# INTEGRATED MACHINE LEARNING APPROACH FOR TERRAIN ANALYSIS AND PATH OPTIMIZATION IN QUADRUPED ROBOTS: K-NN CLASSIFICATION, K-MEANS CLUSTERING AND Q-LEARNING ON DIVERSIFIED SOIL TYPES

Karen Valeria Villanueva Novelo, 2009146@upy.edu.mx; Victor Alejandro Moo Quintal, 2009098@upy.edu.mx; Angel Huerta, 2009071@upy.edu.mx; Esteban Rodríguez, 2009116@upy.edu.mx; Joshua Zamora Ramírez, 1909191@upy.edu.mx.
Professor: Victor Alejandro Ortiz Santiago, victor.ortiz@upy.edu.mx.

## I. Introduction

THE exploration and navigation of diverse terrains, particularly in remote or extraterrestrial environments, present significant challenges for robotic systems. Legged robots, with their dynamic and adaptable locomotion capabilities, appear as potential solutions in these demanding contexts. Their ability to walk through complex landscapes, from fine-grained, granular terrains to uneven rocky surfaces, is crucial, especially in scenarios where wheeled robots may face limitations as noted by Kolvenbach et al. [4]. The ability of these robots to navigate different surfaces safely and efficiently is essential for tasks like planetary exploration and search and rescue operations.

The key to enhancing the functionality of legged robots lies in the effective utilization of sensor data. Sensors attached to robotic legs can provide invaluable information about the terrain, enabling these robots to adapt their movement strategies in real-time. This adaptability is essential for safe and efficient navigation, particularly in unpredictable environments like those found on other planets. The data gathered from these sensors, encompassing various force, torque, and inertial measurements, can be leveraged to classify different types of terrain. These classifications, in turn, inform the robot's decision-making process, aiding in the selection of the most suitable paths and movement patterns.

However, the processing of sensor data presents its own set of challenges, primarily due to its complexity and high dimensionality. Machine learning algorithms, particularly K-Nearest Neighbors (KNN), have emerged as powerful tools for analyzing this data. KNN is recognized for its ability to handle multi-dimensional datasets and adapt to varying data distributions, as discussed by Tokuç [10]. The selection of the appropriate machine learning algorithm is critical, as it needs to balance computational efficiency with accuracy, especially for real-time applications.

In this project, the focus is on exploring various machine learning models to classify terrain types using sensor data from legged robots, with particular attention given to the K-Nearest Neighbors (KNN) algorithm. This algorithm was selected for its proficiency in handling multi-dimensional datasets and its flexibility with diverse data distributions. The reasons for choosing KNN are discussed, including its capability to process sensor data effectively in real-time, a critical requirement for autonomous robotic navigation.

Furthermore, the preprocessing of sensor data is addressed, highlighting the importance of standardization and cleaning to ensure the accuracy and reliability of the machine learning models. Such preprocessing is crucial in managing the inherent variability and complexity of sensor readings, which may include noise and outliers.

This project contributes to the ongoing development of legged robots, particularly in navigating challenging terrains. It aims to demonstrate how the intelligent processing of sensor data can enhance the adaptability and decision-making capabilities of these robots, potentially leading to more effective and versatile robotic systems in various applications, including planetary exploration and terrestrial search and rescue missions. Additionally, this work adds to the broader field of robotics by showcasing the potential of machine learning in improving the interaction between robots and their environments.

### A. Theorethical framework

*1) Terrain recognition:* Terrain recognition is a critical aspect of robotic navigation and efficiency. The ability to identify and adapt to different terrain types impacts several key areas:

Fuel Efficiency and Energy Conservation: The identification of terrain types facilitates adjustments in speed and movement, optimizing fuel or energy consumption. This aspect is particularly relevant for the ANYmal robot, as efficient locomotion extends its operational autonomy, as noted by Vangen et al. [2].

Reduced Vehicle Wear and Safety: The understanding of terrain conditions aids in circumventing surfaces that could lead to wear or pose safety hazards. In the case of ANYmal, this translates to an extended operational lifespan and the avoidance of potentially damaging scenarios.

Travel Time and Obstacle Avoidance: The selection of routes with more favorable terrain can considerably diminish travel time. For a quadruped robot like ANYmal, this entails not just pinpointing the most straightforward path but also strategizing routes that bypass obstacles, a concept underlined in the findings of Kozlowski and Walas [1].

Stress Reduction and Specific Tasks: Easier navigation leads to reduced strain on the robot's mechanical systems. In specialized scenarios, such as search and rescue operations,

the recognition of terrain is crucial for task planning and execution, as highlighted by Vangen et al. [2].

Autonomous Driving and Interaction with the Environment: In the realm of autonomous navigation, the real-time recognition of terrain is indispensable. It enables ANYmal to make informed decisions, adjust its movements, and effectively interact with its environment, including overcoming steps or circumnavigating obstacles in various settings, as demonstrated by research [1][2].

The integration of advanced sensors and AI for real-time terrain analysis, as emphasized by Kozlowski and Walas [1] and Vangen et al. [2], is crucial for the operational success of robots like ANYmal.

*2) Quadruped ANYmal robot:* ANYmal is a four-legged robot that has been methodically designed for independent operation in challenging and complex conditions. Driven by specifically designed torque-controllable actuators, the robot platform demonstrates the ability to climb and run dynamically across difficult terrain. Using depth cameras and LIDAR improves ANYmal's ability to sense its surroundings, which enables it to navigate with greater accuracy, autonomously determine its location, and select its footing with caution. This combination of cutting-edge capabilities allows ANYmal to function with flexibility and agility in most challenging circumstances [3].

ANYmal is designed to do certain jobs in commercial and industrial contexts, especially in demanding conditions like search and rescue missions or oil and gas platforms. The robot can travel and function in challenging environments by utilizing its ability to perceive its surroundings. Driven by incredibly accurate actuators, ANYmal exhibits the capacity to perform dynamic running gaits [4].

ANYmal can carry an additional cargo of up to 15 kg and has an ideal size that allows it to navigate and overcome obstacles with ease. The robot, a battery charger, and a single laptop are all that are needed for a simple platform setup. ANYmal has an integrated battery that allows it to operate continuously for two to four hours, depending on the type of activities. Is water and dust proof, with an IP67 rating, specifically built for real-world applications. Its impact-resistant, ruggedized casing adds to its dependability and durability in a variety of operating conditions. [3] The compact version of ANYmal weighs less than 30 kg, yet it possesses the capacity to transport a range of equipment, including batteries, optical and thermal cameras, microphones, dynamic lighting systems, and gas detection sensors. ANYmal's adaptable payload capabilities and lightweight construction make it an excellent choice for a range of applications in tough and dynamic conditions [4].

A combination of carbon fiber and aluminum is used in the construction of the main body and leg parts of ANYmal in order to produce a lightweight design. The onboard batteries, which weigh 3 kg and have a capacity of about 650 Wh, provide power for more than two hours of independent use. Leg padding and a protective frame help to minimize damage in the event of a fall, making deployment and transportation easier. The robot can interact with its environment more effectively because to the tactile feet that optoforce sensors provide. Furthermore, revolving Hokuyo UTM-30LX sensors help create a thorough 3D perception of the surroundings. A modular pan-tilt head with different sensory payload capabilities may be easily installed onto the robot platform to improve ANYmal's flexibility in a variety of settings. ANYmal may be tailored for various purposes and sensory needs because to its modular architecture [5].

According to Kolvenbach [6], ANYmal employs point feet that are composed of an elliptical nitrile rubber (NBR) sole that is reinforced by 15 mm of memory foam (Poron XRD) in order to reduce peak loads upon impact. On hard surfaces, the diameter of the foot usually generates an area of $8\,\mathrm{cm}^2$, which increases to $28\,\mathrm{cm}^2$ when the foot is completely submerged in compressible terrain. These feet, which weigh 325 g total weight (including the shank), have a 6-Axis Force/Torque (F/T) sensor made in-house that can sample data at 400 Hz. Although this foot design works well in compacted soils, it has drawbacks in loose, loose soils because of greater sinkage.

*3) Q-learning:* Q-learning is a well-researched technique that helps an agent learn the optimal behaviors in a dynamic environment. Q-learning is independent of previous environmental knowledge and belongs to a class of algorithms called model-free algorithms. Reinforcement learning (RL) is a branch of machine learning that focuses on teaching agents how to make choices and act in a way that maximizes their total accumulation of rewards. Fundamentally, reinforcement learning is based on the idea that agents learn concepts by interacting with their surroundings and receiving feedback in the form of incentives or punishments for their behavior [1]. Q-learning is one of the most popular approaches in reinforcement learning, allowing agents to repeatedly explore and find the optimal actions. This method makes use of a value function known as the Q-function, which represents the expected overall reward linked to carrying out a particular activity in a particular state. Q-learning enables agents to gradually improve their decision-making skills through ongoing updates to this Q-function, resulting in better decisions over time. Q-learning models work in an iterative process where different parts work together to make model training easier. The agent learns through environmental exploration in this iterative process, and the model is updated concurrently with the exploration. The main components of Q-learning are Q-values, Agents, States, Actions, Rewards, and Episodes. The goal of Q-learning's numerous additions and enhancements is to address the intricacies of the real world. An example of this improvement is dealing with large state and action spaces by using function approximation techniques, like neural networks, instead of a Q-table. This modification, referred to as deep Q-learning, has demonstrated exceptional performance, especially in domains with complex games. Q-learning is a powerful algorithm that allows agents to learn the best course of action by striking a balance between exploration and exploitation. By continuously updating Q-values, agents gradually get closer to an ideal strategy that maximizes cumulative rewards. Because of its versatility and ability to learn without prior information, Q-learning has become a key component of reinforcement learning and has applications in a wide range

of fields, including robotics and gaming [1].

## II. Problem Statement

THIS research project addresses the challenge of developing a practical and efficient method for classifying different types of terrain using sensor data collected from legged robots. This problem is essential in the broader goal of improving the mobility of legged robots in various challenging environments, such as those found in extraterrestrial or complex terrestrial landscapes.

The primary aim is to create a machine learning model that can accurately identify different terrains based on the data obtained from the sensors on robotic legs. These sensors provide complex, high-dimensional data, including force, torque, and inertial measurements. Processing and effectively using this data is a significant challenge, given its complexity and the need for real-time analysis in autonomous robotic navigation.

A crucial aspect of this project is the preprocessing of the sensor data. This involves cleaning and standardizing the data to make it suitable for machine learning applications. The raw sensor data often contains variability, noise, and outliers, which must be managed to ensure accurate and reliable model training and prediction.

Another key task is evaluating and optimizing the performance of the algorithm and other machine learning models such as supervised, unsupervised and reinforcement learning used in this project. The goal is to find a balance between computational efficiency and the accuracy of terrain classification, especially important in scenarios where quick decision-making is crucial.

The project aims to enhance the adaptability and decision-making capabilities of legged robots. By improving how these robots interpret and respond to different terrains, the project contributes to the development of more capable and versatile robotic systems. These advancements could be particularly beneficial in areas like planetary exploration and search and rescue missions, where robots must navigate through diverse and often unpredictable environments.

## III. Methods and tools

### A. Proposed solution

The problem of classifying different terrains for legged robots was approached by developing a machine learning model that processes sensor data to identify various ground types. The solution involved several key steps:

Firstly, the data collected from the robot's sensors was standardized. This means it was cleaned up and adjusted to ensure consistency, which is important for the machine learning model to work properly.

Secondly, clustering algorithms were used to sort the data into groups without needing examples to learn from. This helped the robot understand different terrains it might encounter.

Then, a supervised learning method called K-Nearest Neighbors was applied. This method used examples to teach the robot how to recognize different terrains. It was chosen for its simplicity and effectiveness, making it a good fit for the robot's needs.

Lastly, a technique called Q-learning allowed the robot to learn from experience by trying different paths and learning from the results. It was observed that over time, the robot got better at finding its way, which was shown by it getting more rewards and taking fewer steps.

By evaluating and tweaking the model as needed, it was ensured that the robot could make quick and accurate decisions, which is very important for tasks where time is critical, like rescue operations.

### B. Dataset Used

The dataset for this project was primarily extracted from the work by Kolvenbach et al. [7], which includes diverse sensor readings using two types of robotic legs with different properties and sensors attached. These sensors were tested on various types of materials, created to simulate the terrain of different types of objects. However, for this project, only their properties were considered. The materials include very fine-grained, porous, and highly compressible dust (ES-1), very fine sand (ES-2), medium-coarse sand (ES-3), and solid rock (Bedrock). Variations and mixtures used were coarse sand (CS), bedrock covered in ES-2, and rocks on ES-3.

The focus of this project was on the 'Pointfoot_dataset', containing various preprocessed readings from the Pointfoot type leg sensors, as used in the ANYmal quadruped robot model. The ANYmal's robot leg consists of a nitrile rubber (NBR) sole of elliptical shape, supported by 15 mm of memory foam (Poron XRD) to reduce peak loads during impact, and integrates six 6-Axis Force/Torque sensors, sampling at 400 Hz [7].

A total of 2600 impacts were recorded with the testbed, and an additional 240 impacts with the ANYmal robot. The minimal post-processing of the acquired foot sensor data involved identifying the impact peak and extracting the raw impact oscillation, resulting in a signal length of 200 sample points or 0.5 s per impact [7].

The classification approach that was used in the mentioned study for analyzing this dataset is based on the analysis of oscillations resulting from a controlled impact on the soil, sensed by Force/Torque and IMU sensors in the feet.

### C. Preprocessing

The dataset, obtained in matrix format (.mat files), was composed of nine files named 'data_all_...', each corresponding to different types of Martian soil simulants [7]. These simulants included ES-1, ES-2, ES-3, Bedrock, Coarse Sand (CS), and variations like Bedrock embedded in ES-2 and Rocks on ES-3. The material ES-4 was excluded due to its minimal observations compared to other materials, creating an unbalanced dataset. Also, ES-4's contribution to the study was not considered critical as it was a variation of ES-3.

The preprocessing involved merging seven of these files: 'data_all_cs', 'data_all_bedrock', 'data_all_es1', 'data_all_es2', 'data_all_es3', 'data_all_pebbles_in_es3', and

'data_all_bedrock_embedded_es2' [7]. Each file's data were individually labeled post-merging. The terrains were then manually classified and ranked according to their optimally for the movement of the quadruped robot. The classification was as follows:

- Bedrock (New Label 7): Solid, rough limestone, most optimal for the robot.
- Bedrock covered in ES-2 (New Label 6): Stable limestone with a layer of fine sand.
- ES-3 (New Label 5): Gravelly medium-coarse sand, moderately complex terrain.
- Coarse Sand (CS) (New Label 4): Variable and challenging terrain.
- ES-2 (New Label 3): Very fine sand, significant challenges in traversal.
- Rocks on ES-3 (New Label 2): Uneven terrain with volcanic rocks.
- ES-1 (New Label 1): Highly compressible dust, least optimal terrain.

The dataset files contained large matrices with each matrix representing sensor signals for a specific soil type. For instance, the Pointfoot_dataset's 'data_all_cs.mat' file included a matrix (35000 x 6) representing 175 impacts with six columns for Force/Torque data (Fx, Fy, Fz, Tx, Ty, Tz) [7]. These matrices, consisting of rows corresponding to stacked impacts (200 rows per impact), encapsulated the detailed sensor data necessary for the study.

The preprocessing included standardizing the sensor data from all soil types. This step was important for several reasons:

1) **Scaling Features**: The sensor measurements, like force and torque, were in different units and had a wide range of values. Standardizing these measurements brought them to a similar scale. This is important for models that use distance calculations, ensuring that no single type of measurement has too much influence.
2) **Normalizing Data**: The data varied a lot in its distribution, with some values being much more common than others. By standardizing (making the data have an average of 0 and a standard deviation of 1), we made the data more regular. This helps make the process of training models more straightforward and effective.
3) **Reducing Impact of Outliers**: Outliers, or very unusual values, can affect the average and spread of the data. Standardizing the data made these outliers less influential, leading to a more balanced and fair training process for the models.

After standardization, it was observed that duplicate data was removed. Duplicates could have resulted from repeated measurements and could have caused bias in the training. Their removal ensured a more accurate and general understanding by the models.

This initial dataset was made more consistent and seemed suitable for machine learning techniques. It was representative of the different terrains, which was important for the development of models that predict how the robot will perform on various types of soil. However, it was decided to create an additional dataset using clustering techniques to introduce terrain variability into the project. While the original dataset provided basic categorizations of terrains, it lacked the complexity needed to simulate the diverse environments a legged robot might encounter.

### D. Data Analysis Tools

For preprocessing and analyzing the dataset, we used several tools:

- **MATLAB**: For initial data extraction and manipulation from .mat files.
- **Python and Pandas Library**: For cleaning, transforming, and standardizing the data. Pandas is good for working with large datasets.
- **Scikit-learn**: For standardizing data and developing machine learning models. Its tools are helpful for preprocessing and model training.
- **Jupyter Notebooks**: For an interactive development environment, allowing us to explore, visualize, and document the data analysis process.

These tools together made it easier and more effective to handle, preprocessing, and analyze the complex dataset, preparing it for machine learning analysis.

## IV. Development

### A. Non-supervised model: K-means

*1) Model selection:* The previous dataset then was passed to the clustering process using the K-Means algorithm. This unsupervised machine learning technique was employed to identify patterns within the sensor data for each soil type, dividing them into distinct subclasses. These subclasses represented variations within each soil type, providing a more granular understanding of the terrain.

**Segmentation:** The dataset was first segmented based on soil type, creating separate groups for each terrain like Bedrock, ES-2, and Coarse Sand, among others.

**K-Means Clustering:** For each soil type segment, the K-Means algorithm was applied to cluster the data into three subclasses. This division aimed to capture the subtle variations within each soil type, which might influence the robot's navigation.

**Centroid Calculation:** Post-clustering, centroids for each subclass were calculated. These centroids represented the average sensor reading profile for each subclass, serving as a reference point for further analysis.

*2) Ranking and Composite Labeling:* Each subclass's centroid was evaluated for its proximity to the Bedrock centroid, considered the most optimal terrain. This measurement provided a ranking of subclasses based on their similarity to the optimal Bedrock terrain.

To facilitate the use of this dataset in a reinforcement learning context, composite labels were created for each soil type and subclass combination. These labels, like "Bedrock_subclass_1" provided a unique identifier for every distinct terrain type encountered in the dataset.

*3) Final Dataset:* The final dataset consisted in information about the terrain variations and their potential impact on the robot's movement. It became a crucial component in training a reinforcement learning algorithm, where the robot learns to navigate efficiently by selecting paths through optimal terrains. The dataset's structure and composition directly supported the development of intelligent decision-making capabilities in the robot simulation, enhancing its adaptability and effectiveness in navigating diverse terrains.

### *B. Supervised Learning model: K-NN*

*1) Model selection:* The dataset includes eight key features: Fx, Fy, Fz, Tx, Ty, Tz, Subclass, and Distance_To_Bedrock_Centroid. These represent various sensor readings and additional parameters, offering a comprehensive view for terrain categorization. This diversity in features underscores the complexity and intricacies necessary for effective terrain classification.Preliminary analysis of the dataset indicated a diverse range of values across features, suggesting complex relationships and interactions essential for terrain categorization.

Given the task's classification nature, several algorithms were tested in order to select the best one:

Decision Tree: The application of the Decision Tree algorithm on the provided dataset resulted in an accuracy of 100.0%. This unusually high accuracy suggests that the model may be overfitting the data. In practical scenarios, especially with complex datasets, achieving perfect accuracy is rare and often indicates that the model is too closely tailored to the specific patterns of the training data, potentially impacting its ability to generalize to new, unseen data.And that is a feature in Decision Trees algorithms, they tend to overfit in complex datasets. [8]

K-Nearest Neighbors (KNN): The application of the K-Nearest Neighbors (KNN) algorithm, with k=5, on the dataset resulted in an accuracy of approximately 96.07%. This result highlighted KNN's capability to handle the dataset's multidimensional nature effectively. [9]

In the same way, it was taken into account the model's computational efficiency, especially for real-time applications, as the robot will be doing. And in this case, the response time for training the model and making predictions with the K-Nearest Neighbors (KNN) algorithm was approximately 1.44 seconds.

*2) Dataset preprocessing:* In the preprocessing of the dataset for the application of the K-Nearest Neighbors (KNN) algorithm, a crucial step involved transforming the target variable, 'Composite_Label', from a text format into a numerical one. This transformation is essential because KNN, an algorithm based on distance metrics, requires numerical input to calculate similarities between data points. The chosen method for this conversion was label encoding, which assigns a unique integer to each distinct label in the 'Composite_Label' column. This process ensures that the algorithm can effectively interpret and process the target variable.

Additionally, the decision to drop the 'Subclass' and 'Soil_Type' columns from the dataset was made based on specific considerations. The 'Subclass' column, which is closely related to the 'Composite_Label', could introduce a redundancy in the data, potentially leading to biased or skewed results in the KNN algorithm. Similarly, the 'Soil_Type' column, being a categorical variable, would require conversion into a numerical format for use in KNN. However, including this feature could introduce unnecessary complexity and potential noise, especially if the number of unique soil types is high. Removing these columns helps streamline the dataset, focusing the algorithm on the most relevant features and reducing the risk of overfitting or underperformance due to irrelevant or redundant information. This approach enhances the efficiency and accuracy of the KNN model, allowing it to better generalize and predict on new, unseen data.

In the same way, in machine learning, a common practice is to split the dataset into a training set and a testing set. This approach allows the model to learn from a portion of the data (the training set) and then be evaluated on a separate portion (the testing set) that it hasn't seen before. For this dataset, an 80-20 split is employed, meaning that 80% of the data is used for training the KNN model, while the remaining 20% is reserved for testing. This split strikes a balance between having enough data to train the model effectively and enough data to test and validate its performance accurately. The training set provides a robust foundation for the model to learn the underlying patterns and relationships, while the testing set offers a reliable means of assessing the model's generalization capabilities and predictive accuracy on new, unseen data. This methodology is crucial for evaluating the performance of the KNN algorithm and ensuring that it can make accurate predictions in real-world scenarios.

*3) KNN Hyperparameter Tuning:* One of the most important things to consider when optimizing the KNN algorithm is the fine-tuning of the 'k' value. This parameter determines how many nearest neighbors influence the classification of each data point.

It was selected the KNN algorithm with k=5 because lower values of k can make the model more sensitive to noise in the data, potentially leading to overfitting. On the other hand, slightly higher values like 5 might provide better generalization by considering more neighbors.

*4) KNN selection:* One of the primary advantages of the KNN algorithm was its adaptability to high-dimensional datasets. The dataset in question comprised sensor readings across multiple axes, resulting in a multi-dimensional feature space. KNN's inherent ability to handle such multi-dimensional data made it a really good choice for this specific application. [11]

KNN is a non-parametric algorithm, which implies that it does not assume any specific data distribution. This characteristic was considered an advantage for the work, given the diverse and complex nature of terrain data. It enabled the algorithm to flexibly adapt to varying terrain conditions. KNN's hyperparameter 'k,' representing the number of nearest neighbors considered for classification, offered the possibility for optimization of the code. Through systematic tuning, the optimal 'k' value (k = 5) was chosen, determining a balance

between model accuracy and generalization. [11]

Real-time processing requirements were a fundamental consideration for the terrain classification task, given the robotic application. KNN's computational efficiency, as evidenced by its training and prediction times, made it compatible with real-time development, ensuring timely terrain recognition and response. [12]

The KNN algorithm makes decisions based on the local neighborhood of data points. This feature aligns with the task of terrain classification, where the characteristics of adjacent sensor readings often provide important insights into the terrain type. The way the algorithm uses closeness to make decisions was good in this case, due to the small changes that can be presented in the different terrains the sensors will be reading. [13]

The selection of the KNN algorithm for terrain classification was driven by its inherent characteristics, compatibility with the dataset's multi-dimensional characteristics, adaptability to non-parametric data, and its ability to make local decisions based on proximity. The fine-tuning of the 'k' hyperparameter, along with its computational efficiency, made KNN the optimal choice for this work. Consequently, KNN exhibited compatibility and suitability for achieving accurate and real-time terrain classification in the context of the four-legged robot application which obtains data with different sensors.

### C. Reinforcement Learning model: Q-Learning

In the development of an autonomous navigation system for a legged robot, a Q-learning based reinforcement learning approach was implemented.

*1) Model selection:* Q-learning, a form of model-free reinforcement learning, was selected for its ability to learn optimal action-value functions in an environment with stochastic transitions and rewards. This approach is well-suited for scenarios where the robot must make decisions without prior knowledge of the environment's dynamics. [6]

**Simulation Environment:** The environment is conceptualized as a 10×10 grid, representing different types of terrain. Each cell in the grid is assigned a specific soil type, with varying levels of optimality for navigation.

**Reward and Penalty Mechanism:** The model incorporates a system of rewards and penalties, tailored to the characteristics of each soil type. Soils closer to the optimal Bedrock type incur lower penalties or higher rewards, guiding the robot towards more favorable paths.

The following is a list of the different soil types and their associated penalties:

- **'Bedrock_0'**: 0
- **'Bedrock_1'**: 0
- **'Bedrock_2'**: 0
- **'Coarse_Sand_0'**: 1
- **'ES-3_0'**: 1
- **'ES-2_0'**: 1
- **'Bedrock_ES-2_0'**: 1
- **'Rocks_on_ES-3_2'**: 1
- **'ES-1_1'**: 1
- **'Rocks_on_ES-3_1'**: 2
- **'ES-3_1'**: 2
- **'Coarse_Sand_1'**: 2
- **'ES-3_2'**: 3
- **'Coarse_Sand_2'**: 3
- **'ES-1_0'**: 3
- **'Rocks_on_ES-3_0'**: 4
- **'ES-2_2'**: 4
- **'ES-2_1'**: 4
- **'ES-1_2'**: 5
- **'Bedrock_ES-2_1'**: 5
- **'Bedrock_ES-2_2'**: 5

This penalty was selected according to the final dataset, which ranked every type of soil. When the robot explores its environment using Q-learning, it learns to associate actions with states and their respective Q-values. These Q-values represent the expected cumulative rewards the robot can achieve by taking a specific action from a particular state. The penalties come into play by influencing these Q-values.

*2) Hyperparameter Selection and Rationale:* The efficacy of Q-learning is heavily influenced by the choice of hyperparameters. The following parameters were carefully selected:

- **Learning Rate (0.5):** This parameter controls the rate at which new information updates the Q-values. A moderate learning rate was chosen to strike a balance between learning from new experiences and retaining prior knowledge.
- **Discount Factor (0.9):** A higher discount factor was used, emphasizing the importance of future rewards. This encourages the algorithm to plan for long-term gains, a critical aspect in complex navigation tasks.
- **Epsilon-Greedy Strategy:** Starting with an epsilon value of 1.0, which gradually decays to 0.01, allows the model to initially explore the environment extensively and then gradually shift its focus to exploiting known paths.

*3) Detailed Methodology and Evaluation:* **Training Process:** The training involved running multiple episodes, each consisting of the robot navigating from a starting point to a predetermined goal. The Q-values were updated at each step based on the observed rewards or penalties, using the Q-learning update rule.

**Model Evaluation:** The evaluation of the model was multifaceted. The model's progress was evaluated by tracking the total rewards and steps in each episode, with increasing rewards and fewer steps indicating improved efficiency. The success rate, measured by how often the robot reached its goal, directly assessed the model's performance. Visualizing the robot's navigation paths in various episodes provided insights into its learned strategies and their evolution. Additionally, a heatmap was generated post-training, showing the frequency of visits to grid cells and revealing the robot's terrain preferences, helping to identify frequently traversed and less-traveled paths.

## V. Results and Discussion

### A. Terrain classifier

The K-Nearest Neighbors (KNN) algorithm was selected as the primary model for terrain classification in this project,

based on its compatibility with the high-dimensional nature of the sensor data from the ANYmal robot. The model demonstrated an impressive accuracy of approximately 88.84%, which was a significant improvement over other tested algorithms like Decision Trees and Random Forests. This high level of accuracy was achieved through the careful selection of the 'k' value, which was finalized at k=5. This choice struck a balance between sensitivity to noise and generalization capabilities, making the KNN algorithm adept at handling the dataset multidimensional characteristics.

The KNN model's success was primarily due to its ability to make decisions based on the local neighborhood of data points, an approach well-suited for the terrain classification task. The algorithm's adaptability to non-parametric data and its computational efficiency, crucial for real-time processing requirements, being suitable for the project.

The potential of further applications of the dataset is wide. The plan is to utilize the dataset to create different clusters of the soils to detect more specific types of terrains. For example, the category of 'bedrock' could be subdivided into three groups: optimal, medium, and suboptimal bedrock, with corresponding scores of 50, 45, and 40 points, respectively.

This refined classification system would allow for a more deep understanding of terrain types, enabling the ANYmal robot to make more informed decisions about where to step. Implementing reinforcement learning in the future could further enhance the robot's ability to choose the most optimal trajectory on a given path, taking into account factors like energy conservation, time efficiency, and resource management.

By advancing the model to recognize these subtle differences in terrain types, the robot could avoid less favorable surfaces, extending its operational life, ensuring safety, and improving overall efficiency.

### *B. Q-learning algorithm for path optimization*

*1) Visualization of the Terrain and Path:* The Q-learning model's performance was assessed through various visualizations representing the terrain, the robot's path, and its interaction with the environment.

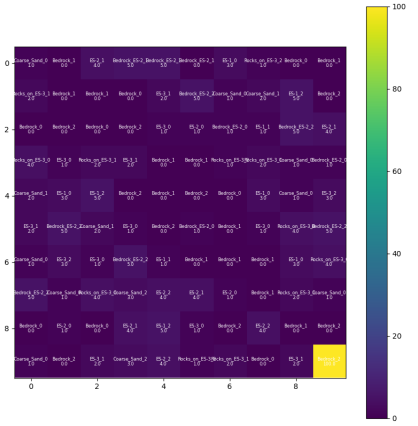
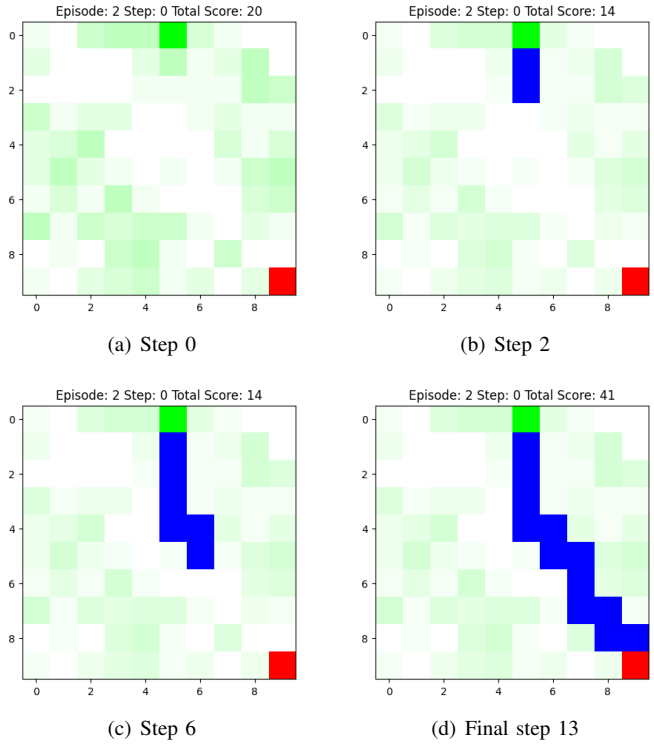


Fig. 1. Grid world with terrain types and penalties

**Figure 1** shows the grid world used in the simulations, color-coded to illustrate the types of terrain and their associated penalties. The 'Bedrock' types, with zero penalty, are the most desirable terrains, while higher penalties are ascribed to terrains like 'Coarse Sand' and 'ES-1', which present greater navigational challenges.




(a) Step 0 (b) Step 2

(c) Step 6 (d) Final step 13

Fig. 2. Robot's Q-learning path from start to goal

**Figure 2** illustrates different instances of the robot's navigational path within the environment, with the start and goal positions distinctly marked. The path, shown in blue, represents the robot's decision-making process under the Q-learning policy. The progression from green (start) to red (goal) denotes the robot's journey, with the color intensity in the background representing the degree of penalty or reward encountered. The start score is 20, during the path the robot encounters penalization and lose points, until it reaches the end to gain points.

*2) Analysis of Rewards and Steps per Episode:* Two key metrics were tracked during the training episodes: the total rewards per episode and the number of steps per episode.

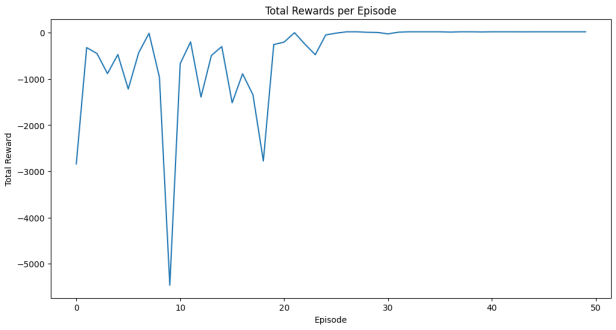



Fig. 3. Reward trend showing robot's learning and optimization.

**Figure 3** displays the trend in total rewards per episode. An initial period of fluctuation is observed, reflecting the exploration phase where the robot encounters various penalties. As the episodes progress, a steady increase in total reward is noticeable, indicating the robot's learning and adaptation to the environment, seeking paths that maximize rewards. It can be seen that, from the episode 23, the robot achieve its maximum optimal path.

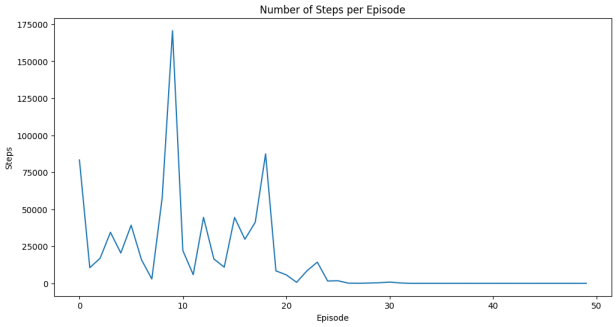



Fig. 4. Decreasing steps reflect robot's improved efficiency.

**Figure 4** shows the number of steps the robot takes in each episode. A significant reduction in steps over time suggests that the robot is learning to reach the goal more efficiently, minimizing unnecessary movements and optimizing its path based on the learned Q-values.

Initially, the robot's movements were exploratory, often leading to paths with higher penalties. This behavior is typical of reinforcement learning models in the early stages of training, characterized by a trial-and-error approach. Over time, the robot's behavior transitioned from exploration to exploitation, as reflected in the increasing trend of total rewards and the decreasing trend of steps taken per episode.

The model showed a capacity for adaptation, where it gradually learned from the environment and its previous experiences.

The model's effectiveness was not only theoretical but also practical. The implementation of the Q-learning algorithm facilitated a improvement in the robot's navigational decisions. The visualization of the robot's path demonstrated a clear progression from random movements to a strategy aimed at reaching the goal efficiently.

The evaluation of the model's performance was multifaceted, inclusing both quantitative metrics, such as rewards and steps, and qualitative analyses, such as path visualization. The results revealed a successful learning curve, where the robot exhibited a marked improvement in identifying and following optimal paths as training progressed.

### *C. Challenges*

The initial goal to implement the project within ROS using a 4-legged robot like the Mini Pupper for physical terrain navigation was not possible due to technical and operational constraints. Instead, the project focused towards utilizing the ANYmal dataset, which brought about a significant shift in objectives. The focus transitioned to a less complex environments to enhance virtual adaptability and terrain recognition.

The project's initial goals had to be adapted due to practical constraints, leading to a focus on a simpler simulated environments rather than 3D terrain navigation in real tine. This shift presented challenges in aligning the Q-learning approach with the new objectives.

A significant challenge in the non-supervised learning segment was the creation of a new dataset using clustering despite already having a predefined dataset. The rationale for this additional step was to introduce terrain variability, which is crucial for the robot's ability to recognize and navigate through a diverse range of terrains. Clustering added a layer of complexity to the project, as it necessitated the fine-tuning of the number of clusters and the interpretation of cluster centroids in relation to the robot's sensor readings.

One of the main challenges was ensuring the Q-learning model's convergence to an optimal policy. Given the complexity of the terrain grid and the variability of soil types, the model had to navigate a balance between exploring new paths and exploiting known, optimal routes. Fine-tuning the model's hyperparameters, such as the learning rate, discount factor, and epsilon values for the epsilon-greedy strategy, was crucial to achieve effective learning without falling into suboptimal policies.

Designing an appropriate reward function to accurately reflect the cost or benefit of traversing different terrains was a significant challenge. The reward function had guide the robot toward the most favorable terrains, which required a good understanding of each soil type's impact on the robot's performance. This part must be improved, due to the hardware limit, more complex rewards assignment were impossible and it was not studied in depth. Also, the real-time classifier to study the terrain in real time was not implemented.

### *D. Further steps*

The next phase will focus on implementing the supervised learning algorithm in real time. The key objective is to enable the robot to navigate physical terrains with the same proficiency it has demonstrated in the virtual domain. To achieve this, the trained model will be integrated into the robot's control system, where it can process sensor data on the fly and make immediate decisions based on the learned behaviors.

Further research will also explore the integration of the clustering approach with the supervised learning model to enhance the robot's ability to distinguish between subtle variations in soil types. This could involve a feedback loop where the robot refines its terrain classification model based on real-world experiences, potentially using a semi-supervised learning approach that combines labeled and unlabeled data.

## VI. Conclusions

The project has successfully united three distinct areas of machine learning to address the challenge of robotic navigation across varying terrains. Through unsupervised learning techniques, it was achieved a breakdown of complex soil categories into more manageable subclasses. This provided a detailed landscape for the robot, enhancing its navigation by effectively 'feeling' the ground.

In supervised learning, the robot was trained to recognize these terrains using the K-Nearest Neighbors (KNN) algorithm. This required translating the terrain labels into a numerical format, equipping the robot with the 'vision' to distinguish between different ground types accurately.

Reinforcement learning brought the 'thought process' to the robot's movement, enabling it to determine the most efficient paths by learning from its actions. The robot's ability to make increasingly better choices over time was a testament to the efficacy of this approach.

The evaluation of the robot's performance revealed clear progress. It was observed that the robot became more adept at choosing its path, evidenced by a decrease in unnecessary steps and an increase in successful navigation. The visual representation of its path and the statistical data on rewards and steps concludes that the robot was indeed learning and improving its ability to move trough various terrains.

Looking forward, the project sets the stage for real-world applications. While the current achievements are within a simulated environment, the next steps involve translating these methods to physical robots navigating actual terrains. This transition presents its own set of challenges, yet the foundations laid by this project provide a robust base for tackling these future challenges. The goal is to create autonomous robots capable of intelligent navigation in any environment, potentially transforming tasks in hazardous or inaccessible locations for humans.

## References

[1] J. M. Idelsohn, "A learning system for terrain recognition," Pattern Recognition, vol. 2, no. 4, pp. 293-301, 1970. doi: 10.1016/0031-3203(70)90019-1.

[2] P. Kozlowski and K. Walas, "Deep neural networks for terrain recognition task," in 2018 Baltic URSI Symposium (URSI), Poznan, Poland, 2018, pp. 283-286. doi: 10.23919/URSI.2018.8406736.

[3] B. Priyaranjan, "Development of quadruped walking robots: A review," Department of Mechanical Engineering, National Institute of Technology, Arunachal Pradesh 791112, India, 2021, pp. 1-8, doi: 10.1016/j.asej.2020.11.005.

[4] "ANYmal research". ANYmal Research. [Online]. Available: https://www.anymal-research.org/

[5] M. Hutter et al., "ANYmal - a highly mobile and dynamic quadrupedal robot," 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Korea (South), 2016, pp. 38-44, doi: 10.1109/IROS.2016.7758092.

[6] N. Singh. "Understanding Q-Learning: A Powerful Reinforcement Learning Technique". Medium. [Online]. Available: https://medium.com/@navneetsingh95791/understanding-q-learning-a-powerful-reinforcement-learning-technique-29a3da36f611

[7] H. Kolvenbach, C. Bartschi, L. Wellhausen, R. Grandia, and M. Hutter, "Haptic Inspection of Planetary Soils With Legged Robots," in IEEE Robotics and Automation Letters, vol. 4, no. 2, April 2019, doi: 10.1109/LRA.2019.2896732.

[8] A. R. Kapil, "Decision Tree Algorithm in Machine Learning: Advantages, Disadvantages, and Limitations," Analytixlabs, October 1, 2022. [Online]. Available: https://www.analytixlabs.co.in/blog/decision-tree-algorithm/

[9] Genesis, "Pros and Cons of K-Nearest Neighbors," FromTheGenesis, September 25, 2018. [Online]. Available: https://www.fromthegenesis.com/pros-and-cons-of-k-nearest-neighbors/

[10] A. Fleiss, "What are the advantages and disadvantages of random forest?," Rebellion Research, Feb. 19, 2023. https://www.rebellionresearch.com/what-are-the-advantages-and-disadvantages-of-random-forest

[11] A. A. Tokuç, "k-Nearest Neighbors and High Dimensional Data," Baeldung, November 9, 2022. [Online]. Available: https://www.baeldung.com/cs/k-nearest-neighbors

[12] "Deep Insights Into K-nearest Neighbors," Towards AI, July 25, 2020. [Online]. Available: https://towardsai.net/p/machine-learning/deep-insights-into-k-nearest-neighbors

[13] S. Uddin, I. Haque, H. Lu, et al., "Comparative performance analysis of K-nearest neighbour (KNN) algorithm and its different variants for disease prediction," Scientific Reports, vol. 12, p. 6256, 2022. doi: 10.1038/s41598-022-10358-x.

## Appendix A
## Repository containing the 3 models

GitHub repository available at https://github.com/Cokitoquintal/qlearn_terrain_path_robot[1].

[1] https://github.com/$Cokitoquintal/qlearn/terrain/path/robot$