UiO : Department of Informatics
University of Oslo

Terrain classification using 3D optical tactile sensor

A machine learning approach

Jiader Chou

Master's Thesis Spring 2017

Terrain classification using 3D optical tactile sensor

Jiader Chou

Abstract

Identifying different types of terrains is an important ability for every legged robot to achieve a stable locomotion. A variety of sensors has been applied to different robots in order to discriminate between terrains accurately. The tactile sensor has the benefits of measuring properties from terrain by physical contact between the sensor and the surface. However, the tactile sensor has rarely been utilized on quadruped robots in previous studies, and little attention has been paid to the type of sensor. There is a variety of types of tactile sensors, each with their benefits and drawbacks. The optical tactile sensor has high sensitivity, small size, light weight and low detection time, which are important properties to distinguish between different surfaces.

This thesis investigates the possibility of identifying different terrains using 3D optical tactile sensors and machine learning. The measurements were retrieved from a quadruped robot developed at the University of Oslo on four different terrains. The proposed approach has the ability to classify terrains in real-time on the physical robot, and a custom segmentation method was presented for extracting desired sensor data. The segmented sensor data was the basis for creating five different feature sets and tested on five different classifiers: support vector machine, artificial neural network, naive Bayes, k-nearest neighbors, and decision tree. The experimental results demonstrated to be among the top performing approaches compared to earlier work with an accuracy of 94.8% with the support vector machine.

Contents

List of Figures

List of Tables

Preface

I want to thank my supervisor, Kyrre Glette for the guidance and support throughout the thesis work. I would also thank my second supervisor, PhD candidate Tønnes Frostad Nygaard, for his advisement and technical discussion.

Last, but not least, I would like to thank my fellow students, friends, and family for their support.

Chapter 1

Introduction

Humans have the ability to adapt their walking styles on different terrains to achieve stable locomotion. This ability to adapt locomotion is based on previous experience. For instance, one would not run on an icy sidewalk in order to prevent slipping or falling. In contrast, on more rough surfaces, one may freely decide the speed to move without worrying about losing grip or balance. To attain this ability to adapt while moving, robots must first be able to detect and distinguish among different terrains.

Terrain classification is the process of identifying different types of terrain by measuring features such as texture, slope, roughness, hardness, and friction. It is a popular research field where countless studies can be found in the literature [1, 2, 3, 4, 5, 6]. Some of the importance of terrain classification is shown in [7], where different controllers were suited for different terrains by letting a quadruped robot hop on a soft and hard terrain. Another study investigated the effect of performance with different gait parameters on different terrains [4]. The results indicated there is a trade-off between the energy consumption and physical speed of the robot by controlling the velocity of the leg motors on different types of terrains.

Robot's perception of different surfaces plays an important rule for successful terrain classification. Researchers often obtain features from terrain from a distance using sensors such as cameras [6] or laser scanners [8]. Other studies measure properties through robot's interaction on different surfaces such as leg joints [9], and accelerometers [10], or by physical contact between the sensor and a surface such as the tactile sensor [11, 12, 13]. Degrave et al. [11] investigated different types and combinations of sensors for a quadruped robot to identify which were suitable and provided most information on the terrain. The result indicated that the combination of tactile sensor and proprioceptive joint angle were the most informative of all the sensors.

In most of the studies, the tactile sensor is often fused with other sensors due to terrain classification [14, 15, 16, 17, 13]. Exclusively using a tac-

tile sensor is not as common [11, 16], nor does the researchers report the type of the tactile sensor employed in experiments. It exists a variety of tactile approaches that are based on different technologies such as resistive [18, 19, 20, 21], piezoelectric [22, 23], capacitive [24, 25], magnetic [26], and optical [27, 28, 29, 30, 31, 32, 33], where every tactile type has its benefits and drawbacks. Thus, selecting the type of tactile might be crucial to achieving feasible results due to the terrain classification problem. The optical sensor has a high sensitivity, small size, light weight and low detection time [34], which are important properties to distinguish between different surfaces. However, further research is necessary to determine whether an optical sensor is suitable for terrain classification. This work utilizes similar sensor and robot platform as presented in [11], but instead of evaluating the different type of sensors, this thesis will rather investigate optical sensor with different approaches for terrain classification.

1.1 Goal of the thesis

The main goal of this thesis is to evaluate 3D optical force sensor for the terrain classification problem. This thesis will also be investigating and developing a reliable approach for data processing, preprocessing, feature selection, and classification for the presented sensor.

1.2 Outline

This thesis is divided into five additional chapters: background, software and tools, implementation, experiments and results, and discussion.

Chapter 2: Background The background chapter presents theory on which this thesis is based, including a survey of existing work.

Chapter 3: Software and tools The software and tools chapter gives an overview of the tools, programming framework and libraries used in this thesis.

Chapter 4: Implementation The implementation chapter explains the reasons behind the various choices of implementations used to preprocess data from the sensor, and evaluates a learning model.

Chapter 5: Experiments and results The experiments and results chapter presents experiments and its results along with a short analysis.

Chapter 6: Discussion The discussion chapter discusses the results from the experiments. Last is a conclusion along with the future work of this thesis.

Chapter 2

Background

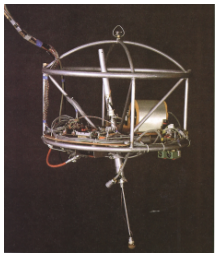
2.1 Legged robots

Legged robots have been a popular topic of robotic research, mainly due to their ability to traverse on rough terrain. Stable locomotion of legged robots is achieved through the gait. A gait is a sequence of cyclic motions of foot contacts with the ground that produce locomotion [35]. The characteristic of gait is the sequence of which legs are lifted and placed on the ground. Thus, a robot that has a variety of gaits has the ability to locomote in many different ways.

There are many types of legged robots, which are often defined by the number of legs on the robots. The following paragraphs will introduce four different types of legged robots and are organized by the number of legs in ascending order.

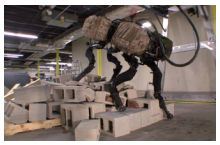
Monopod The monopod is a simple one-legged robot design. The locomotion of monopod robots is performed through hops, and hence also called "hopping robots". Having only a single point of ground contact, the challenge is achieving stability. An example of a one-legged robot is developed by Marc Raibert, shown in figure 2.1a [36].

Biped A biped is a robot with two legs. The studies on biped robots have been a popular research field, especially towards developing humanoids. Creating a humanoid implicates that the robot is able to imitate human behavior, such as walking, running, jumping, traversing on stairs, etc. The well-known NAO robot [37], shown in figure 2.1b has impressive abilities. Not only is the NAO robot able to walk, but it is also capable of seeing, hearing and speaking.



(a) One legged robot [36]

(b) The NAO robot [39]



(c) The bigdog [38].

(d) Lauron V robot [40].

Figure 2.1: Overview of different types of legged robot: monopod (a), biped (b), quadruped (c), and hexapod (d).

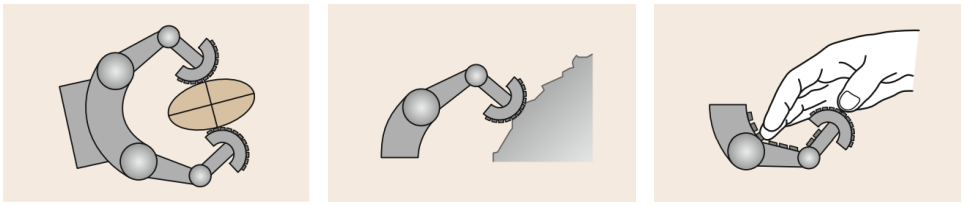
Quadruped The quadruped is a robot designed with four legs and is inspired by animals. The benefit of the quadruped robot is more easy to attain the stability of locomotion due to many legs, and therefore capable of traversing on rough terrain. For instance, the BigDog [38] shown in figure 2.1c has shown impressive performance. Hydraulic actuators make the BigDog stronger and able to carry loads from 50kg to 150kg, depending on the terrain. Other abilities include jumping, running, and maintaining stability even if it gets pushed or is walking on slippery terrain.

Hexapod A hexapod is a six-legged robot, and is inspired by insects, but mostly spiders. Having six legs provides a more stable walking system than a quadruped robot. However, the leg coordination might be more complex, due to having to control six legs. Lauron, shown in figure 2.1d is an example of a hexapod developed by The FZI Research Center for Information Technology.

2.2 Tactile sensor

Tactile sensors are designed to measure properties through direct physical interaction [41]. The tactile sensing provides many types of information to be obtained:

- **Contact** is the most simple data obtained from the sensor, which detects whether there is a touch from external agents.
- **Force** provides the amount of locally applied force.
- **Geometrical information** gives the geometrical shape of the contact area. However, it is also able to deduce the type of object in contact with the sensor, for instance, determine whether an object is spherical or cylindrical.



(a) Manipulation (b) Exploration (c) Response

Figure 2.2: Three types of application that can be used of the tactile sensor: manipulation (a), exploration (b), and response (c) [41].

- **Mechanical properties** give measurements such as slip condition, thermal or roughness of an object.

Based on the information described in the list above, the tactile sensor can be used for manipulation, exploration or response. These applications are shown in figure 2.2. Using tactile sensor for manipulation is to control, for instance, the grip force on an object. Exploration reflects the possibility of identifying objects by assimilating information about properties such as hardness, friction, and roughness from materials and surfaces. Response refers to the detection of, and reaction to, contact from external agents, and sensing if it is a gentle touch or strong impact.

2.3 Optical tactile force sensor

Optical force sensors use light reflection, based on the physical principles of light waves to measure force. Some of the first optical tactile sensors [27, 29] consist of an optical waveguide, made of transparent glass, which is illuminated along its edge by a light source, and use cameras to analyze the images. This approach is uniaxial, that is the sensor is only capable of measuring the force applied, but not the force direction. Ohka et al. [42] modified and improved the technology and further optimized it in [43, 44, 30]. The developed optical sensor consists of a charge-coupled device (CDD) camera, a light source, an acrylic hemispherical dome, and an array of rubber sensing elements. It has shown promising results, in that the sensor can identify differences in tribological behaviors between abrasive paper and a Teflon surface.

Another design and very common technology is based on Fiber Bragg Gratings (FBG) [33]. The FBG sensors are suitable for distributed strain monitoring and offer advantages such as relative measurement and linear response. By exploiting the relationship between the variation of the external force and the FBG wavelength applied, the force can be measured.

Tar and Cserey [45] presented an alternative to low-cost 3-axis optical sensors by using optoelectronic components. The design of the sensor consists of a hollow compliant convex surface made of silicone rubber, three pho-

todiodes and an infra LED based sensor. The force is measured by the deformation of the silicone rubber. For instance, if a force is applied to the silicone rubber, it will cause a deformation that changes the amount of light to each of photodiodes, which in turn will change their force vector accordingly. Using the optical method to measure the force has offered highly dynamic sensory range, low noise, and high speed operation.

Another interesting approach uses three sets of optical sensors to develop a 6-axial force sensor, developed by Hirose and Yoneda [46]. The design of the sensor is cylindrical and contains three photosensors which measure the force around the cylinder. Another and more recently design of 6-axial optical force sensors can be found in [47, 48].

2.4 Machine learning

Machine learning is the process of building a model from a dataset in order to make predictions or decisions on new datasets without being explicitly programmed to do so. Each dataset consists of a feature vector which belongs to a specified class. The training process consists of analyzing each feature vector and producing an inferred function, which is used for labeling new and unseen datasets into a class. The learning algorithm can be separated into supervised, unsupervised, reinforcement and evolutionary learning [49].

Supervised learning Supervised learning algorithms predict new data based on a labeled dataset. That is, the system in the learning process knows the correct answers of each dataset, which is also the basis for the prediction. The learning process usually stops when the algorithm converges towards an acceptable level of performance.

Unsupervised learning Unsupervised learning algorithms make predictions from data points without labels. The system has to organize the data on its own which is the basis for predictions.

Reinforcement learning Reinforcement learning algorithms choose an action for each dataset and receive a reward indicating how good the decision was. Based on rewards, the algorithm modifies its strategy in order to get the highest reward.

Evolutionary learning Evolutionary learning uses biological evolution such as reproduction, mutation, recombination, and selection as a learning process.

2.4.1 Classifier

The classifier is where the learning process occurs, and produces the inferred function. The following paragraphs will introduce the technical background of five classifiers; the artificial neural network, the support vector machine, the naive Bayes, the k-nearest neighbors, and the decision trees.

Artificial neural network

The Artificial Neural Network is inspired by neurons in the human brain. A common representation of a neuron is the perceptron shown in figure 2.3. It consists of weighted set inputs w_i, an adder which sums weighted input signals, and an activation function to decide whether it should fire for the current input x_i. By connecting many perceptrons, one will obtain a neural network. Note that neurons only depend on its own inputs and errors. That is, a neuron will not be affected by other neurons' performances. Each neuron gives a result based on its own weights and the input, adding them together, and comparing the result to its own threshold. The only thing neurons share is the input and output.

The learning process of a perceptron in supervised learning aims to be able to reproduce a particular pattern to a class, which consists of firing and non-firing neurons for a given input. If some of the neurons yield a wrong output, for instance, a neuron does not fire when it should, then its weights will be adjusted to make it fire right the next time. There is a possibility to add more layers to the neural network, which would make it able to handle non-linear separable problems. This is also called a multi-layer perceptron (MLP), or a multi neural network.

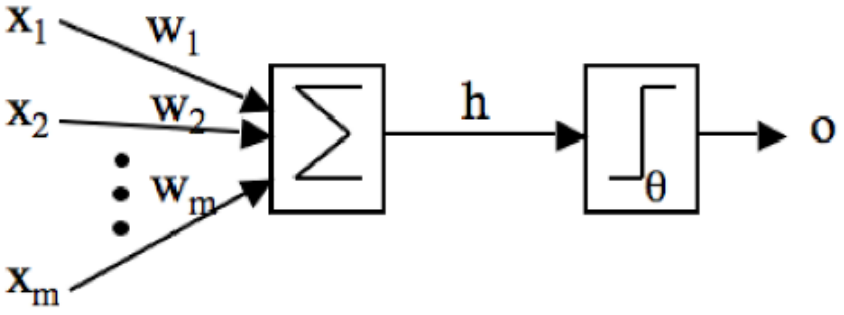


Figure 2.3: Model of a single perceptron [49].

Multi-Layer Perceptron A multi-layer perceptron consists of two or more layers between the input and output. Those layers are also called hidden layers because its value cannot be changed directly, and it is only observed in the training set. The training process can be divided into a forward

algorithm and a backward algorithm. The forward algorithm starts first by calculating the activations of the first hidden layer. These activations and the next set of weights will be used to calculate the activations for the next layer, which could either be a hidden layer or the output. The output will then be compared to a target to compute an error. The backward algorithm will use the error to adjust the weights between the output layer and hidden layer. The algorithm stops when it has reached the inputs and changed weights in the entire graph.

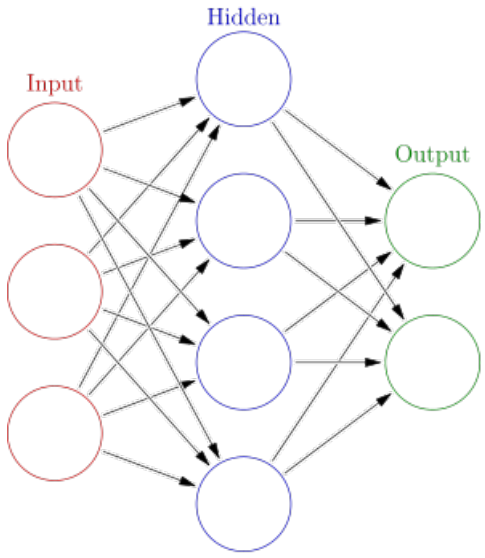


Figure 2.4: Model of a multi-layer perceptron with one hidden layer [50].

Support vector machine

The support vector machine (SVM) algorithm was introduced by Cortes and Vapnik [51], and the classifier often provides a significantly better performance than other algorithms, when the data set is not extremely large [49]. Consider the two-class classification shown in figure 2.5 where the classes are circles and crosses. The dotted line is the hyperplane/decision boundary created by SVM, and shows where each class belongs. If the decision boundary was moved by a small amount, there would be a risk that a datapoint from one class that lies close to the boundary would be on the wrong side. Finding the best decision boundary is done by defining the optimal margin. The margin is defined as the largest region where it separates the classes without having any points inside. The data points that lies on the margin are the support vectors. Finding the optimal margin can be done by using the dot product between each datapoint [51].

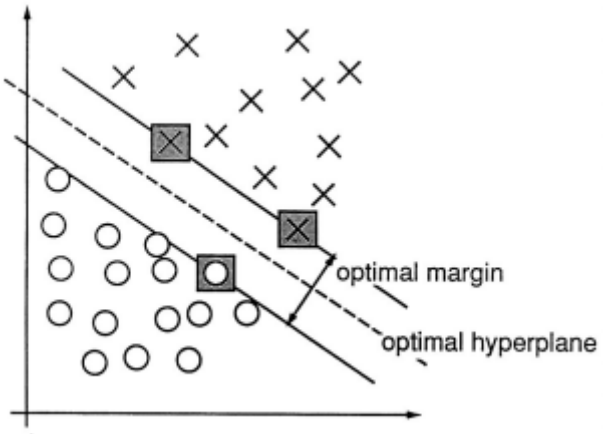


Figure 2.5: Model of optimal margin and hyperplane created by SVM on a two-class classification problem [51].

A weakness of SVM is that the classification is only feasible when the classes are linearly separable as in figure 2.5. However, this issue can be prevented by transforming the data into a higher dimensional space where the data is linearly separable as shown in figure 2.6. Recall that the decision boundary is only dependent on the dot-product of each data point, which means the transformation itself is not needed. Instead, use a function that implicitly computes the high-dimensional dot-product. This function is referred to as a kernel function. There are many types of kernel function as shown in table 2.1, where each of them is more appropriate than another depending on how complex the problem is. For instance, the dot product is sufficient if a problem is linearly separable in the original space. Meanwhile, the Gaussian radial basis function (RBF) or polynomial may be a better option for a more complex problem.

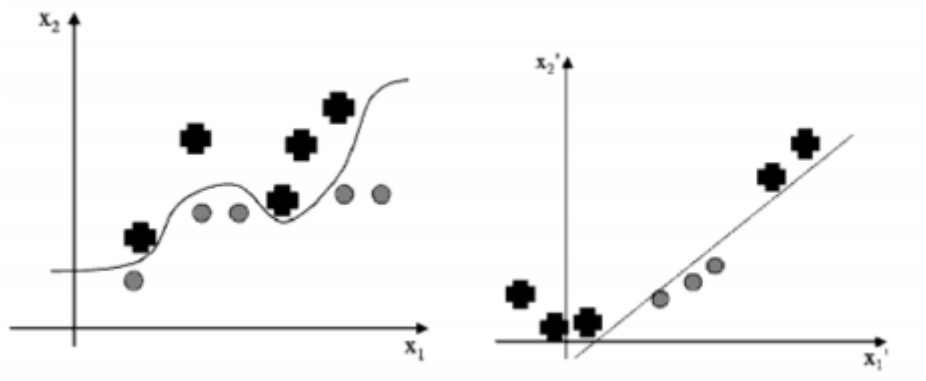


Figure 2.6: Transformation of the data from non linearly separable to linearly separable space [49].

Kernel name	Values
Linear	$\vec{x} \cdot \vec{x}$
Gaussian radial basis function	$exp(-\gamma \|\|\vec{x} \cdot \vec{y}\|\|^2)$
Polynomial	$(1 + \vec{x} \cdot \vec{y})^d$

Table 2.1: SVM kernel functions.

Naive Bayes

The naive Bayes algorithm is a probabilistic classifier based on Bayes' theorem with the assumption that the effect of a feature on a given class is independent of the values of other variables [49]. The assumption simplifies computation, hence the name naive. Consider a vector with n features, X_j, and a class, C_i, then the Bayes theorem can be formulated as in equation 2.1.

$$P(C_i|X_j) = \frac{P(X_j|C_i)P(C_i)}{P(X_j)} \tag{2.1}$$

The given output, $P(C_i|X_j)$ is the probability that the features X_j belong to a class C_i. Similarly to $P(X_j|C_i)$ which is the probability of the class C_i, belongs to this set of features X_j. $P(C_i)$ and $P(X_j)$ are the prior probabilities of C_i and X_j respectively. The problem with using the Bayes theorem occurs when the number of features increases, which requires more computation time. Thus, assumptions of independence reduce the computation time.Hence, the equation 2.1 can be reformulated as in equation 2.2.

$$P(C_i|X_j) = P(C_i) \prod_k P(X_j^k|C_i) \tag{2.2}$$

The prediction is based on selecting the class C_i with the highest probability.

K-nearest neighbors

K-Nearest Neighbors (KNN) is one of the simpler classifiers presented by Cover and Hart [52]. The classification process consists of looking at the k-nearest classes and classifying the new data as the class with the largest majority of them. Choosing the value for k will therefore be crucial to achieving high classification accuracy. An illustration of the effect of performance on different k values is shown in figure 2.7. All the blue squares and red triangles are the training data, while the green circle represents new data, which is to be classified. The solid and dashed circle is when the k is set to either 3 or 5 respectively, to illustrate the boundary between k nearest among all classes. If k is set to 3, the green circle will be classified as a triangle, which is the majority of them. Conversely, if k is set to 5, then the green circle will be classified as a square.

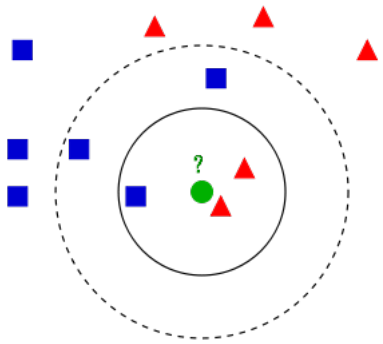


Figure 2.7: An example of a KNN classification [53]. The green circle is to be classified as either a blue square or red triangle. The solid circle is when k = 3, while the dashed circle is when k = 5.

A common method to find the k-nearest data points is to calculate the Euclidean distance. The Euclidean distance is expressed in equation 2.3 [54].

$$D(x,y) = \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2} \tag{2.3}$$

The x and y in equation 2.3 represent the actual and unseen classes, and m is the number of input variables. The algorithm will then count the k classes with the shortest distance to determine which class the unseen data belongs to.

Decision tree

The decision tree is a non-parametric classifier presented by JR Quinlan [55]. A tree, as shown in figure 2.8, consists of nodes, which represent one of the features, and leaf nodes are associated with classes. The process of the decision tree is first to construct a tree with nodes and edges based on a training data, then predict on a new data set by following a path from the root to a leaf node.

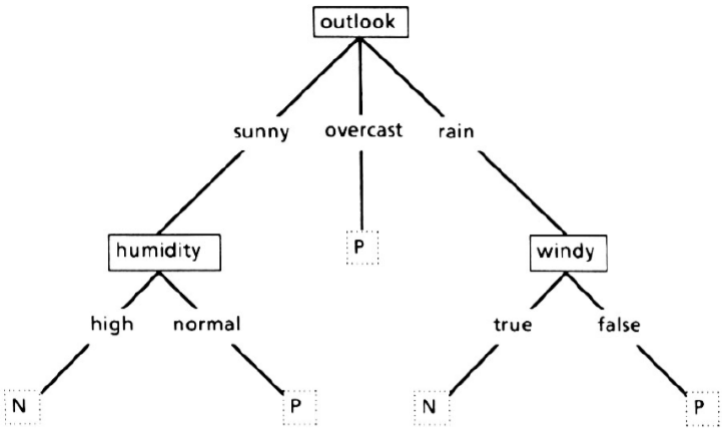


Figure 2.8: An example of a simple decision tree [55]. This decision tree classifies a day to be P or N based on outlook, humidity and wind.

An important aspect of decision trees is how to construct one based on the features. Although there are a few different methods, most are based on the same principle: by starting at the root, and choosing the most discriminative feature at each step [49]. The attractiveness with trees is that they are efficient, and it is easy to understand and interpret the data.

2.4.2 Hyperparameter tuning

Every classifier has parameters that need to be set and may strongly affect the performance induced by them. Consequently, it is recommended to set appropriate parameters in order to optimize a classifier. But it is hard to determine the optimal values of parameters since it often differs for different datasets. A strategy to finding these parameters, recommended by Hsu et al. [56], is to use a grid search. A grid search will exhaustively search through a desired specified subset and output the parameters with the best results. A benefit of using a grid search, instead of searching by heuristics or approximations, is that it avoids getting stuck in local optima. However, the biggest weakness is computation complexity when the search space increases.

2.5 Features

An important part of achieving a good classification is finding good features from sensor data. The features are what distinguish classes and will be used as a learning set. Thus, finding good features is crucial to obtaining an accurate classifier.

2.5.1 Curse of dimensionality

The curse of dimensionality occurs when one includes too many features in the input vector. The dimensionality of feature vectors will increase, and similarly the complexity of the underlying pattern may also increase, which might cause a poor performance of the classifier. To prevent the curse of dimensionality one can add more training samples to uncover the underlying pattern.

2.5.2 Features extraction

Feature extraction is the process of building a new set of features from the original set and use it as a training set. Those extracted features should make it easy for a classifier to distinguish between the various classes. A common method is extracting statistical features.

Statistical features

The following paragraphs will elaborate five commonly used statistical features; mean, variance, standard deviation, skewness, and kurtosis.

Mean The mean is generally referred to as the average, and is defined by the sum of the values divided by the number of values and is given in equation 2.4 [57].

$$\bar{x} = \frac{1}{N} \sum_{N-1}^{j=0} x_j \tag{2.4}$$

Variance Variance describes the spread between numbers in a data set [57]. The variance is given in equation 2.5.

$$Var(x_0 \dots X_{N-1}) = \frac{1}{N} \sum_{N-1}^{j=0} (x_j - \bar{x})^2 \tag{2.5}$$

Standard deviation Standard deviation is a measure of spread of a data set from its mean [57]. High deviation indicates that the data points are further from the mean. This can be calculated by taking the square root from the variance and is given in equation 2.6.

$$\sigma(x_0 \dots X_{N-1}) = \sqrt{Var(x_0 \dots X_{N-1})} \tag{2.6}$$

Skewness Skewness describes asymmetry of a distribution and is given in equation 2.7 [57].

$$Skew(x_0 \dots X_{N-1}) = \frac{1}{N} \sum_{N-1}^{j=0} \left[\frac{x_j - \bar{x}}{\sigma} \right]^3 \tag{2.7}$$

The skewness can be either negative or positive depending on whether data points are skewed to the left or the right. A negative skewness is when data is skewed to the left, while positive skewness is when the data is skewed to the right, as shown in figure 2.9.

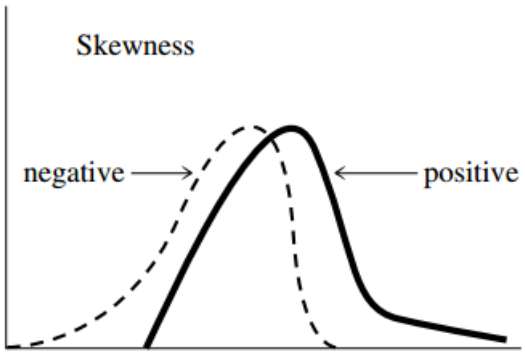


Figure 2.9: Skewness [57].

Kurtosis Kurtosis measures the peak and tails of a distribution relative to a normal distribution [57]. Using kurtosis might help to understand general characteristics about the distribution of the data. The kurtosis is given in equation 2.8.

$$Kurt(x_0 \dots X_{N-1}) = \left\{ \frac{1}{N} \sum_{N-1}^{j=0} \left[\frac{x_j - \bar{x}}{\sigma} \right]^4 \right\} - 3 \tag{2.8}$$

A positive kurtosis of distribution has a sharper peak and heavier tails relative to normal distribution, while a negative kurtosis has a flatter peak and lighter tails relative to the normal distribution which is shown in figure 2.10.

2.5.3 Features selection

The process of feature selection is to select a subset of features from the original set. Selecting good features has the benefit of increasing classifier performance, preventing the curse of dimensionality mentioned in section 2.5.1, and reducing storage requirements and training time. But, one has to be aware that even a feature that can individually be completely useless, might be relevant when used together with other features [58]. There are three types of feature selection algorithms: filter-, wrapper-, and embedded methods.

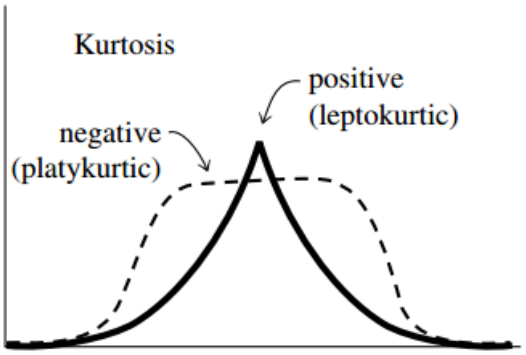


Figure 2.10: Kurtosis [57].

Filter Filter feature selection is independent of any classifier. It uses statistical measures to assign each feature a score. Features will either be kept or removed based on the score. The filter methods are considered fast and effective.

Wrapper Wrapper feature selection will try different combinations of the feature set, evaluate by a classifier and keep the feature set with the best outcome.

Embedded Embedded feature selection performs feature selection as part of the learning procedure and is usually specific to a given classifier.

2.5.4 Features scaling

Different features often have a different range of values which may cause a skew in the distribution. This is an issue particularly for classifiers that involve distances in their computation such as SVM and KNN described in section 2.4.1. When a feature has a large range, it will dominate other attributes and cause poor performance of the classification. To reduce bias effect caused by skewed distributions, it is common to standardize the feature vectors. The standardizing weights all feature equally in their representation. A common way is to standardize the feature vector to zero mean and unit variance as given in equation 2.9, where x, $\bar{x}$ and σ are the feature to be standardized, mean and standard deviation, respectively.

$$z = \frac{x - \bar{x}}{\sigma} \tag{2.9}$$

2.6 Model validation

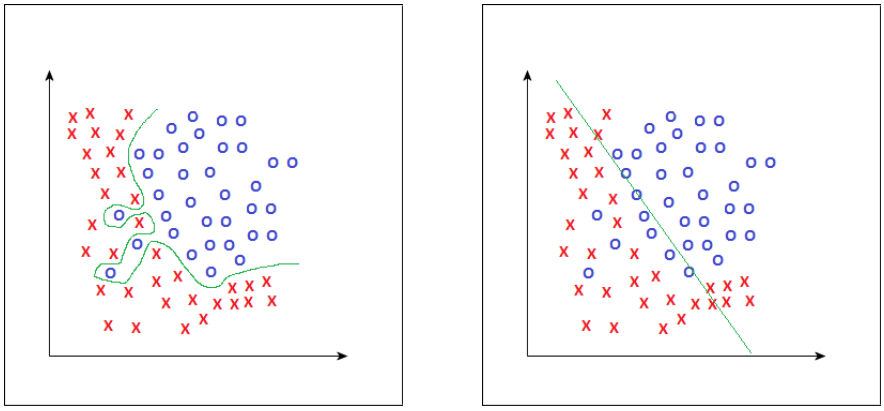
Model validation relates to evaluating the performance of a classifier.

2.6.1 No Free Lunch Theorem

The well-known No Free Lunch theorem [59] in machine learning states that there is no best classifier for every problem. That is, even if a model achieves great performance for one problem, it might not hold for another problem. Thus, it is recommended to apply several different classifiers for various problems.

2.6.2 Overfitting and underfitting

Overfitting and underfitting the data is an issue in machine learning which causes poor performance of classification. Overfitting occurs if the learning process is done too extensively, which might make the classifier adapt about to inherent noise in the training set as shown in figure 2.11a. Meanwhile, underfitting occurs if there is not enough training data and the classifier will not be able to generalize a new data set as shown in figure 2.11b.The cross-validation estimates how accurately the classifier model will perform in practice, which might prevent the model from overfitting or underfitting.



(a) Overfitting (b) Underfitting

Figure 2.11: Illustration of when a model is overfitted 2.11a, and underfitted 2.11b.

2.6.3 Cross-Validation

Cross-validation is a statistical method to assess the quality of learning models [60]. The process of cross-validation is first to remove some of

the data before the training begins. After the training, the model will use the removed data to test the performance. The intention is to evaluate the classifier performance in a more realistic scenario by predicting new and unseen data. The K-fold is a common cross-validation method.

K-fold The process of K-fold is to partition the data into k subsets, where one subset is used for testing, and the other is used for training. When the trained model has assessed the test set, a new subset is selected as the test set. This process will be repeated k times, that is when all subsets have been used as a test set. Setting k to the length of feature vectors is also known as leave-one-out cross-validation (LOOCV). LOOCV only uses one feature vector as a test set, with the remaining as a training set as shown in figure 2.12. Estimations based on LOOCV tend to be almost unbiased, but unreliable due to high variance. However, it is widely used especially when there are only small amounts of data available in order to use as many training samples as possible.

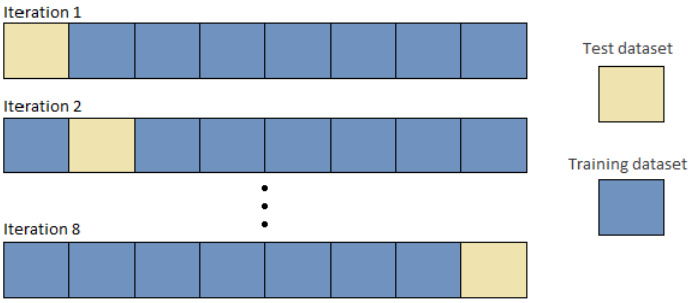


Figure 2.12: An instance of k-fold cross validation. In this case it is the LOOCV, since the k is set to be 8 which is the length of the feature vectors.

2.6.4 Evaluating Classifiers

Evaluating the performance of a classifier can be done by calculating metrics based on correct or wrong outputs. For instance, a two-class classifier with classes "positive" and "negative" will have four different outcomes:

1. True Positive (TP) is a correct prediction of class positive.
2. True Negative (TN) is a correct prediction of class negative.
3. False Positive (FP) is a wrong prediction of class positive.
4. False Negative (FN) is a wrong prediction of class negative.

These four variables can be further used to calculate the precision, accuracy, recall and f-score.

Precision Precision gives the number of correct detected class members as given in equation 2.10.

$$Precision = \frac{TP}{TP + FP} \tag{2.10}$$

Accuracy Accuracy gives the ratio of correct to incorrect predictions as given in equation 2.11.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.11}$$

Recall Recall gives the number of detected actual class members as given in equation 2.12.

$$Recall = \frac{TP}{TP + FN} \tag{2.12}$$

F-score F-score is a balanced measure of recall and precision as given in equation 2.13.

$$F\text{-}score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.13}$$

The accuracy gives an indication of the overall performance of a model. However, there is a possibility that the model only classifies three of four classes correctly and still gets a high accuracy. Thus, it will be recommended to analyze the f-score. A low f-score indicates that the class either has a low precision, recall or both. A low precision score indicates that the classifier has difficulty in predicting the current class, while low recall indicates that it is more likely that a class is to be classified as other classes.

A confusion matrix gives an insight of which classes were easier to predict than others. The confusion matrix consists of a square matrix with one row for predicted class, and a column for actual class (or vice-versa). Consider the confusion matrix of the 3-class classification in table 2.2. The model got the correct classification of all instances that belongs to class C_1. Meanwhile, the model misclassified five instances of C_2 as C_3, and is not able to classify any instance that belongs to class C_3 correctly.

		Actual		
		C_1	C_2	C_3
Predicted	C_1	8	0	0
	C_2	0	8	5
	C_3	5	3	0

Table 2.2: Confusion matrix of a 3-class classification.

2.7 Existing work on terrain classification

Terrain classification has been applied to both wheeled and legged robots. Wheeled robots have the benefit of achieving stable locomotion by changing their speed on different terrains, while the legged robots must either change their gait, walking speed or both. Changing the gait for a legged robot can be complex, but has the benefit of being able to traverse on more difficult terrains.

The most commonly used legged robots in terrain classification are quadrupeds [11, 6, 15, 17] and hexapods [16, 9, 4], due to their stability on rough terrain. However, there are few studies where the monopod [61] and the biped [62] have been used.

2.7.1 Terrain sensing

In order to classify various terrains, the system must obtain information from the terrain either by remote sensing, local sensing or both.

Remote sensing Remote sensing obtains information about a terrain from a distance and does not measure the terrain physically, such as with cameras and laser scanners.

Filitchkin et al. [6] presents a visual terrain classification by using a single, compact camera to change the gait patterns of a quadruped robot. Three types of gait were used during the experiment, a gait designed for a flat surface, a gait for rough terrain and a mixture of the former two. To know which gait should be chosen for each terrain, an initial test by assigning a gait to each terrain type was required. There were in total four different terrains: small rocks, rocks, grass, and tile. The experiment consists of letting the robot identify the terrain every few steps and switch gait according to which terrain it was on. Robot performance was measured by comparing the traversal time between each gait independently and the changing gait. The results show that the changing gait is able to classify terrain and traverse through the terrain faster. Meanwhile, the other two gaits were quick, but not able to classify big rocks, and the last gait was able to classify all terrain but had a slower traversal time.

Plagemann et al. [8] used a laser ranger finder to predict terrain elevation at unseen locations. The research extended the Gaussian process model to achieve a more accurate prediction of elevations. The results show that the proposed method is capable of accurately predicting elevations unobserved even in the presence of noise. These features gave the possibility of planning the foot trajectories of the robot to reach a goal location

A weakness of using the remote sensors is that it does not give insight into

the characteristics of the current terrain itself. For instance, remote sensors might have difficulties distinguishing between terrains that are covered with either compacted or uncompacted snow. Thus, a preferable option is to measure terrain directly by local sensing.

Local sensing Local sensing measures aspects of the interaction between the robot and terrain when the robot walks through. This gives a measurement of mechanical surface properties and provides useful information such as how the environment is currently affecting robot performance.

Walas [12] attached a 6-axis tactile sensor on a hexapod to discriminate five different terrains which were soft ground, artificial grass, gravel, pebbles, and sand. The experiment's design was to classify the terrains while the robot locomotes on it. There were in total 10 trials for each terrain where each trial consisted of six steps. The author investigated the performance of each single signal from the tactile sensor independently. The results showed that the information from the force in the x and y-directions and torque in x, y, and z-directions did not give informative properties from the terrain, where the highest accuracy was less than 60%. However, by using data from the force and torque in the z-direction, the author achieved an accuracy of 76.67%.

Wu et al. [13] designed a capacitive tactile sensor mounted on a small two-legged robot. Six different terrains were used in experiments, but the terrains were further grouped into four terrain classes based on their friction and stiffness properties. The four classes consisted of a high friction hard surface class, a low friction hard surface class, a deformable surface class, and a granular class. The results show that the proposed tactile sensor, in combination with motor torque and robot gait, gave an accuracy of over 90%. The author concluded that the tactile sensor was one of the most useful sensors due to perceiving informative features from the terrain.

Stejskal et al. [63] presents a road-following hexapod robot which uses the feedback from robot servo drives that provided information about the leg motion. The road following consists of letting the robot blindly walk on asphalt. After each gait cycle, the robot will determine whether it was on new terrain or on asphalt. If it is determined as off-road, then the robot will steer back to the asphalt. Three different terrains were used in the experiments: asphalt, dirt, and grass. The results show that the robot was most confused by dirt, which accounted for about 86% of misclassified samples. The author states that the confusion is because of similar leg motion when the robot either locomotes on asphalt or dirt. However, the overall result of terrain classification had an accuracy of 96.2%, which can be considered as a feasible approach.

Kim et al. [61] used the ground reaction force and torque sensors of a one-legged robot for terrain classification. The goal of the research was

to compare the performance of the neural network and the support vector machine. Four different terrains were used in the experiments: flat, grass, sand, and gravel. The sensor data was collected by walking through each terrain many times. Different features were extracted from the sensor data and further partitioned into a training and test set. The results show that the support vector machine achieved an accuracy of 78.75%, which was a slightly better performance than the neural network with an accuracy of 78.6%.

Hoepflinger et al. [64] present a novel approach to terrain classification for legged robots by using properties from joint motor currents and force sensing resistors. The goal was to improve the guiding of foot placement and stability of legged robots in rough terrain. Usually, experiments are done by having a robot walk through terrains. However, in this experiment, the author separated one of the robot's legs and mounted it to a sample holder of a testbed. The work designed two experiments, where different properties of surfaces were investigated. The first experiment consisted of distinguishing four different shapes of terrain: a convex and a concave cone, a convex hemispherical bulge, and a concave hemispherical indentation. Meanwhile, the second experiment was to distinguish between three different types of surfaces such as abrasive paper and a low friction PTFE coating. The sensor data from both experiments was collected by performing a scratching motion on the terrain. The results from the first experiment show that the presented approach was able to distinguish between different terrain shapes with an accuracy of 93.8%. Results from the second test, on the other hand, show that the presented approach is had difficulty distinguishing different types of abrasive papers and led to an accuracy of 73.3%.

2.7.2 Learning algorithms

There is a vast number of classifiers and a variety have been used within terrain classification, such as neural networks [11, 14, 65], adaptive Bayesian filtering [66, 15], support vector machines [61, 67, 68] and decision trees [15]. The No Free Lunch theorem described in section 2.6.1 appears in previous work [10] that SVM, KNN, and naive Bayes gave higher accuracies than the decision tree, while in [15] better performance was achieved by SVM, decision tree and naive Bayes than by KNN. Thus, it is recommended to build several algorithms, and choose the best of them for the specified problem.

Most studies based their terrain classification on supervised learning. However, Giguere and Dudek [2] presented a new clustering method for terrain classification using unsupervised learning. This makes a robot able to automatically distinguish terrain without any human interaction or feedback. The same authors [14] designed a tactile probe and demonstrated the possibility of utilizing the sensor in unsupervised learning.

2.7.3 Features

As mentioned in section 2.5, finding good features is a crucial part of the classification process. Earlier work has extracted features in combinations of statistical features in the time domain with frequency domain features [69, 2, 64]. Other researcher only extract features in the time domain [12] or in the frequency domain [70, 71]. The following paragraphs will give an insight of features extracted in past work.

Giguere and Dudek [69] extracted features such as mean, variance, skewness, kurtosis, fifth moment, and the sum of the variation over time in the time domain. The feature in frequency domain consists of the sum of higher half of amplitude spectrum extracted.

Hoffmann et al. [17] defined features in the time domain such as minimum, maximum, mean, kurtosis, skewness, median, standard deviation, the approximation of the integral, amplitude of Hilbert transform. Other features were extracted in the frequency domain such as the frequency with the highest amplitude and its magnitude, similar to the second and third highest amplitude.

Kertész [72] computed median, maximum, skewness and root mean square from of the accelerometer angles in x, y and z-directions. Those features were also extracted in the frequency domain for the z-direction. Features extracted from force sensors were interquartile range, maximum, skewness, RMS amplitude and the highest amplitude in the frequency domain.

Best et al. [9] extracted five statistical features in the time domain such as minimum, maximum, mean median, and standard deviation. However, some statistical features are also calculated in the frequency domain with the energy additionally.

Walas [12] suggested four statistical features in the time domain which were variance, skewness, kurtosis, and the fifth moment from the 6-axis tactile sensor.

The paragraphs above shows that many of previous work has been using statistical features, where some of them are described in section 2.5.2. The work in [61] is an example of extracting good features is crucial to good performing model. The researcher used statistics with a support vector machine and gave an accuracy of 40%, while a principal component analysis gave an accuracy of 78.75%.

2.7.4 Model evaluation

A common method to evaluate a model is using the k-fold cross-validation [73, 11, 3, 17, 15, 72]. However, the selection of k varies. A common k value

is set either to 2 [73, 11], 10 [9, 3, 17, 15, 72] or equal to the length of the feature vectors [9].

Mrva et al. [73] used 2-fold cross-validation and achieved an overall accuracy of 99.4%, and states the possibility of obtaining 100% with more folds. Best et al. [9] used 10-fold cross validation and achieved an accuracy of 99% while the leave-one-out-cross validation decreased slighty the classifier performance with accuracy of 97.4%. The studies seems to achieve feasible results. However, as stated in [72], the k-fold only gives a reasonable estimate of performance. That is, it does not give insight on how well it predicts with unseen data. This is because experiments always use the same samples which are involved in either training or testing. The model might be less generalized, because the data more likely to refer to itself, and there might be difficulty with predicting unseen data. Most authors are aware of this issue and have rectified it by partitioning the samples to make the training and test sets independent [9]. The learning process will only be used from a certain set, and testing from another set.

Not all studies evaluated their models by k-fold cross-validation, but also either with new data to get a better estimation [64, 16, 14]. A simpler method of validation is to randomly partition the data set with 70% used for training, and the rest for testing [14]. The most realistic scenario is to base the evaluation on a robot traversing through different terrains [73].

Chapter 3

Software and tools

This chapter gives an introduction to different tools and libraries used throughout the thesis.

3.1 Optical force sensor

OptoForce (3D force sensor) [74] shown in figure 3.1 is a similar sensor to that of Tar and Cserey [45].

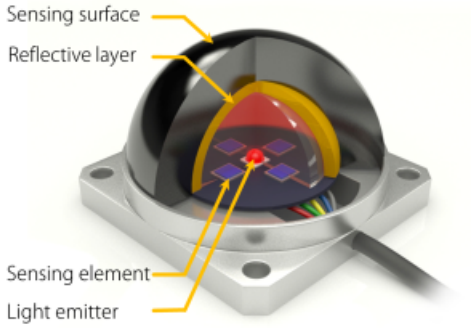


Figure 3.1: 3D OptoForce sensor [75].

The sensor consists of a light emitter (LED) and four sensing elements (photodiodes) which are wrapped within two layers; a reflective layer and a sensing surface. The four photodiodes obtain the force by measuring the infrared light reflected by the reflective layer. If a force is applied on the sensing surface, the amount of reflected light on each photodiode will change accordingly. The forces in the x- and y-directions are measured by the difference in the amount of reflected light between the two opposing photodiodes for each direction, while the force in the z-direction is the average of the four measurements. The force axes of the sensor are shown in figure 3.2. OptoForce sensor is a relatively new sensor (2015), and the

manufacturer claims the sensor can guarantee precise measurements even up to a 200% overload [76].

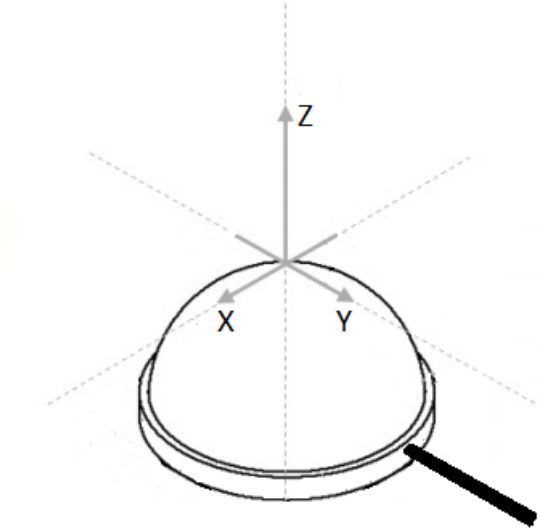


Figure 3.2: Force axes of the OptoForce sensor.

3.2 Robot

A Quadruped robot developed at the University of Oslo is shown in figure 3.3. The legs are about 45 cm long and mounted with optical force sensors. The robot also provides five different gaits developed by Nygaard et al. [77]. All gaits are optimized by an evolutionary algorithm. To elaborate on each gait, the gait names will be simplified as letters. Gait A is optimized to achieve the highest speed. This makes the robot walk very fast and is not very stable. Gait B is optimized to achieve the most stable locomotion, which leads to a slow speed. The remaining three gaits are all optimized to achieve both stable and fast locomotion. However, gait C tend to favor on achieving a faster speed rather than stability. Conversely, gait D has tend to favor on achieving a stability rather than faster speed. Meanwhile, the gait E is a mixture of the gait C and gait D.

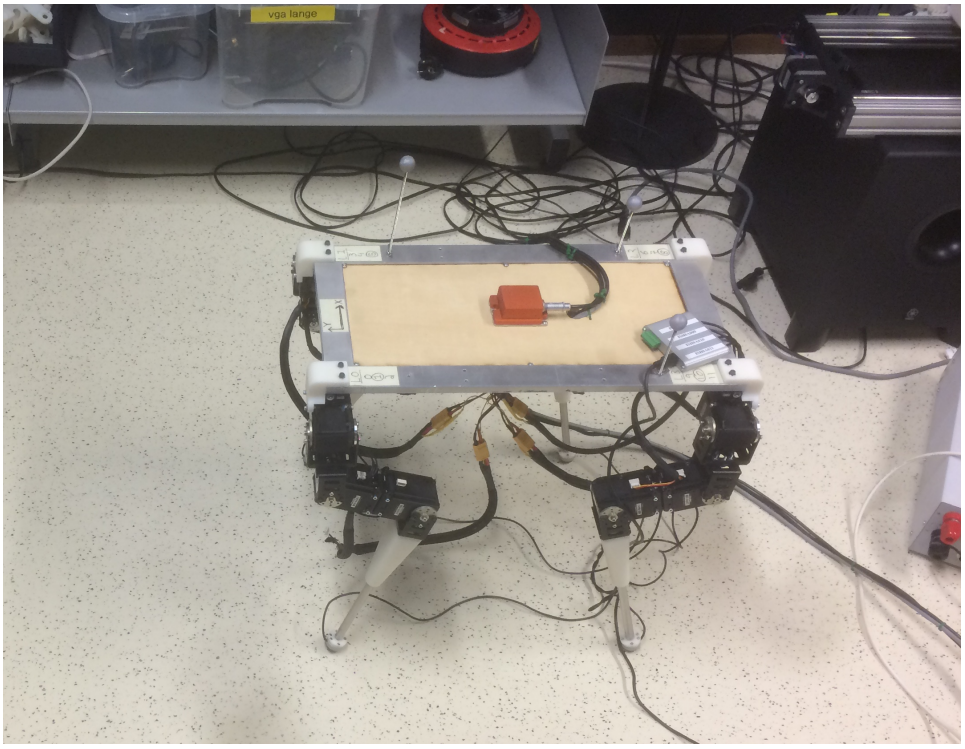


Figure 3.3: Quadruped Robot developed at the University of Oslo mounted with the optoforce sensor.

3.3 Robot operating system

The robot mentioned in section 3.2 operates on a Robot Operating System (ROS) [78]. ROS is an open-source software framework and is used to create robot applications. ROS consists of nodes that can be grouped into packages, shared and distributed. A node is a process that performs the computation. For instance, one node could be controlling the robot, while another controls the prediction of terrain. This allows parallel execution of the data collection and class prediction in real time. Currently, the most compatible programming languages are Python, C++, and Lisp.

3.3.1 Messages and topics

A message is a data structure which could be an integer, floating point, array, etc. Nodes can use a message and publish it under a given topic [79]; this feature is also called "publisher". The topic is used to identify the content of the message where a node has the possibility to subscribe to the topic and receive data, which is also called "subscriber." This can be seen as many-to-many, one-way communication. That is, many nodes can subscribe to a topic, but they do not have the opportunity to pass any message back to the publisher.

3.3.2 Services

Service and client give the possibility of a request and reply interaction between two ROS nodes [79]. A service can be either requesting or replying to the client. The client calls the service by sending the request message and awaiting the reply.

3.3.3 Rosbag

Rosbags are used for recording and playing back ROS message data [79]. This has the benefit of storing sensor data which is necessary for developing and testing algorithms.

3.3.4 OptoForce package

A ROS package for the Optoforce sensor described in section 3.1 can be found in [80]. The package contains a node that is able to read data from the sensor with a frequency of 100Hz and will publish the data stream as floats to a topic. This gives the opportunity for other nodes to obtain data from the sensor by subscribing to the node.

3.4 Python libraries

Python provides lots of libraries which reduce developing time. The libraries are well-documented and have the freedom to customize each algorithm to use.

Scikit-learn Scikit-learn [81] is an open source library that provides tools for data mining and data analysis. The library is dependent on numpy, scipy and matplotlib. All classifiers and preprocessing data tools can be found in this library.

Runstats Runstats [82] is a library which computes statistics, such as max, mean, skew, variance and standard deviation, some of which are described in section 2.5.2.

Chapter 4

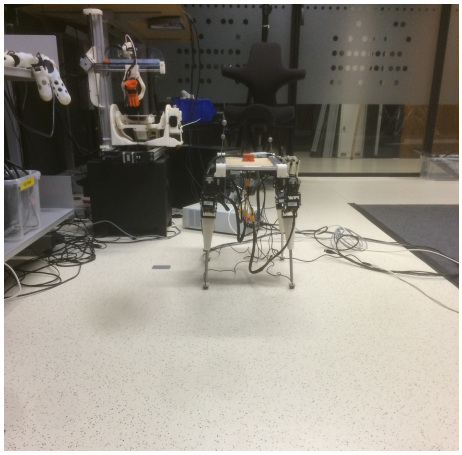
Implementation

This chapter attempts to present the environment setup and explain the various choices which have been made to create machine learning models.

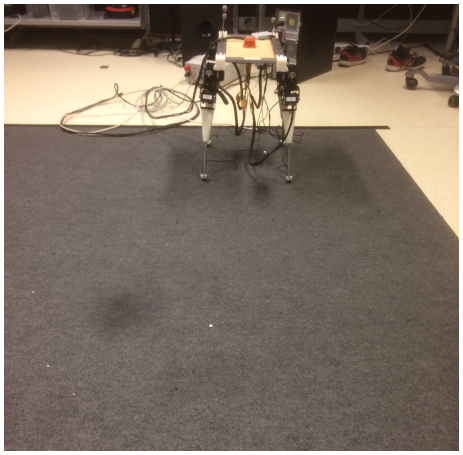
4.1 Environment setup

Four different terrains are used in the experiments: floor, carpet, soft mat, and hard mat as shown in figure 4.1. A reason for choosing these terrains is to create a more challenging task for the classifier, because of their similar properties. The assumption is that if the algorithm manages to discriminate between floor and carpet, it should be able to distinguish other terrains as well. Floor, carpet and hard mat have the most similar properties and also the most difficult to identify. They are all slippery and have nearly equal hardness. Soft mat, on the other hand, differs from the others with its softness and high friction. The experiments will be investigating whether the classifier can distinguish these terrains with minor differences.

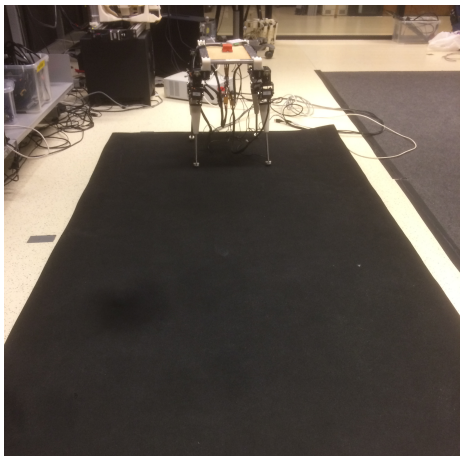
The quadruped robot and the optical force sensor used in this thesis are mentioned in chapter 3. Although the robot provides five different types of gaits as described in section 3.2, only gait E is used in all experiments. The reason is that the other gaits had issues when walking on the soft mat. The robot either got stuck with a slow gait, or fell with a fast gait.



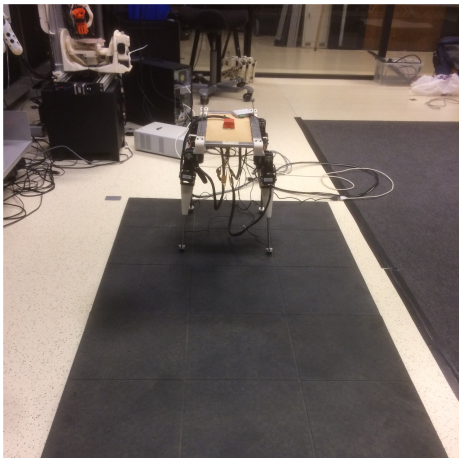
(a) Floor



(b) Carpet



(c) Soft mat



(d) Hard mat

Figure 4.1: Four different terrains used in all experiments: floor (a), carpet (b), soft mat (c), and hard mat (d).

4.2 Choice of implementation language

The robot operates on ROS which is currently only compatible with C++, Python or Lisp. Thus, segmentation of data, described later in section 4.3.3, is implemented in C++, due to fast computation. However, C++ does not provide many learning libraries. Hence the learning algorithm is implemented in Python.

4.3 Data from optical force sensor

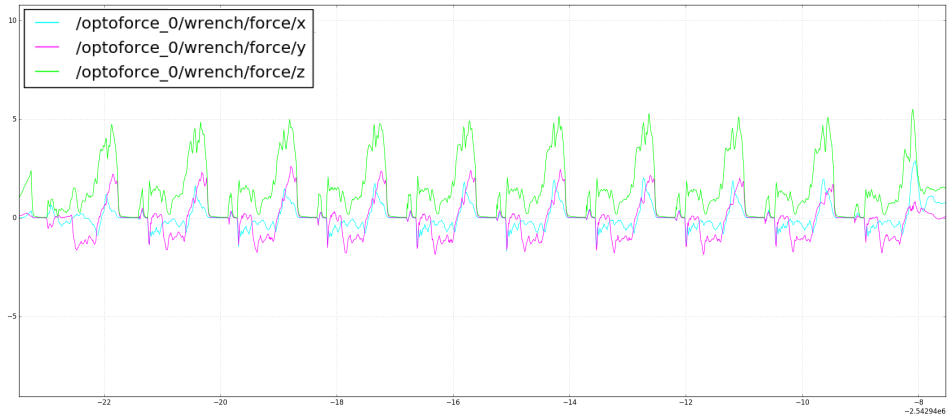
This section will first describe how the measurement from the sensor is obtained. Next, it will present how the data is segmented into sequences and used as the basis for creating feature vectors.

4.3.1 Data collection

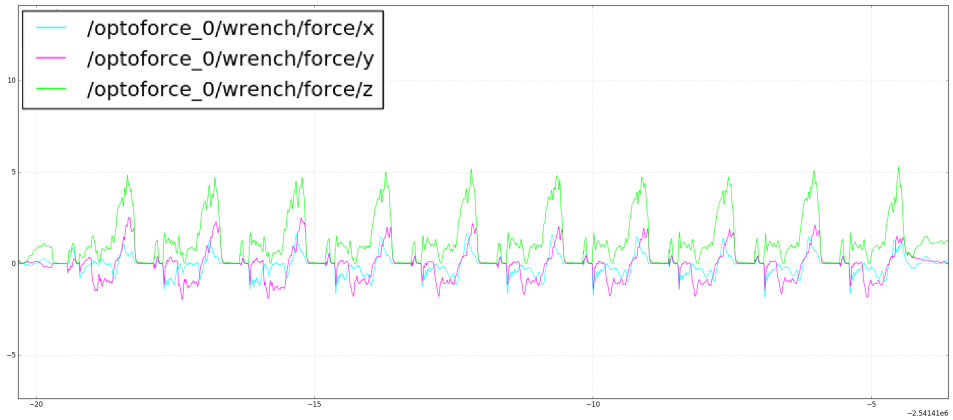
Sensor data was obtained by having the robot walk 10 steps on each terrain. The trials were recorded into rosbags, which makes it possible to re-simulate the trials. Five trials were recorded for each terrain, which is in total 20 trials. This gives in total 200 steps on each terrain from one sensor, and 800 steps with all four sensors together. However, this thesis will only use sensor data provided from the front left foot of the robot to evaluate the performance of each classifier.

4.3.2 Analyzing the sensor data

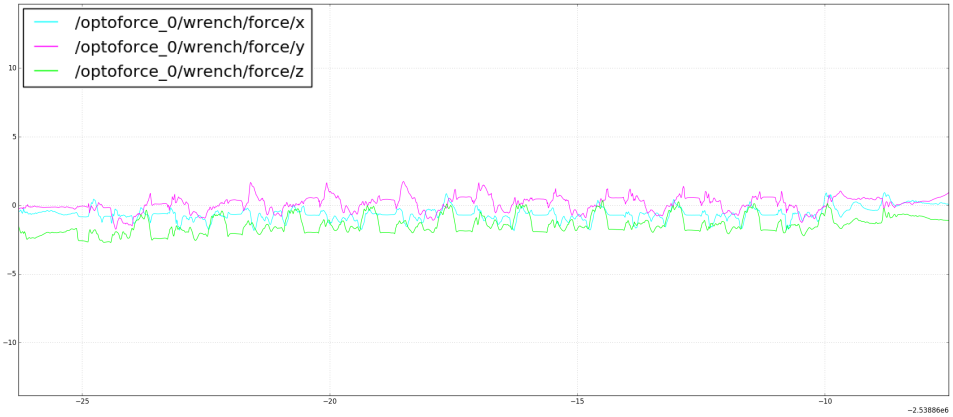
The analyze of the sensor data aims to find common characteristics, and to be able to segment desired data. Sensor data arrives in a stream as shown in figure 4.2. The periodic sequences are for each step. A common characteristic of all terrain is when the foot is in the air. There is no contact between the sensor and the surface and will give a minor change, almost constant, in the sensor data. When there is a contact between the sensor and a surface it provides a big force variation in the all three directions. This characteristic will be used to segment desired data, described in the next section 4.3.3.



(a) Floor



(b) Carpet



(c) Soft mat

(d) Hard mat

Figure 4.2: Example of sensor data from each terrain: floor (a), carpet (b), soft mat (c), and hard mat (d).

4.3.3 Data segmentation

An appropriate method to segment sensor data is using a sliding window algorithm. However, it is difficult to decide on an acceptable size of the window, and to determine whether the data is relevant from the window, and windows which are too big are inefficient. Thus, a custom algorithm to segment data was created.

It is considered that the most informative data of terrain is taken when the foot is on the ground. The custom algorithm will be storing the data sequences when the foot is on the ground, and stops when it is not on the ground. As mentioned in section 4.3.2, a characteristic for all terrain when a foot is in the air, is that the x, y, and z-direction to a sensor have minor changes in their values within a short sequence. Using this property gives the possibility to determine when a foot is either in the air or on the ground. In the implementation, two thresholds have to be set; one to consider the minor change in each direction, and one to consider the minimum length of the minor change sequence, which are set to 0.009 and 15 respectively. That is, when the data sequence consists of at least 15 elements, and the difference between the current data and its neighbor do not exceed 0.009, it is considered that the foot is in the air. When a foot reaches the ground, there will be a big change in each direction for the sensor, and the algorithm will start to store the data from the sensor into an array until it is in the air again. Each step will be used as a sample in the training and test sets. The source code of segmenting desired sensor data is given in appendix A.

4.4 Feature sets

As mentioned in section 2.5, a good classifier is dependent on good features. As most previous works have extracted features both in the time domain and frequency domain, this thesis will also be extracting in both domains. Five different feature sets will be created and applied to learn and evaluate each of classifiers. The following paragraphs will introduce each feature set.

Feature set one - raw data This feature set uses all data from each step in the x, y, and z-directions. As the length of sensor data varies, the feature vector is decimated in the front and end of the sequence to achieve a fixed length, in this case, 125. The choice of sequence length and method of decimating is described and reasoned in the next section 4.5. The feature vector set is shown in equation 4.1.

$$f_{set1} = \{x_1, \ldots, x_{125}, \\ y_1, \ldots, y_{125}, \\ z_1, \ldots, z_{125}\} \tag{4.1}$$

The vector contains 125 features from all 3 directions, which is in total 375 features.

Feature set two - statistical features This feature set extracts statistical features from the dataset. As elaborated in section 2.7.3, much of the early work has utilized statistical features for the terrain classification. Thus, this thesis will be extracting similar features. Calculation of some statistical metrics are described in section 2.5.2. The features created in this set will be from each direction, x,y and z in the time domain:

1. The maximum value of the dataset in the time domain
2. The minimum value of the dataset in the time domain
3. The mean of the dataset in the time domain
4. The variance of the data set in the time domain
5. Skew in the time domain
6. Kurtosis in the time domain
7. Standard deviation in the time domain

The feature set is shown in equation 4.2.

$$f_{set2} = \{x_{max}, x_{min}, x_{skew}, x_{kuortosis}, x_{std}, x_{var}, x_{mean}, \\ y_{max}, y_{min}, y_{skew}, y_{kuortosis}, y_{std}, y_{var}, y_{mean}, \\ z_{max}, z_{min}, z_{skew}, z_{kuortosis}, z_{std}, z_{var}, z_{mean}\} \tag{4.2}$$

The vector contains 7 features from all 3 directions, which is in total 21 features.

Feature set three - complete frequency spectrum Previous work has shown that using frequency has given promising results. In this feature set, the raw data from the time domain is transformed into the frequency domain by fast Fourier transformation. After transformation, a decimation is used to achieve a fixed length. Since the minimum length in the time domain is 125, and the fast Fourier transform is symmetric, the minimum length of the entire spectrum will be 62.5, but in this approach, the thesis will be decimating to a length of 61. Contrary to decimating in the front and end of the sequence, it is considered that data at the end of the sequence is

less important than in the front, hence these features will be discarded. The feature set is shown in equation 4.3.

$$\begin{aligned} f_{set3} = \{&fx_1, \ldots, fx_{61}, \\ &fy_1, \ldots, fy_{61}, \\ &fz_1, \ldots, fz_{61}\} \end{aligned} \tag{4.3}$$

The vector contains 61 features from all 3 directions, which is in total 183 features.

Feature set four - statistical features This feature set computes the statistical metrics similarly to feature set two but in the frequency domain. Additionally, the energy of the spectrum suggested in [9], by calculating the sum of the squares of the amplitudes, is also included.

1. The maximum value of the dataset in the frequency domain
2. The minimum value of the dataset in the frequency domain
3. The mean of the dataset in the frequency domain
4. The variance of the data set in the frequency domain
5. Skew in the frequency domain
6. Kurtosis in the frequency domain
7. Standard deviation in the frequency domain
8. Energy

The feature vector is shown in equation 4.4.

$$\begin{aligned} f_{set3} = \{&fx_{max}, fx_{min}, fx_{skew}, fx_{kuortosis}, fx_{std}, fx_{var}, fx_{mean}, fx_E, \\ &fy_{max}, fy_{min}, fy_{skew}, fy_{kuortosis}, fy_{std}, fy_{var}, fy_{mean} fy_E, \\ &fz_{max}, fz_{min}, fz_{skew}, fz_{kuortosis}, fz_{std}, fz_{var}, fz_{mean}, fz_E\} \end{aligned} \tag{4.4}$$

The vector contains 8 features from all 3 directions, which is in total 24 features.

Feature set five - set two and four Good features might come from different feature sets. This set will collect feature sets two and four into

one unified set as seen in equation 4.5.

$$\begin{aligned} f_{set5} = \{ & x_{max}, x_{min}, x_{skew}, x_{kuortosis}, x_{std}, x_{var}, x_{mean}, \\ & y_{max}, y_{min}, y_{skew}, y_{kuortosis}, y_{std}, y_{var}, y_{mean}, \\ & z_{max}, z_{min}, z_{skew}, z_{kuortosis}, z_{std}, z_{var}, z_{mean}, \\ & fx_{max}, fx_{min}, fx_{skew}, fx_{kuortosis}, fx_{std}, fx_{var}, fx_{mean}, fx_E, \\ & fy_{max}, fy_{min}, fy_{skew}, fy_{kuortosis}, fy_{std}, fy_{var}, fy_{mean} fy_E, \\ & fz_{max}, fz_{min}, fz_{skew}, fz_{kuortosis}, fz_{std}, fz_{var}, fz_{mean}, fz_E \} \end{aligned} \tag{4.5}$$

The vector contains 15 features from all 3 directions, which is in total 45 features.

4.5 Achieving a fixed length of the sequences

In order to use raw data as input, a fixed length is required. After many trials, the length of the data sequences varies between 125 and 135. Thus, to ensure that all sensor data from each step is obtained the data samples will be decimated to a constant length of 125. A simplified method for achieving a fixed length is in listing 4.1. Note the technique is simplified to a one-dimensional array, while this thesis uses a 3-dimensional array, but the intention is the same. Contrary to discarding the data either in the front or end of the sequence, this technique will be discarding on both sides, to retain the data in the middle, which is considered more informative data.

```
def fix_length(data_sequence):
    fixed_length = 125
    total_length = len(data_sequence)

    #Total amount data to be removed
    total_cut = total_length-fixed_length

    #In case the sequence length is odd
    rest = total_cut%2

    #Calculate the start and end position
    start_pos = total_cut/2
    end_pos = total_length - (total_cut/2)+rest

    #Retrieve fixed length of sequense data
    fixed_sequence = data_sequence[start_pos:end_pos]
    return fixed_sequence
```

Listing 4.1: A simplified method to achieve a fixed length of data sequence

4.6 Evaluation procedure

This section presents two processes which are used to evaluate a certain classifier. The first process is collecting sensor data samples into a single file. The second process is using the file to evaluate a certain machine learning model.

4.6.1 Collecting data samples

The first process shown in figure 4.3 is to gather data for each terrain which will be stored into files. Those files will further be collected into a single file and used as the training and test samples in the next process.

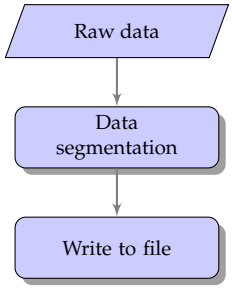


Figure 4.3: Process of storing data samples.

Process of storing samples

1. **Raw data**
   Raw sensor dataset is retrieved from the optical force sensor.
2. **Data segmentation**
   The sensor data will be partitioned into samples as described in section 4.3.3.
3. **Write to file**
   This step will write each data sequence into a file labeled with the terrain name.

4.6.2 Create and test machine learning model

When the data samples of each terrain are unified into a single file, the learning process as shown in figure 4.4 is to evaluate each classifier. Rectangles are processes that take a given input and produce an output. The dashed rectangles are processes that can be omitted.

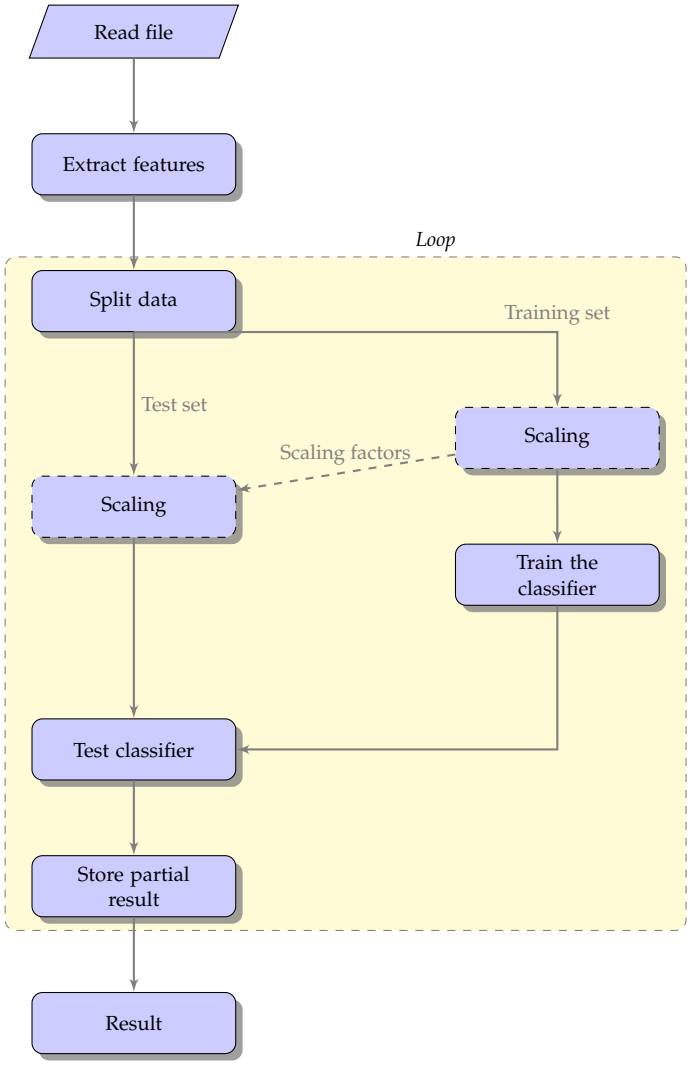


Figure 4.4: Process of creating and evaluating a classifier.

Details of evaluating a classifier

1. **Read file**
   This step will take a file created from section 4.6.1 as input.
2. **Extract features**
   This step will create one of the five feature sets as mentioned in section 4.4.
3. **Loop**
   (a) **Split data**
   The extracted feature vectors are partitioned into training and testing sets. The thesis will be using LOOCV described in section 2.6.3. Thus training set will consist of 199 samples and only one sample in the test set.
   (b) **Scaling**
   This step takes the training set as input and standardizes the data as mentioned in section 2.5.4. The scaling factor will be further applied to the test data.
   (c) **Train the classifier**
   The classifier will take the training set as input and train. The hyperparameters of the classifier are set to default values.
   (d) **Test classifier**
   This step uses the model to predict the test set.
   (e) **Store partial result**
   This step will store the partial result of the test sample. If every data sample has been used in the testing set, the next step is number 4. If there is still more data to be tested, the loop starts over again from step 3a.
4. **Result**
   This step will compute several measures of performance for a certain chosen classifier, specified in section 2.6.4.

4.6.3 Evaluation

To find the optimal feature combination and algorithm for the optical sensor, the following steps will be investigated:

1. Evaluate the performance of all five classifiers with different feature sets. Additionally, investigate whether a feature scaling is appropriate.
2. The learning process will be integrated with two different feature selection methods to achieve better performance.
3. Some of the top performing classifiers along with the feature set will be further tested on new data samples.
4. Select the two best performers and use a grid search with the training samples to find the best parameters, and test on data samples collected in step 3.
5. The best performing classifier will be further used to evaluate the performance when the robot walks through two different terrains.
6. Lastly, test whether it is appropriate to use currently training samples to predict the other sensors.

Chapter 5

Experiments and results

This chapter is divided into six main sections, where each of them consists of an experiment. The first section will present the result of each classifier with the different sets of features. The second section will present a modified implementation to increase the performance and its result. The third section will select some of the well-performing classifiers with their feature sets to evaluate on a new set of data samples. The fourth section will use a grid search to tune parameters of the two best performing classifiers, and compare to earlier results. The fifth section presents a real-time implementation used to evaluate the performance of the best classifier when the robot walks on two different terrains. The last section shows the possibility of using training samples from the sensor mounted on the front left foot of the robot, to predict the sensor data provided from the front right foot.

5.1 Evaluation of classifier

This section presents the results of each classifier on different feature sets along with an analysis.

5.1.1 Results

Table 5.1 shows the results of each classifier when using the learning approach described in section 4.6.2. The color indicates how high the accuracy is, where green is an accuracy of 90% and over, yellow is 80%-89%, while red is 79% and under. The experiments were done with and without feature scaling.

Feature set	Classifier	Accuracy	
		Not scaled	Scaled
Set one - raw data	Naive Bayes	83%	83%
	Decision tree	94%	96%
	KNN	89%	91%
	Neural network	95%	96%
	SVM	89%	93%
Set two - statistical features in time domain	Naive Bayes	82%	82%
	Decision tree	83%	84%
	KNN	84%	84%
	Neural network	83%	88%
	SVM	76%	85%
Set three - complete frequency domain	Naive Bayes	91%	91%
	Decision tree	83%	82%
	KNN	91%	88%
	Neural network	93%	93%
	SVM	52%	93%
Set four - staticial features in frequency domain	Naive Bayes	81%	81%
	Decision tree	80%	82%
	KNN	88%	84%
	Neural network	85%	88%
	SVM	83%	86%
Set five - set 2,4	Naive Bayes	82%	82%
	Decision tree	79%	80%
	KNN	88%	84%
	Neural network	87%	89%
	SVM	83%	86%

Table 5.1: Results from five different feature sets on five different classifiers when using the approach shown in figure 4.4. The colors represent how high the accuracy is. Green has an accuracy of 90%, yellow is 80%-89%, while red 79% and under.

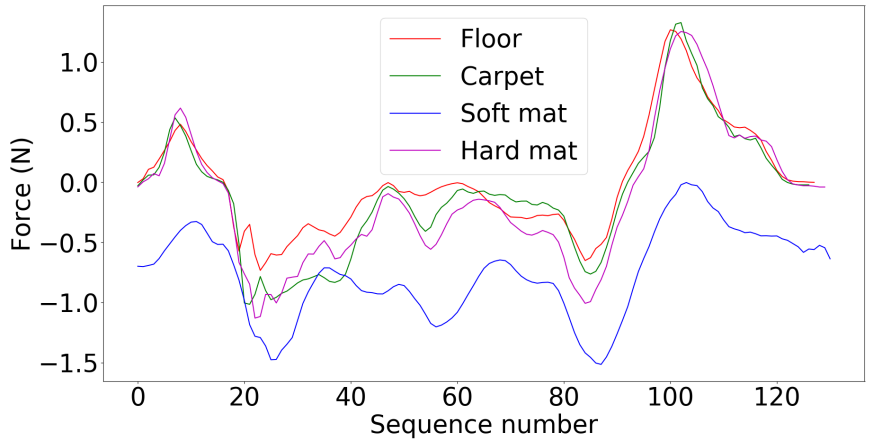
5.1.2 Analysis

All classifiers have an accuracy of at least 70%. Top performing classifiers are the decision tree and neural network with an accuracy of 96%. However, KNN, naive Bayes, and SVM also perform well, with an accuracy of 90%.

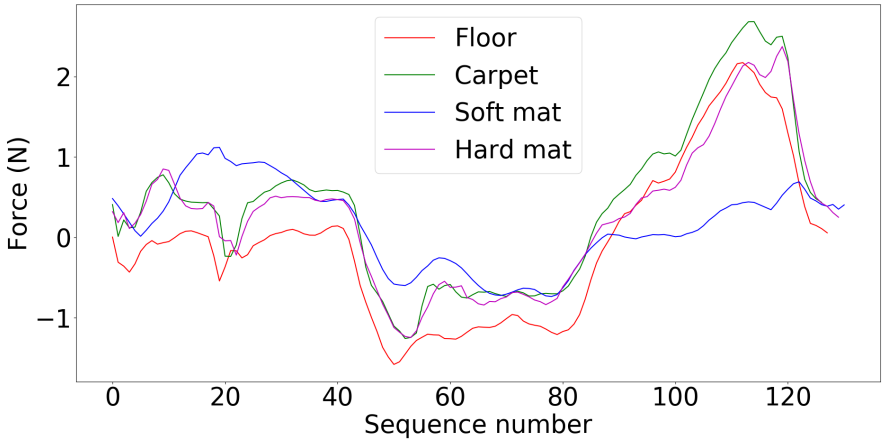
Regarding the feature scaling, out of SVM and KNN which use distances in their computation, SVM had the biggest effect. SVM with the scaled feature on set three went from 52% to 93%. KNN, on the other hand, was slightly affected by scaling, but mostly gave a lower accuracy. The neural network achieved a small improvement, while the decision tree either performed better when scaled or not scaled, dependent on the feature set. Navie Bayes did not undergo any effect by standardized features. Since the feature scaling achieved a big improvement with the SVM and minor improvement on the other classifiers, scaled features will be used in further experiments.

Among the best results are those from either whole raw data sequences or a complete frequency domain where mostly accuracy was over 90%. While statistics features extracted in either the time domain, frequency domain, or both, gave an accuracy of between 80% to 89%.

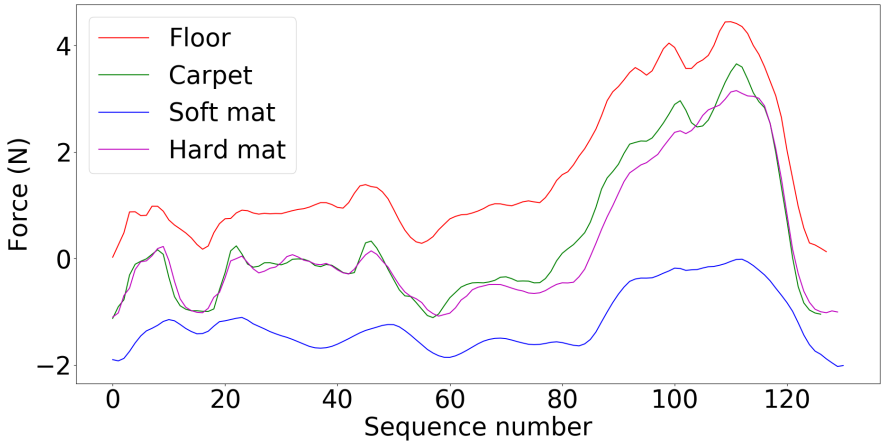
Besides looking at the performance of each classifier, it is also interesting to look at how data provided from the sensor responds to different terrain. In the time domain, which can be seen in figure 5.1, of the terrains, soft mat differs the most. Forces from floor, carpet and hard mat in the x-direction are relatively similar as shown in figure 5.1a. Forces in the y-direction as seen in figure 5.1b are more distinguishable, but the data streams are still quite similar to each other. Meanwhile, forces in the z-direction as seen in figure 5.1c are even more distinguishable. Only carpet and hard mat provide similar forces but differ in the shapes of curves. Floor and carpet, on the other hand, have similar shapes of curves but vary in the forces. In the frequency domain, again, of the terrains, soft mat differs the most. In contrast to similar data from floor, carpet and hard mat in the time domain, in the frequency domain, the floor is more distinguishable as seen in figure 5.2. The carpet and hard mat have an almost equally magnitude in the x-direction. However, there is a slight difference between them in the y- and z-direction shown in figures 5.2b, and 5.2c respectively.



(a) Mean of 10 samples in x-direction

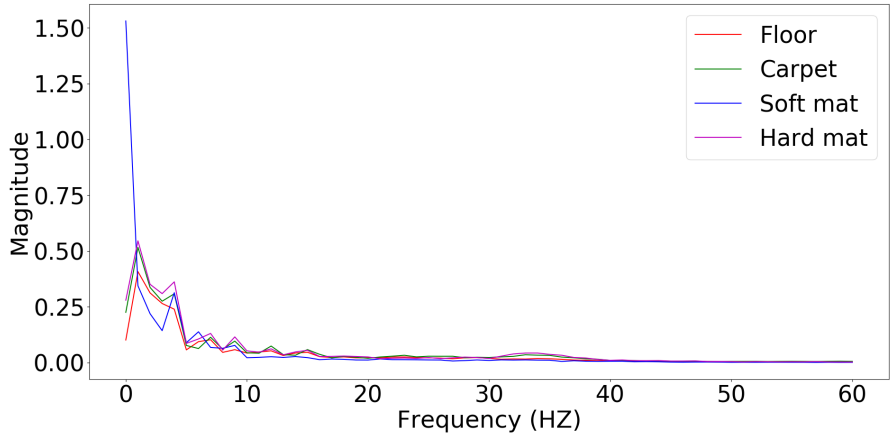


(b) Mean of 10 samples in y-direction

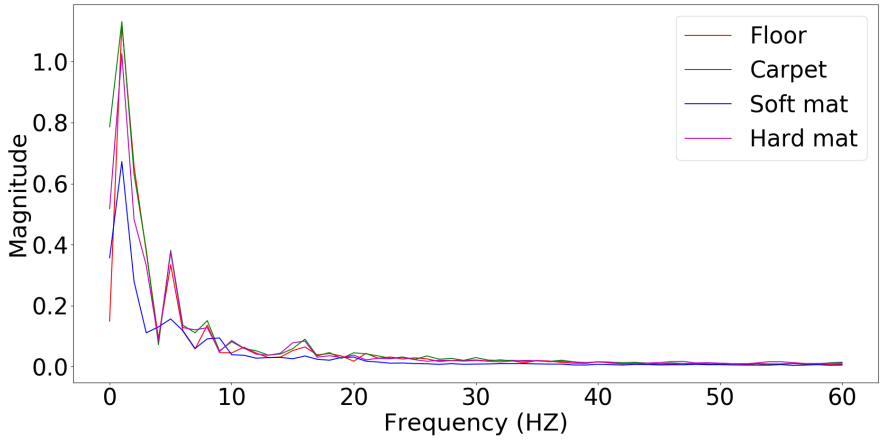


(c) Mean of 10 samples in z-direction

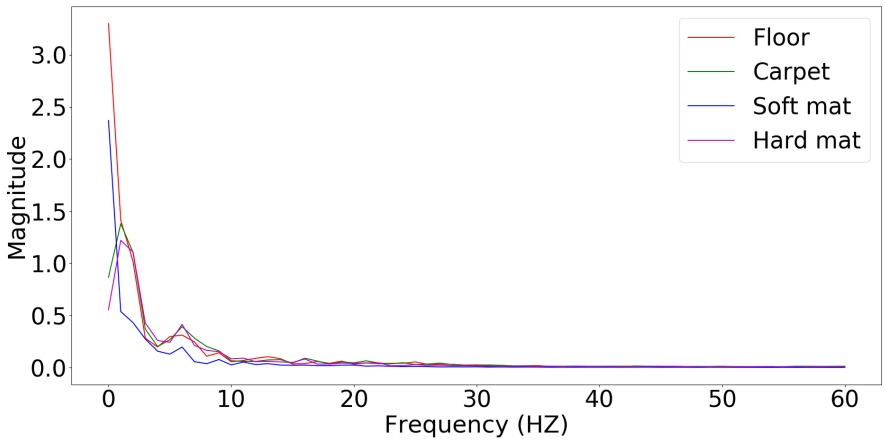
Figure 5.1: The mean of sensor data on 10 steps for each terrain in the time domain: the mean in the x-direction (a), the mean in y-direction (b), and the mean in z-direction (c).



(a) Mean of 10 samples in x-direction



(b) Mean of 10 samples in y-direction



(c) Mean of 10 samples in z-direction

Figure 5.2: The mean of sensor data on 10 steps for each terrain in the frequency domain: the mean in x-direction (a), the mean in y-direction (b), and the mean in z-direction (c).

5.2 Integration of feature selection

Even though there are classifiers which outperform the others, there is still improvement potential. The result achieved was without a feature selection method. The purpose of not using the feature selection was to investigate the performance without removing any features. However, as mentioned in section 2.5.3, selecting features from a set can increase the classifier performance and prevent the curse of dimensionality. Thus, in the following experiment, the feature selection will be integrated with the previous implementation described in section 4.6.2 to see if there is any improvement. The modified implementation is shown in figure 5.3, where the added function is colored as red. The algorithm will select feature on the currently entire data set before the data is partitioned in the cross-validation. The filter and wrapper method will be used, and both are libraries provided by scikit-learn mentioned in section 3.4. The filter method will be removing features with low variance, while the wrapper method is recursive feature elimination and removes features to a desired number.

Removing features with low variance Removing Features with Low Variance (RFLV) is a filter method that removes all features where the variance does not meet some threshold. The default value of the threshold will remove all features that contain the same values, which is unlikely to occur in this experiment. To keep as many features as possible and at the same time removing features with relatively low variance, the threshold will be set to 0.2.

Recursive feature elimination Recursive Feature Elimination (RFE) is a wrapper method that uses a supervised classifier to rank each feature and remove features with a low rank. A chosen estimator will be used to train the set of features, and weigh them. Features with the smallest weights are pruned, and start over to estimate the current remaining set of features. This will be repeated until the desired number of features is reached. In this experiment, the default value will be used, which removes half of the feature set.

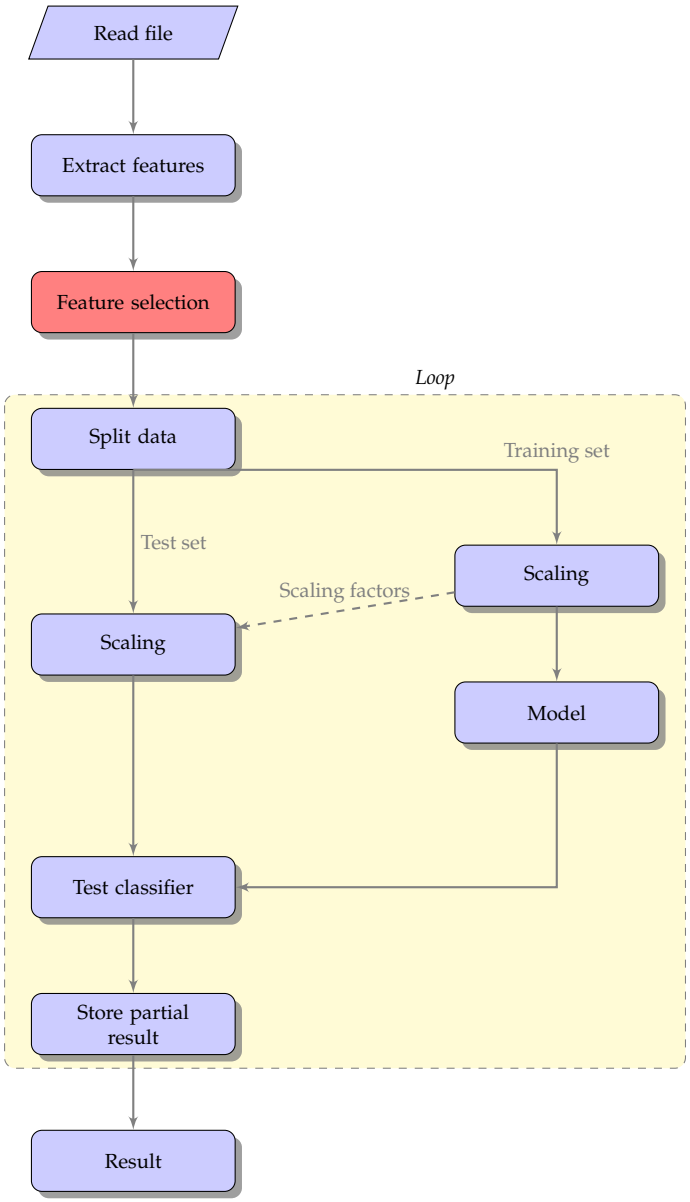


Figure 5.3: Modified evaluation process from section 4.6.2. The new function is feature selection and is colored red.

5.2.1 Results

Table 5.2 shows the new number of features after they were removed for each feature set, and table 5.3 shows the results for when feature selection is applied to the earlier learning approach. The five highest performing models are marked with bold text and will be used in further experiments.

Feature set	**Feature selection**		
	None	**RFLV**	**RFE**
Set one	375	230	187
Set two	21	8	10
Set three	183	2	91
Set four	24	11	12
Set five	45	19	22

Table 5.2: New length of each feature set after applying the feature selection method.

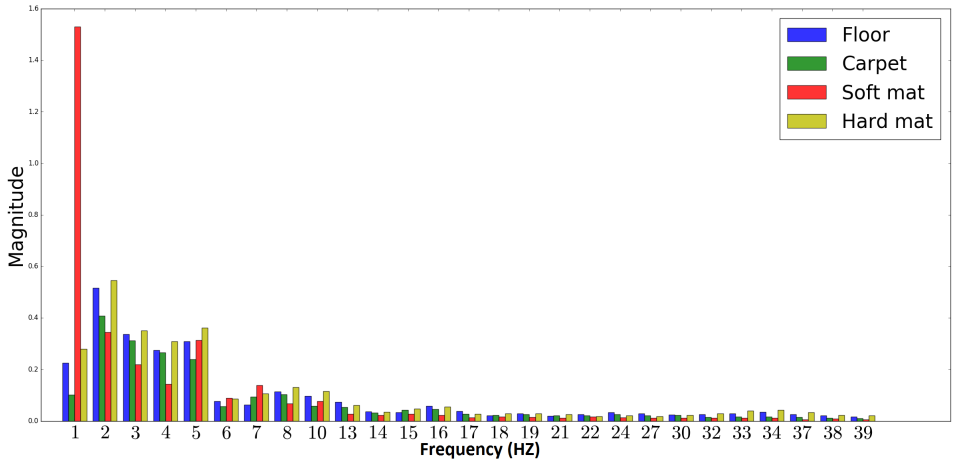
Feature set	Classifier	Accuracy		
		Old result	Filter	Wrapper
Set one - raw data	Navies Bayes	83%	78%	81%
	Decision tree	**96%**	96%	93%
	KNN	91%	95%	93%
	Neural network	**96%**	94%	95%
	SVM	93%	91%	94%
Set two - statistical features in time domain	Naive Bayes	82%	74%	84%
	Decision tree	84%	79%	81%
	KNN	84%	82%	82%
	Neural network	88%	81%	84%
	SVM	85%	80%	83%
Set three - complete frequency domain	Naive Bayes	91%	76%	94%
	Decision tree	82%	76%	81%
	KNN	88%	83%	88%
	Neural network	93%	78%	94%
	SVM	93%	79%	**97%**
Set four - staticial features in frequency domain	Navie Bayes	81%	75%	79%
	Decision tree	82%	82%	81%
	KNN	84%	85%	86%
	Neural network	88%	84%	90%
	SVM	86%	82%	86%
Set five - set 2,4	Naive Bayes	82%	74%	84%
	Decision tree	80%	82%	82%
	KNN	84%	86%	86%
	Neural network	**89%**	81%	85%
	SVM	86%	83%	84%

Table 5.3: Results from five different feature sets on five different classifiers when applying the feature selection method. The colors represent how the new accuracy is compared to the old accuracy. Green represent increased accuracy, yellow represent the same accuracy, and red represent decreased accuracy.

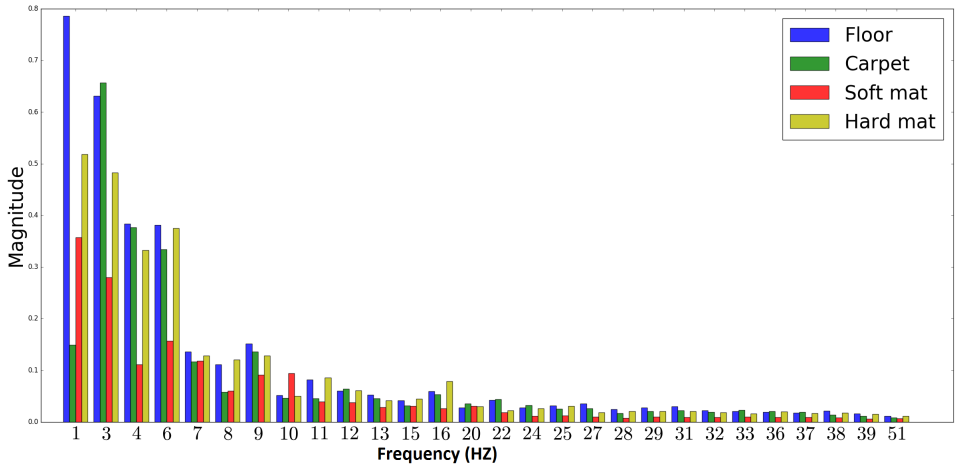
5.2.2 Analysis

Adding the feature selection has decreased performance of most classifiers. The RFLV on the feature set three gave the poorest performance. This is also the set which had the most features removed, as seen in table 5.2. However, the sets that contain statistical features have been removed to nearly equal length for both methods, and they generally have achieved approximately the same results. The RFE selection also has some classifiers that performed less accurately, but the decreased accuracy is less than the filter method. However, SVM with feature set three has improved with an accuracy of 97%.

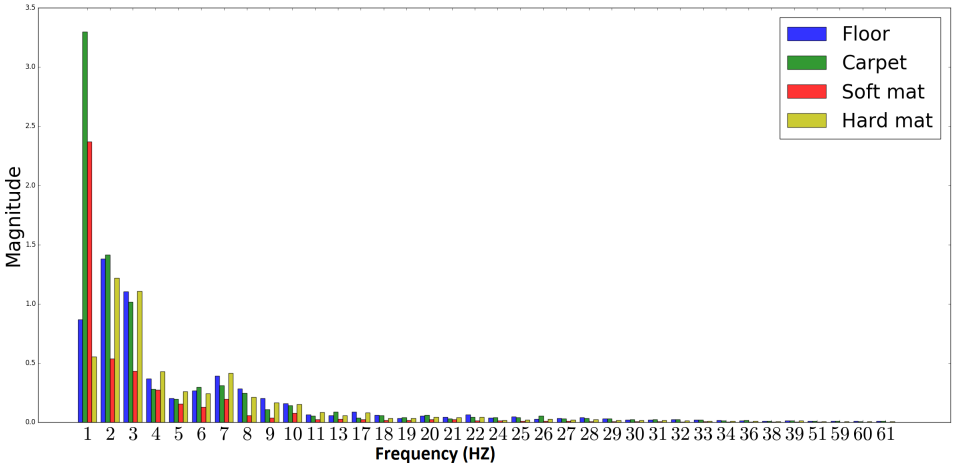
It will be interesting to investigate the features selected from each method. However, this thesis will only present features which gave the highest accuracy, that is the RFE on feature set three. Other selected features can be seen in appendix B. To give a more reliable comparison between each feature on each terrain, the figure 5.4 is the mean of ten steps on each terrain in the frequency domain. The first feature in the x, y, and z-directions is the feature that shows most differences among them. Further behind in the feature set, each feature becomes more equal. The floor, carpet, and hard mat show some differences but tend to overlap each other. Meanwhile, the soft mat differs the most among the terrains.



(a) Features in x-direction



(b) Features in y-direction



(c) Features in z-direction

Figure 5.4: A comparison between each feature selected by RFE on feature set three.

5.3 Cross-validation on unseen data

Note that the k-fold validation in earlier experiments was only using the same single set both for training and testing, which make the data referring to themselves and has the risk of not generalizing the classifier. In that case, results achieved from table 5.1 and table 5.3 only give an estimation on how successful a classifier is, but does not give an insight on how well it predicts on unseen data. One way to avoid this issue is to make the training and testing sets independent of each other, so the predictions will be based on new and unseen data [9]. The following experiment will be using high performing classifiers which are shown in table 5.3, marked with bold text. As there are three shared second-best classifiers, all of them will be included in this experiment. Additionally, the neural network without a selection method with the feature set five achieved a satisfying result which will also be further investigated. The reason is that the four best performing classifiers only use either the entire raw data or frequency domain. Thus, it will be interesting to compare a classifier that uses statistical features as well.

In this experiment, 50 new data samples for each terrain is collected. The cross validation will use the old dataset as a training set, while the newly collected samples will be used as a test set. The cross validation technique in this experiment is still LOOCV, in order to train as many samples as possible. To achieve a reliable performance estimation, it can be done by running the cross-validation multiple times [60]. Thus, the training and test sets will be randomly shuffled before each round of cross validation, and each classifier will be run five times. The table 5.4 shows the classifier and feature set used in this experiment.

Feature set	**Classifier**	**Feature selection**
Set three	SVM	RFE
Set one	Neural network	None
Set five	Neural network	None
Set one	Decision tree	None
Set one	Decision tree	RFLV

Table 5.4: Overview of top performing classifiers with their feature set which will be used to test on unseen data.

5.3.1 Results

An overview of the performance of the five top performed classifier is shown as a box plot in figure 5.5. Again, the top performing is the SVM, and second best is the neural network with the feature set five. These will also be used in further experiments. The other classifiers have heavily decreased their accuracy in their performance.

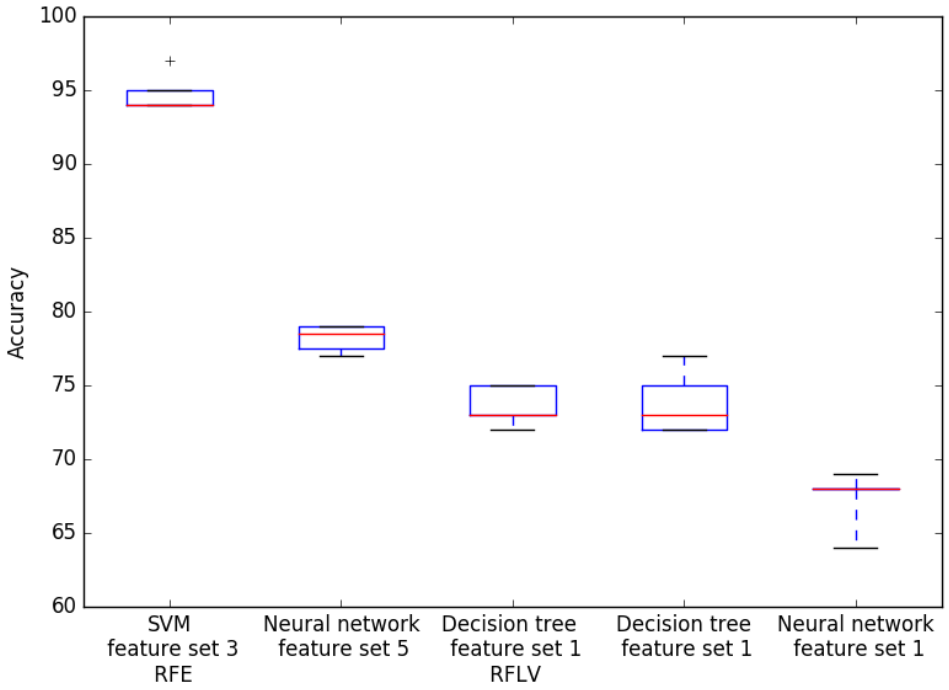


Figure 5.5: Boxplot of the five top performing models on unseen data.

5.3.2 Analysis

This section will present a more detailed description of each classifiers performance with statistics metrics mentioned in section 2.6.4. The following paragraphs are organized by average accuracy in descending order.

SVM - feature set three - RFE

This is the best performing classifier, and table 5.5 shows the performance of each run. The average performance of predicting unseen data has an accuracy of 94.8%, and compared to the old result from table 5.3 it has decreased by 2.2%. The classifier easily predicts floor, carpet, and soft mat, while it has some difficulties predicting hard mat. When the robot is on the hard mat, it has an average precision of 84%, and it tends to predict it as carpet. Because the hard mat tends to be predicted as other terrains, the f-score for floor and carpet is decreased a little as seen in table 5.5c. The hard mat and carpet have relative equal f-scores. However, the carpet got a lower recall, but higher precision, while the hard mat got a higher recall, but lower precision.

Terrain	**Run**				
	1	**2**	**3**	**4**	**5**
Floor	100%	100%	100%	100%	100%
Carpet	94%	96%	96%	96%	98%
Soft mat	100%	100%	100%	100%	100%
Hard mat	82%	94%	82%	80%	82%

(a) Precision - SVM - feature set three - RFE.

Terrain	**Run**				
	1	**2**	**3**	**4**	**5**
Floor	90.9%	92.6%	92.6%	92.6%	87.5%
Carpet	87%	98%	87.2%	85.7%	98%
Soft mat	100%	100%	100%	100%	100%
Hard mat	100%	100%	100%	100%	100%

(b) Recall - SVM - feature set three - RFE.

Terrain	**Run**				
	1	**2**	**3**	**4**	**5**
Floor	95.2%	96.2%	96.2%	96.2%	98%
Carpet	90.4%	97%	91.4%	90.6%	97%
Soft mat	100%	100%	100%	100%	100%
Hard mat	90.1%	96.9%	90.1%	88%	90.1%

(c) F-score - SVM - feature set three - RFE.

Run	Accuracy	Confusion matrix			
		Floor	**Carpet**	**Soft mat**	**Hard mat**
1	94%	50	0	0	0
		3	47	0	0
		0	0	50	0
		2	7	0	41
2	97%	50	0	0	0
		2	48	0	0
		0	0	50	0
		2	1	0	47
3	94%	50	0	0	0
		2	48	0	0
		0	0	50	0
		2	7	0	41
4	94%	50	0	0	0
		2	48	0	0
		0	0	50	0
		2	8	0	40
5	95%	50	0	0	0
		1	49	0	0
		0	0	50	0
		2	7	0	41

(d) Accuracy and confusion matrix for the SVM with RFE using feature set three after five runs. Regarding the confusion matrix, the rows show the actual terrains and the columns show the predicted terrains.

Table 5.5: Results of the SVM using feature set three with RFE after five runs. The results consist of: precision (a), recall (b), f-score (c), and accuracy and confusion matrix (d).

Neural network - feature set five

This is the second best performing classifier, and in table 5.6 shows the performance of each run. The average performance of predicting unseen data has an accuracy of 78.2%, and compared to the old result from table 5.3 it has decreased by 10.8%. The classifier has difficulties predicting the hard mat and tends to predict it as carpet. Soft mat is the easiest to identify, but there is a minor amount of error in the recall when the robot is on the floor. The floor has the second highest accuracy with over 80%. It has a precision of at least 88%, and the recall is under 80%, because there is shown misclassification of floor when the robot is on hard mat and floor. The carpet has a precision of at least 78%, but the recall is under 63.5%, while the hard mat has a higher recall of at least 71%, and lower precision of approximately 41%.

Terrain	**Run**				
	1	**2**	**3**	**4**	**5**
Floor	96%	92%	88%	90%	94%
Carpet	80%	78%	84%	78%	80%
Soft mat	100%	100%	100%	100%	100%
Hard mat	40%	40%	42%	40%	42%

(a) Precision - neural network - feature set five.

Terrain	**Run**				
	1	**2**	**3**	**4**	**5**
Floor	78.7%	79.3%	80%	76.3%	78.3%
Carpet	63.5%	60.9%	63.6%	60.3%	63.5%
Soft mat	100%	100%	98%	98%	100%
Hard mat	76.9%	71.4%	75%	76.9%	77.8%

(b) Recall - neural network - feature set five.

Terrain	**Run**				
	1	**2**	**3**	**4**	**5**
Floor	86.5%	85.2%	83.8%	82.6%	85.4%
Carpet	70.8%	68.4%	72.4%	68%	70.8%
Soft mat	100%	100%	99%	99%	100%
Hard mat	52.6%	51.3%	53.8%	52.6%	54.6%

(c) F-score - neural network - feature set five.

Run	Accuracy	Confusion matrix			
		Floor	**Carpet**	**Soft mat**	**Hard mat**
1	79%	48	2	0	0
		4	40	0	6
		0	0	50	0
		9	21	0	20
2	77.5%	46	4	0	0
		3	39	0	8
		0	0	50	0
		9	21	0	20
3	78.5%	44	5	1	0
		1	42	0	7
		0	0	50	0
		10	19	0	21
4	77%	45	4	1	0
		5	39	0	6
		0	0	50	0
		9	21	0	20
5	79%	47	3	0	0
		4	40	0	6
		0	0	50	0
		9	20	0	21

(d) Accuracy and confusion matrix for the neural network using feature set five after five runs. Regarding the confusion matrix, the rows show the actual terrains and the columns show the predicted terrains.

Table 5.6: Results of the neural network using feature set five after five runs. The results consist of: precision (a), recall (b), f-score (c), and accuracy and confusion matrix (d).

Decision Tree - feature set one - RFLV

This is the third best performing classifier, and table 5.7 shows the performance of each run. The average performance of predicting unseen data has an accuracy of at least 73.6%, and compared to the old result from table 5.3 it has decreased by 22.4%. Again, it has problems predicting the hard mat, which it tends to predict as carpet. In contrast to the three best performing models, this model consists of more misclassification in every class. However, the floor and soft mat still have among the highest accuracies with f-score of at least 88%. The carpet has a high precision, but low recall, and vice versa with the hard mat.

Terrain	**Run**				
	1	**2**	**3**	**4**	**5**
Floor	94%	94%	88%	90%	90%
Carpet	78%	82%	82%	82%	80%
Soft mat	88%	94%	94%	90%	94%
Hard mat	32%	28%	34%	30%	24%

(a) Precision - decision tree - feature set one - RFLV.

Terrain	**Run**				
	1	**2**	**3**	**4**	**5**
Floor	79.7%	77%	78.6%	76.3%	78.9%
Carpet	56.5%	55.4%	58.6%	54.7%	53.3%
Soft mat	91.7%	100%	94%	100%	94%
Hard mat	66.7%	77.8%	70.8%	65.2%	66.7%

(b) Recall - decision tree - feature set one - RFLV.

Terrain	**Run**				
	1	**2**	**3**	**4**	**5**
Floor	86.3%	84.7%	83%	82.6%	84.1%
Carpet	65.5%	66.1%	68.4%	65.6%	64%
Soft mat	89.8%	96.9%	94%	94.7%	94%
Hard mat	43.3%	41.2%	45.9%	41.1%	35.3%

(c) F-score - decision tree - feature set one - RFLV.

Run	Accuracy	Confusion matrix			
		Floor	**Carpet**	**Soft mat**	**Hard mat**
1	73%	47	1	2	0
		5	39	1	5
		0	3	44	3
		7	26	1	16
2	75%	47	3	0	0
		6	41	0	3
		0	2	47	1
		8	28	0	14
3	75%	44	3	3	0
		5	41	0	4
		0	0	47	3
		7	26	0	17
4	73%	45	4	0	3
		6	41	0	3
		0	3	45	2
		8	27	0	15
5	72%	45	2	3	0
		6	40	0	4
		0	1	47	2
		6	32	0	12

(d) Accuracy and confusion matrix for the decision tree using feature set one with RFLV after five runs. Regarding the confusion matrix, the rows show the actual terrains and the columns show the predicted terrains.

Table 5.7: Results of decision tree using feature set one with RFLV after five runs. The results consist of: precision (a), recall (b), f-score (c), and accuracy and confusion matrix (d).

Decision Tree - feature set one

This is the fourth best performing classifier, and table 5.8 shows the performance of each run. The average performance of predicting unseen data has an accuracy of at least 73.8%, and compared to the old result from table 5.3 it has decreased by 22.2%. It still has problems predicting the hard mat, which it tends to predict it as carpet. It also shows a similar misclassification as the third best performing classifier when it is on the soft mat. However, soft mat still has the highest precision and recall of at least 92% and 87% respectively. The carpet varies in precision between 76% - 82%, while the recall is under 50%. The hard mat achieved a precision of between 28% - 36% and a big variation in the recall with a range of 66.7% to 78.3%.

Terrain	**Run**				
	1	**2**	**3**	**4**	**5**
Floor	86%	90%	88%	88%	96%
Carpet	82%	76%	82%	82%	78%
Soft mat	92%	94%	94%	92%	98%
Hard mat	28%	34%	34%	28%	36%

(a) Precision - decision tree - feature set one.

Terrain	**Run**				
	1	**2**	**3**	**4**	**5**
Floor	78.1%	77.6%	78.6%	75.9%	78.7%
Carpet	56.9%	57.6%	58.6%	58.6%	65%
Soft mat	88.5%	92.2%	94%	90.2%	87.5%
Hard mat	66.7%	68%	70.8%	66.7%	78.3%

(b) Recall - decision tree - feature set one.

Terrain	**Run**				
	1	**2**	**3**	**4**	**5**
Floor	81.9%	83.3%	83%	81.5%	86.5%
Carpet	67.2%	65.5%	68.4%	68.4%	70.9%
Soft mat	90.2%	93.1%	94%	92%	92.5%
Hard mat	39.4%	45.3%	45.9%	39.4%	49.3%

(c) F-score - decision tree - feature set one.

Run	Accuracy	Confusion matrix			
		Floor	**Carpet**	**Soft mat**	**Hard mat**
1	72%	43	5	2	0
		4	41	2	3
		0	0	46	4
		8	26	2	14
2	73%	45	4	1	0
		5	38	2	5
		0	0	47	3
		8	24	1	17
3	75%	44	3	3	0
		5	41	0	4
		0	0	47	3
		7	26	0	17
4	72%	44	2	4	0
		6	41	0	3
		0	0	46	4
		8	27	1	14
5	77%	48	0	2	0
		5	39	2	4
		0	0	49	1
		8	21	3	18

(d) Accuracy and confusion matrix for the decision tree using feature set one after five runs. Regarding the confusion matrix, the rows show the actual terrains and the columns show the predicted terrains.

Table 5.8: Results of the decision tree on feature set one after five runs. The results consist of: precision (a), recall (b), f-score (c), and accuracy and confusion matrix (d).

Neural network - feature set one

This is the fifth best performing classifier and table 5.9 shows the performance of each run. The average performance of predicting unseen data has an accuracy of 67.4%, and compared to the old result from table 5.3 it has decreased by 28.6%. In contrast with the other models, this has the poorest result of predicting the floor, which it tends to predict as carpet. Again, the hard mat is the most difficult to predict and has the highest probability of being predicted as carpet, but it is also likely to be predicted as floor as well. The carpet has the second highest precision with a least 96%, but a low recall of under 48.5%. The floor has a precision of over 40% and recall of between 64.5% - 75%. The hard mat achieved the lowest precision with under 24%, but has a high recall with over 83.3%.

Terrain	**Run**				
	1	**2**	**3**	**4**	**5**
Floor	50%	52%	40%	54%	52%
Carpet	98%	96%	96%	98%	98%
Soft mat	100%	100%	100%	100%	100%
Hard mat	22%	22%	20%	18%	24%

(a) Precision - neural network - feature set one.

Terrain	**Run**				
	1	**2**	**3**	**4**	**5**
Floor	67.6%	72.2%	64.5%	75%	72.2%
Carpet	48%	47.1%	44.9%	47%	48.5%
Soft mat	100%	100%	100%	100%	100%
Hard mat	100%	91.7%	83.3%	90%	92.3%

(b) Recall - neural network - feature set one.

Terrain	**Run**				
	1	**2**	**3**	**4**	**5**
Floor	57.5%	60.5%	49.4%	62.8%	60.5%
Carpet	64.4%	63.2%	61.2%	63.5%	64.9%
Soft mat	100%	100%	100%	100%	100%
Hard mat	36.1%	35.5%	32.3%	30%	38.1%

(c) F-score - neural network - feature set one.

Run	Accuracy	Confusion matrix			
		Floor	**Carpet**	**Soft mat**	**Hard mat**
1	68%	25	25	0	0
		1	49	0	0
		0	0	50	0
		11	28	0	11
2	68%	26	24	0	0
		1	48	0	1
		0	0	50	0
		9	30	0	11
3	64%	20	30	0	0
		0	48	0	2
		0	0	50	0
		11	29	0	10
4	68%	27	23	0	0
		0	49	0	1
		0	0	50	0
		9	32	0	9
5	69%	26	24	0	0
		0	49	0	1
		0	0	50	0
		10	28	0	12

(d) Accuracy and confusion matrix for the neural network using feature set one after five runs. Regarding confusion matrix, the row is the actual terrains and the columns is the predicted terrains.

Table 5.9: Results of the neural network using feature set one after trials runs. The results consist of: precision (a), recall (b), f-score (c), and accuracy and confusion matrix (d).

5.3.3 Summary

In this experiment, five classifiers with different feature sets were investigated on new data samples. Compared to the old results from table 5.3, every classifier has decreased their accuracy. However, the SVM is still the best performing classifier with an average accuracy of 94.8%, while the others are decreased to under 80%.

Regarding different sets of features, it can be seen that using decimated raw data from the time domain gave the poorest performance. The neural network only achieved an average accuracy of 67.4%. It is also worth noting that the decision tree with and without the feature selection method did not result in any notable differences. Feature set five which uses statistics features from time and frequency domains gave feasible results, and did not decrease as much as the models using feature set one.

A common similarity between most of the models is that the soft mat is the easiest terrain to identify, although some of the models have shown minor confusion. Every model has difficulty predicting hard mat, and tend to predict it as carpet. This misclassification also leads to a lower recall and f-score for the carpet. But most of the models are able to have a high precision of prediction on floor and carpet. Even though the hard mat is often mistaken for carpet, the other three terrains are less often mistaken for hard mat.

5.4 Parameter tuning

Previously, the evaluation of classifiers was based on default hyperparameters. However, tuning parameters might increase the performance of the classifier as mentioned in section 2.4.2. The reason for not using the parameter tuning in earlier experiments is due to time-consuming and difficult to find the correct settings, and default values often give an adequate performance. In this experiment, an exhaustive grid search will be used to find optimal parameters on two top performing classifiers as shown in the table 5.10, and further tested on unseen data.

Feature set	**Classifier**	**Feature selection**
Set three	SVM	RFE
Set five	Neural network	None

Table 5.10: Two best performing models used to find the best parameters.

Neural network A crucial part of the neural network is the choice of the number of hidden nodes, and hidden layers. These choices are fundamental to achieving a promising result of the algorithm. It is common to use two hidden layers, but there is no theory on how to choose the number of neurons [49]. To find the best number of neurons, one has to test many different values and chose the one that gives the best results. An exhaustive grid search is therefore appropriate to test every possible combination of neurons in one and two hidden layers, as seen in table 5.11. The parameters obtained from the grid search will be further tested with three different representations of terrains.

The current representation of the terrain is that each terrain is assigned to four nodes. Hence, predictions of current terrain will be based on the highest score of these nodes. Another representation of terrain is binary numbers. This representation, instead of selecting the highest score from one node, will be looking at a pattern of firing and non-firing neurons. Thus, two other alternatives representing classes with binary numbers are shown in table 5.12, which will also be tested with the best parameters found with the grid search. The first representation is by two binary numbers as shown in table 5.12a. This representation will only have two output nodes, and it will always predict a terrain even if it appears misfiring. Representation of each terrain with four binary numbers, on the other hand, will give the output as unknown terrain when at least two neurons are misfiring.

Parameter	Hidden layer	Values
Neurons	1st	1,...,100
Neurons	2nd	0,...,100

Table 5.11: Parameter and values used in grid search on the neural network.

Binary	Class
00	Floor
01	Carpet
10	Soft mat
11	Hard mat

(a) Representation of each terrain with two binary numbers.

Binary	Class
0001	Floor
0010	Carpet
0100	Soft mat
1000	Hard mat

(b) Representation of each terrain with four binary numbers.

Table 5.12: Representation of each terrain with two binary numbers (a), and four binary numbers (b).

SVM Hyperparameters that will be searched for SVM is shown in table 5.13. The kernel can be seen as a similarity function. Each kernel has a different way to compute their similarities. Specifications of the computation for each kernel is given in table 2.1. When one does not want a perfect separation between all data, one can allow some misclassification by tuning the parameter C. The parameter C determines how much one wants to avoid misclassifying each training sample. A large C value gives a better separation of the data, but might not be able to generalize data. Meanwhile, a small C value allows more misclassification and might contain more errors.

Parameter	Values
Kernel	RBF, linear, polynomial
C	0.1,0.2,...,0.9 and 1,2,...,1000

Table 5.13: Parameter and values used in grid search on the SVM.

5.4.1 Results

Table 5.14 shows the result after a tuned parameter. The best result is marked with bold text and will be used in the last two experiments.

Classifier	Feature set	Feature selection	Output type	Number of output	Old result	New result
Neural network	Set 5	None	String	1	78.2%	77.6%
			Binary	2	NaN	78.9%
			Binary	4	NaN	74.2%
SVM	**Set 3**	**RFE**	**String**	**1**	**94.8%**	**93.6%**

Table 5.14: Results after tuning parameters on the neural network and SVM.

5.4.2 Analysis

Tuning parameters did not have a effect on the performance. Mostly it slightly decreased, but the neural network with two outputs had a minor increase. Again, the best performing classifier is SVM with feature set three.

Regarding the binary representation of terrains, there is shown a decrease of accuracy when having a representation of the terrain with four binary numbers. There is a minor increase of accuracy when the output consists of two binary outputs.

5.5 Classification in real-time

A more realistic practical scenario is to let the robot traverse through different terrains. Thus, the earlier approach will be modified for this experiment. In this experiment, the robot will be walking on two different terrains. There are in total five different setups.

1. Floor to carpet
2. Hard mat to floor
3. Hard mat to soft mat
4. Soft mat to hard mat
5. Soft mat to carpet

5.5.1 Real-time implementation

The real-time implementation can be seen in figure 5.6, which is a modification of the earlier approach. In this implementation, a ROS service node mentioned in section 3.3.2, is created after modeled a classifier. The training is the same set as used in earlier experiments with 50 samples for each terrain. The service will be waiting for a request from another node, which is the segment data node. The segment data gathers a data stream from the optoForce sensor. The method will collect interesting data as described in section 4.3.3. After a step is extracted from the sensor, it will then be passed to the service. The service will receive the data and start to preprocess the data. It will extract features, select features, and scale according to the training samples. The last step will classify, and output the terrain label of the preprocessed data. Python provides a probability function on SVM, which is used to estimate the probability of each terrain when predicting. It will be interesting to look how high probability each terrain is to be predicted. However, the documentation is aware that the probability involves Platt scaling, which may be inconsistent with the scores. For example, the highest probability is not necessarily the highest score achieved without the probability function. Thus, one has to be aware of this when analyzing the results. After given a result, the program will be waiting for a new partition of sensor data from segment data algorithm.

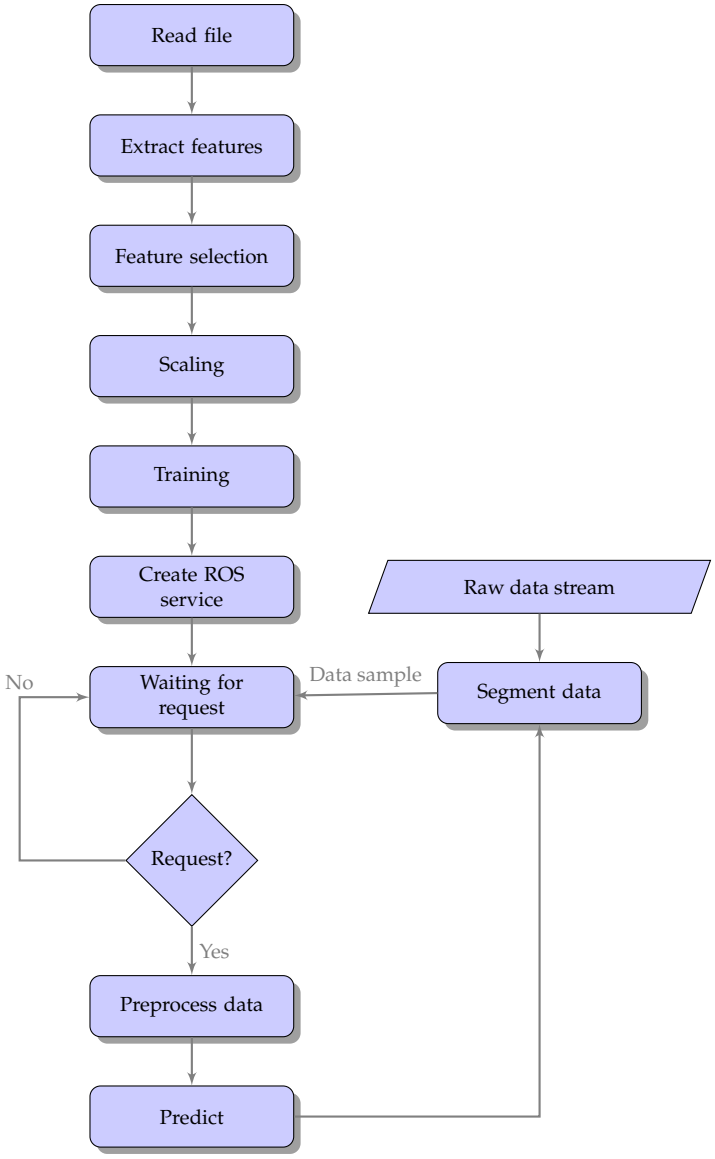


Figure 5.6: Real-time implementation.

5.5.2 Result and analysis

The following paragraphs present the results along with an analysis of each experiment, when the robot traverses through two different terrains in this order:

1. Floor to carpet
2. Hard mat to floor
3. Hard mat to soft mat
4. Soft mat to hard mat
5. Soft mat to carpet

Floor to carpet

The data stream when the robot traverses from floor to carpet is shown in figure 5.7, where the vertical line is the transition between the terrains, and table 5.15 shows probabilities for each terrain on each step. The overall accuracy of predicting correctly is 88.9%. However, how certain the classifier is to predict correctly is under 78%. The fifth step is the transition and as shown, there is some confusion between floor and carpet. It is also worth noticing that at the fourth step the carpet has almost the same probability as the floor with 56.4% and 42.5% respectively. The soft mat and hard mat have small probabilities in every step and is less likely to be predicted.

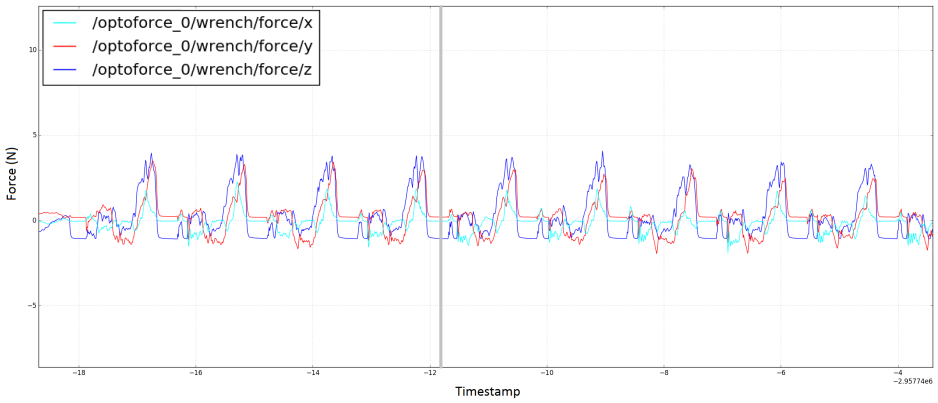


Figure 5.7: Data stream of the robot traversing from floor to carpet. The thick gray horizontal line is the boundary between the two terrains.

Step	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Floor	0.782	0.680	0.747	0.561	0.564	0.282	0.206	0.252	0.282
Carpet	0.105	0.289	0.162	0.425	0.353	0.704	0.730	0.573	0.599
Soft mat	0.008	0.009	0.010	0.008	0.010	0.005	0.009	0.009	0.015
Hard mat	0.104	0.022	0.081	0.007	0.073	0.010	0.055	0.166	0.104

Table 5.15: Estimated probability of each terrain per step from floor to carpet. Values are marked green to represent correct predictions. For incorrect predictions, the actual value is marked yellow while the predicted value is marked red.

Hard mat to floor

The data stream when the robot traverses from the hard mat to floor is shown in figure 5.8, where the vertical line is the transition between the terrains, and table 5.16 shows probabilities for each terrain on each step. The overall accuracy of predicting correctly is 50%. The first two steps have shown a high accuracy of predicting correctly with over 90%. The third step, on the other hand, which is the step before the transition, has a lower probability of the hard mat, with a similar probability as the carpet. It is also worth noticing that the force in the y-direction at this step differs from earlier steps. The fourth step is the transition, and is also where the model misclassifies the new terrain. It rather predicts it as carpet than floor.

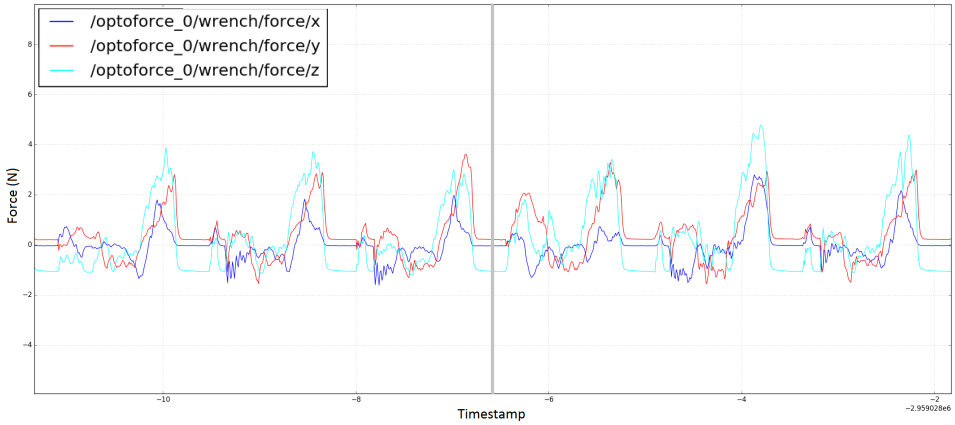


Figure 5.8: Data stream of the robot traversing from hard mat to floor. The thick gray horizontal line is the boundary between the two terrains.

Step	1	2	3	4	5	6
Floor	0.006	0.008	0.172	0.263	0.141	0.078
Carpet	0.008	0.073	0.375	0.641	0.615	0.822
Soft mat	0.006	0.004	0.008	0.023	0.019	0.014
Hard mat	0.980	0.915	0.444	0.073	0.225	0.085

Table 5.16: Estimated probability of each terrain per step walking from hard mat to floor. Values are marked green to represent correct predictions. For incorrect predictions, the actual value is marked yellow while the predicted value is marked red.

Hard mat to soft mat

The data stream when the robot traverses from hard mat to soft mat is shown in figure 5.9 and table 5.17 shows probabilities for each terrain on each step. The overall accuracy of predicting correctly is 100%, while the average of choosing the correct terrain is 72.7%. The forces in the y- and z-directions are much higher when walking on the hard mat than the soft mat. There is some uncertainty on the first, third and sixth step, where the probability of predicting correctly is less than 54%.

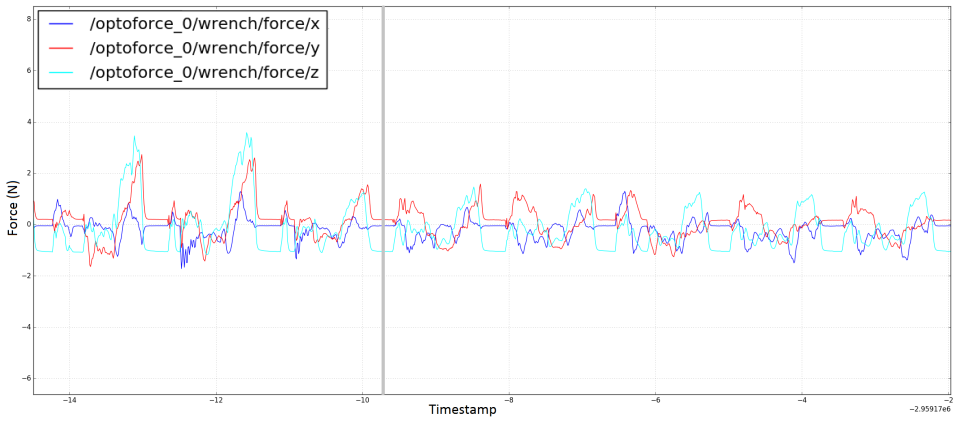


Figure 5.9: Data stream of the robot traversing from hard mat to soft mat. The thick gray horizontal line is the boundary between the two terrains.

Step	**1**	**2**	**3**	**4**	**5**	**6**	7	8
Floor	0.329	0.015	0.081	0.059	0.022	0.115	0.059	0.048
Carpet	0.105	0.045	0.105	0.041	0.015	0.095	0.037	0.033
Soft mat	0.023	0.006	0.284	0.759	0.923	0.477	0.816	0.834
Hard mat	0.542	0.935	0.529	0.142	0.040	0.313	0.088	0.085

Table 5.17: Estimated probability of each terrain per step walking from hard mat to soft mat. Values are marked green to represent correct predictions. For incorrect predictions, the actual value is marked yellow while the predicted value is marked red.

Soft mat to hard mat

The data stream when the robot traverses from the soft mat to hard mat is shown in figure 5.10, and table 5.18 shows probabilities for each terrain on each step. The overall accuracy of predicting correctly is 100%, while the average of choosing the right terrain is 79%. Again, the forces in the y-, and z-directions are much higher when walking on the hard mat than the soft mat. There is some uncertainty on the first and fourth steps, where the probability of predicting correctly is less than 53.5%.

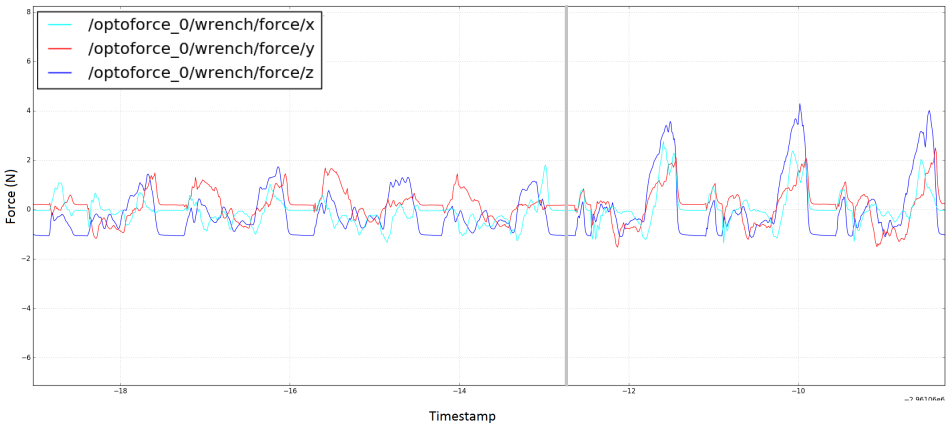


Figure 5.10: Data stream of the robot traversing from soft mat to hard mat. The thick gray horizontal line is the boundary between the two terrains.

Step	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Floor	0.099	0.019	0.018	0.138	0.021	0.008	0.038
Carpet	0.065	0.023	0.017	0.134	0.050	0.005	0.022
Soft mat	0.535	0.871	0.911	0.384	0.009	0.009	0.011
Hard mat	0.300	0.086	0.054	0.344	0.921	0.978	0.930

Table 5.18: Estimated probability of each terrain per step walking from soft mat to hard mat. Values are marked green to represent correct predictions. For incorrect predictions, the actual value is marked yellow while the predicted value is marked red.

Soft mat to carpet

The data stream when the robot traverses from the soft mat to carpet is shown in figure 5.11, and table 5.19 shows probabilities for each terrain on each step. The overall accuracy of predicting correctly is 87.5%. Again, the forces in the y- and z-directions are much higher when walking on carpet than the soft mat. There is some uncertainty on the first and fourth steps, where the probability of predicting correctly is less than 53.5%. The fifth step is when the robot is on the new terrain, and has a probability of predicting it correct of 66.7%. Regarding step six, the classifier makes a wrong prediction between carpet and hard mat with a probability of 93.3%.

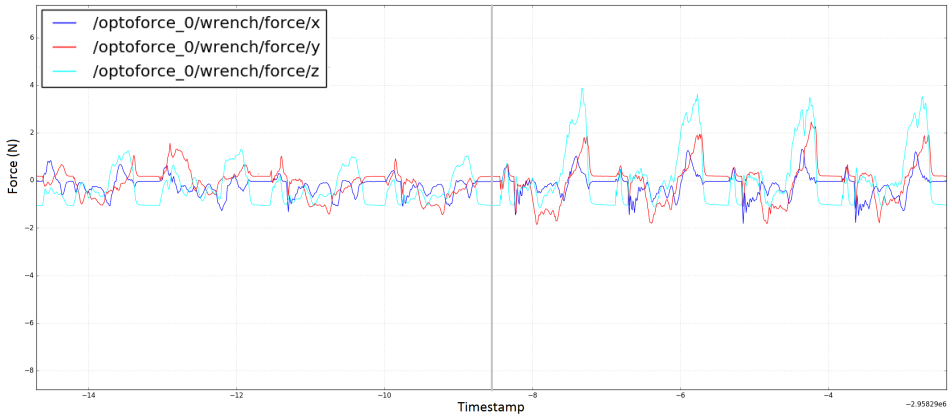


Figure 5.11: Data stream of the robot traversing from soft mat to carpet. The thick gray horizontal line is the boundary between the two terrains.

Step	**1**	**2**	**3**	**4**	**5**	**6**	7	8
Floor	0.021	0.022	0.068	0.031	0.100	0.020	0.016	0.018
Carpet	0.017	0.019	0.049	0.023	0.667	0.041	0.942	0.904
Soft mat	0.909	0.914	0.762	0.891	0.017	0.007	0.006	0.006
Hard mat	0.054	0.045	0.121	0.055	0.216	0.933	0.037	0.072

Table 5.19: Estimated probability of each terrain per step walking from soft mat to carpet. Values are marked green to represent correct predictions. For incorrect predictions, the actual value is marked yellow while the predicted value is marked red.

5.5.3 Summary

A more realistic practical scenario is investigated. Five different setups where the robot walks between two different terrains achieved an overall accuracy of 85.3%. It is worth noting that the robot often did not have a straight walk, which is also some of the reason why the step in each experiment varies.

A common characteristic is that the probability decreases, particularly on the steps between the transition of terrain. Another characteristic is that the first and last step also tend to have a lower accuracy. Regarding the data provided from the sensor, the soft mat has a smaller force in the y- and z-directions than the other terrains. There is also no misclassification of the soft mat. However, there is some misclassification between floor and carpet, and hard mat and carpet.

5.6 Prediction on other sensor

All previous experiments have only used data provided from the sensor mounted on the front left foot of the robot. Thus, it will be interesting to investigate the possibility of using the current training set to predict the data provided from a sensor mounted on the front right foot. The reason for testing the front right foot is that it has a more similar leg motion to the front left foot. The back foot, on the other hand, has a different leg motion which provides different data in each direction. In this following experiment, 30 data samples from the right foot for each terrain will be stored into a file and used to as test samples. The model used is still SVM on the feature set five.

5.6.1 Results

Table 5.20 shows the results of using the training set provided from left front foot to predict data from right front foot.

Terrain	**Floor**	**Carpet**	**Soft mat**	**Hard mat**
Floor	29	1	0	0
Carpet	3	27	0	0
Soft mat	0	0	30	0
Hard mat	2	13	0	16
Precision	96.7%	90%	100%	53.3%
Recall	85.3%	65.9%	100%	100%
F-score	90.6%	76.1%	100%	69.5%
Accuracy			84.3%	

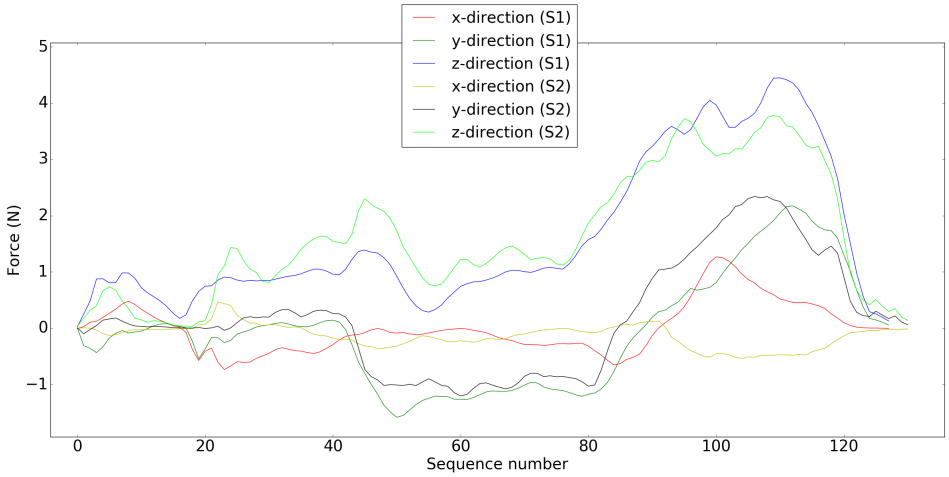
Table 5.20: Results of predicting sensor data provided from the front right foot.

5.6.2 Analysis

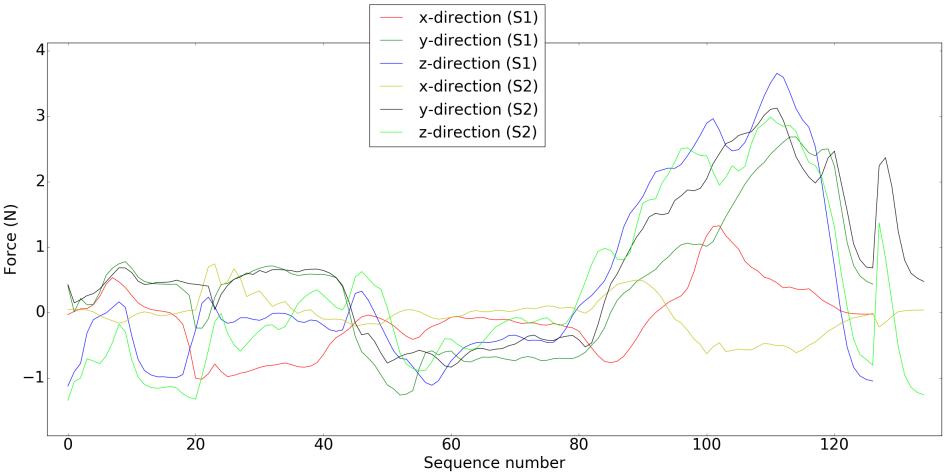
The overall accuracy of predicting data from the sensor mounted on the right foot is 84.3%. It has a precision of over 90% when predicting floor, carpet, and soft mat. Hard mat, on the other hand, achieved only a precision of 53.3%. The low precision is due to its tendency to be predicted as carpet. Results achieved in this experiment are similar to earlier experiments as in section 5.5.2, when testing the classifier on new data samples.

Further analysis will be looking at the data provided from each sensor on each terrain. Thus a mean of 10 steps on each terrain with each sensor is shown in figure 5.12. The force in the x-direction between the two sensors differs from each other in every terrain. Meanwhile, the forces in the

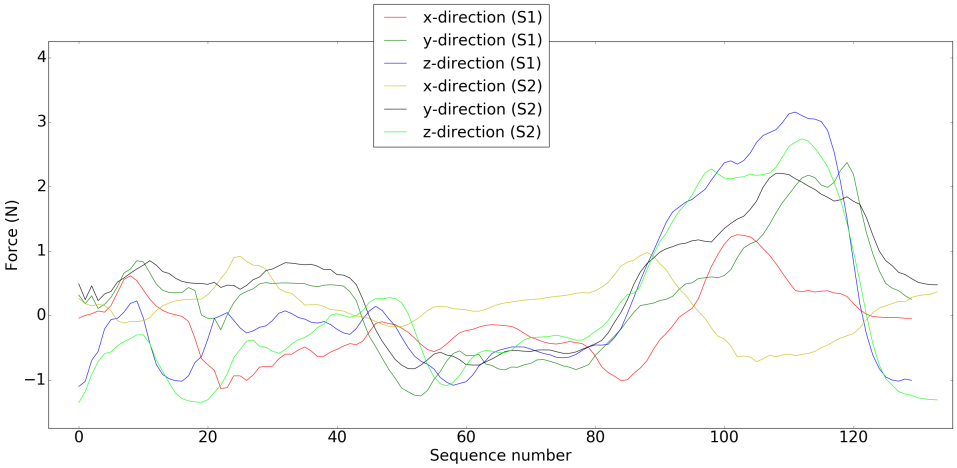
y- and z-directions have similar values, except for the soft mat, where the force provided from the right front foot is much higher.



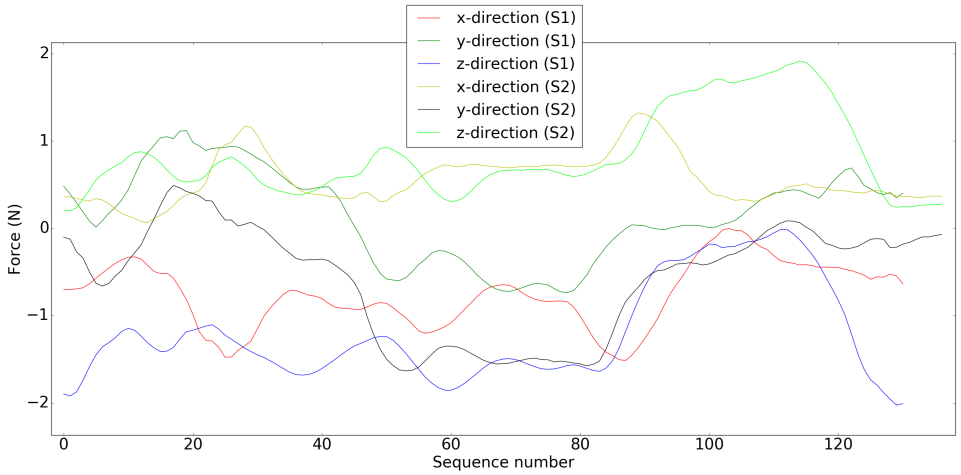
(a) Floor



(b) Carpet



(c) Hard mat



(d) Soft mat

Figure 5.12: Comparison between sensor data with the sensor mounted on the left and on the right front foot for each terrain: floor (a), carpet (b), hard mat (c), and soft mat (d). (S1) and (S2) refer to sensor data provided from front left foot and front right foot respectively.

Chapter 6

Discussion

6.1 Classifiers performance

Many classifiers have given a good performance; however, when the models were tested on new samples, the performance decreased, as seen in section 5.3. The neural network and decision tree on raw sensor data did not give adequate performances. There is a high chance that these classifiers were overfitted and therefore not able to generalize the data well. It is also worth mentioning that the method of decimating the raw sensor data to a fixed length described in section 4.5 might also affect the performance. The method discards features at the start and the end of the feature vector because they were considered as less important. However, the longer the time series is, the more features are removed from the feature vector, with a risk of removing important features. The best performing classifier was the SVM with an overall accuracy of 94.8%. Note that only SVM used the complete frequency domain as features, and therefore it is hard to tell whether it was the classifier or the feature set that achieved the high accuracy.

Common characteristics are that the classifier tended to predict hard mat as carpet, and the soft mat in every trial achieved a high f-score as shown in section 5.3. These common characteristics might be due to the sensor data as seen in figures 5.1 and 5.2. Sensor data on hard mat and carpet have the most similar features and are most likely to be confused. While the sensor data on soft mat distinguishes from the other terrains, which is also the reason for a high f-score. Several feature selection methods were applied to remove common features and keep distinctive features. It is shown in table 5.3 that the RFE on feature set three did obtain minor improvement and achieved an accuracy of 97%. It indicates that good features are kept and it is able to distinguish between each terrain, although there is still difficulty predicting the hard mat. The issue of predicting hard mat can be seen in figure 5.4, where there are especially many similar features behind the first ten of the sequence in the x-,y-, and z-directions. A further improvement is to test different values on the RFE, by discarding more features and keeping

more distinctive features.

6.2 Hyperparameter-tuning

The achieved parameters from the grid search did not show notable improvement on the performance of the neural network and SVM. The SVM was already a well performing model, and is therefore difficult to find parameter values to obtain a better performance. Neural network, on the other hand, only searched for the number of neurons in one or two hidden layers, while there are many other settings which should be tested to achieve a more optimal classifier.

Regarding having binary numbers as output for the neural network, there was a minor improvement of the model when representing terrain with two binary numbers. A reason could be that, instead of choosing the output with the highest score, it treats every output equally. That is, every neuron that meets its threshold will be taken into account. Representations with four binary numbers, on the other hand, consider a terrain as unknown when more than two neurons fire. Allowing unknown terrains in the model leads to a higher rate of misclassification. However, despite being able to identify unknown terrains, it may be suitable if the robot is exploring or detecting new terrains.

6.3 Transition between terrains

The result achieved in the transition between terrains in section 5.5.2 has given an overall accuracy for all trials of 86.8%. The overall accuracy of transition from floor to carpet is 88.9%, which indicates that the classifier is able to identify terrains even with minor differences in their properties. However, after the transition from hard mat to floor, the classifier is confused between floor and carpet, as seen in table 5.5.2. This confusion has been shown from an earlier experiment in table 5.5, where the carpet tends to have a small misclassification as floor. It is also worth mentioning that the robot tends to not walk straight during the experiments, which might cause the confusion between the terrains. All of the training samples were obtained when the robot walked straight. Thus, there is a high chance of affecting the classification when the robot walks toward the right or left.

Regarding estimated probability on the first step on the new terrain, one can observe that the estimated probabilities are typically low. These issues might come from the fact that all training samples were collected when the robot walked on a certain terrain. Preventing this can be done by gathering training samples where the robot traverses through different terrains. Another thing to be observed is the step in the beginning, where the estimated probability of correct terrain is also typically low. It might indicate that the

first step has a particular behavior, as in the middle the robot got more stable. Thus, extracting more data from the first step may improve the result.

Regarding how fast the robot detects new terrain, when the two terrains have great differences in their properties, such as floor and soft mat, it detects the new terrain quickly. Terrains with less difference in their properties may not be detected as fast. Results of the transition from floor to carpet in table 5.15 show that the classifier sensed the new terrain after the second step on the new terrain. Note that the experiments are mostly based on transitions with the soft mat, which only gives an indication of how fast it detects the terrains with high difference in their properties. Thus, to give a more reliable evaluation, more experiments on transitions between floor, carpet, and hard mat is necessary.

6.4 Predicting on the other sensor

The sensor data provided from the front right foot differs somewhat from the data from the front left foot, as seen in figure 5.12. The differences may come from when mounting the sensor on the robot. There is a possibility that the axis from the sensor on the front left foot does not precisely point in the same direction as the sensor on the front right foot. Despite differences in sensor data, the results from table 5.20 indicate that most of the terrains have been predicted correctly, but predicting the hard mat correctly is still challenging. Thus, it is possible to use the training data from the front left foot on the front right foot when incorporating them, but to achieve a more reliable result, it is best to train each sensor independently.

6.5 Compared to earlier work

To give an overview of the thesis performance, a comparison with earlier work is shown in table 6.1. These approaches were all performed with local sensing and tested on unseen data. The thesis approach has among the top performers and obtains similar result as in [14, 68, 61, 9] with an accuracy of more than 90%. Note that the experiments in [14, 68] were performed by a wheeled robot, which more easily achieves maintained stability and less noisy data when preprocessing data from a sensor than a legged robot. The thesis approach also outperforms past work which utilized more than one sensor [15, 3]. The work in [9] only utilized the servo and demonstrated an accuracy of 97%. However, the authors were incorporating all six servos from each leg which produces more sensor data to be preprocessed. It is also worth mentioning the work achieved higher accuracy when rocks and mulch were considered as one terrain. Utilizing more sensors or incorporating all four sensors in this work might also improve the results. The most similar comparison is with work reported in [11], where a similar quadruped robot and tactile sensor were used in

the experiments. However, the work does not report the type of tactile sensor used, which makes it difficult to compare the sensor type itself. The experiments were tested on blue foil, styrofoam, linoleum, cardboard, and rubber, which have great differences in their properties and achieved an accuracy of 84.69%. As the robot in this thesis is able to distinguish between floor, carpet, and hard mat, it should also be able to predict all of those as well.

Approach	Sensor type	Classifier	Number of terrains	Accuracy
Hoepflinger el at. [64]	Force sensing Joint motor currents	Adaboost	4	73.3%
Walas [12]	Tactile sensor	Linear Discriminant Analysis	5	76.67%
Kim el at. [61]	Ground reaction force Torque sensors	SVM	4	78.75%
Degrave el at. [11]	**Tactile sensor**	**Reservoir Computing**	**5**	**84.69%**
Kertész [15]	Accelerometer Paw sensor	Naive Bayes	5	90.9%
Bermudez el at. [3]	Vibration motor control data magnetic encoders	SVM	3	93.8%
Giguere and Dudek [14]	Tactile probe	Neural network	10	94.3%
Best el at. [9]	Servos	SVM	4	97.3%
Thesis approach	Optical tactile sensor	SVM	4	94.8%

Table 6.1: Comparison between thesis approach with other approaches on unseen data. Thesis result is the mean of the accuracy achieved from LOOCV on unseen data using the best performed model obtained in section 5.3.2. The most similar approach with robot platform and sensor used is marked with bold text.

6.6 Conclusion

The aim of this thesis was to investigate the possibility of terrain classification using a 3D optical force sensor. The approach was machine learning on force data provided from the sensor mounted on a quadruped robot. Four different terrains - floor, carpet, hard mat, and soft mat - were used in the experiments. The data set consisted of the robot walking on each terrain separately. Although the terrains such as the floor, carpet, and hard mat have similar properties, analysis of the sensor data showed that the sensor is able to perceive small differences between them. The differences can be found in the sensor data in the y- and z-directions, where the amount of force and the shape of time series differ between each terrain. Sensor data

in the x-direction, on the other hand, provided almost identical features from the floor, carpet, and hard mat.

The evaluation of the force sensor was done by using five different classifiers - naive Bayes, decision tree, KNN, neural network, and SVM - with five different feature sets. The feature sets consist of either using decimated raw data in the time domain or frequency domain, or statistical features in the time domain, frequency domain or both. Many of the classifiers obtained a high accuracy during the cross validation. However, when selecting five well-performing models on unseen data, the accuracy dropped as anticipated. The neural network and decision tree, which had an accuracy of over 90%, decreased to 64% and 78% respectively. This may indicate that the models were overfitted and did not generalize data well. The combination of selected features from the complete frequency domain and SVM gave the best results with an accuracy of 94.8%. This feature set seems to contain distinctive and informative features that make the terrains distinguishable. However, the model still had difficulty predicting the hard mat.

A real-time implementation was implemented in order for the purpose of a more realistic scenario when the robot was walking through different terrains. The real-time implementation was able to segment sensor data and make the classification fast. However, since the algorithm was only tested on a certain gait and flat surfaces, one might achieve a different performance when another gait or more rough terrain is chosen. The experiment showed a feasible result with an overall accuracy of 86.8%. Note that there were some trials where the robot did not have straight locomotion, which might have decreased the performance

The last experiment demonstrated the possibility of using training data from a sensor mounted on the front left foot to predict front right foot data samples. The data provided from both sensors showed differences but had some of the same shapes of the time series. The results of this experiment indicate that one can obtain more training samples by using sensor data from both sensors. However, in case the legs have different walking behaviors, it may be more appropriate to have separate training samples for each sensor.

The results confirm that the optical force sensor is suitable to use for terrain classification. Comparing the results obtained from cross-validation on unseen data from earlier studies, the thesis approach has achieved among the top performances with an accuracy of 94.8% on SVM. Note that this work only utilized one sensor on a legged robot to discriminate the terrains, while much of the other past work have either used sensor fusion or wheeled robots, which provide more stable locomotion to achieve a good result. As the sensor in this thesis is able to distinguish terrains such as floor, carpet, and hard mat, which have similar properties, there is a high chance of it correctly predicting outdoor terrains as well.

6.7 Future work

The classifier is based on the features from the sensor data. However, features of the terrain itself such as hardness and friction are not analyzed directly, which can be addressed in future work.

This thesis has only experimented with flat terrains. It will be interesting to include rougher surfaces or other types and investigate the performance with the current model.

This thesis has mostly used sensor data from one leg. However, incorporating all four legs might increase the performance of terrain classification.

Many authors have utilized sensor fusion to achieve feasible results. Thus, using data from other sensors such as leg motion, servos etc. could further aid the performance.

Instead of explicitly labeling the terrain, having unsupervised learning could be appropriate. This gives the robot the ability to explore and detect terrains by itself.

Only one robot was used to evaluate the sensor. However, different platforms may not give the same results. Thus, different robot platforms should be tested to achieve a more reliable evaluation of the sensor.

One of the important abilities for a robot to achieve terrain classification is to change their gaits on different terrains. Thus, the approach can be further extended to give the robot the ability to select appropriate gaits between different terrains.

Bibliography

[1] P. Giguere, G. Dudek, C. Prahacs, and S. Saunderson, "Environment identification for a running robot using inertial and actuator cues," in *ROBOTICS: SCIENCE AND SYSTEMS*, pp. 271–278, 2006.

[2] P. Giguere and G. Dudek, "Clustering sensor data for autonomous terrain identification using time-dependency," *Autonomous Robots*, vol. 26, no. 2, pp. 171–186, 2009.

[3] F. L. G. Bermudez, R. C. Julian, D. W. Haldane, P. Abbeel, and R. S. Fearing, "Performance analysis and terrain classification for a legged robot over rough terrain," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 513–519, Oct 2012.

[4] S. Manjanna, G. Dudek, and P. Giguere, "Using gait change for terrain sensing by robots," in *2013 International Conference on Computer and Robot Vision*, pp. 16–22, May 2013.

[5] A. C. Larson, R. M. Voyles, J. Bae, and R. Godzdanker, "Evolving gaits for increased discriminability in terrain classification," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3691–3696, Oct 2007.

[6] P. Filitchkin and K. Byl, "Feature-based terrain classification for littledog," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1387–1392, Oct 2012.

[7] W. Bosworth, J. Whitney, S. Kim, and N. Hogan, "Robot locomotion on hard and soft ground: Measuring stability and ground properties in-situ," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3582–3589, May 2016.

[8] C. Plagemann, S. Mischke, S. Prentice, K. Kersting, N. Roy, and W. Burgard, "Learning predictive terrain models for legged robot locomotion," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3545–3552, Sept 2008.

[9] G. Best, P. Moghadam, N. Kottege, and L. Kleeman, *Terrain classification using a hexapod robot*, pp. 1 – 8. Australian Robotics and Automation Association, 2013.

[10] C. Weiss, N. Fechner, M. Stark, and A. Zell, "Comparison of different approaches to vibration-based terrain classification," in *Proceedings of the 3rd European Conference on Mobile Robots, EMCR 2007, September 19-21, 2007, Freiburg, Germany*, 2007.

[11] J. Degrave, R. V. Cauwenbergh, F. Wyffels, T. Waegeman, and B. Schrauwen, "Terrain classification for a quadruped robot," in *2013 12th International Conference on Machine Learning and Applications*, vol. 1, pp. 185–190, Dec 2013.

[12] W. Krzysztof, "Tactile sensing for ground classification," *Journal of Automation, Mobile Robotics and Intelligent Systems*, vol. 7, no. 2, pp. 18–23, 2013.

[13] X. A. Wu, T. M. Huh, R. Mukherjee, and M. Cutkosky, "Integrated ground reaction force sensing and terrain classification for small legged robots," *IEEE Robotics and Automation Letters*, vol. 1, pp. 1125–1132, July 2016.

[14] P. Giguere and G. Dudek, "A simple tactile probe for surface identification by mobile robots," *IEEE Transactions on Robotics*, vol. 27, pp. 534–544, June 2011.

[15] C. Kertész, "Exploring surface detection for a quadruped robot in households," in *2014 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pp. 152–157, May 2014.

[16] K. Walas, "Terrain classification and negotiation with a walking robot," *Journal of Intelligent & Robotic Systems*, vol. 78, no. 3, pp. 401–423, 2015.

[17] M. Hoffmann, K. Štěpánová, and M. Reinstein, "The effect of motor action and different sensory modalities on terrain classification in a quadruped robot running with multiple gaits," *Robotics and Autonomous Systems*, vol. 62, no. 12, pp. 1790 – 1798, 2014.

[18] K. N. Tarchanidis and J. N. Lygouras, "Data glove with a force sensor," in *IMTC 2001. Proceedings of the 18th IEEE Instrumentation and Measurement Technology Conference. Rediscovering Measurement in the Age of Informatics (Cat. No.01CH 37188)*, vol. 1, pp. 380–385 vol.1, May 2001.

[19] E. S. Hwang, J. h. Seo, and Y. J. Kim, "A polymer-based flexible tactile sensor for both normal and shear load detections and its application for robotics," *Journal of Microelectromechanical Systems*, vol. 16, pp. 556–563, June 2007.

[20] A. Wisitsoraat, V. Patthanasetakul, T. Lomas, and A. Tuantranont, "Low cost thin film based piezoresistive {MEMS} tactile sensor," *Sensors and Actuators A: Physical*, vol. 139, no. 1–2, pp. 17 – 22, 2007. Selected Papers From the Asia-Pacific Conference of Transducers and Micro-Nano Technology (APCOT 2006)Asia-Pacific Conference of Transducers and Micro-Nano technology.

[21] Y. Yamada and M. R. Cutkosky, "Tactile sensor with 3-axis force and vibration sensing functions and its application to detect rotational slip," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pp. 3550–3557 vol.4, May 1994.

[22] K. Motoo, F. Arai, and T. Fukuda, "Piezoelectric vibration-type tactile sensor using elasticity and viscosity change of structure," *IEEE Sensors Journal*, vol. 7, pp. 1044–1051, July 2007.

[23] G. M. Krishna and K. Rajanna, "Tactile sensor based on piezoelectric resonance," *IEEE Sensors Journal*, vol. 4, pp. 691–697, Oct 2004.

[24] J. L. Novak, "Initial design and analysis of a capacitive sensor for shear and normal force measurement," in *Proceedings, 1989 International Conference on Robotics and Automation*, pp. 137–144 vol.1, May 1989.

[25] M. Shimojo, "Mechanical filtering effect of elastic cover for tactile sensor," *IEEE Transactions on Robotics and Automation*, vol. 13, pp. 128–132, Feb 1997.

[26] Z. Chi and K. Shida, "A new multifunctional tactile sensor for three-dimensional force measurement," *Sensors and Actuators A: Physical*, vol. 111, no. 2–3, pp. 172 – 179, 2004.

[27] H. Maekawa, K. Tanie, K. Komoriya, M. Kaneko, C. Horiguchi, and T. Sugawara, "Development of a finger-shaped tactile sensor and its evaluation by active touch," in *Proceedings 1992 IEEE International Conference on Robotics and Automation*, pp. 1327–1334 vol.2, May 1992.

[28] S. Begej, "Planar and finger-shaped optical tactile sensors for robotic applications," *IEEE Journal on Robotics and Automation*, vol. 4, pp. 472–484, Oct 1988.

[29] H. R. Nicholls, *Tactile Sensing Using an Optical Transduction Method*, pp. 83–99. Berlin, Heidelberg: Springer Berlin Heidelberg, 1990.

[30] M. Ohka, H. Kobayashi, and Y. Mitsuya, "Sensing characteristics of an optical three-axis tactile sensor mounted on a multi-fingered robotic hand," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 493–498, Aug 2005.

[31] K. Kamiyama, K. Vlack, T. Mizota, H. Kajimoto, K. Kawakami, and S. Tachi, "Vision-based sensor for real-time measuring of surface traction fields," *IEEE Computer Graphics and Applications*, vol. 25, pp. 68–75, Jan 2005.

[32] A. Kolker, M. Jokesch, and U. Thomas, "An optical tactile sensor for measuring force values and directions for several soft and rigid contacts," in *Proceedings of ISR 2016: 47st International Symposium on Robotics*, pp. 1–6, June 2016.

[33] J.-S. Heo, J.-H. Chung, and J.-J. Lee, "Tactile sensor arrays using fiber bragg grating sensors," *Sensors and Actuators A: Physical*, vol. 126, no. 2, pp. 312 – 327, 2006.

[34] A. Dutta, *Brief Review on Integrated Planar Waveguide-Based Optical Sensor*, pp. 9–69. Cham: Springer International Publishing, 2016.

[35] G. C. Haynes and A. A. Rizzi, "Gaits and gait transitions for legged robots," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pp. 1117–1122, May 2006.

[36] M. H. Raibert, "Legged robots," *Commun. ACM*, vol. 29, pp. 499–514, June 1986.

[37] "Nao." Available at https://www.ald.softbankrobotics.com/en/cool-robots/nao/find-out-more-about-nao. Accessed: 2017-04-03.

[38] M. Raibert, K. Blankespoor, G. Nelson, and R. Playter, "Bigdog, the rough-terrain quadruped robot," *{IFAC} Proceedings Volumes*, vol. 41, no. 2, pp. 10822 – 10825, 2008. 17th {IFAC} World Congress.

[39] "Nao - image." Available at https://en.wikipedia.org/wiki/Nao_(robot)#/media/File:Nao_Robot_(Robocup_2016).jpg. Accessed: 2017-04-03.

[40] A. Roennau, G. Heppner, M. Nowicki, and R. Dillmann, "Lauron v: A versatile six-legged walking robot with advanced maneuverability," in *2014 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp. 82–87, July 2014.

[41] M. R. Cutkosky, R. D. Howe, and W. R. Provancher, *Force and Tactile Sensors*, pp. 455–476. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[42] M. Ohka, Y. Mitsuya, S. Takeuchi, H. Ishihara, and O. Kamekawa, "A three-axis optical tactile sensor (fem contact analyses and sensing experiments using a large-sized tactile sensor)," in *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, vol. 1, pp. 817–824 vol.1, May 1995.

[43] M. Ohka, Y. Mitsuya, K. Hattori, and I. Higashioka, "Data conversion capability of optical tactile sensor featuring an array of pyramidal projections," in *1996 IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems (Cat. No.96TH8242)*, pp. 573–580, Dec 1996.

[44] M. Ohka, Y. Mitsuya, Y. Matsunaga, and S. Takeuchi, "Sensing characteristics of an optical three-axis tactile sensor under combined loading," *Robotica*, vol. 22, no. 2, p. 213–221, 2004.

[45] Á. Tar and G. Cserey, "Development of a low cost 3d optical compliant tactile force sensor," in *2011 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pp. 236–240, July 2011.

[46] S. Hirose and K. Yoneda, "Development of optical six-axial force sensor and its signal calibration considering nonlinear interference," in *Proceedings., IEEE International Conference on Robotics and Automation*, pp. 46–53 vol.1, May 1990.

[47] C. Melchiorri, L. Moriello, G. Palli, and U. Scarcia, "A new force/-torque sensor for robotic applications based on optoelectronic components," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6408–6413, May 2014.

[48] G. D. Maria, C. Natale, and S. Pirozzi, "Tactile sensor for human-like manipulation," in *2012 4th IEEE RAS EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pp. 1686–1691, June 2012.

[49] S. Marsland, *Machine Learning: An Algorithmic Perspective*. Chapman & Hall/CRC, 1st ed., 2009.

[50] "mlp." Available at https://en.wikipedia.org/wiki/Artificial_neural_network#/media/File:Colored_neural_network.svg. Accessed: 2017-03-01.

[51] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[52] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, pp. 21–27, January 1967.

[53] "Knn classification." Available at https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#/media/File:KnnClassification.svg. Accessed: 2017-03-14.

[54] Y. Bao, N. Ishii, and X. Du, *Combining Multiple k-Nearest Neighbor Classifiers Using Different Distance Functions*, pp. 634–641. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004.

[55] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[56] C. wei Hsu, C. chung Chang, and C. jen Lin, "A practical guide to support vector classification," 2010.

[57] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. New York, NY, USA: Cambridge University Press, 3 ed., 2007.

[58] I. Guyon and A. Elisseeff, *An Introduction to Feature Extraction*, pp. 1–25. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.

[59] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 67–82, Apr 1997.

[60] P. Refaeilzadeh, L. Tang, and H. Liu, *Cross-Validation*, pp. 532–538. Boston, MA: Springer US, 2009.

[61] K. Kim, K. Ko, W. Kim, S. Yu, and C. Han, "Performance comparison between neural network and svm for terrain classification of legged robot," in *Proceedings of SICE Annual Conference 2010*, pp. 1343–1348, Aug 2010.

[62] K. Walas, D. Kanoulas, and P. Kryczka, "Terrain classification and locomotion parameters adaptation for humanoid robots using force/-torque sensing," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pp. 133–140, Nov 2016.

[63] M. Stejskal, J. Mrva, and J. Faigl, "Road following with blind crawling robot," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3612–3617, May 2016.

[64] M. A. Hoepflinger, C. D. Remy, M. Hutter, L. Spinello, and R. Siegwart, "Haptic terrain classification for legged robots," in *2010 IEEE International Conference on Robotics and Automation*, pp. 2828–2833, May 2010.

[65] R. Jitpakdee and T. Maneewarn, "Neural networks terrain classification using inertial measurement unit for an autonomous vehicle," in *2008 SICE Annual Conference*, pp. 554–558, Aug 2008.

[66] P. Komma, C. Weiss, and A. Zell, "Adaptive bayesian filtering for vibration-based terrain classification," in *2009 IEEE International Conference on Robotics and Automation*, pp. 3307–3313, May 2009.

[67] I. Halatci, C. A. Brooks, and K. Iagnemma, "Terrain classification and classifier fusion for planetary exploration rovers," in *2007 IEEE Aerospace Conference*, pp. 1–11, March 2007.

[68] C. Weiss, H. Frohlich, and A. Zell, "Vibration-based terrain classification using support vector machines," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4429–4434, Oct 2006.

[69] P. Giguere and G. Dudek, "Surface identification using simple contact dynamics for mobile robots," in *2009 IEEE International Conference on Robotics and Automation*, pp. 3301–3306, May 2009.

[70] E. G. Collins and E. J. Coyle, "Vibration-based terrain classification using surface profile input frequency responses," in *2008 IEEE International Conference on Robotics and Automation*, pp. 3276–3283, May 2008.

[71] E. Coyle, E. G. Collins, and R. G. Roberts, "Speed independent terrain classification using singular value decomposition interpolation," in *2011 IEEE International Conference on Robotics and Automation*, pp. 4014–4019, May 2011.

[72] C. Kertész, "Rigidity-based surface recognition for a domestic legged robot," *IEEE Robotics and Automation Letters*, vol. 1, pp. 309–315, Jan 2016.

[73] J. Mrva and J. Faigl, "Feature extraction for terrain classification with crawling robots," in *Proceedings ITAT 2015: Information Technologies - Applications and Theory, Slovensky Raj, Slovakia, September 17-21, 2015.*, pp. 179–185, 2015.

[74] "Optical force sensors - introduction to the technology." Available at https://optoforce.com/file-contents/optoforcewhitepaperopticalforcesensors-0.pdf?v6. Accessed: 2017-05-14.

[75] "Optoforce." Available at http://optoforce.com/technology/. Accessed: 2017-02-27.

[76] "Optoforce." Available at https://optoforce.com/file-contents/comparison-1.pdf. Accessed: 2017-03-22.

[77] T. F. Nygaard, J. Torresen, and K. Glette, "Multi-objective evolution of fast and stable gaits on a physical quadruped robotic platform," in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8, Dec 2016.

[78] "Robot operating system." Available at http://wiki.ros.org/ROS/Introduction. Accessed: 2017-04-17.

[79] "Ros concepts." Available at http://wiki.ros.org/ROS/Concepts. Accessed: 2017-04-17.

[80] "Ros driver for the optoforce sensor." Available at https://github.com/shadow-robot/optoforce. Accessed: 2017-03-15.

[81] "scikit-learn." Available at http://scikit-learn.org/stable/index.html. Accessed: 2017-03-15.

[82] "runstats." Available at https://pypi.python.org/pypi/runstats/. Accessed: 2017-03-15.

Appendices

Appendix A

Code segmentation of sensor data

The custom method of segmenting desired sensor data is shown in listing A.1.

```
//Used to storing raw sensor data continuously
std::vector<std::vector<float>> rawDataS0;

//Used to detect whether the foot is on ground or not
std::vector<std::vector<float>> stabilityLst;

//Used to storing desired sensor data
std::vector<std::vector<float>> interestDataS0;

//Enables when the foot is on ground
int start_to_fill = 0;
int fill = 0;


void checkDesiredData(int sensorNr)
{
    int maxBoundary = 145;
    if(sensorNr == 0) {
        interestDataS0.resize(interestDataS0.size() - stabilityLst
            .size());
        if (interestDataS0.size() < expected_length_data ||
            interestDataS0.size() > maxBoundary) {
            interestDataS0.clear();
            return;
        }
        }
//Writes the desired data into a file (not shown in code)
    writeToFile();
}

int isStable(std::vector<std::vector<float>> stabilityArray)
{
    float boundary = 0.009;
    int stabilityCount = 0;
    int pos1 = stabilityArray.size()-2;
```

```
    int pos2 = stabilityArray.size()-1;

    int i;
    for(i=0;i<3;i++){
        float current = stabilityArray.at(pos1).at(i);
        float next = stabilityArray.at(pos2).at(i);

        if (fabs(next - current) > boundary) {
            return 0;
        }
    }
    return 1;
}

void handleDataS0()
{
    if(stabilityLst.size() > 2){
        if(fill == 0 && isStable(stabilityLst)){
            fill = 1;
        }
        else if(fill) {
            if (!isStable(stabilityLst)) {

                if (stabilityLst.size() > stabThres) {
                    start_to_fill = 1;
                    int lastElement = stabilityLst.size()-1;
                    int secondLast = stabilityLst.size()-2;
                    interestDataS0.push_back(stabilityLst.at(
                        secondLast));
                    interestDataS0.push_back(stabilityLst.at(
                        lastElement));
                }
                fill = 0;
                stabilityLst.clear();
            }else if( start_to_fill && stabilityLst.size() >=
                stabThres  ){
                start_to_fill = 0;
                checkDesiredData(0);
            }
        }
        else{
            fill = 0;
            stabilityLst.clear();
        }
    }
}

//This function obtains sensor data
void optoforceCallback0(const geometry_msgs::WrenchStamped::
    ConstPtr& msg)
{
    std::vector<float> tmp;
        //Storing sensor data
    tmp.push_back(msg->wrench.force.x);
    tmp.push_back(msg->wrench.force.y);
    tmp.push_back(msg->wrench.force.z);
    rawDataS0.push_back(tmp);

    if(start_to_fill){
```

```
            interestDataS0.push_back(tmp);
        }
        stabilityLst.push_back(tmp);
        handleDataS0();
}
```

Listing A.1: A simplified algorithm to segment desired sensor data.

Appendix B

Selected features

The following paragraph shows the five different feature sets with their features selected by RFLV and RFE.

Feature set one

RFLV

$$\begin{aligned} f_{set1} = \{ & x_8, x_{19}, \ldots, x_{24}, x_{27}, x_{53}, \ldots, x_{61}, x_{80}, \ldots, x_{83} \\ & x_{86}, \ldots, x_{119} \\ & y_{14}, \ldots, y_{27}, y_{52}, \ldots, y_{60}, y_{81}, \ldots, y_{84}, y_{101}, \ldots, y_{123} \\ & z_1, \ldots, z_{125} \} \end{aligned} \tag{B.1}$$

RFE

$$\begin{aligned} f_{set1} = \{ & x_1, x_5, x_6, x_9, x_{12}, x_{13}, x_{14}, x_{19}, x_{20}, x_{23}, x_{25}, x_{26}, \\ & x_{35}, x_{39}, x_{45}, x_{47}, x_{50}, x_{51}, x_{52}, x_{55}, x_{56}, x_{60}, x_{62}, x_{64}, x_{65}, x_{66}, \\ & x_{72}, x_{73}, x_{74}, x_{76}, x_{80}, x_{81}, x_{84}, x_{88}, x_{91}, x_{93}, x_{94}, x_{97}, x_{98}, x_{103}, \\ & x_{105}, x_{107}, x_{108}, x_{109}, x_{111}, x_{113}, x_{114}, x_{116}, x_{118}, x_{119}, x_{120}, x_{123}, x_{124}, x_{125}, \\ & y_1, y_2, y_3, y_6, y_9, \ldots, y_{13}, y_{16}, y_{17}, y_{26}, \ldots, y_{30}, y_{33}, y_{34}, y_{37}, \ldots, y_{40}, \\ & y_{42}, y_{44}, y_{47}, \ldots, y_{52}, y_{54}, y_{57}, \ldots, y_{60}, y_{63}, y_{65}, y_{66}, y_{67}, y_{69}, \\ & y_{70}, y_{71}, y_{73}, y_{74}, y_{76}, y_{77}, y_{78}, y_{81}, y_{83}, y_{85}, y_{86}, y_{89}, y_{91}, y_{94}, y_{97}, y_{98}, \\ & y_{101}, y_{102}, y_{103}, y_{105}, y_{106}, y_{109}, y_{110}, y_{114}, y_{116}, y_{118}, y_{121}, y_{122}, y_{124}, y_{125}, \\ & z_2, z_3, z_4, z_6, z_9, z_{10}, z_{11}, z_{13}, z_{27}, \ldots, z_{30}, z_{32}, z_{35}, z_{36}, z_{38}, z_{39}, z_{40}, \\ & z_{43}, z_{44}, z_{47}, z_{48}, z_{49}, z_{51}, z_{53}, z_{54}, z_{55}, z_{58}, z_{59}, z_{60}, z_{63}, z_{64}, z_{66}, z_{67}, z_{68}, \\ & z_{72}, \ldots, z_{75}, z_{78}, z_{79}, z_{81}, \ldots, z_{84}, z_{88}, z_{89}, z_{92}, z_{95}, \\ & z_{100}, z_{103}, z_{106}, \ldots, z_{109}, z_{115}, z_{116}, z_{119}, z_{120}, z_{121}, z_{122}, z_{125} \} \end{aligned} \tag{B.2}$$

Feature set two

RFLV

$$f_{set2} = \{x_{max}, x_{kurtosis}, x_{skew}, y_{max}, z_{min}, z_{max}, z_{mean}, z_{var}\} \tag{B.3}$$

RFE

$$f_{set2} = \{x_{var}, x_{std}, y_{max}, y_{kurtosis}, y_{skew}, y_{var}, z_{max}, z_{kurtosis}, z_{var}, z_{std}\} \tag{B.4}$$

Feature set three

RFLV

$$f_{set3} = \{fx_1, fz_1\} \tag{B.5}$$

RFE

$$\begin{aligned} f_{set3} = \{&fx_1, \ldots, fx_8, fx_{10}, fx_{13}, \ldots, fx_{19}, fx_{21}, fx_{22}, fx_{24}, fx_{27} \\ &fx_{30}, fx_{32}, fx_{33}, fx_{34}, fx_{37}, fx_{38}, fx_{39}, \\ &fy_1, fy_3, fy_4, fy_6, \ldots, fy_{13}, fy_{15}, fy_{16}, fy_{20}, fy_{22}, fy_{24}, fy_{25}, \\ &fy_{27}, fy_{28}, fy_{29}, fy_{31}, fy_{32}, fy_{33}, fy_{36}, fy_{37}, fy_{38}, fy_{39}, fy_{51}, \\ &fz_1, \ldots, fz_{11}, fz_{13}, fz_{17}, \ldots, fz_{22}, fz_{24}, fz_{34}, \\ &fz_{36}, fz_{38}, fz_{39}, fz_{51}, fz_{59}, fz_{60}, fz_{61}\} \end{aligned} \tag{B.6}$$

Feature set four

RFLV

$$f_{set4} = \{fx_{max}, fx_{kurtosis}, fx_{skew}, fx_E, fy_{max}, fy_E, fz_{min}, fz_{max}, fz_{mean}, fz_{var}, fz_E\} \tag{B.7}$$

RFE

$$f_{set4} = \{fx_{var}, fx_{std}, fy_{max}, fy_{mean}, fy_{kurtosis}, fy_{skew}, fy_{std}, fy_E, fz_{min}, fz_{max}, fz_{kurtosis}, fz_{var}\} \tag{B.8}$$

Feature set five

RFLV

$$f_{set5} = \{x_{max}, x_{kurtosis}, x_{skew}, fx_{max}, fx_{kurtosis}, fx_{skew}, fx_E, \\ y_{max}, fy_{max}, fy_E, \\ z_{min}, z_{max}, z_{mean}, z_{var}, fz_{min}, fz_{max}, fz_{mean}, fz_{var}, fz_E\} \tag{B.9}$$

RFE

$$f_{set5} = \{x_{var}, x_{std}, fx_{mean}, fx_{var}, fx_{std}, fx_E, \\ y_{skew}, fy_{max}, fy_{kurtosis}, fy_{skew}, fy_E, \\ z_{min}, z_{max}, z_{mean}, z_{kurtosis}, z_{var}, fz_{min}, fz_{max}, fz_{mean}, fz_{kurtosis}\} \tag{B.10}$$