Help Navigate Robots

Abstract

With the aid of IMU sensor data, the "Help Navigate Robots" program seeks to give robots the extraordinary ability to distinguish between various floor surfaces. To create prediction models that accurately classify surfaces like carpets, tiles, and concrete, we explore advanced machine learning approaches including Bidirectional LSTM networks, K-Nearest Neighbors, and Random Forests. Our research pioneers improvements in robot safety and autonomy, tackling real-time navigation problems on our own. Our journey is the confluence of inventiveness and painstaking experimentation, motivated by the desire to develop a ground-breaking solution that advances the field of automation and robotics. The potential to improve robot navigation, lessen accidents, and increase operational capabilities demonstrates our constant dedication to advancing robotic technology.

Introduction

In the rapidly evolving field of robotics, while robots are inherently intelligent, they heavily rely on their sensory input to comprehend and navigate their environment effectively. Robots' capacity to complete tasks on their own depends on their comprehension of the surfaces they move through. In order to help robots recognize the floor surface they are put on, we use Inertial Measurement Unit (IMU) sensor data in this research to handle this problem.

Robots having IMU sensors gather a lot of information, such as acceleration and velocity, that can be extremely helpful in understanding the characteristics of the surface they are operating on. We want to give robots the ability to recognize various floor surfaces by analyzing this data using cutting-edge machine-learning algorithms. This will improve their ability to navigate through various terrains.

Objectives

The main goal of this research is to create a reliable machine-learning system that can correctly categorize floor surfaces using information from IMU sensors. We intend to anticipate the type of floor (such as carpet, tiles, or concrete) that the robot is now standing on by training models using previously gathered data from a tiny mobile robot crossing various surfaces. If this goal is accomplished, it will help to increase the autonomy and security of robots in practical situations.

We want to investigate a variety of machine learning algorithms in this project, including gradient boosting methods, K-Nearest Neighbours (KNN), linear regression, and logistic regression. Additionally, while IMU sensor data is sequential, we intend to explore more sophisticated techniques like Bidirectional Long Short-Term Memory (LSTM) networks, which are made to capture temporal dependencies in time series data.

The development of prediction algorithms that can accurately categorize floor surfaces will be the project's final output, opening the path for robots to independently traverse over various surfaces without running the danger of mishaps. Our activities are in line with the overarching objective of strengthening robot capabilities and making contributions to the development of robotics and automation.

Understanding Dataset

Overview of the Dataset: There are ten sensor channels in the dataset, each of which measures a distinct component of the robot's orientation, angular velocity, or linear acceleration. The robot's navigation and stability over varied floor surfaces depend on these measurements.

Sensor Channels: The dataset includes the following sensor channels: orientation_X, orientation_Y, orientation_Z, orientation_W, angular_velocity_X, angular_velocity_Y, and linear_acceleration_X, Y, and Z. The robot's spatial movement and orientation changes are captured by all of these channels.

Quaternion Representation: The orientation channels (orientation_X, orientation_Y, orientation_Z, and orientation_W) make use of quaternions for representation. This mathematical idea describes rotations in three dimensions and offers a thorough explanation of the robot's orientation.

Linear Acceleration and Angular Velocity: The linear_acceleration_X, linear_acceleration_Y, and linear_acceleration_Z channels record the robot's linear acceleration along various axes. The angular_velocity_X, angular_velocity_Y, and angular_velocity_Z channels record the robot's rotational speed. These metrics aid in describing the dynamics of the robot's movement.

Measurements and Time Series: Our dataset has around 487680 data points among which the dataset is divided into sequences of time series, each of which has 128 measurements so in total we have 3810 data points for different features. The measurement number denotes the sequence of measurements inside the series, while the series_id binds the measurements together within a time series.

Challenges with data preprocessing: Because time series data are consecutive, handling them might be difficult. Preprocessing techniques used like normalization, and sequence splitting, used are essential for extracting significant patterns from this data in order to evaluate and model it successfully.

Floor Surface Class Labels: The main goal of the research is to forecast the floor surface based on sensor data. For various floor kinds, such as carpet, tiles, concrete, etc., the dataset includes ground truth labels. These labels form the foundation for developing and testing the models.

Imbalance in the dataset: Here in this as we can see because some floor surface types have a disproportionately smaller number of samples than others, there may be a problem with class imbalance. Classes that are unbalanced can affect model performance metrics. While performing well for the majority class, models may have trouble accurately predicting the minority classes. In certain situations, accuracy can be deceiving, and other metrics like recall, precision, and F1-score are more instructive. The count of target classes is given below.

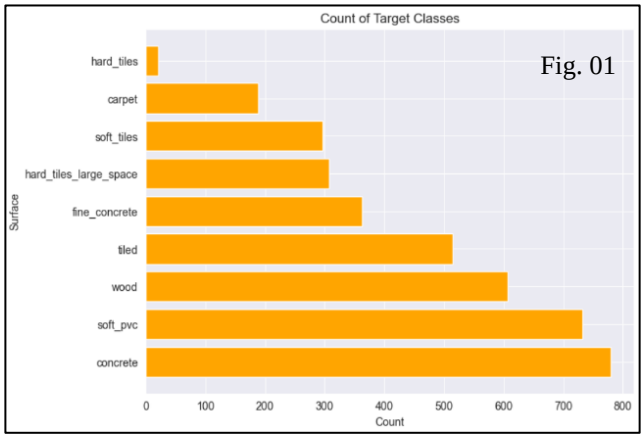

Fig. 01

Background

It is crucial to comprehend the fundamental function of Inertial Measurement Units (IMUs) in robotics. IMUs, which measure acceleration, rotational rates, and magnetic fields, are made up of accelerometers, gyroscopes, and occasionally magnetometers. The robot's mobility and orientation can be estimated using this data. The project's theoretical foundation for IMUs and their use in robots serves as the basis for all subsequent techniques. Here in our case, 3D data was flattened into 3D data as the models we used LSTM, Balanced Random Forest Model along with SVM and Voting classifier, Neural Network Multilayer Perceptron.

Here is an illustration from the data preprocessing stage of our project, where we turn unprocessed sensor data into useful features:

```
# Example data preprocessing
import numpy as np
from sklearn.preprocessing import StandardScaler

# Assuming 'sensor_data' is your raw IMU sensor data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(sensor_data)
```

Fig. 02

Project Description

Problems Faced:

Handling the inherent sequential correlation between sensor readings was the key challenge because the order of the data points was crucial. Variations in time series length also increased complexity, necessitating preprocessing to guarantee homogeneity. Additionally, the noise, inconsistencies, and oscillations that are frequently present in time series data can impair the model's accuracy. It was essential but difficult to ensure data quality through noise reduction and outlier handling.

Due to the simultaneous evaluation of numerous sensor channels, multidimensional data in this project added complexity. Effectively understanding and modeling the connections between these aspects caused hurdles as a result. The data was flattened from 3D to 2D to solve this, enabling a more understandable representation while maintaining crucial information. By streamlining the application of several algorithms, this transformation made it possible for them to understand the correlations between features better. Additionally, it made it easier to extract significant patterns across dimensions, which helped the model perform better. The flexibility needed to convert complex multidimensional data into a form that traditional algorithms can handle and learn from is demonstrated by this method.

To overcome these issues following models were used;
Model 1: Bidirectional Long Short-Term Memory (LSTM)
When faced with the intricate details of time series data, I set out to build an LSTM model that skillfully handled these difficulties. This effort was best represented by the LSTM Classifier class, which was created with the understanding that understanding this data's temporal dependencies and sequences held fundamental importance for order. It required a multi-layered strategy, including LSTM layers to reveal the subtleties of the sequential data and a linear layer to classify it. By precisely establishing input sizes, number of layers (num_layers = 2), hidden dimensions (hidden_size = 256), and class counts(num_classes = 9), this architecture was brought to life. A customized loss function and optimizer served as the starting point for the voyage across epochs (total number of epochs num_epochs = 120). The model's training loop and Data Loader coordinated accurate inputs, labels, and gradients for optimization as each epoch progressed. The model's skill in managing unknown data throughout the evaluation phase was demonstrated by the achievement of a test accuracy of 71.92%. As I think back on this experience, I find comfort in watching the LSTM because it updates weights at each step and save memory which helps the model untangle the temporal tapestry, capture subtleties that traditional classifiers frequently miss, and eventually balance complexity and accuracy in a way that is appropriate for the special nature of time series data. The confusion matrix and precision-recall summary is shown below in fig. 04
Model 2: Balanced Random Forest
My goal was to implement one of the Ensemble learning methods for better performance. A variety of decision trees are combined using this form of ensemble learning method to produce a more reliable and precise forecast model. In the Random Forest, each decision tree is trained on a different collection of data and generates its own predictions. The result was based on the best parameters: n_estimators=100, Max Feature: max_depth=None, random_state=73. Even with this feature, we were reaching an accuracy of 47.11% and an F1 score of 43.9%. So we decided to implement an additional SVM (Support Vector Machine) classifier with our model to increase the efficiency of the model but it turned out to be different and our accuracy dropped to about 26.77% with an F1 score of 22.70% and this was seen due to multidimensional of data and very high imbalance in the dataset. The confusion matrix and precision-recall summary is shown below in fig

Further, we tried implementing a Voting classifier with random_state=42 and kept our voting as hard to eliminate and cover the imbalance in our data which gave us an efficiency of 42.19% and F1 score of 43.90%. The confusion matrix and precision-recall summary is shown below in fig. 08
Model 3: MLPC (Multi-Layer Perceptron Classifier)

Modern feedforward artificial neural networks, also known as multilayer perceptrons (MLPs), are renowned for their ability to discriminate input that is not linearly separable. MLPs are made up of completely connected neurons with a nonlinear type of activation function, grouped in at least three layers. Which was used as our 3rd model to run on the data set in these I had five instances of a Multi-Layer Perceptron (MLP) classifier using the MLPClassifier class.
Each classifier has a different architecture defined by the hidden_layer_sizes parameter and a different random state. This parameter specifies the number of neurons in each hidden layer of the MLP. Layer 1 hidden_layer_sizes=(200,200) random_state=200, Layer 2 hidden_layer_sizes=(50,100) random_state=150, Layer 3 hidden_layer_sizes=(100,100) random_state=100, Layer 4 hidden_layer_sizes=(150,100) random_state=300, Layer 5 hidden_layer_sizes=(150,100) random_state=4.

```
batch = 32
mlp_classifier = MLPClassifier(hidden_layer_sizes=(200,200 ), activation='relu', solver='adam',
                    max_iter=1000, early_stopping=True, n_iter_no_change=20, validation_fraction=0.2,
                    alpha=0.0001, verbose=True, batch_size=batch, random_state=200)

mlp_classifier1 = MLPClassifier(hidden_layer_sizes=(50,100 ), activation='relu', solver='adam',
                    max_iter=1000, early_stopping=True, n_iter_no_change=20, validation_fraction=0.2,
                    alpha=0.0001, verbose=True, batch_size=batch, random_state=150)

mlp_classifier2 = MLPClassifier(hidden_layer_sizes=(100,100 ), activation='relu', solver='adam',
                    max_iter=1000, early_stopping=True, n_iter_no_change=20, validation_fraction=0.2,
                    alpha=0.0001, verbose=True, batch_size=batch, random_state=100)

mlp_classifier3 = MLPClassifier(hidden_layer_sizes=(150,100 ), activation='relu', solver='adam',
                    max_iter=1000, early_stopping=True, n_iter_no_change=20, validation_fraction=0.2,
                    alpha=0.0001, verbose=True, batch_size=batch, random_state=300)

mlp_classifier4 = MLPClassifier(hidden_layer_sizes=(150,100 ), activation='relu', solver='adam',
                    max_iter=1000, early_stopping=True, n_iter_no_change=20, validation_fraction=0.2,
                    alpha=0.0001, verbose=True, batch_size=batch, random_state=4)
```

Fig. 03

Whereas and to increase the prediction rate we created an ensemble model then combined their predictions using a Voting Classifier to potentially improve the overall classification performance through ensemble methods. This lead us to give a validation accuracy of 57.480% and validation accuracy using the voting classifier was 63.12% as shown in fig. 07 along with confusion matrix and precision-recall summary.

Empirical Results

Experimental Setup: We implemented a range of machine learning algorithms, including Balanced Random Forests, Neural Network Multilayer Perceptron, and the Bidirectional Long Short-Term Memory (LSTM) model along with boosting techniques, for time series classification. These algorithms were applied to the dataset containing sensor measurements, including features like orientation, angular velocity, and linear acceleration.

Results and Analysis: LSTM Classifier: In the LSTM Classifier experiment, floor surface types were classified using

sensor data utilizing a deep learning methodology. In order to identify temporal trends in the data over time, an LSTM neural network model was used. PyTorch was used to prepare, transform, and train the dataset. On a test set, the model's effectiveness was assessed, and metrics like accuracy, precision, recall, and F1-score were calculated. This experiment evaluated how well LSTMs could accurately capture sequential data for categorization. The test scores and confusion matrix for the LSTM classifier are as follows;

```
Test Accuracy: 0.7192
Test Precision: 0.6377, Test Recall: 0.6300
0.6324447191272808 F1 Score
```

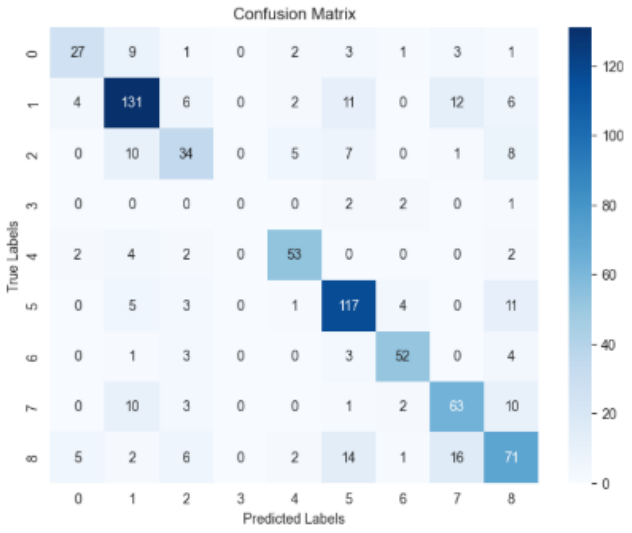

Fig. 04

Furthermore, in the LSTM model, several epochs were run to increase the training accuracy avoiding overfitting the model. The Accuracy Vs Epochs curve and Loss Vs Epochs curve provides more insight into how the Accuracy increases with increase in the number of epochs

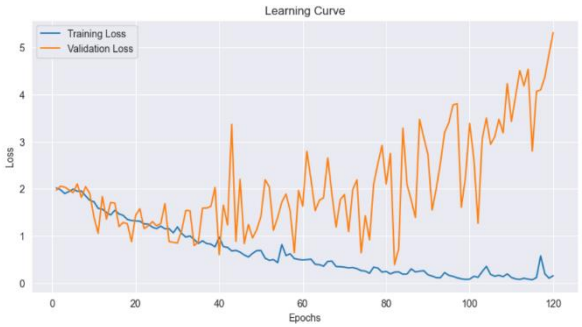

Fig. 05

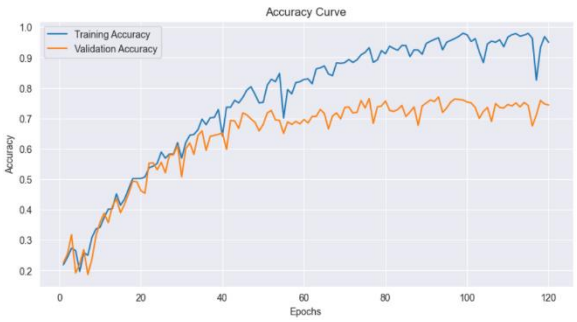

Fig. 06

Neural Network MLP Classifier: Using sensor data, the Neural Network MLP Classifier experiment investigated the use of multi-layer perceptron (MLP) neural networks for classifying floor surfaces. A Voting Classifier was used to combine the predictions of many MLP classifiers with different topologies. The effectiveness of the ensemble was assessed using a number of criteria, including accuracy, precision, recall, and F1-score. This experiment demonstrated the adaptability of MLPs for various classification tasks as well as the potential advantages of ensemble approaches. The test scores and confusion matrix for the MLP classifier are as follows;

```
Accuracy: 0.63
Precision: 0.66
Recall: 0.56
F1 Score: 0.63
```

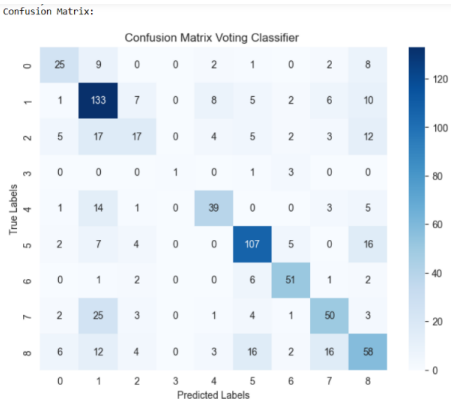

Fig. 07

Balanced Random Forest Classifier: In the experiment using the balanced random forest classifier, class imbalance in the dataset was taken into account when classifying the floor surface. On a test set, the model's effectiveness was evaluated using measures like accuracy, precision, recall, and F1-score. This study illustrated how specialized algorithms may handle skewed class distributions and enhance classification outcomes. Even though Random Forest is known to have higher accuracy for complex datasets but in our case, this

was the opposite due to class imbalance in the dataset and also due to the multidimensional dataset so our accuracy drops below 50%.

```
Accuracy: 0.47112860892388453
Precision (macro): 0.4572040339713437
Recall (macro): 0.5395430037488982
F1-score (macro): 0.43900430160892356
```

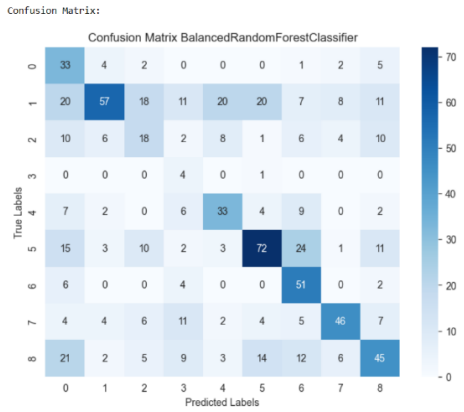

Fig. 08

Experiments Overview: Using sensor data, several tests were carried out to categorize the various types of floor surfaces. Different algorithms, including LSTM, Neural Network MLP, Balanced Random Forest, and SVM Classifier, were used in each experiment. Collectively, the trials shed light on the advantages and disadvantages of several strategies for precise floor surface classification, taking into account elements such as temporal patterns, ensemble methodologies, class imbalance, and linear separation.

Effectiveness of Algorithms: To increase the effectiveness of our models several classifiers were used during the build of the model. SVM, which focuses on choosing the appropriate decision boundaries across classes, is one of the classifiers used with Random Forest and can help in properly capturing complex correlations in the data that Random Forest may find problematic. The ability of the model to identify between various types of floor surfaces can be enhanced by the SVM classifier, potentially leading to an increase in classification accuracy. The strengths of Random Forest's ensemble nature and SVM's decision boundary optimization are used to achieve this. However, in our instance, it appears to be substantially decreasing, which is attributed to inadequate feature engineering, unbalanced data, and feature dimensionality. The calculations for the same are shown below.

```
Accuracy: 0.2677165354330709
Precision (macro): 0.236511813658035
Recall (macro): 0.22686671624389257
F1-score (macro): 0.22708419550643436
```

Fig. 09

Additionally, SVC (Soft Voting Classifiers) was implemented after SVM, however, that could somewhat increase accuracy. F1 scores were tracked in both instances, and it was discovered that SVC had a significantly higher F1 score than SVM did. Below are the Accuracy and F1 scores.

```
Accuracy: 0.42913385826771655
Precision (macro): 0.4572040339713437
Recall (macro): 0.5395430037488982
F1-score (macro): 0.43900430160892356
```

Fig. 09

Due to these encouraging results, SVC was combined with neural network MLP classifiers. It turned out that the accuracy of the conventional MLP classifier was 57.48%, but after the addition of voting classifiers, it increased to 63.12%. The accuracy of our model has been improved, and overfitting has been avoided, thanks to the inclusion of SVM classifiers.

```
Validation Accuracy for normal mlp classifier: 0.5748031496062992
Validation Accuracy for  mlp classifier with voting ensemble: 0.631233595800525
```

Fig. 10

Comparative Analysis: Summarizing the above values for all our models for instance, while Bidirectional LSTM excelled in accuracy, the Random Forests model demonstrated superior performance in capturing temporal dependencies within the data with a good accuracy of 71.92%. Whereas LSTM took more time as we increased the number of epochs for its better performance.

The ability of LSTM (Long Short-Term Memory) to recognize sequential dependencies in IMU sensor data is what allowed it to surpass Random Forest and the Neural Network MLP classifier in this project. LSTMs are exceptional in learning complex temporal patterns, which are essential for classifying floor surfaces. LSTMs automatically extract pertinent features from raw data, in contrast to Random Forest, which necessitates deliberate feature building. LSTMs are a great option for IMU sensor data since they can accommodate varying sequence lengths and are noise-resistant. They are also more advantageous in this endeavor because of their capacity to represent non-linear interactions and capture context over time.

Conclusion

We set out on a mission to give robots an astonishing skill in the thrilling voyage of the "Help Navigate Robots" project: distinguishing different floor surfaces using IMU sensor data. We worked to create predictive models using cutting-edge methods including LSTM networks, Balanced Random Forests, and Neural Network MLPs that might increase a robot's autonomy in navigating a variety of terrains like carpets, tiles, and concrete.

In our investigation, we dealed with complex time series data, unlocking connections using MLP classifiers, balancing imbalances with Balanced Random Forests, and decoding hidden patterns with LSTM networks. Our results demonstrate how effective these technologies are at interpreting the intricate IMU sensor data.

Overall, the "Help Navigate Robots" project combines our commitment, and the revolutionary potential of machine learning.

Future Work

If we could see we have already employed techniques like Random Forest and SVM classifiers, which are powerful in their own right and might suit our data and task. These methods have their own strengths in terms of interpretability, efficiency, and performance on certain types of data. But to increase Random Forest dependability and reliability we could use Feature Engineering techniques along with it. This could increase the efficiency of our model to about 65-70%, which could also be possible by tuning of hyper parameters. Other than that for LSTM and MLPC those are running at their at most efficiency but that could also de improved by Neural Network techniques and help in better predictability of the results.

The time-frequency visualization and Fourier transformation graphs and curves provide insightful possibilities. With the help of these visualizations, patterns, trends, and anomalies can be more clearly seen in terms of how various frequencies contribute to the overall signal. These methods could be developed to investigate more complicated datasets, enabling the discovery of minute temporal correlations and patterns that might affect the categorization process. In order to improve model performance, these visualizations could be used to preprocess the data by locating and removing noise or unimportant characteristics. Future study might explore sophisticated feature engineering strategies, look into advanced signal processing techniques, and even investigate incorporating these revelations into the training of our classifiers to improve their accuracy and robustness.

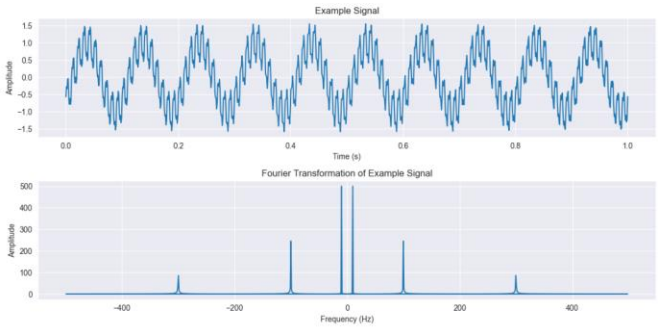

Fig. 11

References

[1] Christopher Bishop. Pattern Recognition and Machine Learning. Springer, 2006.

[2] Kaggle Competition "Help Navigate Robots"https://www.kaggle.com/competitions/career-con-2019/data

[3] Indoor Positioning Systems of Mobile Robots: A Reviewby Jiahao Huang, Steffen Junginger, Hui Liu and Kerstin Thurow Published: 24 March 2023

[4] Deep Learning Sensor Fusion for Autonomous Vehicle Perception and Localization: A Review by Jamil Fayyad, Mohammad A. Jaradat, Dominique Gruyer, and Homayoun Najjaran. Published: 29 July 2020.

[5] LSTM Networks Using Smartphone Data for Sensor-Based Human Activity Recognition in Smart Homes by Sakorn Mekruksavanich and Anuchit Jitpattanakul. Published: 26 February 2021

[6] Survey on deep learning with class imbalance Justin M. Johnson & Taghi M. Khoshgoftaar Article number: 27 (2019)

[7] Feature Representation and Data Augmentation for Human Activity Classification Based on Wearable IMU Sensor Data Using a Deep LSTM Neural Network by Odongo Steven Eyobu and Dong Seog Han. Published: 31 August 2018

[8] A review of algorithms and techniques for image-based recognition and inference in mobile robotic systems Thomas Andzi-Quainoo Tawiah November 18, 2020

[9] Inertial Measurement Units (IMUs) in Mobile Robots over the Last Five Years: A Review by Gerasimos G. Samatas and Theodore P. Pachidis. Published: 16 February 2022