Terrain Type Classification Based on Sound

A Master Thesis Report

Submitted by

Abraham Gebru Tesfay

July 8, 2015

Supervisor

Dr.-Ing. Ivana Kruijff-Korbayova

Advisor

Dipl.-Inf. Bernd Kiefer

Reviewers

Dr.-Ing. Ivana Kruijff-Korbayova
PD Dr. Helmut Horacek

Saarland University
DFKI Language Technology Lab
Department of Computer Science

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Ich erkläre hiermit an Eides Statt, dass die vorliegende Arbeit mit der elektronischen Version übereinstimmt.

Statement in Lieu of an Oath

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis. I hereby confirm the congruence of the contents of the printed data and the electronic version of the thesis.

Saarbrücken, on July 8, 2015

Abraham Gebru Tesfay

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlichtwird.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken, on July 8, 2015

Abraham Gebru Tesfay

Abstract

The autonomous mobility of mobile robots has great contribution to human beings exploring hazardous terrains. The motivation of this thesis is to detect different types of terrains traversed by a robot based on acoustic data from the robot-terrain interaction thereby helping to make the mobile robots more autonomous. The acoustic data was collected using a microphone mounted on our robot. Then, these recorded datasets were used to train classifiers so as to distinguish different terrain types from one another. Different acoustic features and classifiers were investigated, such as Mel-frequency cepstral coefficient and Gamma-tone frequency cepstral coefficient for the feature extraction, and Gaussian mixture model and Feed forward neural network for the classification. We analyze the system's performance by comparing our proposed techniques with some other features surveyed from distinct related works. Thus, we demonstrate the effectiveness of our approach using five different terrain classes which are trained using real data sets gathered from different ground surfaces. The experimental result indicates the average accuracy obtained is approximately 93.6% and it is enhanced to 95.2% with an increase in audio duration. In real applications, it is better to decrease the detection time and our system still has satisfactory performance using human-like terrain labeling even for smaller audio duration. These are very promising results which show that acoustics is an interesting domain that needs to be extensively explored to improve the autonomy of tracked robots.

Acknowledgements

First and foremost, I would like to express my heartfelt gratitude to my supervisor Dr. ing. Ivana Kruijff-Korbayová for providing me an opportunity to accomplish my thesis at DFKI. Her constant encouragement, willingness to help, immense knowledge and excellent supervision have been such an enormous motivation for me. Furthermore, I wouldn't have envisioned having a better supervisor for my master's thesis.

My sincere thanks to Bernd kiefer, who has been such a brilliant advisor, for his invaluable guidance. During our periodic discussions, your constructive suggestions have helped me to gain such a wonderful experience on doing scientific research. Besides, I have learned a lot while working with you.

Then, I would like to thank PD Dr. Helmut Horacek for becoming the reviewer of my thesis. My profound appreciation goes to all members of Cosy lab for creating enthusiastic atmosphere and being very sociable.

Furthermore, I would like to thank my immediate family, specifically my father, mother, brothers, sister and my girl friend for your love, encouragement, patience, endless motivation and support. Last but not least, my wholehearted gratitude to my friends who confronted me in tough times and putting a smile on my face. Thank you to almighty God and everyone for your help in realizing my dream come true.

Contents

Abbreviations

ADC Analog to Digital Converters
ALSA Advance Sound Linux Architecture
DCT Discrete Cosine Transform
DFT Discrete Fourier Transform
EM Expectation-Maximization
ERB Equivalent Rectangular Bandwidths
FFNN Feed Forward Neural Network
FFT Fast Fourier Transform
FN False Negative
FP False Positive
GFCC Gammatone Frequency Cepstral Coefficients
GMM Gaussian Mixture Model
GPS Global Positioning System
GTF Gammatone Filter
GUI Graphical User Interface
IMU Inertial Measurement Unit
INS Inertial Navigation System
LED Light Emitting Diode
MAP Maximum A Posterior Probability
MFCC Mel-Frequency Cepstral Coefficients
MLE Maximum Likelihood Estimate
MSE Mean Square Error
OSS Open Sound System
PCB Printed Circuit Board
PoE Power over Ethernet
ROC Receiver Operating Characteristics
SFM Spectral Flatness Measure
SVM Support Vector Machine
TN True Negatives
TP True Positive
TRADR Long-Term Human-Robot Teaming for Robot Assisted Disaster Response
UGV Unmanned Ground Vehicle
ZCR Zero-Crossing Rate

Chapter 1

Introduction

This chapter initially explains the general introduction to mobile ground robots. Then, it presents motivation of the thesis, particular problem addressed and outline of the thesis. Finally, the chapter summarizes the state-of-the-art in terrain type classification.

1.1 Background in mobile ground robots

The autonomous mobility potential of mobile ground robots (rovers) provides a tremendous contribution for humans to explore the surface of other planets or other hazardous areas. The drawback of close teleoperational supervision of robots is the remarkable constraint on the distance traversed by a robot due to a limited communication with operators on Earth. Thus, improvements in robot autonomy will help mobile robots to travel longer distances with restricted human control, which in return results to a scientific data from locations that were previously unaccessible. The inability of current robots to autonomously indicate terrain regions such as unknown hazards, creates a significant problem on the successful realization of a mission. In general, autonomous navigation is limited to benign environments previously determined by human operators. Furthermore, the capacity to detect possible dangers from a distance would pave a way to safe autonomous travel.

The mobile ground robots must be able to handle harsh, complex and unpredictable terrain conditions. The ability of these robots to detect and avoid dangerous situations depends on how much precise, accurate and reliable perception system they have. In addition, an increase in accuracy can occur by the fusion of data from different sensors, so it is of great significance for scholars in robotics to proceed searching the possible sensor choices. Mobile robots can use proprioceptive sensors such as global positioning system (GPS), inertial navigation system (INS) and compass to measure values internal to the system. On the other side, exteroceptive sensors are designed to observe their environment such as contact sensors for measuring interaction forces, range sensors for calculating the distance to objects in their operational area and vision sensors to extract information from images. These sensors might be reliable in some factors

and unreliable in other factors, so it is recommended to use combination of different sensors to get accurate terrain classification. To mention an example, optical sensors are used in scene interpretation to understand the surroundings without the requirement of interaction but they are unreliable due to the variation in appearance. The factors that could change the appearance include variation in lighting conditions, changes in consistency such as dirty water, distortion of water due to wind and misleading appearance of surface vegetation. They could obscure features such as color, intensity, shape and reflective properties coming from camera imagery. In contrast, acoustic sensors are not sensitive to changes in appearance and none of these factors have a great deal of influence on the sound of the vehicle-terrain interactions, but the direct contact with hazardous terrain types may damage the robot. However, interaction can be the only choice when the optical sensors are not working properly. These two sensing modalities might be unreliable due to other factors in different circumstances. Thus, they can be used to adjust each other's mistakes.

This thesis is part of the TRADR project [1] which stands for Long-Term Human-Robot Teaming for Robot-Assisted Disaster Response [1]. TRADR uses the experience gained from the Natural human-robot cooperation in dynamic environments (NIFTi) project [2]. TRADR develops technology for the successful accomplishment of different missions by teams comprising human rescuers and different robots. For this purpose, it applies user centric design approach which involves coordination with end users and the technology. The goals of TRADR are to develop persistent environment models, to improve the experience of team member's on how to operate in disaster areas (persistent multi-robot models) and to enhance the team-work.

The motivation behind this thesis is to detect different types of terrains traversed by the robot based on the acoustic data recorded with a microphone from the robot-terrain interaction. In order to classify the different terrain types, human-like terrain labeling would be useful so that the robot can use human concepts to communicate with a human operator about the terrain. Thus, the robot is able to label the terrain in a way that makes sense to a human. It also gives an information for the operator what kind of terrain the robot traverses. The detection of different terrains could protect our robot from damage which might occur when the robot collides or gets into hazardous terrain types. In addition, it might help our robot to analyze further decisions made based on the detected terrain type. Even though it is beyond the scope of the thesis, the other inspiration is to make mobile robots autonomous. Though the tracked vehicles are capable of accomplishing the rescue operations, a human operator is needed to operate them remotely. Hence, improving the autonomy of these robots is a relevant topic at this time, particularly in the field of mobile robots performing outdoor missions.

The problem addressed in this thesis is terrain type classification based on acoustic data. To solve this problem, our initial approach was to collect data from different ground surfaces. Then, extract the relevant information using different feature extraction techniques and classify the

[1] TRADR is an EU-funded project. URL: www.tradr-project.eu.

different terrains with the help several classifiers. The data collection is accomplished with the help of microphone mounted on our robot. To the best of our knowledge, this is the first time that human-like terrain labeling is used to classify different terrains based on sound using TRADR robot. So part of our contribution is to implement and compare the performance of different feature extraction techniques and classifiers proposed by researchers.

1.2 Outline

This paper is structured as follows: Chapter 1 explains in general the background of mobile ground robots, the motivation for the thesis, the problem addressed and summarizes the state-of-the-art in terrain type classification. Chapter 2 introduces the basics about acoustics and the goals of feature extraction methods in general. Then, we describe in detail some concepts of the feature extraction techniques for Mel frequency cepstral coefficient and Gamma-tone frequency cepstral coefficient employed in our experiment and several other features that have been presented in other publications.

Given the feature vector as data points, chapter 3 presents the different classification schemes such as Gaussian mixture model and Feed forward neural network which can be used to train our system. Thus, the basic principles of the well known training algorithms are thoroughly explained. Chapter 4 initially discusses the basic components of TRADR robot hardware. Then, we explain how to set up our robot step by step for data collection. Not only the external hardware components deployed on our robot but also the software selected to record an audio is discussed. Moreover, the position of microphones in the experiments and the characteristics of the five terrain types are also illustrated in this chapter.

Chapter 5 provides description of the simulation environment and results. we illustrate the parameters used specific to each implementation and elaborate the training algorithms in each classification scheme. Then, we discuss the effectiveness of our system by analyzing its performance measurement parameters considered in each experiments. Hence, we examine the performance of the system by comparing different classifiers and feature extraction techniques introduced in chapter 2 and 3. Finally, chapter 6 provides the conclusion and future work of the thesis.

1.3 State-of-the-art

This section presents related works in acoustics and several other works in terrain classification to visualize mobile robot terrain interactions. Liby and Stentz [3] used acoustic data to classify vehicle terrain interactions that a mobile robot can have with an outdoor environment. The microphones recorded data from different mobile robot terrain interactions to differentiate the benign and hazardous terrains. Then, the acoustic data was hand labeled and used offline to

train a support vector machine (SVM) which is the supervised multi class classifier. Moreover, they have suggested that acoustics is a promising method for further improving the perception system in mobile robots. Brooks and Iagnemma [4] measured the vibration signals via a contact microphone rather than the normal air microphone, which means that it picks the vibrations through solids, to distinguish between beach grass, sand and rock. In addition, they proposed a self-supervised learning framework to predict the properties of distant terrain based on the properties of the previously traversed terrains. Furthermore, they have shown that the labels from the proprioceptive (vibration-based) terrain classifier are useful to train the exteroceptive (vision-based) terrain classifier. Reinstein, Kubelka and Zimmermann [5] classified the terrains based on the values of coefficients correcting the robot's odometry in contrast to a standard robot terrain classification which provides human labeled discrete terrain categories. Besides, they used an inertial measurement unit to extract features from the vibrational data.

Hoiem et. al. [6] proposed a system capable of determining sound objects such as gunshots, dog barks, laughter, sword clashes, laser guns, screams or car horns in complex audio environments. They have utilized boosted decision tree classifiers in order to attain very good performance on many sound instances. Menna et. al. [7] presented a real time 3D path and motion planner for tracked vehicles that tries to alleviate some drawbacks of the current path planners. In three main experiments taken in a training rescue area, on fire escape stairs and in a non-planar testing environment they have proved that the robot is able to autonomously move in almost any kind of structure. Durst and Krotkov [8] classified objects such as a clay brick, a concrete brick, a ceramic tile, a wooden block and a zinc ingot based on the sound that resulted when the object was struck with a cane. To classify the different objects, the most important two spikes which are in the frequency domain were used as features. Amsellem and Soldea [9] proposed a classifier based on the fusion of audio and visual modalities. The classification of the first modality is based on the sound of a cane striking against the resonant object and the visual modality performs classification of objects captured in 3D images.

Collins and Coyle [10] compared several classifiers on the problem of vibration based terrain classification by including multiple vibration measurements recorded using an inertial measurement unit, to differentiate different terrain types such as sand, clay, grass, gravel and asphalt. To alleviate the vehicle mobility problem in these ground types, their vehicle control system can be tuned for different conditions so that they are able to adjust to different terrains. Alex et. al. [11] presented an event detection approach based on Mel and Gammatone frequency cepstral coefficients as inputs for a bag of features. They come up with different competitive outputs using a Gaussian mixture model on three different databases namely the ClEAR sound event dataset, the D-CASE event dataset and the smart room recording datasets.

Wellington and Stentz [12] implemented an online adaptive method on an autonomous farm vehicle to automatically estimate the true ground height in vegetation and tested the system in farm land. Scanlon [13] used a single acoustic sensor in contact with a patient's thorax or neck to continuously monitor the sound and collected information related to the function of

heart, lungs and digestive tract. This helped to assess, diagnose and treat the cardiac and respiratory functions for health status monitoring. Cohn et. al. [14] used a low cost contact microphone on a home gas regulator for controlling gas usage in its home appliances such as water heater, furnace and fireplace. They have shown that their approach have the capability to be easily and safely installed without any guidance from a professional. A contact microphone is also known as transducer, pickup or piezo, and similar to an accelerometer. It is designed specifically to sense audio vibrations in solid objects. They are more similar to accelerometers than to air microphones because the normal microphones are used for detecting vibrations in the air while contact microphones are designed with the capability to pick up surface vibrations. Thus, contact microphones should be stuck to the body of the instrument to sense vibrations directly from the device.

In this section we looked in to several body of work in acoustics and other terrain classification domains. They used different variety of sensors for data collections in their corresponding applications. Similar to the work done by Liby and Stentz [3], we used acoustics data only to distinguish robot terrain interaction. However, the data collection methods, the robot, the feature extraction techniques and the classification schemes used are different. In chapter 2, we will discuss the different kinds of feature extraction techniques employed to extract relevant information from an acoustic data.

Chapter 2

Audio Features

The next task after having recorded audio samples for our database from different terrain types is to extract relevant information viz important features, which can be helpful in classification. We will start with a brief introduction into the acoustics and the goals of feature extraction methods in general. Then, we describe the feature extraction techniques for Mel frequency cepstral coefficient and Gamma-tone frequency cepstral coefficient, and in section 2.4 several other features that have been proposed in the literature. The aim of this chapter is to illustrate in detail basic concepts behind the most common feature extraction techniques which are implemented in different applications and at the same time briefly discuss other features suggested in other publications. The proposed features are also able to imitate the behavior of human ear.

2.1 Introduction to acoustics

Nowadays the applicability of acoustic signals ranging from analyzing speech to recording earthquake, tracking submarines, audio and noise control, to medical diagnostic applications initiate a huge interest in the research community. Acoustic waves are mechanical waves involving some disturbances or vibrations in gases, liquids, and solids. These vibrations have the capacity to stimulate the human ear and to create a sound sensation in the brain. The three different sections of an acoustic spectrum are audio, ultrasonic, and infrasonic. The range of frequencies which is audible to human ear is in the range of 20 Hz to 20KHz and it is more favorable when it is between 1KHz to 4KHz. Medical applications rely on frequencies higher than 20KHz known as ultrasonic frequency ranges while applications such as estimating earthquakes depend on lower frequencies mentioned as infrasonic ranges.

The goal of every feature extraction technique is to capture essential information using a particular reduced representation of data. This has a great contribution to enhance the effectiveness and efficiency of classification and can be accomplished through different methods. The first technique is to eliminate redundancy and to remove irrelevant information from the audio data. Redundancy refers to the part of a signal which is not required to be transmitted so as to

completely recover an original signal at the receiving end. To mention an example, using the same background image in a video is redundant information. On the other hand, irrelevancy indicates to the part of signal that has no meaningful information to a human being at the receiver side. For instance, sound signals having frequencies above 20 kHz are not audible by humans so it is irrelevant transmitting those signals. At the same time it is good to eliminate variability from the data that has no relevant value in classification. The other method is to organize or restructure the data so as to gain performance improvement of the classifier. The feature vector should also be robust against any background noise.

The generic audio classifier is visualized as shown in Figure 2.1. The feature extraction technique extracts the important information from a speech or an audio signal and feeds it to a classifier. In addition to this input, the classifier has also another input data collected from a stored acoustic model. The database contains the parameters of each trained class. Different kinds of classifiers can be employed, in our case we used Gaussian mixture model and feed forward neural network. Finally, the classifier compares extracted features from the audio signal with the stored acoustic data and retrieves a corresponding predicted class. There are a lot of feature extraction techniques used in different projects; the audio features implemented in this thesis are briefly explained in the following sections. These features are selected since they are the most common feature extraction techniques suggested by researchers. Besides, these features are able to mimic the behavior of human ear.

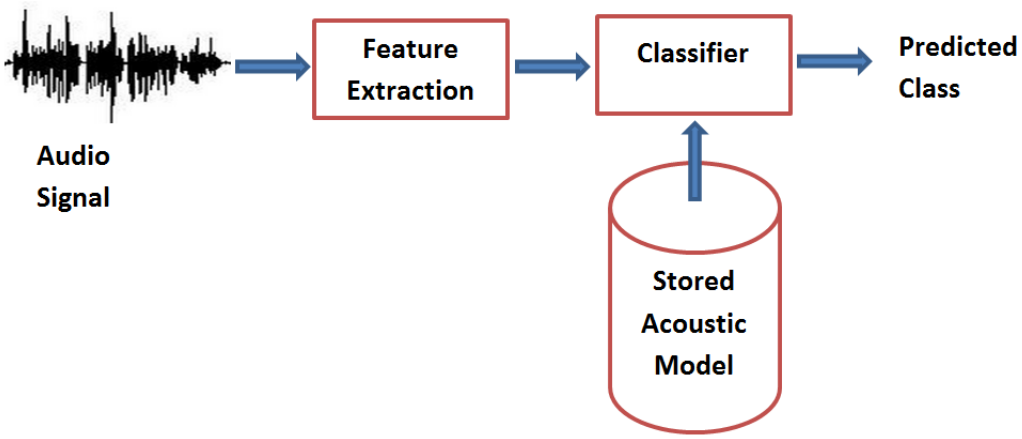


Figure 2.1: ***The Generic Audio classifier.***

2.2 Mel-frequency cepstral coefficients (MFCC)

A wide range of techniques are employed to represent an audio signal. Perhaps the most popular one is Mel-frequency cepstral coefficients. The main goal of MFCC feature extraction technique is to impersonate the behavior of human ear. It is widely used in applications such as face recognition [16], image processing [17], automatic speech and speaker recognition. The high

level steps to calculate MFCCs are shown as follows:

I. Divide the continuous signal into short frames.

II. Compute the periodogram of the power spectrum for every frame.

III. Multiply mel filterbank and power spectra with proper spacing done by a mel scale.

IV. Find the logarithm of all mel filterbank energies.

V. Compute the DCT of the Mel spectrum.

VI. Keep only 2-13 DCT coefficients and exclude the remaining coefficients.

The basic concept of a MFCC processor as described is shown in Figure 2.2.

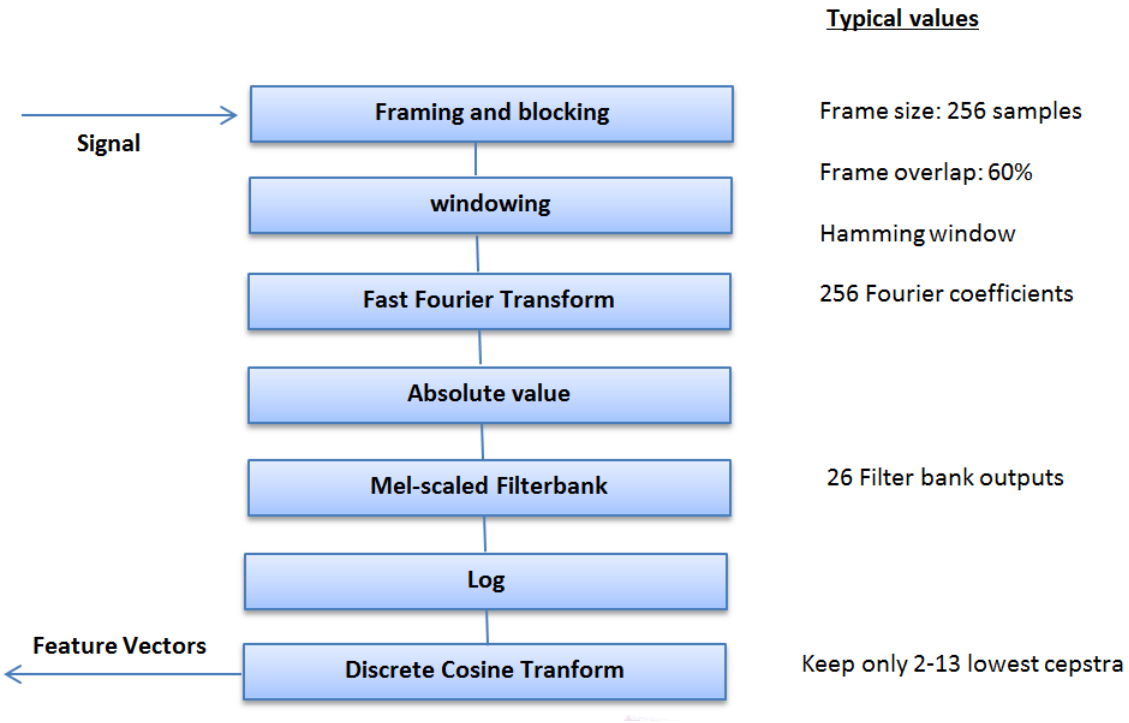


Figure 2.2: ***MFCC processor.***

2.2.1 Framing and Blocking

Since an audio signal is frequently changing, the continuous signal is divided into short overlapping frames to get a statistically stationary audio signal. The continuous audio signal is blocked into frames of N samples where the adjacent frames being overlapped by $N - M$ samples ($M < N$). While the first frame comprises the first N samples, the second frame begins M samples after the first frame. The standard value of N and M are 256 and 100 samples, respectively. The drawback of taking shorter frame size is that we don't have enough samples to get genuine information and similarly longer frame size causes periodic change of the frame information.

2.2.2 Windowing

The next step is to window each frame in order to minimize discontinuities at the beginning and ending of the frame. The main idea here is to minimize disruptions by using a window to diminish both beginning and ending of the frame to zero. There are different kinds of windows with their own pros and cons. Some of them are more advantageous if applied to sinusoidal or random signals. Some are helpful in detecting an exact frequency of a peak in a spectrum, while some accurately specify the level of an amplitude peak. The appropriate window should be chosen in accordance to the specific application and Table 2.3 shows the most common windows with their features. Hamming window is the commonly used window function as compared to

Window	Best for these Signal Types	Frequency Resolution	Spectral Leakage	Amplitude Accuracy
Barlett	Random	Good	Fair	Fair
Blackman	Random or mixed	Poor	Best	Good
Flat top	Sinusoids	Poor	Good	Best
Hanning	Random	Good	Good	Fair
Hamming	Random	Good	Fair	Fair
Kaiser-Bessel	Random	Fair	Good	Good
None (boxcar)	Transient & Synchronous Sampling	Best	Poor	Poor
Tukey	Random	Good	Poor	Poor
Welch	Random	Good	Good	Fair

Figure 2.3: ***Types of windows [30].***

the existing many window functions such as blackman window and flat top window. Hamming window is formulated as shown in Equation 2.1 and plotted as in Figure 2.4.

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right) \qquad 0 \leq n \leq N-1 \tag{2.1}$$

The output after windowing can be expressed in Equation 2.2 as follows:

$$y(n) = x(n) \times w(n) \qquad 0 \leq n \leq N-1 \tag{2.2}$$

Where N stands for the number of samples in the frame.

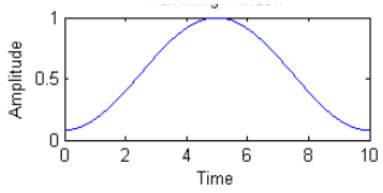


Figure 2.4: ***Hamming window.***

2.2.3 Fast Fourier transform (FFT)

The Fast Fourier transform is a fast algorithm for efficiently calculating the Discrete Fourier transform (DFT) on the N samples of the frame. It converts each frame from the time domain to the frequency domain. The Fourier transform of a continuous signal can be formulated as in Equations 2.3 and 2.4.

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-i2\pi ft}dt \tag{2.3}$$

$$x(t) = \int_{-\infty}^{\infty} X(f)e^{i2\pi ft}df \tag{2.4}$$

For $-\infty < f < \infty$, $-\infty < t < \infty$, and $i = \sqrt{-1}$. The corresponding Fourier transform for the sampled version of a continuous signal can be defined as in Equations 2.5 and 2.6.

$$X(j) = \frac{1}{N}\sum_{k=0}^{N-1} x(k)e^{-i2\pi jk/N} \tag{2.5}$$

$$x(k) = \sum_{j=0}^{N-1} X(j)e^{i2\pi jk/N} \tag{2.6}$$

For $j = 0, 1, \cdots N-1$ and $k = 0, 1, \cdots N-1$. While $X(j)$ computes the frequency domain function, $x(k)$ represents the time domain function. When the $x(k)$ series is real, then the real part and imaginary part of $X(j)$ are an even function and an odd function, respectively. The interpretation is that the FFT coefficients between $N/2$ and $N-1$ can be visualized as the negative frequencies between $-N/2$ and -1. Some of the common operations related with FFT are computing a spectrogram, the convolution of two time series and the correlation of two time series. A spectrogram is defined as a visual representation of the power spectrum of the sound as a function of time. The power spectrum of an original audio signal is estimated by a square of the complex Fourier coefficients' magnitude. Even though it is possible to evaluate these operations without the help of FFT, it is preferred to use FFT due to its computational saving. But aliasing, which is a phenomenon where high frequency components impersonate low frequencies, is one of the problems experienced by using DFT. The next task is to calculate the power spectrum of each frame so as to identify which frequencies are present in the frame.

2.2.4 Mel scale

The mel scale indicates how precisely to space our filter banks. According to the different studies conducted, human perception of frequency contents of an audio signal follows a linear frequency spacing below $1000Hz$ and a logarithmic spacing above $1000Hz$ as shown in Figure 2.5. Thus, for each actual frequency in Hz, the corresponding mel scale is computed using the approximate formula in Equation 2.7.

$$m_f = 2595\log_{10}(\frac{f}{100} + 1) \tag{2.7}$$

Figure 2.5: ***Mel scale [17].***

We apply the mel filterbank, which are a set of triangular filters, to the power spectra with proper spacing done according to the mel scale. Then, we take the logarithm of the energies to match our features more closely to what humans actually hear since loudness doesn't follow a linear scale. The general form of the filterbank is represented in Figure 2.6. The logarithm of these energies, which is called as Mel spectrum, can be employed for determining the first 13 coefficients using DCT. The higher frequency coefficients are discarded since they have little information for classifying the audio signal.

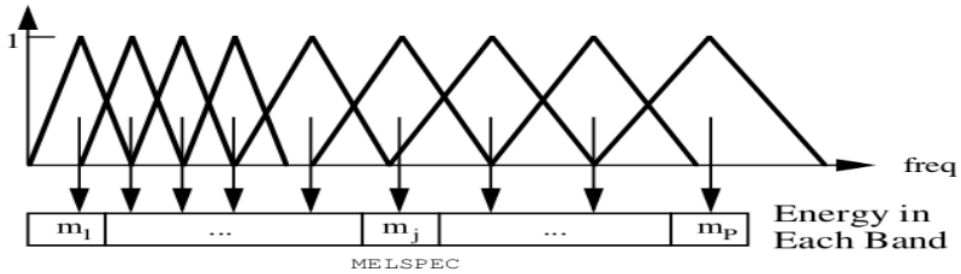


Figure 2.6: ***Mel scale filter bank [20].***

2.2.5 Discrete cosine transform (DCT)

The final task is to convert the log mel spectrum back to the time domain using discrete cosine transform. Once MFCC is computed, we have a representation of the spectrum analogous to the human hearing perception. The overall envelope is more important than the fine details of the spectrum. When we compute the DCT and ignore the higher coefficients, we are keeping the essential information in the lower coefficients. Since the higher coefficients are more noise-like, they don't have beneficial information and are not relevant for training. So, each frame of the audio signal is converted to a sequence of MFCC feature vectors and the output is formulated

in Equation 2.8 as follows:

$$C_n = \sqrt{\frac{2}{k}} \sum_{l=1}^{k} (logD_k) \cos\left[m(l + \frac{1}{2})\frac{\pi}{k}\right] \tag{2.8}$$

k shows number of samples, D_k indicates the DFT values, $m = 0, 1 \cdots k-1$, C_n represents the MFCC and m, which in our case is 13, is the number of coefficients. So, the total number of coefficients extracted from each frame is 13. Note that the first coefficient ,C_0, constitutes the mean value of the input audio signal and carries little specific information so we exclude it from the DCT. This first coefficient is known as a direct current (DC) coefficient while all the remaining coefficients are mentioned as an alternating components (AC) coefficients. These coefficients or waveforms are also referred as cosine basis functions and at the same time they are orthogonal to each other. So, this orthogonality indicates that one waveform cannot be expressed as a linear combination of any other waveforms. Thus, summation overall points of the multiplication of one waveform with the other yields zero value while multiplication of a waveform with it self yields a constant value.

2.3 Gammatone frequency cepstral coefficients (GFCC)

This section explains an acoustic feature extraction technique known as Gammatone frequency cepstral coefficient which is based on Gammatone filters.

2.3.1 Gammatone filter (GTF)

Johannesma introduced Gammatone filters in 1972 to elaborate the impulse response function of cochlea in the auditory system [21]. The Gammatone filter banks are modeled to imitate the behavior of human hearing [22] and they are overlapped non-uniform bandpass filters.

The time domain impulse response of a Gammatone filter is described in Equation 2.11. The function is an amplitude modulated sinusoidal tone of frequency f_cHz, where the envelope is governed by the well known Gamma distribution. The naming is originated from these two components by observing the nature of the impulse response [23].

$$Gamma\ distribution: \quad At^{n-1}e^{-2\pi bt} \tag{2.9}$$

$$Sinusoidal\ tone: \quad \cos(2\pi f_c t + \phi) \tag{2.10}$$

$$Gamma\ tone: \quad At^{n-1}e^{-2\pi bt}\cos(2\pi f_c t + \phi) \tag{2.11}$$

The parameter n is the filter order and regulates the shape of the envelope, A describes the output gain, b controls the duration of impulse response, i.e., the bandwidth as given in Equation 2.12, f_c determines the center frequency of filter. Moreover, ϕ is the phase shift which describes the relative position of the carrier with respect to the envelope and for simplicity it is set to

zero. According to some studies [24], [25], the gammatone filter has a good approximation of human auditory filter when between 3 and 5 filter orders are used. The Gammatone impulse response is visualized in Figure 2.7.

$$b = 1.019 * 24.7 * \left(4.37 * \frac{f_c}{1000} + 1\right) \tag{2.12}$$

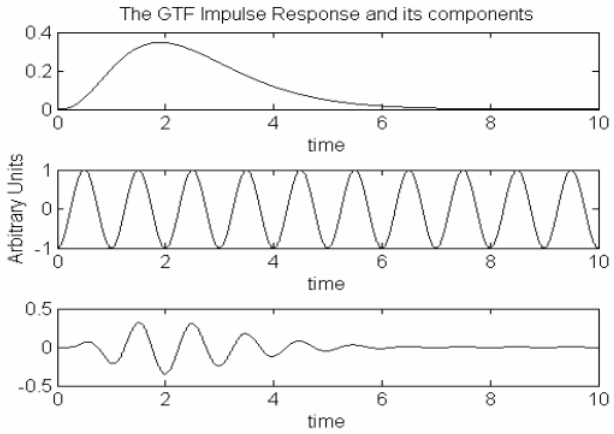


Figure 2.7: ***The gammatone distribution is on top, in the middle is the sinusoidal tone and the gammatone impulse response is described in the bottom [26].***

The gammatone filters are also known as channels. Note that there are around 3000 inner hair cells in the cochlea of the human auditory system. Since each hair oscillates at a different frequency, it means that the system has approximately 3000 bandpass filters. So as to imitate the behavior of the human auditory system, the center frequency of the gammatone filters are equally distributed based on the bark scale as given in Equation 2.13.

$$Bark = 13 \arctan\left(0.00076 f\right) + 3.5 \arctan\left(\left(\frac{f}{7500}\right)^2\right) \tag{2.13}$$

The bandwidth of each auditory filter can also be described by a psychoacoustic measure known as equivalent rectangular bandwidths (ERB) as in Equation 2.14.

$$ERB[f_0] = 6.23 * 10^{-6} f_0^2 + 93.39 * 10^{-3} f_0 + 28.52 \quad (100 < f_0 > 10000) \tag{2.14}$$

The approximate frequency response of gammatone filters for modeling the human auditory filter is given by Equation 2.15. We can estimate an n^{th} order gammatone filter by a cascade of n identical first order gammatone filters.

$$GT(f) \approx \left[1 + \frac{j\left(f - f_0\right)}{b}\right]^{-n} \quad (0 < f < \infty) \tag{2.15}$$

One of the factors that contribute to the acceptance of GTF in the speech-recognition community is its simple description in terms of time-domain impulse response. Since the time-domain implementation avoids the complex spectral evaluation, it might help on an efficient hardware implementation [26]. In contrast, one of its drawbacks is its complex frequency domain illustration. Due to this complexity, it is difficult to execute GTF in the analog domain.

2.3.2 GFCC basics

GFCCs are based on gammatone filters designed to simulate the behavior of the human ear. GFCC is also a FFT based feature extraction technique and the computation process is similar to the MFCC feature extraction scheme. The block diagram for GFCC is shown in Figure 2.8.

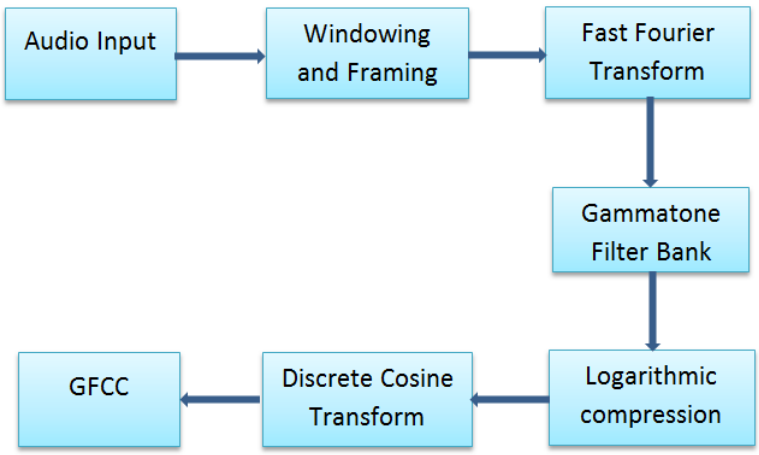


Figure 2.8: ***GFCC block diagram [27].***

Similar to the explanation for MFCC in section 2.2, the first step of the algorithm is to divide an audio file into frames with duration between $20-30ms$. In such a small interval, the continuous audio signal can be considered as stationary for facilitating the signal analysis. Hamming window is the preferred alternative to minimize discontinuities at the begin and end of the frame. Then, FFT converts the audio signal from the time-domain to the frequency domain. So as to simulate the cochlea, we use a group of filters known as gammatone filter bank and apply them to the signal's Fast Fourier transform. In order to model the human loudness perception, the logarithm is employed for each filter bank output. Finally, the GFCC feature vectors are extracted applying a DCT.

2.4 6D and 9D feature vectors

6D feature vectors also known as gianna features [3] are computed by combining 6 different scalar values. These scalar values are ZCR, short time energy (STE), energy entropy, spectral centroid, spectral rolloff, and spectral flux [28], [29]. Similarly, 9D feature vectors also called as gianna and shape features [3] combine the above mentioned 6D gianna features and other 4D shape features. Note that these shape features are also combinations of the scalar values called spectral centroid, standard deviation, skewness and kurtosis [28].

2.4.1 Zero-crossing rate (ZCR)

ZCR is a measure of the rate of sign changes of an audio signal rated by the value of the signal [28]. If consecutive samples have different algebraic signs then a zero crossing occurs. The periodic and noisy signals have small ZCR and high ZCR, respectively. The ZCR is formulated as in Equation 2.16.

$$ZCR = \frac{1}{M-1} \sum_{d=1}^{M-1} \Pi\{S_m S_{m-1} < 0\} \tag{2.16}$$

M indicates the length of signal S, the value of the function $\Pi(a < b)$ becomes 1 whenever the inequality $a < b$ is true, otherwise its value is 0.

2.4.2 Spectral flatness measure (SFM)

SFM is also sometimes known as "tonal coefficients" and is used in applications such as digital signal processing to characterize an audio spectrum [28]. High spectral flatness shows that all spectral bands of the spectrum has a similar amount of power analogous to white noise while low spectral flatness indicates the spectral power is condensed in a comparatively small number of bands like sine waves. As can be observed from Equation 2.17, it is computed by a ratio of the geometric mean of the power spectrum to the arithmetic mean of the power spectrum, where x(n) represents the amplitude.

$$SFM = \frac{\sqrt[N]{\prod_{n=0}^{N-1} x(n)}}{\frac{\sum_{n=0}^{N-1} x(n)}{N}} \tag{2.17}$$

2.4.3 Spectral centroid and spread

Spectral centroid is a measure frequently related with the brightness of a sound and shows the center of mass of the spectrum [28]. This feature is found by calculating the "center of gravity" evaluated using the information from the Fourier transform frequency and magnitude values. Equation 2.18 gives the mathematical formulation, where x represents the center frequencies of the bins and p(x) is the probability of x.

$$\mu = \int x.p(x)dx \tag{2.18}$$

Similarly the spectral spread or variance indicates the deviation of the spectrum from its mean and can be calculated as in Equation 2.19.

$$\sigma^2 = \int (x-\mu)^2.p(x)dx \tag{2.19}$$

2.4.4 Spectral skewness

Spectral skewness is a feature which measures the asymmetry of a probability distribution in the spectrum of a signal [28]. A negative or positive skewness shows the concentration of energy on a lower and a higher parts of the spectrum, respectively. A zero value indicates a symmetric distribution, so the probability density function is relatively evenly distributed on both sides of the mean. It is calculated from the third order moment in Equation 2.20. Then, the spectral skewness is evaluated as $\frac{m_3}{\sigma^3}$.

$$m_3 = \int (x-\mu)^3 . p(x) dx \tag{2.20}$$

2.4.5 Short time Energy (STE) and energy entropy

STE is found by evaluating the addition of the squares of amplitudes [29] as in Equation 2.21, where x(n) is amplitude.

$$STE = \sum_{n=0}^{N-1} x^2(n) \tag{2.21}$$

The energy entropy is another measure which illustrates the abrupt changes in the level of energy in the signal [29]. The mathematical formulation is shown in Equation 2.22, where K represents the number of sub frames and σ is the normalized STE of a sub frame.

$$E = -\sum_{i=0}^{K-1} \sigma^2 . \log_2(\sigma^2) \tag{2.22}$$

2.4.6 Spectral roll off, Spectral kurtosis and Spectral flux

Spectral roll off is frequency domain feature and it is defined as the frequency where 95% of the signal energy is concentrated [28] as in Equation 2.23. f_c represents roll off frequency while $\frac{sr}{2}$ indicates the Nyquist frequency.

$$\sum_{0}^{f_c} a^2(f) = 0.95 \sum_{0}^{\frac{sr}{2}} a^2(f) \tag{2.23}$$

Spectral flux is a measure of the change in power spectrum of a signal between successive frames [29]. Equation 2.24 gives the mathematical formulation. Where N indicates the power spectrum, j represents a frame and k is a sample in each frame.

$$Flux_j = \sum_{k=0}^{S-1} (N_{j,k} - N_{j-1,k})^2 \tag{2.24}$$

Spectral kurtosis is defined as the measure of the probability density function's flatness with respect to its mean value [28] and it is evaluated from the fourth moment in Equation 2.25.

Thus, spectral kurtosis is calculated as $\frac{m_4}{\sigma^4}$.

$$m_4 = \int (x - \mu)^4 . p(x) dx \tag{2.25}$$

Once we have extracted important information using the different feature extraction techniques, the next task is to feed those features as input in to classifiers for both training as well as testing the system. Thus, the next chapter 3 introduces to different classification schemes utilized in our thesis.

Chapter 3

Classifiers

This chapter describes the most common classification schemes used in pattern recognition applications. In the beginning, we briefly present the basic concepts of Gaussian mixture models. In this context, we explain the expectation maximization algorithm which is the most well-established technique for estimating parameters. In addition, this chapter presents how to decide whether the feature vector belongs to the model or not in accordance to evaluation of maximum posterior probability of feature vectors. This chapter also introduces the inspiration for artificial neural networks, the different layers of neural networks, the different activation functions employed and the back-propagation algorithm used to train the network properly. Finally, the basic concepts of another classification scheme known as support vector machine is illustrated.

3.1 Gaussian mixture model (GMM)

Different kinds of models can be used to train a system. GMM is among the mature methods which have an increasing demand in the pattern recognition community. A Gaussian mixture model is an unsupervised classifier, because we estimate the probability density function of the observations during a training stage, since in the beginning the data points are not assigned to any of the Gaussian. Furthermore, it is a parametric probability density function stated as a weighted sum of a given number of gaussian component densities as shown in Figure 3.1. In order to classify a given data having MFCC or GFCC feature vector, the GMM parameters are estimated in the training stage. The most common method to achieve this is Expectation Maximization algorithm. To build a classifier with GMMs, every class constructs its gaussian mixture independently in the training stage. Because of that, it is not necessary to retrain the whole system when a new class is added to the classification problem. One drawback that can restrict its practical usage in real time applications is that it demands huge amount of memory to store the various coefficients and complex exponential calculations.

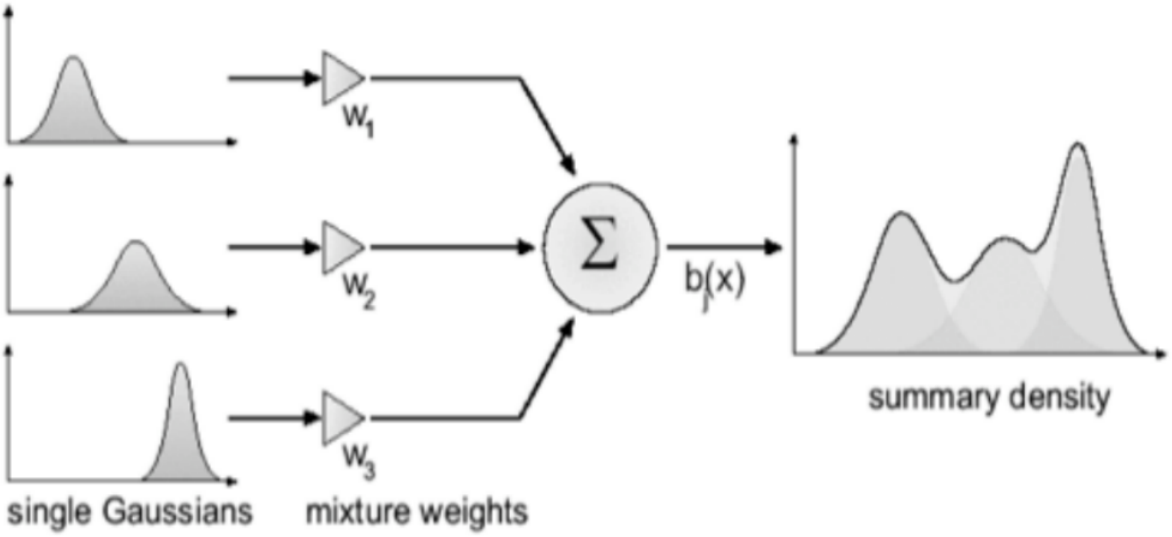


Figure 3.1: ***1-D Gaussian Mixture Model [18].***

3.1.1 Univariate and Multivariate Gaussian distribution

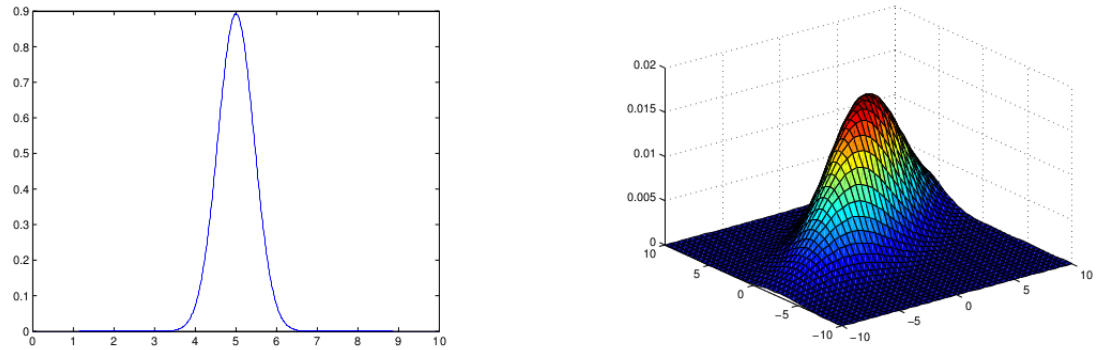
The Univariate Gaussian distribution is also known as normal distribution. It plays a very essential role in different methods of statistics. A normally distributed continuous random variable is symmetric with respect to its mean. While the mean of a distribution describes the center position of its plot, the standard deviation determines how much a given value deviates from the center of its graph. Depending on the value of standard deviation, the curve will become either flat and wide or tall and narrow. This belly shaped distribution is defined by Equation 3.1 and plotted in Figure 3.2a.

$$p(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right) \tag{3.1}$$

The parameter μ is the mean and σ the standard deviation of the distribution. A distribution is called standard normal distribution when the $\mu = 0$ and $\sigma = 1$. A vector variable X having a multivariate gaussian distribution with mean μ and covariance Σ is given by the probability density function in Equation 3.2 and shown in Figure 3.2b.

$$p(x;\mu,\Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) \tag{3.2}$$

Where $x = [x_1, \cdots, x_n]^T$ and it is defined as $X \sim N(\mu, \Sigma)$ in which N is the normal distribution, n is the number of features in the model, μ is the n dimensional mean vector, Σ is a n by n covariance matrix and $|\Sigma|$ denotes its determinant. In most cases the features are independent and thus Σ is a diagonal matrix.



(a) Univariate Gaussian density **(b) Multivariate Gaussian density**

Figure 3.2: ***On the left is a univariate Gaussian density with a single variable and on the right is a multivariate Gaussian density with two variables [31].***

3.1.2 GMM training

A Gaussian mixture model then consists of K multivariate Gaussian distributions with a linear combination of weights $\pi_{k\epsilon\{1\cdots K\}}$. During training, we estimate the GMM parameters, which are the mean vectors, covariance matrix and mixture weights, of the multivariate gaussian probability density functions for given training data points consisting of feature verctors. As stated in section 3.1 the most common and well-established technique for estimating the parameters is the EM-algorithm. This algorithm is an iterative method which estimates the mean and covariance matrix till some convergence threshold is reached. The EM algorithm has two steps which are known as expectation step and maximization step. The algorithm initially assumes a given set of parameters for the components of the Gaussians and it is assured to converge to a local optimum. The steps are explained as follows:

1: Pick an initial guess for the parameters: means μ_j, covariances Σ_j and mixing coefficients π_j.

E-step: Compute posterior probabilities or responsibilities using the current values of the parameters as in Equation 3.3 for all data points x_i , $1 \leqslant i \leqslant N$ and all mixture components, $1 \leqslant k \leqslant K$. Note that for each data point x_i, the weights are defined such that $\sum_{k=1}^{K} \pi_k = 1$.

$$\gamma_k(x) = \frac{\pi_k N(x|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j N(x|\mu_k, \Sigma_k)} \tag{3.3}$$

M-step: Re-estimate the mean, covariance and mixture coefficients using the probabilities from the E-step. The updated mean vector, covariance and mixing coefficients are found

as indicated by Equations 3.4, 3.5 and 3.6, respectively.

$$\mu_j^{new} = \frac{\sum_{n=1}^{N} \gamma_j(x_n)\, x_n}{\sum_{n=1}^{N} \gamma_j(x_n)} \tag{3.4}$$

$$\Sigma_j^{new} = \frac{\sum_{n=1}^{N} \gamma_j(x_n)\,(x_n - \mu_j)\,(x_n - \mu_j)^T}{\sum_{n=1}^{N} \gamma_j(x_n)} \tag{3.5}$$

$$\pi_j = \frac{1}{N} \sum_{n=1}^{N} \gamma_j(x_n) \tag{3.6}$$

Note that x_i and μ_j are d dimensional vectors, and the covariance matrix is of dimensionality $d * d$.

Step 4: Evaluate log likelihood value as in Equation 3.7 and repeat the E-step and M-step until convergence.

There are two options in initializing the algorithm. The first one is to conduct the E-step by setting some initial parameters which are the mean, the covariance matrix and the mixture coefficients. The second option is to accomplish first M-step by starting with a set of initial posterior probabilities or responsibilities $\gamma_k(x)$ for every data points. When either the magnitude of the estimated parameters or the value of the log-likelihood calculated after each iteration doesn't appear to change in a significant amount or falls below some threshold then it indicates that the algorithm has converged. Note that the log-likelihood is defined as in Equation 3.7.

$$\ln\ p(X|\mu, \Sigma, \pi) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k N(x_n|\mu_k, \Sigma_k) \right\} \tag{3.7}$$

Here we provide an example of the EM algorithm for a mixture of two gaussians applied to some data sets in Figure 3.3 below. Plot (a) indicates the data points in green and the two gaussian mixture components are shown as blue and red circles. Plot (b) shows the result of an initial E-step where every data point is assigned to either of the gaussian components according to the value of posterior probabilities. The condition after the first M-step is indicated in Plot (c) in which the blue gaussian has moved to the center of mass of the blue data-set and the same happened for the red gaussian. Figures (d), (e) and (f) indicate the results after 2, 5 and 20 iterations of the EM algorithm, respectively.

3.1.3 GMM testing

The next step after the training phase of the GMMs is to use it as a classifier in the testing phase. In this step we decide whether a given feature vector belongs to a model based on an evaluation of the Maximum a posterior probability(MAP) relative to the model.

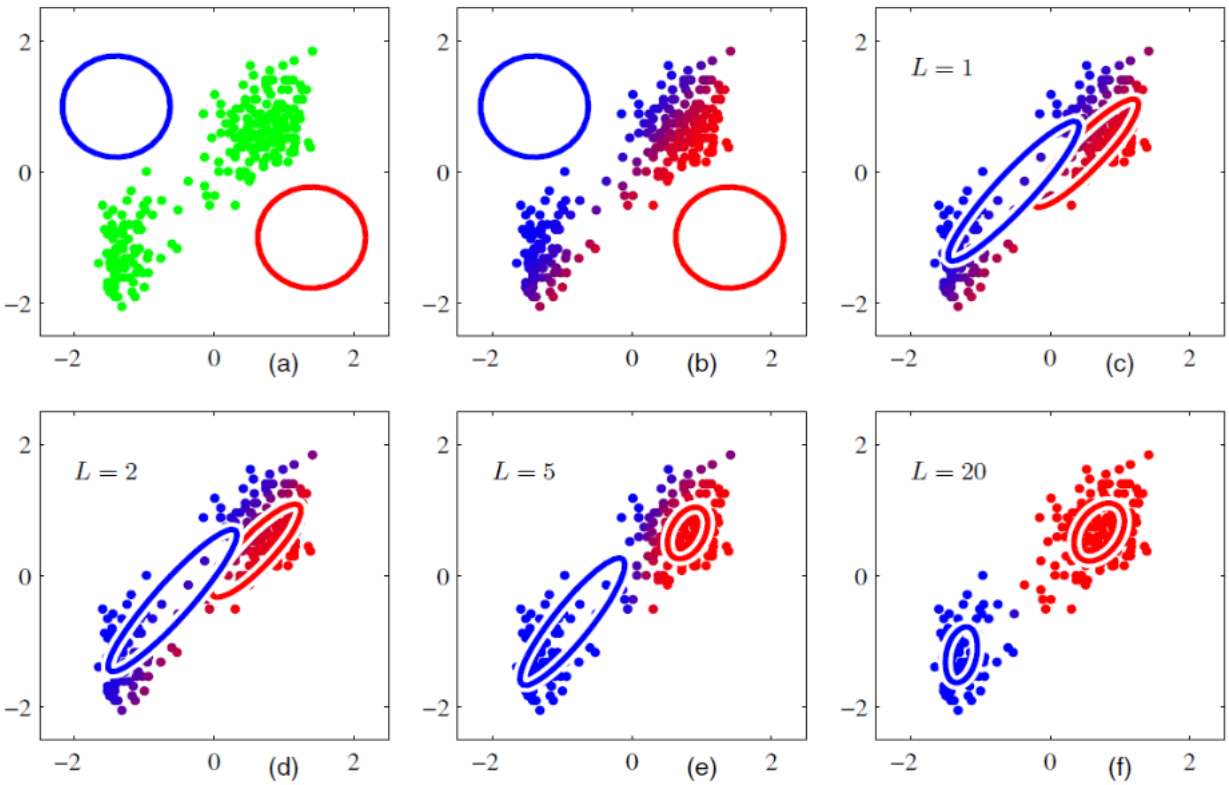


Figure 3.3: ***Illustration of EM algorithm for a given data set [19].***

Maximum posterior probability of the parameters θ is described as finding the parameters θ that maximizes the posterior distribution. The θ in our case represents the index of the GMM model for every class. Thus, our system outputs the index of a class which has the highest MAP value. The posterior distribution of θ is given by $p(\theta|x)$ and mathematically as $p(\theta|x) = \frac{p(x|\theta)g(\theta)}{p(x)}$. Since the denominator of posterior function doesn't depend on θ, it doesn't have contribution in the maximization. Equation 3.8 shows the formula for estimating MAP, where θ is the GMM class and x is the feature vector of the observation.

$$\theta_{MAP}(x) = \underset{\theta}{\operatorname{argmax}}\, P(x|\theta)g(\theta) \tag{3.8}$$

Maximum likelihood estimate (MLE) of the parameter θ is defined as searching the parameters θ value which maximizes its likelihood function. The likelihood function is a function in terms of its parameters which means it is a joint probability of the perceived data conditioned on the parameters. The likelihood function of the parameter θ is given by $P(x|\theta)$. Then, the MLE estimate of θ is formulated as in Equation 3.9. In MLE there could be more parameters that satisfies the property, so it might not be unique. Furthermore, MLE could fail to exist which means there might not be θ that achieves the maximization property.

$$\theta_{ML}(x) = \underset{\theta}{\operatorname{argmax}}\, P(x|\theta) \tag{3.9}$$

Some of the advantages of MLE are: its computation is easy and it is usually interpretable. In addition, it has asymptotic properties such as consistency, normal and efficiency. MLE is also invariant under re-parameterization. On the other hand, it has also some disadvantages. Since it is a point estimate, there is no representation of uncertainty. In addition, it can over-fit the data which means the value of the parameter estimate is sensitive to a random change in the data.

Similar to MLE, MAP has also some pros and cons. The advantages of MAP are the simplicity of its calculation and interpretation of results. In contrast to MLE, they avoid over fitting of data. Furthermore, it usually converges to same value as MLE. To mention some of the problems, there is no representation in uncertainty for the parameter θ because it is point estimate. In contrary to MLE, MAP are also not invariant under re-parameterization.

3.2 Neural network basics

Many people compare computers with the human brain and actually they have some similarities. The human brain consists of billions of cells known as neurons. The three parts of a neuron are the central body, dendrites or cells input to central body and the axions or cell outputs which carry information outside the central body. The structures called synapses are used to convert the activities of axions to electrical pulses. Similarly, computers contain billions of switching devices called transistors which are packed on an integrated circuit. However, their main difference is that both operate differently. While the transistors in computer are connected in a very simple manner, the interconnection among neurons is very complex. computers use a limited set of instructions for executions and most of them are used as a passive data storage, while large portions of our brains work actively in parallel. The conventional computers are not able to solve a problem unless they follow the specific set of instructions and that is why their problem solving capacity is restricted to only known problems. The invention of neural network concepts tried to bridge this gap.

The inspiration behind artificial neural networks is the process of information retrieval in the human brain. The idea is to simulate the activities of a brain in a computer and perceive patterns like humans do. The good thing about neural networks is that we don't need to program it to perform some activities. In other words, it can adapt to the environment like the human brain and it learns by example. Some neural networks might have the capacity to generalize to datasets other than the trained data if they are properly trained. Neural networks usually utilize supervised learning for practical applications. In supervised learning the inputs and the desired outputs are fed as training data to the network. However, it requires a lot of training data points and it might take a lot of computational time for successful training. When the network is trained properly, we can feed it new inputs and the system should be able to properly estimate the target output. Some of the advantages of using neural networks are

its adaptive learning which is the capability to learn from the provided input information, its parallel computation of different activities and its fault tolerance.

A typical neural network consists of some set of artificial neurons or nodes which are organized in a series of layers as shown in Figure 3.4. The nodes in each layer also contain an activation function. The input nodes collect information from an external environment which will be used to train the network. The output nodes contained in the last layer, provide a response in accordance with the information it has learned before. The activity of output nodes rely on the behavior of hidden nodes and the weights between hidden and output nodes. There could be one or more hidden layers, containing many nodes between input and output layer, which perform most of the activities of artificial brain. The behavior of the hidden layers depends on the input nodes and the weight between input and hidden nodes. The links among nodes bear weights which might have either a positive value to magnify the input or a negative value so as to inhibit the signal. The magnitudes of the weights indicate the significance of one node for another one. Assigning the value of weight by hand is not feasible when the number of nodes becomes large. It is necessary to implement an algorithm which tunes the weights to the desired output. This process of determining the weights is known as learning or training the neural network.

Many different types of neural networks have been proposed since the first model by McCulloch and Pitts [32]. They differ by the activation function used, the training algorithm implemented, the topology of the system etc. Feed forward neural networks and feedback neural networks are two examples. The information in feed-forward neural network travels in a direct way from the input node to the hidden node and then to the output node. There is no cycle between the neural layers. It is simple and often applied in pattern recognition areas. Feedback neural networks are also known as recurrent neural networks (RNN) and the information travels in both directions there by creating a cycle. They are very powerful, complex and at the same time dynamic.

Each neuron has an activation function which produces the corresponding output for a provided input. These functions introduce nonlinearity to the network so as to make it more powerful and differentiate the complex relationships that are found in feature vectors. There are a lot of different activation functions. Three common functions are the linear, threshold and sigmoid functions. The output is linear to the input when linear function is employed and they are most of the time used in input and output layers. The threshold function outputs one of two levels based on whether the input is above or below the threshold value. It is difficult to train networks with threshold functions due to the fact that the error function is a stepwise constant which can not be easily differentiated. Sigmoid or logistic functions are among the most well known and widely used non-linear functions. Even though all of the mentioned functions are rough approximations, sigmoid functions approximate real neurons better than other functions.

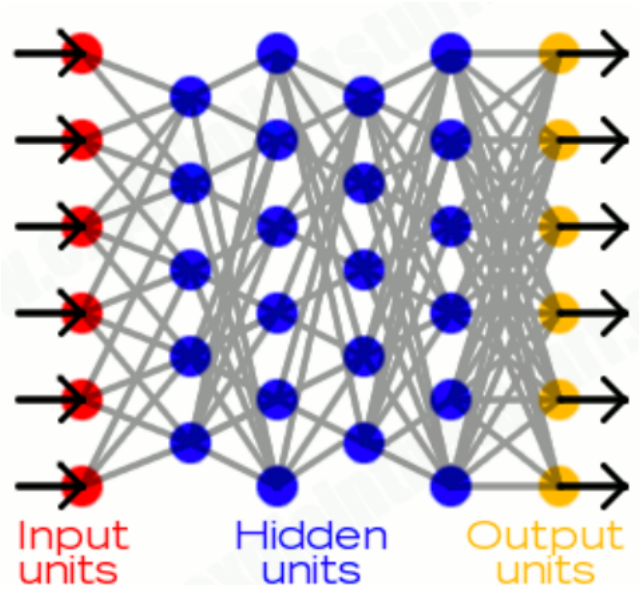


Figure 3.4: ***Neural Network Structure [33].***

They are formulated as in Equation 3.10, where β is a slope parameter.

$$\sigma(t) = \frac{1}{1 + e^{-\beta t}} \tag{3.10}$$

The diagram of this function is shown in Figure 3.5. Note that finding the derivative of these functions is easy which greatly facilitates updating the weights during training.

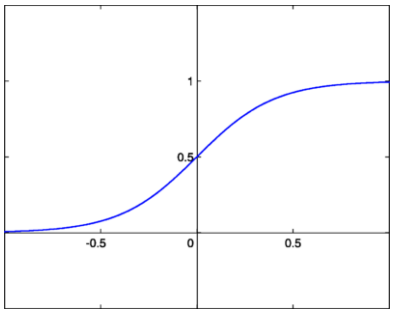


Figure 3.5: ***Sigmoid Function.***

3.2.1 Back-propagation algorithm

To train the network properly, the weights between each node should be adjusted correctly so that the error between the desired output and actual output is minimized. A well known learning algorithm employed is the so-called back-propagation of error. It is an iterative process that reduces the error learning by example. Initially the network provides an output for a given input and set of weights. Then, we compute the mean squared error between the desired output and actual output. In order to estimate the error of a hidden node just before the output layer, we multiply the weights between the hidden node and output nodes with corresponding output

node error and sum up the products. This result equals the error for the provided hidden node. This procedure continues for all nodes from one layer to another layer in backward direction opposite to the normal information flow. For each example case, the weight change is updated so as to minimize the error. This iterative process repeats until the overall error is below some threshold value.

3.3 Support vector machine (SVM)

SVMs are another method to solve the classification problem through finding separating hyperplanes. In other words, given a model of labeled training data, the goal of the algorithm is to output an optimum hyperplane which classifies the test data. Vladimir Vapnik and colleagues introduced the SVM in 1995 [34]. However, due to the fact that the data is usually not linearly separable, SVM came up with the idea of the kernel based feature space in order to represent the data in higher-dimensional space. SVM has a solid theoretical background and became practically successful.

For simplicity let us see how to find an optimal hyperplane which can linearly separate the blue and red 2-D data points of two different classes. We can observe from Figure 3.6 that there could be many straight lines which can separate the two classes. But all of them are not optimal solutions to the problem. An optimal solution is a line which is far as much as possible from the data points. If a line is near to data points, it might be very sensitive to noise and could fail to classify data samples correctly.

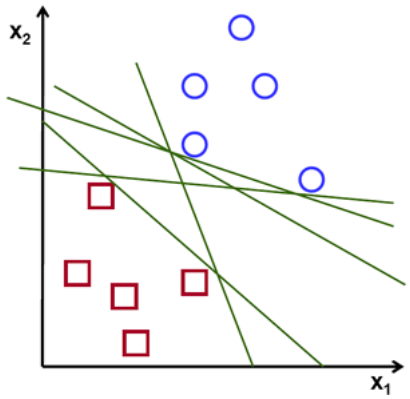


Figure 3.6: ***Separating lines for a 2D-points [36].***

3.3.1 Compute optimal hyperplane

The ultimate goal of the SVM algorithm is to find a hyperplane which provides the largest minimum distance to training data points. As can be shown in Figure 3.7, the width of the margin is twice the minimum distance to the data points. For each data point x_i, with $i = 1 \cdots n$

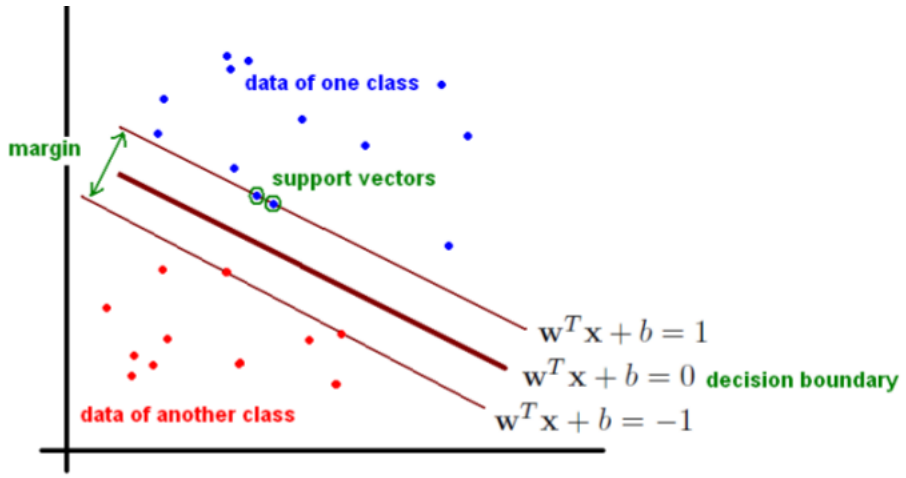


Figure 3.7: ***Separating lines for a 2D-points [19].***

data points and a class label $y \in \{1, -1\}$, a decision boundary is parameterized by a vector W and a constant b as in Equation 3.11.

$$W^T X + b = 0 \tag{3.11}$$

While any data above the decision boundary have a value of 1, any data below the boundary is labeled as -1. We can figure out if the class is classified correctly by checking if both y and $W^T X + b$ are either positive or negative. Mathematically, this can be compactly written as in Equation 3.12.

$$y_i(X_i.W + b) \geq 1 \quad \forall i \tag{3.12}$$

We also define two hyperplanes which are parallel to each other above and below the decision boundary as shown in Figure 3.7 and formulated as in Equations 3.13 and 3.14. Support vectors are defined as the training examples which are near to the canonical hyperplanes. Thus, our objective is to maximize the distance between these newly added two boundaries.

$$X_i.W + b \geq +1 \; when \; y_i = +1 \tag{3.13}$$

$$X_i.W + b \leq -1 \; when \; y_i = -1 \tag{3.14}$$

Note that the vector W is perpendicular to both straight lines. So, the distance to support vectors or to newly added boundaries is given by the formula in Equation 3.15.

$$M = \frac{2}{\parallel W \parallel} = \frac{2}{\sqrt{W^T W}} \tag{3.15}$$

Where M is the margin. Our desire is to increase the gap between data points in the two classes and at the same time maximizing this distance could help to minimize the classification problem. Thus, maximizing $\frac{2}{\sqrt{W^T W}}$ is the same as minimizing $\frac{\sqrt{W^T W}}{2}$. Since square root

function is monotonic function, we can in turn minimize $\frac{W^T W}{2}$. In summary, the optimization problem is given as in Equations 3.16 and 3.17.

$$minimize_{W,b} \ \frac{W^T W}{2} \tag{3.16}$$

$$subject\,to: \ y_i(W^T X_i + b) \geq 1 \ (\forall data\,points \ X_i) \tag{3.17}$$

This is a quadratic optimization problem in which we minimize the quadratic function subject to some linear inequality constraints. To solve the optimization problem Lagrange multipliers [35] are introduced.

3.3.2 Kernel functions

Kernel function is a similarity function which is mainly used to map the data into higher dimensional space. This function helps us to do certain calculation faster, hoping that the data could be separated in the higher dimensional space. When there is no straight line or hyperplane in 2 dimension which can be utilized to separate the data points, then the usage of kernel function helps us to tackle this problem. Furthermore, the kernel function is given as in Equation 3.18.

$$K(X_i, X_j) \equiv \phi(X_i)^T \phi(X_j) \tag{3.18}$$

When data vectors X_i are mapped in to higher dimensional feature space by the function ϕ, the formulation of the quadratic optimization problem is similar to above except the replacement of X_i with $\phi(X_i)$. Mathematically, SVM is formulated as in Equations 3.19 and 3.20.

$$minimize_{W,b} \ \frac{W^T W}{2} + C \sum_i \xi_i \tag{3.19}$$

$$subject\,to: \ y_i(W^T \phi(X_i) + b) \geq 1 - \xi_i and \ \xi_i \geq 0 \ (\forall data\,points \ X_i) \tag{3.20}$$

Nowadays, researchers proposed different kinds of kernels but the choice of a kernel varies in accordance to the kind of problem at hand. The most common ones are:

Gaussian kernel: This type of kernel is an instance of radial basis function kernel and is formulated as in Equation 3.21. The value of sigma has a pivotal role in the performance of this kernel. We use Gaussian kernel when the number of features are small and number of training samples are large.

$$K(X_i, X_j) = exp(-\gamma \parallel X_i - X_j \parallel^2), \gamma > 0 \tag{3.21}$$

Polynomial kernel: This similarity function is defined as in Equation 3.22 and it performs well when all the training data are normalized. But most of the time it performs worse as compared to Gaussian kernel. The customizable parameters are the slope sigma, constant r and degree d.

$$K(X_i, X_j) = (\gamma X_i^T X_j + r)^d, \ \gamma > 0 \tag{3.22}$$

Sigmoid kernel: Since the origin of this kernel is from neural networks, it is well known for SVM. The tunable parameters are the slope sigma and constant r. It is formulated as in Equation 3.23.

$$K(X_i, X_j) = \tanh(\gamma X_i^T X_j + r)^d) \tag{3.23}$$

Chapter 4 illustrates the system setup and data collection. It introduces the software and hardware components required for recording an audio. Then, it explains the recording setup step by step and briefly presents the position of microphone. The next chapter also discusses the selected five terrain types.

Chapter 4

System setup and data collection

This chapter provides a brief introduction to the basic components of TRADR hardware which is extensively used in our experiment. It describes how to setup the robot for data collection. External hardware components deployed on our robot and the software required to record an audio are described. In addition, recording setup and position of the microphone in the experiments are discussed. Finally, this chapter explains the different varieties of samples recorded in each of the five terrain types considered.

4.1 Unmanned ground vehicle (UGV) hardware

This section introduces the basic components of TRADR robot hardware. Tracked vehicles are more suited for search and rescue applications than wheeled vehicles due to the large contact area of the tracks with the ground. We used the tracked vehicle known as TRADR robot, which is shown in Figure 4.1, for making experiment in the thesis. The platform is a version of the Absolem platform from Bluebotics [1] and it was developed for the NIFTi project [2]. The robot platform, with two tracks on the side connected to the central body, is portable and can speed up to 5km/hr in flat surfaces. Each track has two active flippers and the mechanical differential system helps the rotation of the tracks, thus stability shows improvement on sloppy grounds. Some of the sensors installed on the robot are rotating 2D laser scanner to build up the 3 dimensional models of the area, microphones to collect sounds and vision systems placed on an arm to collect some images and/or videos. The robot platform can be divided into two main systems known as locomotion system and platform body.

[1] http://www.bluebotics.com/mobile-robotics/absolem/
[2] http://www.nifti.eu/

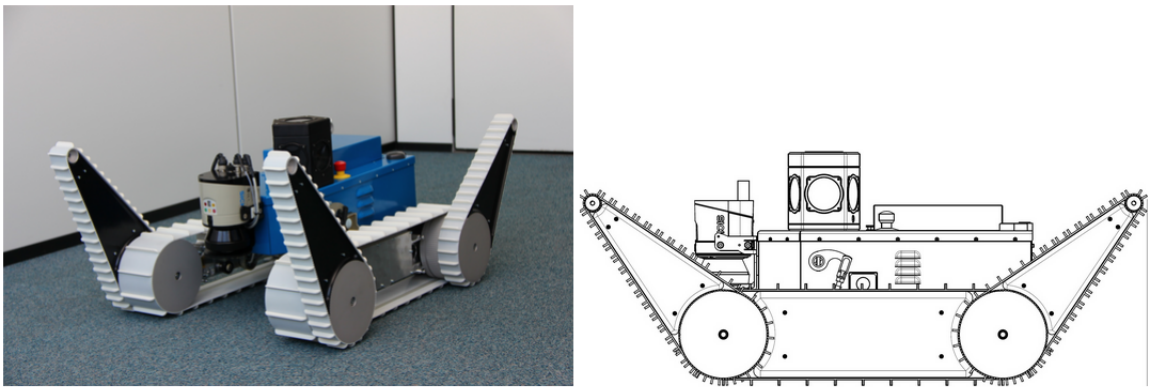


Figure 4.1: ***TRADR robot [37].***

4.1.1 Locomotion system

The locomotion system is comprised of two main tracks, which are located on the left and right side of the body, and four flippers as shown in Figure 4.2. While brackets are used to mechanically connect the two main tracks and body, a differential is employed to link the two main tracks together. Each of the main tracks have three motors. One of the motors is used for traction of the main track and the two flippers, the remaining two are required for controlling the flippers. Besides it is possible to control the flippers individually.

4.1.2 Platform body

The platform body accommodates the main electronic components such as an embedded PC, power PCB which is used for managing power distribution and 3D PCB used for handling the rotation of 3D sensor. Furthermore, the sensors deployed on the body are 3D sensor, omnicamera, IMU and GPS. The 3D sensors are rotating 2D laser scanners which are used to scan the environment around the robot. The omnicam helps to get a bird's eye view around the robot. In addition, an emergency stop button, a pull wire emergency switch and status LEDs are situated on top of the body for safety matters.

4.2 Robot system setup

In general this section explains the robot setup for data collection. The logitech cordless rumblepad2 receiver is connected to robot's USB port as shown in Figure 4.3a. Then, we mount a laptop on top of the robot as shown in Figure 4.3a in order to save the recorded audio files. The next step was to deploy an external microphone (Elecom table microphone) on back of the robot near ground where robot terrain interaction occurs. This is an electret capacitor microphone and its directional characteristics is omni-directional. Besides, it has high sound collecting properties and suitable for active robot terrain interaction without bothering much

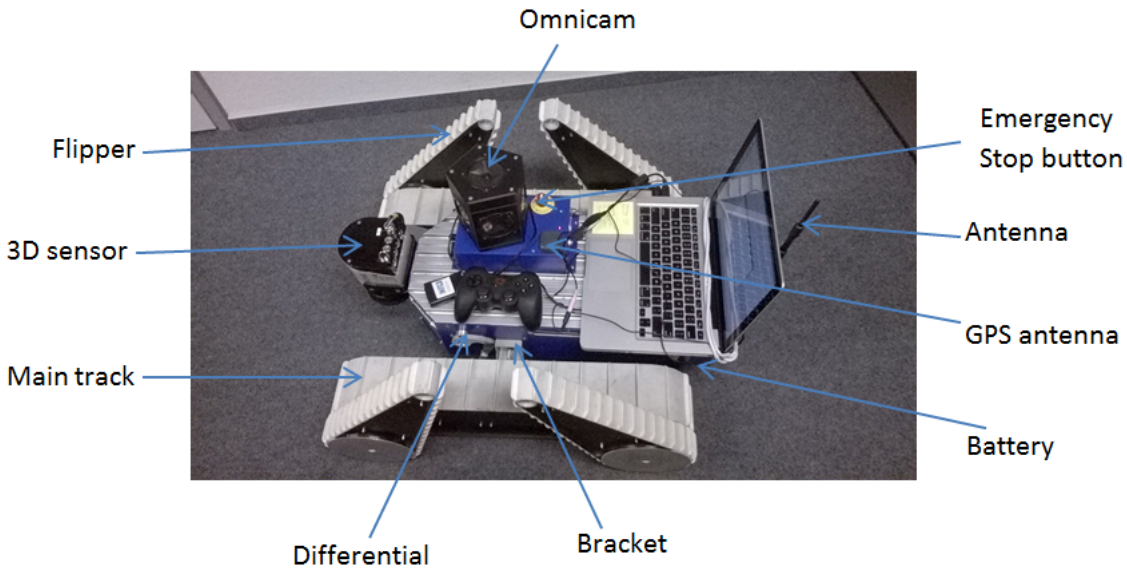


Figure 4.2: ***UGV hardware.***

by its surrounding noise. The frequency ranges from 30 to 16000Hz.

(a) Robot system setup **(b) Joystick**

Figure 4.3: ***The robot setup for data collection showing where the sensors and laptop are mounted. Left side and right side of Figure 4.3a are the close up of the joystick receiver and Elecom Table microphone, respectively. While Figure 4.3b shows the logitech cordless rumblepad2 transmitter used for running robot manually.***

A USB adapter, in our case a Sennheiser USB adapter, is used to connect the microphone with the laptop. Since in our laptop headphone and microphone jack are connected together, we prefer to use USB sound adapter. There are some advantages of using external USB adapter. First, external USB sound adapter accomplishes an analog to digital conversion (ADC) that happens outside a computer which may improve the recording quality. Second, these adapters render correct voltage for the microphones and produce more predictable outputs. Note that using an external sound conversion device is not required but it is highly recommended.

Audacity is an open source program used to record and manipulate sound as digital audio waveforms, supporting sound file formats such as WAV, AIFF, MP3 and Ogg vorbis. An audio

file in audacity is editable and every action is undo-able. Audacity is cross platform and runs on different operating systems such as Windows, Max Os X and unix platforms. Even though audacity is one of the powerful audio editors that have no limit on the size of any track, it also has drawbacks. The disadvantage is that on most systems it cannot support multitrack recording, in other words it can not record more than two channels at the same time. Figure 4.4 shows a screenshot of the audacity interface.

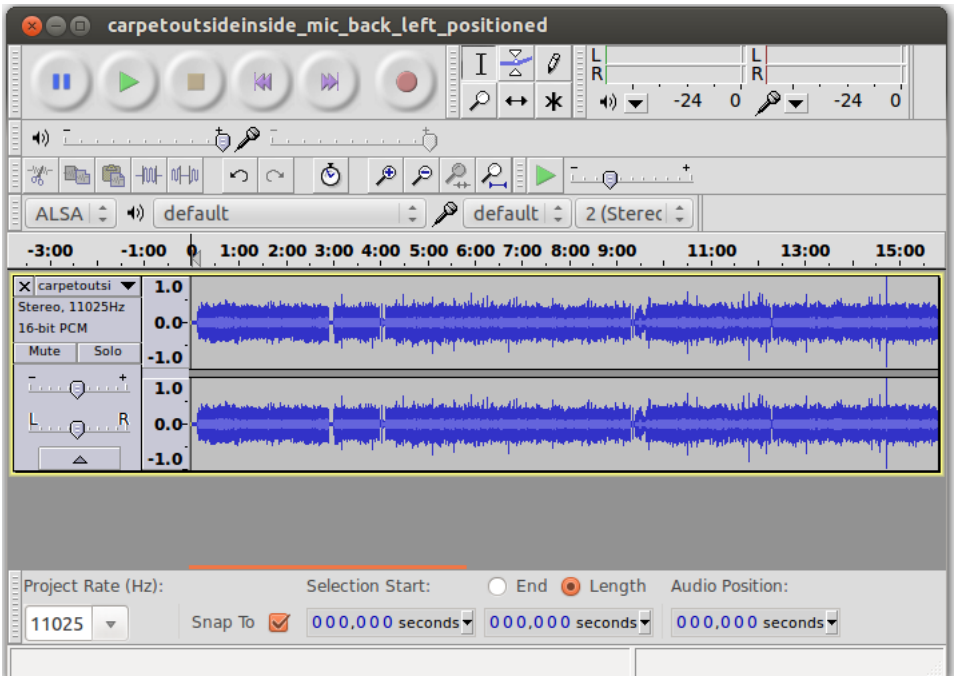


Figure 4.4: ***Audacity.***

In audacity we can modify the preference in edit menu such as device for recording, project frequency, channels (either mono, stereo or more than 2 channels). In our case we are using one microphone, so we record using one mono channel. Monophonic (mono) sound is the most basic sound format and uses mostly a single microphone for recording and usually uses only one loudspeaker for reproduction. In addition, the signal doesn't have any level or phase information and everyone hears the same sound. In contrary, stereophonic (stereo) sound is a reproduction of sound that exploits two or more independent audio channels so that we can have the feeling of sound heard from different directions. Stereo recording is accomplished with a carefully positioned two or more than two special microphones. Similarly, the reproduction is enhanced with a carefully placed loudspeakers. Some of the specific settings used for the recording are shown in Table 4.1.

Host	Channel	Sample rate	Sample format	Device recording
ALSA	1(mono)	11025Hz	16 bit PCM	default

Table 4.1: Audacity specific setting.

Note that Advance Sound Linux Architecture (ALSA) is a linux kernel component which provides device drivers for sound cards.

So as to accept data using external microphone, the input sound setting of the laptop is changed. Thus, we select the microphone usb headset in the input sound setting as in Figure 4.5. After adjusting the setting, we are ready to record sound.

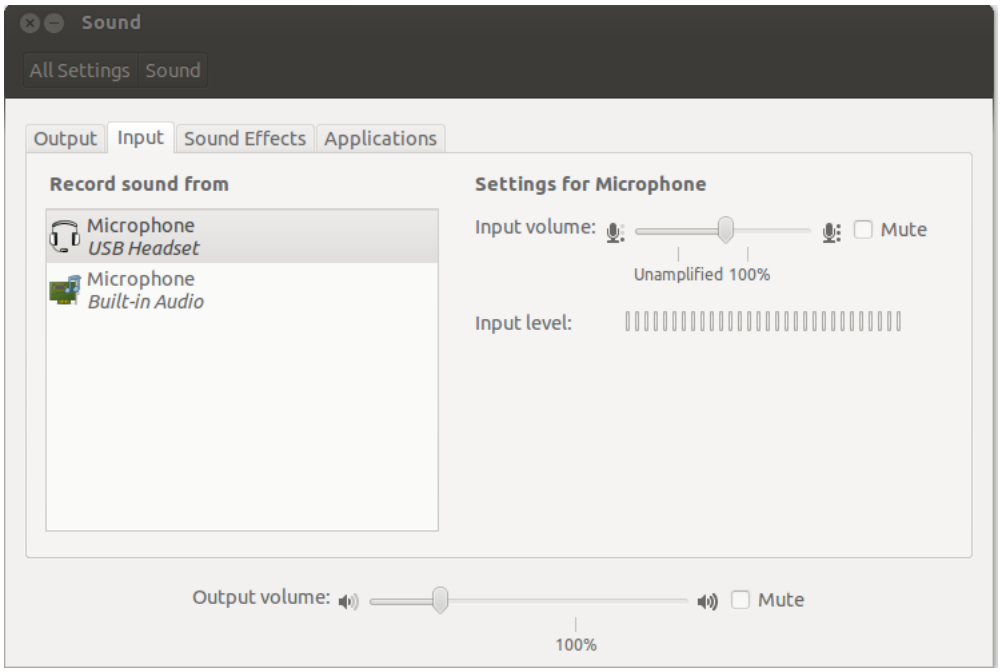


Figure 4.5: ***Sound setting.***

After booting or turning on the robot using the power button switch, we have set the robot to launch basic drivers automatically. After 15 second time delay, the driver is started. The motion of flippers indicate that the drivers are started and the drivers look for the Joystick. Once connection is created between the joystick receiver and transmitter, we run it manually using the logitech cordless rumblepad2 as shown in Figure 4.3b. In order to turn on the receiver, we press the button on it and a green light flashes on and off.

4.3 Position of microphone

This section discusses how to determine the best position for the microphone on our robot so as to minimize the effect of background noise on classification. Initially, the planned system setup for recording sound was to put different microphones on the robot and see the effect of noise originating from robot's CPU, fans and wheels. However, we couldn't record multichannel audio at once due to unavailability of the device that is required for multichannel recording. We have instead recorded sound putting one microphone at different parts of the robot in different circumstances and visualize the effect of noise. The microphone locations nominated for trial

were the left and right front position, center position, left and right back position of our robot. In all the suggested scenarios the microphone was near ground, where the robot terrain interaction occurs, in order to minimize errors as much as possible and at the same time record accurate sound. Actually while running the robot, the noise from its CPU and fan was insignificant. Since both back positions are a bit farther from the small noise coming from the embedded pc, they provide a little bit better sound quality compared to the other positions. So in our data collection, the microphone was mounted in the back position. Furthermore, since the effect of background noise such as noise from the embedded pc was not noticeable, we didn't use any filter mechanism in implementation of the algorithms. In the experiments, 16 bit acoustic data were collected at a frequency of $11025Hz$.

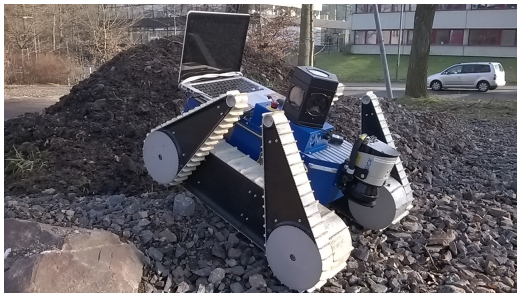
4.4 Data collection

The data collection was performed at the campus of the Saarland University, Saarbruecken, Germany. The data were collected using the Elecom table microphone mounted on the robot that was connected to the laptop using a USB adapter. The microphone was placed on the back of the robot near ground to capture robot terrain interaction sound and to minimize the effect of background noise. The joystick was used to manually drive the robot in different places around the campus. We drive the robot using the default constant speed of the logitech cordless rumblepad2 joystick. The data was collected from four outdoor and one indoor robot terrain interaction, each of them considered as distinct class in our classification task.

The class in an indoor terrain interaction is known as carpet class. The other four classes in the outdoor terrain interactions are categorized as grass, pavement, gravel and sand. The only hazardous terrain type considered for the platform is the sand class. The remaining four interactions are collectively known as benign interactions. In our data collection, we didn't consider other hazardous terrain interactions such as splashing in water and hitting with rocks in order to protect the different sensitive sensors in particular and our robot in general from damage. Data were collected for each class by driving our robot in many flat as well as uncomfortable surfaces. For reliable terrain classification, the locations nominated for running our robot have different characteristics so that there will be variation in the data sets of each class. Furthermore, when the robot traversed in a terrain, the sound varied in accordance with a contact between main track of the robot and road surface.

The sites for gravel terrain as manifested in Figure 4.6 vary from dense structured gravel to dirt road with a wet crushed stones mixed with varying level of mud. In some occasions the robot slipped since some of the places were not only flat but also inclined ones.

As shown in Figure 4.7 the grass terrain varies from neat and wholesome grassland to mixture of grass with dropped down leaves from plants. The concentration of vegetation alters in the grass land. For the pavement terrain as illustrated in Figure 4.8, the locations differ from smooth,



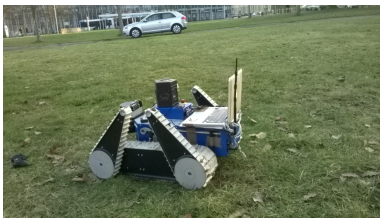
(a) Gravel in hilly surface

(b) Wet gravel

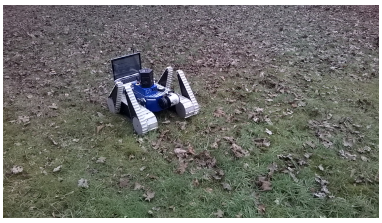
(c) Crushed stones

(d) Fine grained gravel

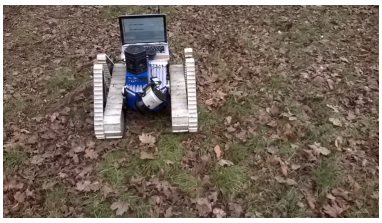
Figure 4.6: ***Examples of various environments employed for the gravel class ranging from fine grained gravel to a crushed stones mixed with varying level of wet mud.***



(a) Wholesome grass land



(b) Grass with leaves



(c) Grass with dense leaves

Figure 4.7: ***Sample snapshots of grass terrain type where the robot traverses in environments varying from dense grass land to a mixture of grass with vegetation.***

durable new asphalt pavement to old roads with fine grained stones or concrete.

The location for carpet terrain type as indicated in Figure 4.9a is inside DFKI, Saarbruecken offices and its surroundings. Similarly, the location for sand terrain type as shown in Figure 4.9b is a construction area in the campus and differs from a fine grained sand to mixture of wet sand with mud and small size stones. After the robot collected 31 samples for sand class, we were not able to continue recording because the belt of main track shifted a little bit from its normal position. The soft sand enters inside the main track belt and creates a problem on rotation of tracks. So, it is highly recommended not to use the platform on soft sand because it has a risk of forcing the track going out of its way.

Note that in all terrain types, the sound is different while the robot tries to turn and change its position to traverse in other direction. This might be because the tracks slip around a single

(a) Smooth asphalt **(b) Old road** **(c) Pavement**

Figure 4.8: ***Examples of different sound collected for pavement class in different environments ranging from rough and old roads to smooth asphalts.***

(a) Carpet terrain **(b) Sand terrain**

Figure 4.9: ***On the left side is a robot in carpet terrain and the right side shows the sand terrain in a construction area near Mensa inside Saarland university.***

static point while there is no phenomenon of normal wheel rotation.

4.5 Software tools

The programming language used for implementation of the software is Matlab. The operating systems utilized are Ubuntu 12.04 and Windows 7. The implementation of terrain classification based on sound using MATLAB is due to its capability to test algorithms right away without recompilation and its ability to read and write audio files in different formats easily using functions such as *wavread* and *wavwrite*. It also has a huge and getting bigger database of built-in algorithms for signal processing and at the same time GMM's training algorithm is already implemented in the *Statistics Toolbox* using *gmdistribution* class. Similarly, we also used the *pattern recognition toolbox* to test a feed forward neural network as classifier. In addition, the written documentation is clear, understandable and supported with plenty examples. It has also online resources such as web seminars. Moreover, MATLAB's built-in graphing tools and GUI builder helps to analyze and interpret our data easily for intelligent decision making process.

In this chapter we looked at the system setup and data collection. In chapters 2 and 3, we theoretically describe the feature extraction techniques and the classifiers used for training the system. In the next chapter, the simulation environment, the different experiments done and

their corresponding results are analyzed.

Chapter 5

Experiments and results

Chapter 2 and 3 gave a general description of feature extraction techniques and classification schemes. In this chapter, we present the different experiments accomplished and analyze the corresponding results. Initially, we illustrate the parameters used which are specific to each implementation and explain some of the GMM training algorithm convergence issues. Then, we briefly introduce the different performance parameters considered. The first experiment accomplished is done using MFCC, GFCC, 6D and 9D feature vectors as feature extraction techniques and GMM as classifier. The precision, recall, accuracy and confusion matrix results are analyzed in detail. The next two experiments are performed to check at what point reduction of training data begins to have an influence on the system performance. The first experiment was done by keeping test data size fixed, varying only the training data size. The second was executed by varying both training as well as test data size. Since in real applications a shorter detection time is better, we conduct another experiment to measure the effect on accuracy when changing the size of test audio samples.
The next section of this chapter describes the experiment done using MFCC, GFCC, 6D feature vector and 9D feature vector as feature extraction techniques and a feed-forward neural network as classifier. We present the performance parameters: confusion matrix, mean-squared error and receiver operating characteristics. Finally, this chapter describes the graphical user interface designed which could help users to interact easily with our system.

5.1 Preprocessing

In chapter 2, we provided the general description of feature extraction techniques used to represent an audio signal. In this section, we explain the parameters employed specific to our implementation. We divided the original files into 4 second chunks, resulting in 324 chunks for each class except for the 'sand' class which has only 31 chunks. The total length of audio for all classes in our database is 88.5 minutes. The audio was recorded in 16 bit PCM mono, with a sampling rate of $11025Hz$. The advantage of working with a small length chunk of sound is

to minimize the computational time during execution. To evaluate the spectrogram for each chunk, the frames comprise 256 samples with 60% overlap. The next frame begins 100 samples after the first one, and the duration of each frame is approximately $25ms$. Then, a hamming window is utilized to minimize the discontinuities at the beginning and end of a frame. The number of coefficients used for the FFT are 256. This provides a spectrum with 256 Fourier coefficients. But in order to decrease execution time, we keep only the first half coefficients. Finally, the power spectrum is computed by squaring the absolute value of FFT coefficients.

5.2 MFCC and GFCC implementations

In MFCC implementation, the first and last triangle filters are centered at $100Hz$ and $5512.5Hz$, respectively. The number of filters employed are 26. Then, multiplying a normalized filter bank with the power spectrum of a signal results in 26 MFCCs. The feature vector of each frame is represented by taking only the $2-13$ coefficients from the output of DCT.

In the GFCC implementation, we substitute the MFCC filterbank by linear phase gammatone filters. We use 23 filters of order 4, so that the multiplication of Gammatone filters with the power spectrum provides 23 GFCCs. Similar to MFCC, the first coefficient of the DCT doesn't contain relevant information and is not included as a feature. So the feature vector for the individual frames consist of only the coefficients $2-23$.

In the implementation, the MFCC features were compared with the GFCC features with regard to the recorded data sets in different terrains. While 13 MFCC coefficients provided enough information, we took 23 coefficients for both feature extraction techniques in order to have a fair comparison.

5.3 GMM classifier training algorithm

We trained our system using the GMM model and training algorithm as implemented in the *Statistics Toolbox*^{TM} software using the *gmdistribution* class. This class fits the data using an *expectation maximization* algorithm. We conducted experiment with the K number of Gaussian components to find the value which gives the best accuracy. The inputs for the *gmdistribution.fit* class are the features extracted using either MFCC, 6D feature vector, 9D feature vector or GFCC and the k number of components. The dimension of the data is n by d , where n represents number of observations and d is dimension of the feature vector.

In few scenarios, the *gmdistribution* converges to a solution which gives a singular or close-singular covariance matrix (ill-conditioned covariance matrices) for one or more gaussian components. A solution having singular covariance matrix is most of the time nonsense. Occasionally, this problem may be avoided by trying another set of initial values. Some of the reasons for this problem could be:

1) The dimension of data is comparatively higher, but there are not enough adequate observations.

2) Having highly correlated features of our data and some or all of them are discrete.

3) Using too many components to fit the data.

We used the following three suggested solutions in order to circumvent getting ill-conditioned co-variance matrices.

1) If solutions with an ill-conditioned covariance matrix are not a big issue, we can utilize *'Regularize'* choice in *gmdistribution.fit* to append a small positive number to the diagonal of every covariance matrix.

2) To utilize an equivalent covariance matrix for each Gaussian component, you can assign the value of *'SharedCov'* to *true.*

3) For every Gaussian component put *'diagonal'* as the value of *'CovType'.*

Note that in a GMM model, adding a new class is not difficult because only the GMM parameters for the class have to be estimated. In addition, there is not significant effect on the classifier and it only involves an introduction of supplementary case to the results of the classifier.

5.4 Performance measurements

Let C be the set of all classes and $c_1 \cdots c_N$ be an element of C. N represents the number of classes and in our case its value is 5. The performance parameters [38] used for the classes in Table 5.1 are described as follows. Note that the definitions are with respect to class c_i and the other classes are symbolized as c_j where $j \neq i$.

		Actual label (Gold standard)	
		c_i (positive)	c_j (negative)
Predicted label	c_i (positive)	True positives	False positives
	c_j (negative)	False negatives	True negatives

Table 5.1: Outcomes of two class prediction.

True positive (TP) is the number of true positive sound files belonging to class c_i and correctly classified as class c_i.

False positive (FP) is the number of false positive sound files tested and incorrectly classified as belonging to class c_i. The parameter indicates an estimation of a sound file which belongs to class c_j and wrongly classified as a member of another class c_i.

True negative (TN) is the number of sound files tested which belong to the labeled class c_j and at the same time is classified as a member of class c_j. Note that TP and TN are correct classifications.

False negative (FN) is the number of sound files tested which are members of class c_i but are recognized as belonging to class c_j.

The field of information retrieval comes up with five core metrics to assess the performance of a classifier. These metrics are precision, recall, accuracy, F-measure and confusion matrix [38].

Precision is the proportion of true positives against the sum of true positives and false positives of a class as in Equation 5.1.

$$Precision = \frac{TP}{TP + FP} \tag{5.1}$$

Recall is expressed as the number of true positives divided by the sum of true positives and false negatives of a class as shown in Equation 5.2.

$$Recall = \frac{TP}{TP + FN} \tag{5.2}$$

While higher precision indicates a lower false positive values, higher recall shows a lower false negative values. When both performance measures have higher score, it shows that the classifier is providing accurate results and at the same time most of them are positive results. If a classifier has higher recall and lower precision, this shows that the results are many but it also indicates most of its predicted labels are incorrect as compared to the actual labels. In contrary, a classifier with higher precision and lower recall indicates small retrieved results but most of its predicted labels are correct in comparison to the correct labels (gold standard).

Accuracy is the ratio calculated by dividing sum of true positives and true negatives to the total number of files in its database as defined in Equation 5.3. A feasible analogy for grasping the principles behind precision and accuracy is to visualize a basketball player shooting baskets. The player will be considered as accurate if he shoots correctly into the basket. But the player will be regarded as precise if he shoots the ball to same location regardless of whether it is close to the basket or not.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{5.3}$$

F measure is a combined measure and it is defined as a weighted harmonic mean of precision and recall measurements. Mathematically it is formulated as shown in Equation 5.4.

$$F = \frac{2 * P * R}{P + R} \tag{5.4}$$

Confusion matrix is mainly used to measure the performance of a classifier by analyzing prediction of the classifier. While the rows of the matrix indicate predicted instances, the columns show actual labels. The diagonal entries of the matrix provide the correct number of classification for each class but the off-diagonal instances represent the number of incorrectly classified objects. Note that the precision, recall and overall accuracy can be computed from the confusion matrix. An example for a confusion matrix of a two class classifier is shown in Table 5.2.

		Actual label	
		Class1	Class2
Predicted label	Class1	A	B
	Class2	C	D

Table 5.2: Confusion matrix example.

A is the number of correct predictions for *class1*. **B** is the number of items classified by mistake as members of *class1* but actually belong to *class2*. **C** represents the number of instances classified wrongly as elements of *class2* though they are members of *class1*. **D** indicates the number of correct predictions for *class2*.

5.5 Experiment using MFCC, GFCC, 6D feature vector, 9D feature vector and GMM

We divided the total set of samples S into two parts, Training and Testing set. Each sound sample has a duration of 4 seconds. While the feature extraction techniques used are MFCC, 6D feature vector, 9D feature vector and GFCC, the classifier employed in the implementation is GMM. The classifier is then trained offline in order to distinguish the five labeled classes from one another based on the value of maximum posterior probability of the feature vectors.

The data from the robot terrain interaction were recorded in different places for each class on our campus. The number of data points used in the training stage and testing stage are shown in Table 5.3. Each sound sample has a duration of 4 seconds and the total size of the audio recorded in our database [1] from all terrain types is $88.5min$. The confusion matrix evaluated

class	Train	Test
Grass	140	175
Pavement	140	175
Gravel	140	175
Carpet	140	175
Sand	20	11

Table 5.3: Number of data points for each class.

using MFCC, GFCC, 6D feature vector and 9D feature vector is given in Table 5.4, 5.5, 5.6 and 5.7.

All methods were evaluated using the test and training data sizes in Table 5.3, and a 5-fold cross-validation. The given values are of this cross validation of training and testing instances of the labeled class. In MFCC, we use 2 gaussians for the GMM model and 23 channels of the Mel filter bank. The first coefficient from DCT is excluded because it contains the mean of input

[1] http://ox6.dfki.de/publications/infostore/10/Bernd%20Kiefer?secret=694424e304f6dbb14f3b1eaff75b9e83

		Actual label				
		Grass	Pavement	Gravel	Carpet	Sand
Predicted	Grass	175	0	1	0	0
	Pavement	0	175	14	0	1
	Gravel	0	0	142	0	0
	Carpet	0	0	0	175	0
	Sand	0	0	18	0	10

Table 5.4: MFCC Confusion Matrix with 95.2% accuracy.

		Actual label				
		Grass	Pavement	Gravel	Carpet	Sand
Predicted	Grass	156	8	33	3	0
	Pavement	1	113	17	6	0
	Gravel	14	10	104	0	1
	Carpet	4	42	11	123	2
	Sand	0	2	10	43	8

Table 5.5: GFCC Confusion Matrix with 70.8% accuracy.

signal which doesn't have significant audio specific information. The 2-13 cepstral coefficients of MFCC were enough for classifying the four classes except the gravel class. So, to improve the results for the gravel class, the number of features were increased to 23.

For the grass, pavement and carpet using 60 samples in the training were good enough to provide satisfactory results in classification. These three classes have an average precision of 96.4 % and an average recall of 98.3%. In contrast, recall and precision value for the gravel class using the same number of samples in training are 66% and 99%, respectively. However, if the number of samples increases to 140, the recall of the gravel class rises to 81% as shown in Figure 5.2. For the sand class, 20 samples were enough to get a satisfactory (90.9% recall) classification output. In summary, as stated by the confusion matrix in Table 5.4 the accuracy of using MFCC feature extraction technique is 95.2%.

Some of the gravel instances were wrongly classified as sand and pavement instances. This shows us that there is some similarity in the recorded audio of test samples. This might be because of the similarity of fine grained gravel audio samples with audio recorded in rough old asphalt. Some of the gravel data sets were taken on wet soil surfaces which might make a great contribution to classify it by mistake as samples from the sand class.

We also implemented another feature extraction technique called GFCC which is based on Gammatone filter bank. In our implementation, we replaced the filterbank of the MFCCs by

		Actual label				
		Grass	Pavement	Gravel	Carpet	Sand
Predicted	Grass	130	6	20	15	0
	Pavement	8	116	51	4	0
	Gravel	1	11	53	2	1
	Carpet	36	43	42	155	0
	Sand	1	0	10	0	10

Table 5.6: 6D feature vector Confusion Matrix with 65.4% accuracy.

		Actual label				
		Grass	Pavement	Gravel	Carpet	Sand
Predicted	Grass	128	10	29	10	0
	Pavement	12	97	35	3	0
	Gravel	9	33	81	5	1
	Carpet	24	32	23	157	0
	Sand	2	2	7	0	10

Table 5.7: 9D feature vector Confusion Matrix with 66.5% accuracy.

gammatone filters. We used the same training and test data sizes and got an accuracy of 70.8% in accordance with the confusion matrix in Table 5.5. The best number of gaussians for the GMM model was 1 and the number of channels (filters) utilized by the gammatone filterbank were 23. In addition, we have also used 6D feature vector and analyzed the results. In accordance to Table 5.6 its performance, with an accuracy of 65.4%, is the worst as compared to the other feature extraction techniques. Using 9D feature vector, the accuracy rises to 66.5% as shown in Table 5.7. But still the performance is lower than the other features.

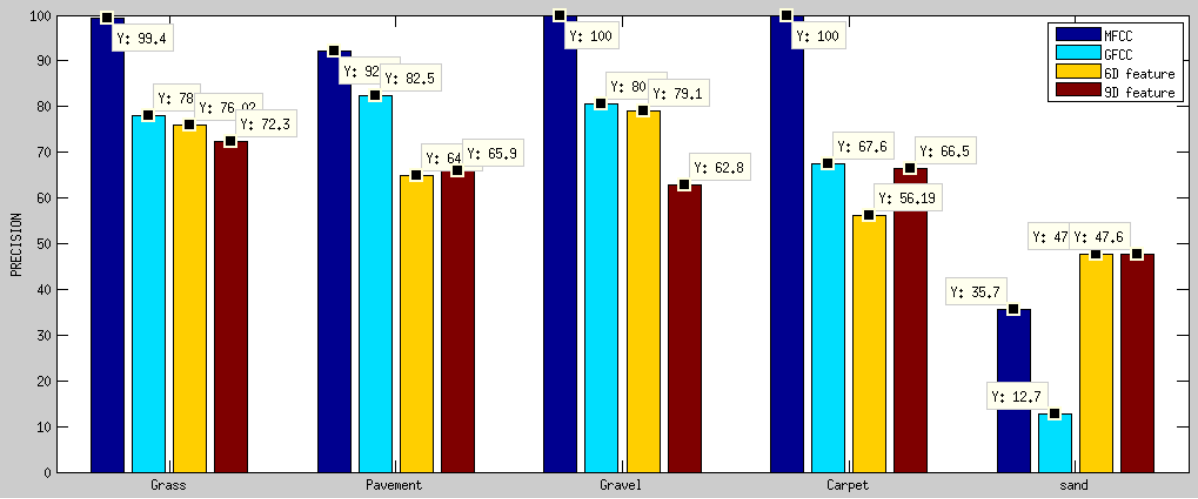


Figure 5.1: ***precision of the five classes.***

The precision of sand recognition in all algorithms is lower as compared to the other four classes. This reduction can be due to the fact that there is less training data samples. However, the recall value in MFCC shows that all relevant instances of sand are retrieved except for one sample. The higher precision value of recognizing grass(99.4%), gravel(100%) and carpet(100%) using MFCC as can be observed in Figure 5.1 indicates that almost all of the retrieved samples are relevant.

Furthermore, higher recall value of recognizing grass, pavement and carpet as shown in Figure 5.2 describes that MFCC algorithm returns all the relevant results. In general, when using GMM as classifier, MFCC has a far better performance than GFCC, 6D and 9D feature vectors. Moreover, the difference in performance of all feature extraction techniques is shown in Figures 5.1 and 5.2.

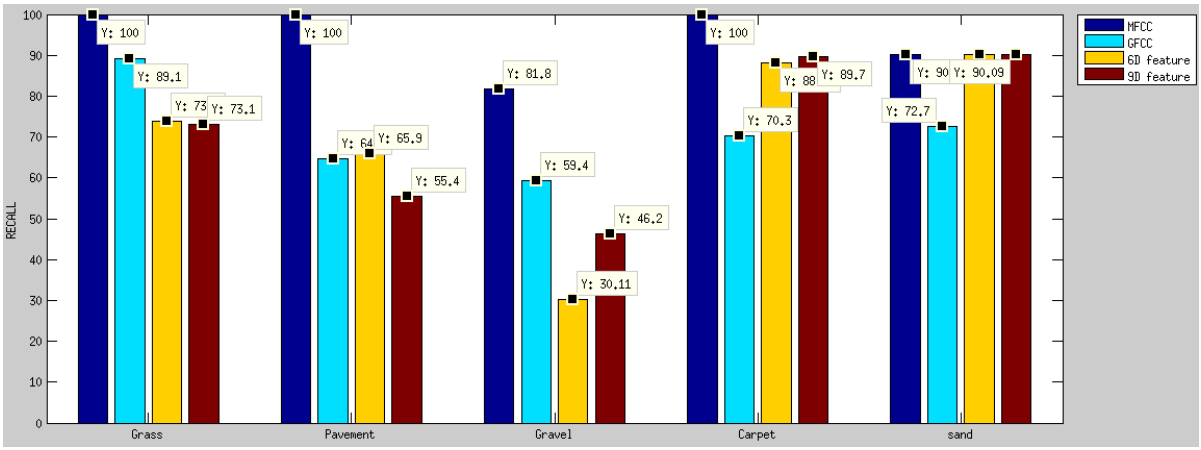


Figure 5.2: ***Recall of the five classes.***

In comparison, Liby and Stentz [3] used a microphone to collect acoustic data sets in benign and hazardous terrain types. They considered three different classes in each terrain type categories. The data collection procedures are different from ours since they use a different vehicle. Using 9D (gianna and shape) feature vector and binary SVMs between each pair of classes as classifier, they achieved an average accuracy of 92% overall classes.

5.6 Accuracy for fixed and variable test data size, and variable training data size

This section explains the two experiments performed in order to check at what point reduction of training data begins to have an influence on system performance. The first experiment shows the change in accuracy found by fixing the test data size to 175 samples and varying the training samples from 20 to 140. As can be observed in Figure 5.3, the X-axis indices A, B, C, D, E, F and G represent using $20, 40, 60, 80, 100, 120$ and 140 training data samples, respectively. While using MFCC, the accuracy varies from 88.9% to 95.6% in accordance to the size of training data. On the other hand in GFCC, the accuracy increments from 49.6% to 71%. As can be seen from Figure 5.3, changing training data size in MFCC does not cause a big difference in the accuracy. But using more than 60 training samples for each class excluding sand are enough for achieving better accuracy. On the other hand, in GFCC small change in training data size has significant difference in the accuracy value. Thus, from our experimental result, GFCC has better performance when the number of training samples are more than 120. We can deduce that as the number of training datasets increases the performance improves as well without over-fitting.

The second experiment reveals change in accuracy perceived by varying both training data size and testing data size. In this experiment, we used the data removed from training for test. As can be seen from Figure 5.3, in MFCC the accuracy varies from 86.4% to 95.6%, while in GFCC

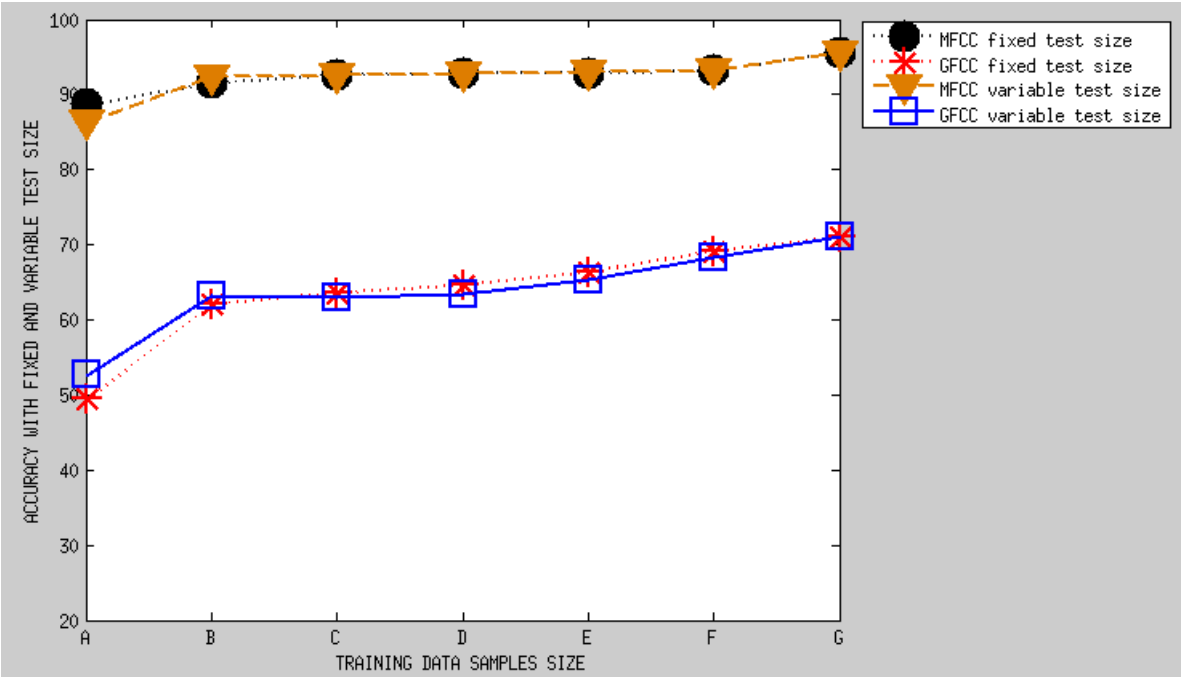


Figure 5.3: ***Accuracy with fixed and variable test data size. The X-axis indices*** A, B, C, D, E, F ***and*** G ***represents using*** $20, 40, 60, 80, 100, 120$ ***and*** 140 ***training data samples, respectively.***

the variation is significant which ranges from 52.8% to 71%. We can deduce that, in MFCC more than 60 training samples and in GFCC more than 120 training samples are enough for attaining satisfactory performance.

5.7 Effect of varying audio duration on accuracy

Another experiment is accomplished to evaluate an accuracy change due to variation in duration of each audio sample. In real applications, it is better to decrease the detection time or duration of audio sample, so that the robot takes an immediate action accordingly for hazardous terrains. Initially, we divided the continuous audio signal into chunks of one second, two seconds and four seconds. Then, our system is trained and tested separately for the three different scenarios and their accuracy is evaluated. We used the same audio length in the corresponding scenarios. It is also possible to train on longer ones and test the sample of varying audio length. It can be seen from the plot in Figure 5.4, while using MFCC, when an audio length varies from one to four second, its accuracy fluctuates in an interval between 93% and 95.6% . We can observe that there is no significant change in accuracy by reducing an audio length from four to one second. Similarly in GFCC, utilizing a bigger audio length has better performance as compared to using a lower audio length but still 5% accuracy difference in human-like labeling is not very big. It will be more realistic to decrease an audio length to 1 sec since it might not be feasible to use 4 seconds audio length during online testing. Moreover, in accordance to the experimental

result, our system still has satisfactory performance using human-like labeling even for smaller audio duration.

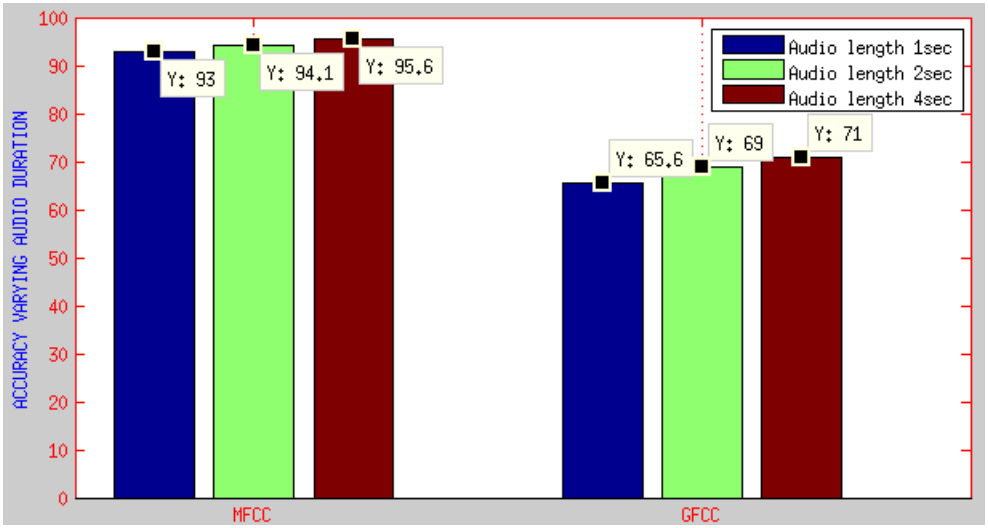


Figure 5.4: ***Accuracy with three different audio length.***

5.8 Neural network experiment

In this experiment, we use the Matlab R2013a neural network pattern recognition toolbox for classification. The feature extraction techniques used are MFCC, GFCC, 6D (gianna) feature vector [3] and 9D (gianna and shape) feature vector [3]. Since lower detection time is feasible for real applications, an audio duration of 1 second is considered. The inputs to our neural network implementation are training data, corresponding target outputs for the training data and number of neurons in each layer. The result of the experiment is a neural network architecture with corresponding weights of all links binding the nodes in each layer calculated by minimizing the mean of squared errors. The implementation has matrices A and B as input to the network. Matrix A consists of training datasets and its dimension is n by p. While n is the length of the feature vector, in our scenario it differs depending on the feature extraction method assumed, and p represents the number of training samples. Matrix B associates the desired output for the corresponding training samples and has a dimension of m by p. Note that m represents number of classes to be classified, in our case it is 4, and p is the number of training samples. We recorded 300 samples for each class and in the implementation p contains 1200 samples.

The dimension of MFCC and GFCC feature extraction techniques for each audio sample is 12∗109 matrix. We tried two different ways to feed these features as inputs to the feed forward neural network (FFNN). In the first place, we have changed the matrix feature vector of each audio sample to a column vector. Thus, the number of inputs passed for both methods is 1308. The second option is to find the mean overall frames of a feature vector for a given audio sample. So every audio file will have its corresponding 12 coefficients as input to the network. Similarly,

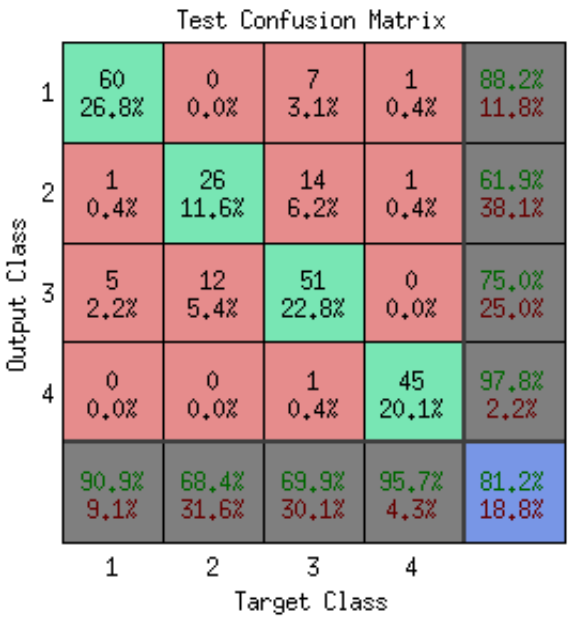
the inputs of gianna and gianna shape feature vectors are 6 and 9, respectively. Furthermore, in this two layer FFNN 10 neurons in hidden layer and 4 output neurons is considered. For every test input sample, only one node in an output layer generates a value of $'1'$. While the remaining other nodes provide an output value of $'0'$. Since the samples collected for sand class are small, we have trained the system only using the remaining four classes. The training algorithm utilized in this experiment is scaled conjugate gradient back-propagation algorithm *(Trainscg)*. The transfer function employed in our scenario for the hidden and output neurons is sigmoid function. We trained the data points using the toolbox and then execute the rest of test samples against the trained network. 70% of the original data are used for training, 15% are validation data and the remaining 15% are testing data.

The confusion matrix using the four feature extraction techniques is shown in Figures 5.5 and 5.6. The green colored squares in the diagonal of the matrix show correct classifications, the red colored squares indicate incorrect classifications. Smaller percentage in the red squares represents fewer misclassification which shows that the system is good enough to classify accurately. The gray squares on the matrices right side and matrices bottom indicate precision and recall measures, respectively. If there are higher wrong classifications, it is recommended to train the network again with a varying number of hidden neurons. Moreover, the overall accuracy of the network is indicated by blue colored squares.

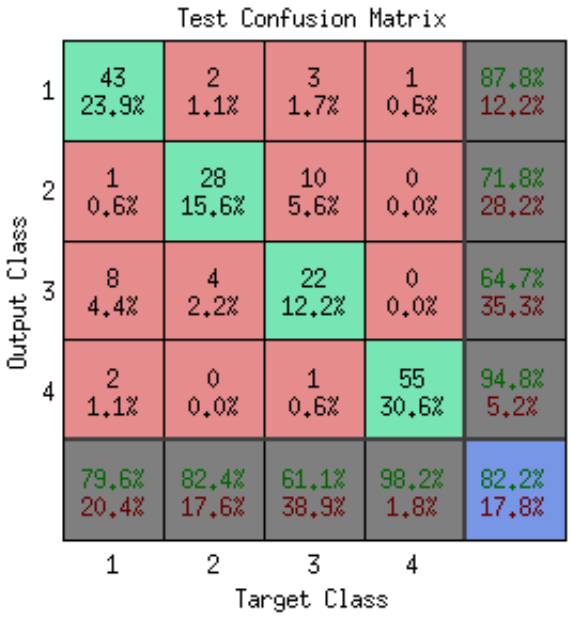
Plot 5.5a shows the confusion matrix using gianna features with an average accuracy of 81.2%. Plot 5.5b shows the slight improvement in accuracy to 82.2% by gianna and shape features. As can be seen from plot 5.5c, GFCC with 1308 inputs has the worst average accuracies as compared to the others. But the accuracy rises to 82.8% when the number of inputs are 12 as shown in Figure 5.6a. In contrast, Figure 5.5d indicates that MFCC is by far the best method with an average accuracy of 93.9% for our scenario. We can observe from plot 5.6b that the accuracy of MFCC doesn't change significantly when 12 number of inputs are used.

The network performance of the system is also measured using mean square error (MSE). This function calculates the performance based on the mean of squared errors. The performance for the three data sets using MFCC can be observed in Figure 5.7a. As can be seen from the plot, the mean of squared error starts initially with a big value and then rapidly decreases to smaller value. When the non-training validation set error passes a minimum threshold, the training of its network stops. In Matlab, the default stopping criteria is when the validation error doesn't decrease for 6 consecutive epochs. An epoch represents the handing over of all training samples to the experiment.

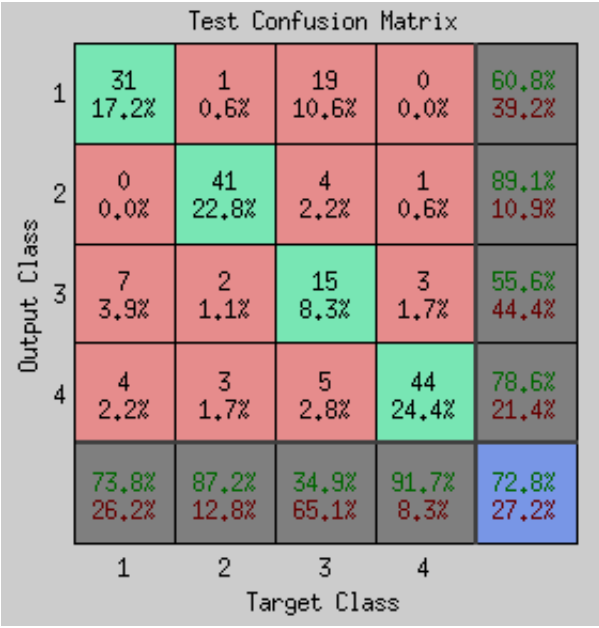
The other performance measure is receiver operating characteristics (ROC) plot. This figure represents the relation between true positives and false positives for all terrain types. As can be observed from the diagram 5.7b, more than 85% of the four terrains are correctly classified before very small percentage of the terrains are misclassified as other terrains. The movement of the line from bottom left corner to top left corner then to top right corner, indicates that our



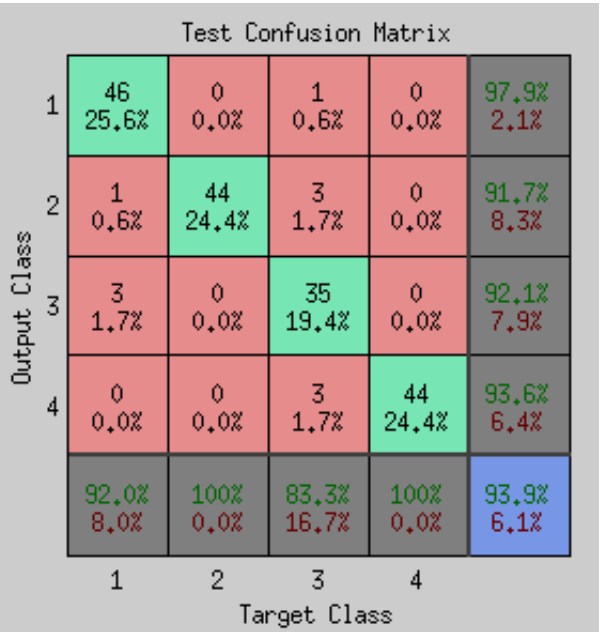
(a) 6D confusion matrix



(b) 9D confusion matrix



(c) GFCC confusion matrix

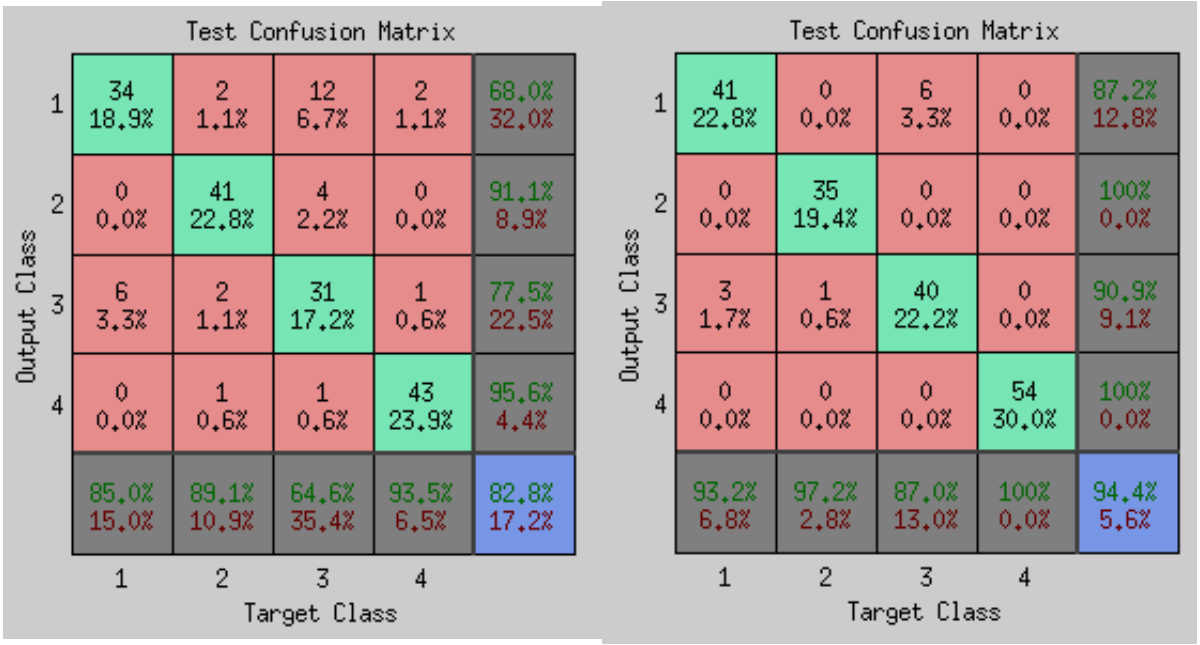


(d) MFCC confusion matrix

Figure 5.5: ***The indexes 1, 2, 3, 4 represents grass, pavement, gravel and carpet class, respectively. Plots (a), (b), (c) and (d) show gianna, gianna and shape, GFCC with 1308 inputs and MFCC with 1308 inputs confusion matrix, respectively.***

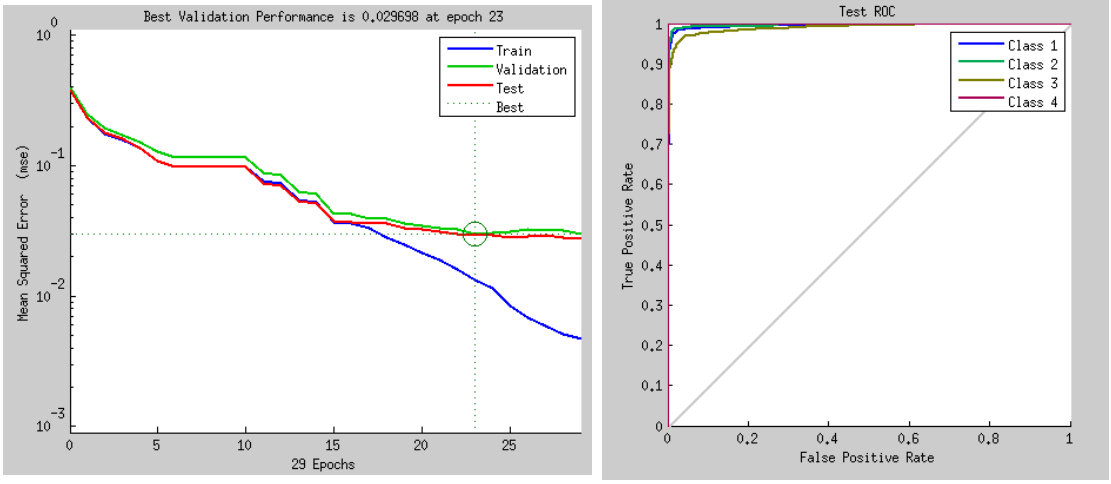
classifier achieved better performance.

When the network is not performing properly, we can use some techniques to enhance the results. The first option is to initialize both its network and training again. Note that the parameters are distinct every time we initialize the network and its solution might also change



(a) GFCC confusion matrix **(b) MFCC confusion matrix**

Figure 5.6: ***The indexes 1, 2, 3, 4 represents grass, pavement, gravel and carpet class, respectively. Plots (a), (b) show GFCC with 12 inputs and MFCC with 12 inputs confusion matrix, respectively.***



(a) MSE using MFCC **(b) ROC using MFCC**

Figure 5.7: ***The legend class 1, 2, 3 and 4 represents class grass, pavement, gravel and carpet, respectively.Plots (a) shows MSE with feed forward back propagation network using MFCC. plot (b) represents test ROC curve for MFCC using FFNN.***

accordingly. The second approach is to increase the number of neurons in hidden layer. The third alternative is to attempt using different training algorithms. The last option could be to try utilizing more training data because this might help in designing a network that generalizes better to unseen data. We used the first, second and last options to improve the performance of the network. The detailed confusion matrix plots, MSE plots and ROC curves for the other feature extraction techniques can be seen in appendices A, B, C and D.

5.9 Graphical user interface

The graphical user interface developed helps users to interact easily with our system and make the program easier to use. The typical advantages of GUI are for making user-friendly, speed up user's work, more attractive and fancy for non technical people. Generally, it looks more professional when it is properly developed. However, it might have some disadvantages if it is not properly built. In addition, it is slower than a command line interface and demands more memory resources as compared to non-graphical one.

Users can select an audio recorded in different terrains from the popup menu as modeled in Figure 5.8. It is designed to be situated on the left side of our GUI and shows list of audios that can be utilized for testing. Our system manipulates the selected audio as input to display predicted terrain type to screen. The display positioned in middle helps to unveil predicted terrain type. On the right side, there are four push buttons utilized for training and testing our data using both MFCC and GFCC feature extraction techniques. Initially, we train the system by clicking either *'TRAIN MFCC'* or *'TRAIN GFCC'* buttons. Then, by pressing either *'TEST MFCC'* or *'TEST GFCC'*, it compares an audio nominated from popup menu with the trained GMM model for each class and in return displays the predicted terrain to screen. Finally, the other display, which is found below the predicted terrain type screen, informs whether the estimation was correct or wrong.

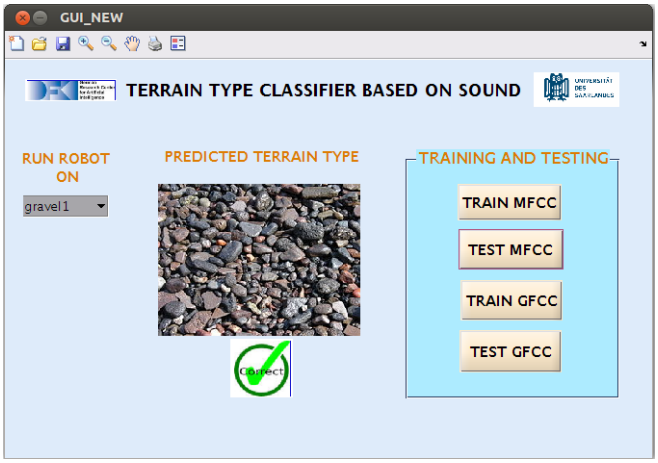


Figure 5.8: ***Graphical user interface.***

Chapter 6

Conclusion and Future work

This chapter summarizes the results and concludes the work done in the thesis. In addition, it presents the possible future work that can be accomplished as extension of our system.

This thesis proposes classification of different terrain types based on robot terrain interaction using only acoustic data. We used human-like labeling of the terrain for verbal description of the robot's activities. The presented terrain type classification is based on the analysis of sound recorded by a microphone sensor with GMM and feed forward neural network classification schemes. We implemented MFCC and GFCC feature extraction techniques and compared the performance to the 6D and 9D feature vectors suggested by Liby and Stentz [3]. We survey also other related work on terrain type classification based on acoustics and other features. Liby and Stentz attained an average accuracy of 92% across all classes using 9D (gianna and shape) feature vector and support vector machine (SVM) as classifier. They considered the interactions in grass, pavement, gravel, water, hard objects and slippery terrain types.

The five terrain types considered in our implementation are grass, pavement, gravel, carpet and sand. However, due to the occurrence of technical problem during data collection when our robot traverses on sand, it was not possible to collect as many sand samples as for the other four classes. This is no real limitation, since it is not recommended to run the robot on sand surfaces anyway.

To train the classifier on a variety of terrain instances of each type, the audio samples for each class were collected in a variety of places. Then, the classifiers were trained offline in order to classify given test samples. We varied the test sample length ranging from 1 to 4 sec to see if the method could be used in real time fashion. The experimental results indicate that MFCC gives by far better performance than the other feature extraction methods in both GMM as well as FFNN classifiers. However, for features other than MFCC, the accuracy improves when the FFNN classifier is employed. We achieved an overall accuracy of 93.5% for 1 second test samples, which rises to 95.6% when the audio duration is increased to 4 sec. The gravel performance was not as good as the other terrains and sometimes was confused with sand and

pavement. We observed also that the performance improves with an increase in the number of training data points without over-fitting. We believe that enlarging the size of database for each class will help to attain a more generic classifier. These are very promising results and it shows that acoustics is an interesting technique for terrain perception of a tracked robot.

It is possible to estimate the reliability of the classification by comparing the resulting maximum posterior probabilities from the GMMs of each class. This can be done by evaluating the confidence level of the probabilities. The classification would be more reliable if the confidence level is higher.

Even though we attained good accuracy using five terrain types, it is recommended to include additional terrain types as well for a future study. Adding a new class only means adding a GMM model for the new class and an introduction of an additional possible output in the classifier. According to different studies, speed has a huge impact on the acoustic signature of the robot terrain interaction. The effect of speed was minimized by driving the robot manually at constant speed for all of the experiments. Although it was out of the scope of this project/thesis to experiment with data sets collected at different speeds, future work can be accomplished to visualize the real impact of driving at different speed on performance of terrain classification.

Since one sensor can be reliable in one factor and unreliable in the other factor, it is recommended to use fusion of different sensor modalities. Future work could be extended on how to combine different feature vectors for better prediction of gravel class in particular. This combination of features could be from either one sensor or different sensor modalities. The area of sensor data fusion is a hot research area at this time and it is playing a pivotal role in mobile robot interactions. Furthermore, we couldn't record multichannel audio due to unavailability of the device that allows multichannel recording. Future work could analyze its influence on the quality of the classification. In addition, since the effect of noise from the embedded pc on robot was not significant in our implementation, we didn't use any noise extraction techniques. For further studies, we suggest to study the internal as well as background noises and try to evaluate more rigorously whether there is an effect on accurate terrain classification.

As extension to our system, one could possibly also consider doing a human classification experiment and see how well people can guess the terrains from the sounds only. That would give a sense of how well the classification performs compared to human hearing. Using the proposed feature extraction techniques in the thesis, one could also try to determine the correction coefficients for the adaptive traversability approach developed by Reinstein, Kubelka and Zimmermann [5].

List of Figures

List of Tables

Bibliography

[1] Ivana Kruijff-Korbayova, Francis Colas, Mario Gianni, Fiora Pirri, Joachim de Greeff, Koen Hindriks, Mark Neerincx, Petter Ogren, Tomas Svoboda, Rainer Worst, "TRADR Project: Long-Term Human-Robot Teaming for Robot Assisted Disaster Response", Springer-Verlag Berlin Heidelberg 2015.

[2] Kruijff G, Colas F, Svoboda T, van Diggelen J, Balmer P, Pirri F, Worst R (2012) Designing intelligent robots for human robot teaming in urban search and rescue. In: Proceedings of the AAAI 2012 spring symposium on designing intelligent robots

[3] J. Libby and A. J. Stentz, Using Sound to classify Vehicle-Terrain Interactions in Outdoor Environments, Robotics Institute Carnegie Mellon University Pittsburgh,PA 15213.

[4] C. A. Brooks and K. Iagnemma, "Self-Supervised Terrain classication for Planetary Surface Exploration Rovers," Journal of Field Robotics, vol. 29, no. 1, 2012.

[5] M. Reinstein, V. Kubelka and K. Zimmermann, "Terrain Adaptive Odometry for Mobile Skid-steer Robots",2013 IEEE International Conference on Robotics and Automation (ICRA) Karlsruhe, Germany, May 6-10, 2013.

[6] D. Hoiem and R. Sukthankar, SOLAR: Sound Object Localization and Retrieval in Complex Audio Environments, in IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2005.

[7] M. Menna, M. Gianni, F. Ferri, and F. Pirri, Real-time autonomous 3d navigation for tracked vehicles in rescue environments, in IROS, 2014.

[8] R. S. Durst and E. P. Krotkov, Object classication from Analysis of Impact Acoustics, in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 1995.

[9] A. Amsellem and O. Soldea, Function-Based classication from 3D Data and Audio, in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2006.

[10] E. J. Coyle and E. G. Collins, "A Comparison of classifier Performance for Vibration-based Terrain classification," in 26th Army Science Conference, 2008.

[11] A. Plinge, R. Grzeszick and G.A.Fink, “Bag of features approach to acoustic event detection”, Dortmund, Germany.

[12] C. Wellington and A. Stentz, “Online Adaptive Rough-Terrain Navigation in Vegetation,” in IEEE International Conference on Robotics and Automation (ICRA), 2004.

[13] M. V. Scanlon, “Acoustic Sensor For Health Status Monitoring,” in IRIS Acoustic and Seismic Sensing, 1998.

[14] G. Cohn, S. Gupta, J. Froehlich, E. Larson, and S. N. Patel, “GasSense: Appliance-Level, Single-Point Sensing of Gas Activity in the Home,” in IEEE International Conference on Pervasive Computing and Communications (PerCom), 2010.

[15] Philipos G. Loizon, “Mimicking the human ear”, IEEE signal processing magazine, September 1998.

[16] S.Biswas” MFCC based Face Identification” Titech Japan.

[17] S.Gupta, J.Jaafar, W.Fatimah w.Ahmad and A.Bansal , “FEATURE EXTRACTION USING MFCC ”, Signal and Image Processing : An International Journal (SIPIJ) Vol.4, No.4, August 2013.

[18] Heittola, T. (2004). Automatic classification of Music Signals. Master Thesis.

[19] C.M Bishop Pattern Recognition and Machine learning, Springer 2006.

[20] The HTK Book (for HTK Version 3.4).

[21] P. I. M. Johannesma, ”The pre-response stimulus ensemble of neuron in the cochlear nucleus,” Proceedings of the Symposium of Hearing Theory, 1972.

[22] W. H. Abdulla, ”Auditory Based Feature Vectors for Speech Recognition Systems”, In Advance in Communication and Software Technologies, N.E. Mastorakis and V.V. Kluev, Editor, WSEAS Press, pp 231-236,2002.

[23] E. de Boer and H. R. de Jongh, (1978) ”On cochlear encoding: potentialities and limitations of the reverse-correlation technique,” J. Acoust. Soc. Am., 63, 115-135.

[24] R. Schlüter, I. Bezrukov , H. Wagner, H. Ney,“Gammatone features and feature combination for large vocabulary speech recognition ”, RWTH Aachen University, Aachen, Germany.

[25] Yushi Zhang and Waleed H. Abdulla, “Gammatone Auditory Filterbank and Independent Component Analysis for Speaker Identification systems ”Department of Electrical and Computer Engineering The University of Auckland, Private Bag 92019, Auckland, New Zealand.

[26] A. G. Katsiamis, Student Member, IEEE, E. M. Drakakis, Member, IEEE, and R. F. Lyon, Fellow, IEEE “Practical Gammatone-like Filters for Auditory Processing”.

[27] E.Garg and M.Bahl, " Emotion Recognition in Speech Using Gammatone Cepstral Coefficients ", iternational Journal of Application or Innovation in Engineering and Management (IJAIEM) ,Volume 3, Issue 10, October 2014.

[28] G.Peeters. A large set of audio features for sound description (similarity and classification) in the cuidado project. April 2004.

[29] T. Giannakopoulos, K. Dimitrios, A. Andreas, and T. Sergios, "Vi-olence Content classification Using Audio Features," in Hellenic Artificial Intelligence Conference, 2006

[30] http://www.physik.uniwuerzburg.de/ praktiku/Anleitung/Fremde/ANO14.pdf.

[31] Chuong B. Do, "The Multivariate Gaussian Distribution", October 10, 2008.

[32] McCulloch, W. and W. Pitts(1943). A Logical Calculus of the Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biophysics, Vol. 5, pp. 115-133.

[33] http://www.explainthatstuff.com/introduction-to-neural-networks.html

[34] Vladimir Vapnik, Corinna Cortes. "Support vector networks," Machine Learning, vol. 20, pp. 273-297, 1995.

[35] Christopher J.C. Burges. "A Tutorial on Support Vector Machines for Pattern Recognition", Data Mining and Knowledge Discovery 2, 121-167, 1998

[36] "Introduction to Support Vector Machines", OpenCV 2.4.11.0 documentation.html.

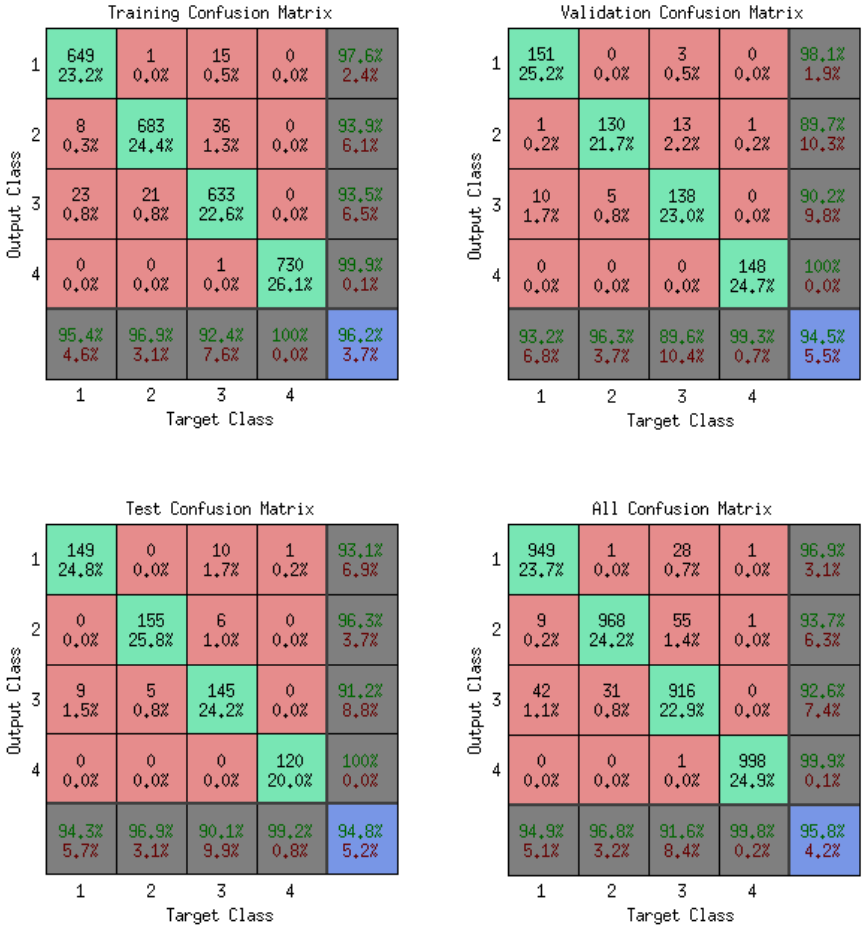
[37] https://trac.dfki.de/talkingrobots/wiki/NIFTi.robot overview.

[38] Ian H. Witten, Eibe Frank, Mark A. Hall "Data Mining: Practical Machine Learning Tools and Techniques, Third Edition", pp. 164, 175-176.

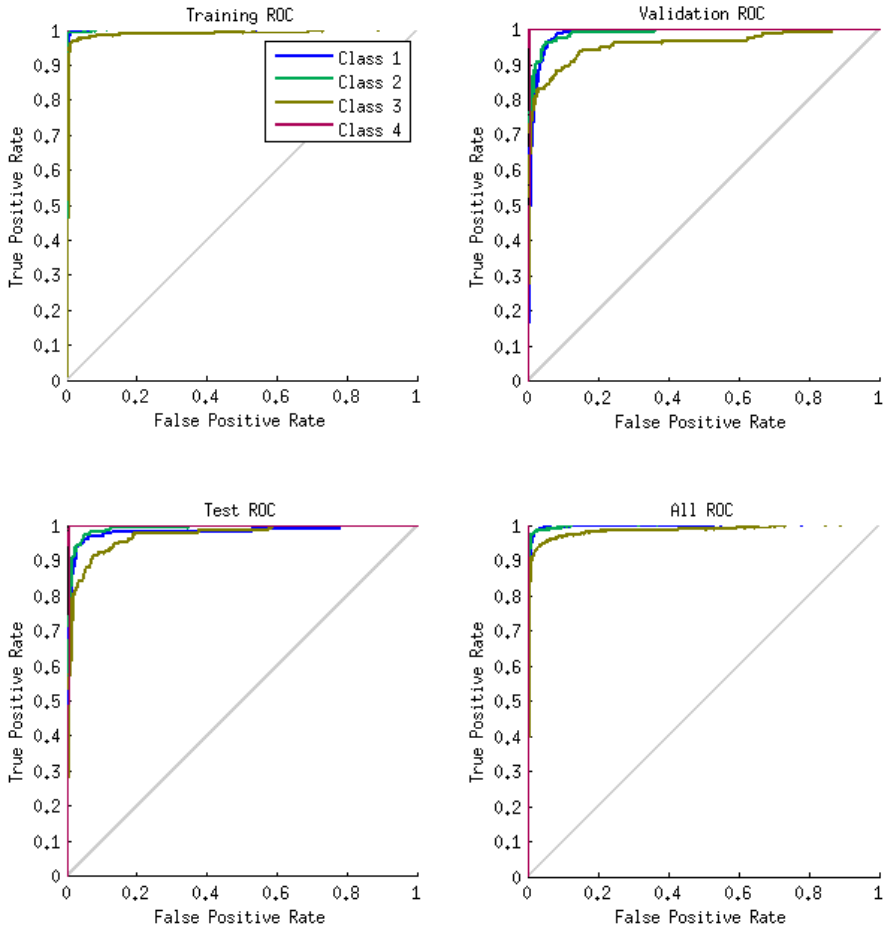
Appendix A

MFCC using FFNN

A.1 MFCC confusion matrix using feed forward neural network



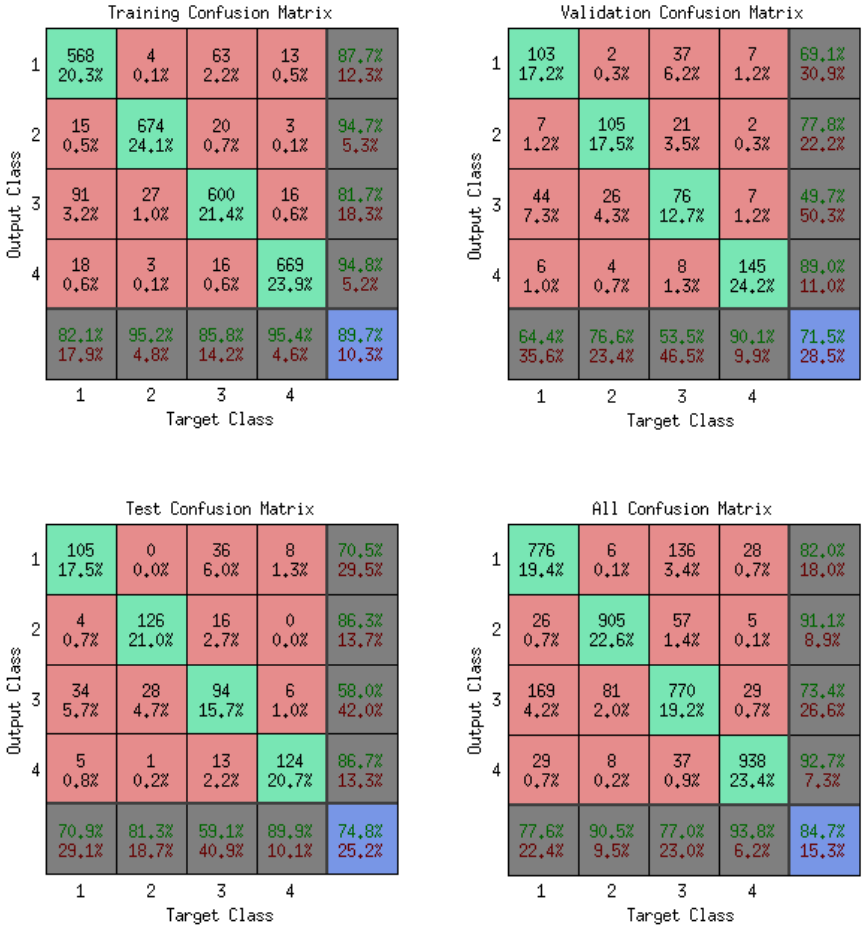
A.2 MFCC ROC using feed forward neural network



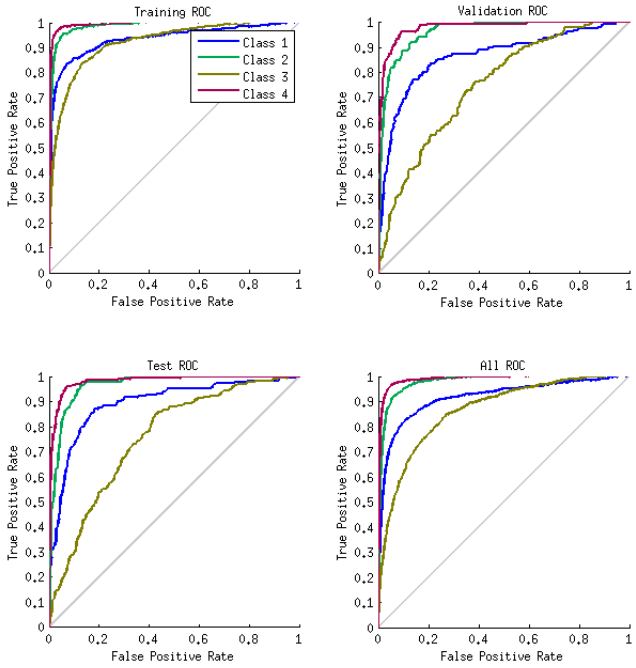
Appendix B

GFCC using FFNN

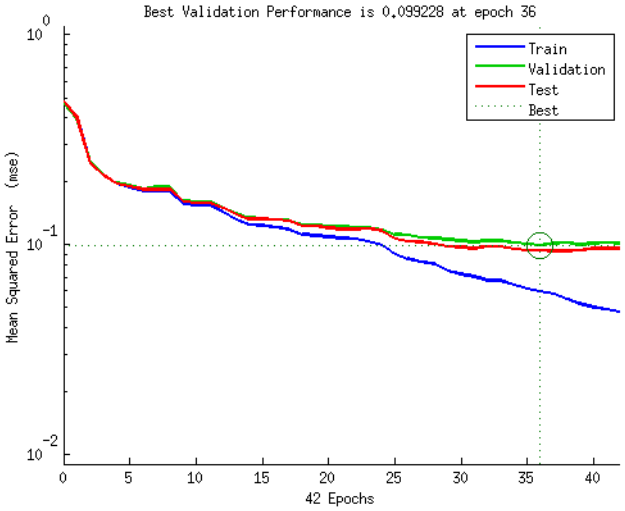
B.1 GFCC confusion matrix using feed forward neural network



B.2 GFCC ROC using feed forward neural network



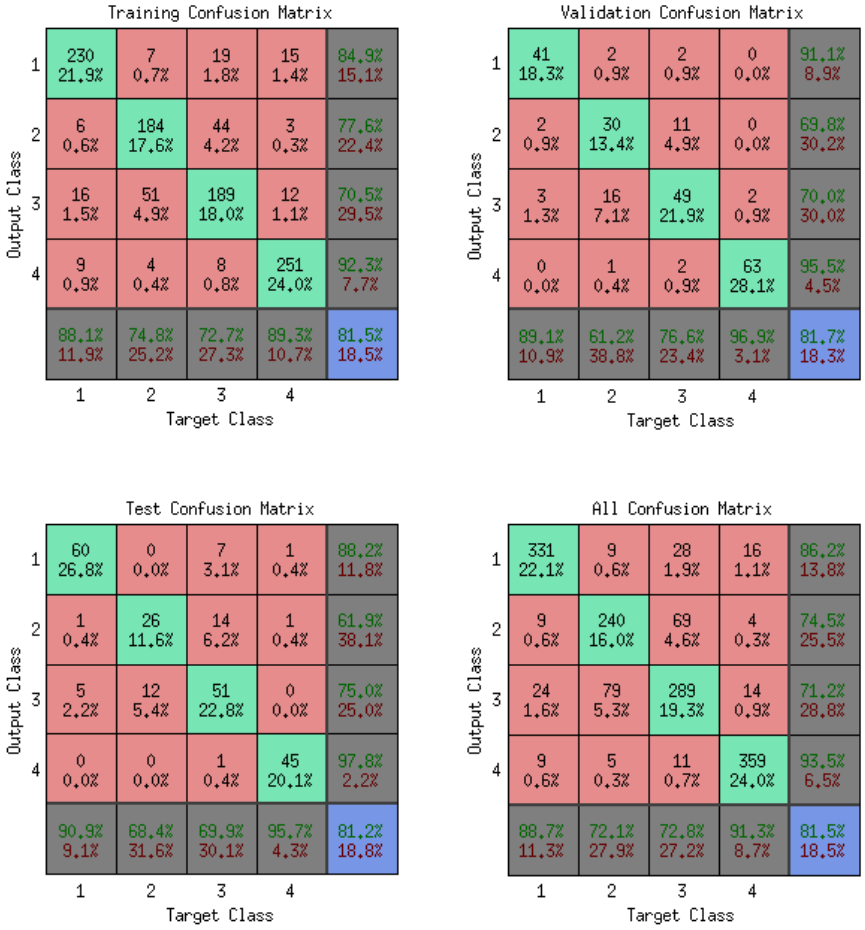
B.3 GFCC MSE using feed forward neural network



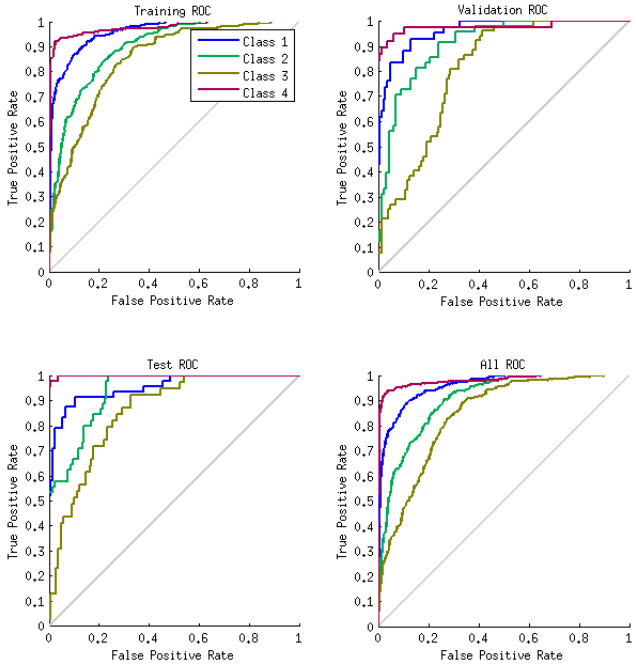
Appendix C

6D feature using FFNN

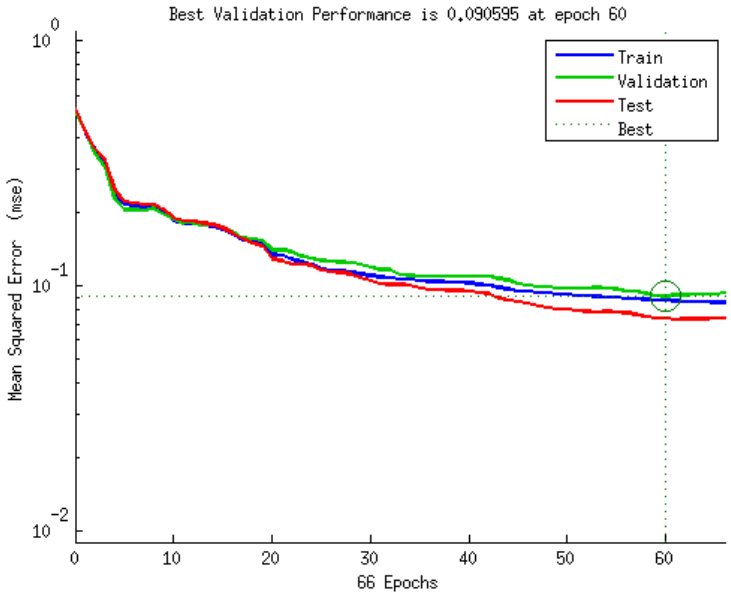
C.1 Gianna confusion matrix using feed forward neural network



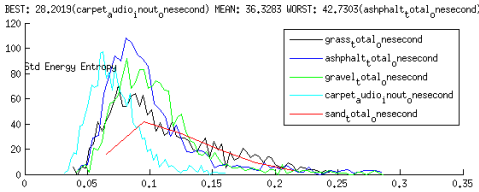
C.2 Gianna ROC using feed forward neural network

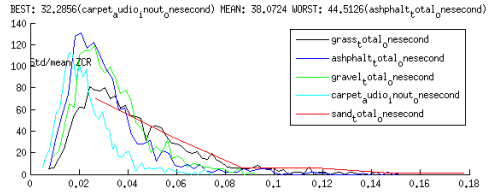


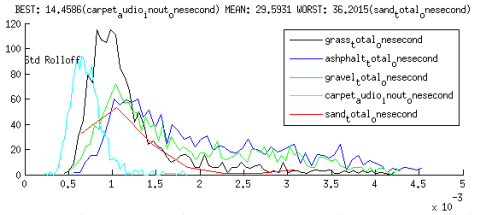
C.3 Gianna MSE using feed forward neural network

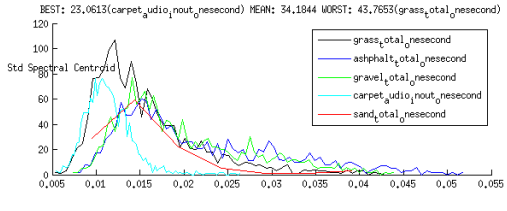


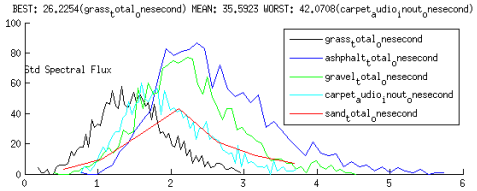
C.4 Gianna feature vectors plots

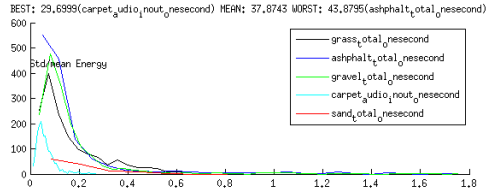








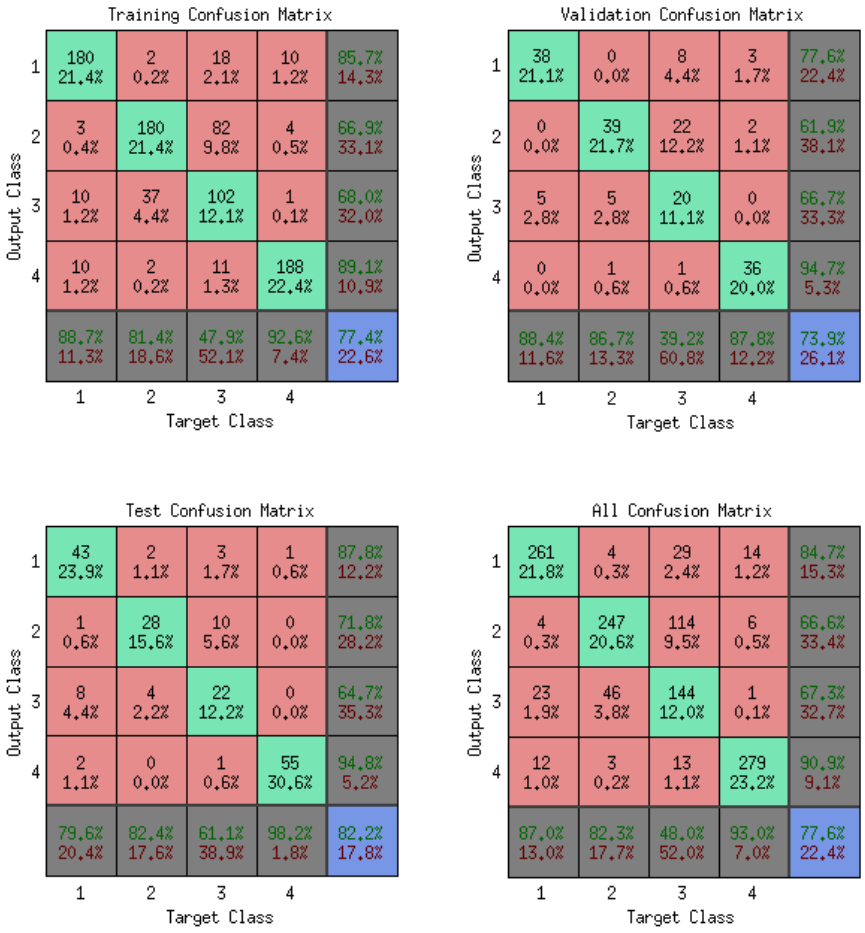




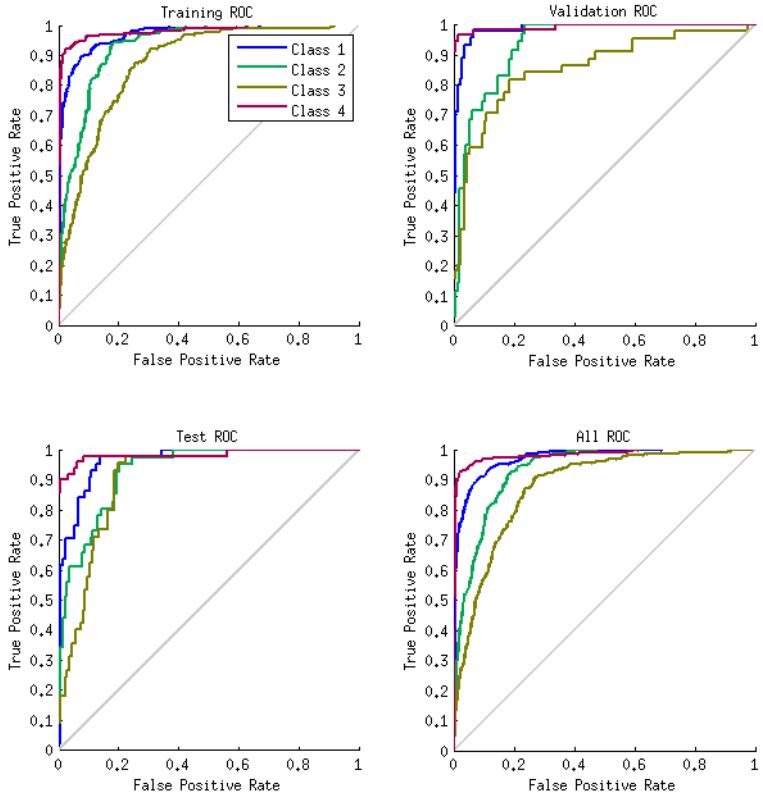
Appendix D

9D feature using FFNN

D.1 Gianna and shape confusion matrix using feed forward neural network



D.2 Gianna and shape ROC using feed forward neural network



D.3 Gianna and shape MSE using feed forward neural network