UiO : Department of Informatics
University of Oslo

Terrain classification using 3D optical force sensor

A machine learning approach

Jiader Chou

Master's Thesis Spring 2017

Terrain classification using 3D optical force sensor

Jiader Chou

March 29, 2017

Abstract

This thesis will use optical force sensor for terrain classification. The experiment use different classifier to predict 4 different terrains...

Contents

List of Figures

List of Tables

Preface

Chapter 1

Introduction

Humans will adapt their walking style on different terrain to have a stable locomotion, and to avoid falling. Adapting the walking style comes of past experiences. For instance one has experienced that running on icy road may cause a fall, while running on dry road will be fine. To achieve this ability, the robot must have the ability of distinguish different terrains.

New and improved sensors have made it possible to distinguish terrain even more accuracy. A popular sensor used is gather information visually from terrain such as camera [8], laser scanners [9]. These types of sensor gather information from terrain indirectly. While other use sensor that measure the properties of the terrain more directly such as tactile, joint angle, accelerometer and gyroscope. Various of combination of sensors have been investigated in previous work. Degrave et al.[10] investigated different types and combinations of sensors for a Quadruped Robot to identify which of them is suitable and provide most information on the terrain. The result showed that the most informative sensors were a combination of tactile sensors and proprioceptive joint angle sensors. A type of sensor which use refracted light intensity to measure the contact force has been less focused within terrain classification, that is the 3-axis optical force sensor. Due to high sensitivity, small size, light weight and low detection time [11], the sensor suit to obtain information from terrain and distinguish them. Additionally the sensor measure the force in three dimensions. The goal of this thesis is to use an 3D optical force sensor and investigate whether it can distinguish terrains with slightly difference such as floor and carpet.

To evaluate the quality of terrain classification with the sensor, several learning techniques are needed. There are many different learning techniques and some is more suitable for some sensor than other sensor. The thesis will also look into how to preprocess data, select different features and test several classifier to find which is most suitable.

1.1 Outline

The thesis is divided into five chapters: introduction, background, implementation, experiments and results and conclusion.

Chapter 1: Introduction Gives a brief introduction of the terrain classification, and motivation in this project.

Chapter 2: Background The background chapter presents first related work of terrain classification, followed by the optical force sensor used in this thesis. The last section presents various machine learning approaches.

Chapter 3: Implementation The implementation chapter present the experiment setup, how the data is preprocessed and used in learning process.

Chapter 4: Conclusion Lastly, the conclusion chapter is discussion of experiments, along with suggestions for future work.

Chapter 2

Background

2.1 Terrain classification

Terrain classification is the process of identify different types of terrain such as rocks, wood, grass and sand. The challenging of discriminate terrain is the variation of its characteristics such as slope, roughness, hardness and friction. The terrain classification has been applied 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 traverse on more difficult terrains. In this thesis a legged robot will be used to perform terrain classification.

2.1.1 Types of legged robots

2.1.2 Terrain classification for legged robots

The importance of terrain classification for legged robots is that the terrain has a major factor affecting the decision for the gait change [12]. Most of the legged robot has the benefit of changing their gait and walking speed to achieve a stable locomotion. For instance a running gait can be efficient on flat road, but fail to maintain the stability of the robot when used on ice, cause of slipping. An example of application is shown in Giguere et al. [13], where a amphibious legged robot can switch from walking to swimming gait, according to which of the terrain it is on.

Bosworth et al. [14] used a quadrupedal robot hopping on soft and hard terrain and showed that different controllers were better suited for different terrains.

Manjanna et al. [12] investigated the effect of performance of with different gaits parameters on different terrains. Some of the result showed that the energy consumption and the walking speed is tied to the terrain type. That is, a trade-off between the physical speed and the power consumption of the robot can be achieved by controlling the cycle-frequency of the leg ro-

tation.

2.1.3 Terrain sensing

To be able to classify various of terrains, the system must obtain information from the terrain either by remote sensing, local sensing or both.

Remote sensing The remote sensing obtain information of a terrain from a distance and do not measure the terrain physically. The camera is most widely used and discriminating the terrain is based on analyzing images.

Filitchkin et al. [8] presents an visual terrain classification by using a single, compact camera to change the gait patterns of a quadruped robot. Three types of gaits were used during the experiment, "a gait that is optimized for speed on relatively flat surfaces, a gait designed for high clearance on rough terrain at the expense of speed and a mixture of the two firsts". To know which gait should be chosen for each terrain, an initial test by assigning a gait to each terrain type is required. There were totally four different terrain small rocks, rocks, grass and tile. Lastly, the experiment was let the robot execute a terrain classification cycle every few steps and switched to one of three gait according which terrain it is on. To measure the robot performance, they compared the traversal time between for each three gait and the changing gait. The result indicate that changing gait for each terrain is faster than using a single gait to traverse.

A weakness of using the remote sensors, is that does not give insight into the characteristic of currently terrain. For instance remote sensor has difficulties to distinguish between a terrain that is covered with either compacted or uncompacted snow. An another option is to measure terrain directly with local sensing.

Local sensing Local sensing measure aspects of the interaction between the robot and terrain as the robot moves through the terrain. This gives a measurement of mechanical terrain properties and provide useful information such as how the environment is affecting currently robot performance.

Stejskal et al. [15] present a road following hexapod robot by using the feedback from robot servo drives. The road following consists of let the robot blindly walk on road. After each gait cycle, the robot will determines whether it is on new terrain or on road. If it is determined as off road, then the robot will steer back to the road. Data from the servo provides information about the leg motion which were position error, current speed and torque. Three different terrains, asphalt, dirt and grass were used during the experiment. The robot was most confused by dirt, which had about

86% of misclassified samples. The author states that the transition from asphalt to dirt was usually flat, which means that the leg motion of walking on flat dirt has a similar leg motion of walking on asphalt. Beside of that, the overall result of terrain classification had an accuracy of 96.2%, which can be considered as feasible approach and data from leg motion keeps the robot on the road.

Kim et al. [16] used the ground reaction force and torque sensors of an one-legged robot due to terrain classification. The goal of the research was to compare two different classifiers (neural network and support vector machine) for distinguish four different terrains, which were flat, grass, sand, and gravel. The data were collected by walking through each terrain many times. Different features were extracted from the data, and partitioned into a training and test set. The result shown the support vector machine got an accuracy on 78.75%, which performed slightly better than neural network on 78.6%.

Hoepflinger et al. [17] present a novel approach to terrain classification for legged robots by using properties from joint motor currents and force sensing resistor. The goal were to improve the guiding of foot placement and stability of legged robots in rough terrain. Usually experiments is done by having a robot walk through terrains. However in this experiment the author separated one of the legs robot and mounted to a sample holder of a testbed. The first experiment consist of distinguish four different shaped terrain: a convex and a concave cone, a convex hemispherical bulge, and a concave hemispherical indentation. The data were collected by the knee-joint oscillated slightly with an amplitude of about one degree on the terrain. In the second experiment were to distinguish between three different types of abrasive paper and a low friction PTFE coating. Collecting of data were done by performing a scratching motion on the terrain. The result of terrain shape classification had a high success rate. Convex cone and concave cone had 100% accuracy, while the concave hemisphere bulge 95% and 80% on for the convex hemispherical bulge. Regarding surface classification show that the algorithm performed slightly worse. A type of a abrasive paper had a accuracy of 53.3%. Even a low accuracy the author state it is overall an satisfy result, since the average correct prediction of different types of abrasive paper was 2 of 3, and prediction of teflon coated surface had an high accuracy 93.3%.

Many other research of terrain classification can be found in [18, 19, 12, 12, 20]

2.2 Optical force sensor

Optical force sensors use the refracted light intensity to measure the deformation of the silicon [21][22]. The sensor used in this thesis is OptoForce(3D force sensor) [23]. A construction of the sensor is shown

on the figure 2.1.

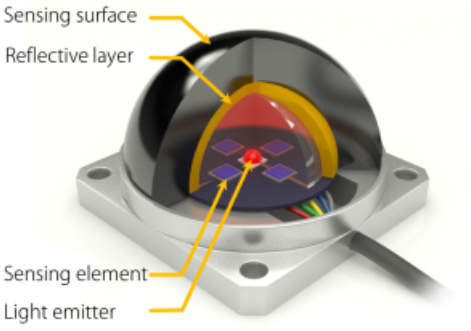


Figure 2.1: Figure showing construction of the 3d optical force used in thesis [1]

The sensor consists of a light emitter (LED) and four sensing elements(photodiodes) which is wrapped within two layers, a reflective layer and a sensing surface. The four photodiodes will measure the force by measure the infared light reflected by the reflective layer. If a force is applied on the sensing surface, the amount of reflected light to each photodiodes will change accordingly. The forces in x- and y-direction is measured from the difference in amount of reflected light between the two opposing photodiodes for each direction, while the force in z-direction is the average of the four measurements. Which direction x, y and z-direction is shown in figure 2.2. Optoforce sensor is relative new sensor(2015), and the manufacturer claim the sensor can guarantee precise measurements even up to 200% overload [24]. A several work where the sensor is used can be found in [25, 26, 27].

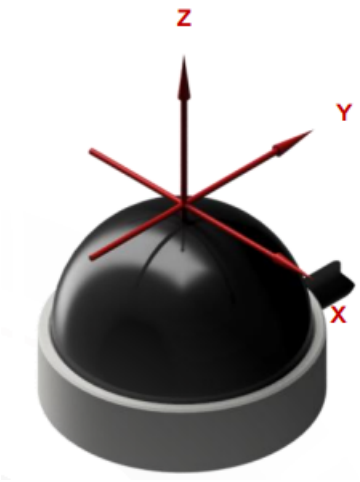


Figure 2.2: Figure showing the x,y and z-direction for the 3d optical force used in this thesis [2].

2.3 Machine learning

Machine learning is a type of artificial intelligence where it can learn to adapt or predict from earlier dataset without being explicitly programmed. Each dataset consists of an feature vector, with an class for each dataset as output. The training process consists of analyzing the each features vector and produce an inferred function, which can be used for comparing new and unseen dataset to a class. The learning algorithm can be separated into supervised, unsupervised, reinforcement learning, evolutionary learning. Supervised learning is most common used within terrain classification, while few unsupervised learning. In this project will supervised learning be used.

Supervised learning Supervised learning algorithms make predictions based on labled dataset. That is, the system know the correct answers to each dataset and make prediction based on that. The learning process usually stops when the algorithm converge towards an acceptable level of performance.

Unsupervised learning Unsupervised learning algorithms make predictions from data points without label. That is, the system has to organize the data on its own and make prediction based on that. A commom method to organize the data is by clustering.

Reinforcement learning Reinforcement leanring algorithms will gets to choose an action in response to each dataset. It will then either receieve a reward indicating how good the decision was. Based on rewards, the algortihms modifies its strategy in order to get the highest reward.

Evolutionary learning Evolutionary learning: Biological organisms adapt to improve their survival rates and chance of having offspring in their environment, using the idea of fitness (how good the current solution is).

2.3.1 Classifier

The classifier is the component with the learning process and produce the inferred function. There is a vast number of classifiers and various of each classifier has been used within the terrain classification, neural network [10, 28, 29], adaptive bayesian filtering [30, 31], support vector machines [16, 32, 33] and decision tree [31]. The following section will introduce technical background of five different classifiers which is intended to be used in this thesis.

Neural network

Artificial Neural Network (ANN) is a computational model based on brain, and is one of the most used learning algortihm. It can be found in many

different research field, such as signal processing, image processing, control, natural language processing etc. A simple model of neuron is shown in figure 2.3. A neuron consists a set of weighted inputs w_i, an adder which sums weighted inputs signals and an activation function to decide whether the neuron should fire with an output o, for the current input, x_i.

A single neuron will not be able to generalize the data set. However, by connecting many neurons togehter, one will obtain neural network, which is more capable to generalize data. One of the most known neural network is the perceptron. Note that neurons in the perceptron are completely indepedent of each other, but are dependent of all inputs and its error. That is, a neuron will not be affected by other neurons performance. Each neurons gives a result based on own weights and the input, adding them together, and comparing the result to its own threshold. The only thing neurons share is the input.

The process of perceptron in supervised learning, is to learn to reproduce a particular target, which is a pattern of firing and non-firing neurons for given input. If some of the neuron got a wrong output, for instance a neuron did not fired when it should, then its weights should be adjusted to make it fire right next time. This is a single layer perceptron and good use if a problem is linearly separable. Adding more layers will make the neural network more complex and powerful. This is also called multi-layer perceptron (MLP), or multi neural network.

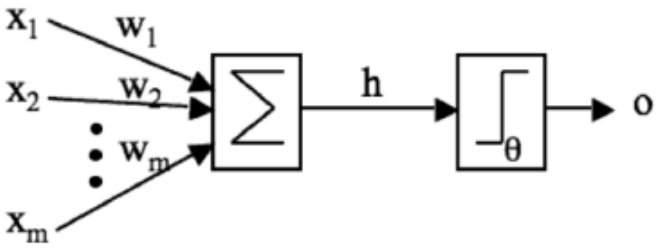


Figure 2.3: Figure showing The three principal approaches of feature selection. The shades show the components used by the three approaches: filters, wrappers and embedded methods [3]

Multi-Layer Perceptron The difference of a multi-layer perceptron and a single-layer perceptron, it that it has two or more layers between the input and output. These layer is also called hidden layer because its values are not possible to change their values directly, and only observed in the training set. The training process can be divded into two parts: forward and backwards. The forward algorithm start first by calculating the activations of the first hidden layer, and use those activations and the next set of weights to calculate the activcation of 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 backwards algorithm will be using the

error to adjust the weights, firstly, between the output-layer to the hidden-layer, and the error in the hidden layer will be computed. The backwards algorithm stops when it has reach the inputs changed all their weights.

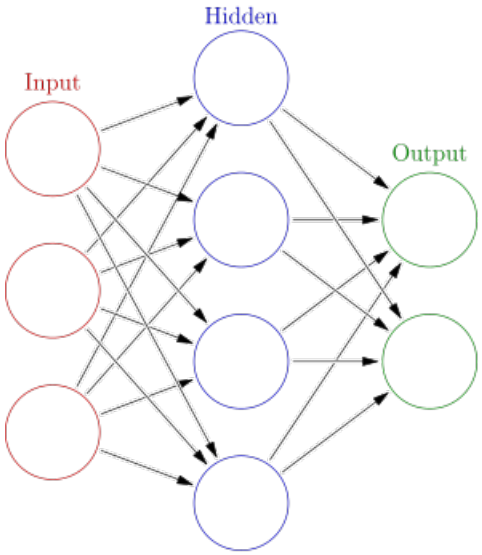


Figure 2.4: Figure showing a multi-layer perceptron with 1 hidden layer [4]. It is possible to increase the number of hidden layers and nodes to each layer

SVM

SVM algorithm was introduced by Vapnik and Chervonekis in 1963, while the standard used today is introduced by Cortes and Vapnik in 1995 [5]. In figure 2.7 is an instance of 2-class classification, SVM. The dotted line is the boundary for the two classes: circle and cross. To predict new set of data, it will check which side of the decision boundary it lies. If a dataset lies under the bouandary line (dotted line) it will classify it as cross and circle if above. If the decision boundary was moved small amount, there is it has a high risk of misclassifying the dataset which lie close to it. This is why SVM will find the the optimal separating hyperplan. It will classifies training data more correctly, and will also generalize better with new unseen data. To find the best decision boundary will be the maximizing the margin. It will be found by find the furthest hyperplane from a data point. If a dataset is non-linearly separable data, the SVM will transform the data into higher dimensional where the data is linearly sepearable to make the classification.

Naive bayes

Naive Bayes algorithm is based on Bayes' theorem the assumption of independence between every pair of features, hence the name "naive". Bayes's theorem in machine learning is often to used to find the probability that a data set A occur, based on the data set B. Several research based the

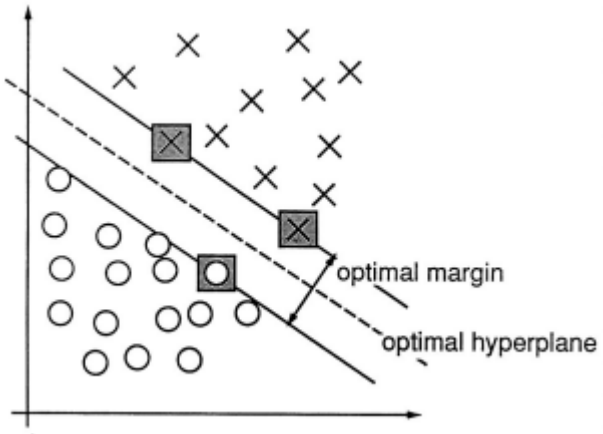


Figure 2.5: Figure showing an example of the optimal margin for a two dimensional two-class classification problem [5]. Each of features vector with values according to the x- and y-axis, represents as classes: circles and crosses. The vectors with values according to the x- and y-axis. The support vectors are the hatced classes which lies on the straight line. "The dotted line is the decision boundary/hyperplane created by the SVM".

terrain classification on naive bayes which has given good result [34, 35]. The bayes' theorem suit to mathematical express the terrain identification problem [13, 34]. This is mathematically simple and allow to evaluate unexpected mismatch between sample classes and the real environments. The terrain classification problem can be expressed as [34]:

$$P(\theta_l|X) = \frac{p(X|\theta_l)P(\theta_l)}{p(X)} \tag{2.1}$$

The θ_l represent the class, which would be the possible terrains the robot is expected to predict. The X represent the observations, which are information from different sensors, also called feature vector. On the formula the 2.1 $P(\theta_l|X)$ is the conditional probability distribution. $P(X|\theta_l)$ is the likelihood that the terrain is has with these observations, and the $P(\theta_l)$ is the priori distribution of the classes, and $p(x)$ is the distribution of the observation. Classification is done by selecting a class that maximizes the a posteriori probability $P(\theta_l|X)$ according to the Maximization of A Posteriori (MAP) decision rule

$$p(X|\theta_l)P(\theta_l) = \max_j\{p(X|\theta_j)P(\theta_j)\} \tag{2.2}$$

The CPDF for each class can be estimated either parametrically or non-parametrically. An accurate nonparametric density is hard to estimate, especially when the number of features is high. One of the most popular parametric probability density functions is the normal distribution, which only requires an estimate of the mean M_l and covariance matrix $\sum_1$ for the class θ_l

$$p(X|\theta_l) = \frac{1}{(2\pi)^{N/2}|\Sigma_l|^{1/2}} \times exp\{-\frac{1}{2}(X - M_l)^T \Sigma_l^{-1}(X - M_l)\} \quad (2.3)$$

K-nearest neighbors

K-nearest neighbors (KNN) one of the simple classifiers presented by T. M. Cover and P. E. Hart in [36]. The classification process of new data consists of looking the k-nearest datapoints and classify the class which consist most classes. For instance if k is set to be 5, then it find the 5 nearest datapoint, and classify the the class which consist most of the 5 classes. An example is shown 2.6

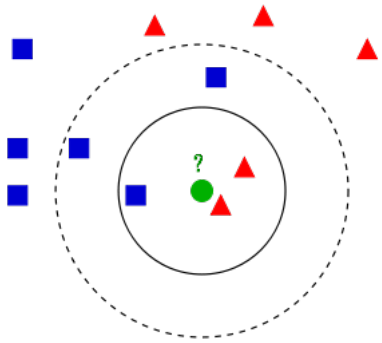


Figure 2.6: Figure showing an example of a KNN[6]. All the blue squares and red triangles are the training data, while the green circle is to be classified. The straight and dashed circle illustrate which data set is taken account when k is set either 3 or 5. If k set to 3, the circle will be classified as triangle, since the tree nearest neighbor consist more triangles than squares. If k set to 5, then the cicle will be classified as square.

There are many approachs to find the k-nearest, but the most common is to calculate the Euclidean Distance. The euclidean distance can be expressed [37]:

$$D(x, y) = \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2} \quad (2.4)$$

The Euclidean distance 2.4, the x and y represent the actual and unseen class, and m is the number of features to each of classes in x and y. The algorithm will then count k classes with shortest distance to determine which class the unseen data belongs to. A weakness of using Euclidean distance function is that if one of the vector has a large range, then it will dominate other attributes. In order to avoid this issue, it is common to scale the feature, described in section 2.4.4

Decision tree

The decision tree is a non-parametric classifier presented by JR Quinlan [7]. The process of the decision tree is constructing a tree consisting of nodes

and edges using the dataset. Each of nodes represents a test on one features, and the edges represent an outcome of the test. Leaf nodes represent the outcome class. Predicting process is following a path from the root to a leaf node.

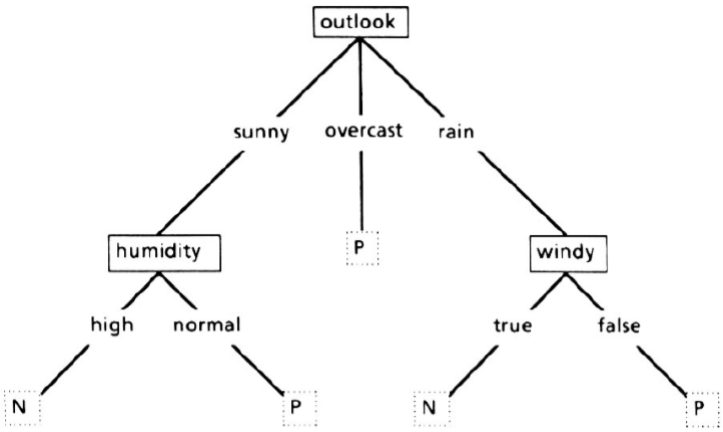


Figure 2.7: Figure showing an example of a simple decision tree [7]. This decision tree classify a day to be P or N based on outlook, humidity and windy.

2.4 Features

An important part of good classification, is finding good features from data from sensors. The features will be used as input, to the learning, and it is the data which distinguish between the classes. Finding good features will make learning to predict more accuracy.

2.4.1 Curse of dimensionality

The curse of dimensionality occurs when one include too many features to the input vector. When the dimensionality of features vector increase, the complexity of underlying pattern might increase and the performance of the classifier will be degrade. To prevent the curse of dimensionality one can add more samples to uncover the underlying pattern.

2.4.2 Features extraction

Feature extraction is the process one would use techniques to build a new set of features from the original set and use it as input. The features extraced should make it easy for a classifier to distinguish between the various classes. An example of extracting good features can be seen in [16]. Using statistic with support vector machine gave an accuracy on 40%, while principal component analysis gave an accuracy on 78.75%. Note that the author only used variance, kurtosis and skewness as statistic features.

A variety of features is used to preprocess raw sensory streams. Leaving visual feature aside, the most common is the combination of stastical moments (mean, varianc,skewness, kurtosis etc.) in the time domain with frequency domain features [38] [18] [17]. While others only use the frequency domain [39] [40].

Statistical features

Formler[41]

Mean The mean is generally referred to the average, and is defined as sum of the values divided by the number of values:

$$\bar{x} = \frac{1}{N} \sum_{N-1}^{j=0} x_j \tag{2.5}$$

Variance Variance describe the spread between numbers in a data set. The variance can bew written as:

$$Var(x_0 \dots X_{N-1}) = \frac{1}{N} \sum_{N-1}^{j=0} (x_j - \bar{x})^2 \tag{2.6}$$

Standard deviation Standard deviation is a measure of spread of a data set from its mean. High deviation indicate that the data points are further from the mean. This can be calculated by taking the square root from variance:

$$\sigma(x_0 \dots X_{N-1}) = \sqrt{Var(x_0 \dots X_{N-1})} \tag{2.7}$$

Skewness Skewness describes asymmetry of a distribution. The formula is:

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

The skewness can be either negative or positive depending on wheter data points are skewed to the left or to the right. If the data is skewed to the left indicate a negative skewness, while postivie skewness is when the data is skewed to the right. An illustration of both positive and negative skewness is shown in figure 2.8.

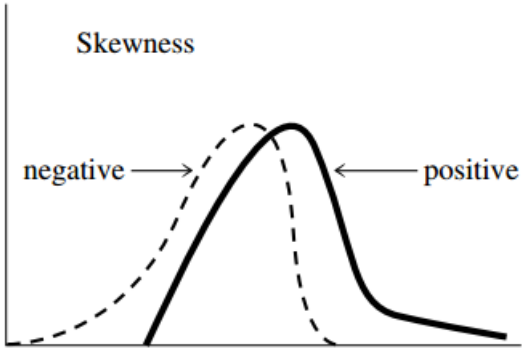


Figure 2.8: Skewness

Kurtosis Kurtosis measure the peak and tails of a distribution relative to a normal distribution. Using kurtosis might help to understand general characteristics about the distribution of the data. The formula is:

$$Kurt(x_0 \dots X_{N-1}) = \left\{ \frac{1}{N} \sum_{N-1}^{j=0} \left[\frac{x_j - \bar{x}}{\sigma} \right]^4 \right\} - 3 \tag{2.9}$$

A positive kurtosis of a distribution has a sharper peak and heavier tails realative to normal distribution. While a negative kurtosis has flatter peak and lighter tails realative to normal distribution. An illustration of both positive and negative kurtosis is shown in figure 2.9

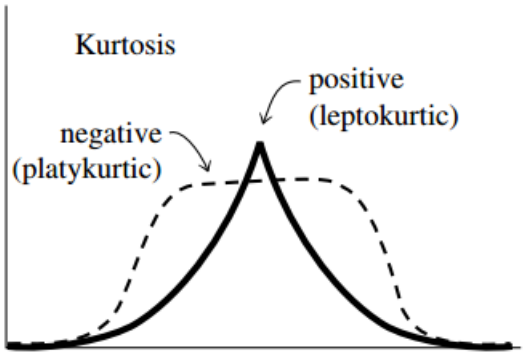


Figure 2.9: Kurtosis

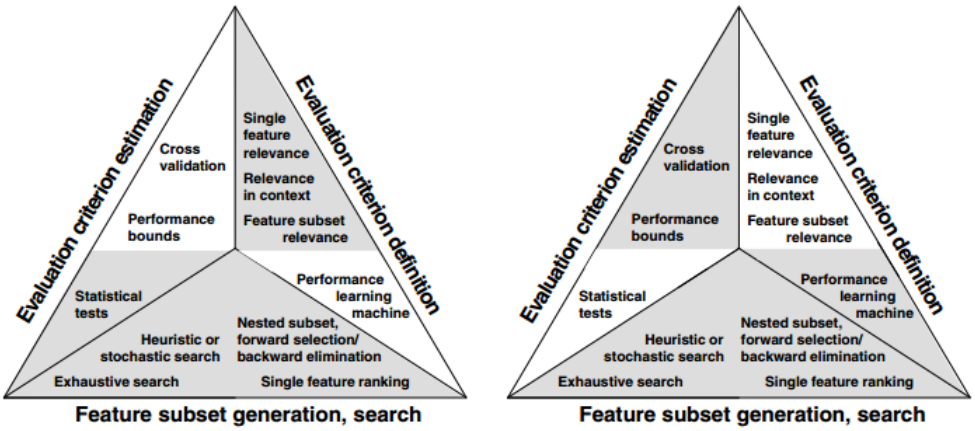
Time domain

Fourier transform

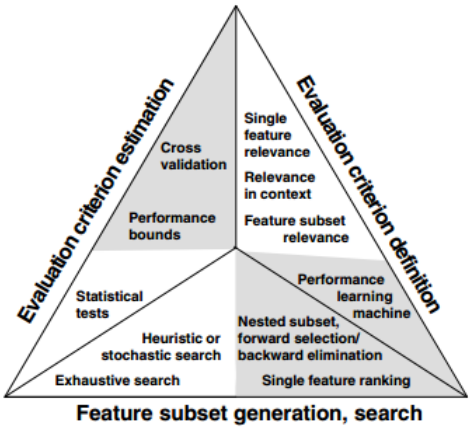
2.4.3 Features selection

The process in feature selection is to select a subset of features from the orignial, that are relevant. Selecting good features has benefit of increasing classifier performance, reducing storage requirements and

reducing training time, and defying curse of dimensionality. Note that a features can individually be completely useless can become relevant when used together with other features [3]. There are three types of feature selection algorithms: filter-, wrapper- and embedded methods.



(a) Filters (b) Wrappers



(c) Embedded methods

Figure 2.10: The figure showing the three principal approaches of feature selection. The shades show the components used by the three approaches: filters, wrappers and embedded methods. [3].

Filter Filter feature selection is independent of any classifier. It uses statistical measure to assign each feature a score. Feature will either be kept or removed based on the score. The filter methods are considered fast and effective, specially when the number of features is large and the number the number of available training examples comparatively small.

Wrapper Wrapper feature selection try different combination of the feature set, evaluated by a classifier and keep the feature set with best result.

Embedded Embedded feature selection perform feature selection as part of the learning procedure and are usually specific to given classifier.

2.4.4 Features scaling

To reduce bias effect cause by skewed distributions, it is common to standardization the feature vectores. By scaling each features, it that avoid features with big number dominate compared features with small numeric ranges. It particularly affect classifier which use distance between classifier such as KNN and SVM. By standardized feature values the features will weights equally in their representation.

2.5 Model validation

2.5.1 No free lunch theorem

The well-known no free lunch theorem[42] in machine learning states that there is no best classifier for every problem. That is, even a model has a great performance for one problem, might not hold for another problem. Previous work has shown that SVM, KNN and NB gave higher accuracy than the decision tree [43], while in [31] show that NB, SVM and DT were better than KNN. It is therefore common to build several algorithms, and chose the best of them for the specified problem.

2.5.2 Ovefitting and underfitting

Overfitting and underfitting the data is a issue in machine learning which cause a poor performance of classifier. Overfitting the data occurs if there are too training data or if it has adapted to the noise in the data. While underfitting occurs if there are not enough training data and will not be able to generalize to new data. An example for each case is shown in figure 2.11. The cross-validation estimate how accurately the classifier model will perform in practice, which might prevent overfitting or underfitting the data.

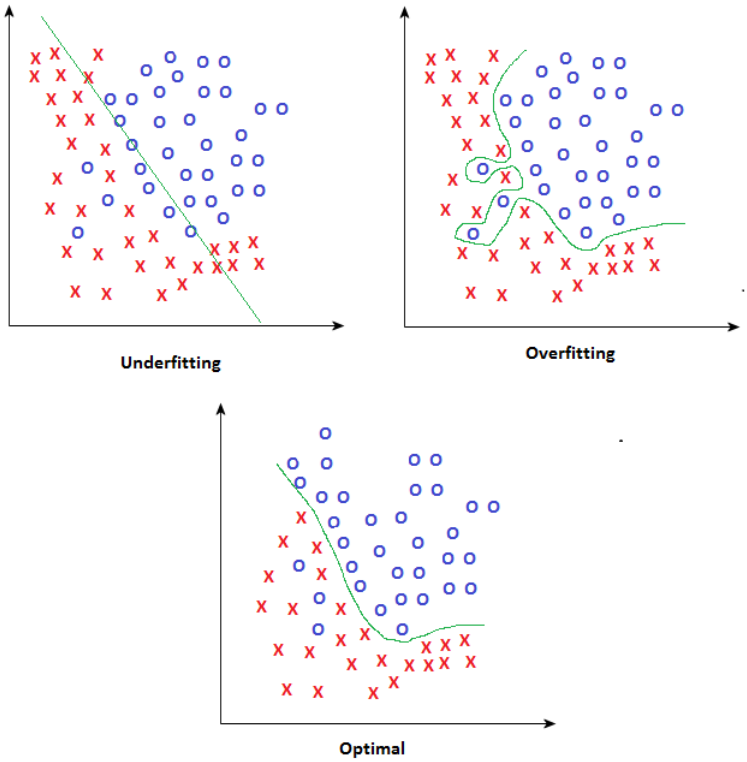


Figure 2.11: The figure showing an example when the classifier is underfitted, overfitted or optimal.

2.5.3 Cross-Validation

Cross-validation is used to assess the quality of a model. The process of cross-validation is to first remove some of the data before the training begins, and after the training, use the removed data to test the performance. The intention is to evaluate the classifier performance in more realistic scenario by predicting new and unseen data. There are several types of cross-validation. However the most common used in terrain classification is the k-fold cross-validation [10, 19, 44, 31, 45].

K-fold The process of k-fold is to partitioned the data into k subset, where one subset is used for testing and the other are used for training. When the trained model has assessed the test set, a new subset is selected as test set. This process is repeated k times, that is when all subsets have been used as a test set. Setting the k to the length of feature vectors, it is also know as leave-one-outcross validation (LOOCV). LOOCV only use one feature vector as test set, while the rest as training set as shown in figure 2.12. The advantages is that using as many training samples as possible.

2.5.4 Evaluating Classifiers

Evalution of the performance of the classifier can be shown by looking at the output, which could be either correct or wrong. For instance a two-

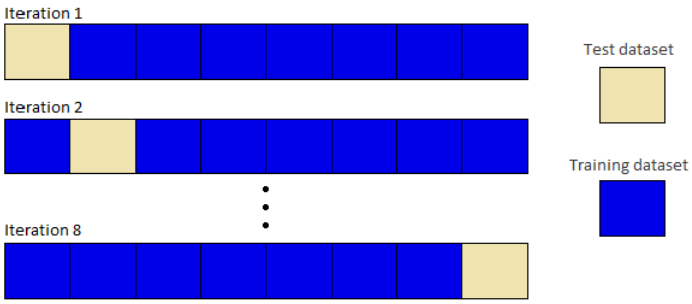


Figure 2.12: The figure showing an instance of k-fold cross valdiation. This is when k=n

class classifier with classes, "postive" and "negative" will have four different outcomes:

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

Those 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.

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

Accuracy Accuracy gives the ratio of correct prediction.

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

Recall Reacall gives the number of detected actual class members.

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

F-score F-score is a balanced measure of recall and precision

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

It will be interesting to see which of class was easier to predict than others. This can be seen by using confusion matrix. The confusion matrix consists a square matrix with one row for predicted class, and and column for actual class (or vice-versa). An example of confusion matrix is shown on table 2.1

		Actual		
		C_1	C_2	C_3
Predicted	C_1	8	0	0
	C_2	5	8	1
	C_3	5	3	8

Table 2.1: A confusion matrix of 3-class classification.

Chapter 3

Implementation

This chapter intends to explain the various choices which has been made to create machine learning models.

3.1 Implementation environment and language

The robot can be controlled by Robot Operating System (ROS), which is a software component framework developed for robot applications. There are many choices of chosing an algorithm, but limited to integrate it with ROS. Labraries will be beneficial to use to reduce development time. ROS is most comapatible with Python, C++, Lisp, and MatLab, but there is experimental libraries in Java and Lua. There are many available libraries to different type of language. Python has a library called scikit-learn. It has many different type of supervised- and unsupervised learning. With all of this labrary it will be easy to test different classifier to find out what is the best. C++ is much faster than Python, but does not come with many learning algorithm, and it can be difficult to debug it. MatLab could also be used, it has labraries, so it would not matter wether langauge it was chosen between these two. Gathering data is used c++ and the machine learning part is in python. All the system is integreted. In the ROS implementation the algorithm split into several ROS nodes to allow parallell execution of the data collction and class prediction in real time whereas the python implementation runs sequentially.

Python

Python provide lots of libraries which provide multiple learning algorithms. The libaray's are well-documented and have the freedom to customize each algoithm to use. The library's used in this project to test different classifiers are scikit-learn, numpy, scipy and runstats.

Scikit-learn Scikit-learn is an open source library that provide tools for data mining and data analysis [46]. The library is dependent on Numpy, Scipy and matplotlib. All of classifiers, and preprocessing data tool are from this library.

Runstats Runstats is library is used to compute statistics, such as max, mean, skew, variance and standard deviation [47]

3.2 Robot

The robot used in this work is an Quadruped Robot called "dyret" at the University of Oslo. This is shown in figure 3.1 The OptoForce sensor is attached to each feet on the robot. For simply set the sensor is placed in such way that it has the same direction as y-axis, which will easier to reproduce the experiments if the foot get deattach. As long one of the axis point same direction as the robot, it will be easier to reproduce experiments. Since the sensor is sensitive to x,y and z-direction, it is important that attaching both sensor in front the same way, and the for back also the same. The robot has 4 different gaits which can be used, but only one gait is used in this project. The reason is that the other gait had issues while walking on the soft mat. The robot either got stuck (slow gait) or tipped down(fast gait). The gait is a balanced gait which let the robot go straight forward, how many steps it should walk can be done by changing a parameter in code.

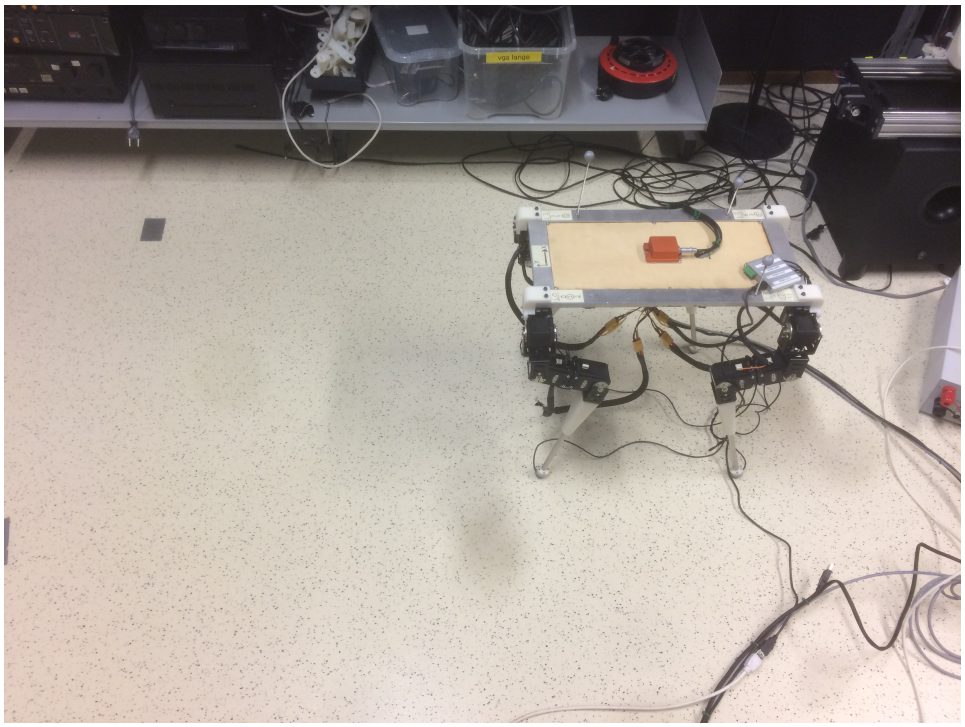
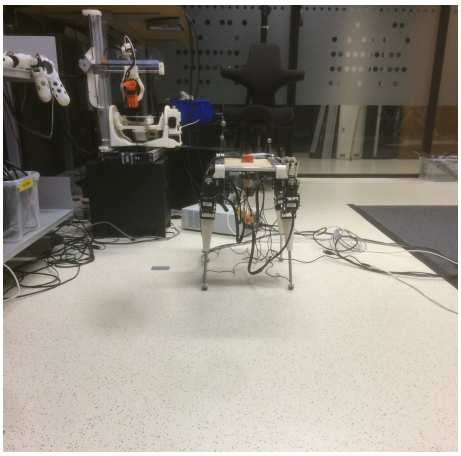


Figure 3.1: Figure showing dyret used in this project

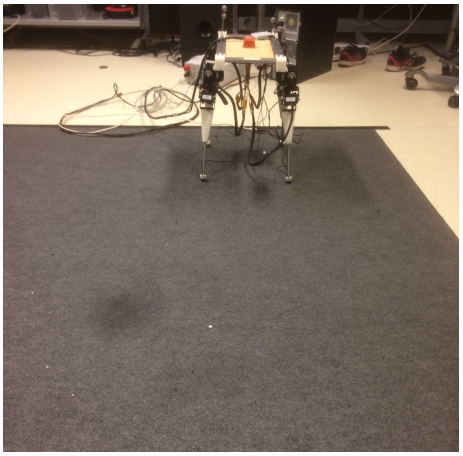
3.3 Terrain

The robot will walk on 4 different terrains, floor, carpet, soft mat, and hard mat. The reason of chosing those terrains is the similarity between the

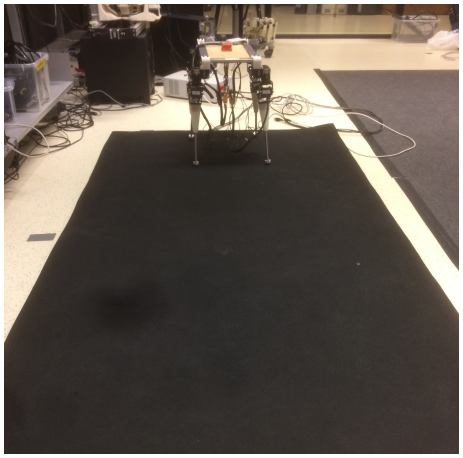
terrain. In this project it will be to see if the classifier can seperate different terrain even with minor differences. Floor and the hard mat is probably the most difficult to distinguish. The carpet also minor difference, but it has more friction than the floor and the hard mat. The last terrain, a soft mat, should be easy to predict since it distinguish by softness and high friction.Different terrains is shown on figure 3.2.



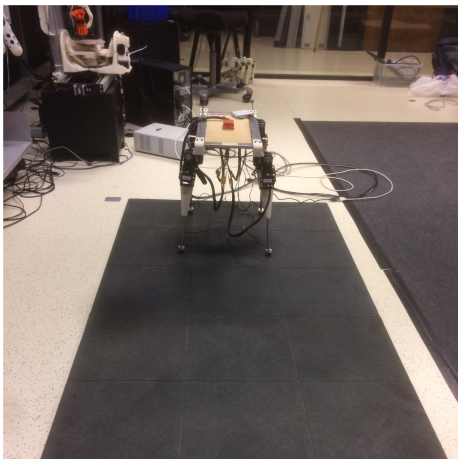
(a) Floor



(b) Carpet



(c) Soft mat



(d) Hard mat

Figure 3.2: Four different terrain used for classification

3.4 Data from optoforce sensor

The data is collected by letting the robot walk 10 step per run. The data is saved into rosbag, which make it possible resimulate the data from earlier test. Integrating the optoforce with ros has an advantage, which is rosbag. The project using this package [48] in order to integrate with ros. For each terrain let the robot walk 10 step. Totalt 50 step is gathered from each for four different terrain. Reading from the sensor is set to default, 100Hz and

filtering frequency, as a string to 15Hz. Since the data arrives as a stream, we window it into discrete chunks when creating the feature vector for the classifier. Larger data windows increase accuracy at the cost of increased latency when detecting change. We chose to use a one second sliding window.

3.5 Segmentation of data

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

3.5.1 Analyzing data from sensor

An important stage is get the gather the interest information from the data set. There are many approach to get data such as sliding window. In this project was to first analyzing how the data looks like and find whetere there are any characterizatic between each terrain. A run for each terrain is shown on 3.3, it is collected by one sensor.

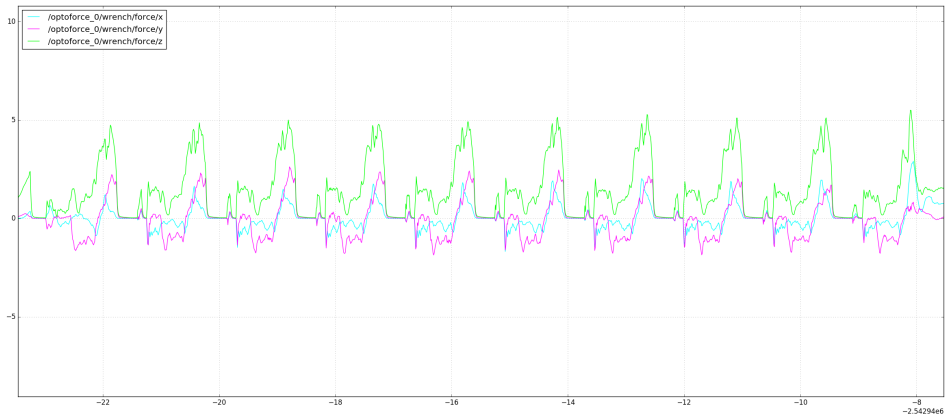


Figure 3.3: Figure showing `rqt_plot` of a testrun on the floor.

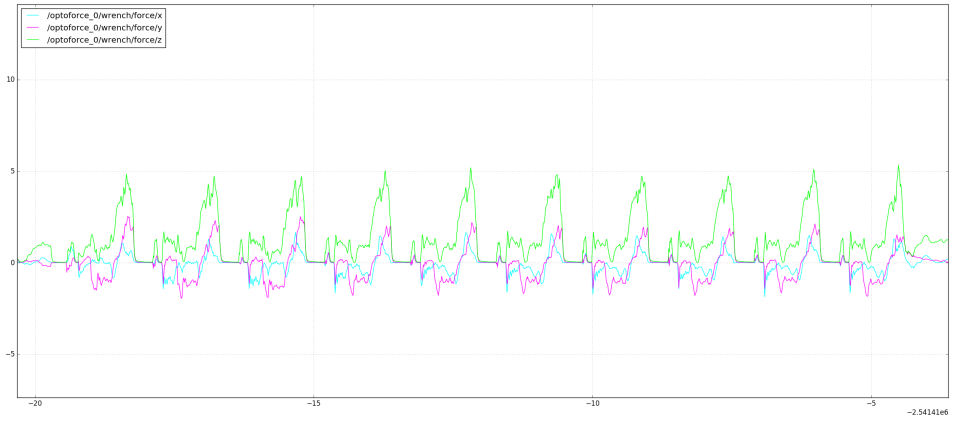


Figure 3.4: Figure showing `rqt_plot` of a testrun on the carpet terrain.

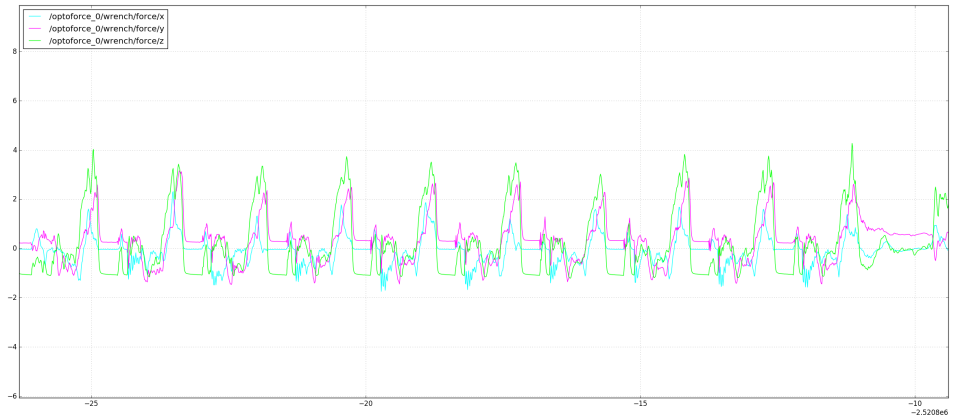


Figure 3.5: Figure showing `rqt_plot` of a testrun on the hard mat terrain.

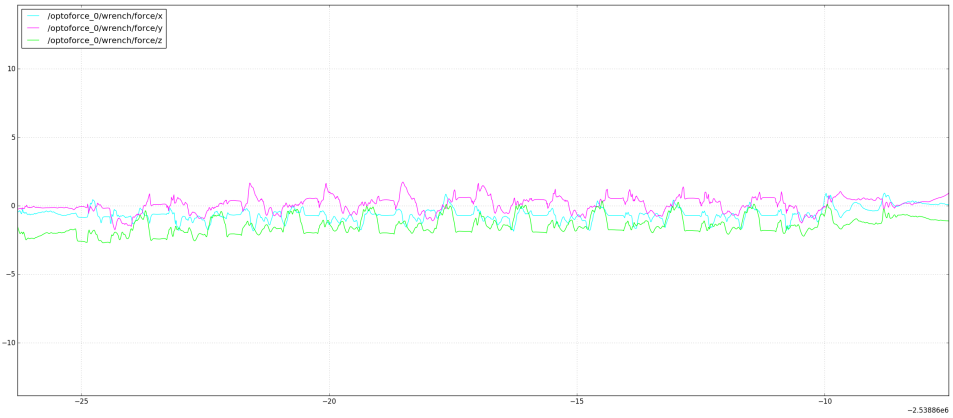


Figure 3.6: Figure showing `rqt_plot` of a testrun on the soft mat.

The first of the robot is get ready to walk, while we can see an almost periodic sequence when the robot walk. As a sensor is in the air, the x,y, z-direction will have minor change almost constant. The idea of getting the interested data is to detect when a foot is in air. That is when there are minior change in x,y,z-direction. When there are big change in each direction, start to save the data from sensor, and stop when it is in the air again.

To see how each terrain will differ to each other. A mean of ten steps for each terrain and each direction is shown 3.7.

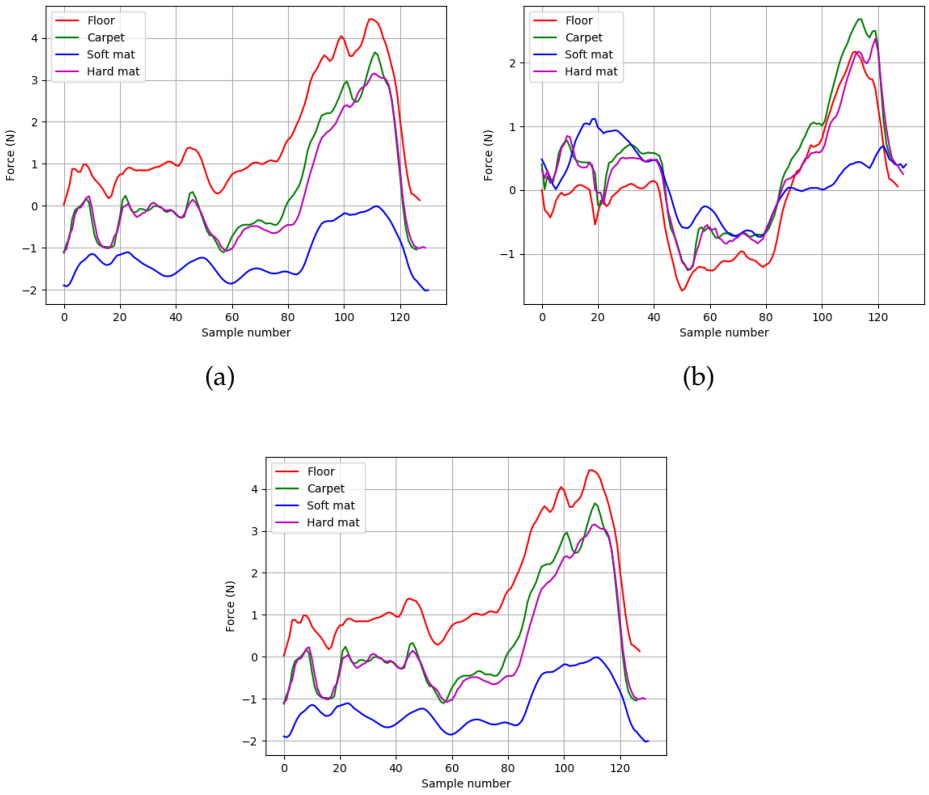
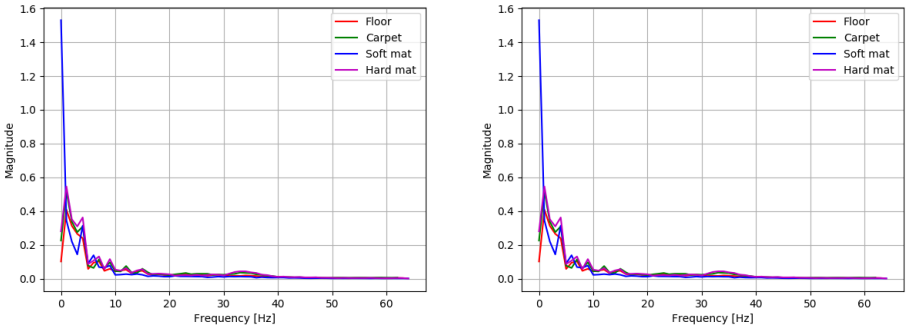


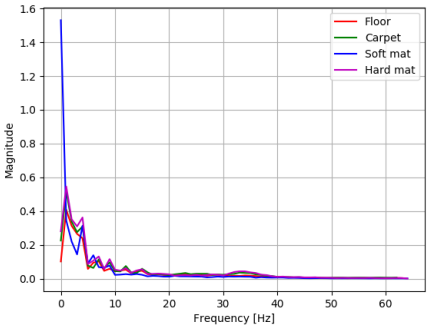
Figure 3.7: Figur showing the mean

Note that the soft mat is the one which differ from all of the terrains. The challenging will be to predict the hard mat and the carpet since the data is to close to each other. But it has a similar shape of graph as floor, which also could make it difficult to distinguish. Many researcher like to see the fequency domain. The figure 3.8 showing a mean of ten step in fft.



(a)

(b)



(c)

Figure 3.8: FFT of

3.6 Feature extraction

As mentioned in section 2.4, a good classifier is depended of good features. In the section 2.4.2 previous work have extracted features both in time domain and frequency domain. In this project, five different feature sets will be created from either time domain, frequency domain or both. The algorithm to extracting each step.

1. Having two global variable, one to say
2. The minimum value of the dataset in time domain
3. Skew in time domain
4. Kuortosis in time domain
5. Standard deviation in time domain

Time domain

Feature set one - raw data This feature set use all of the data from each step in x,y and z-direction. The raw data is decimated in order to acheieve

a constant length, in this case 125, and compressed into one feature vector. The feature vector can be seen in 3.1.

$$f_{set1} = \{x_1, \ldots, x_{125}, y_1, \ldots, y_{125}, z_1, \ldots, z_{125}\} \tag{3.1}$$

The feature contain 125 feature from each 3 sensors, which mean 375 features.

Feature set two - statistical features This feature set extracting statistical feature from the dataset. Earlie work is it well used. This is using some of the features from this article [44]. The features from this set will be for each direction x,y and z:

1. The maximum value of the dataset in time domain
2. The minimum value of the dataset in time domain
3. Skew in time domain
4. Kuortosis in time domain
5. Standard deviation in time domain

The feature set will be look like:

$$\begin{aligned} f_{set2} = \{ & x_{max}, x_{min}, x_{skew}, x_{kuortosis}, x_{std}, x_{var}, x_{trapz} \\ & y_{max}, y_{min}, y_{skew}, y_{kuortosis}, y_{std}, y_{var}, y_{trapz} \\ & z_{max}, z_{min}, z_{skew}, z_{kuortosis}, z_{std}, z_{var}, z_{trapz} \} \end{aligned} \tag{3.2}$$

The feature contain 7 features from each direction, which mean total 21 features.

Frequency domain

Feature set three - complete frequency spectrum Previous work has shown that using frequency has shown good result. In this feature set, the raw data from time domain is transformed into the frequency domain by fft. After tranformation a decimation need to be used, to have a fixed length.

$$f_{set3} = \{fx_1, \ldots, fx_{61}, fy_1, \ldots, fy_{61}, fz_1, \ldots, fz_{61}\} \tag{3.3}$$

Feature set four - staticial features This feature set compute the frequency domain of each direction of the raw data. The features mostly same as feature set two and additionally the energy of the spectrum.

1. The maximum value of the dataset in fequency domain
2. The minimum value of the dataset in fequency domain

3. Skew in fequency domain
4. Kuortosis in fequency domain
5. Standard deviation in fequency domain
6. Energy

The feature which will be feed into classifier:

$$\begin{aligned} f_{set3} = \{&fx_{max}, fx_{min}, fx_{skew}, fx_{kuortosis}, fx_{std}, fx_{var}, fx_E \\ &fy_{max}, fy_{min}, fy_{skew}, fy_{kuortosis}, fy_{std}, fy_{var}, fy_E \\ &fz_{max}, fz_{min}, fz_{skew}, fz_{kuortosis}, fz_{std}, fz_{var}, fz_E\} \end{aligned} \tag{3.4}$$

Feature set five - set two and four Good feature might come from different feature sets. The features from set two and four will be collected as a one set.

3.6.1 Tuning Parameters

The parameter in this project is default values.

3.7 Learning approach

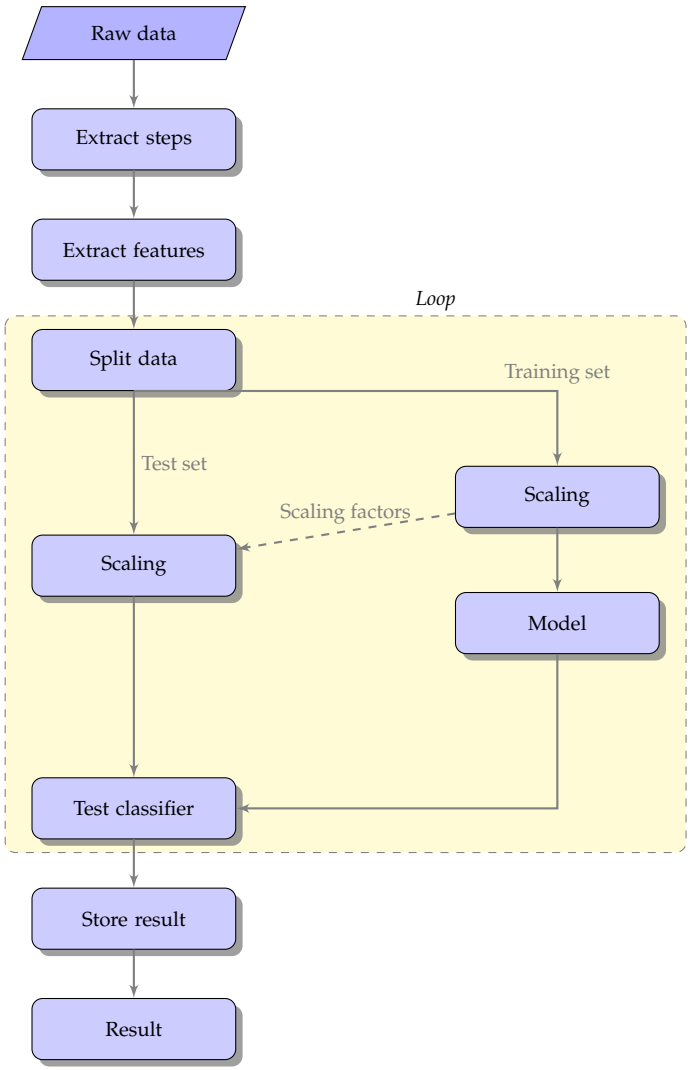


Figure 3.9: The figure showing steps

Details of the learning process

1. **Raw data**
   The dataset as it were when received/obtained from the optoforce sensor

2. **Extract steps**
   This step will extract the interested data from the raw data as described and reasoned in section

3. **Extract features**
   Extract features This step will create one of the six feature sets mentioned in section 3.3.

4. **Loop**

   (a) **Split data**

Split data set into a training set and testing set.

Train and test split Here, the dataset will be split into a training set and testing set as described in section 3.4. That is, the heartbeats from one animal that has not previously used as a testing test in the current realization, will be used as such, and the rest will be used in the training set.

(b) **Scaling**
This step takes the training set as input and standarazation as mention in section 2.4.4. It will first scale the training data, and then scaling factors are applied to the test data.

Scale first the training set and then transform the testing set.

(c) **Train the classifier**
The classifier will take the training set as input and train.

(d) **Test classifier** This step use the model to predict the test set. It will store the score. If there are still more data that need to be predicted start from top again.

5. **Result**

Compute the average results as mentioned in section 2.5.4

Chapter 4

Experiments and results

The chapter present the results of all model in previous chapter. The two-class classification will be investiated first. Further will be to using methods to improve each of classifier. The best classifier will be used to test.

4.1 Results of classifier

Table 4.1 shows the result when using the learning approach. The accuracy is the average accuracy. The experiments were done twice, both with and without the feature scaling.

Feature set	Classifier	Accuracy	
		Not scaled	Scaled
Set one - raw data	Bayes Navies	0.83	0.83
	Decision tree	0.94	0.96
	KNN	0.89	0.91
	Neural network	0.95	0.96
	SVM	0.89	0.93
Set two - statistical features in time domain	Bayes Navies	0.82	0.82
	Decision tree	0.83	0.84
	KNN	0.84	0.84
	Neural network	0.83	0.88
	SVM	0.76	0.85
Set three - complete frequency domain	Bayes Navies	0.91	0.91
	Decision tree	0.83	0.82
	KNN	0.91	0.88
	Neural network	0.93	0.93
	SVM	0.52	0.93
Set four - staticial features in frequency domain	Bayes Navies	0.81	0.81
	Decision tree	0.80	0.82
	KNN	0.88	0.84
	Neural network	0.85	0.88
	SVM	0.83	0.86
Set five - set 2,4	Bayes Navies	0.82	0.82
	Decision tree	0.79	0.80
	KNN	0.88	0.84
	Neural network	0.87	0.89
	SVM	0.83	0.86

Table 4.1: My caption

4.1.1 Analysis

The result shown that all classifier has a accuracy at least 70%. The Decision tree and Neural Network has the highest accuracy with 96% from scaled feature set one.

Regarding the feature scaling, SVM classifer is the one with huge affect of the scaled. The KNN which is also distances is also affected by scaling, but most of them gave lesser accuracy. While Neural Network, Decision tree and Bayes Navies do not rely on distance between features, only Bayes Navies was not affected. The Neural Network and Decision Tree had minor change in their accuracy. Since the feature scaling had great affect on the SVM and minor on the other classifier, scaled features will be used in further experiments.

Note that the mostly highlig result is from feature set one. The benefit of using raw data is ensuring that all the information is in the data. The more data might be easier to recognize a pattern or find characteristic for each terrain. Conversely, by extraction features might loose important features, hence to not high accuracy.

Even there are two classifier which outpeform the other, there are still a improving potensial to all of them. The improvement will be investigated in next section.

4.2 Further improving all classifier

The classifier approach state earlier has leaved the feature selection out. The reason was to see how well each classifier predicted terrain based on all features. It has shown in section 4.1 that it gave high accuracy for most of them. As stated in section 2.4.3, selecting good features has benefit of increasing classifier perfomance. In this approach feature selection is implemented into the earlier approach. Two different feature selection methods is tested, filter and wrapper.

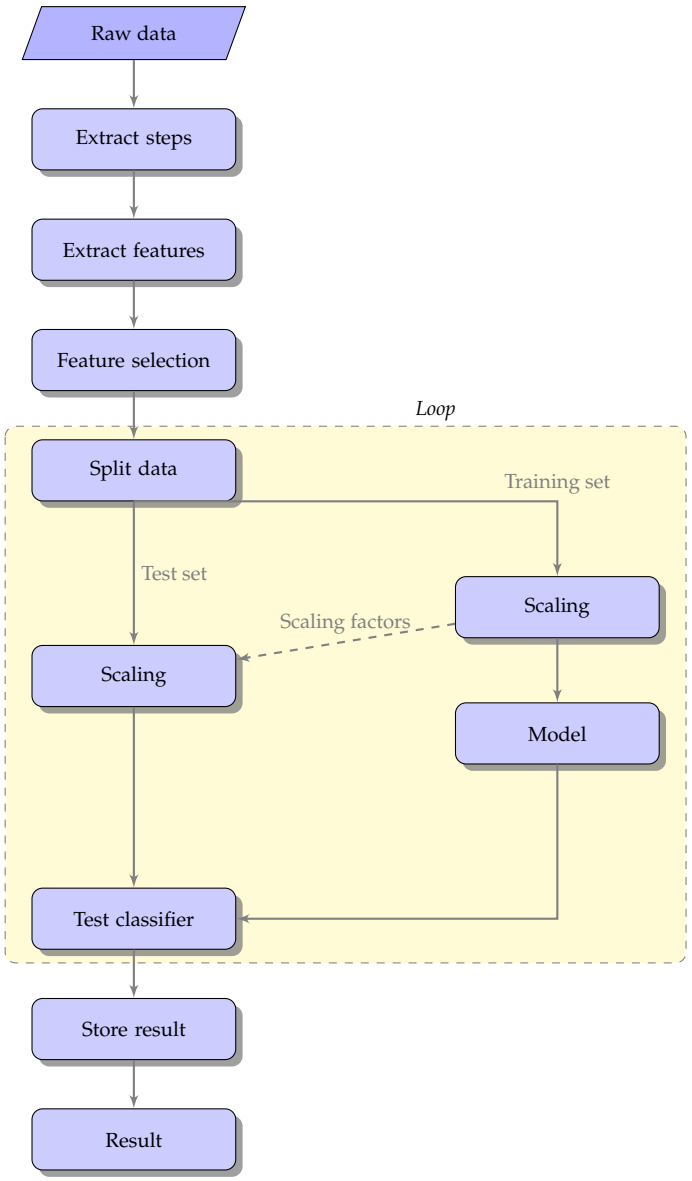


Figure 4.1: The figure showing steps

4.3 Experiment 2 - classifier improvement

Table 4.4 shows the result when using the learning approach. The accuracy is the average accuracy. The experiments were done twice, both with filter and wrapper method.

Feature set	Classifier	Accuracy		
		Old result	Filter	Wrapper
Set one - raw data	Bayes Navies	0.83	0.78	0.81
	Decision tree	0.96	0.96	0.93
	KNN	0.91	0.95	0.93
	Neural network	0.96	0.94	0.95
	SVM	0.93	0.91	0.94
Set two - statistical features in time domain	Bayes Navies	0.82	0.74	0.84
	Decision tree	0.84	0.79	0.81
	KNN	0.84	0.82	0.82
	Neural network	0.88	0.81	0.84
	SVM	0.85	0.80	0.83
Set three - complete frequency domain	Bayes Navies	0.91	0.76	0.94
	Decision tree	0.82	0.76	0.81
	KNN	0.88	0.83	0.91
	Neural network	0.93	0.78	0.95
	SVM	0.93	0.79	0.97
Set four - staticial features in frequency domain	Bayes Navies	0.81	0.75	0.79
	Decision tree	0.82	0.82	0.81
	KNN	0.84	0.85	0.86
	Neural network	0.88	0.84	0.90
	SVM	0.86	0.82	0.86
Set five - set 2,4	Bayes Navies	0.82	0.74	0.84
	Decision tree	0.80	0.82	0.84
	KNN	0.84	0.86	0.83
	Neural network	0.89	0.81	0.89
	SVM	0.86	0.83	0.86

Table 4.2: My caption

4.4 Analysis

From table one can see that the filter selection has made the classifier more inaccurate. It seems that important features were removed. As in section. that one feature may not be important by itself, but might be important to others. The wrapper selection, on the other hand, provide a significant better result compared to the filter. It also has improved the SVM to 97% accuracy.

Note that the k-fold validation test using always use the same single set both for training and testing, which make it reffering to themselve than to generalize model of particular terrain class. In this case results from 4.4 only gives an estimation on how well a classifier is, but does not tell how well it predict on unseen data. One way to avoid this issue is make the training set and testing set independed to each other, then the predicition will predict on new data in the k-fold [49]. The following experiment only four of the highest classifier shown in 4.4 would be investigated further. As there are 3 second best classifier, all of them will be in further experiments. This is divided into further experiments.

4.5 Cross validation with unseen data

In this experiment 50 new samlpes for each terrain is collected. This is divided into further experiments to comparing the two best classifier.

1. SVM with wrapper and use feature set two
2. Neural Network and use feature set one.
3. Decision tree with feature set one
4. Decision tree with filter and use feature set one

Note the training set and test set will be randomly shuffled, which might cause an inaccurate measurement of classifier. To achieve more accuracy measure of classifier, each test will be runned five times.

Run	Accuracy	Precision	Recall	Confusion matrix			
				Floor	Carpet	Soft mat	Hard mat
				50	0	0	0
1	0.97	p1	r1	2	48	0	0
				0	0	50	0
				2	1	0	47
				50	0	0	0
2	0.97	p2	r2	2	48	0	0
				0	0	50	0
				2	1	0	47
				50	0	0	0
3	0.97	p3	r3	2	48	0	0
				0	0	50	0
				2	1	0	47
				50	0	0	0
4	0.97	p4	r4	2	48	0	0
				0	0	50	0
				2	1	0	47
				50	0	0	0
5	0.97	p5	r5	2	48	0	0
				0	0	50	0
				2	1	0	47

Table 4.3: Dette er SVM wrapper feature set 2 fft

SVM - Wrapper - Feature set two

On the table 4.3 showing accuracy, precision, recall and confusion matrix for each run. Results of SVM has not been change during each run. That indicate that the good features is kept and is able to distinguish between each terrain. There is shown a small confusion between floor, carpet and hard mat, which are three with similar terrain characteristic. While the soft mat has an own characteristic and easy to detect.

Run	Accuracy	Precision	Recall	Confusion matrix			
				Floor	Carpet	Soft mat	Hard mat
1	0.68	P1	R1	25	25	0	0
				1	49	0	0
				0	0	50	0
				11	28	0	11
2	0.68	P2	R2	26	24	0	0
				1	48	0	1
				0	0	50	0
				9	30	0	11
3	0.64	P3	R3	20	30	0	0
				0	48	0	2
				0	0	50	0
				11	29	0	10
4	0.68	P4	R4	27	23	0	0
				0	49	0	1
				0	0	50	0
				9	32	0	9
5	0.69	P5	R5	26	24	0	0
				0	49	0	1
				0	0	50	0
				10	28	0	12

Table 4.4: Neural Network - Feature set 1 without selection

Neuarl Network - Feature set one

On the table 4.4 showing accuracy, precision, recall and confusion matrix for the neural network. Comparing to the old result from 4.4, the classifier perform more poorly on unseen data. One of the reason can be that the Neural Network is overfitted, and is not able generalize good data. The feature set is use is also raw data, which might cause learning with noises. The classifier has particulary difficulty of distinguish hard mat with floor and carpet. It is also shown that it has difficulty of chosing terrain between floor and carpet. While it has no problem of predicting the soft mat.

Run	Accuracy	Precision	Recall	Confusion matrix			
				Floor	Carpet	Soft mat	Hard mat
				43	5	2	0
1	0.72	P1	R1	4	41	2	3
				0	0	46	4
				8	26	2	14
				45	4	1	0
2	0.73	P2	R2	5	38	2	5
				0	0	47	3
				8	24	1	17
				44	3	3	0
3	0.75	P3	R3	5	41	0	4
				0	0	47	3
				7	26	0	17
				44	2	4	0
4	0.72	P4	R4	6	41	0	3
				0	0	46	4
				8	27	1	14
				48	0	2	0
5	0.77	P5	R5	5	39	2	4
				0	0	49	1
				8	21	3	18

Table 4.5: Decision tree feature set 1 without filter

Decision Tree - Feature set one

On the table 4.5 showing accuracy, precision, recall and confusion matrix for the neural network. Comparing to the old result from 4.4, the classifier perform more poorly on unseen data. One of the reason can be that the Decision tree is overfitted, and is not able generalize good data. It does predict better than the Neural Network. The Neural Network has more complicated learning process, where the weights is adjusted according to the error. The amount of the hidden layer and hidden nodes must also be adjusted. Another reason which may cause poor performance is that the parameters are all on defaults. The feature set is use is also raw data, which might cause learning with noises. The classifier has particulary difficulty of distinguish hard mat with floor and carpet. It is also shown that it has difficulty of chosing terrain between floor and carpet. While it has no problem of predicting the soft mat.

Run	Accuracy	Precision	Recall	Confusion matrix			
				Floor	Carpet	Soft mat	Hard mat
1	0.73	P1	R1	47	1	2	0
				5	39	1	5
				0	3	44	3
				7	26	1	16
2	0.75	P2	R2	47	3	0	0
				6	41	0	3
				0	2	47	1
				8	28	0	14
3	0.75	P3	R3	44	3	3	0
				5	41	0	4
				0	0	47	3
				7	26	0	17
4	0.73	P4	R4	45	4	0	3
				6	41	0	3
				0	3	45	2
				8	27	0	15
5	0.72	P5	R5	45	2	3	0
				6	40	0	4
				0	1	47	2
				6	32	0	12

Table 4.6: decision tree featureset 1 filter

Decision Tree - Filter - Feature set one

On the table 4.6 showing accuracy, precision, recall and confusion matrix for the neural network. Comparing to the old result from 4.4, the classifier perform more poorly on unseen data. One of the reason can be that the Decision tree is overfitted, and is not able generalize good data. Another reason which may cause poor performance is that the parameters are all on defaults. The result is similar to the earlier, but this has a more stable accuracy. It vary has a variation on 2% while the earlier is 5%. The classifier have particulary issue to distinguish between hard mat and carpet.

4.6 Classification in real-time

To get a more realistic practical scenario is based on evaluation of the classification, where the robot traversing in different terrains. For this experiment, the robot will be walking on two different terrains. There are total five different setup. The uequal length of each steps is that the robot tend to walk out off current terrain, thus stops the experiments.

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

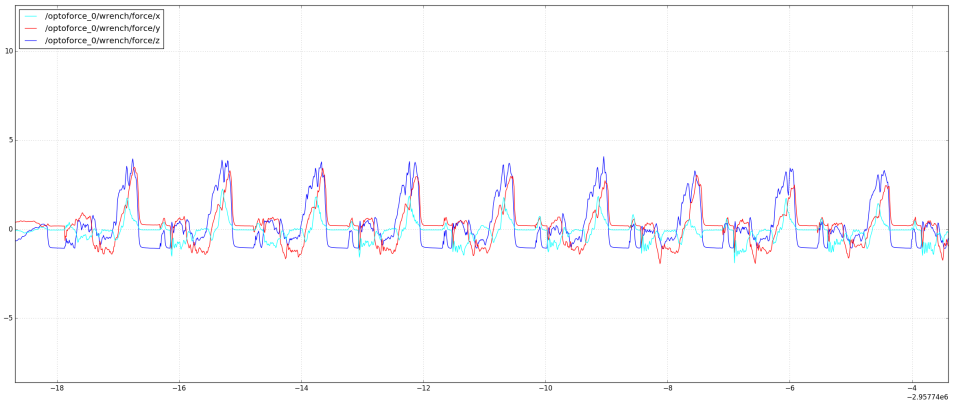


Figure 4.2: Figure showing an the optofoce used in this thesis [1]

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 4.7: Gulvet4teppe

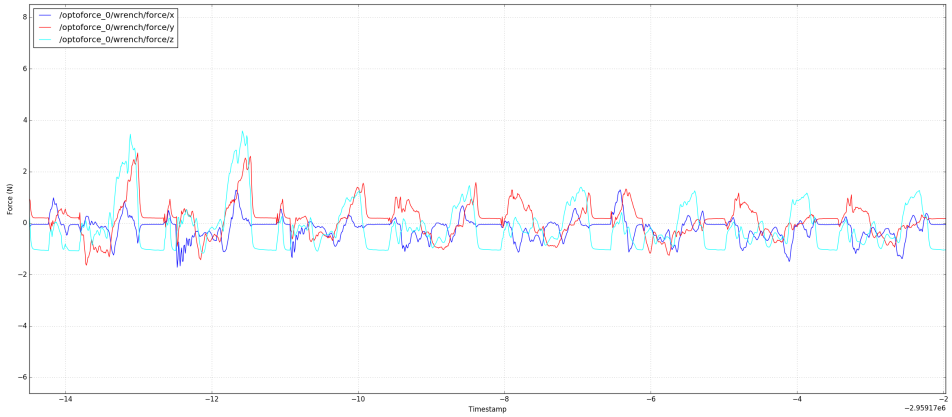


Figure 4.3: Figure showing an the optofoce used in this thesis [1]

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 4.8: Hard matte 3 myk matte

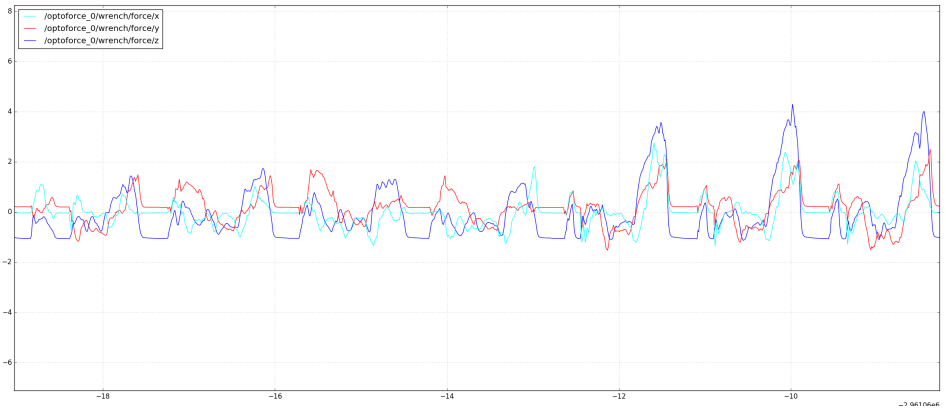


Figure 4.4: Figure showing an the optoforce used in this thesis [1]

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 4.9: MM 4Resten MBNY

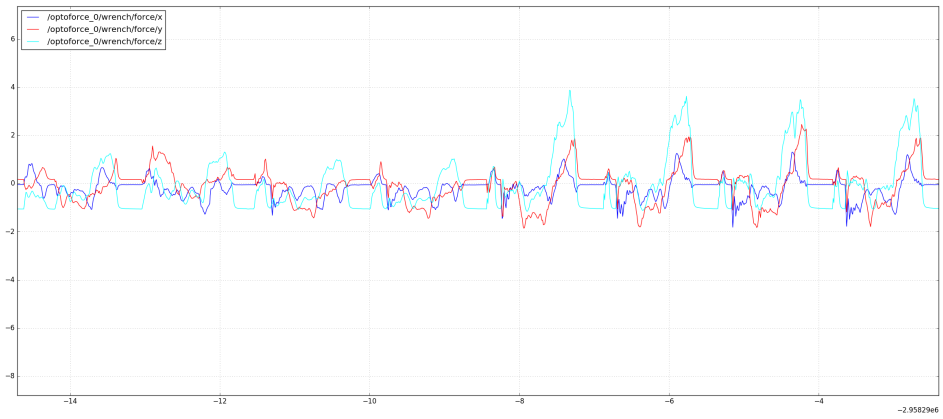


Figure 4.5: Figure showing an the optofoce used in this thesis [1]

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 4.10: MM4 Teppe

4.7 Prediction next sensor?

4.8 Prediction different length of legs?

Chapter 5

Conclusion

5.0.1 Furture work

Improvment of classifier The classifier is based on the features of the robot motion and interaction with the terrain. However, the features of the terrain itself such as slopiness, slipperiness, convexity, hardness, are not analyzed directly - they may be hidden inside the classifier and could be addressed in the future work.

Other classifier, other features extraction.

The learning is based on each step, maybe collect data from more than one step.

Rely only on optoforce sensor, other sensor such as servo, knee etc....

Teste om the holder å se på en retning? Oter experiments To og to bein på forskjellige underlag

http://www2.ift.ulaval.ca/ pgiguere/terrainID.html

Bibliography

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

[2] "Omd-20-se-40n-datasheet v2.2." Available at https://optoforce.com/file-contents/OMD-20-SE-40N-DATASHEET-V2.2.pdf. Accessed: 2017-03-29.

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

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

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

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

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

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] A. 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.

[22] 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.

[23] "Optical force sensors - introduction to the technology." Available at https://optoforce.com/file-contents/optoforcewhitepaperopticalforcesensors-0.pdf. Accessed: 2017-03-22.

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

[25] S. Caccamo, P. Güler, H. Kjellström, and D. Kragic, "Active perception and modeling of deformable surfaces using gaussian processes and position-based dynamics," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pp. 530–537, Nov 2016.

[26] S. Caccamo, Y. Bekiroglu, C. H. Ek, and D. Kragic, "Active exploration using gaussian random fields and gaussian process implicit surfaces," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 582–589, Oct 2016.

[27] A. Burka, S. Hu, S. Helgeson, S. Krishnan, Y. Gao, L. A. Hendricks, T. Darrell, and K. J. Kuchenbecker, "Proton: A visuo-haptic data acquisition system for robotic learning of surface properties," in *2016 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pp. 58–65, Sept 2016.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] D. Tick, T. Rahman, C. Busso, and N. Gans, "Indoor robotic terrain classification via angular velocity based hierarchical classifier selection," in *2012 IEEE International Conference on Robotics and Automation*, pp. 3594–3600, May 2012.

[35] W. Mou and A. Kleiner, "Online learning terrain classification for adaptive velocity control," in *2010 IEEE Safety Security and Rescue Robotics*, pp. 1–7, July 2010.

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

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

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

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

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

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

[49] G. Best, P. Moghadam, N. Kottege, and L. Kleeman, "Terrain classification using a hexapod robot," in *proceedings of the Australasian Conference on Robotics and Automation(ACRA '13)*, 2013.