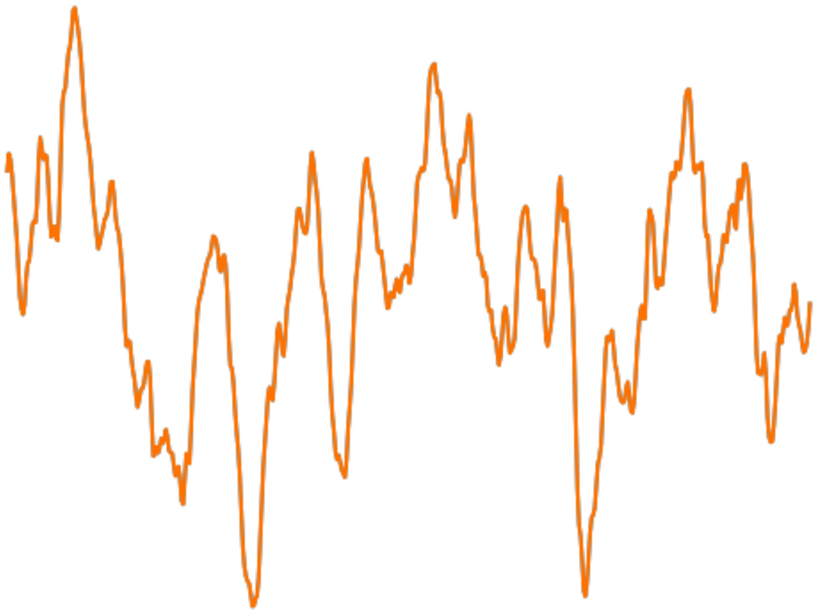
CHALMERS
UNIVERSITY OF TECHNOLOGY



Classification of soundscapes in real time

Master's thesis in Biomedical Engineering Signal and Control

EMELIE LILLIECREUTZ

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Master's thesis 2019

Classification of soundscapes in real time

EMELIE LILLIECREUTZ

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Signal processing and Biomedical engineering
Chalmers University of Technology
Gothenburg, Sweden 2019

Classification of soundscapes in real time
EMELIE LILLIECREUTZ

Supervisor: Abel Gladstone Mangam, 3M Peltor
Examiner: Sabine Reinfeldt, Department of Electrical Engineering

Master's Thesis 2019
Department of Electrical Engineering
Signal processing and Biomedical engineering
Chalmers University of Technology
SE-412 96 Gothenburg

Cover: Plot of industrial noise signal

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2019

Classification of soundscapes in real time
EMELIE LILLIECREUTZ
Department of Electrical Engineering
Chalmers University of Technology

Abstract

3M™Peltor™develops active hearing and communication hearing protectors for a variety of fields. The hearing protectors are not only protecting the user against hearing loss, but also helping to perceive sounds from the surroundings and to communicate with colleagues. Since hearing protectors are used at such different occasions, the soundscapes are variant. However, the hearing protectors function equally in all environments, which means that they may not always function optimally. In order to do that, the hearing protectors should be adapted depending on the surrounding soundscape. In that case, the soundscape must be identified and classified.

The aim of this project was to find a possible solution for adapting the hearing protectors to different soundscapes. Interesting soundscapes were defined based on user areas. Furthermore, it was evaluated which soundscapes that were possible to identify and classify. Lastly, they were classified. The k Nearest Neighbour (k-NN) algorithm was used for classification. The classification was based on time and frequency domain audio features. Furthermore, the ability to work in real time was evaluated by a complexity analysis.

This project shows that k-NN is a method that can achieve good results when classifying signals of classes that are present in the data set. However, the method works quite poorly when classifying signals of other classes than those in the data set. Classification of ’Speech’, ’Industrial noise’ and ’Alarm’ reached an accuracy of 89%. This result was obtained by using the combination of mel frequency cepstral coefficients, zero-crossing rate and energy, together with $k = 1$ and a data set with 45 samples, i.e. 15 samples from each class.

Furthermore, the ability to work in real time is dependent on the size of the data set, the number of features used and the frame size. However, based on an approximate computational time of 1.86 ms for audio frames of size 512, it is indicated that is should be possible to implement this in real time.

Keywords: Audio features, audio classification, k-NN, k Nearest Neighbours, complexity analysis

Acknowledgements

First and foremost, I would like to thank my supervisor Abel Gladstone Mangam at 3M Peltor for supporting me through this project with guidance, advice and interesting thoughts. A thank should also be given to 3M Peltor for giving me the opportunity to work with this project.

I also want to thank my examiner Assoc. Prof. Sabine Reinfeldt at the Department of Electrical Engineering, who has given me support and assured me that the project was on the right track.

Emelie Lilliecreutz, Gothenburg, June 2019

Contents

List of Figures

List of Tables

1
Introduction

This chapter presents the background, purpose and objectives of the study. Furthermore, the delimitations are declared.

1.1 Background

A landscape is the composition of all visible features within an area and its landforms [1]. These features may both be created by humans and be natural. Furthermore, the sound energy created by a landscape is called a soundscape. It is created by the combination of three distinct sound sources: geophonies, biophonies and antrophonies. Geophonies are nonbiological sound energies, which are sounds from natural phenomena such as wind, waves or thunder, while biophonies are the sounds from animal vocalization such as voices or bird song and antrophonies are all the sounds from technical devices, such as engines or industries [2].

3M™Peltor™develops active hearing and communication hearing protectors for a variety of fields, which have a wide variety of soundscapes. Among others, industry workers, hunters, police officers and soldiers use the hearing protectors. They are not only protecting the user against hearing loss, but also helping to perceive sounds from the environment and to communicate with colleagues [3].

3M™Peltor™has made it possible to combine communication and attenuation by using a level dependent function. The level dependent function is an electronic sound reproduction system, which limits the sound level inside the ear muffs so that it never exceeds 82 dB(A) [4]. The A-weighting (dB(A)) curve is a mimic of the human hearing and it is utilized to measure the relative loudness perceived by the human ear [5]. If the surrounding sound is low, it is transmitted and played by a loudspeaker inside the earmuffs, i.e. the sound is amplified. However, if the surrounding sound is loud, the hearing protectors provide full protection [4].

Since hearing protectors are used at such different occasions, the soundscapes are varying. The soundscape may also change within the same user area as different events occur, such as warnings and speech in an otherwise noisy environment. However, despite the fact that the soundscapes where the hearing protectors are used vary, they function equally. This means that they may not function optimally for all occasions. The active hearing protectors that are currently used have high amplification, which is not always needed nor desired. In order for the hearing protectors to function optimally, there is a need of more adaptable hearing protectors. Hearing protectors with the ability to adjust the level dependent function depending on environment may therefore fulfill this need. To make that possible, the soundscapes must be identified and classified.

1.2 Purpose

The aim of this Master Thesis is twofold;

1. Find soundscapes that are interesting to identify.
2. Evaluate which of these soundscapes that are possible to identify and classify them.

This means that the project will investigate a possible solution for adapting the hearing protectors to different soundscapes.

The goal is to create a classifier, which can classify soundscapes into four classes with over 90% accuracy. These soundscape classes will be determined after evaluating which soundscapes that are interesting for classification. User areas where the hearing protectors should have different applications determine the soundscapes of interest. These applications can for example be attenuation or high amplification.

1.3 Delimitations

In order to have a reasonable size of the project, some limitations had to be made;

- The thesis will focus on classification of recorded signals to predefined audio classes.
- The number of classes for the classification will be limited to four.
- The possibility to implement the classifier in real time will be evaluated by a complexity analysis.
- The signals that will be classified are signals that only contain one audio type, i.e. they do not contain combinations of sounds.
- Feature extraction will be used together with the k-Nearest Neighbor algorithm for classification.

2
Theory

This chapter covers sound, signal processing and audio classification related aspects intended to provide a foundation for understanding the key concepts of the thesis.

2.1 Soundscape

Simply, a soundscape can be described as a composition of voluntary and involuntary combination of sounds from physical and biological sources [2]. However, it is not only defined by the composition of sounds, but also how the acoustic environment is perceived by humans [6]. The sounds that mix or mask in order to create a soundscape can be divided into geophonies, biophonies and antrophonies [2].

Geophonies are all the sounds from non-biological natural phenomena. This includes wind, rain, thunder, and running water. The morphology of a landscape highly affects the geophonies of a soundscape. Elements such as humidity, temperature and the scenery of a landscape affect the transmission of sounds [2].

Biophonies are represented by all the nonhuman sounds from biological agents, which are sounds from living organisms in a given ecosystem. Latitude, season and hours of day yield different patters of the biophonies [2].

Antrophonies are the results of artificial sounds such as cars, industries or airplanes. Due to the urban development, the antrophonies are the most common sound source in the soundscape of large parts of the Earth [2].

Antrophonies are most common in urban areas while less common in nature and where human intrusion is decreased. Geophonies are equally common in all landscapes since there is wind, thunder etc. everywhere and since these phenomena are independent of human intrusion. Biophonies are most common where there is less intrusion by humans [2].

2.2 Audio Features

Giannakopoulos et al. [7] describe audio features as extracted data, which contain information about the properties of an audio signal. There are many features both in the time and frequency domain that can be extracted in order to find information about the time and spectral properties of a signal. These features may be used for recognition of patterns in the signal. In audio classification, a combination of time and frequency features is often required [7].

Davis and Mermelstein [8] also highlight the importance of representing audio signals by informative parameters in audio classification tasks. The audio features are selected in order to eliminate non-informative data and to extract the informative data of an audio signal. Since the extracted data is further used for classification of the signal, the choice of representative parameters highly affects the classification result [8]. In order to classify a signal, it is important to understand which parameters that are representative for that specific signal.

Features are often extracted from short frames. Audio signals are therefore divided into 10-30 ms long audio frames[9]. This is due to the fact that most audio signals are non-stationary and the short frames may be considered as 'quasi-stationary' [10]. Each frame is then homogeneous, which facilitates the classification of the frames [7]. The features that are extracted from the frames represent the short-time properties of the signal [9].

2.2.1 Time-Domain Audio Features

Time domain features are extracted directly from the unprocessed audio signal and may be used to analyze audio signals in a simple way [7]. Common time-domain audio features are zero-crossing rate and energy. However, these are often combined with frequency-domain audio features [7]. In the following paragraphs, time-domain audio features used in this work will be presented.

2.2.1.1 Energy

The short-term energy is given by the equation

$$E(i) = \sum_{n=1}^{W_L} |x_i(n)|^2. \tag{2.1}$$

$x_i(n)$ is the data points in the ith frame and W_L is the length of the frame [7]. In order to obtain a value that is independent of the frame length, the result from equation 2.1 can be normalized by dividing it by W_L,

$$E(i) = \frac{1}{W_L} \sum_{n=1}^{W_L} |x_i(n)|^2. \tag{2.2}$$

Since speech signals have different characters and silent parts, the energy will have high variation for these signals. Parts that are silent will have low energy, while parts with speech have high energy [7]. In this project, the energy will be used in order to distinguish between silent and loud audio frames.

2.2.1.2 Zero-Crossing Rate

Zero-crossing occurs when a signal changes sign from negative to positive or vice versa [11]. The Zero-Crossing Rate (ZCR) is the number of zero crossings during a constant time period [11] and is given by the equation

$$Z(i) = \frac{1}{2W_L} \sum_{n=1}^{W_L} \mid sgn[x_i(n)] - sgn[x_i(n-1)] \mid . \tag{2.3}$$

where,

$$sgn[x_i(n)] = \begin{cases} 1, & x_i(n) \geq 0 \\ -1, & x_i(n) < 0 \end{cases} \tag{2.4}$$

and W_L is the frame length. Noisy signals often results in a high sign-change rate and ZCR can therefore be used to measure the noisiness of a signal [7].

Furthermore, the ZCR can be considered a simple way to measure the frequency of a signal. However, speech signals are broadband signals and the interpretation of the average ZCR is not as precise, but the average of short time ZCRs can be a rough estimate [12].

2.2.1.3 Pulse Counter

Teledyne [13] presents a report that describes an approach for pulse detection. According to the report, the first step is to find the baseline, M(k). The impulses will then be detected relative to the baseline. In order to not detect false impulses, such as noise and outliers, a smoothing filter may be used on the signal.

Furthermore, the report mentions that the second step is to have a trigger event, which is the point where the pulse begins. This point is the data point which has a value higher or equal to a given threshold, T(k), where

$$T(k) = M(k) + trigger\_level. \tag{2.5}$$

Trigger_level should be chosen such that the trigger level is higher than the noise level of the signal [13], see figure 2.1.

The next step according to Teledyne's report is to find the end of the pulse. The end of the pulse occurs when the signal falls back against the baseline again. A threshold is set, which is called the Reset level, see figure 2.1. The point where the signal passes the Reset level, R(k), is the end of the pulse. The reset level is given by,

$$R(k) = T(k) - reset\_hysterisis, \tag{2.6}$$

where reset_hysterisis is a constant that decides the reset level relative to the trigger point [13].

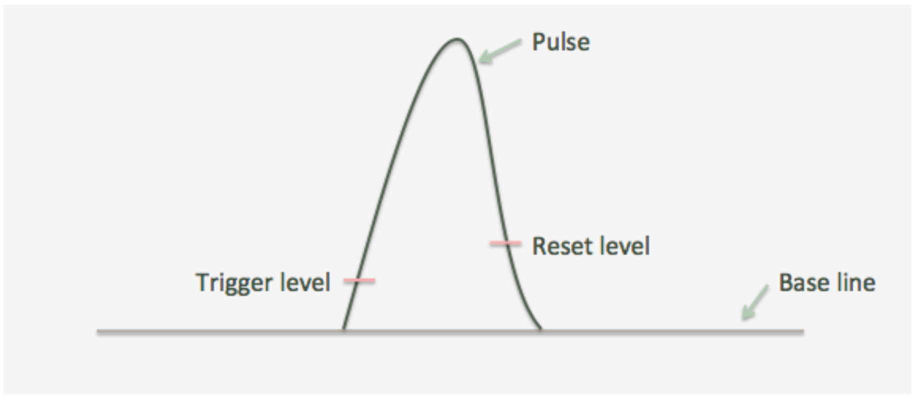


Figure 2.1: Illustration of a pulse with a trigger and reset level

Sometimes, the signal regions right before and after the pulse, called the pulse edges, are of interest [13]. This may be due to the fact that one wants to find single impulses, which are sudden pulses that have a certain number of time without pulses before and after it occurs.

2.2.2 Frequency-Domain Audio Features

The frequency distribution of a signal is obtained from the Discrete Fourier Transform (DFT) of a signal. There are many important audio features that can be computed from the DFT of the audio frame. Three of them are Spectral Centroid, Spectral Flux and Mel Frequency Cepstral Coefficients, which will be described in detail below [7].

2.2.2.1 Spectral Centroid

The spectral centroid measures the position of the spectrum [7]. One can say that the centroid represents the ’center of mass’ of the spectrum and is connected to the brightness of the sound [14].

The value of spectral centroid for the ith audio frame is given by

$$C_i = \frac{\sum_{k=1}^{W_{fL}} kX_i(k)}{\sum_{k=1}^{W_{fL}} X_i(k)}. \tag{2.7}$$

Brighter sounds yield higher spectral centroid values [7].

2.2.2.2 Spectral Flux

The measurement of the spectal change between two following frames as called the spectral flux. It is given by the equation

$$Fl_{(i,i-1)} = \sum_{k=1}^{Wf_L} (EN_i(k) - EN_{i-1}(k))^2. \tag{2.8}$$

It is the squared difference in normalized magnitudes of spectra of the two frames. Due to the different characters and silent parts in speech, the spectral flux have higher values for these signals than for instance music signals [7]. This feature may be used in order to determine the timbre of an audio signal [14].

2.2.2.3 Mel-Frequency Cepstrum Coefficients (MFCCs)

The MFCCs are among the most popular audio features [10] and represent the cepstrum coefficients of the signal on the mel scale [7]. The mel scale is the logarithmic scale of the normal frequency scale. This feature is effectively used in recognition of structures in signals and in modeling the subjective pitch and frequency content of audio signals [9]. It has been used for speech recognition, music genre classification and speech clustering among others [7].

Li et al. [15] present MFCCs as the most important feature in speech recognition, which may be due to the fact that MFCCs contain a large amount of information. The book further explains that the first coefficient, $MFCC_0$, often is discarded since it only represents the mean of the filter bank energies. However, coefficient 1-13 is often used in audio classification [7]. The second coefficient, $MFCC_1$, describes the spectral tilt and $MFCC_2$ describes the shape of the audio spectrum [15]. Periodicity of the signal is represented by the higher order MFCCs [15].

There are five main steps in MFCCs extraction, which are shown below.

1. The first step is to divide the audio signal in frames of a fixed size. Pre-emphasis of the frame is performed in order to improve the efficiency of the analysis and a first order ‘pre-emphasis’ high pass filter emphasizes higher frequencies. Windowing is also performed in order to weight the frame and eliminate outliers in the audio frame. A Hamming window is used, since it is the most common for windowing in frequency domain [10].

2. The DFT of the frame is computed, which results in a power spectrum representation of the signal [7]. The implementation in frequency domain will increase the computational efficiency in the following step [15].

3. The magnitude of the DFT is used as an input to a mel-scale filter bank, which contains L filters with overlapping triangular frequencies [7], see figure 2.2. The filter bank maps the power spectrum onto Mel-frequency scale bands [10].

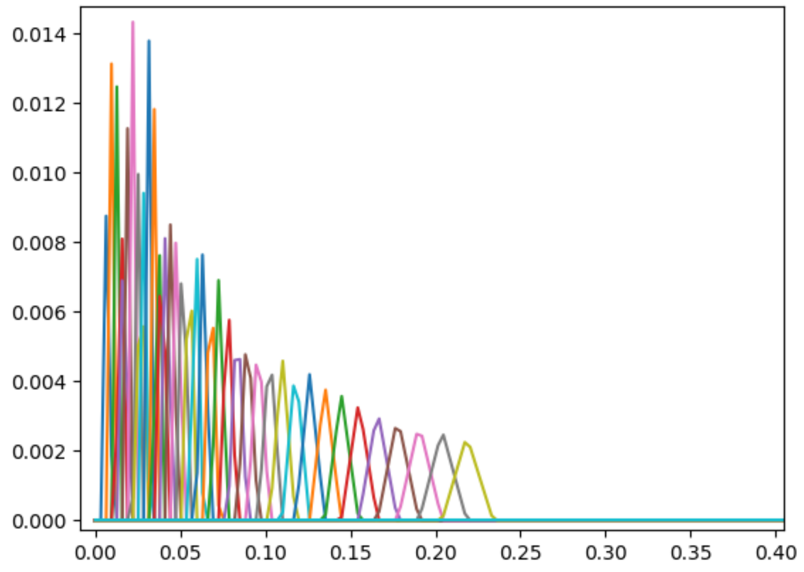


Figure 2.2: Mel-scale filter bank

4. A discrete cosine transform is performed on the logarithms of the outputs from the last filter in order to obtain the Mel-frequency cepstral coefficients [10]. This is given by the equation

$$c_m = \sum_{k=1}^{L} (\log \widetilde{O_k}) \cos[m(k - \frac{1}{2})\frac{\pi}{L}], \tag{2.9}$$

where $\widetilde{O_k}$, k = 1, . . . ,L, is the power at the output of the kth filter and m = 1, . . . , L [7]. Figure 2.3 shows the MFCCs of a speech signal.

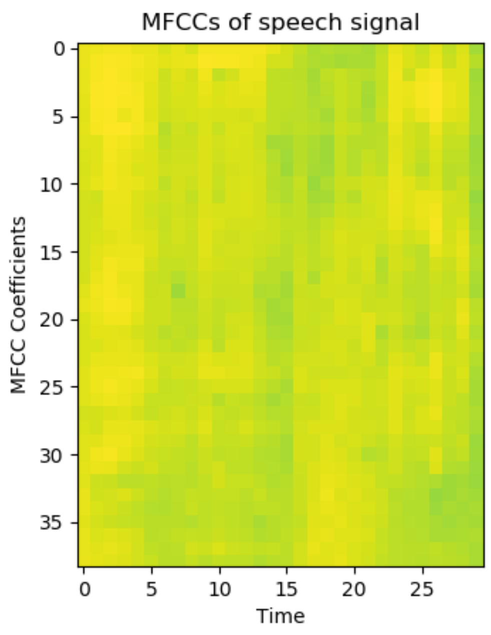


Figure 2.3: MFCCs of a speech signal

2.3 Level Detection

In order to obtain a smooth representation of a signal, level detection can be used [16]. According to Käsbach's report [17], the detector compares the current data value, x[i], with the previous data value, x[i-1], in order to filter it properly. The report presents the level detector function as a first order low pass filter, which uses different time constants depending on whether x[i] - x[i-1] is positive or negative. The time constant is further used in order to calculate the filter coefficient, α [16].

The level detection filter is given by,

$$s[n] = \alpha s[n-1] - (1-\alpha)r[n] \tag{2.10}$$

where α is the filter coefficient, r[n] is the input and s[n] is the output [16].

Giannoulis et al. [16] state that the time constant, τ, is defined as the time it takes for the system to reach $1 - \frac{1}{e}$ of its final value. This gives,

$$\alpha = e^{\frac{-1}{\tau f_s}}. \tag{2.11}$$

τ is the attack or release constant. Which one it is depends on whether x[i] - x[i-1] is positive or negative. The attack and release constant determine how long it takes for the smoothed signal to rise or fall to a new input value r[n] [17]. It is the values of the attack and release constants that define how the smoothed signal will look.

Larger attack and release constants will result in a smoothed signal that rises and falls slowly. Contrary, small constants will yield a smoothed signal that rises and falls quickly, which makes it follow the original signal closer.

Using a level detection function will not only give an envelope of the signal [16], but it will also smooth out the data by removing noise and outliers, which removes insignificant variations between data points [18].

Figure 2.4 shows the smoothed and original version of an alarm signal. The attack time is rather low since the smoothed signal only reaches half of the amplitude of the original signal.

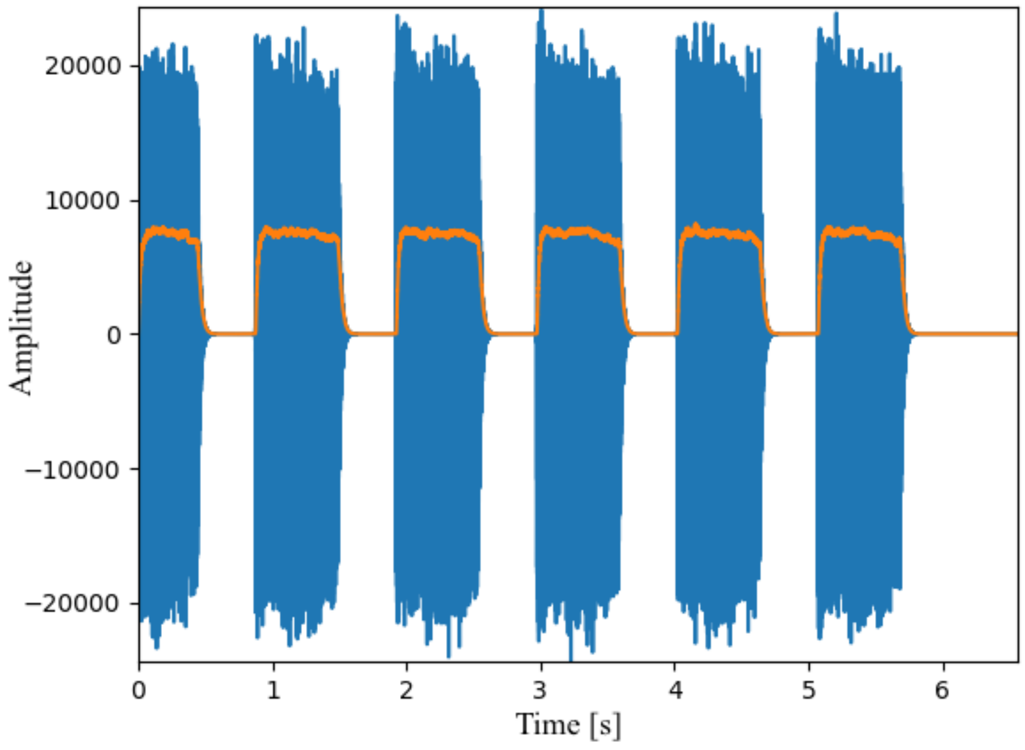


Figure 2.4: Level detection implemented on alarm signal, where blue represents the original signal and orange represents the smoothed signal

2.4 Audio Classification

Extracted features are used together with a classifier in order to classify an audio signal. When classifying, the signal is given a class label, which is estimated based on the extracted features. Often, an audio signal is classified to a set of predefined audio classes. This requires a sample set, which consists of data with known class labels belonging to these classes. For the following presented classification approach, it is assumed that audio frames with homogeneous data are classified, i.e. each audio frame only contains one audio type [7].

2.4.1 The k-Nearest Neighbor (k-NN) Classifier

The k-NN method is one of the simplest machine learning algorithms used for classification [19]. The book *Introduction to Audio Analysis: A MATLAB Approach* by Giannakopoulos et al [7] introduces the k-NN classifier as a classification method that does not require a training stage. The book further explains that instead of samples for training, the samples are directly used for classification. These data samples have predefined class labels, which is necessary when classifying an unknown object. The samples can for example be vectors of audio features. The samples will be compared with the feature vector of the unknown object in the classification process [7].

The k nearest neighbors are detected among the samples, where k is a predefined integer larger than one. The nearest neighbors are the samples with features that are most similar to the features of the unknown object. Which objects that are most similar is evaluated by a distance measure [7].

If $\mathbf{x}$ is the feature vector of the unknown object and $\mathbf{v}_i$ is the feature vector of the ith sample, the distance measure computes the distance between these vectors. The most common distance measure is the Euclidean distance [7], which is given by the following equation

$$d(\mathbf{x}, \mathbf{v}_i) = \sqrt{\sum_{j=1}^{n}(x(j) - v_i(j))^2[20]}. \tag{2.12}$$

Where i = 1, ..., M, and M is the total number of samples [7]. 'n' is the dimensionality of the feature space [20]. The dimensionality is determined by the number of variables that describes each sample [20]. The number of features determines the dimensionality in the case where each sample is described by a feature vector. The variable j is updated from 1 to n so that the distance is computed for each dimension [7].

The k samples with the shortest distance to the unknown sample are the k nearest neighbors [7]. In order to classify the object, it is evaluated how many of the nearest neighbors that belong to each class. The object is assigned the most common class label among the nearest neighbors [21]. In figure 2.5, classifications using k = 1 and k = 3 are illustrated. The unknown sample will in both cases be assigned to Class 1, since the largest number of neighbors belong to that class.

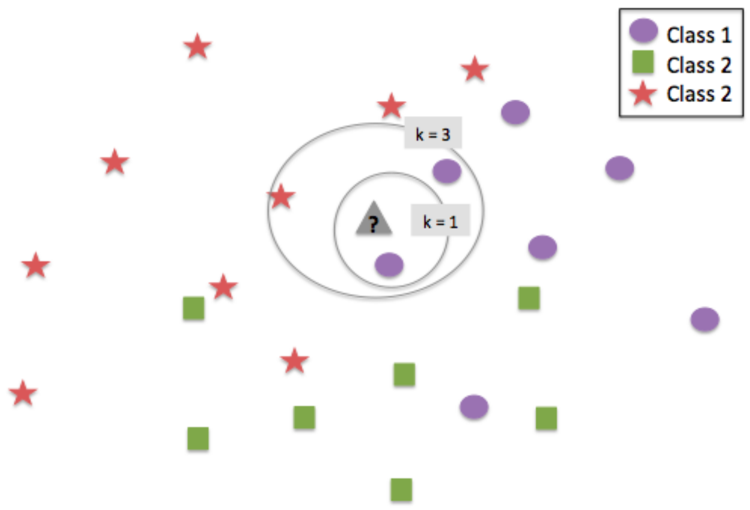


Figure 2.5: Illustration of the k-NN method

Furthermore, it is possible to estimate the posterior probability of the classification by the following equation,

$$P(\omega_i|\mathbf{x}) = \frac{k_i}{k}. \tag{2.13}$$

P is the probability that the feature vector $\mathbf{x}$ corresponds to the estimated class label i [7].

Better results will be obtained with a larger data set [7]. However, an increased number of samples will yield an increased computational time. Using a smaller number of samples will therefore reduce the computational burden of the k-NN algorithm [21]. The best value of k can be found by tests. Usually, a small value is preferred [7].

A further step when using a k-NN classifier is to weight the contribution of the neighbors so that the close neighbors contribute more to the result than the distant neighbors [19]. A common way to weight the neighbors is to compute the weight, w, as follows

$$w = \frac{1}{d} \tag{2.14}$$

where d is the distance to the neighbour [19].

2.5 Performance Measures

In order to evaluate the performance of the classifier, the accuracy, sensitivity and specificity can be calculated. To compute these, it is necessary to count the true positives, false positives, true negatives and false negatives of a classification. True positives, TP_x, are, for a certain class x, correctly classified frames. True negatives,

TN_x, are the frames that are correctly *not* classified to class x. On the contrary, false positives, FP_x, are frames that are classified as x, while in fact belonging to another class. False negatives, FN_x, are the frames that belong to class x, but are classified as another class [22].

2.5.1 Accuracy

The accuracy is a measurement of how well the classification works, i.e with what accuracy it correctly classifies signals. It can be calculated as,

$$\text{accuracy} = \frac{\text{number of correctly classified frames}}{\text{total number of frames}}[23]. \tag{2.15}$$

2.5.2 Sensitivity

The sensitivity is the classifier's ability to correctly classify the frame as x. It can be computed according to the following equation

$$SEN_x = \frac{TP_x}{TP_x + FN_x}[22]. \tag{2.16}$$

2.5.3 Specificity

The specificity is the classifier's ability to correctly classify a frame to another class than x. It can be computed according to the following equation

$$SPEC_x = \frac{TN_x}{TN_x + FP_x}[22]. \tag{2.17}$$

2.6 Complexity Analysis

Vrajitour et al. [24] mention that when analyzing an algorithm, it is examined how long it takes and how much space is needed for running the code with an input of size n. In order to find the complexity and estimated running time, all operations used throughout the code i stated. The complexity of each operation is expressed as a function of n [24].

In order to find the complexity of the code, the functions will be rewritten in terms of big-O [24], see section 2.6.1. Furthermore, since the project aims to investigate the ability to implement the classification in real time, the running time of the code is of great interest. Section 2.6.2 explains how the running time of a code is estimated.

2.6.1 Big-O Notation

Vrajitour et al. [25] define Big-O as,

> "Let f(n) and A(n) be positive functions defined on the positive integers. We say f(n) is big-oh of A(n) as n $\to \infty$ if and only if there exists a positive constant C such that f(n) $\leq C \cdot A(n)$ for all large values of n. We denote this by writing $f(n) = O(A(n))$ as n $\to \infty$" [25].

The big-O notation should not include any multiplicative constants, since the idea with this approach is to remove irrelevant constants. For example, if a time complexity, *T(n)*, of an algorithm is described by the function $T(n) = D \cdot n + C$, where D is the operations in a loop with n iterations and C is the operation that occurs only once. Then T(n) can be written as $O(n)$, where $O(n)$ is the big-O notation of the function T(n). Furthermore, the book states that the growth rate of f(n) is *no faster* than the growth rate of A(n). However, the growth rate of f(n) may be *slower* than the growth rate of A(n) [25].

Donald Bren School of Information & Computer Science [26] gives examples of the time complexity of built in functions in Python. For example, simple functions such as get value at index (A[i]), store value at index (A[i] = 0) or length of array (len(A)) have the time complexity $O(1)$. All simple operations on integers with less than 12 digits also have the time complexity $O(1)$.

Furthermore, operations that are performed on a list where the list has to be searched have the time complexity $O(n)$, where n is the length of the list. Simple operations on lists such as adding or removing an element or clearing the list have the time complexity $O(1)$ [26].

One of the most time consuming algorithms according to Donald Bren School of Information & Computer Science [26] is the sort function, with a time complexity of O(n $\cdot \log n$). Another operation with the time complexity O(n $\cdot \log n$) is the discrete Fourier transform (DFT) [27]. Resampling has according to Hol's report [28] the time complexity O(n).

Additionally, many algorithms contain loops or nested loops. The time complexity of loops are determined by the number of iterations. This presumes that basic operations are executed each iteration. Nested loops will have the time complexity O(n^2) [24].

Donald Bren School of Information & Computer Science [26] furthermore states that the if-statement,

```
if test1:
    code1
elif test2:
    code2
```

has the time complexity corresponding to the maximum complexity of *test1* and *test2*, given that the complexity of code1 and code 2 is O(1).

2.6.1.1 Composing of Complexities

Usually, codes do not consist of solely one operation, but a composition of operations. It is therefore important to know how to combine the information about the complexities. The composition is performed since it is preferable to analyze the total complexity of a code, instead of the complexities of single operations [26].

According to Donald Bren School of Information & Computer Science, the law of addition for Big-O notation is

$$O(f(n)) + O(g(n)) = O(f(n) + g(n)), \tag{2.18}$$

where f(n) and g(n) are separate functions. Furthermore, the time complexity of these functions can be expressed in the same way,

$$O(n) + O(\log n) = O(n + \log n). \tag{2.19}$$

Since only the greater complexity is taken into consideration, it results in

$$O(n + \log n) = O(n)[26]. \tag{2.20}$$

It is worth noting that the big-O notation only expresses the time complexity of the code. This means that it shows how the running time of the code is dependent of the array length n. Figure 2.6 shows how the running time increases with larger n for different complexities.

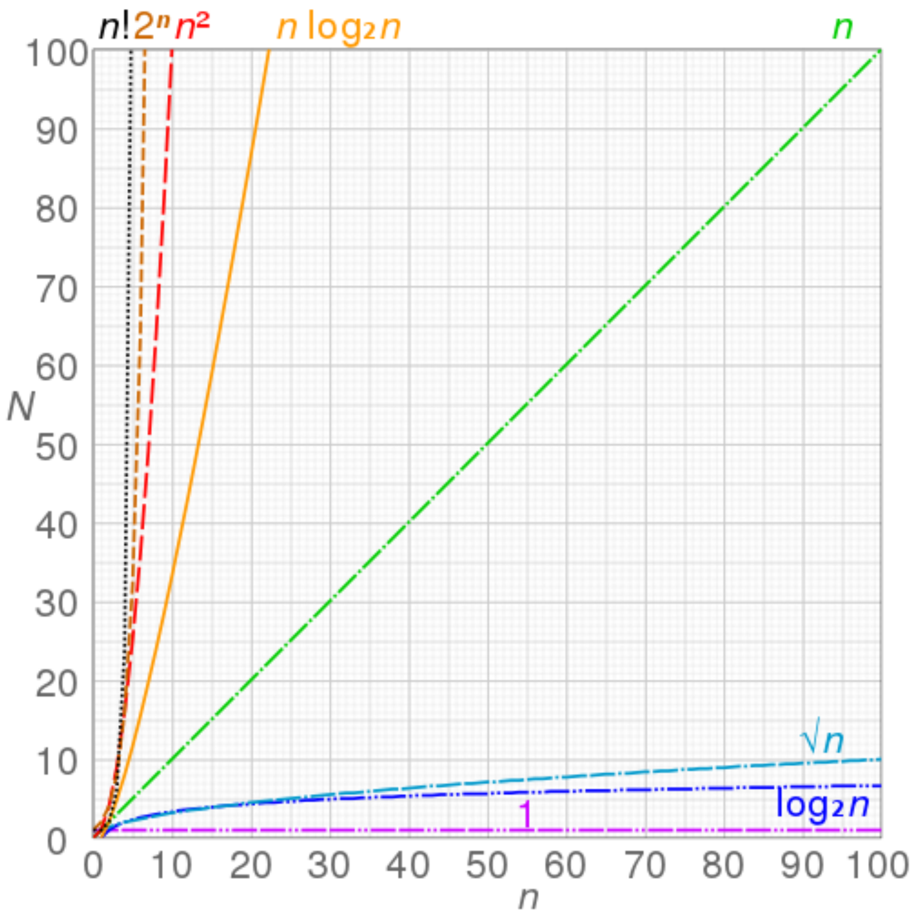


Figure 2.6: Comparison of computational complexities [29]

2.6.2 Estimation of Running Time

In order to compute the running time of a code, the processor that is used must be known. Furthermore, the clock cycles that are required for each type of operation must be known, together with the running frequency of the processor. It is also dependent on how the compiler translates the original code to machine code [24]. The table below lists commonly used operations and their required clock cycles.

Instruction	**Clock cycles**
Multiplication	1 [30]
Division	12 [30]
Addition	1 [30]
Subtraction	1 [30]
Load	2 [30]
Sort	n^2 (n=length of array)
$\log_{10}$	115 [31]
FFT	37536 [32]
Absolute value	1 [33]
Compare	1 [30]
Square root	14 [33]
Bitwise and	1 [30]

Table 2.1: Table of instruction and the clock cycles required for the ARM Cortex-M4 processor

Furthermore, loops require further clock cycles in addition to those required for the operations within the loop. It requires two additional clock cycles that correspond to two instructions called the loop overhead; decrement the loop count and a conditional branch [34].

3

Related Work

This chapter presents related work within audio classification. This will yield information and inspiration for possible ways to approach the classification problem.

3.1 Speech Classification

There are several reports on previous work within speech recognition and classification, where some of them will be presented shortly in this section.

Liu [35] presents in her paper that Mel Frequency Cepstral Coefficients (MFCCs) and Gammatone Frequency Cepstral Coefficients (GFCCs) has been used in order to classify emotion and intensity in speech. A neural network was used, which classified signals based on extracted features from emotional song and speech. This method gave an accuracy of approximately 75%.

Badshah et al. [36] presents another study that is using MFCCs and random forest in order to classify different emotions in speech signals. The study showed an accuracy of 82.2% when classifying the signals to six classes.

Most speech recognition systems use MFCCs [37]. Abdalla et al. [37] presents a solution for a speaker recognition system. MFCCs are extracted in order to capture the characteristics of the speech signal, according to the article. The features were further used for classification of the signal as a speaker or non-speaker signal. The classification was performed with great success and obtained an average accuracy of more than 98% on clean signals and 93% on noisy signals [37].

Jalil et al. [38] describe how the short-term energy, magnitude, zero-crossing rate and autocorrelation can be used in order to discriminate voiced signals from unvoiced parts of signals. According to the study, high energy and magnitude, while the zero-crossing rate is low, indicate that the signal is voiced. Additionally, the zero-crossing rate can also determine where the speech starts and ends [38].

3.2 Industrial Noise Classification

No previous work within industrial noise classification was found. However, there are reports that presents solutions for classification of other noisy environments.

Alsouda et al. [39] found a solution for classifying noise into six different classes with an accuracy of 85-100%. The k-NN algorithm was used as classifier, with k=1, and the data set consisted of MFCCs of 3000 audio samples.

Another noise classification presented by Xia et al. [40] was to use a Guassian Mixture Model as classifier. Normalized Sub-Band Power Spectrum was used as feature in order classify noise in audio signals. An average accuracy for 14 noise classes was 97.64% [40].

3.3 Alarm Classification

Carmel et al. [14] mention in their report the importance of equipment that can detect alarm sounds in order to further warn hearing impaired or distracted people. Furthermore, the report presents a technique built on machine learning that can detect alarm sounds in a noisy environment with an accuracy of 98%. Several audio features, extracted from the envelope of the signal, were used to train the classifier. According to the article, not all features contribute to the detection of alarm. However, the features that do contribute to a good classification of alarms were features based on the spectral flux, the pitch, the short time energy and MFCCs [14].

According to Ellis [41] report on detecting alarms sounds, most part of all alarms have a frequency that lies within the 3kHz region. The report further claims that the spectral centroid would be a good feature in alarm detection [41], since it represents the 'center of mass' of the spectrum [14].

3.4 k-NN Classifier in Feature Based Classification

Several reports show that the k-NN classifier has been successfully used in feature based audio classification within healthcare [42][43]. Palapinan et al. [42] describe how the audio feature autoregressive coefficients may successfully be used together with a k-NN classifier in order to classify pulmonary acoustic signals. The result in the paper shows that respiratory sound classification with the k-NN method has a mean accuracy of 96.12%.

Chin-Hsing et al [43] report shows the result of using MFCCs combined with the k-NN method when classifying normal and abnormal lung sounds. The mean accuracy of these results were 92.25%, where the sound was classified into seven classes. Both Palapinan et al and Chin-Hsing et al's studies show an accuracy higher than 92% when using the k-NN method with feature based classification [42][43].

Furthermore, feature extraction of audio signals has been used in steganalysis [19]. Bhattacharyya et al present in their paper that a k-NN classifier performs well in feature based steganalysis, i.e. classify audio signals with hidden messages. The result shows that the classifier had an accuracy of 100%. The features that were used in the study were SNR and entropy [19].

Another application of the k-NN method is music classification based on mood feature extraction [44]. Sudarma et al. [44] claim that it is possible to classify the music that a person is listening to based on extracted mood features. This is possible since different music will raise different emotions. The article further explains that the moods were divided into four classes; Contentment, Exuberance, Depression and Anxious. A data set of 400 samples were used for the classification stage and different k between 1-16 were tested. The result shows that $k = 3$ gave the highest accuracy, which was 86.55% and the average classification time per music file was 0.01021 seconds. According to this study, the accuracy was not increased with a larger k. On the contrary, the accuracy seems to decrease with a larger k [44].

4 Methods

This chapter aims to describe how the study was performed and the methods used throughout the project. It also describes all the tests that were performed in order to obtain the final solution for the classifier.

4.1 Literature Review

A literature review was conducted in order to find valuable information about the subject and to create a theoretical framework early in the process. This information included knowledge about soundscapes, audio features, audio classification, feature based classification and other signal processing related aspects to provide a foundation of understanding of the project. The knowledge of these subjects was fundamental in order to approach the problem in a suitable way.

During the literature review, an examination of previous work within audio classification was performed in order to find information about previous studies within the area. Furthermore, the study gave inspiration how to approach the problem.

To find information and articles related to the subject, the search engines Chalmers Library and Google Scholar were used together with keywords such as: *audio features, audio classification, k-NN, k Nearest Neighbours and complexity analysis.* Furthermore, articles related to the project were accessed from 3M Peltor.

4.2 Data

Four soundscape classes were specified in this project. The classes were defined from application areas. The application areas for the hearing protectors were identified such that there were different needs of hearing protection for each area.

145 audio samples corresponding to the four specified classes were obtained from wikimedia.com and soundbible.com, where there are several audio files free to use. The samples were divided into one data set and one test set.

For the classes 'Speech', 'Industrial noise' and 'Alarm', there were 40 samples of each class. 15 samples from each class were used to create the data set and the remaining samples were used for the test set. For the class 'Other', there were only 25 samples since these were only used in the test set. This resulted in a data set with 45 samples and a test set with 100 samples.

For the classification of three audio classes, the test set only consisted of industrial noise, speech and alarm samples. However, for the classification of four classes it consisted of all samples. In order to correctly evaluate the accuracy, there were an equal amount of samples from each class.

30 frames from each audio sample were classified each test in order to have a sufficient amount of test data. True positives, false positives, true negatives and false negatives were noted for each test to evaluate the method. The sensitivity, specificity and accuracy were then computed for the classifier.

4.3 Feature Extraction

The selection of features was based on the information obtained in the literature review, the related work section and on the analysis of the tests that were performed successively during the project, see appendix A. At first, only normalized MFCCs were used to classify unknown audio signals. But since a high accuracy was required, more features were added along the project work.

The feature extraction was implemented as algorithms in Python, where the algorithms were based on the equations in section 2.2 *Audio Features*. The algorithms were written in order to extract relevant features of the sound waves. Furthermore, the features were used to classify sound waves.

4.3.1 Pre-Processing of Data

In order to have a fair classification, the features must be extracted from audio signals of the same format. By having the same amplitude, sample rate and data format of all signals, these factors will not affect the classification. Instead, the frequency properties, time properties and shape of the signal contribute to the classification.

The audio signals were therefore normalized in order to obtain the same amplitude for all files. By doing so, the shape of the signal became of higher importance for the classification than the amplitude. The signals were also resampled in order to obtain the same sample rate for all signals. The signals were furthermore converted to floats since a .wav-file can have the data formats int16, int32 or floats and the same format on all files is preferred. Moreover, the signals were divided into short audio frames, where each frame was 512 bits long, which corresponds to a time of 11 ms.

The energy was calculated for each frame. This was done in order to ascertain that the frame contained sound. The frames with very low energy are most probably silent frames. These frames were disregarded due to the fact that silent frames can belong to any class and may therefore be falsely classified. The threshold that decides whether the frame contained sound or not was manually obtained. The threshold was found by comparing the energy of silent and loud frames. Note that this was performed on the normalized, resampled and converted frames.

Furthermore, the features for the classification can be extracted from the envelope of the signal as well. This means that the pre-processing of the data was extended with a level detection function, which smooth out the data. By doing so, noise and outliers that may disturb the classification could be removed.

4.3.2 Verification of Feature Extraction

All feature extraction algorithms were verified by tests where the right outcomes were known in advance.

A sine signal was used in order to verify several feature algorithms. First, it was used to verify that MFCCs were correctly extracted. MFCCs show the envelope of the short time power spectrum [45]. The power of a sine wave will lie within one frequency. A plot of correctly computed MFCCs of a sine signal will therefore show a straight line. The MFCCs of a sine wave were plotted in order to verify that it showed a straight line.

Furthermore, a sine signal with known frequency was used to verify that the ZCR was correctly computed. With a known frequency, the expected ZCR could be calculated. The ZCR-algorithm was verified by controlling that the expected value of ZCR was obtained.

The pulse counter algorithm was verified with an impulse signal. The feature extraction was verified by showing that the result for an impulse signal was 1.

The energy is the sum of the squared samples of a signal. The energy of a signal with the constant amplitude 1 and a specific length should therefore be equal to the length. The algorithm was tested with a signal with amplitude 1 and a known length. Furthermore, the energy for a signal with amplitude 0 should be zero.

The flux is the spectral difference of two successive frames. Using two identical frames should therefore yield the result 0.

The spectral centroid is according to section 2.2 *Audio Features* the 'center of mass' of the spectrum. The center of the mass of a sine signal should lie at the frequency of that signal, since it only contains one frequency.

4.4 Implementation of the k-NN Classifier

The k-NN classification method was used in order to classify unknown audio signals. The classifier was first implemented for only noise and sinus classification. This was done in order to evaluate its ability to classify audio signals. It was essential that the classifier worked for simple classification problems before developing it further for more complex signals. When the noise/sine classification worked correctly, the classifier was tested for a set of signals with the four specified classes.

The k-NN classifier required a data set of features from audio samples of the specified audio classes. As mentioned in section 4.2 *Data Samples*, 45 audio samples were chosen to create the data set. The data set was created by extracting features from the audio files. The features were labeled with the true class labels and stored in .json files.

The features that were saved were average values. The features were calculated for all frames throughout the signal. These values were then used to compute the average feature values of the signal. The result was one feature vector, which represents the average features of that signal. The average was computed since it represents the signal more in general.

4.4.1 Distance Computation

The classifier computed the Euclidean distances for each feature between the unknown sample and all the samples in the data set. The values were saved in arrays, see figure 4.1, where every index of the arrays corresponds to a data sample. Each array contains distances for one specific feature. After the distances were computed, they were added in order to obtain the total distance for each data sample, which is illustrated in figure 4.1. The shortest distance was then found from the resulting array.

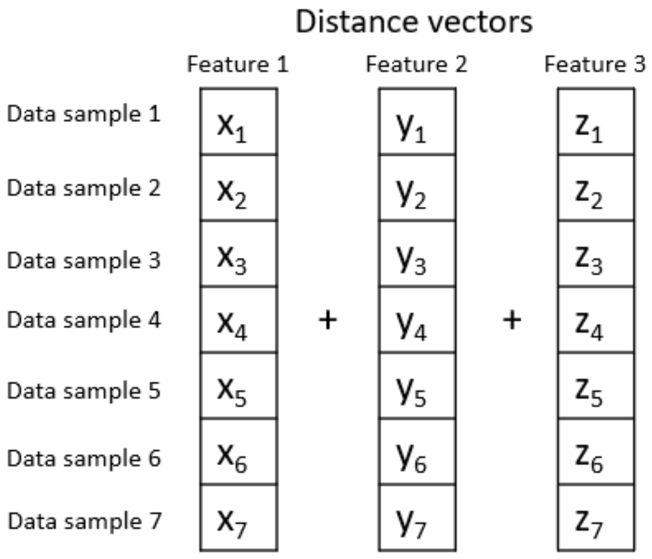


Figure 4.1: Illustration of the addition of distance vectors.

In order for the features to contribute equally to the classification, the feature values needed to be normalized. This was due to the fact that the feature values could differ in scale from one to several hundreds. Without normalization, the computed distances could possibly vary as much.

4.4.2 Normalizing Features

The features were normalized when creating the data set. In order to do this, the greatest possible value of each feature was found, i.e. the normalization factor. Dividing the features by the normalization factor resulted in values between 0 and 1. The features of the unknown sample were normalized in the same way. In order to find the normalization factor for each feature, the equations used to extract the features were studied together with the frame length and sample rate.

The MFCCs were normalized by the maximum value of the MFCC vector and the spectral centroid was normalized by the frame length, as described in the book *Introduction to audio analysis: a MATLAB approach* by Giannakopoulos et al. [7].

The normalization factor for the ZCR was defined as the maximum zero-crossings that can occur during one frame. The maximum number of zero-crossings is obtained if the signal changes sign between each sample. A frame with n samples can maximum have *n-1* zero-crossings, which became the value of the normalization factor.

The energy is computed as the sum of all squared samples. Since the signal was normalized in the pre-processing stage, the maximum value that each sample can have is 1. A frame with n samples can therefore have the maximum value n, which was chosen as the normalization factor. The theory section 2.1 also states that the energy is often normalized by the frame length, n.

Furthermore, the normalization factor and maximum value for the pulse counter is the frame length divided by the minimum pulse length.

The maximum value that can be obtained for the spectral flux is half of the frame length, $n/2$. This is due to the fact that the frame is divided in half and the maximum value which the subtracted samples can obtain is 1. Because, the maximum value of normalized magnitudes is 1 and the minimum value is 0.

The classifier computed the distances between the normalized feature values.

4.4.3 Weighting of Distances

When studying the feature values after the normalization it was clear that they were still in different scales. MFCCs had values between 0.1-1, while the ZCR and energy had values that were a tenth of that size. The distances for each feature could therefore differ as much.

This information gave the idea of weighting the distances in order to increase the accuracy by increasing the impact that the energy and ZCR have on the classification. Several weightings were tested for both the ZCR and energy in order to evaluate which was the best weighting. The weighting of the features are illustrated in figure 4.2.

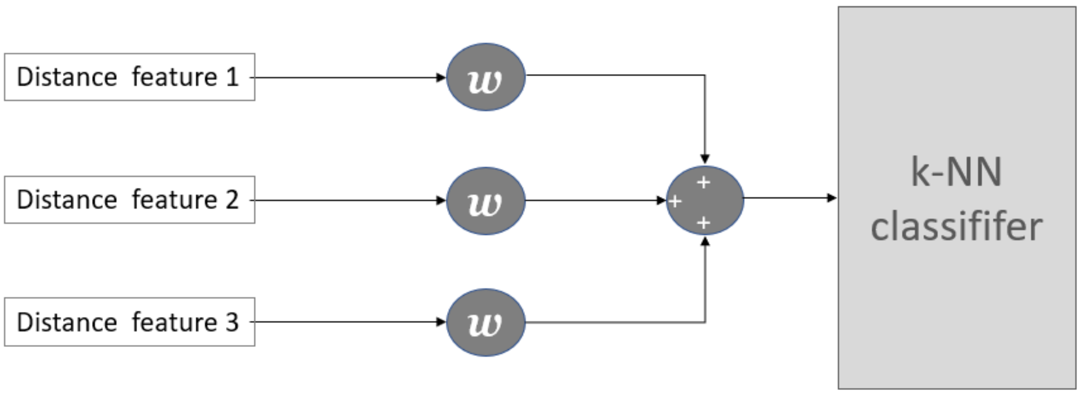


Figure 4.2: Illustration of the weighting of feature distances described in section 4.4.3

4.5 Evaluation and Testing

A number of tests were performed in order to evaluate the accuracy of different feature combinations. These tests were performed with the test set mentioned in section 4.2 *Data Samples*. The feature combinations were tested for both the original and smoothed signal in order to evaluate which one that gave the highest accuracy. Furthermore, tests were performed in order to evaluate the number of k that shall be used to obtain the highest accuracy. A fourth test was performed to find the threshold for the distance where signals should be assigned the class 'Other'. The sensitivity and specificity were calculated according to section 2.5.2 *Sensitivity* and 2.5.3 *Specificity*. The accuracy was calculated according to section 2.5.1 *Accuracy*.

Test 1: Evaluation of Feature Combinations To find a classifier with high accuracy, different combinations of features had to be tested in order to find the combination, which yields the best results.

The first test was performed using only MFCCs. The next feature selection was based on the results of this test together with the theory in section 2.2 *Audio Features* and 3 *Related Work*. Further features were chosen based on the tests that were successively performed. The number features used was therefore increased along the project work. New features and combinations were tested until a high accuracy was obtained. Each combination was tested for a classification of 2250 audio frames, i.e. 750 frames from each class 'Speech', 'Industry' and 'Alarm'.

Test 2: Evaluation of Classification Using Smoothed Signal In order to evaluate if classification of the original signal or the smoothed signal gave the best results, the feature combinations in test 1 were used in classification of a smoothed signals as well. The results gave an indication which method that was best to proceed with. The data set in this test contained features extracted from smoothed signals.

Test 3: Evaluation of Number of k Different values of k were tested in order to find the number of neighbours that yield the highest accuracy. These tests were performed on the feature combinations that showed the best results for $k = 1$.

Test 4: Finding Threshold for Fourth Class 'Other' Once a good accuracy for classification of 'Speech', 'Industrial noise' and 'Alarm' was found, the next step was to add the fourth class 'Other'. This class covers all the sound signals that do not belong to any of the three previously mentioned classes.

The data set used by the classifier does not contain any features corresponding to signals of the class 'Other', i.e. the data set only have 'Speech', 'Industrial noise' and 'Alarm' samples. This means that test signals of the class 'Other' do not have samples in the data set which they should be paired with.

Whether a signal should be assigned the class 'Other' or not should therefore be determined by the distance to the nearest neighbor. If the distance is too long between the unknown sample and the nearest sample in the data set, the unknown sample does probably not belong to any of the classes represented in the data set. The unknown sample should in that case belong to the class 'Other'. A threshold should define this distance. If the distance from the unknown sample to the nearest neighbor is larger than the threshold, the sample should be assigned the class ’Other’.

Several thresholds were tested aiming to find the value where as few samples as possible are falsely classified. By studying the distance for correctly classified frames of the first three classes, an approximate value of the threshold was found. Several thresholds around this value were tested in order to find a suitable threshold.

The test set used for this classification contained, in addition to the previous test set, signals with sounds of animals, football crowds and songs. These are sounds, which should be assigned to the class ’Other’. The tests were performed with the feature combination and k that gave the best results in the previous tests.

4.6 Complexity Analysis

A complexity analysis was performed in order to evaluate the classifier’s ability to work in real time. The analysis was divided in two parts:

1. Big-O notation
2. Estimation of running time

All operations used in the code had to be stated in order to find the Big-O notation and the estimated running time. The Big-O notation for each operation was determined in order to evaluate to total complexity of the code. The estimated running time was found by computing the total number of clock cycles required for running the code. The number of clock cycles and the running frequency of the processor were finally used to estimate the running time in seconds.

The analysis gave an indication of which processes that are computationally costly and which have a short computational time. It also gave an indication if the classifier is possible to use in real time.

5 Results

This chapter presents all the results of the project.

5.1 Specification of Classes

Four audio classes were specified in this project:

1. Speech
2. Industrial noise
3. Alarm
4. Other

A user area or a specific need defined each class. As mentioned in the limitations, the sounds that are classified in this project only contain one audio type.

Hearing protectors are often worn by users who communicate with each other or with other people in the surroundings that do not wear hearing protectors. Thus, the class ’Speech’ was chosen. It could be a usable function to identify and classify speech so that the users are aware of when someone is speaking to them.

Workers in noisy industrial environments often use hearing protection. However, the user may move between noisy and noiseless areas of the industry. It is therefore desirable that the hearing protectors can identify when the user is located in a noisy environment. The sounds that belong to this soundscape class only consist of antrophonies.

Alarms may also be triggered in environments where hearing protectors are used. The alarms can range from warnings from machines and trucks to fire alarms. If the user is located in a noisy environment, the hearing protectors are providing full protection in terms of attenuation. In that case, there is a risk that the user may not notice the alarm. This can in turn lead to injuries since the user may not be aware of risks in the surroundings. A hearing protector that can identify alarm sounds and warn the user can therefore be helpful. Thus, the third class was chosen to be 'Alarm'. The sounds that belong to this soundscape class only consist of antrophonies.

The goal with this master thesis is to classify sound signals into four classes. The fourth class was therefore specified as 'Other', which include all the sounds that do not belong to the three previously mentioned classes. It was important to have a class that covers all other audio signals, even though they might vary. The sounds that belong to this soundscape class can consist of geophonies, biophonies or antrophonies.

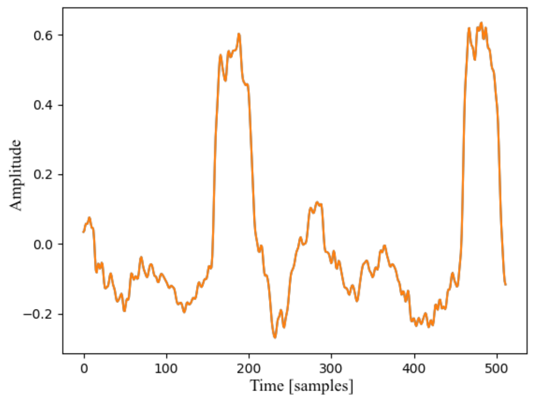
5.2 Feature Selection

A list of the features that were chosen for this project are shown below.

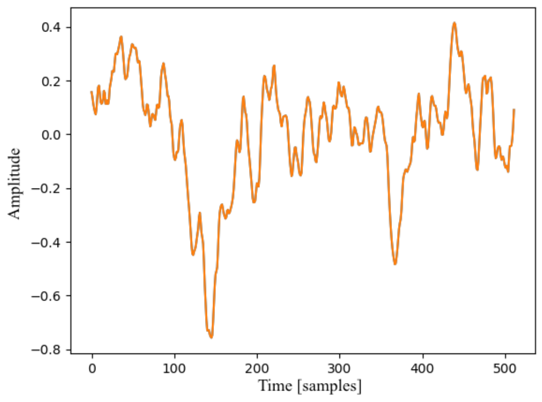
- **MFCCs**. According to the literature and related work review, this is the feature that contains the most information and is also the one that has been most frequently used in previous work. It has also shown to be a successful feature in both speech, noise and alarm classification. Since earlier classification tasks have shown a high accuracy by using only MFCCs, this was an obvious choice. Using MFCCs, the periodicity, spectral tilt and the spectral shape is obtained with only one feature.

- **Zero-Crossing Rate (ZCR)**. Since the specified classes have signals that either are semi-periodic or noisy, the ZCR was a good choice as the second feature. This feature distinguishes well between noisy and clean signals. Furthermore, it can be considered a rough estimation of the pitch, which has shown to be a good feature for alarm classification.

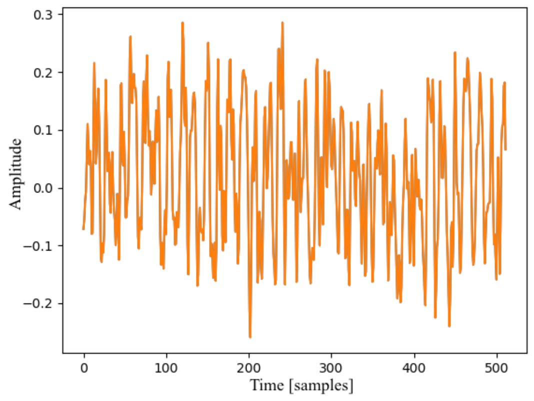
- **Pulse Counter**. Since the ZCR did not show much improvement from only using MFCCs, see appendix A, the next step was to try a pulse counter. When studying different signals of the classes, speech seems to contain several clear pulses in each frame. On the other hand, the frames that contain industrial noise or alarm sounds are very noisy, see figure 5.1. A pulse counter should therefore be a good feature to distinguish between speech and the other classes.



(a) Speech



(b) Industrial noise



(c) Alarm

Figure 5.1: Signals of speech, industrial noise and alarm. Note that the speech signal has two clear pulses, while the others are very noisy.

- **Energy**. As mentioned in section 3.3 *Alarm Classification*, the short-time energy has been successfully used in alarm classification. Furthermore, when studying the energy of frames from the different audio classes, it turned out that they had very different values of energy for each signal. For that reason, energy was chosen as a feature.

- **Spectral Centroid**. Since alarm signals seemed to be the most difficult to classify, features that are most useful in alarm classification were chosen. It was interesting to investigate if the combination of MFCCs and a feature that is especially powerful in alarm classification could yield good results. As mentioned in section 3.3 *Alarm Classification*, the frequency of most alarms is in the area of 3 kHz. The spectral centroid may therefore be useful when distinguishing between alarm, industry and speech.

- **Spectral Flux**. Spectral flux is one of the features that contribute most to alarm classification according to the article that was mentioned in section 3.3 *Alarm Classification.* The feature was chosen since alarm sounds seem to be the most difficult to classify, see appendix A. However, the spectral flux may be tricky to implement in real time since successive frames should be compared. In order to overcome that problem, each frame was divided in two segments. These two segments were in turn compared with each other, instead of two successive frames.

5.3 Classification

The classification was divided in two parts:

1. Classification of sound signals that belong to the three classes ’Speech’, ’Industrial noise’ and ’Alarm’, see section 5.3.1.
2. Classification of signals that belong to the three previous classes together with signals that belong to the class ’Other’, see section 5.3.2.

5.3.1 Classification of Three Classes

After several tests, it was clear that the classification of the original signal gave better results than the classification of the smoothed signal, see appendix A - B. This is probably due to the fact that smoothing the signal removes many important details. The tests in appendix C also showed that $k = 1$ gave the best accuracy, while increasing k decreased the accuracy. Larger k may give lower accuracy since the data set is too small.

The tests in Appendix A - C showed that the combination that gave the best accuracy was MFCCs, ZCR and Energy together with k = 1 for original signals. The accuracy was further increased by weighting the distances, as described in section 4.4.3 *Weigthing of Distances.* By changing the impact of the distances to $w_1\cdot$ MFCCs, $w_2\cdot$ ZCR and $w_3\cdot$ energy, the accuracy increased. This gave the results presented in table 5.1 below. The green cells represent correctly classified frames and the red cells represent falsely classified frames.

Classified label → True label ↓	Speech	Industry	Alarm
Speech	**628**	99	23
Industry	9	**739**	2
Alarm	3	116	**631**

Table 5.1: Test using $\omega_1\cdot$ MFCCs, $\omega_2\cdot$ ZCR and $\omega_3\cdot$ energy, k=1

These results correspond to the accuracy

$$\text{Accuracy} = \frac{628+739+631}{2250} = 88.8\% \tag{5.1}$$

the sensitivities

$$\text{Sensitivity}_{speech} = \frac{628}{750} = 0.837 \tag{5.2}$$

$$\text{Sensitivity}_{industry} = \frac{739}{750} = 0.985 \tag{5.3}$$

$$\text{Sensitivity}_{alarm} = \frac{631}{750} = 0.841 \tag{5.4}$$

and the specificities

$$\text{Specificity}_{speech} = \frac{739+2+116+631}{1500} = 0.992 \tag{5.5}$$

$$\text{Specificity}_{industry} = \frac{628+23+631+3}{1500} = 0.857 \tag{5.6}$$

$$\text{Specificity}_{alarm} = \frac{739+9+628+99}{1500} = 0.983. \tag{5.7}$$

The industrial noise has very high sensitivity but rather low specificity, which is due to the fact that many speech and alarm samples are falsely classified as industrial noise. This means that 98.5% of all industrial noise signals are correctly classified. However, there is a high risk that some of the samples that are classified as industrial noise in fact belong to another class.

On the contrary, the speech and alarm classification has low sensitivity. 83.5% of all speech signals are correctly classified and 84.1% of all alarm signals are correctly classified. However, the classification has high specificity. The frames that are classified as speech or alarm are correctly classified with a 99.2% respectively 98.3% certainty. Approximately 15% of speech and alarm signals are falsely classified industrial noise.

Note that table 5.1 only shows the test results for one classification. The classification method was tested several times using random frames from the training set and all results showed an accuracy between 88-91%.

5.3.2 Classification of Four Classes

A study of the distances for correctly classified frames in the test presented in section 5.3.1 *Classification of Three Classes* showed that most distances were between 0 and 1. However, some distances for alarm sounds had values up to 1.5. Thresholds close to 1 were therefore tested in order to find a suitable value.

The results showed that the classification of four classes did not yield as good results as the tests for three classes, see appendix A and E. It seemed hard to classify signals that did not have any samples in the data set to be paired with. However, the best threshold was T = 0.8, which gave the results presented in table 5.2 below. The green cells represent correctly classified frames and the red cells represent falsely classified frames.

Classified label → True label ↓	Speech	Industry	Alarm	Other
Speech	**618**	97	0	35
Industry	9	**739**	2	0
Alarm	0	67	**341**	342
Other	124	232	102	**292**

Table 5.2: Test using normalized $\omega_1\cdot$ MFCCs, $\omega_2\cdot$ ZCR and $\omega_3\cdot$ energy, k=1, T = 0.8

These results correspond to the accuracy

$$\text{Accuracy} = \frac{618 + 739 + 341 + 292}{3000} = 66.3\% \tag{5.8}$$

the sensitivities

$$\text{Sensitivity}_{speech} = \frac{618}{750} = 0.824 \tag{5.9}$$

$$\text{Sensitivity}_{industry} = \frac{739}{750} = 0.985 \tag{5.10}$$

$$\text{Sensitivity}_{alarm} = \frac{341}{750} = 0.455 \tag{5.11}$$

$$\text{Sensitivity}_{other} = \frac{292}{750} = 0.389 \tag{5.12}$$

and the specificities

$$\text{Specificity}_{speech} = \frac{739 + 2 + 67 + 341 + 342 + 232 + 102 + 292}{2250} = 0.941 \tag{5.13}$$

$$\text{Specificity}_{industry} = \frac{618 + 35 + 341 + 342 + 124 + 102 + 292}{2250} = 0.824 \tag{5.14}$$

$$\text{Specificity}_{alarm} = \frac{618 + 97 + 35 + 9 + 739 + 124 + 232 + 292}{2250} = 0.954. \tag{5.15}$$

$$\text{Specificity}_{other} = \frac{618 + 97 + 9 + 739 + 2 + 67 + 341}{2250} = 0.832. \tag{5.16}$$

According to these results, the ’Speech’ and ’Industrial noise’ classes were not affected by the threshold. However, the ’Alarm’ class was highly affected. It is also clear that the signals of the class ’Other’ are almost randomly classified.

The tests with lower thresholds than 0.8 showed that ’Industrial noise’ retained the high accuracy, while the classification of speech and alarm sounds decreased in accuracy, see appendix E.

5.4 Complexity Analysis

There are three main steps in the classification that were each examined separately in order to find the computational complexity and running time of the code. The three parts are shown in figure 5.2.

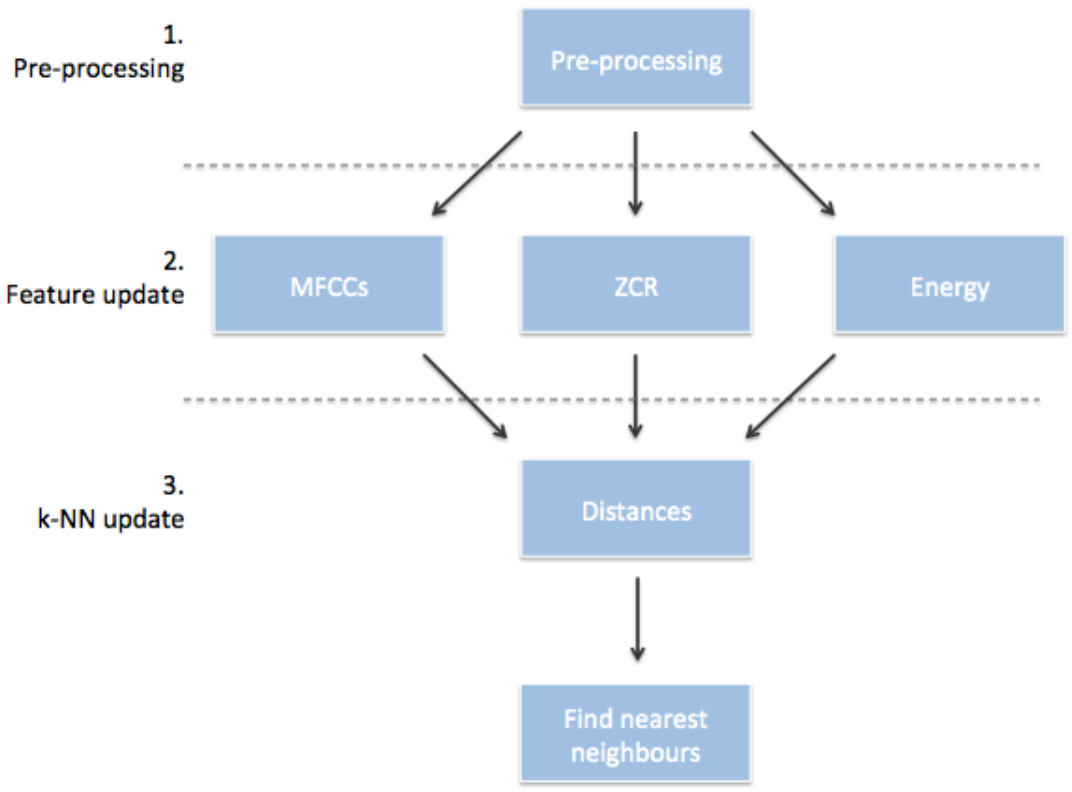
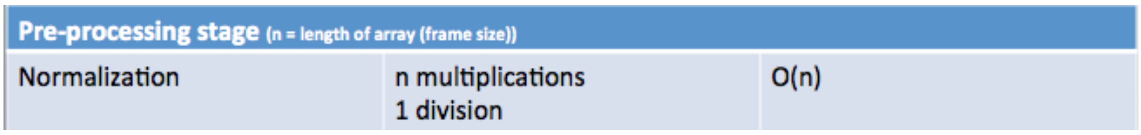


Figure 5.2: Simple flowchart of the main steps in the classification process

5.4.1 Big-O Notation

The big-O notations for the operations in the code are all found in theory section 2.6.1 *Big-O Notation.*

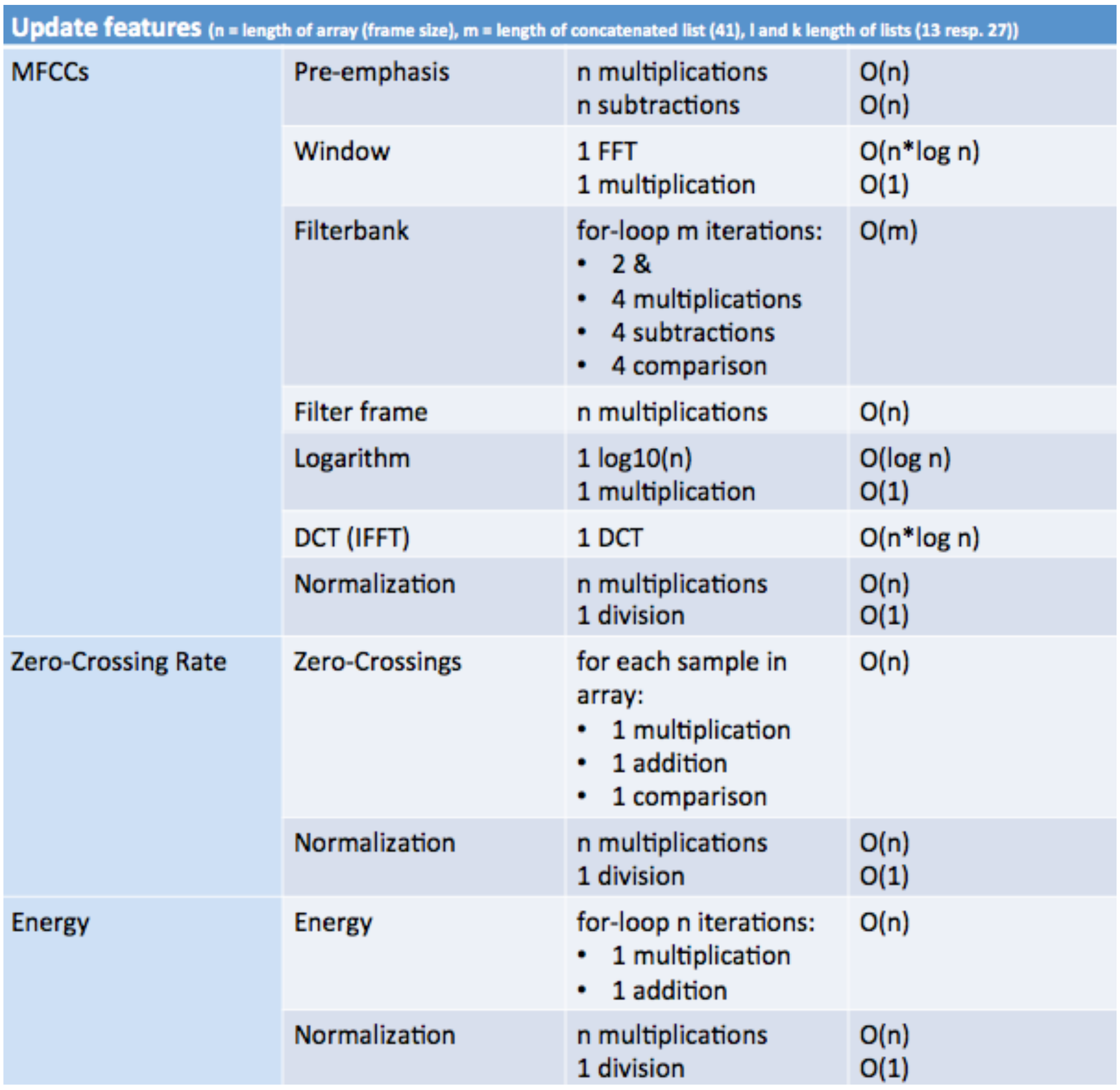
The first step in the classification is the pre-processing stage. Only the normalization is taken into consideration in this stage, see table in figure 5.3. Furthermore, the big-O notation for the operation is stated. As seen in figure 5.3, the maximum complexity of the pre-processing stage is O(n).



Pre-processing stage (n = length of array (frame size))		
Normalization	n multiplications 1 division	O(n)

Figure 5.3: Table of the operations in the pre-processing stage

Step number two is the feature update stage. This stage computes the features for the current audio frame. Figure 5.4 shows a table of the processes required for this stage, together with the big-O notation for each operation. The highest complexity for the feature update stage is $O(n \cdot \log n)$, where $n = 512$, which is the frame size.



Update features (n = length of array (frame size), m = length of concatenated list (41), l and k length of lists (13 resp. 27))			
MFCCs	Pre-emphasis	n multiplications n subtractions	O(n) O(n)

	Window	1 FFT
		1 multiplication
			O(n*log n)
			O(1)

	Filterbank	for-loop m iterations: • 2 & • 4 multiplications • 4 subtractions • 4 comparison	O(m)
	Filter frame	n multiplications	O(n)

	Logarithm	1 log10(n)
		1 multiplication
			O(log n)
			O(1)

	DCT (IFFT)	1 DCT	O(n*log n)

	Normalization	n multiplications
		1 division
			O(n)
			O(1)

Zero-Crossing Rate	Zero-Crossings	for each sample in array: • 1 multiplication • 1 addition • 1 comparison	O(n)

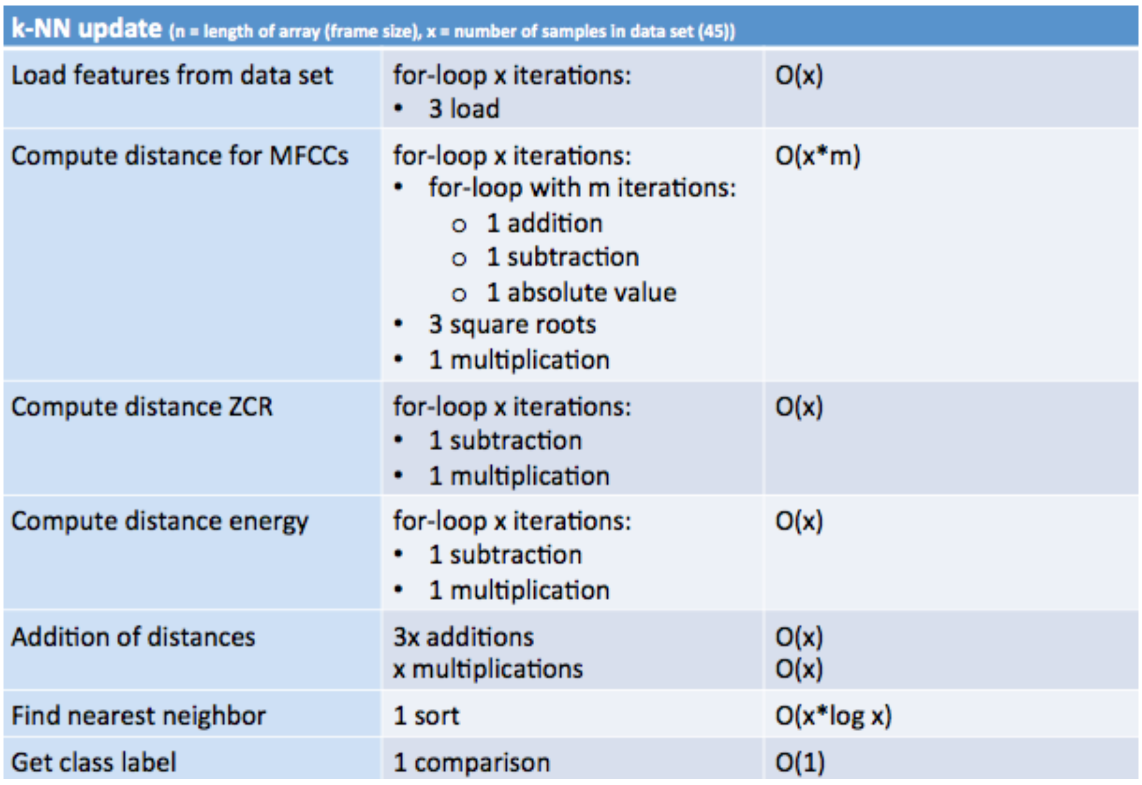
	Normalization	n multiplications
		1 division
			O(n)
			O(1)

Energy	Energy	for-loop n iterations: • 1 multiplication • 1 addition	O(n)

	Normalization	n multiplications
		1 division
			O(n)
			O(1)

Figure 5.4: Table of the feature update operations

The last step is to classify the audio frame based on the extracted features, which is the k-NN update stage. All operations and big-O notations for this stage is stated in the table shown in figure 5.5. The maximum complexity for the k-NN stage is O(x · m), where x = 45 and m = 41.



k-NN update (n = length of array (frame size), x = number of samples in data set (45))

Load features from data set	for-loop x iterations: • 3 load	O(x)
Compute distance for MFCCs	for-loop x iterations: • for-loop with m iterations: o 1 addition o 1 subtraction o 1 absolute value • 3 square roots • 1 multiplication	O(x*m)
Compute distance ZCR	for-loop x iterations: • 1 subtraction • 1 multiplication	O(x)
Compute distance energy	for-loop x iterations: • 1 subtraction • 1 multiplication	O(x)
Addition of distances	3x additions x multiplications	O(x) O(x)
Find nearest neighbor	1 sort	O(x*log x)
Get class label	1 comparison	O(1)

Figure 5.5: Table of the k-NN operations

The combined complexity of the three parts is O(n) + O($n \cdot \log n$) + O($x \cdot \log x$).

The complexity of the pre-processing stage and the feature extraction stage is O(n) + O($n \cdot \log n$). This means that the complexity of both stages depend on the frame size. The total complexity of these stages is O($n \cdot \log n$), since it is defined by the highest complexity of the algorithms. This means that the computational time increases more than linearly with a larger frame size.

The total complexity of the third stage, the k-NN classification, is O($x \cdot \log x$). This stage is not dependent on the frame size, but on the size of the data set. The computational time of the classification increases more than linearly with a larger data set.

Since the big-O notation of the separate parts of the code is O($n \cdot \log n$) and O($x \cdot \log x$), it is clear that the computational time increases depending on several parameters. The computational time will increase more than linearly with a larger n and x. A graph of the increase in computational time can be seen in figure 2.6 in section 2.6 *Complexity Analysis*.

5.4.2 Estimation of Running Time

In order to find the estimated running time of the code, the operations listed in figure 5.3-5.5 are compiled together with the required clock cycles for each instruction. See table 5.3 below.

Instruction	**Quantity**	**Clock cycles required per instruction**
Multiplication	3929	1
Division	4	12
Addition	2499	1
Subtraction	2566	1
Load	135	2
Sort	1	x^2 (x = 45 (size of sorted array))
FFT	2	37536
Comparison	677	1
Bitwise and	82	1
Log	1	115
Absolute values	1846	1

Table 5.3: Table of total number of instructions and the required clock cycles

The total number of clock cycles required for the classification of one frame is 89129, which can be computed from table 5.3 above. Note that this number of clock cycles do not include the loop overhead, since it is disregarded in this calculation.

If, for example, an ARM Cortex-M4 processor with the running frequency 48 MHz is used, the running time of the code is

$$\frac{1}{48 \cdot 10^6} \text{ s/cycle} \cdot 89129 \text{ cycles} = 1.86 \text{ ms.} \tag{5.17}$$

This means that it takes 1.86 ms for the classifier to classify a frame of size 512, when using a data set with 45 samples. Note that this is an approximation of the running time. However, even with an 100% margin of error, classification in real time would be feasible.

The computational time increases with both a larger data set and by using more features in the classification. This is due to the fact that with more samples in the data set, more distances have to be computed in order to find the nearest neighbors. The same is for the number of features used. When using more features, more distances have to be computed since the distance for each feature is calculated.

6
Discussion

This section will discuss and analyze the results of this project. The method of this project will be debated with reference to the results. Furthermore, factors that might have contributed to the results will be analyzed.

6.1 Classification

The results of this project indicate that the k-NN method works well for classifying signals of classes that have samples present in the data set. However, signals that do not correspond to any of the classes in the data set are more difficult to classify. The distances to the nearest neighbor for different classes seem to vary. It is therefore hard to find a suitable threshold for the samples that should not be classified.

However, a suggestion for solving this problem is to enlarge the data set. A broader variation of samples within the classes 'Speech', 'Industry' and 'Alarm' may result in shorter distances between the unknown sample and the nearest neighbors. A larger data set means more samples to compare the unknown sample with. The possibility that the unknown sample matches really well with a data sample is therefore increased.

Furthermore, industrial noise seems to be the easiest to classify correctly. Many speech and alarm signals are also falsely classified as industrial noise. This may be caused by noisy test signals, which can have features that are similar to those of industrial noise. Moreover, both industrial noise and male voices have low fundamental frequencies. This can be a contributing factor to the falsely classified speech signals.

The accuracy of industrial noise classification may also be high since these sounds do not vary much. Speech signals, on the other hand, can vary a lot. Factors such as who is speaking, the fundamental frequency of the speech, different words and characters, silent parts, etc. makes each speech signal different. It may therefore be hard to find a close neighbor for speech signals in the data set.

Since the classification is performed in real time, it is just short frames that are classified. Furthermore, it is only frames which contain sound that are classified.

This might remove some of the characteristics of speech and alarm signals. It is typical for both signals to vary between silent and loud parts. However, this property is impossible for the classifier to analyze since no whole signals are classified and since all silent frames are disregarded.

Another aspect to take into consideration with this classification is that only frames with homogeneous data were classified. This means that each frame only contained one audio type. It is not clear how the classifier would perform if the data was contaminated by noise or other disturbing errors. Or, if the data contain combined audio signals from several classes. However, signals that contain several types of sounds should be possible to classify with this method. Either by extend the data set with samples of combined sounds or by always classify the loudest of the combined sound.

6.2 Data

When using the k-NN method it is reasonable to question if a sufficient amount of data was used and if the data covers all cases that the method is expected to classify. The data set obviously contained data samples from all specified classes. However, it is not certain that it contained samples that cover all variations of the classes. A larger data set may therefore increase the accuracy of the classification. But unfortunately, a larger amount of data will make the classification more computationally costly. This is due to the fact that the unknown sample will be compared to a larger data set.

The data set in this project was balanced so that it contained an equal amount of samples from each class. However, more samples from the classes that are most difficult to classify can be beneficial. Additionally, since industrial noise signals seem to be the easiest to classify, the number of samples from this class may be decreased.

According to previous work mentioned in section 3 *Related Work*, 400 to 3000 samples were used in the data set in order to achieve good classifications. This indicates that the data set used in this project contains too few samples in order to obtain the best possible accuracy. However, if the data set was enlarged to 3000 samples, the computational time would be extremely long. Further work must therefore be performed on the data set. By studying the samples of the data set it is possible to merge all samples that are very similar. The data set can then be reduced to a few samples, which may represent several hundreds. This results in a small data set which covers a broad range of signals. A threshold must be set in order to determine how much the feature values may differ in order to merge them. The merge also results in fewer samples of the classes that have similar sound signals, while the classes with varying signals will be represented by more samples.

With a larger data set, a larger k may also be used in order to possibly increase the accuracy. Since there are more samples in the data set, the possibility to match with several samples of the correct class must be increased.

6.3 Reliability of The Tests

Another question relevant for this project is if the classification tests were limited by the size of the test set, since only 100 audio files were available. 30 frames from each file had to be used for each test in order to have a sufficient amount of samples to classify. Even though each frame looks different, they are taken from the same files and may therefore be very similar. The quality and content of the audio files may also affect the test results to a higher extent with a small test set. Files that are, for example, contaminated by noise will always be included in the tests and may lead to false classifications.

A larger data set could eliminate such problems. Samples may be contaminated by noise in a larger data set as well. However, since random files are chosen for each classification test, these may not always be included. The tests would also be more trustworthy if the tests were performed with a broader range of signals and with different random signals each time.

Despite all these concerns, the tests in this project have given a good indication of how well the classifier works.

6.4 Implementation in Real Time

The results of this project show that the classifier has a rather low computational time when classifying 512 samples long audio frames with a data set of 45 samples. It also showed that the highest computational complexities were dependent on the frame size and the size of the data set. So, by decreasing the frame size, the computational time could be even lower. Furthermore, the code could possibly be made more efficient.

The highest accuracy that was obtained in this project was approximately 89%. Even though this is a high accuracy, it is not sufficient enough to use in hearing protectors. An accuracy closer to 100% is preferable. However, such accuracy is not reasonable to try to achieve. There may always be deficiencies or cases were the classification will be incorrect. Since the hearing protectors are used in order to protect people from hearing impairment, there has to be a fallback plan for the cases where the classification is inaccurate. The fallback plan will ensure the safety of the user. For example, a very noisy environment can be falsely classified as an environment where high amplification is required. In that case, a backup is needed which perceives that the sound level is high and which ensures that the right protection is given.

Furthermore, the probability mentioned in section 2.4.1 can be used. If the probability that the samples is correctly classified is high, the level dependent function may be updated according to the classification. If, on the other hand, the probability is low the level dependent function will be updated according to the fallback plan. This approach requires that a k larger than 1 is used.

As mentioned in section 5.1 *Specification of Classes*, the classification of alarm signals was chosen since it may help users to easier detect alarms and warnings around them. If hearing protectors offer this function, the users might rely on the fact that they will be warned when there is an alarm or warning nearby. This could lead to less attention on events around them. On the other hand, if the users are aware that they might not hear alarm signal, they may be more observant. Thus, it is of high importance that this function works properly if it is offered in the hearing protectors. Otherwise, it might lead to more injuries due to inattention.

7 Conclusion

The k Nearest Neighbor is a method that can achieve good results in classification of signals that are present in both the data and the training set. Using MFCCs, the Zero-Crossing Rate and the short time Energy with k=1 achieved an accuracy of approximately 89% for this task. However, according to the results of this project, it works quite poorly when classifying signals of other classes than those that are represented by the samples in the data set.

Furthermore, the ability to work in real time is dependent on the size of the frame, the data set and the number of features used in the classification. However, the computational time achieved with the frame size, data set and features used for this project is well below what can be considered real time. Therefore, even an increased data set could work in real time applications.

In order to further evaluate the ability of this method, the number of data samples in the data and training set should be extended. The data set should be extended in order to increase the accuracy of the method and the training set should be extended in order to ascertain the reliability of the method.

References

[1] A. Stevenson and C. A. Lindberg, *New Oxford American dictionary.* Oxford University Press, 2010, p. 2018, ISBN: 9780199891535.

[2] A. Farina, "Soundscape and Landscape Ecology", in *Soundscape Ecology*, Dordrecht: Springer Netherlands, 2014, pp. 1–28. DOI: `10.1007/978-94-007-7374-5{\_}1`. [Online]. Available: `http://link.springer.com/10.1007/978-94-007-7374-5_1`.

[3] 3M Peltor Communication Solutions, *Industry.* [Online]. Available: `http://peltorcomms.3m.com/Americas/Page.asp?PageNumber=1031`.

[4] *1520/1525 Level Dependent Ear Muffs.* [Online]. Available: `http://multimedia.3m.com/mws/media/1568830/1520-1525-level-dependent-muffs.PDF?&&COrrrrQ`.

[5] T. McMinn, "&quot;A-Weighting&quot;: Is it the metric you think it is?", Tech. Rep., 2013. [Online]. Available: `http://www.acoustics.asn.au/conference_proceedings/AAS2013/papers/p39.pdf`.

[6] F. Aletta, J. Kang, and Ö. Axelsson, "Soundscape descriptors and a conceptual framework for developing predictive soundscape models", *Landscape and Urban Planning*, vol. 149, pp. 65–74, 2016. DOI: `10.1016/j.landurbplan.2016.02.001`. [Online]. Available: `http://dx.doi.org/10.1016/j.landurbplan.2016.02.001`.

[7] T. Giannakopoulos and A. Pikrakis, *Introduction to audio analysis : a MATLAB approach*, p. 283, ISBN: 9780080993881.

[8] Institute of Electrical and Electronics Engineers., S. IEEE Acoustics, S. IEEE Acoustics, and IEEE Signal Processing Society., *IEEE transactions on acoustics, speech, and signal processing.* [Institute of Electrical and Electronics Engineers]. [Online]. Available: `http://resolver.ebscohost.com/openurl?sid=EBSCO%3aedb&genre=article&issn=00963518&ISBN=&volume=28&issue=4&date=19800104&spage=357&pages=357-366&title=IEEE+Transactions+on+Acoustics%2c+Speech+&atitle=Comparison+of+parametric+representations+for+monosyllabic+word+recognition+in+continuously+spoken+sentences.&aulast=Davis%2c+S.&id=DOI%3a10.1109%2fTASSP.1980.1163420&site=ftf-live`.

[9] P. Dhanalakshmi, S. Palanivel, and V. Ramalingam, "Classification of audio signals using AANN and GMM", *Applied Soft Computing*, vol. 11, pp. 716–723, 2011. DOI: `10.1016/j.asoc.2009.12.033`. [Online]. Available: `https://ac-els-cdn-com.proxy.lib.chalmers.se/S1568494609002907/1-`

s2.0-S1568494609002907-main.pdf?_tid=08f36b2e-dc34-4146-9951-e83e3df3516f&acdnat=1549890789_5fc576aa9ebd96d8b3eda11bed918066.

[10] B. W. Schuller, *Intelligent Audio Analysis*, ser. Signals and Communication Technology. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, ISBN: 978-3-642-36805-9. DOI: 10.1007/978-3-642-36806-6. [Online]. Available: http://link.springer.com/10.1007/978-3-642-36806-6.

[11] V. S. Ramaiah and R. R. Rao, "Multi-Speaker Activity Detection using Zero Crossing Rate", Tech. Rep., 2016. [Online]. Available: http://resolver.ebscohost.com/openurl?sid=EBSCO%3aedb&genre=book&issn=&ISBN=9781509003969&volume=&issue=&date=&spage=0023&pages=0023-26&title=2016+International+Conference+on+Communication+&atitle=Multi-speaker+activity+detection+using+zero+crossing+rate..

[12] D. S. Shete and S. B. Patil, "Zero crossing rate and Energy of the Speech Signal of Devanagari Script", Tech. Rep. 1, 2014, p. 1. [Online]. Available: www.iosrjournals.org.

[13] "The art of pulse detection Application note", Tech. Rep., 2016. [Online]. Available: www.teledyne-spdevices.com.

[14] D. Carmel, A. Yeshurun, and Y. Moshe, *Detection of Alarm Sounds in Noisy Environments*. 2017, ISBN: 9780992862671. [Online]. Available: http://sipl.technion.ac.il/.

[15] H. Li, K.-A. Toh, and L. Li, *Advanced topics in biometrics*. World Scientific, 2012, pp. 49–67, ISBN: 9814287849.

[16] D. Giannoulis, M. MassBerg, and J. D. Reiss, "Digital Dynamic Range Compressor Design-A Tutorial and Analysis , MICHAEL MASSBERG", Tech. Rep., 2012. [Online]. Available: www.gyraf.dk/gy_pd/1176/1176sch.gif.

[17] J. Käsbach, "Implementation and analysis of a multi-channel dynamic-range compressor", Tech. Rep. [Online]. Available: http://johannes.kaesbach.de/Acoustics_files/Implementation%20and%20analysis%20of%20a%20multi-channel%20dynamic%20range%20compressor.pdf.

[18] *Data Smoothing and Outlier Detection - MATLAB &amp; Simulink - MathWorks Nordic.* [Online]. Available: https://se.mathworks.com/help/matlab/data_analysis/data-smoothing-and-outlier-detection.html.

[19] S. Bhattacharyya and G. Sanyal, "Feature Based Audio Steganalysis (FAS)", *Computer Network and Information Security*, vol. 11, pp. 62–73, 2012. DOI: 10.5815/ijcnis.2012.11.08. [Online]. Available: http://www.mecs-press.org/ijcnis/ijcnis-v4-n11/IJCNIS-V4-N11-8.pdf.

[20] Brian Halibisky, *'n'-Dimensional Euclidean Distance.* [Online]. Available: https://hlab.stanford.edu/brian/euclidean_distance_in.html.

[21] K. Hattori and M. Takahashi, "A new edited k-nearest neighbor rule in the pattern classi&quot;cation problem", Tech. Rep. [Online]. Available: https://sci2s.ugr.es/keel/pdf/algorithm/articulo/2000%20-%20PR%20-%20Hattori%20-%20IS_MENN.pdf.

[22] R. Parikh, A. Mathai, S. Parikh, G. Chandra Sekhar, and R. Thomas, "Understanding and using sensitivity, specificity and predictive values.", *Indian journal of ophthalmology*, vol. 56, no. 1, pp. 45–50, 2008, ISSN: 0301-4738. [On-

line]. Available: http://www.ncbi.nlm.nih.gov/pubmed/18158403%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2636062.

[23] A. Baratloo, M. Hosseini, A. Negida, and G. El Ashal, “Part 1: Simple Definition and Calculation of Accuracy, Sensitivity and Specificity.”, *Emergency (Tehran, Iran)*, vol. 3, no. 2, pp. 48–9, 2015, ISSN: 2345-4563. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/26495380%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4614595.

[24] D. Vrajitoru and W. Knight, “Deterministic Analysis of Algorithms”, in, 2014, pp. 169–293. DOI: 10.1007/978-3-319-09888-3{_}5. [Online]. Available: http://link.springer.com/10.1007/978-3-319-09888-3_5.

[25] ——, “Fundamental Notations in Analysis of Algorithms”, in, 2014, pp. 63–94. DOI: 10.1007/978-3-319-09888-3{_}3. [Online]. Available: http://link.springer.com/10.1007/978-3-319-09888-3_3.

[26] Donald Bren School of Information & Computer Science, *Complexity of Python Operations*, 2016. [Online]. Available: https://www.ics.uci.edu/~brgallar/week8_2.html.

[27] M. Lohne, “The Computational Complexity of the Fast Fourier Transform”, Tech. Rep., 2017. [Online]. Available: https://folk.uio.no/mathialo/texts/fftcomplexity.pdf.

[28] J. D. Hol, “Resampling in particle filters”, Tech. Rep., 2004. [Online]. Available: http://www.diva-portal.org/smash/get/diva2:19698/FULLTEXT01.pdf.

[29] *File:Comparison computational complexity.svg - Wikimedia Commons*. [Online]. Available: https://commons.wikimedia.org/wiki/File:Comparison_computational_complexity.svg.

[30] *Cortex-M4 Technical Reference Manual*, 2010. [Online]. Available: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0439b/CHDDIGAC.html.

[31] J. Sanchez and M. P. Canton, *Software solutions for engineers and scientists*, p. 944, ISBN: 1351835890.

[32] T. Lorenser, *Whitepaper: DSP capabilities of Cortex-M4 and Cortex-M7 Processors*, 2016. [Online]. Available: https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/white-paper-dsp-capabilities-of-cortex-m4-and-cortex-m7.

[33] *ARM® Cortex®-M4 Processor Technical Reference Manual, FPU instruction set table*, 2015. [Online]. Available: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.100166_0001_00_en/ric1417175938834.html.

[34] *Loop Iteration - an overview | ScienceDirect Topics*. [Online]. Available: https://www.sciencedirect.com/topics/computer-science/loop-iteration.

[35] G. K. Liu, “Evaluating Gammatone Frequency Cepstral Coefficients with Neural Networks for Emotion Recognition from Speech”, Tech. Rep. [Online]. Available: https://arxiv.org/pdf/1806.09010.pdf.

[36] A. M. Badshah, J. Ahmad, M. Y. Lee, and S. W. Baik, “Divide-and-Conquer based Ensemble to Spot Emotions in Speech using MFCC and Random For-

est", Tech. Rep. [Online]. Available: https://arxiv.org/ftp/arxiv/papers/1610/1610.01382.pdf.

[37] M. I. Abdalla and H. S. Ali, "Wavelet-Based Mel-Frequency Cepstral Coefficients for Speaker Identification using Hidden Markov Models", Tech. Rep. 2, 2010. [Online]. Available: https://arxiv-org.proxy.lib.chalmers.se/ftp/arxiv/papers/1003/1003.5627.pdf.

[38] M. Jalil, F. A. Butt, and A. Malik, "Short-time energy, magnitude, zero crossing rate and autocorrelation measurement for discriminating voiced and unvoiced segments of speech signals", Tech. Rep., 2013. [Online]. Available: http://resolver.ebscohost.com/openurl?sid=EBSCO%3aedb&genre=book&issn=&ISBN=9781467356121&volume=&issue=&date=&spage=208&pages=208-212&title=2013+The+International+Conference+on+Technological+Advances+in+Electrical%2c+Electronics+&atitle=Short-time+energy.

[39] Y. Alsouda, S. Pllana, and A. Kurti, "A Machine Learning Driven IoT Solution for Noise Classification in Smart Cities", Tech. Rep. [Online]. Available: https://arxiv-org.proxy.lib.chalmers.se/pdf/1809.00238.pdf.

[40] B. Xia and C. Bao, "Wiener filtering based speech enhancement with Weighted Denoising Auto-encoder and noise classification", 2014. DOI: 10.1016/j.specom.2014.02.001. [Online]. Available: http://dx.doi.org/10.1016/j.specom.2014.02.001.

[41] D. P. W. Ellis, "DETECTING ALARM SOUNDS", Tech. Rep. [Online]. Available: https://www.ee.columbia.edu/~dpwe/pubs/crac01-alarms.pdf.

[42] R. Palaniappan, K. Sundaraj, S. Sundaraj, N. Huliraj, S. Revadi, and B. Archana, "Pulmonary Acoustic Signal Classification using Autoregressive Coefficients and k-Nearest Neighbor", 2014. DOI: 10.4028/www.scientific.net/AMM.591.211. [Online]. Available: www.scientific.net..

[43] C. Chin-Hsing, H. Wen-Tzeng, T. Tan-Hsu, C. Cheng-Chun, and C. Yuang-Jen, "Using K-Nearest Neighbor Classification to Diagnose Abnormal Lung Sounds", 2015. [Online]. Available: http://resolver.ebscohost.com/openurl?sid=EBSCO%3aedsdoj&genre=article&issn=14248220&ISBN=&volume=15&issue=6&date=20150601&spage=13132&pages=13132-13158&title=Sensors%2c+Vol+15%2c+Iss+6%2c+Pp+13132-13158+(2015)&atitle=Using+K-Nearest+Neighbor+Classificati.

[44] M. Sudarma and G. Harsemadi, "Design and Analysis System of KNN and ID3 Algorithm for Music Classification based on Mood Feature Extraction", 2016. [Online]. Available: http://resolver.ebscohost.com/openurl?sid=EBSCO%3aedb&genre=article&issn=20888708&ISBN=&volume=7&issue=1&date=20170201&spage=486&pages=486-495&title=International+Journal+of+Electrical+&atitle=Design+and+Analysis+System+of+KNN+and+ID3+Algorithm+for+Music+.

[45] C. M. Bishop, *Mel Frequency Cepstral Coefficient (MFCC) tutorial*, 2006. [Online]. Available: http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/.

A

Classification with normalized features

This section presents results from tests using different combinations of features. The green cells in the tables represent correctly classified frames and the red cells represent falsely classified frames.

A.1 MFCCs

Classified label → True label ↓	Speech	Industry	Alarm
Speech	622	96	32
Industry	10	734	6
Alarm	2	271	477

Table A.1: Test using normalized MFCCs, k=1

Accuracy: 81.5%

A.2 MFCCs and Zero-Crossing Rate

Classified label → True label ↓	Speech	Industry	Alarm
Speech	614	113	23
Industry	9	726	15
Alarm	0	243	507

Table A.2: Test using normalized MFCCs and ZCR, k=1

Accuracy: 82.1%

A.3 MFCCs and Pulse Counter

Classified label → True label ↓	Speech	Industry	Alarm
Speech	628	83	39
Industry	30	680	40
Alarm	7	205	538

Table A.3: Test using normalized MFCCs and pulse counter, k=1

Accuracy: 82.0%

A.4 MFCCs, Zero-Crossing Rate and Pulse Counter

Classified label → True label ↓	Speech	Industry	Alarm
Speech	629	94	27
Industry	30	672	48
Alarm	4	213	533

Table A.4: Test using normalized MFCCs, ZCR and pulse counter, k=1

Accuracy: 81.5%

A.5 MFCCs, Zero-Crossing Rate and Energy

Classified label → True label ↓	Speech	Industry	Alarm
Speech	618	109	23
Industry	9	737	4
Alarm	1	210	539

Table A.5: Test using normalized MFCCs, ZCR and energy, k=1

Accuracy: 84.8%

A.6 MFCCs and Energy

Classified label → True label ↓	Speech	Industry	Alarm
Speech	621	106	23
Industry	9	737	4
Alarm	2	176	572

Table A.6: Test using normalized MFCCs and energy, k=1

Accuracy: 85.8%

A.7 Energy

Classified label → True label ↓	Speech	Industry	Alarm
Speech	172	409	169
Industry	215	440	95
Alarm	72	82	596

Table A.7: Test using normalized energy, k=1

Accuracy: 53.7%

A.8 Zero-Crossing Rate

Classified label → True label ↓	Speech	Industry	Alarm
Speech	422	245	83
Industry	298	307	145
Alarm	57	167	526

Table A.8: Test using normalized ZCR, k=1

Accuracy: 55.8%

A.9 MFCCs and Spectral Centroid

Classified label → True label ↓	Speech	Industry	Alarm
Speech	610	100	40
Industry	23	687	40
Alarm	0	231	519

Table A.9: Test using normalized MFCCs and spectral centroid, k=1

Accuracy: 80.7%

A.10 MFCCs, Zero-Crosssing Rate and Spectral Centroid

Classified label → True label ↓	Speech	Industry	Alarm
Speech	605	112	33
Industry	22	680	48
Alarm	0	212	538

Table A.10: Test using normalized MFCCs, ZCR and spectral centroid, k=1

Accuracy: 81.0%

A.11 MFCCs, Pulse Counter and Spectral Centroid

Classified label → True label ↓	Speech	Industry	Alarm
Speech	589	111	50
Industry	36	668	46
Alarm	4	217	529

Table A.11: Test using normalized MFCCs, pulse counter and spectral centroid, k=1

Accuracy: 79.4%

A.12 MFCCs, Zero-Crossing Rate, Energy and Spectral Centroid

Classified label → True label ↓	Speech	Industry	Alarm
Speech	599	125	26
Industry	33	695	22
Alarm	9	109	632

Table A.12: Test using normalized MFCCs, ZCR, energy and spectral centroid, k=1

Accuracy: 85.6%

A.13 MFCCs, Zero-Crossing Rate, Energy and Spectral Flux

Classified label → True label ↓	Speech	Industry	Alarm
Speech	566	153	31
Industry	47	685	18
Alarm	12	95	643

Table A.13: Test using normalized MFCCs, ZCR, energy and spectral flux, k=1

Accuracy: 84.2%

B
Classification of smoothed signal

This appendix presents the results from the classification of smoothed signals. The green cells in the tables represent correctly classified frames and the red cells represent falsely classified frames.

B.1 MFCCs

Classified label → True label ↓	Speech	Industry	Alarm
Speech	329	253	107
Industry	259	346	145
Alarm	27	90	633

Table B.1: Test using normalized MFCCs, k=1

Accuracy: 58.1%

B.2 MFCCs and Zero-Crossing Rate

Classified label → True label ↓	Speech	Industry	Alarm
Speech	329	253	107
Industry	259	346	145
Alarm	27	90	633

Table B.2: Test using normalized MFCCs and ZCR, k=1

Accuracy: 58.1%

B.3 MFCCs and Pulse Counter

Classified label → True label ↓	Speech	Industry	Alarm
Speech	307	271	172
Industry	278	187	285
Alarm	36	87	627

Table B.3: Test using normalized MFCCs and pulse counter, k=1

Accuracy: 49.8%

B.4 MFCCs, Zero-Crossing Rate and Pulse Counter

Classified label → True label ↓	Speech	Industry	Alarm
Speech	307	271	172
Industry	278	187	285
Alarm	36	87	627

Table B.4: Test using normalized MFCCs, ZCR and pulse counter, k=1

Accuracy: 49.8%

B.5 MFCCs and Energy

Classified label → True label ↓	Speech	Industry	Alarm
Speech	329	367	54
Industry	270	423	57
Alarm	53	131	566

Table B.5: Test using normalized MFCCs and energy, k=1

Accuracy: 58.6%

B.6 MFCCs, Zero-Crossing Rate and Energy

Classified label → True label ↓	Speech	Industry	Alarm
Speech	329	367	54
Industry	270	423	57
Alarm	53	131	566

Table B.6: Test using normalized MFCCs, ZCR and energy, k=1

Accuracy: 58.6%

B.7 MFCCs, Zero-Crossing Rate, Energy and Spectral Flux

Classified label → True label ↓	Speech	Industry	Alarm
Speech	329	367	54
Industry	220	467	63
Alarm	76	132	542

Table B.7: Test using normalized MFCCs, ZCR, energy and spectral flux, k=1

Accuracy: 59.5%

C

Evaluation of number of *k*

This appendix presents the results from the evaluation of which number of k that yield the best result.

C.1 MFCCs, Zero-Crossing Rate and Energy

Two values of k were tested for the combination MFCCs, ZCR and energy in order to evaluate the accuracy when using k>1. The green cells in the tables represent correctly classified frames and the red cells represent falsely classified frames.

C.1.1 k = 3

Classified label → True label ↓	Speech	Industry	Alarm
Speech	662	63	23
Industry	60	690	0
Alarm	9	381	360

Table C.1: Test using normalized MFCCs, ZCR and energy, k=3

Accuracy: 76.1%

C.1.2 k = 7

Classified label → True label ↓	Speech	Industry	Alarm
Speech	621	106	0
Industry	60	690	0
Alarm	5	406	339

Table C.2: Test using normalized MFCCs, ZCR and energy, k=7

Accuracy: 73.3%

C.2 MFCCs and Energy

Two values of k were tested for the combination MFCCs and energy in order to evaluate the accuracy when using k>1. The green cells in the tables represent correctly classified frames and the red cells represent falsely classified frames.

C.2.1 k = 3

Classified label → True label ↓	Speech	Industry	Alarm
Speech	663	64	23
Industry	60	690	0
Alarm	10	385	335

Table C.3: Test using normalized MFCCs and energy, k=3

Accuracy: 75.0%

C.2.2 k = 7

Classified label → True label ↓	Speech	Industry	Alarm
Speech	625	102	0
Industry	60	690	0
Alarm	10	436	304

Table C.4: Test using normalized MFCCs and energy, k=7

Accuracy: 72.0%

D

Weighting of distances

This appendix presents the results from the weighting of features. The green cells in the tables represent correctly classified frames and the red cells represent falsely classified frames.

D.1 MFCCs + $\omega_1 \cdot$ Energy

Classified label → True label ↓	Speech	Industry	Alarm
Speech	621	106	23
Industry	9	738	3
Alarm	2	144	604

Table D.1: Test using normalized MFCCs + $\omega_1 \cdot$ Energy, k=1

Accuracy: 87.2%

D.2 MFCCs + $\omega_1 \cdot$ Energy

Classified label → True label ↓	Speech	Industry	Alarm
Speech	625	102	23
Industry	9	739	2
Alarm	2	135	613

Table D.2: Test using normalized MFCCs + $\omega_1 \cdot$ Energy, k=1

Accuracy: 87.9%

D.3 MFCCs + $\omega_1 \cdot$ Energy

Classified label → True label ↓	Speech	Industry	Alarm
Speech	628	99	23
Industry	9	740	1
Alarm	6	131	613

Table D.3: Test using normalized MFCCs + $\omega_1 \cdot$ Energy, k=1

Accuracy: 88.0%

D.4 MFCCs + $\omega_1 \cdot$ ZCR + $\omega_2 \cdot$ Energy

Classified label → True label ↓	Speech	Industry	Alarm
Speech	628	99	23
Industry	9	739	2
Alarm	3	116	631

Table D.4: Test using normalized MFCCs + $\omega_1 \cdot$ ZCR + $\omega_2 \cdot$ Energy, k=1

Accuracy: 88.8%

D.5 MFCCs + $\omega_1 \cdot$ ZCR + $\omega_2 \cdot$ Energy

Classified label → True label ↓	Speech	Industry	Alarm
Speech	627	100	23
Industry	9	737	4
Alarm	2	155	593

Table D.5: Test using normalized MFCCs + $\omega_1 \cdot$ ZCR + $\omega_2 \cdot$ Energy, k=1

Accuracy: 87.0%

D.6 MFCCs + $\omega_1\cdot$ ZCR + $\omega_2\cdot$ Energy

Classified label → True label ↓	Speech	Industry	Alarm
Speech	629	98	23
Industry	9	737	4
Alarm	2	118	630

Table D.6: Test using normalized MFCCs + $\omega_1\cdot$ ZCR + $\omega_2\cdot$ Energy, k=1

Accuracy: 88.7%

D.7 MFCCs + $\omega_1\cdot$ ZCR + $\omega_2\cdot$ Energy

Classified label → True label ↓	Speech	Industry	Alarm
Speech	629	98	23
Industry	9	739	2
Alarm	4	117	629

Table D.7: Test using normalized MFCCs + $\omega_1\cdot$ ZCR + $\omega_2\cdot$ Energy, k=1

Accuracy: 88.8%

D.8 MFCCs + $\omega_1\cdot$ Energy + $\omega_2\cdot$ Spectral Flux

Classified label → True label ↓	Speech	Industry	Alarm
Speech	627	100	23
Industry	9	741	0
Alarm	6	121	623

Table D.8: Test using normalized MFCCs + $\omega_1\cdot$ Energy + $\omega_2\cdot$ Spectral Flux, k=1

Accuracy: 88.5%

D.9 MFCCs + $\omega_1\cdot$ Energy + $\omega_2\cdot$ Spectral Centroid

Classified label → True label ↓	Speech	Industry	Alarm
Speech	631	96	23
Industry	9	740	1
Alarm	5	130	615

Table D.9: Test using normalized MFCCs + $\omega_1\cdot$ Energy + $\omega_2\cdot$ Spectral Centroid, k=1

Accuracy: 88.3%

E

Testing Threshold for Class 'Other'

This appendix presents the results from the tests of threshold for the class 'Other'. The green cells in the tables represent correctly classified frames and the red cells represent falsely classified frames.

E.1 T = 1

Classified label → True label ↓	Speech	Industry	Alarm	Other
Speech	618	97	0	35
Industry	9	739	2	0
Alarm	1	114	375	260
Other	153	275	144	178

Table E.1: Test using normalized $\omega_1\cdot$ MFCCs, $\omega_2\cdot$ ZCR and $\omega_3\cdot$ energy, k=1, T = 1

Accuracy: 63.7%

E.2 T = 0.8

Classified label → True label ↓	Speech	Industry	Alarm	Other
Speech	618	97	0	35
Industry	9	739	2	0
Alarm	0	67	341	342
Other	124	232	102	292

Table E.2: Test using normalized $\omega_1 \cdot$ MFCCs, $\omega_2 \cdot$ ZCR and $\omega_3 \cdot$ energy, k=1, T = 0.8

Accuracy: 66.3%

E.3 T = 1.2

Classified label → True label ↓	Speech	Industry	Alarm	Other
Speech	627	99	0	24
Industry	9	739	2	0
Alarm	2	115	479	154
Other	163	300	196	91

Table E.3: Test using normalized $\omega_1 \cdot$ MFCCs, $\omega_2 \cdot$ ZCR and $\omega_3 \cdot$ energy, k=1, T = 1.2

Accuracy: 64.5%

E.4 T = 0.7

Classified label → True label ↓	Speech	Industry	Alarm	Other
Speech	476	28	0	246
Industry	9	734	1	6
Alarm	0	26	319	405
Other	84	181	49	436

Table E.4: Test using normalized $\omega_1 \cdot$ MFCCs, $\omega_2 \cdot$ ZCR and $\omega_3 \cdot$ energy, k=1, T = 0.7

Accuracy: 65.5%