suffered from veering in some terrains due to uneven surfaces, however this did not inhibit or hinder classification.

C Feature Engineering

The feature extraction pipeline was shared between the simulation and real-world dataset to ensure identical feature columns for both datasets. Each 4 second window built a 48 dimensional feature vector across all three sensor modalities.

1 IMU Features (29 features)

The IMU features extracted are;

- X, Y, Z gravity removed accelerometer (mean, standard deviation, min, max)
- Acceleration magnitude statistics (RMS, standard deviation, peak)
- X, Y, Z gyroscope statistics (mean, standard deviation)
- Gyroscope magnitude (RMS)
- Velocity features (forward mean and standard deviation, lateral mean, magnitude mean)
- FFT-based gait frequency features (dominant frequency, dominant frequency power, spectral entropy)

Gravity was removed from the real-world feature data by subtracting the mean Z-axis acceleration as the TartanGround dataset provides gravity-removed accelerometer data. The simulated dataset also provides ground-truth body velocity, forcing forward acceleration mean as a substitute on the real-world data resulting in a noted discrepancy between the feature vectors.

2 Depth Features (7 features)

The depth features extracted are;

- Depth statistics (mean, standard deviation, minimum)

- Laplacian roughness (standard deviation of the Laplacian response)
- Sobel gradient magnitude mean (terrain slope indicator)
- Near-pixel ratio (fraction of pixels closer than 1 m)
- Temporal variance across sampled frames (captures robot bounce)

In the real-world data, the PointCloud2 ROS2 topic messages were projected onto a 2D depth image using the time-of-flight camera intrinsics (fx=fy=136, 240x180 resolution). For the simulated data, the depth PNG images were unchanged and loaded directly. Both of these depth images were then resized to a predetermined 120x120 resolution before analyzing the gradient for normalized features.

3 RGB Features (12 features)

The RGB image features extracted are;

- Per-channel color statistics (mean and standard deviation for R, G, B, normalized to [0,1])
- HSV brightness and saturation means
- Local Binary Pattern texture descriptors (P=8, R=1, uniform method: mean, standard deviation, entropy)
- Canny edge density

The images from both datasets were resized to a normalized 128x128 resolution. The local binary pattern algorithm captured texture differences, while the Canny edge density infers smooth surfaces from rough surfaces.

III Methodology

A Algorithms

1 k-Nearest Neighbors

The k-Nearest Neighbors algorithm is a non-parametric regression and classification model that labels data based on a majority vote of the k nearest samples in the feature space. To determine the distance between a test sample $\mathbf{x}$ and a training sample $\mathbf{x}_i$ in the 48 dimension feature space (p=48), the Euclidean distance is calculated:

$$d(\mathbf{x}, \mathbf{x}_i) = \sqrt{\sum_{j=1}^{p} (x_j - x_{ij})^2} \tag{1}$$

The majority vote rule to determine the classification among k nearest neighbors:

$$\hat{y} = \arg\max_{c} \sum_{i \in N_k(\mathbf{x})} 1(y_i = c) \tag{2}$$

where $N_k(\mathbf{x})$ is the set of k training samples closest to $\mathbf{x}$ and $\mathbf{1}(x)$ is the indicator function. As it is possible for the data collected to form irregular clusters and non-convex class regions, kNN is a fitting

model to train as it makes no assumptions about the shape of the decision boundary. For this research, the kNN algorithm provides the baseline model to be compared against Random Forest for classification performance. The evaluated k values are $k \in \{3, 5, 7, 9, 11, 15, 21\}$ for the validation data with k=3 being selected for the real-world data model training.

2 Random Forest

The Random Forest algorithm is an ensemble learning method, constructing decision trees while training. Each of these trees are built from a bootstrap sample $\mathcal{D}_b$, chosen along side a replacement sample from the training data, and a random subset of those features are considered at each split. To determine the best split at each node, the Gini impurity is minimized:

$$G = 1 - \sum_{c=1}^{C} p_c^2 \tag{3}$$

where p_c is the ratio of samples belonging to class c at that node and $C = 5$ is the number of terrain classes. The ensemble prediction is determined by majority vote across all T trees:

$$\hat{y} = \arg\max_{c} \sum_{t=1}^{T} \mathbf{1}(h_t(\mathbf{x}) = c) \tag{4}$$

where $h_t(\mathbf{x})$ is the prediction of the t-th tree. The total decrease in Gini impurity over all nodes where feature j is used for splitting, averaged across all trees, determines the feature importance for feature j. This allows for nonlinear relationships to be managed effectively, a necessary attribute due to the nonlinear nature the data creates, such as the IMU features and terrain types. The Gini impurity also provides feature importance scores which assisted with the ablation analysis in Section IV. The class weight was enabled and set to balanced to handle the data imbalance and the parameters tested $n_{\text{estimators}} \in \{50, 100, 200, 300, 500\}$, with $n = 50$ selected for real-world data.

B Preprocessing and Validation

For feature normalization, python's scikit-learn StandardScaler transformed each feature x_j as :

$$x'_j = \frac{x_j - \mu_j}{\sigma_j} \tag{5}$$

where μ_j and σ_j are the mean and standard deviation of feature j from the training data only, with that same transform being used for the validation and test sets. Normalization is necessary for kNN as the Euclidean distance (Eq. 1) is sensitive to the scale of the features, large scale features (e.g., `dominant_freq_power` ranging 0-60) would overshadow smaller magnitude features (e.g., `rgb_edge_density` ranging 0–0.01). Random Forest does not require this same normalization due to its scale invariance, however it was applied anyways for consistency.

The train/validation/test data split was chosen to be 70/15/15 with preserved class proportions across

each partition. The validation data was used to select the hyper parameters for both models (k for kNN, $n_{\text{estimators}}$ for Random Forest) and test data was used final evaluation only. Along with this, a 5-fold cross-validation was used on the training dataset to supply accuracy estimates with variance.

C Experimental Design

In this research five experiments were conducted on the simulated and real-world datasets to evaluate the in-domain and sim-to-real transfer performance:

1. **Sim-only:** Train and test on TartanGround data (2 classes)
2. **Real-only:** Train and test on Jango real-world data (5 classes)
3. **Sim-to-Real transfer:** Train on all sim data, test on all real data (2 shared classes)
4. **Combined:** Train on sim + real training split, test on real test split (2 shared classes)
5. **Real-only 2-class:** Same train/test split as Experiment 4, without sim data

Along with these experiments, feature ablation on each sensor modality (independent and pairwise combinations, 7 total configurations) and learning curve analysis (10%, 25%, 50%, 75%, and 100% of real-world training data) were used to further evaluate and understand the sensor importance and the data requirements of the trained models.

D Evaluation Metrics

Model performance was determined by:

- Classification accuracy
- Precision (per-class and macro-averaged)
- Recall (per-class and macro-averaged)
- F1-score (per-class and macro-averaged)
- Confusion Matrices

For training set robustness, a 5-fold cross validation accuracy with standard deviation was also recorded. The metrics were chosen to build a full understanding of the models performance. Classification accuracy, while providing the overall accuracy, doesn't tell the complete story and can be misleading. To counteract this, per-class precision and recall are used; precision revealing when a class is over-predicted and recall catching false negatives. As there are underrepresented classes, the metrics are then macro-averaged and then combined into the F1-score. The confusion matrices provide the full classification detail, specifying what terrains are confused with each other.

IV Results

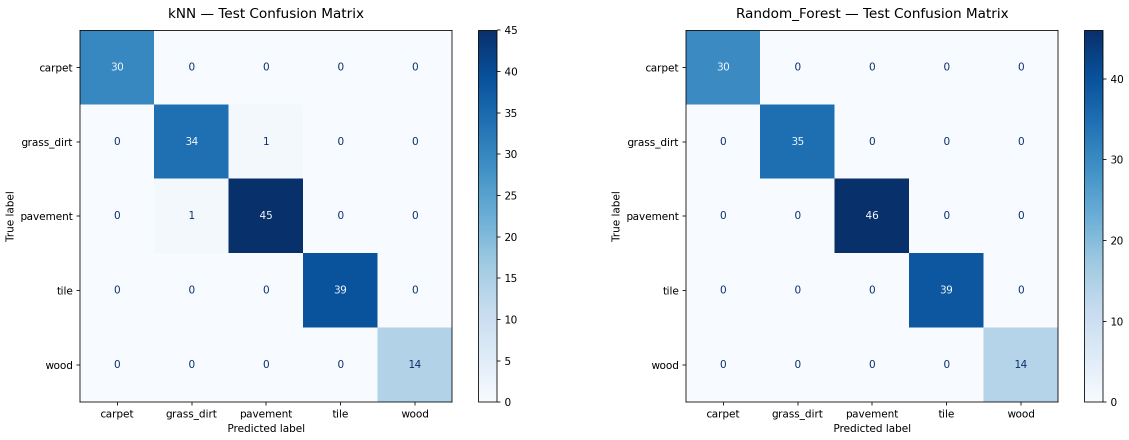
A Baseline Classification

In-domain classification yielded extremely strong performance for both datasets. The TartanGround simulation data achieved 98.75% and 97.00% accuracy for Random Forest and kNN respectively for the two terrain classes. The real-world five class dataset performed even better with Random Forest reaching 99.08% test

accuracy with a macro-averaged F1 score of 99.22% and the kNN model returning a 98.47% accuracy and F1 score of 98.74%, as summarized in Table 2. The cross-validation performed on the training set ensured the results were not caused by overfitting, with Random Forest at 97.90% $\pm$ 0.77% and kNN at 97.38% $\pm$ 1.01%. The confusion matrices in Figure 1 display all five terrain classes being well separated with the only misclassifications occurring between grass/dirt and pavement.

Table 2: Baseline classification results.

Dataset	Model	Accuracy	Macro F1	CV Accuracy
Sim-only (2 cls)	kNN (k=7)	97.00%	96.85%	—
Sim-only (2 cls)	RF (n=300)	98.75%	98.68%	—
Real-only (5 cls)	kNN (k=3)	98.47%	98.74%	97.38% $\pm$ 1.01%
Real-only (5 cls)	RF (n=50)	99.08%	99.22%	97.90% $\pm$ 0.77%



(a) kNN (k=3)

(b) Random Forest (n=50)

Figure 1: Confusion matrices for real-world 5-class classification (test set).

B Cross-Dataset Transfer Analysis

The cross-dataset experiments expose a significant domain gap between the real-world and simulation data. While the in-domain models performed exceptionally well, training on simulation data and testing on real-world data yielded a Random Forest accuracy of only 57.78% with a macro-averaged F1 score of 53.93%. The confusion matrix in Figure 3, clearly shows this picture, with pavement recall falling to a mere 25%. Experiment 4 indicates that the combination of simulation and real-world training data does not improve compared to real-only training (98.15% vs. 99.38% in Experiment 4b), showcasing that the simulation data only serves to add noise rather than assist in the training. These results are presented in Figure 2, with Experiment 3 displaying the sim-to-real transfer gap.

Table 3: Cross-dataset experimental results (RF only; kNN follows same trends).

Experiment	RF Accuracy	RF Macro F1
1. Sim-only (within sim)	98.75%	98.68%
2. Real-only (5 classes)	99.08%	99.22%
3. Sim-to-Real transfer	57.78%	53.93%
4. Combined (sim+real) to Real	98.15%	98.12%
4b. Real-only (2 classes, same split)	99.38%	99.37%

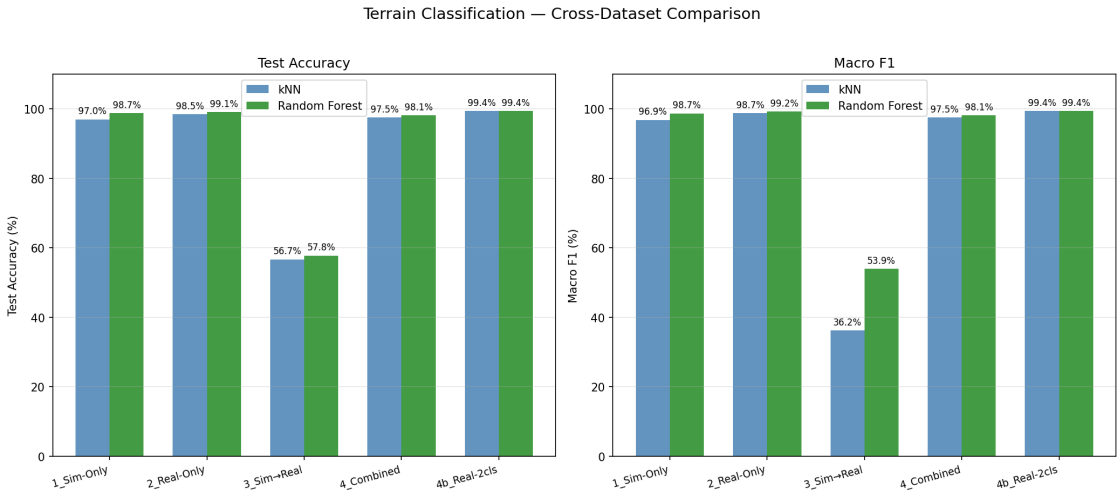


Figure 2: Cross-dataset comparison: accuracy and macro F1 across all experiments.

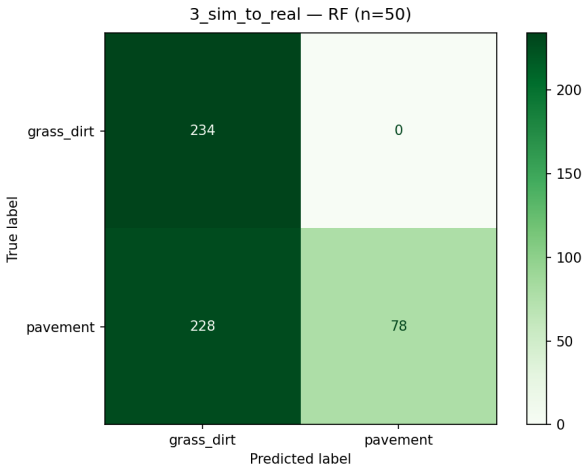


Figure 3: Confusion matrix for Sim→Real transfer (RF). Pavement recall collapses to 25.5%, with the majority misclassified as grass_dirt.

C Feature Ablation Study

Feature ablation analysis reveals an interesting inversion in sensor importance between the simulation and real-world datasets. The real-world domain has the IMU features alone achieving 95.11%, an almost matched performance to the fused model (99.08%). Only IMU features on the simulation data results in only a 67.22%, however the depth and RGB features dominate at 97.05% and 98.56% respectively. For real-world data, fusing all modalities improves accuracy by 3.98% over the best single modality, with IMU and RGB fusion resulting in the best pair at 99.08%. Depth features had the least contribution due to the flat surfaces of the terrains tested. Simulation data saw its best performance with Depth and RGB fusion at 98.80%, only a slight improvement over the Depth and RGB single modality, as detailed in Table 4.

Table 4: Feature ablation: RF test accuracy by sensor modality.

Modality	Features	Real Data	Sim Data
IMU Only	29	95.11%	67.22%
Depth Only	7	70.64%	97.05%
RGB Only	12	93.88%	98.56%
IMU + Depth	36	97.25%	96.88%
IMU + RGB	41	99.08%	97.80%
Depth + RGB	19	96.64%	98.80%
All (Fused)	48	99.08%	98.65%

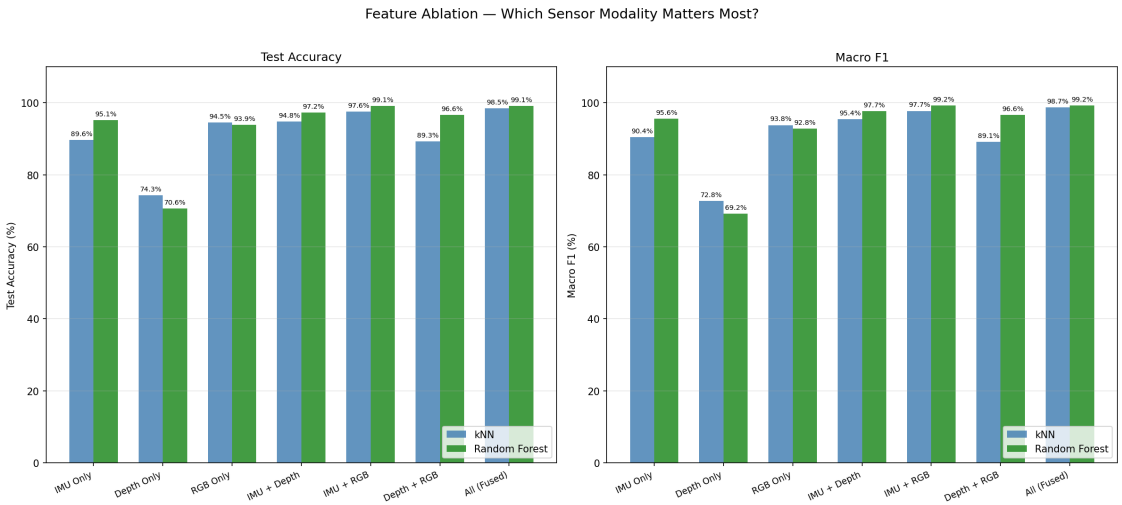


Figure 4: Feature ablation comparison across sensor modalities.

D Learning Curve Analysis

Analyzing the learning curve details the minimum amount of real-world data necessary to achieve usable terrain classification. Presented in Table 5 and Figure 5, Random Forest attains 94.01% $\pm$ 0.31% accuracy

with only 76 training samples, about 15 samples per class. Performance increase is still measurable at 190 samples, however accuracy plateaus at 50% training data and beyond. kNN shares a similar story, reaching 90.64% with only 76 samples and increasing several percent until 50% training data. This trend seen across both models indicate that terrain classification does not require large-scale data collection, only needing about 30 trials across all terrains for practical use.

Table 5: Learning curve: RF accuracy vs. training set size (real-world data, 5 repeats).

Training % (N)	kNN Accuracy	RF Accuracy
10% (76)	90.64% ± 1.64%	94.01% ± 0.31%
25% (190)	94.68% ± 1.05%	97.43% ± 0.57%
50% (381)	97.06% ± 0.57%	98.10% ± 0.49%
75% (571)	97.74% ± 0.37%	98.65% ± 0.31%
100% (762)	98.47% ± 0.00%	99.08% ± 0.00%

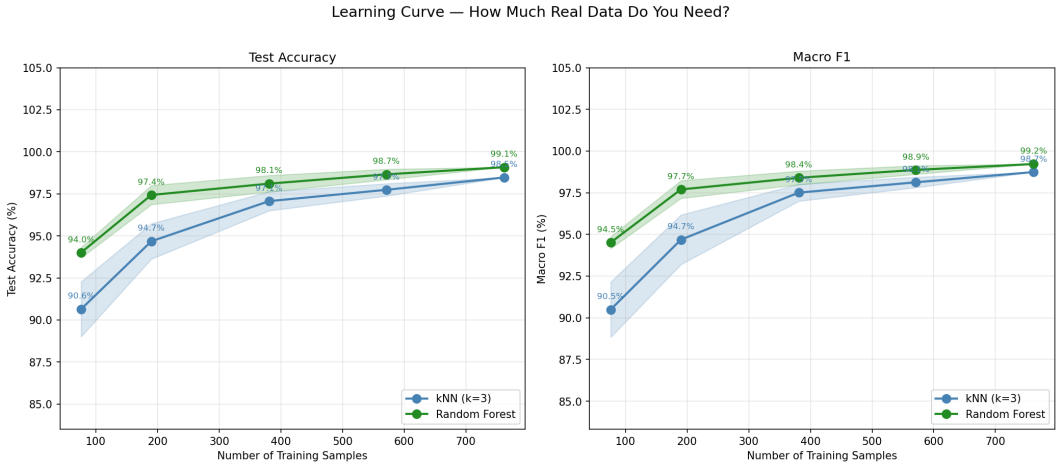
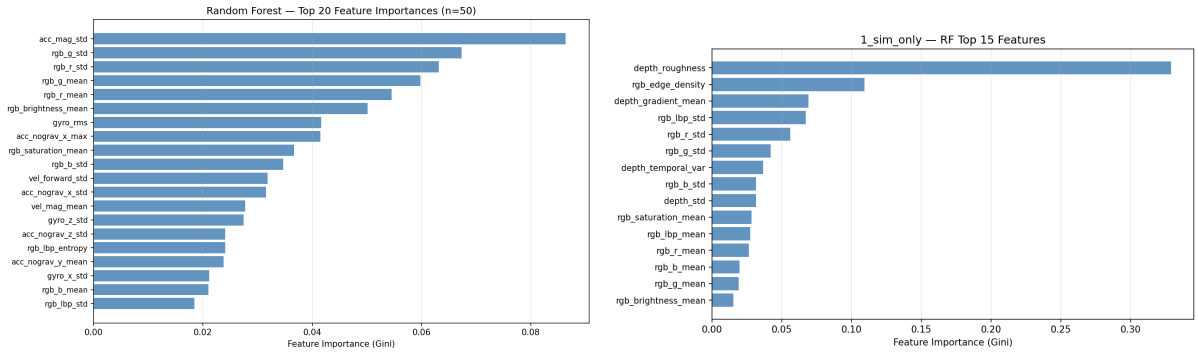


Figure 5: Learning curve showing accuracy vs. number of training samples.

E Feature Importance

Feature importance analysis for Random Forest, as shown in Figure 6, continues to display the domain gap. The real-world data (Figure 6a) has the standard deviation of acceleration magnitude as its most important feature, denoting that terrain-specific vibration patterns are the most prominent classification signal on real hardware. The other top features are a combination of RGB color variance (`rgb_g_std`, `rgb_r_std`) and appearance (`rgb_saturation_mean`, `rgb_brightness_mean`), implying that visual texture supports classification in the real-world. The simulation data, represented in (Figure 6b), has depth and RGB features as the top identifiers. The feature importance analysis directly reflects the ablation analysis and provides extra context into why the sim-trained model fails on real data.



(a) Real-world model

(b) Simulation model

Figure 6: Random Forest feature importances.

V Discussion

A Root Causes of the Sim-to-Real Gap

To understand why the sim-to-real transfer model accuracy performed significantly worse than the other experiments (57.78%), a feature distribution analysis was conducted to compare the importance of each feature for the shared class for both datasets. This resulted in three root causes being identified, all evaluated using the normalized shift between domain means with simulated-based standard deviations (σ).

1. Velocity features are semantically different (9.1σ average shift). The TartanGround data, produces a ground-truth body velocity `vel_forward_std` with a mean of 0.073, characterizing the variation in speed of the simulated robot during the trial. On Jango, the body velocity column uses the forward acceleration standard deviation as a substitute, with a mean of 1.055. In addition, `vel_lateral_mean` is 0.047 in simulation, while on the real robot the value is an order of magnitude higher at 1.026. While these feature columns share the same name, they are representing different physical values.

2. Depth sensors operate at completely different scales (1,025σ average shift). Simulation data from TartanGround renders the depth images at near-ground level with a `depth_mean` of 0.099m. The time-of-flight camera used for depth data is affixed to the robots head, viewing the ground at approximately 0.7 m range with a `depth_mean` of 0.704m. Moreover, the depth sensor's resolution (640×640 rendered vs. 240×180 ToF reprojected), field of view, and noise characteristics make comparison even more difficult due to the differences. Training a model on simulation depth data learns that grass has a depth of about 0.10m, a feature that never occurs in the real-world data.

3. Simulated IMU does not capture real vibration dynamics (2.8σ average shift). Simulated IMU data from the TartanGround dataset produces a smooth, low magnitude accelerations with a `acc_z_std` of 0.55 and a `gyro_z_std` of 0.039. The real-world data suffers from servo vibration, ground reaction forces, and other noisy signals resulting in a `acc_z_std` of 1.61 and a `gyro_z_std` of 0.47. This explains why the IMU modality produced worse results during ablation (67.22%) as the simulation does not create terrain specific vibrations, making the sim-to-real transfer much less effective.

B The Ablation–Transfer Connection

The ablation analysis further explained the transfer failure, indicating what features were prioritized during training for both datasets. Simulated IMU data proved to be far less valuable in Random Forest training, with RGB and depth features dominating earning 98.56% and 97.05% respectively for single modality analysis. Deploying the model on real-world data revealed those depth and RGB features have less importance, with IMU becoming the top feature achieving 95.11%. Along with this, the RGB and depth sensors' physical placement, environment lighting and hardware all together result in a structural mismatch, leading to the model to depend on the wrong features between simulation and real-world data.

C kNN vs. Random Forest

Throughout training and testing, Random Forest consistently performed better than kNN across all experiments. The gap is very apparent in the learning curve analysis with Random Forest achieving 94.01% compared to kNN's 90.64% with only 76 training samples (Table 5). While Random Forest is able to handle class imbalance, such as wood in this research, kNN has no such attribute, further explaining these results. Despite excellent in-domain accuracy, both models performed poorly for the sim-to-real transfer experiment, however Random Forest still beat out kNN (57.78% vs. 56.67%). Random Forest also provides the Gini impurity (Eq. 3), a metric required for the ablation analysis to determine feature importance. With the 48 dimensional feature space, kNN suffers from the curse of dimensionality where the Euclidean distance (Eq. 1) becomes less discriminative as dimensions increase.

D Limitations

There are several limitations in this research that need to be addressed. First, the carpet and wood samples were collected in the same apartment, hurting generalization to different carpet and wood textures. Second, class imbalance is apparent with wood terrain only having 90 windows while pavement has 306 windows. Class weighting served to assist in this imbalance but not eliminate it. Third, samples were collected with a 50% overlap with other samples, leading to a reduction in temporal independence. Additionally, grass/dirt and pavement were collected in similar weather and lighting conditions, making redundant data collection and sample similarity an important limitation to note. Finally, the system classifies fixed duration windows not accounting for terrain transition that appear mid-window.

VI Conclusions

This research developed an end-to-end multi-modal terrain classification system on a custom quadruped robot, with automated ROS2 data collection, simulation and real-world trial feature extraction and comparative model evaluation. The sensor fused 48 dimensional Random Forest classifier reached 99.08% test accuracy across five terrains from real-world data collection, with kNN following closely at 98.47%.

The most significant finding in the experiments is the quantifiable domain gap between simulation and real-world data, in-domain models achieving 90%–99% accuracy while the sim-to-real transfer model only

reaches 57.78%. Feature distribution analysis resulted in three root causes being found:

- Semantic differences in velocity features
- Depth sensor scale mismatch
- Absence of realistic IMU vibration dynamics in simulation

The feature ablation study with single sensor modality testing provides more context into the domain gap, with simulation data prioritizing depth (97.05%) and RGB (98.56%) features while the real-world model prioritizes IMU (95.11%) and RGB (93.88%) features. This inversion of sensor priority continues to contextualize the sim-to-real transfer failure as each data modality relies on different features to be classified.

Conducting a learning curve analysis demonstrated that terrain classification is a practical deployment, requiring a minimal amount of samples before achieving high accuracy score. Random Forest performing at 94% accuracy with only 76 training samples (about 5 minutes of walking data). This paired with the IMU only classification reaching 95.11% means real-world data can be collected with a minimal hardware setup.

Future work for this project will be to close the environmental sensing loop by deploying the trained classifier and giving it access to Jango's gait parameters to allow for dynamic tuning to different terrains. The Random Forest algorithm would allow for a lightweight real-time ROS2 node that would grant Jango adaptive gait control, turning the robot into an environmentally aware and adaptive system.

References

[1] W. Wang *et al.*, "Tartanground: A large-scale dataset for terrain classification in robotics." `https://tartanair.org/tartanground/`, 2023. Accessed: 2026-04-18.

[2] L. Wellhausen *et al.*, "Terrain classification for legged robots using proprioceptive sensing," *Journal of Field Robotics*, vol. 37, no. 6, pp. 969–984, 2020.

[3] J. Kim *et al.*, "Vision-based terrain classification for mobile robots in construction environments," in *Proceedings of the International Symposium on Automation and Robotics in Construction*, 2019.

VITA

Luke Stevenson is graduate student at Northeastern University pursuing a M.S. in Robotics, expected to graduate in 2027. He earned a Bachelor of Science in Computer Science degree from the University of Denver in 2025. While at the University of Denver, he published a data security paper to IEEE BuildSec and worked as a software engineer at Dreamface Technologies, a robotics company. His interests include robotics, AI/ML, embedded systems and environment engineering.

APPENDIX

Appendix A: Feature Descriptions

Table 6: Complete list of 48 features by sensor modality.

Modality	Feature Name	Description
IMU (29)	acc_nograv_x_mean	Mean X-axis acceleration (gravity removed)
	acc_nograv_x_std	Std dev of X-axis acceleration
	acc_nograv_x_min	Min X-axis acceleration
	acc_nograv_x_max	Max X-axis acceleration
	acc_nograv_y_mean	Mean Y-axis acceleration
	acc_nograv_y_std	Std dev of Y-axis acceleration
	acc_nograv_y_min	Min Y-axis acceleration
	acc_nograv_y_max	Max Y-axis acceleration
	acc_nograv_z_mean	Mean Z-axis acceleration
	acc_nograv_z_std	Std dev of Z-axis acceleration
	acc_nograv_z_min	Min Z-axis acceleration
	acc_nograv_z_max	Max Z-axis acceleration
	acc_rms	RMS of acceleration magnitude
	acc_mag_std	Std dev of acceleration magnitude
	acc_mag_peak	Peak acceleration magnitude
	gyro_x_mean	Mean X-axis angular velocity
	gyro_x_std	Std dev of X-axis angular velocity
	gyro_y_mean	Mean Y-axis angular velocity
	gyro_y_std	Std dev of Y-axis angular velocity
	gyro_z_mean	Mean Z-axis angular velocity
	gyro_z_std	Std dev of Z-axis angular velocity
	gyro_rms	RMS of gyroscope magnitude
	vel_forward_mean	Forward velocity mean (sim: ground truth; real: acc proxy)
	vel_forward_std	Forward velocity std dev
	vel_lateral_mean	Lateral velocity mean
	vel_mag_mean	Velocity magnitude mean
	dominant_freq_hz	Dominant FFT frequency of acc magnitude
	dominant_freq_power	Power at dominant frequency
	spectral_entropy	Normalized spectral entropy of acc magnitude
Depth (7)	depth_mean	Mean depth value (metres)
	depth_std	Std dev of depth values
	depth_min	Minimum depth value
	depth_near_ratio	Fraction of pixels < 1 m
	depth_roughness	Std dev of Laplacian response
	depth_gradient_mean	Mean Sobel gradient magnitude
	depth_temporal_var	Temporal variance across frames

RGB (12)	rgb_r_mean	Mean red channel (normalized 0–1)
	rgb_r_std	Std dev of red channel
	rgb_g_mean	Mean green channel
	rgb_g_std	Std dev of green channel
	rgb_b_mean	Mean blue channel
	rgb_b_std	Std dev of blue channel
	rgb_brightness_mean	Mean HSV value channel
	rgb_saturation_mean	Mean HSV saturation channel
	rgb_lbp_mean	Mean of LBP texture descriptor
	rgb_lbp_std	Std dev of LBP texture descriptor
	rgb_lbp_entropy	Entropy of LBP histogram
	rgb_edge_density	Fraction of Canny edge pixels

Appendix B: Full Classification Reports

Real-only 5-class (Random Forest, test set):

	precision	recall	f1-score	support
carpet	1.000	0.983	0.992	60
grass_dirt	1.000	0.971	0.986	70
pavement	0.968	1.000	0.984	92
tile	1.000	1.000	1.000	78
wood	1.000	1.000	1.000	27
accuracy			0.991	327
macro avg	0.994	0.991	0.992	327
weighted avg	0.991	0.991	0.991	327

Real-only 5-class (kNN, test set):

	precision	recall	f1-score	support
carpet	1.000	1.000	1.000	60
grass_dirt	0.971	0.957	0.964	70
pavement	0.968	0.978	0.973	92
tile	1.000	1.000	1.000	78
wood	1.000	1.000	1.000	27
accuracy			0.985	327
macro avg	0.988	0.987	0.987	327
weighted avg	0.985	0.985	0.985	327

Sim→Real transfer (Random Forest, test on real):

```
              precision    recall  f1-score   support

  grass_dirt      0.506     1.000     0.672       234
    pavement      1.000     0.255     0.406       306

    accuracy                        0.578       540
   macro avg      0.753     0.627     0.539       540
weighted avg      0.786     0.578     0.522       540
```

Sim→Real transfer (kNN, test on real):

```
              precision    recall  f1-score   support

  grass_dirt      0.000     0.000     0.000       234
    pavement      0.567     1.000     0.723       306

    accuracy                        0.567       540
   macro avg      0.283     0.500     0.362       540
weighted avg      0.321     0.567     0.410       540
```

Appendix C: Key Code Snippets

IMU feature extraction (`features/imu.py`):

```
def imu_features(acc, gyro, sample_rate_hz=10.0,
                 remove_gravity=False):
    """
    Extract statistical and frequency features from
    a window of IMU data.

    Parameters
    ----------
    acc  : (N, 3) float32 - accelerometer in body frame
    gyro : (N, 3) float32 - gyroscope, bias-corrected
    sample_rate_hz : IMU sample rate
    remove_gravity : subtract mean-Z as gravity estimate

    Returns: dict with 29 feature keys
    """
```

Depth feature extraction (`features/depth.py`):

```
def depth_features(depth_frames):
    """
    Extract surface geometry features from a window
    of depth frames.

    Parameters
    ----------
    depth_frames : list of float32 arrays in metres

    Returns: dict with 7 feature keys
        depth_mean, depth_std, depth_min,
        depth_roughness, depth_gradient_mean,
        depth_near_ratio, depth_temporal_var
    """
```

RGB feature extraction (`features/rgb.py`):

```
def rgb_features(images):
    """
    Extract texture and colour features from a window
    of RGB frames. Uses middle frame as representative.

    Parameters
    ----------
    images : list of BGR numpy arrays

    Returns: dict with 12 feature keys
        rgb_r/g/b_mean, rgb_r/g/b_std,
        rgb_brightness_mean, rgb_saturation_mean,
        rgb_lbp_mean, _std, _entropy,
        rgb_edge_density
    """
```

Project repository structure:

```
TerrainDetection/
  config/              # jango_config.json, tartan_config.json
  features/            # Shared feature extraction
    imu.py, depth.py, rgb.py
  jango_scripts/       # Real-world pipeline
    extract_features_jango.py
    ros2/src/terrain_trial_script/
```

```
    record_terrain_trial.py
  data/                # terrain_dataset/, features_realworld.csv
tartan_scripts/      # Simulation pipeline
  extract_features.py
  data/                # features_simulation.csv
trainer/             # Model training scripts
  train_baseline.py
  train_cross_dataset.py
  train_ablation.py
  train_learning_curve.py
results/             # All plots, confusion matrices, summaries
```