Framework for Analyzing Sounds of Home Environment for Device Recognition

Svilen Dimitrov

A thesis submitted in partial satisfaction of the requirements for the degree Master of Science at the Department of Computer Science at Saarland University

SAARLAND UNIVERSITY

Supervisor

Prof. Dr. Dr. h.c. mult. Wolfgang Wahlster
German Research Center for Artificial Intelligence, DFKI

Reviewers

Prof. Dr. Dr. h.c. mult. Wolfgang Wahlster
German Research Center for Artificial Intelligence, DFKI

Dr. Boris Brandherm
German Research Center for Artificial Intelligence, DFKI

Advisor

Dipl.-Inform. Jochen Frey
German Research Center for Artificial Intelligence, DFKI

Autor

Svilen Dimitrov

Bruchwiesenanlage 4,
D-66125 Saarbrücken, Germany

Submission Date

15. May 2014

Statement in Lieu of an Oath

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Saarbrücken / 15. May 2014

(Svilen Dimitrov)

Declaration of Consent

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

Saarbrücken / 15. May 2014

(Svilen Dimitrov)

Abstract

Home environments are one of the subjects of study by Ambient Intelligent Systems for various purposes, including developments of elderly assistance systems and energy consumption optimization. Sensing the environment, via different sensors, is the first and crucial component of every Ambient Intelligent System. In this thesis we design and develop the Sound-based Device Recognition Framework to investigate the application of environmental sounds usage for touch-free audio-based device recognition in a home environment. For this purpose, we study the characteristics of the sounds dispersed by devices in a home environment. We use the acquired knowledge to implement different Sound Processing techniques for the extraction of a flexible set of features, which can be determined both manually and automatically. For the classification of gathered device acoustic fingerprints we use multiple optimized straightforward techniques of Supervised Learning as well as integrated established ones. Furthermore, we use a feedback from the user for creating an incremental learning system. After establishing a recognition basis for the recognition of fixed length sound buffers on demand, we implement a live recognition mode for real-time environment monitoring, providing runtime setup adjustments. These include changing the selected features, switching between Machine Learning algorithms, and recognition time interval choice, without interruption for modifications of the trained data. We then extend our work with the recognition of untrained simultaneously working known devices, utilizing Semi-supervised Learning. Finally, we create an automatic test utility to evaluate different aspects of the developed framework, including recognition rate performance for the different combinations of features and Machine Learning algorithms, as well as to study the reliability of the automatic mixing of trained data. Our evaluation shows satisfactory results in all tested aspects. Therefore we consider the development of our Sound-based Device Recognition Framework as complete and providing a solid base for further research.

Acknowledgements

I would like to express my gratitude to Jochen Frey and Dr. Boris Brandherm for the admission of this challenging and innovative topic of my thesis, as well as to Prof. Dr. Wolfgang Wahlster and his chair for the provided research opportunity in my desired field.

Secondly I want to thank Yavor Kaloyanov, Dr. Bilyana Taneva and Dr. Stefan Popov for backing me up with useful programming tips and fruitful advices. At this place I am much obliged also to everyone, who shared his work and experience over the internet.

Finally I am deeply grateful to my Family for their unconditional support, patience and understanding ☺

Contents

ABSTRACT .... **III**

ACKNOWLEDGEMENTS .... **V**

CONTENTS .... **VII**

1. INTRODUCTION **1**

1.1. MOTIVATION OF SOUND-BASED RECOGNITION RESEARCH 2
1.2. SCENARIOS OF AMBIENT INTELLIGENT SYSTEMS 3
1.2.1. Ambient Assisted Living *3*
1.2.2. E-Energy *3*
1.3. PROBLEM OF DEVICE RECOGNITION SYSTEMS WITHOUT USAGE OF ENVIRONMENTAL SOUND INFORMATION 3
1.4. SOUND-BASED DEVICE RECOGNITION APPROACH 4
1.5. CHALLENGES 4
1.6. RESEARCH QUESTIONS 4
1.6.1. Reducing complexity of recognition system with retaining recognition rate *4*
1.6.2. What are the relevant sound features for recognizing devices? *5*
1.6.3. Mixing training data for recognition of unanticipated complex activities performed with the usage of known devices *5*
1.7. THESIS OUTLINE 5

2. THEORETICAL BACKGROUND **7**

2.1. AMBIENT INTELLIGENCE 7
2.1.1. Activity vs. Device Recognition *9*
2.1.2. Activities of Daily Living Performed with Devices in a Home Environment *9*
2.1.3. Activity Zones *10*
2.1.4. Techniques for Sensing a Home Environment *11*
2.2. SOUND PROCESSING FOR RECOGNITION 12
2.2.1. Sound Signal Acquisition by Humans and Machines *13*
2.2.2. Sound Filtering *14*
2.2.3. Sound Spectrum Representation *17*
2.2.4. Sound Features *23*
2.3. MACHINE LEARNING 25
2.3.1. Supervised and Semi-Supervised Learning *26*
2.3.2. Nearest Neighbor Methods *26*
2.3.3. Bayes Classification *27*
2.3.4. Feature Selection *29*
2.3.5. Feature Normalization *30*

3. RELATED WORK....31

3.1. SIGNAL PROCESSING 32
3.2. SPEECH RECOGNITION 33
3.3. MUSIC RECOGNITION 34
3.4. ACTIVITY RECOGNITION 36
3.5. DEVICE RECOGNITION 38
3.6. OVERVIEW OF RELATED WORK 39
3.6.1. Comparison of Related Works in Terms of Functionality 39
3.6.2. Detailed Comparison of Activity Recognition Systems 41
3.6.3. Overview of Recognition Capabilities of Related Works 42

4. CONCEPT OF SOUND-BASED DEVICE RECOGNITION 43

4.1. ARCHITECTURE 44
4.1.1. Abstract Architecture 44
4.1.2. Concrete Architecture 44
4.1.3. Spiral Development Process Model 45
4.2. ENVIRONMENT AND INSTALLATION REQUIREMENTS 46
4.2.1. Environment Characteristics 46
4.2.2. Recording Setup 47
4.2.3. Installation from a Software Perspective 47
4.3. SOUND UNIT 48
4.3.1. Sound Processing 48
4.3.2. Features Choice 48
4.4. MACHINE LEARNING ELEMENT 49
4.4.1. Implementation Considerations 49
4.4.2. Adaptivity 49
4.5. INTEGRATION 51
4.5.1. As a Class Library 51
4.5.2. As a Running Service 52
4.6. PRIVACY ISSUES 52

5. SOUND-BASED DEVICE RECOGNITION FRAMEWORK 53

5.1. ENVIRONMENT AND HARDWARE SETUP 53
5.2. FRAMEWORK SOLUTION IN VS2012 55
5.3. FRAMEWORK GRAPHIC USER INTERFACE 55
5.4. SOUND CLASS 57
5.4.1. Sound Transformation 57
5.4.2. Sound Filtering 60
5.4.3. Features Extraction 62
5.4.4. Sound Mixing 66
5.5. MACHINE LEARNING CLASS 68
5.5.1. Activity Class 68
5.5.2. Automatic Feature Selection and Priority Determination 71
5.5.3. Feature Metrics and Normalization 72
5.5.4. Implemented Classifying Algorithms 73
5.5.5. Imported External Libraries 76
5.6. PROCESS FROM ACTIVITY TO ITS DEVICE RECOGNITION 78
5.6.1. Training 78
5.6.2. Recognition 79
5.6.3. Real-Time Recognition 79
5.6.4. Integration Interface 80

6. EVALUATION 83

6.1. TEST SETUP 83
6.1.1. Test Environment Setup and Corpus 83
6.1.2. Software Test Module 85
6.2. TEST RESULTS 88
6.2.1. Identifying the Best Feature Set for the Tests 88
6.2.2. Running Detailed Tests 90
6.2.3. Test of Mixing 92
6.2.4. Comparison between NNMD and Infer.NET 93

7. CONCLUSION & OUTLOOK 95

7.1. CONTRIBUTIONS 95
7.2. REVIEW OF RESEARCH QUESTIONS 96
7.2.1. Could we reduce the complexity of a recognition system, while maintaining the recognition rate?...... 96
7.2.2. Which are the relevant sound features for the task of device recognition, and can we choose these automatically? 96
7.2.3. Could we automatically mix activities for their untrained recognition? 96
7.3. OUTLOOK 96
7.4. OPEN PROBLEMS 97

BIBLIOGRAPHY 99

APPENDIX 105

A. Class Organization 105
B. Activity Type and Device Type 106
C. Multi-dimensional Nearest Neighbor 107
D. Test Records Spectrograms 108
E. Mixed Records Spectrograms 114

Chapter 1

Introduction

In our modern way of life we are surrounded by an increasing number of devices, which we use to perform large variety of activities. Some of those activities are not always straightforward and we often need some assistance to perform them. To make this happen, one has to give some intelligence to the devices to make them able to understand our intentions and fit into our needs. In other words: making those devices sensitive and responsive to our presence, instead of relying on us to learn how to operate them. Making the devices more sensitive to human actions is one of the goals in Activity Recognition. This is the first step of designing a so called Ambient Intelligent System, which at first anticipates human actions with their purpose in a given environment, and then acts in an intelligent manner by predicting and assisting future actions. This should hold especially in the case, where humans are experiencing difficulties in performing those actions, but there are many further applications, such as optimizing electrical energy consumption.

In this thesis we study the sensing component of an Ambient Intelligent System. For this purpose we introduce our Sound-based Device Recognition Framework – a fully developed system for device recognition based on analyzing environmental sounds. Our environment consists of a normal home. Its devices, which are to be recognized, are commonly used for performing daily tasks, like electrical toothbrush or shaver. Most of those devices create or disperse sounds, while being used to perform different activities. We study the most frequently used devices and the nature of the sounds, which accompany their usage. We then use this knowledge to transform those sounds to the selected different acoustic representations in order to extract their most telling characteristics for the purpose of sound-based device fingerprinting. For the gathering of acoustic fingerprints we build a database, which is later used as a knowledgebase for further classification tasks. The latter are performed by trying out different Machine Learning algorithms and evaluating their performance in terms of complexity, recognition accuracy and adaptation capability. We then expand our work by adding further system capabilities, like live recognition using buffers of variable length or automatic mixing of different sounds for recognition of untrained combination of known devices. Finally, we create a module for automatic testing and use it to evaluate different aspects of the implemented recognition techniques in a home setup.

1.1. Motivation of Sound-based Recognition Research

Ambient Intelligence has become a trending field in computer science as a natural consequence of high instrumented environments, where each device is a target to embedding a microchip with increasing computational power. However, not all devices possess some sort of intelligence, nor need they. Furthermore, the so called intelligent devices are often not meant to be intelligent in a way besides accomplishing their function in a constant manner, regardless of its environmental effects and regardless of potential improvement possibilities. From this standpoint, Ambient Intelligence is about providing an intelligent interaction between different environmental parts, to integrate them in a whole intelligent system, which acts as one and adapts to further environmental changes and increases its knowledge (Weber, Rabaey, & Aarts, 2005) (Cook, Augusto, & Jakkula, 2009).

The first component of such system is the environment sensing component, which has as a function to recognize all types of activities, ranging from long to short term and from large scale to small scale activities. Video cameras are a popular choice for a sensor when it comes to recognizing user activities (Tapia, Intille, & Larson, 2004), because they can provide a detailed knowledge about the ongoing activities in a home environment. On the other hand, cameras have some fallacies such as being obtrusive for its inhabitants regarding their presence (Brey, 2005), and usually suffer from bad recognition in sub-optimal light conditions. In addition, cameras are expensive and require computationally intensive algorithms for recognition (Ke, et al., 2013).

Another touch-free technique of recognition, regarding the human perception, is based on analyzing the audible sounds in a given environment. However, most of the sound-based recognizers are limited in recognizing human speech, together with some of its characteristics like speaker recognition and his emotional state in order to obtain detailed information about their subject of interest. On the other hand there are very few studies, which aim to examine in abstract way the daily human activities in a home environment according to their acoustic characteristics (Stager, et al., 2004) (Temko, et al., 2006) (Istrate, et al., 2008) (Wang, et al., 2008) (Lozano, et al., 2010) (Karbasi, et al., Dec 2011) (Sehili, et al., 2012). Despite their generalized way of analyzing sounds, they are all developed in a healthcare perspective and often make the implication that certain sound implies certain activity, which is not necessarily true. This slightly differs from our perspective of building up a set of audibly distinguishable entities, most of which being devices in active state, without attempting to interpret their further meaning. Furthermore in a Sound Processing standpoint, all of the mentioned studies use very similar techniques, which represent a small range of the available sound transformation techniques for recognition (Müller, Ellis, Klapuri, & Richard, 2011). In this context, this study aims to integrate and evaluate also further recognition methods, based on refining and tuning of existing Sound Processing techniques and various Machine Learning algorithms, for the task of device recognition.

1.2. Scenarios of Ambient Intelligent Systems

Among the many applications of Sound-based Device Recognition in Ambient Intelligent Systems, we select two of those scenarios, which we consider as primary use cases for designing our framework. The first scenario is in an area called Ambient Assisted Living, which aims to assist humans, while they perform different activities in their homes. The second scenario is in the area of E-Energy, which aims to optimize the energy usage of households based on energy consumption models.

1.2.1. Ambient Assisted Living

John, 75, lives alone and suffers from hearing disability. He is active at night time and performs normal daily living tasks like cleaning or brushing his teeth. He often forgets to turn off the devices he uses, like vacuum cleaner or his electrical toothbrush. He already has an Ambient Assisted Living system in his home, but it cannot recognize the state of the devices with the conventional sensors like cameras. What John would like to have in his assistance system is a microphone to listen and tell him what is happening according to the produced sounds.

1.2.2. E-Energy

Acme power plants want to optimize the energy allocation for its users. For this purpose they have designed a system for energy consumption models based on device recognition via energy monitoring sockets. Although the system works fine in the general case, there are some deviations, which could not be accurately recognized. So they are looking for an unobtrusive way to increase their precision.

1.3. Problem of Device Recognition Systems without Usage of Environmental Sound Information

In both of the above scenarios we anticipate insufficiency of the installed recognition systems. In the first scenario we anticipate an example, where an audio-based recognition system is inevitable in order to substitute intelligently human hearing awareness. In this case a correct recognition of devices might eventually mean better house assistance systems and reduced risk when forgetting dangerous devices turned on. Recognition techniques based on video are not suitable to recognize forgotten devices in active state, because in most cases, they don’t have motion characteristics. For the second scenario there are already available recognition methods, like the installation of energy monitoring sockets for each device plugged or using a power analyzer (Belley, et al., 2013). However, besides the expensive need of attaching multiple energy monitoring sockets or a power analyzer, not all of the electrical devices are constantly connected to the power, because they rely on batteries for their convenient usage like the electrical toothbrush. So we can see that in both cases there is a lack of sensor input to build up the desired knowledge about the environment, which can be covered by the integration of a Sound-based Recognition component.

1.4. Sound-based Device Recognition Approach

In both scenarios we observe a demand of a low cost addition to existing recognition systems and we identify the problem as a lack of sound-based sensing information of the environment. Regarding those points, this study aims to use existing microphones in the infrastructure in order to recognize different devices, or if integrated to an existing system, to refine its recognition rate. Theoretically thought, one might not need a very extensive sound feature extraction and complicated Machine Learning algorithm, because the set of devices that are to be recognized is relatively small and the existing hardware in every home computer or smartphone is good enough. The goal is to find the sufficient combination between set of features and Machine Learning algorithms to obtain a robust personalized sound-based device recognizer.

1.5. Challenges

The described approach provides an overview of a big system with many facets, which originate from different areas. To gather and intelligently combine their knowledge in a single system is a challenge in software engineering perspective. So the first design requirement, which such system should satisfy, is to be extensible for addition of new capabilities, as well as flexible for linking newly integrated modules. The second design requirement on the framework, coming from the scenarios, is to be exportable for eventual integration into a bigger system.

In a technical perspective, it is a challenge not only to implement many techniques for Sound Processing and Machine Learning, but also to implement them efficiently in order to provide a real-time recognition. It is also a hard task to lower the hardware requirements to any single microphone, instead of demanding a high-end recording setup.

1.6. Research Questions

Besides the mentioned challenges in building up the Sound-based Device Recognition Framework, we extract some of the main research questions, which are either partially or not studied by related works, which this thesis will try to answer.

1.6.1. Reducing complexity of recognition system with retaining recognition rate

Often reducing the size of a recognition scope leads to better recognition rate and reduces the complexity of the system. We are therefore interested, whether within a personalized and detailed view of a household setup, one can reduce the complexity of its audio recognition system, while maintaining a good recognition rate. We will try to solve this problem by implementing multiple sound pattern recognition techniques and compare their recognition rate after testing them in a home environment.

1.6.2. What are the relevant sound features for recognizing devices?

We will try to find out, which are the most important features to categorize different devices according to their sound. To find those features we will implement multiple sound features and will make recognition accuracy test with all of them, as well as combinations between them. Finally we will test, whether the system can identify the relevant features for recognition automatically, and will try to provide a straightforward implementation of automatic feature selection.

1.6.3. Mixing training data for recognition of unanticipated complex activities performed with the usage of known devices

In our everyday life we often perform complex activities, during which multiple devices are used simultaneously, like brushing teeth while showering. We are interested whether we can automatize the process of sound mixing and extracting features for the recognition task of known parallel working devices, which haven't been trained by the system.

1.7. Thesis Outline

The rest of this thesis is structured as follows:

Chapter 2 starts with essential background in the fields of Ambient Intelligence, Sound Processing and Machine Learning. Most of the examples in that chapter are directly from our later setup and implementation.

Chapter 3 reviews related work starting from the general case of Signal Processing for recognition, goes through Speech and Music Recognition, and to the case of generalized Sound-based Activity Recognition being most related to our case of Sound-based Device Recognition. At the end there is a summary and comparison of the mentioned systems.

In Chapter 4 we study the concept of Sound-based Device Recognition. We look into the environment of our use cases and its requirements to our feature design. We then define the two major components of a Sound-based Recognition system with their desired main functionality.

In Chapter 5 follows the implementation of a Sound-based Device Recognition Framework, which satisfies and extends some of the capabilities defined in the previous chapter, as well as, providing further significant developer functionality for tasks, not mentioned in Chapter 4, with automated testing of its different aspects being the most important.

After development details follows a detailed evaluation of the system and its core functionality in Chapter 6.

In Chapter 7 we make a conclusion of this work and its contribution, as well as pointing out its possible development followings and mentioning some complex problems in the Sound-based Activity Recognition field, which can be a subject of bigger studies.

Chapter 2

Theoretical Background

In this chapter we introduce three important branches of Artificial Intelligence – Ambient Intelligence (Subchapter 2.1), Sound Processing for Recognition (Subchapter 2.2) and Machine Learning (Subchapter 2.3). All of them are tightly interconnected, although representing separate disciplines (as illustrated in Figure 2.1).

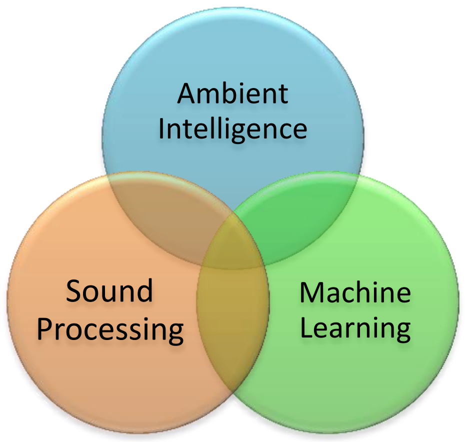


Figure 2.1: *Illustration of all three interleaving research areas, which are of highest influence in this thesis*

One should note that most of the examples and illustrations which occur in this chapter are directly exported from our later architecture, introduced in Chapter 4, and implementation, presented in Chapter 5. A few other examples and terminology are presented here as well, in order to help us understand the research field of Sound-based Activity Recognition with its current state of art, summarized in Chapter 3.

2.1. Ambient Intelligence

Ambient Intelligence, as defined by (Weber, Rabaey, & Aarts, 2005), "is the vision of a technology that will become invisibly embedded in our natural surroundings, present whenever we need it, enabled by simple and effortless interactions, attuned to all our senses, adaptive to users and context-sensitive, and autonomous." Hence one can abstractly describe the technology as perceiving and acting in a given environment with following characteristics,

- where technology is embedded, hidden in the background
- that is sensitive, adaptive, and responsive to the presence of people and objects

- that augments activities through smart non-explicit assistance
- that preserves security, privacy and trustworthiness while utilizing information when needed and appropriate (Weber, Rabaey, & Aarts, 2005)

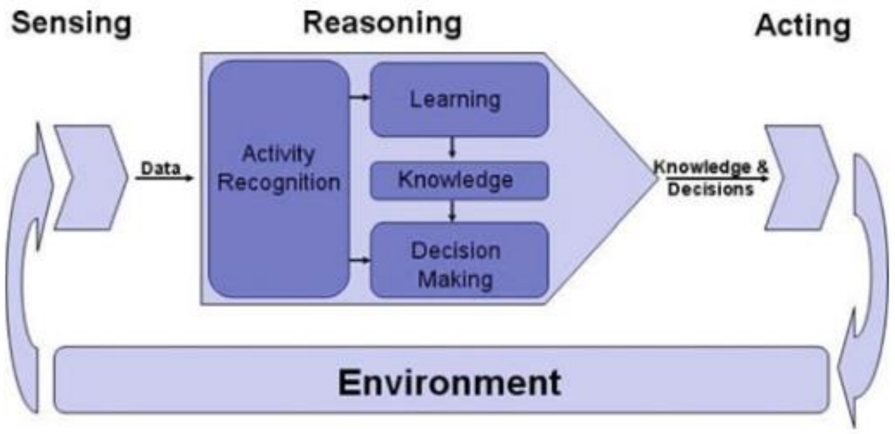


Figure 2.2: *A detailed view of Intelligent System presented in* (Aztiria, Izaguirre, & Augusto, 2010)

An overview of an Ambient Intelligent system can be seen in Figure 2.2. The first part, as mentioned is the environment, which we already briefly characterized. The second part is the environment sensing via different techniques, discussed in Section 2.1.4 after clarifying the environments properties and our recognition scope. Next comes, a reasoning component, which consists of Activity Recognition, Learning, Knowledge, and Decision Making, based on the observations made. Here in this thesis, we will investigate the recognition part, which observes the sound dispersing part of the environment, as described in Section 2.1.1. More specifically, we will discuss the Activities of Daily Living performed with the corresponding devices in Section 2.1.2. and the typical zones, where those activities occur, in Section 2.1.3.

There are different ways in, which we can look at the environment. One way is to look at the environment in an Ambient Assisted Living context, as described in the first introduction scenario (1.2.1.). Ambient Assisted Living is a scientific area of research in order to improve and assist elderly and people with special need in their daily life. Example of assistance realization is by anticipating the daily movements of residents in a home in order to predict their actions in order to assist or even automate them. Another aspect of home environment is its power consumption. Here we have the possibility of automating and optimizing the duration of usage of electrical devices, which affects the overall power consumption. So in this framework we are interested in concrete environmental aspects and designing their improvement. Finally in section 2.1.4., we will make an overview of the most popular techniques for recognition of activities of daily living.

2.1.1. Activity vs. Device Recognition

Activity Recognition is a wide scientific area that aims to recognize all types of activities, mostly performed by an individual or a group of individuals. One should not forget activities performed by different devices such as laundry. However, while making an implication from activity to the usage of some is ok, the other way round is not necessarily true. For example, if we have an activity like cleaning with a vacuum cleaner, we can assume that there is a vacuum cleaner device being used, while recognizing a sound of a vacuum cleaner doesn't imply the activity of vacuum cleaning, because it might imply the possibility of a forgotten running vacuum cleaner. To make such an implication from sounds of devices to the corresponding activity in a confident manner, one has to study continuously the patterns of those activities in terms of their complete occurrence and characteristics using complex Machine Learning algorithms, which we mention in the outlook in Chapter 7. Where probing real-time sound extracts for their live recognition is not strong enough to build up generalized statements about the entireness of their reason of occurrence, just like taking a small fragment of a picture is not enough to build up the whole picture. In relation to this, the scope of this thesis will be to investigate the subject of device recognition in a home environment, with a few exceptions of non-device recognitions, which will be explained in Chapter 5.

2.1.2. Activities of Daily Living Performed with Devices in a Home Environment

Activities of daily living are the most frequent tasks, which an individual performs during his daily time, and are most studied in terms of personal healthcare (Lawton & Brody, 1968) (Katz, 1983). In this Section we call activities of daily living in a home environment, the most frequent tasks, which an individual performs at his home using different devices. Typical examples are brushing teeth or listening to music, while other activities like speaking are not necessarily performed with the usage of some device (see Figure 2.3). Some of those activities can be executed in a combination, creating a new complex activity. One can derive deeper knowledge of the activities like recognizing speakers' personality and mood, while he is speaking or guessing the model of a used toothbrush.

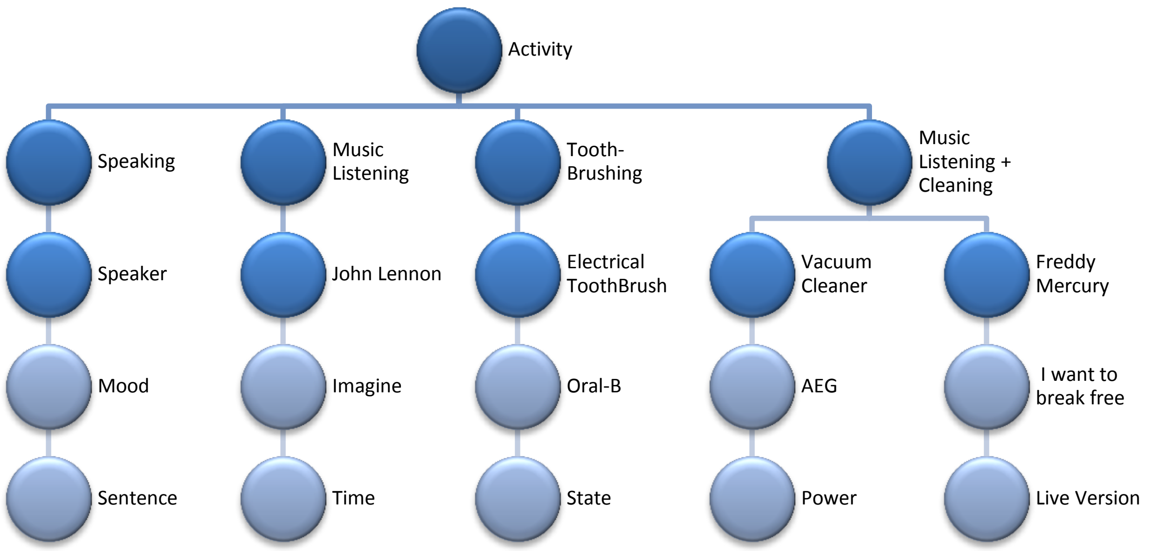


Figure 2.3: *Illustration of different sample types of activities and the different levels of details about those activities. The first type, Speaking, provides information about the speaker, his mood and what he is saying. The second type, Tooth-brushing, provides information about the way one brushes teeth and the device and its state he is using. Music Listening as third type reveals information about the listened song and its interpreter, as well as the time stamp. The fourth type, a combination between Music Listening and Cleaning, is a complex example, which reveals information about those activities as well*

In the field of sound-based activity recognition, most studies for recognition of ADLs are music listening recognition Speech Recognition (see Section 3.2) and (see Section 3.3). Development of general sound-based recognizers started in the last several years (see Section 3.3).

2.1.3. Activity Zones

Activity Zones are parts of the environment where some set of activities occur. Under parts of the environment one can consider different rooms like the kitchen, where one typically performs meal preparation and cleaning. One can obtain further assignment of activity zones in the kitchen as well, like marking the oven area as activity zone, where one usually performs cooking related actions (see Figure 2.4).

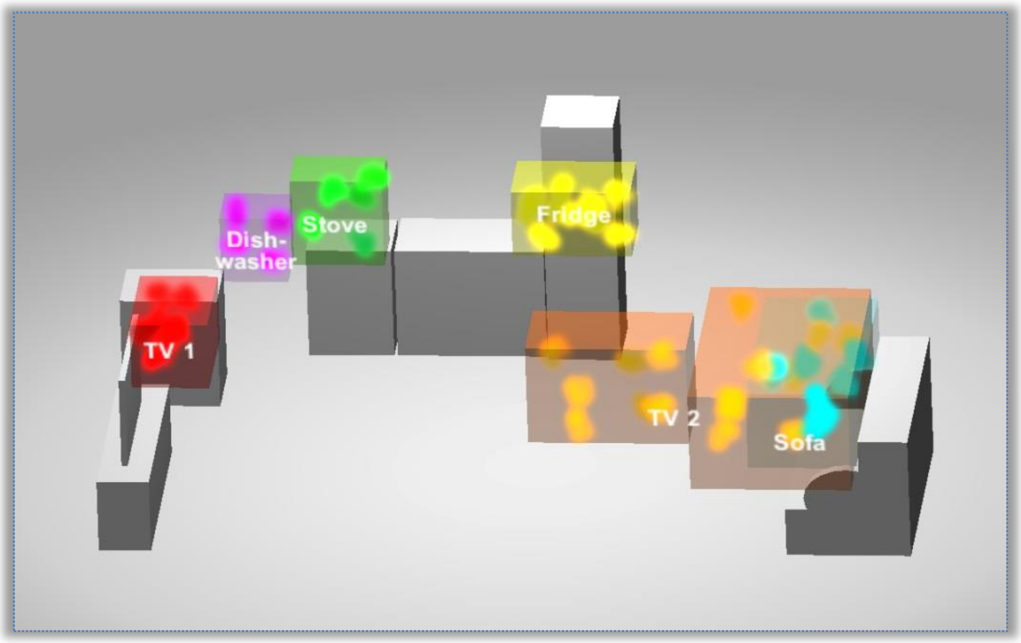


Figure 2.4: *Self-adaptable three-dimensional modeling of Activity Zones in a single room occupancy by* (Frey, Neurohr, & Brandherm, 2014)

For us activity zones play an important role, in every level of their detail. For example, in a multiple room environment with microphones in every room the recognizer can extract the spatial information, in which room is the sound source, and thus reduce the set of possible results. A concrete example for this case is the assumption that the inhabitant cannot take a shower in the kitchen, so the recognizer removes the shower from the list of possible outcomes. From the detailed case of activity zone, we can use the knowledge that if one activity occurs in a specific place, its sound characteristics are very similar between its different occurrences. The later holds especially in the case, where electrical devices with a fixed set of usage programs are mounted stationary, like a toilet.

2.1.4. Techniques for Sensing a Home Environment

In the previous section we have seen an example of three dimensional modeling of activity zones via environmental sensing, performed with Kinect[1]. However, there are many other techniques for activity recognition in a home environment (Tapia, Intille, & Larson, 2004), which could be categorized by the chosen sensors for recognition, including video cameras, audio microphones, and wearable devices, as well as their combinations. A comparison between the different techniques with their most general pros and cons is shown in Table 2.1.

[1] A line of motion sensing input devices providing three-dimensional video output and data from multiple audio channels. Online at http://www.microsoft.com/en-us/kinectforwindows/

Sensor	Pros	Cons
Video	• Good recognition rate • Details with multiple cameras or 3d cameras	• Problems with bad lighting • Exhaustive processing • Low refresh rate • Expensive
Audio	• High refresh rate • Always available • Low cost • No need of focusing	• Noise decreases recognition
Wearable	• Easy processing and classification • Exact information	• Annoying • Expensive
Combination	• Covering weak spots • Good recognition rate	• Complex Machine Learning • Exhaustive processing

Table 2.1 *Comparison between the different recognition techniques according to their sensor types they use*

2.2. Sound Processing for Recognition

In the terminology of Pattern Recognition, Sound Processing is a way to transform the sound wave in order to emphasize different features. Under the term Sound Transformation one understands a wide field of different sound manipulation techniques for different purposes (See Figure 2.5). Simple ones include different filters often realized also by the recording hardware before digitalization of the sound signal (Section 2.2.1.), but usually performed in a preprocessing step (Section 2.2.2.). To understand the mechanisms of Sound Processing in this thesis, we explain complex mathematical transformations in Section 2.2.3. The latter will help us characterizing the sound the way we need in order to extract recognition relevant features (Section 2.2.4).

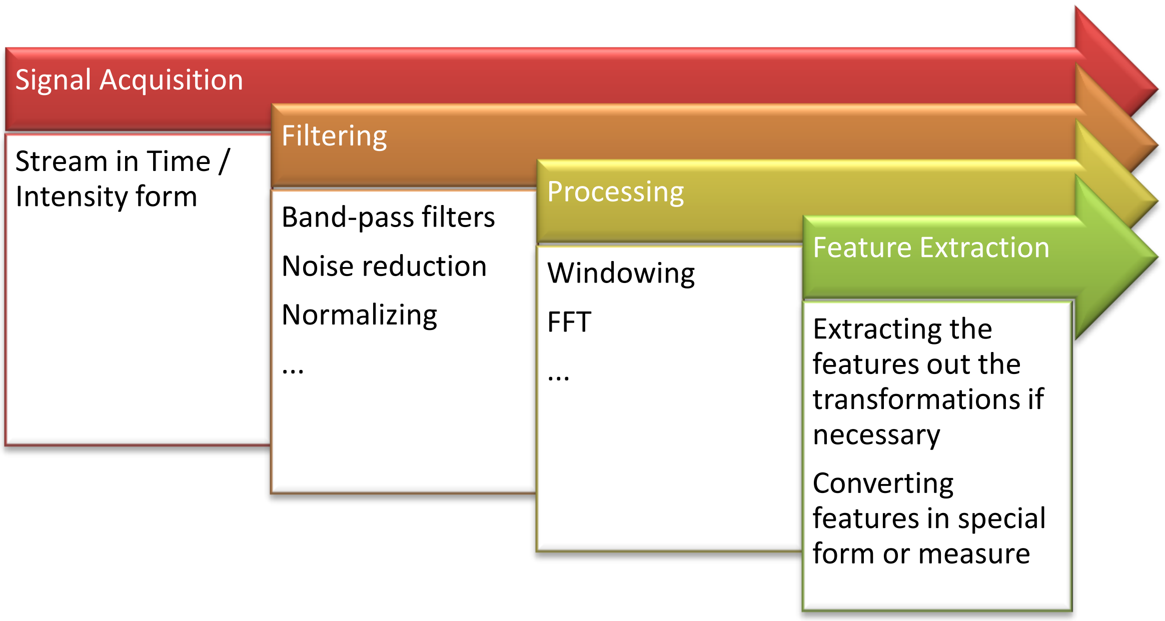

Signal Acquisition
Stream in Time / Intensity form
Filtering
Band-pass filters
Noise reduction
Normalizing
...
Processing
Windowing
FFT
...
Feature Extraction
Extracting the features out the transformations if necessary
Converting features in special form or measure

Figure 2.5: *General dataflow of Signal Processing for every recognition system. Note that every of the points can be surpassed except the signal. E.g. one might want to skip filtering the signal or skip processing it, since there are enough features, which could be extracted out of the raw signal*

2.2.1. Sound Signal Acquisition by Humans and Machines

Signals perceived by humans through their hearing system are called audio signals. Those signals come from a sound source which vibrates in the audible frequency range approximately between 20 Hz and 20 000 Hz. The resulting vibrations are causing different pressure (amplitude) in a medium (usually air), which causes the human eardrum to vibrate and send the information to the brain for interpretation (see Figure 2.6).

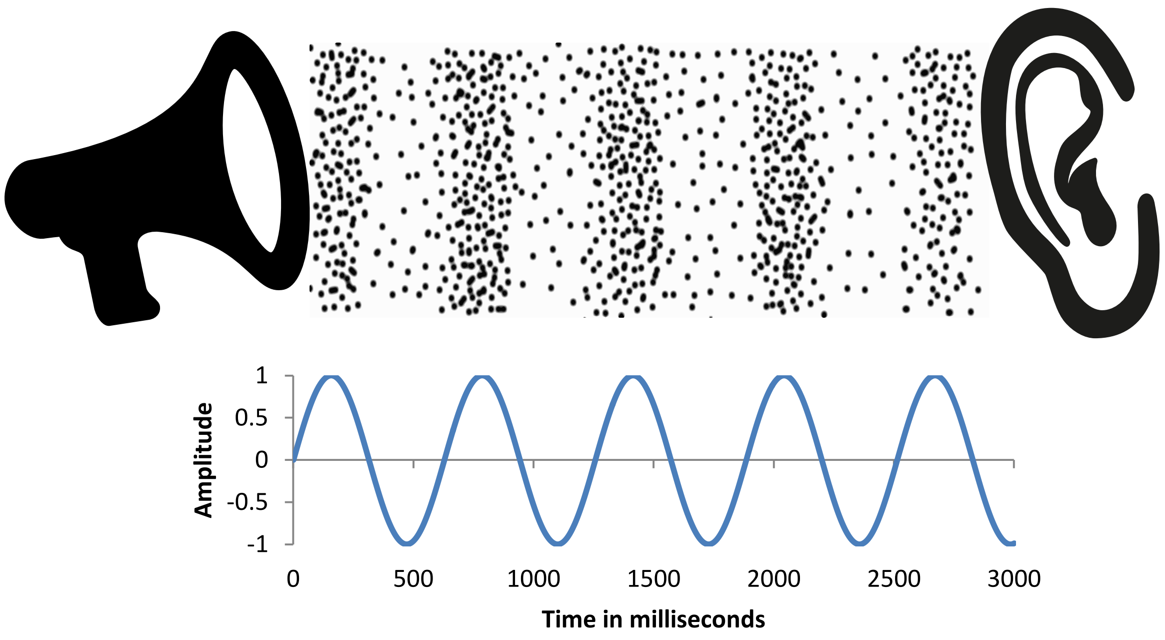


Figure 2.6: *Illustration of the parallel between air pressure waves (level according to dots density) and amplitude plot (amplitude on Y-axis and time on X-axis). Signal of about 1.6 Hz makes 4 complete cycles (periods)*

Similar to that process, a microphone can act as a sound receiver and send the perceived audio signals to a computer, which firstly converts the analog input into digital one. So we are firstly speaking of a sample rate, which refers to the number of sample points per second, and bits per sample, which refers to the number of bits, which we need to encode each sound sample (see Figure 2.7).

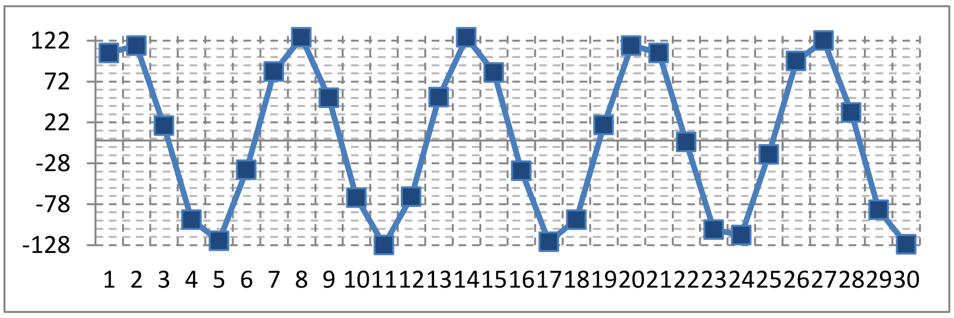


Figure 2.7: *Digitized representation of the same wave form from Figure 2.6, but with 100 times lower sample rate using 8 bits resolution*

In this work, the sound signals resulted by different activities are caught by a computer and processed as an uncompressed LPCM WAVE (Linear Pulse Code Modulation Waveform Audio File Format) sound. It is the pure form of digital sound that the computer captures. The most important settings in this format are the number of channels, as well as the sample rate and the bits per sample (Figure 2.8 shows detailed format layout). By its design and characteristics the WAVE format permits to be used for digital storage of any kind of waves with random number of channels and precision (IBM Corporation & Microsoft Corporation, 1991) (Microsoft Corporation, 2007).

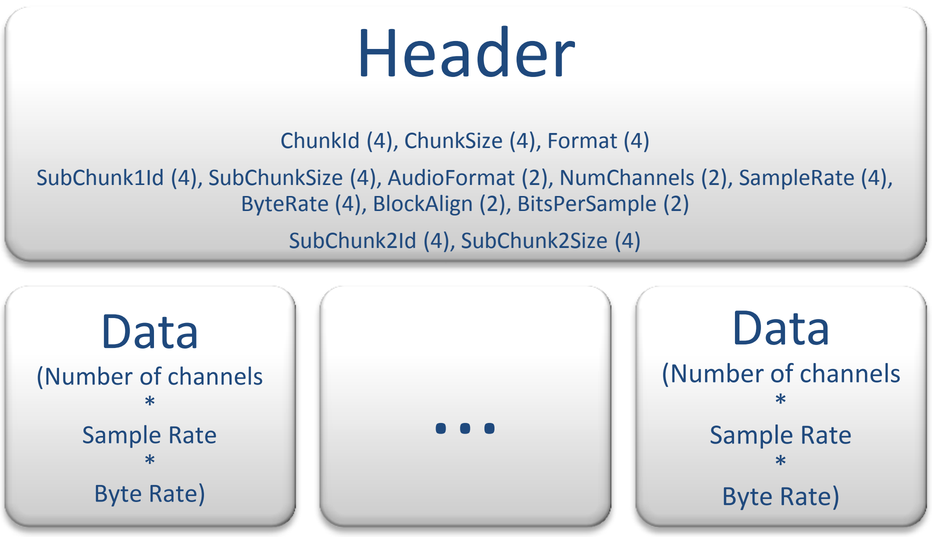

Header
ChunkId (4), ChunkSize (4), Format (4)
SubChunk1Id (4), SubChunkSize (4), AudioFormat (2), NumChannels (2), SampleRate (4), ByteRate (4), BlockAlign (2), BitsPerSample (2)
SubChunk2Id (4), SubChunk2Size (4)
Data
(Number of channels * Sample Rate * Byte Rate)
...
Data
(Number of channels * Sample Rate * Byte Rate)

Figure 2.8: *WAVE format overview. Each segment of the both Header and Data chunks is presented with its size in braces*

A further important note on the recording hardware is that it always supports a high sample rate of at least 44100 Hz, which is also the standard for CD quality. This means not only covering the human hearing frequency range, by any low cost system, but a high refresh rate as well. So we have practically the possibility to exploit every recording system for our recognition purposes, which is a huge advantage compared to standard built-in cameras, which offer refresh rate between 24 and 30 Hz, and are far from covering the visible range of humans, aside from being more expensive.

2.2.2. Sound Filtering

Often the audio recording and playback equipment has some deviations in its frequency response[2]. In order to fix those deviations we need to filter the signal with a set of Filters to obtain the desired frequency response. We need Filters also when we want to operate in a special range of frequencies, cutting all rest frequencies off. So summarized, a Filter is a function, which changes the audio signal characteristics. In the following

[2] Response to different frequencies by the audio equipment

subchapters we define High- and Low-pass Filters for equalization of sound[3], noise reduction and volume normalization.

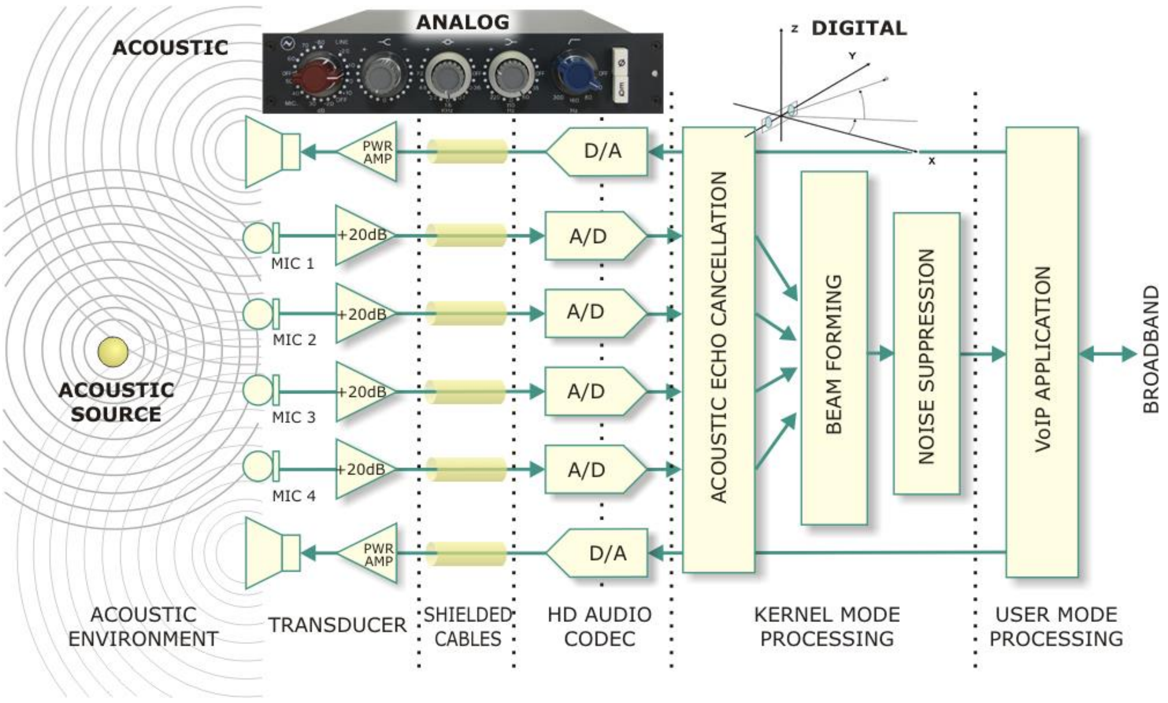


Figure 2.9: *Illustration of multiple Filters applied often by various hardware components (listed as analog) and software components (listed as digital) before some actual application like VoIP[4] receives the sound. Although such setup might seem advanced, one should consider modern audio systems, which often include multiple audio inputs and outputs, which opens the door to intelligent usage of their combined functionality for different purposes.*

One should note that often different hardware apply Filters as well as some operating systems (see Figure 2.9), before actual applications, like our framework, receives the signal. Hardware Filters often include noise or echo cancelation using supplement microphones to subtract the environment noise from the source, as well as automatic equalization of the frequency response for a given setup. Digital Filters often perform similar tasks to the Analog Filters, often considering weaker hardware setup, where various tasks like beam forming[5] might mean expensive hardware. Often various software apps might turn on different features of those, which can change recognition over time. So one should recognize such changes and re-extract features.

[3] Process of adjusting the balance between different frequencies of a sound signal
[4] Technology to deliver voice communications and multimedia sessions over Internet Protocol networks
[5] Technique used in microphone arrays for directional signal reception, which takes into account the position of the sound sources and amplifies them, while attenuating signals from other directions. This eventually leads to accurate feature extraction for the case that one tries to recognize the information of the main sources ignoring the rest of the environment, which is not the case of this thesis, since we consider secondary sources as important too.

High- and Low-Pass Filters

Filters for audio equalization are widely used in all disciplines of audio recognition. Most used are High- and Low-pass Filters to cut off (or silent) frequencies below (Low-pass) or respectively beyond (High-pass) given frequency (see Figure 2.10). This is a useful technique, if the recognizer uses only part of the signal spectrum for its recognition. E.g. if there is a speaker recognizer, which considers only the human speech pitch, which is around 80-300 Hz, one should consider cutting all irrelevant frequencies, to ease the extraction and recognition process. Further important usage of low-pass filters is to avoid aliasing[6] when downsampling a signal.

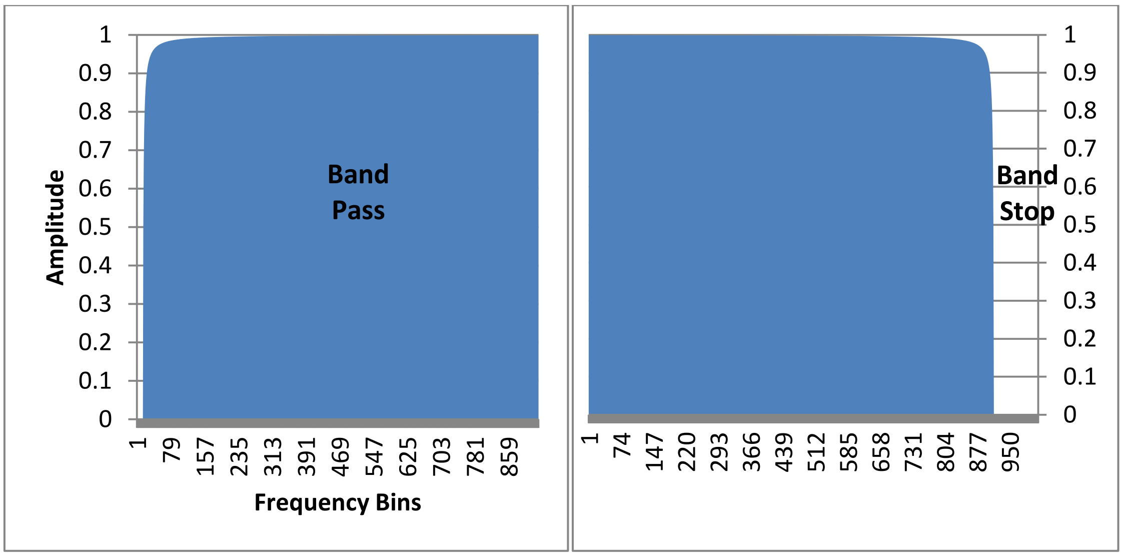


Figure 2.10: *Illustration of first order Filters: High-pass (left), which cuts the frequencies beyond some frequency and Low-pass (right), which cuts frequencies below given frequency. The blue marked area with amplitudes passed by the Filter, while the white marked, doesn't. A Band-pass Filter is a combination of both Filters, while a Band-stop filter is the inverse of the later*

One should note that the frequency cut doesn't happen immediately. This is clearly illustrated by the so called transition band, which is the range of frequencies between the pass band and the stop band of the signal (visually "where it turns the corner") and represents important characteristic of the particular Filter, which we will later use to smooth the signal deviations in its lower frequencies.

A combination of applying both High- and Low-pass Filters is called Band-pass Filter. In general there are more types of filters, which are applied for different purposes, but find seldom application in sound pattern recognition field.

[6] Distortion of the signal, which makes it different from the original one after some operation like downsampling. Example of this is when we need to lower the frequency from 20 kHz to 10 kHz we first need to cut off all frequencies beyond 5 kHz from the source signal, because they cannot be realized in the target signal, and thus result in distortions after downsampling.

Noise Reduction

Techniques for reducing the noise from the audio signal are called Noise Reduction techniques. Such techniques are of particular interest in the case where one has one important source like a speaker in a noisy environment. Reducing the noise from the environment, without affecting the speaker`s signal would eventually mean better signal for recognition. So despite for some cases Noise Reduction can pay off with increasing the recognition rate, for other cases it might worsen it, because some important sounds might be “noise like”. For example in a home environment a vacuum cleaner sound is considered as noise in the context of Speech Recognition, which is not feasible in the context of this thesis. However, some processing steps like those of next section are not that precise due to their nature, so removing signal noise in an equally distributed manner is available in this framework. Further techniques for better Noise Cancellation include the usage of multiple microphones, which enables subtracting noise signals from the important ones. However, in our setup we would like to use single microphone to cover the most general case audio input, such techniques are not of particular interest in this work.

Volume Normalization

When same source performs different sounds at different locations it results in different overall amplitude in the recordings by a static microphone. This translates in difference in all further processing steps. A way to overcome this problem is to normalize all sound input buffers to some amplitude. However, doing so might affect further processing steps in negative way too. Since in our setup we assume that most of the activities occur in specific places, thus having similar loudness related characteristics, we will avoid Volume Normalization.

2.2.3. Sound Spectrum Representation

Sound Spectrum is an important derivation of the sound, which provides detailed information about the nature of the sound wave, in particular its frequency domain. It is usually via Short-time Fourier Transform in combination with Window Function, both described in this section. It transforms small signal frames from the Time / Power domain into Power / Frequency domain. Mel Frequency Cepstrum Coefficients represent another popular transformation obtaining vectors of features, which are derived from the Spectrum and are often used in related works despite the increase of the computational complexity for their extraction.

Short Fast Fourier Transformation

The Fourier transform is a way to represent a sampled signal via mixture of sinusoid waves, called bins. In Sound Processing the Fourier transformation is used to convert the raw sampled signal from its original Time / Intensity representation into the Frequency / Intensity domain called Spectrum (see Figure 2.11).

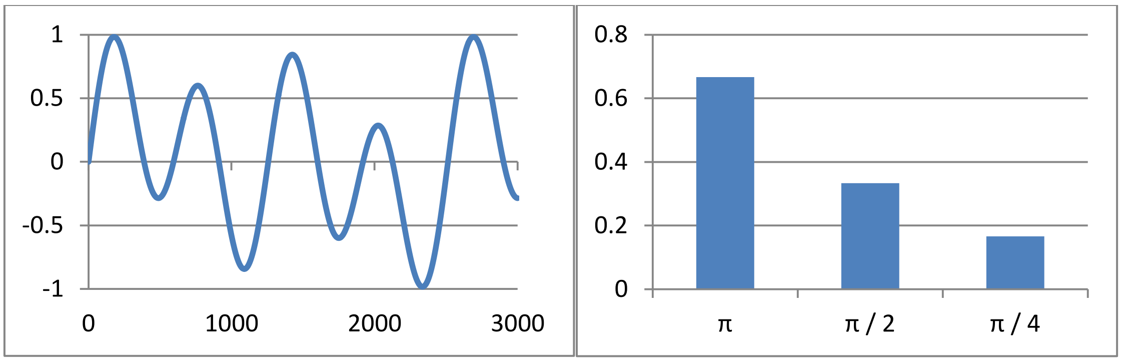


Figure 2.11: *Illustration of a complex wave (left) consisting of three waves of periods π, π/2 and π/4 in its Time (ms) / Intensity representation, and its Frequency / Intensity representation (right).*

As we already mentioned most signals we encounter in practice, such as sounds in home environment, are changing over time and have no periodic nature. However, if we take a closer look at these signals by cutting them into frames consisting of several milliseconds, we can see that for those time intervals, they appear to have some pattern or look as if they are periodic. This means that for those short excerpts of the signal we can assume the audio signal being a periodic signal, which could be represented by the sum of sin waves via Fourier transform, which for this case is called short-time (see Figure 2.12). The mathematical definition for this transformation is shown in the definition below.

$$STFT\{x(t)\} = \int_{-\infty}^{\infty} x(\tau)w(t-\tau)e^{-i2\pi f\tau}d\tau$$

Where w is a Window Function, such as those discussed in the next section, $x(\tau)$ is the signal at time τ of total time length t, which is to be transformed. It is important to mention that the discrete version of this formula is by replacing the integral with the sum over the discrete values in a buffer, also in the above definition by replacing signal with buffer and τ with buffer index. This means that choosing both, the samplling rate and the buffer size, plays an important role for the precision and the outcome of the Fourier transformation.

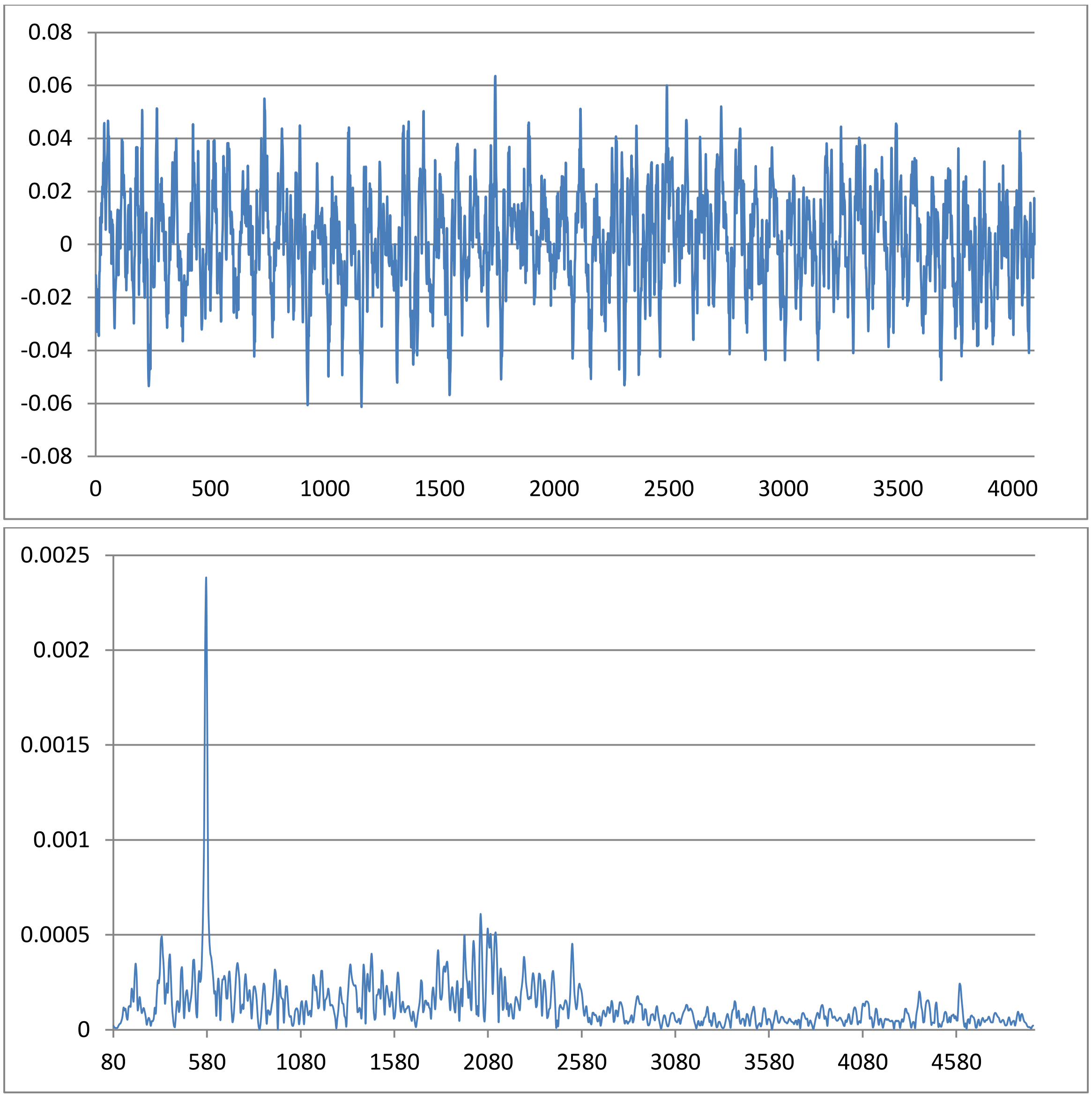


Figure 2.12: *Plots of waveform (above) with buffer length of 4096 values (X-axis) recorded with sampling rate of 44100 Hz (meaning time duration of 10.76 ms), and its short-time Fourier transform (below) in frequencies between 80 and 5000 Hz (X-axis).*

Short-time Fourier Transformation is the most commonly used Fourier Transformation and is obtained via the algorithm called Fast Fourier Transformation (Cooley & Tukey, 1965), which we discuss in the implementation part of this thesis (see Section 5.4.1.). However, there are more similar transformations in this domain. Another recent and a popular transformation are the Wavelets (most used in image processing), which are still not used for this activity recognition based on sounds, but had shown promising results in the speech and Music Recognition fields (Kronland-Martinet, Morlet, & Grossmann, 1987) (Tzanetakis, Essl, & Cook, 2001).

Window Functions

The Short-time Fourier Transformation described in last section assumes that the input signal is periodic, but as we have seen, a real record isn't such, despite our assumption. This leads to so called leakage, which causes erroneous intensities near some peak in the spectrum (see Figure 2.13). This is mostly due to the nature of buffering, which takes fragments of sounds at particular time intervals, without sensing their completeness (which exists only for periodic signals).

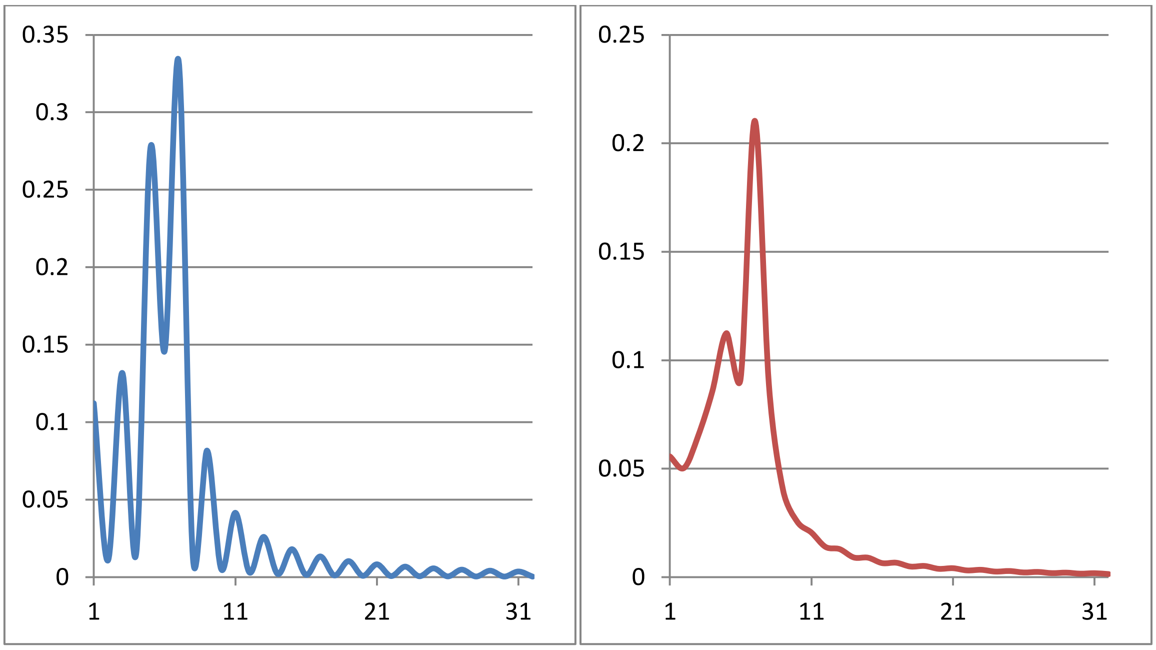


Figure 2.13: *Plots of first 32 bins of a spectrum of same periodic function – one with leakage (left) and other with smaller leakage (right) due to usage of Hamming window.*

To overcome this problem one uses Window Functions in order to smooth the buffer at its ends. By doing so, one forces the completion of all significant periods in a signal. In this thesis we apply three of the most commonly used Window Functions for sound signals – Hamming Window (Def. 2), Hann Window (Def. 3) and Blackmann-Harris Window (Def. 4).

Def 2: $HammingWindow(s) = 0.54 - 0.46\cos\left(\frac{2\pi s}{BufferSize}\right)$

Def 3: $HannWindow(s) = 0.5\left(1 - \cos\left(\frac{2\pi s}{BufferSize}\right)\right)$

Def 4: $BlackmannHarrisWindow(s) = 0.35875 - 0.48829\cos\left(\frac{2\pi s}{BufferSize}\right) +$
$0.14128\cos\left(\frac{4\pi s}{BufferSize}\right) - 0.01168\cos\left(\frac{6\pi s}{BufferSize}\right)$

Where s is the current sample from a buffer of size $BufferSize$.

A sample plots with the Window Functions can be seen at Figure 2.14 together with an example of how a simple signal buffer of 512 samples is windowed.

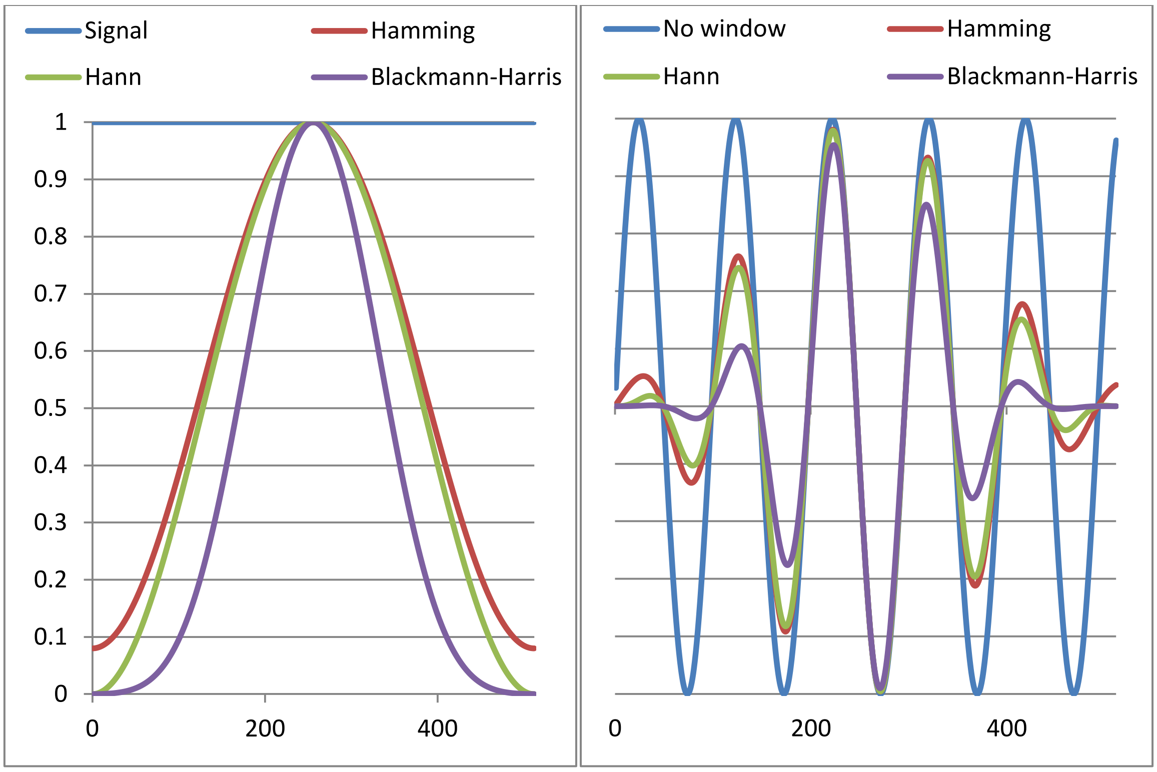


Figure 2.14: *Plots of Hamming, Hann and Blackmann-Harris Window Functions with the original signal. At left we can see constant signal with amplitude 1 and at the right se see signal with period π/2, both sampled at 10 ms.*

Mel Frequency Cepstrum Coefficients

Mel Frequency Ceptrum Coefficients (short MFCCs) represent also a popular sound transformation mainly used in the speech and speaker recognition fields. Their purpose is to extract information from the whole spectrum by mapping it from its original multidimensional domain into usually 12, 24 or 48 dimensional domain.

They are obtained after taking a FFT of the signal as described in the previous section. Then follows mapping of the obtained spectrum powers into a Mel Scale[7] using triangular functions. Then we take the log of each of the Mel frequencies powers and take their Discrete Cosine Transform[8]. The resulted amplitudes in the newly transformed spectrum (also called cepstrum) are the Mel Frequency Ceptrum Coefficients (see the three main steps in Figure 2.15).

[7] Mapping the recorded frequencies to Mel scale frequencies in parallel to the human hearing perception of pitch converting $f\ Hertz$ to $m\ mel$ with the formula $\mathrm{m} = 2595 \log_{10}\left(1 + \frac{f}{700}\right)$. Similar logarithmic representations of the frequencies are often used to simulate the human perception of tones in order to simulate the humans art of recognition.

[8] Used to break sounds into different frequencies, or sum of cosine functions, to approximate them

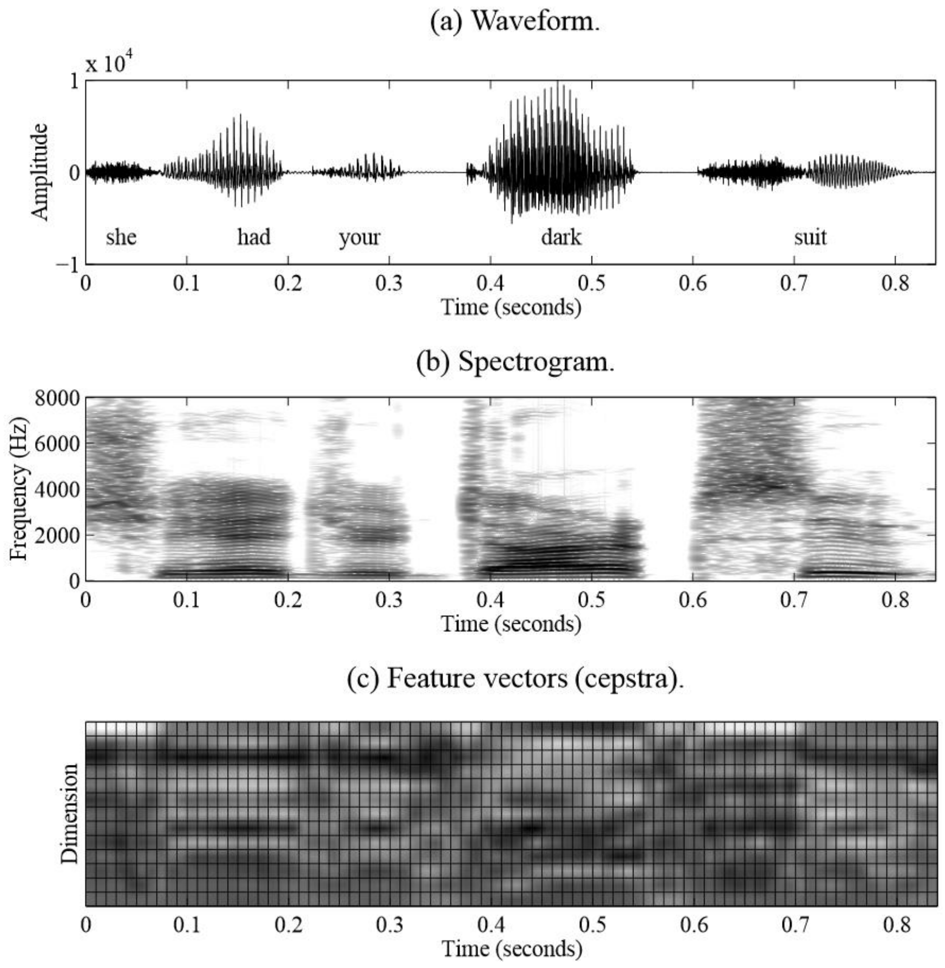


Figure 2.15: *Derivation steps of the Mel frequency cepstrum* (Yuk, 1999). *Above we can see the waveform input of the sound, in the middle its spectrogram*[9] *and down below we can see the resulted Mel Frequency Cepstral Features*

Similar derivations to the Mel Frequency Cepstral Coefficients are the Linear Frequency Cepstral Coefficients and the Antimel Frequency Cepstral Cofficients, which differ essentially by their filter banks and thus produce different recognition results. With those three one can obtain different precision at different frequencies. E.g. with same number of filter banks one would have higher precision at the lower frequencies and lower precision in high frequencies when using MFCCs, while the opposite is true for the AFCCs. LFCCs would have same balanced precision over all frequencies. One can make the LFCCs to have same precision like MFCCs in lower frequencies by choosing more filter banks to represent the cepstrum, which is of course more computationally exhaustive both for processing and later recognition.

There are lots of discussions, which cepstral coefficient features could be best used for speech and speaker recognition (Lei & Lopez-Gonzalo, 2009), which led research in

[9] Three-dimensional visual representation of the spectrum of a sound and its evolution over time. In the current example with displaying the amplitude of the different frequencies using the gray scale

Sound-based Activity Recognition to go in two separate directions – one using the LFCCs (Stäger, 2006), the other using MFCCs (Temko, Malkin, Zieger, Macho, & Nadeu, 2006), while others intend using a combination of both (Vuegen, Karsmakers, & Vanrumste, 2013) for their recognition. However one should note that due to the nature of the features presented in this section, they represent large vectors, which imply slower classification, while some research propose selecting different sets of features for different tasks to obtain better recognition (Zhuang, Zhou, Huang, & Hasegawa-Johnson, 2008) (Karbasi, Ahadi, & Bahmanian, 2011).

2.2.4. Sound Features

As we have seen in the previous sections, the sound curve is a complex function, which one can transform in different ways in order to obtain the form he would like. In our case we would like to concentrate on different aspects of the sound and highlight them for the usage as recognition features. In the following we describe features from the raw sound signal representation (Time / Amplitude), spectral representation (Power / Frequency) and the temporal representation of the spectrum (Time / Frequency) (see Figure 2.16). One should note that those representations are sorted in order of their complexity and computational power starting from the less demanding.

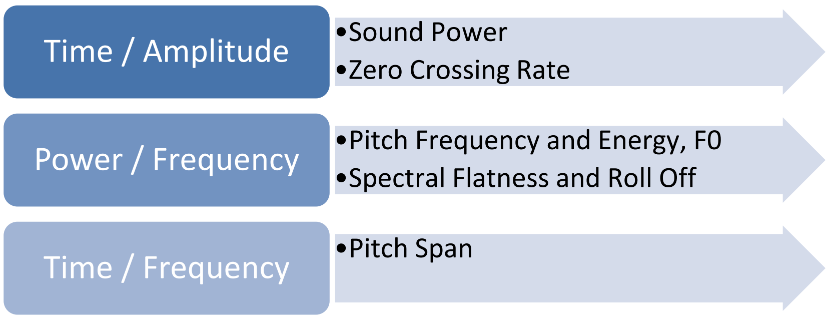


Figure 2.16: *Illustration of the different features sorted in three categories according to their derivation method*

Features from Time/Amplitude Representation of the Sound

Features from the raw representation of the sound, which is in Time / Amplitude domain, are usually not much telling features. However they enjoy literally immediate processing time, which makes them very handy. Those include sound power and zero crossing rate, which are presented in following subsections with examples of their meaning and application.

Sound Power

One of the most obvious sound features is the power. There are different metrics of measuring the loudness of the sound at certain point of time or for a certain period of time. We discriminate between power of a current sample, which is the digital value of this sample, and average power over period of time, by taking the average of the local maximum and minimums of the sound function during this period.

Although the power might seem a very uncertain feature, since there are a lot of different tasks that have similar loudness, one might think of the E-Energy scenario discussed in the introduction, where there might be a problem with deciding between two electric consumption models, which have very similar parameters (see Figure 2.17). In such a problem a single loudness criteria might be enough to recognize the correct device.

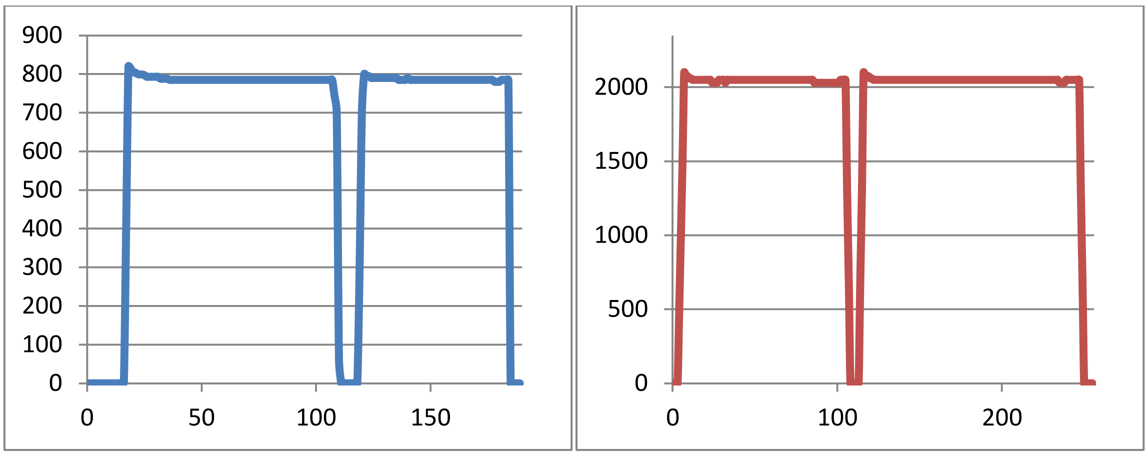


Figure 2.17: *Plots of the energy consumption of toaster (left) and kettle (right). One can clearly see the similarity in the patterns of both devices, despite the actual difference in the exact power consumption (Y-axis) and runtime (X-axis), both factors dependent of the heating elements of both devices and the chosen operating function.*

Zero Crossing Rate

The number of times the signal crosses the zero point for a certain period of time is called Zero Cross Rate. In a simple periodic function, the Zero Cross Rate is also the signal frequency by the factor of two. For example the function at Figure 2.9 has 9 zero crossings for 3 seconds meaning zero crossing rate of 3 crosses per second. The practical meaning of zero crossing rate is that if it is relatively high, it usually means a noisy record.

Features from Amplitude/Frequency Transformation of the Sound

In this section we discuss the features from the spectrum of the sound, by studying its concrete peaks and its overall form and characteristics. One should note that not all sounds have the presented features, while in other cases they are not distinguishable.

Pitch, First Formant and Pitch Energy

The pitch is also called Fundamental Frequency (short F0) and represents the lowest frequency of a sound wave. It can be measured by looking at the first local maximum in the spectrum. In Figure 2.9 we can see that since the lowest frequency of the sound wave is of period of π, it is also the fundamental frequency of the signal, while in the vacuum cleaner example in Figure 2.10 the pitch is at 576 Hz.

The first formant (short F1) is the second peak after the pitch in the spectrum. In Figure 2.9 it is of period $\pi/2$, while in the vacuum cleaner example the first formant is not very clear, but using our definition it is at 2040 Hz.

Further information about the pitch can be derived by looking at the pitch's energy. This is the sum of the energies of the spectrum bins near the pitch. In Figure 2.9 there is no energy near the pitch, so the pitch's energy is only contained in the pitch, while in the vacuum cleaner example we can see that the bins around the pitch contain energy as well.

Spectral Roll Off

The point where the spectral function falls turns down is called Spectral Roll off, and provides important information about the main energy concentration over the frequencies. It is measured by looking at the frequency point, where the spectrum contains a significant part of its energy, like 95% of it. In the vacuum cleaner example it is around the last significant peak at 4613 Hz.

Spectral Flatness

The meaning of spectral flatness is that if it approaches 0, then the signal consists of pure tones, while approaching 1 would mean equal distribution over all bands of the spectrum, similar to noise. In Figure 2.9 we anticipate three pure tones, which means spectral flatness of 0, while the vacuum cleaner example from Figure 2.11 has spectral flatness of 0.6696.

Temporal Features from the Time/Frequency Sound Domain

In this section we mention the single used temporal feature in this thesis – the pitch span over time. One should note that most of the mentioned features in the last section can be considered as temporal too, because they are often averaged for a period of time in cases where the recognition window is larger than the buffering window.

Pitch Span over Time

The difference between the maximal and minimal value of the pitch over time is called pitch span. It is a straightforward measure for the variation of the pitch, which is a good feature to distinguish between monotone and melody-like sounds. One can clearly study this feature using spectrogram like the one showed in Figure 2.15.

2.3. Machine Learning

In Ambient Intelligence perspective Machine Learning can be used to automatize the process of extracting knowledge about a given environment from measured data with the goal to understand better the underlying processes. In this chapter we will introduce the notion of supervised learning methodology (Section 2.3.1.) used in this thesis and one of the most popular classification methods called Nearest Neighbor. We will discuss then another popular family of recognition techniques called Bayes classifiers (Section

2.3.3.) and at the end we will discuss the terms of feature selection (Section 2.3.4.) and normalization (Section 2.3.5.), which both play an important role when working with bigger sets of features of different types, although in a slightly different way.

2.3.1. Supervised and Semi-Supervised Learning

The task of inferring a recognition result from labelled training data is called Supervised Learning and consists of two steps. In the first step, the Training Phase, the classifier gathers data and labels it with the help of supplementary input. In our case the user will label his performed activities and the respective devices to perform them. In the second step, the Recognition Phase, the classifier tries without external help to find the correct label, which in our case is a device recognition attempt. The two steps of supervised learning are repeating over time, which means the user can still provide feedback in order to improve the knowledge of the classifier.

Cases where the recognizer is able to additionally provide correct recognition of untrained data are called Semi-Supervised Learning. In our study such cases include the Machine Learning element training itself by automatic unsupervised mixing of records for their combined recognition.

2.3.2. Nearest Neighbor Methods

In general, there is no established single Machine Learning algorithm or methodology in the field of activity recognition (Aztiria, Izaguirre, & Augusto, 2010). One of the most established family of methods for finding correct class of given set of features are the Nearest Neighbor methods. One of their most straight forward implementations consists of classifying an unknown object by the majority vote of its neighbors found using some distance metric in feature space like Euclidean distance[10]. One should consider that if $k = 1$, then the object is simply assigned to the class of that single nearest neighbor. By using this property, in order to improve classification runtime in Section 5.5.4., we reduce the Nearest Neighbor algorithm to one nearest neighbor search by averaging the trained data through the classes or by selecting a single reference value for recognition. We then implement further optimization of reducing the dimension of feature space, by prioritizing the different features.

[10] Distance $d(r, s)$ between each reference feature r_i and the sampled feature s_i measured by the Pythagorean formula using following equation $d(r, s) = \sqrt{\sum(r_i - s_i)^2}$

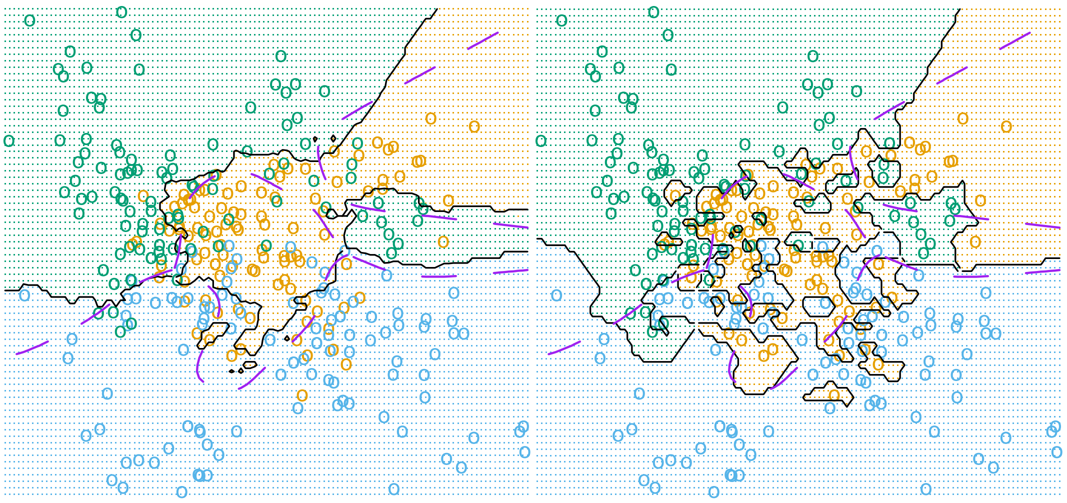


Figure 2.18: *15 nearest neighbors and 1 nearest neighbor algorithm distributions by* (Hastie, Tibshirani, & Friedman, 2009). *Purple line represents the optimal Bayes decision boundary.*

2.3.3. Bayes Classification

Bayes classifiers are a family of classifiers that are based on applying Bayes' theorem for conditional probabilities[11]. A classifier, which applies Bayes' theorem with strong independence[12] assumptions, is called naïve Bayes classifier. Such a classifier relies on priori and posteriori knowledge about the data in order to assign class probabilities for some test data. For example if one brushes his teeth every day, but cleans his room with vacuum cleaner once a week, and does only those two activities. So after three weeks we would have 21 occurrences of Tooth Brushing against three occurrences of Vacuum Cleaning. In Figure 2.19 we can see a plot of this knowledge in terms of two features – zero crossing rate and average loudness. So we have a prior probability of Tooth Brushing activity of 7/8, while for Vacuum Cleaning it would be 1/8. A simple way of computing a posterior probability is if we draw a circle around the unknown occurrence to compute in straightforward way its' likelihood[13], by counting the members of each class inside the circle. We can see that in the chosen circle in the likelihood of the unknown activity there are two occurrences of Vacuum Cleaning and one of Tooth Brushing meaning a posterior probability for Tooth Brushing of 1/21, while for Vacuum Cleaning it is 2/3. So by applying the Bayes rule we assign probabilities for both classes to the unknown sample:

$$Probability\ of\ X\ being\ Cleaning = \frac{1}{8} * \frac{2}{3} = \frac{1}{12}$$

$$Probability\ of\ X\ being\ Toothbrushing = \frac{7}{8} * \frac{1}{21} = \frac{1}{24}$$

[11] Probability of some class to be the correct result based on given set of features
[12] Assumption that any feature is independent from each combination of other features
[13] Function of how likely is the occurrence of some class, which is weaker than its probability

Thus, by having the probability of the unknown sample to be vacuum cleaning twice larger compared to its probability to be tooth brushing, according to our rules we assign it a label of Vacuum Cleaning.

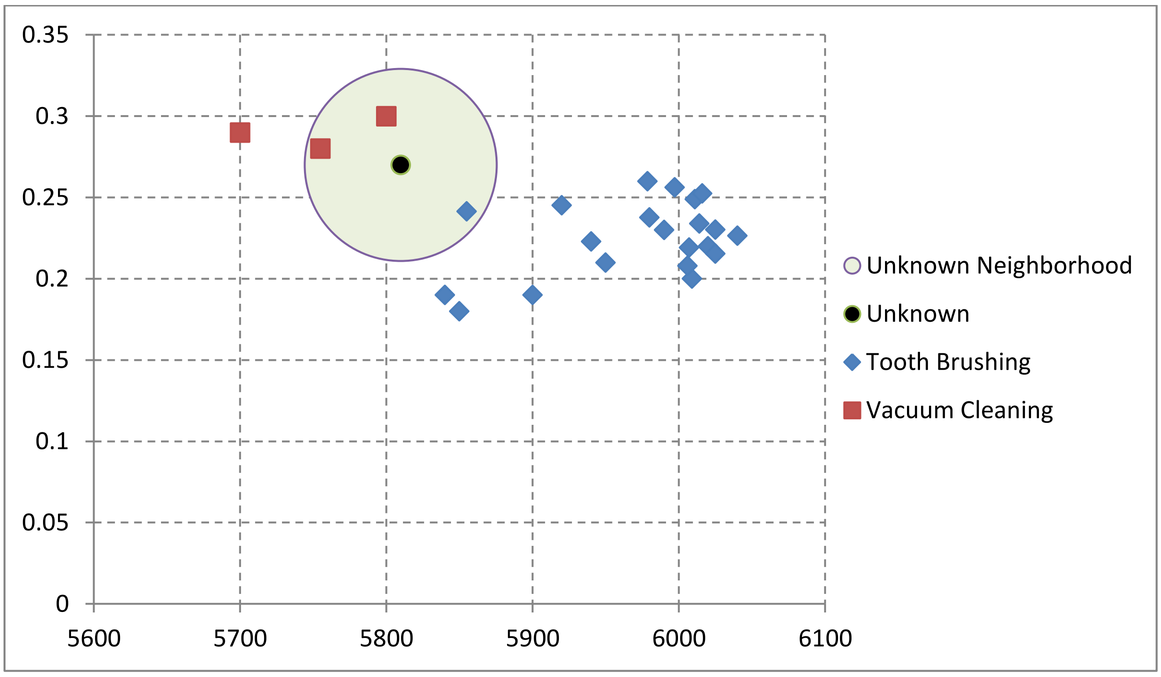


Figure 2.19: *Naïve Bayes classifier example showing two-class problem (Vacuum Cleaning and Tooth Brushing) with two dimensional feature space (Zero Crossing Rate on X-axis, and Average Loudness on the Y-axis). The classifier is attempting to guess the appropriate label of unknown event by knowing its features.*

There are numerous possible improvements of this "naïve" approach. One improvement is about how to look at the distribution of the trained data for the different classes to build up a so called distribution models, which will eventually be used to compute the likelihood. In our example we assume uniform distribution[14], and therefore add the different probabilities in the likelihood computation. However, the most popular distribution model is the Gaussian mixture model, where the feature vectors associated with each class are distributed according to a Gaussian distribution[15]. In this thesis we use a so called Bayes Point Machine Linear Classifier[16], which approximates the optimal Bayes decision[17] by the center of mass of version space[18] (Herbrich, Graepel, & Campbell, 2001).

[14] Distribution, where all types of classes are equally likely to be observed
[15] Family of distributions, which measure the probability that any real valued observation will fall in the interval of two real numbers. For multidimensional cases, like ours with multiple features, their complexity increases rapidly
[16] Linear Classifier makes a classification decision based linear combination of the input feature vectors
[17] A case, where the probability model underlying the classes is known perfectly. The Bayes optimal decision boundary is drawn with purple line in Figure 2.18.
[18] Set of all hypotheses that have not been eliminated as a result of being in conflict with observed data

2.3.4. Feature Selection

In the last few sections we explained how to make good predictions, even for unseen cases, with successful recognition being our main goal. However we still have not discussed one of the main factors for recognition – the features. Namely, from a Machine Learning perspective, how to find out which are the relevant features in order to make good predictions?

The answer of this question is hard and overall there is a lot of research on this topic with many different proposed solutions (Guyon & Elisseeff, 2003) (Hastie, Tibshirani, & Friedman, 2009), some of which theoretically optimal, but unpractical to implement (Koller & Sahami, 1996). So here we will first discuss the benefits of feature selection for certain problems and then will briefly explain couple of ways to perform it in our framework, while the exact implementation will follow later in Section 5.5.2.

The first benefit, which comes in mind after discussing a couple of Machine Learning classification approaches, is that selecting the relevant features for recognition and removing the irrelevant makes for faster result search, thus optimizing the runtime performance.

Further issues with larger a set of features arise, when some of those features are dependent variables[19], which steers the recognizer in wrong direction by introducing the possibility of replicated erroneous information overwhelming the correct information. Similar problems might occur when one tries to use same feature types from different sound representations, for example computing average loudness from the waveform representation and from the spectrum. So despite both variables not being exactly dependent, they still have very similar behavior.

Another problem occurs when some feature is very similar for large number of different classes, thus providing no meaningful information. In our case, many electrical devices had a large subset of similar cepstral coefficients, which was one of the reasons for the decision to leave their integration into the framework for its current version.

The various listed redundant or irrelevant features above give us first ideas of how to select features automatically. One straightforward way is to measure the relative distance between the feature vectors of the different classes and remove those features, which doesn't change much through the classes. Another way is to look at how "stable" the features are for the recorded samples of same class, e.g. their variation.

An exhaustive feature selection for a given setup environment can be done by running a test with all feature combinations to unveil what would be the best feature combination for the particular test, and depending on how much the test approximates real-life conditions, we can make statements on how good the computed feature subset is.

[19] Variables are called dependent if one can find a functional mapping between them

2.3.5. Feature Normalization

Since there are many components in a big recognition system one needs to normalize their inputs into a common metric at each abstraction level (see Figure 2.20). The lowest normalization level is for the different features. Sometimes they have different metrics, which need to be normalized in a common feature vector. An example of different feature metrics is the zero cross rate and the pitch. The next normalization level is for the feature vectors produced by the different sensor instances. This is needed, because different sensors might have different characteristics, like different microphones having different sensitivity or frequency response. The next normalization step is at sensor level, where one needs further synthesis of the gathered feature vectors, which are not necessarily of the same metrics or dimension. This can occur when the extracted features is specific for the given sensor, despite being of the same type. The last normalization step is at component level, where the different sensor types are being normalized to merge in a single feature vector.

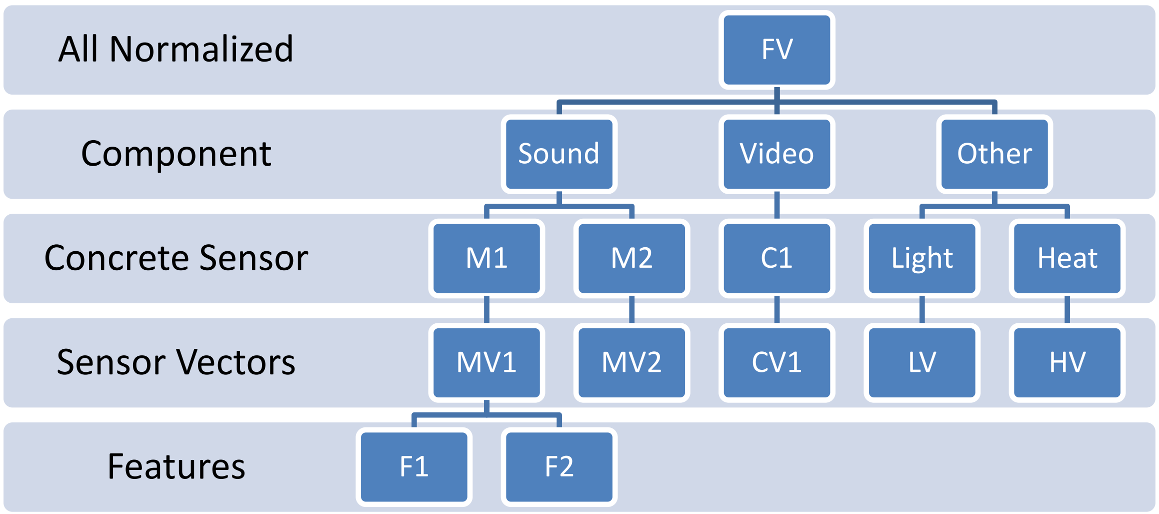


Figure 2.20: *Feature normalization at different levels in order to extract a feature vector from all components and the features from their sensors. In the example at first we have sound component with two microphones and their according feature vectors consisting of different features like those mentioned in 2.2.4. The second component is a video with one camera and its feature vector. The third component is for the remaining sensors, like light sensors all together with the power of the light in a common vector, or heat sensors and a common vector for the measured temperatures. Note that at each level there should be a normalization procedure.*

Chapter 3

Related Work

In the field of Signal Processing (Subchapter 3.1 and Figure 3.1), a lot of studies have been carried out in its branch of Sound Pattern Recognition. Their focus was mainly on building systems for Speech or Speaker Recognition (mostly non-commercial) (Subchapter 3.2) and Music Recognition (mostly commercial) (Subchapter 3.3). In general, one can separate those two types of related works from the general case of Activity Recognition using sound, which have a similar setup to this thesis (Subchapter 3.4).

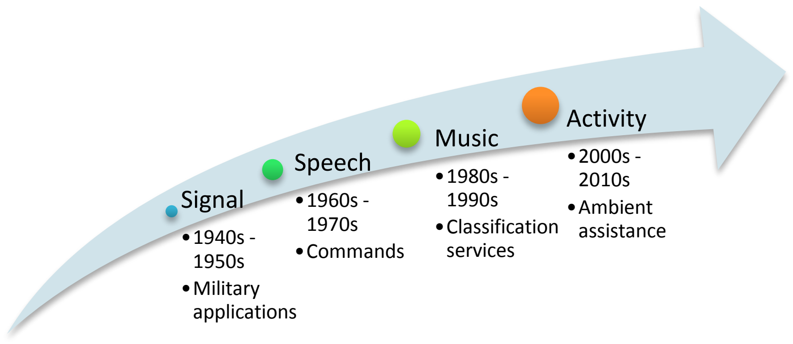


Figure 3.1: *Illustration of the creation of related disciplines over time with the reason of their establishment. Signal Processing for Recognition being the general case and the root of both Speech and Music Recognition, out of which originate all of the later used sound pattern recognition methodologies for the general case of Activity Recognition.*

In the following subchapters we will get familiar with the mentioned main branches of sound pattern recognition. We introduce their purpose, technology and achievements in their corresponding subfields. We make then parallel to the case of study of this thesis. At the end of each subchapter we will generalize what their recognition capability means from the perspective of general Activity Recognition. Finally, we provide an overview with a comparison of the introduced related works and we make a composition of their capabilities.

3.1. Signal Processing

The area of Digital Signal Processing, which is the root of Sound Signal Processing, exploded around the Second World War mainly with applications like radar Signal Processing for automatic detection of enemy or telecommunications encoding. In terms of activity recognition, using those early Signal Processing techniques we could determine, whether there is activity occurring or not (see Figure 3.2 for a three-wise illustration). The rebirth of Signal Processing came with the invention of digital computers around the 50s (Nebeker, 1998).

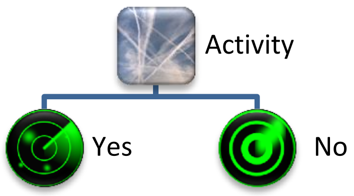


Figure 3.2: *Three-wise illustration of the automatic activity distinctiveness that could be done by Signal Processing from the early years*

Up to date Signal Processing has been a very wide scientific area covering whole areas itself like Image Processing, Sound Processing, Video Processing, and many more. What we can learn from this is that all subfields often have very similar processing algorithms despite having different purposes. So developments in those related subfields could often be translated to the whole area. Examples include the invention of wavelets and their immediate translation into different areas (Daubechies, 1992), or the development of fast Fourier algorithm (Cooley & Tukey, 1965). In general, all signal recognizers share the same workflow consisting of three main components: Signal Acquiring, Signal Processing, and Signal Recognition (see Figure 3.3).

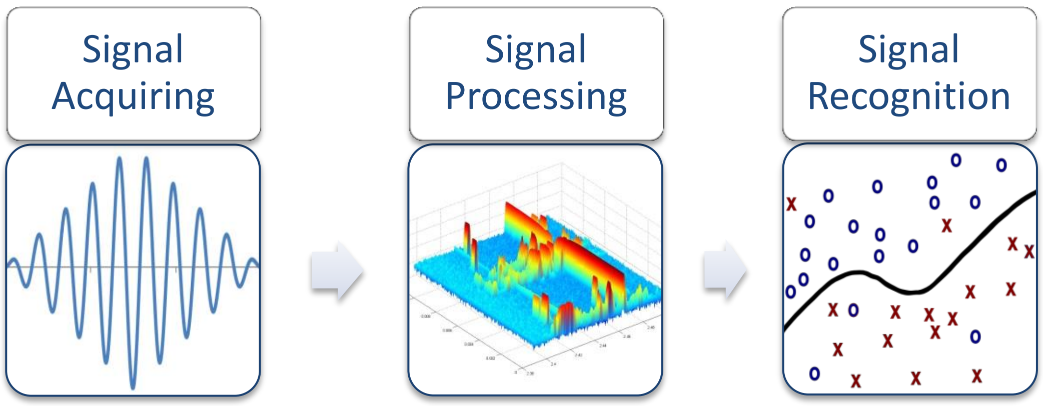


Figure 3.3: *The three main steps in signal recognition: signal acquiring, Signal Processing, and signal recognition, common in all signal recognition systems*

3.2. Speech Recognition

Despite speaking being a special kind of activity, its recognition development started a lot earlier than the general case of Activity Recognition or other Sound Pattern Recognition fields like Music Recognition. First studies for Speech Recognition started six decades ago with recognizing strings of digits with pauses in between (K. H. Davis, 1952). The next milestone was set five decades ago recognizing small set of words – IBM system 16 words (Dersch, 1962). Note the similarity between size of the word set and the approximate activities of daily living, which typically occur in home. Another similarity represents the pauses between commands. Typically activities are also separated by pauses, since one cannot immediately switch between brushing his teeth and cleaning his house.

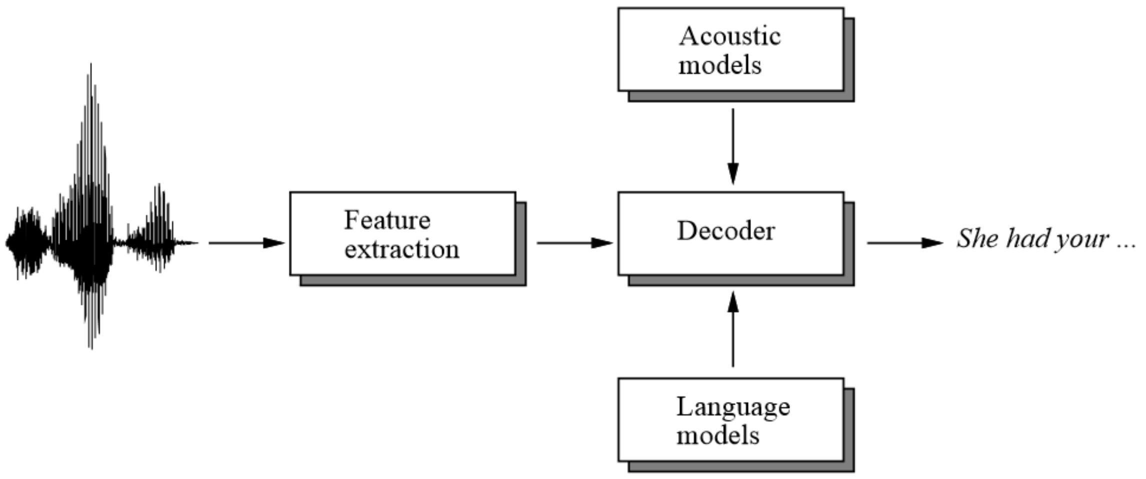


Figure 3.4: *Illustration of the main components of Speech Recognition system* (Yuk, 1999)

Later on there are numerical improvements including numerous acoustic feature representations, which try to emulate the human perception of speech and are coming closer in recognition accuracy to humans (Lippmann, 1997) (Scharenborg & Cooke, 2008). One of the most studied and accomplished recognition techniques are the MFCCs, which are performing for both speech (Young, Kershaw, Odell, Ollason, Valtchev, & Woodland, 2000) and speaker recognition tasks (Feld, 2011) (Beigi, 2011). In the face of Speech and Speaker Recognition we anticipate first accomplishments in direction activity recognition (see Figure 3.5). This is also of particular importance since most of the current research of activity recognition based on sounds has its origins from the Speech and Speaker Recognition fields and use MFCCs as main feature vector part, which means a lack of testing other techniques, which might eventually be performing even better. For the recognition however, most of the modern Speech and Speaker Recognition works apply advanced models like the Hidden Markov Models (Yuk, 1999) (Feld, 2011), which are still beyond the research state of our closely related works.

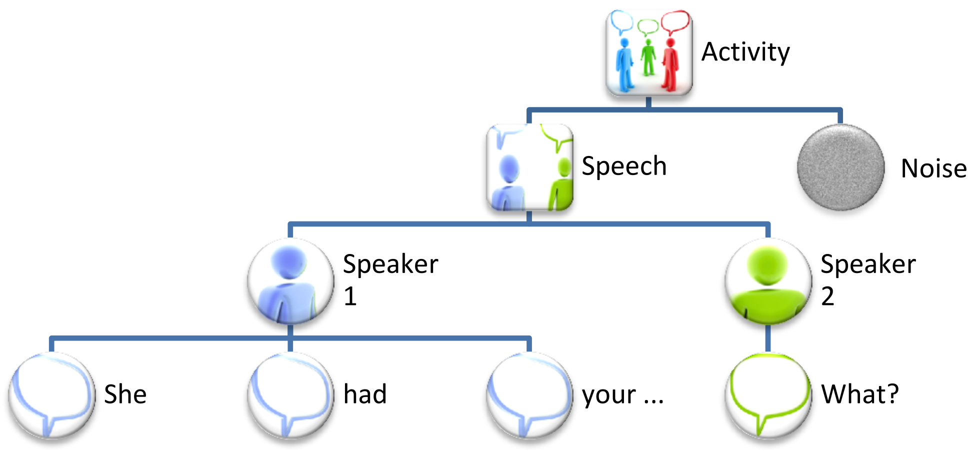


Figure 3.5: *Illustration of the Activity distinctiveness that could be done using a combination of speech and speaker recognizers like (Yuk, 1999) (Feld, 2011)*

3.3. Music Recognition

The beginning of Music Signal Processing could be set around the early 20s with the invention of a musical instrument called Theremin[20] (Glinsky, 2000). Later with the mentioned Signal Processing improvements around the 40s and 50s composers such as Karlheinz Stockhausen created music using signal generators or ring modulators. Then with the boom of general purpose digital computers in 70s and 80s came a new era of synthetizing and modifying sounds (Müller, Ellis, Klapuri, & Richard, 2011).

Despite the big progress in sound synthesis, the field of sound information retrieval started in the mid-70s (Moorer, 1975), while the first International Symposium on Music Information Retrieval was held in 2000[21]. First big applications include automatic sound description, like The CUIDADO Project (Content-based Unified Interfaces and Descriptors for Audio/music Databases available Online) (Vinet, Herrera, & Pachet, 2002). It produced the first entirely automatic chain for extracting and exploiting musical metadata for browsing music using large set of sound features (Peeters, 2004).

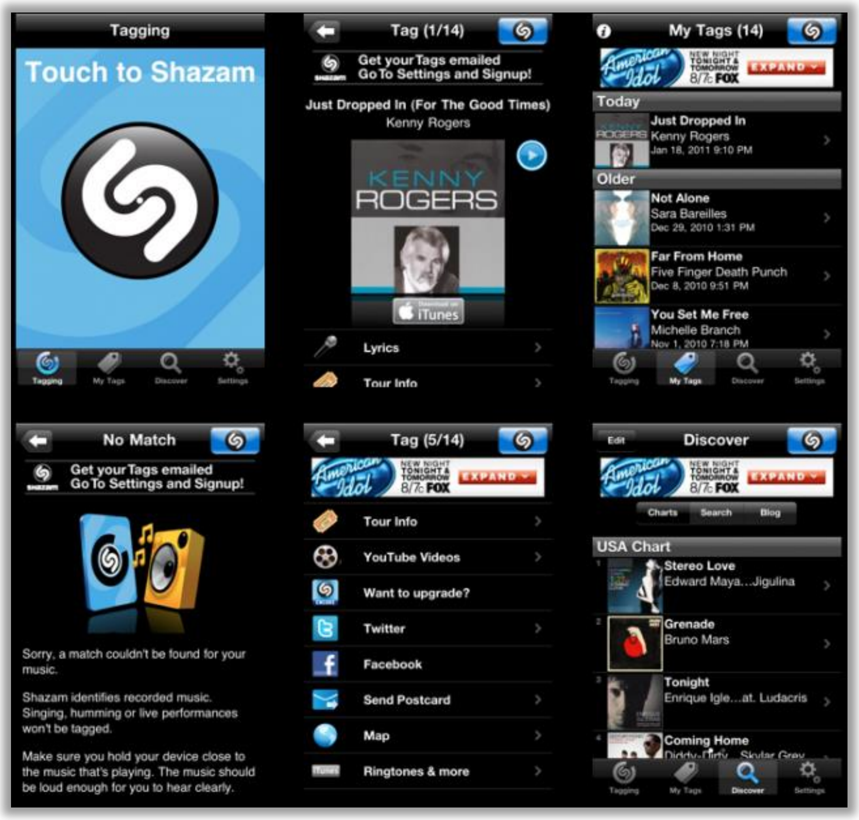
To date, depending on the task, there are numerous of established techniques for Music Recognition besides Mel Frequency Cepstral Coefficients, most notable of which are the Chroma-based audio features[22] (Fujishima, 1999). The most notable Music Recognition systems are Shazam (Wang A. L.-C., 2003) (commercial) and Echoprint (Ellis, Whitman, & Porter, 2011) (non-commercial).

[20] Theremin is the first electronic musical instrument and is played by the musician without physical contact. The capacitance of player's hands near antennae controls an oscillation, which is later transformed to sound.

[21] Online at http://www.ismir.net/

[22] Initially proposed for Chord recognition, are audio features, which closely correlate to the aspect of harmony

Here we will have a closer look at Shazam, because it realizes our idea of recognizer, which incorporates a client / server architecture. The client issues a recognition using a mobile application, which connects to a server for a result. Furthermore it provides different personalization and discovery features for already tagged music or a similar music search based on that (see Figure 3.6 for an overview of Shazams interface). Similar to that in our setup we will also rely on the user to tag his activities first in a training phase and provide random feedback about the recognition.



Sorry, a match couldn't be found for your music.

Shazam identifies recorded music. Singing, humming or live performances won't be tagged.

Make sure you hold your device close to the music that's playing. The music should be loud enough for you to hear clearly.

Figure 3.6: *Collage of screenshots*[23] *of Shazam illustrating different states of the phone application. Besides the offered recognition service one can see a high level of personalization*

Shazam works by taking 10 second recordings and analyzing their spectrum for intensity peaks. Then those peaks are grouped in a hash for a result lookup. We should note that such a straightforward search method can provide a successful search of over 11 million records with a constant time lookup. Other qualities of the method include noise robustness and room acoustics independence, both very important in our case. This leads to considerations about using similar methods to those proposed by the authors and attempting to modify them for our setup.

23 http://vlatte.net/2011/01/shazam-app-review/

Bottom line is that although the described systems are often commercially based and may seem distant relatives to our subject, they are also in the subject of Sound Pattern Recognition and they are dealing with databases, which contain millions of songs. Furthermore, some of the systems also aim to recognize radio and television shows as well, which increases their complexity. In terms of Activity Recognition there is already a high degree of activity details, which can be obtained using the presented recognizers, despite the small set of recognizable activities (see Figure 3.7).

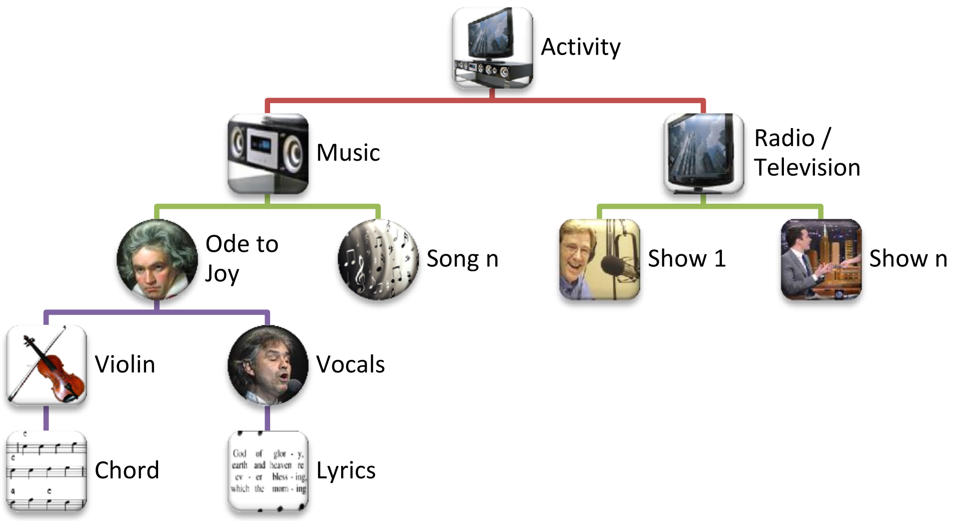


Figure 3.7: *Illustration of the Activity distinctiveness that could be done using a Music Recognition service like Shazam combined with instrument and chord recognition system. Note that one could make an additional tree level for clustering the music in terms of different genres or for similar clustering television and radio shows, which might be important for different context. Also we are distinguishing the activities of music listening, radio listening and television watching, although they might all come from same source (or device).*

An interesting fact is that there are no similar algorithms used for Sound-based Activity Recognition. This is quite unexplainable since one can think of an electrical device as a musical instrument with a specific timbre, which produces specific sound when being operated by a human, in parallel to the musical sounds, which are more dynamic.

3.4. Activity Recognition

First works of general activity recognition based on sounds started with (Wang, Wang, Huang, & Hsu, 2003), who tried out “Home environmental sound recognition based on MPEG-7 features” and (Stager, Lukowicz, & Troster, 2004) and their work of “Implementation and Evaluation of a Low-Power Sound-Based User Activity Recognition System”, which tests three sound features and searches for result using nearest neighbor algorithm. Later refined in Stägers PhD thesis “Low-Power Sound-Based User Activity Recognition” (Stäger, 2006), turning into a complete solution including hardware installation of a microphone with three accelerometers (on hand) and software written recognizer. What we can learn from this work is that even a small set of straight-forward features and search algorithms can obtain good recognition results for a small set of activities. Also it is important to mention that this collaboration of authors are the only ones concerned with the trade-off between power and accuracy later described in details after empirical improvements in (Stager, Lukowicz, & Troster, 2007). An optimized usage of the resources is an important consideration in a bigger setup and could be done automatically as well (Lombriser, Amft, Zappi, Benini, & Tröster, 2011).

The next important development in the field is "Embedded Implementation of Distress Situation Identification through Sound Analysis" (Istrate, Vacher, & Serignat, 2008), which is a complete solution for an elderly care system that produces alarm if something goes wrong. It includes an early distinguishment of the sounds between speech and other acoustic events before making a recognition attempt. If the recognition results in a situation where the user might be in danger it forwards an alarm with the signal to an operator and sends message to close relatives of the user (see Figure 3.8). As we can see this shows a complete example of Sound-based Recognition system for home environments in practice.

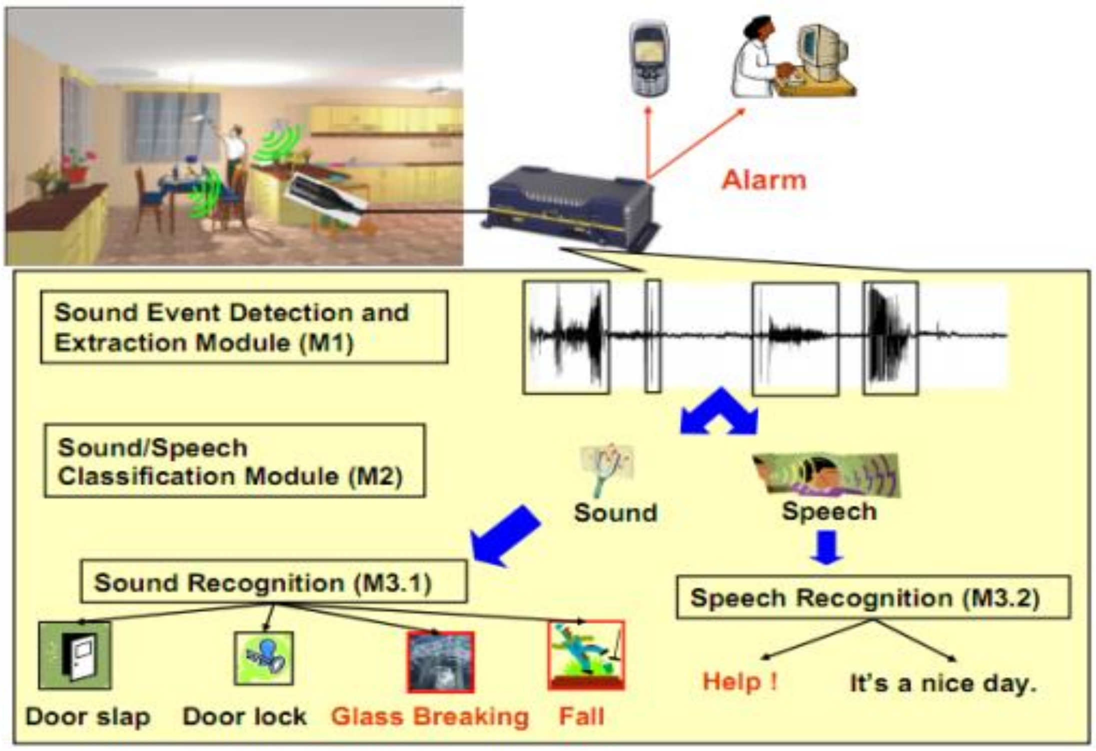

Alarm
Sound Event Detection and Extraction Module (M1)
Sound/Speech Classification Module (M2)
Sound
Speech
Sound Recognition (M3.1)
Speech Recognition (M3.2)
Door slap
Door lock
Glass Breaking
Fall
Help !
It's a nice day.

Figure 3.8: *Illustration of the system developed by* (Istrate, Vacher, & Serignat, 2008)

Most recent developments include "Sound Environment Analysis in Smart Home" (Sehili, et al., 2012), which is a work that installs a microphone to an existing recognition environment, consisting of three rooms (e.g. with installed different types of sensors in each room already) (see Figure 3.9). It is a good example of setup, since we naturally perform different activities in different rooms, thus having less activity types in each room. Also interesting is the high number of installed microphones and their combination with other sensors.

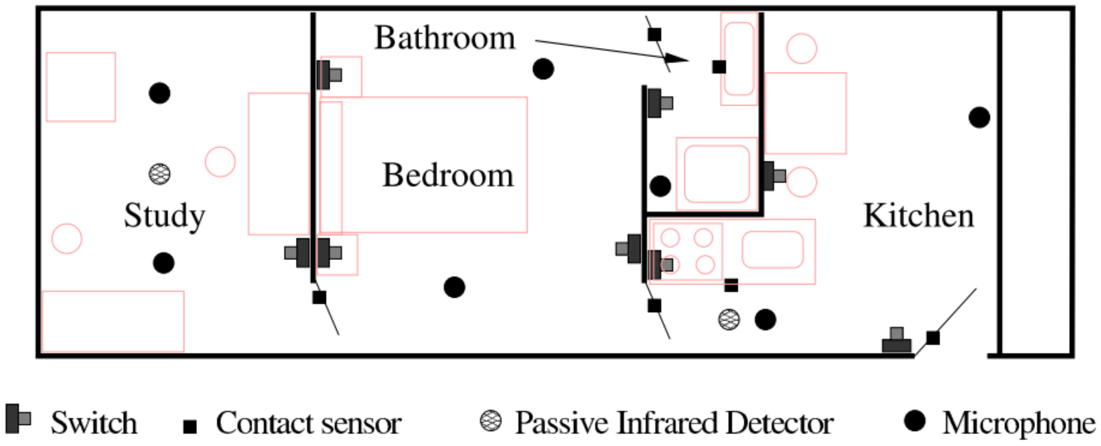


Figure 3.9: *Setup environment consisting of three rooms and multiple sensors of different types* (Sehili, et al., 2012)

Another approach worth mentioning is "Audio Classification Techniques in Home Environments for Elderly/Dependant People" (Lozano, Hernáez, Picón, Camarena, & Navas, 2010), which besides the conventional used techniques in the fields, studies multi-resolution analysis to include temporal information from the sound data. Also a couple of other studies use temporal information – (Wang, Lee, Wang, & Lin, 2008) and (Karbasi, Ahadi, & Bahmanian, 2011), who similar to this thesis identifies the problem of most Activity Recognition systems deriving their Sound Pattern Recognition techniques from the Speech Recognition field.

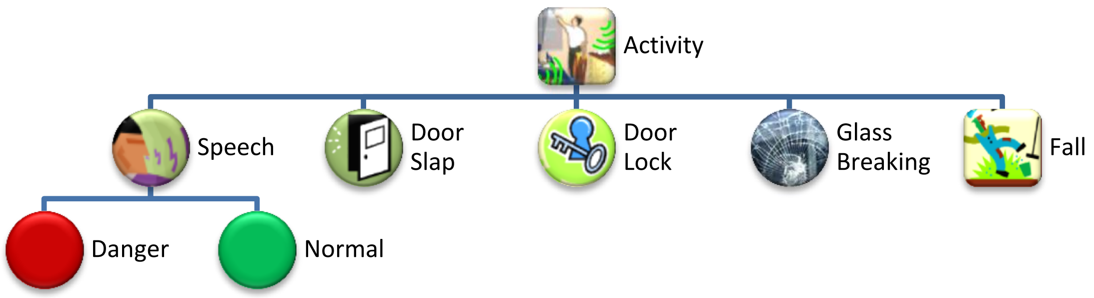


Figure 3.10: *Illustration of the Activity distinctiveness that could be done using a general activity recognizer like the one developed by* (Istrate, Vacher, & Serignat, 2008)

3.5. Device Recognition

Usually under device recognition one understands recognizing different devices according to their interface. E.g. a recognizer scans the environment using different communication protocols like Wi-Fi or Bluetooth and relies on the devices to support those protocols. However, not all devices had to support such communication protocols. Another recent research trend is to recognize electrical devices via a series of methods like using energy monitoring sockets, power analyzer (Hart, 1989) (Belley, Gaboury, Bouchard, & Bouzouane, 2013), and electromagnetic interference (Gupta, Reynolds, & Patel, 2010). The problem is that all of those devices should not only be electrical, but should be connected to the power supply. First of all not all of the devices are electrical, nor should they be. Examples include toilet flush or a tooth brush. Some toothbrushes are electrical but they rely on batteries for their function, which makes them “invisible” for the mentioned technologies in their acting time. A natural way of recognizing devices, according to the human sense would be by analyzing their sounds. This includes the case where someone is using them to perform some activities, as well as the case, where they perform some periodic miscellaneous tasks without being operated directly by humans.

It is also interesting that in the mentioned related works the authors apply the interference that a noise by some device would mean a human producing that noise, while operating with that device. As already mentioned here as well as in Section 2.1.1., this is not necessarily true. There is no system known to us, which defines a set of devices and aims to recognize their activities by monitoring the sounds which they produce.

3.6. Overview of Related Work

In this Subchapter we first summarize the different types of related works in terms of their functionality in Section 3.6.1. Then, in Section 3.6.2., we make a comparison of the activity recognition systems and their capabilities presented in Subchapter 3.4. Finally, in Section 3.6.3., we make first thoughts as to what kind of sound-based information a system, consisting of the related works, should be able to extract.

3.6.1. Comparison of Related Works in Terms of Functionality

In this section we will make an overview of the related works, which we have presented. We group them according to their appearance in this Chapter and show their main attributes in Table 3.1 in the context of a device recognition system, which we will design in the next Chapter.

The first relevant column is the number of entities over which the recognition system operates (Column 2). For the Speech Recognition case it is usually the size of the vocabulary, while for the Music Recognition case it is the number of songs, which the system can recognize. It is an important factor, because the less entities one aims to recognize, the less sophisticated methods one needs to implement, which results in higher processing speed, while retaining recognition accuracy.

For the feature choice (Column 3), most of the closely related works use primary Mel and Linear Frequency Cepstral Coefficients, which as stated is the primary feature source in Speech and Speaker Recognition too. Other often used features in combination with the previous include zero crossing rate, spectral roll off or frequency centroid[24]. For Music Recognition the melody plays an important role, thus in systems, like Shazam, peak intensities and their duration is the base feature, while other systems use bigger feature vectors for detailed recognition. In our work however, a slightly different set of features was chosen, which was found to play an important role for our recognition use case, explained in detail in the next couple of Chapters.

Some of the described Music Recognition services enjoy user input (Column 4), similar to ours, while other systems use static databases, where the user plays no active role in improving the recognizer via feedback.

Another characteristic of a recognition system is its adaptation capability (Column 5), which we consider as an important property of a system, which aims to make better recognitions over time. Under the adaptation it is meant the adaptation scale of the recognition rather than the user's preferences or his interface design personalization. Usually Speech Recognition systems have a high rate of adaptability for the purpose of tuning the recognizer to a specific person and his pronunciation in order to obtain a better recognition rate. Services like Shazam don't need an adaptation property, since they consider music songs in exact time and melody.

The last property of the recognition systems is whether they rely on Machine Learning algorithms for finding a result. For example some of the early Speech Recognition systems described in Section 3.2, use straight forward matching of the closest solution, while all modern Speech Recognition systems use sophisticated Machine Learning techniques. In an interesting contrast, Shazam uses straightforward hash table lookups over 11000000 records for a result.

Method	Entities Count	Features Count	User Input	Adaptability	ML
Speech	Up to 100000	~24	Depending	High	Yes
Music	~10000000	~3	Essential	Not needed	Depends
Activity	~10	~8	None	None	Yes

Table 3.1: *A table comparing the related works from subchapters 3.2 to 3.4*

[24] A measure defining middle point of a spectrum according to its amplitudes in different frequencies

3.6.2. Detailed Comparison of Activity Recognition Systems

In this section we will make a summarized comparison of the presented activity recognition systems (see Table 3.2) in the context of the intended capabilities, which our framework should possess as mentioned in the introduction.

AR System	Sound-based Only	Easy Installation	Optimization Friendly	Feature Selection	Mixing	Adaptability
Stäger et al.	Sometimes in combination with accelerometers	Setup relies on wearable sensors	Performed in both hard- and software	Studied features. No automatic selection	Not performed	Not performed
Istrate et al.	Separate sound event detection and extraction	Flexible setup	Complex techniques	No automatic selection	Not performed	Not performed
Sehili et al.	Multiple sensors in the full setup	Big setup	Complex techniques	No automatic selection	Not performed	Not performed

Table 3.2: *A table comparing the related works from subchapters 3.2 to 3.4*

As we can see on the comparison table, the most competing work has been done by Stäger et al. They have studied into detail different aspects of an activity recognition system, which we consider as important too, like making an optimization friendly installation with carefully selected feature sets. However their biggest difference to our intended framework is that their recognition setup relies on wearable sensors, which is an uncomfortable way of sensing information, as mentioned in Section 2.1.4. The second most competing work has been done by Istrate et al. They perform a large variety of Sound Processing techniques for their recognition, but in their tests they used data from multiple environments and trained the recognizer with 90% of it. In both points our work intends to do completely opposite. We first intend to make a personalized setup, and second to use less training data. Other interesting capabilities, which are out of the scope of this thesis, but are implemented in by them, include sound event detection[25] and an attempt to recognize rare short-time events like glass breaking. The third related work collective also made an excellent job in placing a multiple microphones and exploiting their installation, but we consider their setup too overwhelming for our purposes.

[25] Acoustic Event Detection means to label temporal regions, such that each represents a single event of a specific activity

3.6.3. Overview of Recognition Capabilities of Related Works

After summarizing the related works we can make first thoughts, as to what kind of capabilities a generalized Activity Recognition system should possess, consisting of the presented sound pattern recognition works. Those include Music, Speech and Speaker Recognition, and Activities of Daily Living Recognition (see Figure 3.11).

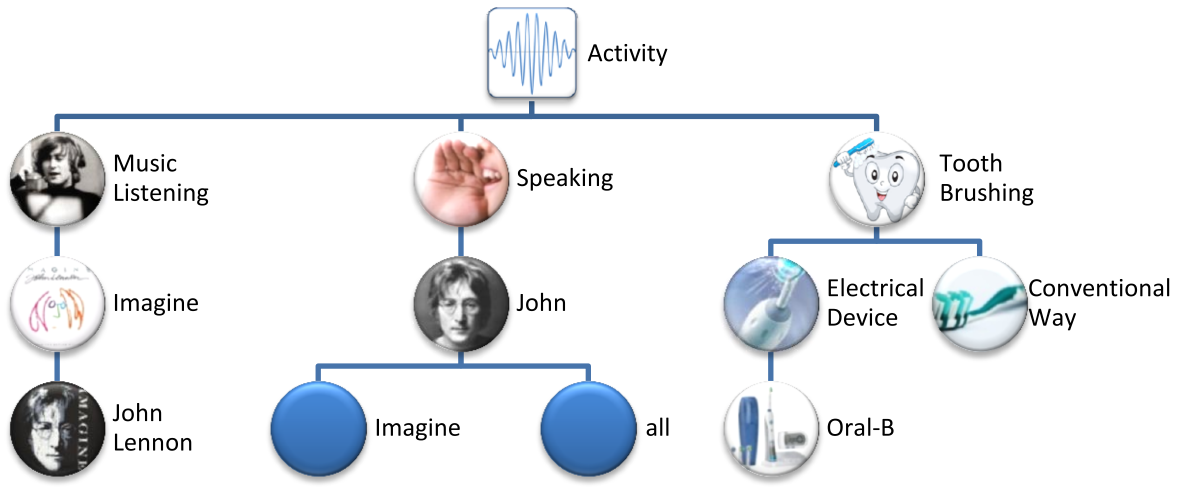


Figure 3.11: *Illustration of the Activity distinctiveness that could be done using a personalized adaptable activity recognizer*

Chapter 4

Concept of Sound-based Device Recognition

In this chapter we merge and extend the gathered knowledge from Chapters 2 and 3 to introduce the concept of Sound-based Device Recognition. To summarize briefly, in a typical home, individuals perform various Activities of Daily Living often with the usage of specific devices. Some of those devices produce distinguishable sounds, which could be captured by existing microphones in the home (e.g. personal computer microphone or smartphone microphone). Usually the performed activities are bound to a certain place in space. This means that from our sound perspective this would “sound” very similar every time. So our feature extraction module should provide us with smartly chosen features that make it possible for us to distinguish the different running devices in a home and send those to a classifier for a further processing and issuing a recognition result (see Figure 4.1).

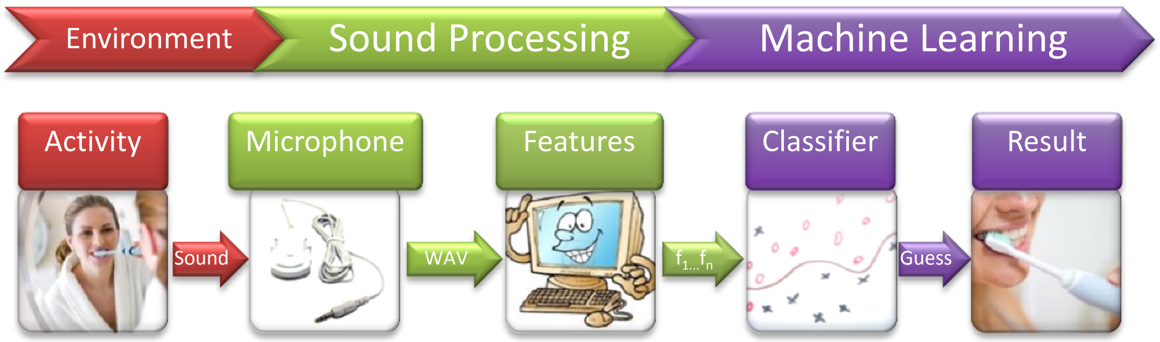


Figure 4.1: *The whole process from user activity to its device recognition attempt by the Sound-based Device Recognition Framework. A separation done within three components – Environment, Sound Processing, and Machine Learning*

In this chapter we first create an abstract architecture of a system for Sound-based Device Recognition. Then we will study its concept starting from the environment, where different activities occur and their acoustics signals are captured. Then we will go through the Sound Processing component and its purpose of extracting audio characteristics. The next step is to get familiar with the Machine Learning component, which is responsible for providing a recognition result after it is supplied with appropriate feature information. After creating the structure of our desired Sound-based Device Recognition system, we define its intended possibilities for integration from a software perspective.

4.1. Architecture

After introducing the recognition process, in this subchapter we introduce the architecture of a Sound-based Recognition system at both abstract and concrete levels. Then we define the development process model, which will be used to implement the architecture.

4.1.1. Abstract Architecture

From a high level architecture point of view (see Figure 4.2), a Sound-based Device Recognition system, deals with the sounds after they have been digitalized and streamed in some way to the recognition system. Ways of digitalizing and streaming include the system doing this with its hardware and microphone input, or within mobile application, which records and streams the information to the system. Then such a system consults its own database for appropriate classification of the input data, while eventually using specialized services for detailed recogtnition information for concrete cases beyond its knowledgebase.

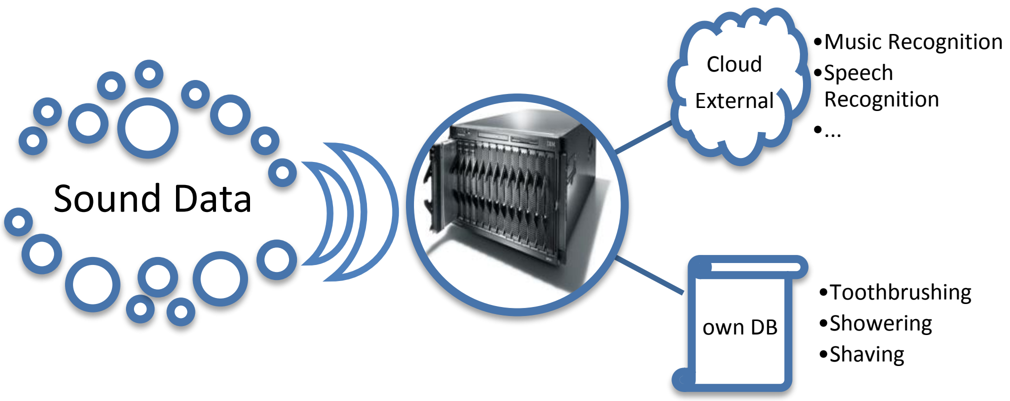


Figure 4.2: *Illustration of the Client / Server (or Agent / Recognition System) architecture. The server in the middle receives sound data either from external device like smartphone, or from its sensors. For the recognition task if own DB lookup is not sufficient the server might use further knowledge from external resources.*

4.1.2. Concrete Architecture

In abstract perspective the work of the illustrated device recognition system is to analyze and classify audio input. So aside from the environment, which we introduce in Subchapter 4.2, there are another two important components, the synthesis of which establish the basis of a Sound-based Device Recognition system. One is the Sound Processing component, which we design in Subchapter 4.3, which employs various mathematical transformations on the sound, already introduced in Subchapter 2.2. The other is the Machine Learning component, described in Subchapter 4.4, which employs different classifying techniques for the device recognition tasks. See Figure 4.3 for illustration of the introduced components of a Sound-base Recognition System, similar to the models in Chapter 3, with respect to their main functionality, as introduced in Chapter 2. One should note that all components are tightly dependable of each other. E.g. the better the chosen features are, the less sophisticated the Machine Learning has to be in order to classify the sounds. The same goes for the connection between Sound Processing and Hardware Setup, the more adjusted a setup is – the clearer the Sound Processing can be. However, as we will see in the next subchapter, we make loose requirements about its setup.

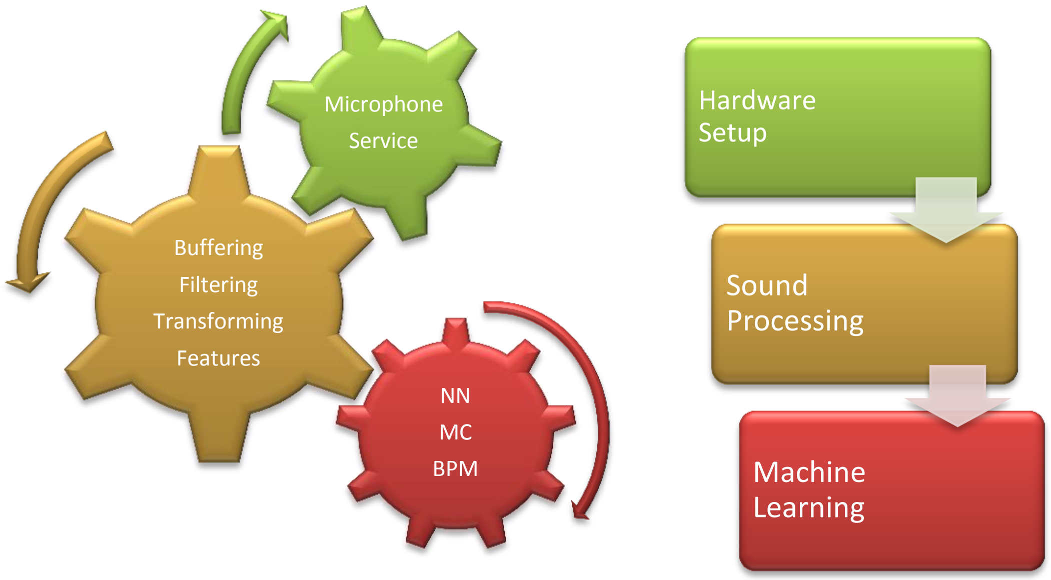


Figure 4.3: *Illustration of the components of the recognition system from software perspective with some of their typical functions*

4.1.3. Spiral Development Process Model

The most suitable development process to implement the described architecture of Sound-based Recognition system among the different development process models introduced in (Pressman, 2009) is the Spiral Development Process Model proposed by (Boehm, 1988) (see Figure 4.4). It describes a standard Evolutionary Process Model and provides the potential for rapid development of increasingly more complete versions of the software. The process iterates through standard steps of cyclic process – Communication, Planning and Designing, Implementation and Testing, and Deployment, described in (Pressman, 2009). In addition, each finished iteration sits on the top of the last iteration and describes a more complete and valuable system.

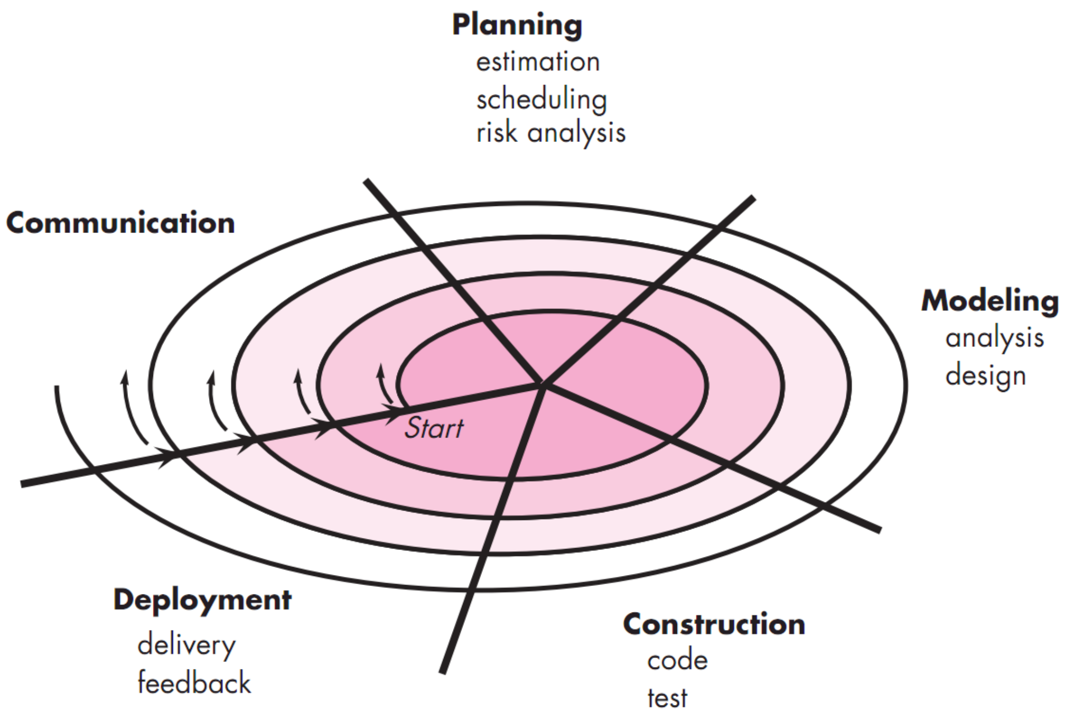


Figure 4.4: *Illustration of the Spiral Process Model (Boehm, 1988) by (Pressman, 2009)*

For example we have first anticipated the need of a Sound-based Recognition system and its most basic functionality, through Subchapters 1.2. and 1.3. Then we estimated their development costs and time to develop. Then we made a model consisting of a hardware and software design in 1.4., which were the basis of our first implementation. After making first functional tests we built a first installation, which covers the process from activity to its device recognition attempt. On that basis we defined what the main fallacies of the system are and decided in which direction the development should continue and so on. This all led to the current version of the Sound-based Recognition Framework implemented in Chapter 5, evaluated in Chapter 6 and to the outlook in Chapter 7.

4.2. Environment and Installation Requirements

In this subchapter we characterize the environment of the recognition system (Section 4.2.1.) and then we describe the recording setup from both Hardware (Section 4.2.2.) and Software (Section 4.2.3.) perspectives.

4.2.1. Environment Characteristics

As a target environment of our framework we choose an average single occupancy (household with single inhabitant). As such it is supposed to have a lot of useful sound properties, like environmental noise usually generate sounds within a broad frequency range (Passchier-Vermeer & Passchier, 2000), speaking of the noises coming from outside our setup environment.

4.2.2. Recording Setup

We choose a single generic sound recording device, e.g. we make no assumption about its capability. There are a few possible setups regarding its placement. One is to choose wearable microphone, and other is to choose static microphone. In order to supply the Sound Processing unit with consistent input one should consider restarting the system when changing the placement setup. Otherwise, it might require complex preprocessing steps like those mentioned in Section 2.2.1. A further note is that for activities, where a wearable setup is not suitable, like showering, we assume static placement.

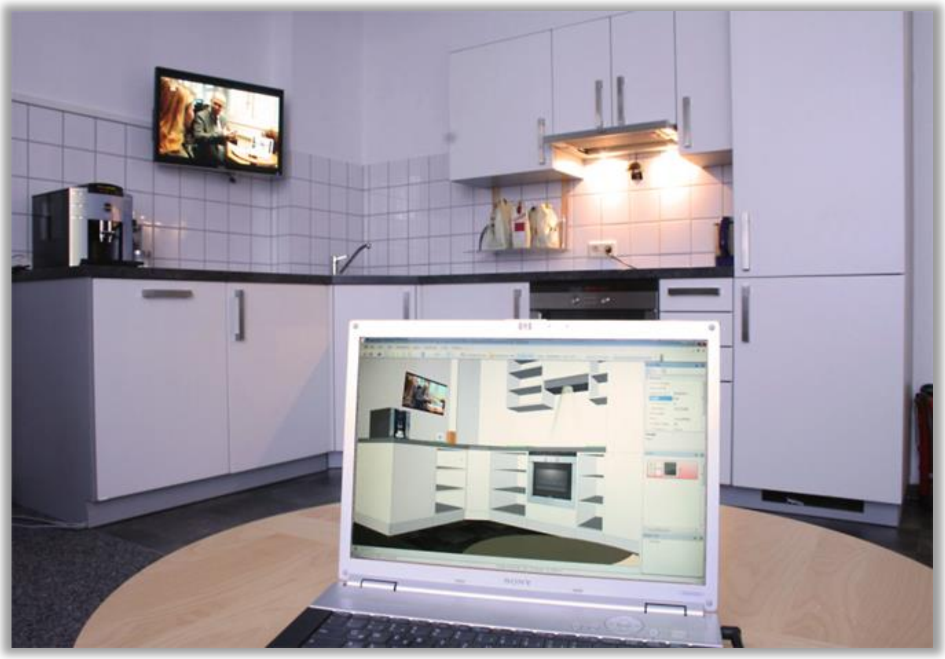


Figure 4.5: *Illustration of laptop that can act as a possible sound recognizer in a kitchen* (Frey, Stahl, Röfer, Krieg-Brückner, & Alexandersson, 2010)

4.2.3. Installation from a Software Perspective

From a software perspective in a static setup, the main processing unit should monitor the sound environment and be able to guess what is happening (similar to Figure 4.5). While a wearable sound sensor would require the client to wear it while performing different daily living tasks and stream the data to the server. In abstraction, the different sounds produced by different activities are captured by the microphone for recognition as shown in Figure 4.6.

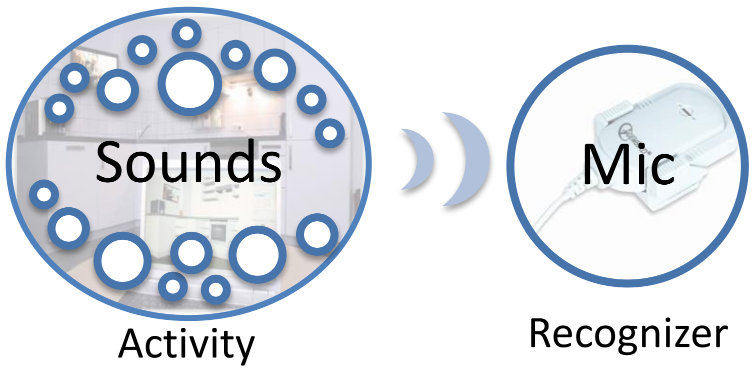


Figure 4.6: *Illustration of different sounds produced by corresponding activities are captured by the microphone and sent to the recognizer*

4.3. Sound Unit

One of the main goals of the Sound Processing Unit is to provide daily activity specific Sound-based Recognition. E.g. to study the existing sound pattern recognition techniques, together with their application in the current setup, and eventually develop their variations to optimize runtime and recognition rate. As a result one should be able to obtain real-time recognition with a good tradeoff between recognition rate and resources used. For this purpose we first discuss the Sound Processing techniques in Section 4.3.1. and then we introduce features choice considerations in Section 4.3.2.

4.3.1. Sound Processing

Initially when developing the Sound Processing unit, one should ask oneself the question, how humans are distinguishing different devices or activities by their sound. A similar research start point has been done in Speech Recognition field, with the derivation of the Mel Frequency Cepstral Coefficients, which are according to the human hearing system. Another similar research start has been done in the Music Recognition field, when the human ability to distinguish a melody, was automated to fingerprint different songs.

4.3.2. Features Choice

As already mentioned in the previous chapter, the main techniques for activity recognition derive from the Speech Recognition and Music Recognition fields. However, there are a vast number of other sound characteristics used to categorize the sound, despite their inability to provide important information either for Speech or for Music Recognition. Other features are not obtainable from tiny sound extracts, thus often ignored for sound recognition tasks. Nevertheless, they play an important role in the context of testing the quality of a music instruments and categorizing songs (Peeters, 2004) (Müller, Ellis, Klapuri, & Richard, 2011), as well as for environmental sounds recognition (Chu, Narayanan, & Kuo, 2009), as mentioned in Subchapter 2.2.

4.4. Machine Learning Element

Knowing that there are a different number of features to be evaluated one should consider appropriate classifying techniques to combine the gathered information. When designing a new system it is important to build up the Machine Learning algorithms from the scratch, instead of running the best overall classifier according to the current research. This enables evaluation of the benefits (such as better recognition rate) and drawbacks (such as slower runtime) on each step of classifiers refinement.

4.4.1. Implementation Considerations

So with an incremental development, we can obtain the important knowledge, whether we can apply less demanding Machine Learning algorithms for recognition in terms of having less training data and user feedback. One should think of the practical application of the framework, which should not count on the user to take a shower ten times in a row or clean his room ten times straight as well as to train the recognizer. In the same manner an unobtrusive system, which won't demand the user to evaluate it constantly, is also a priority in the development. Various algorithms that satisfy this demand are described in Subchapter 2.3.

As we learned in Subchapter 2.2., some features are better for categorizing certain sounds than other. So it is an interesting capability is to decide, which of the implemented features are the most "telling" in terms of recognition reliability. This demands the implementation and test of algorithm, which prioritizes the different features, and according to those priorities calculates a result. Furthermore, it could be the case that in different environments, different features are important to recognize activities, so a system should be able to assign priorities automatically.

Further consideration regarding the features rises when choosing their common metric. One know that by translating the features to some metric in the nearest neighbor algorithm, one could simulate the priorities of the different features, but this won't change the fact that the algorithm would run with all features together. As mentioned in the previous paragraph, a smart algorithm might look at the top priority feature(s) and if those return a confident recognition result, it might not even bother to test the rest of the features. Furthermore, one could be able to automate the process of issuing priorities of the different features in the training phase, by looking at their distributions in space.

4.4.2. Adaptivity

Aside from the classification algorithm one should think of an incremental learning system, which is an important property that the Machine Learning unit should possess. An abstract illustration of a learning system, displaying its incremental learning approach, which aims to make better recognitions over time, can be seen in Figure 4.7. In particular, the system should be able to add new activities and to accept user feedback for its recognitions. In a personalized setup, from the client or user perspective one should also be able to start from scratch and help the system build its Activity Database via Supervised Learning. Over time this will steadily increase the systems knowledgebase for accurate recognitions.

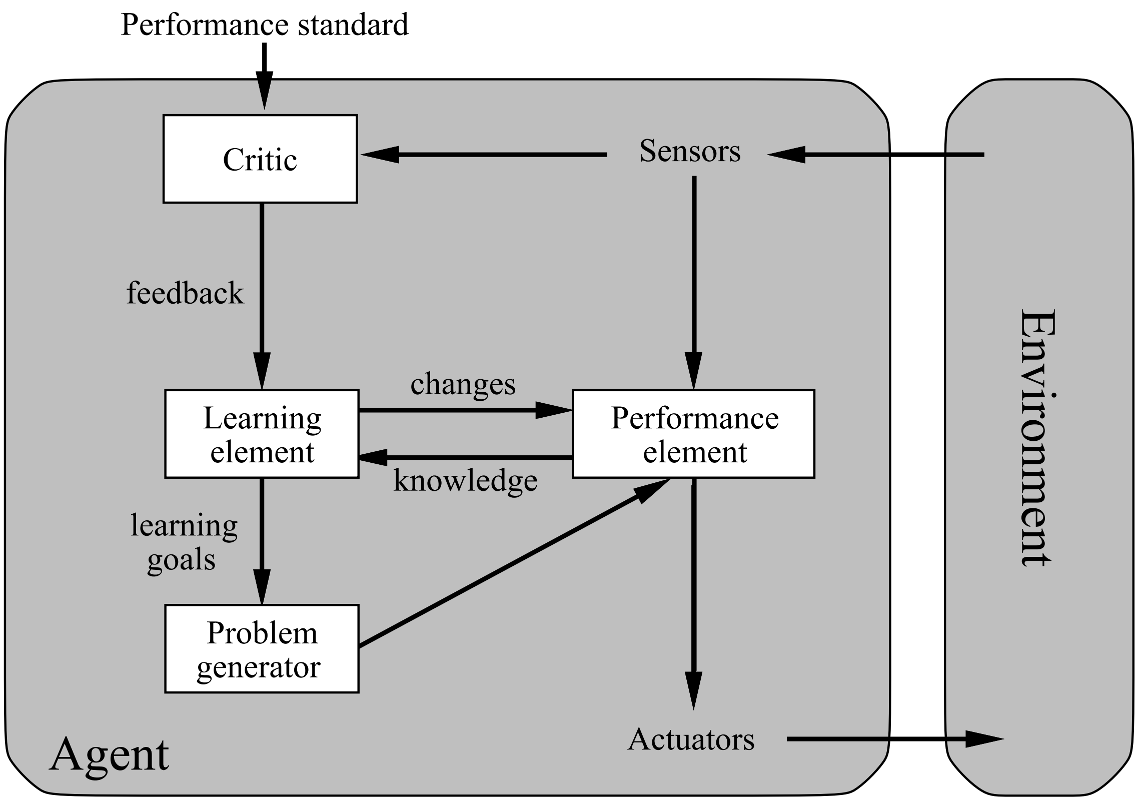


Figure 4.7: Illustration of a learning agent *(Russell & Norvig, 2010)*

When speaking of greater knowledgebase and intelligent incremental learning system, importing established sophisticated classifiers becomes inevitable. Examples of such libraries include Infer.NET (Minka, Winn, Guiver, & Knowles, 2012), WEKA (Mark Hall, 2009), and Accord.NET (Souza, April 2012) (see Figure 4.8).

Figure 4.8: *Logos of the most accomplished Machine Learning libraries - WEKA (Mark Hall, 2009), Infer.NET (Minka, Winn, Guiver, & Knowles, 2012) and Accord.NET (Souza, April 2012)*

A further Machine Learning aspect is creating a personalized recognition model for larger multidimensional training data. Afterwards, a comparison with the implemented optimized classifiers will provide significant information about the tradeoff between speed and recognition accuracy.

Another fundamental reason for making a comparison between different Machine Learning algorithms is that it can serve as a justification of the chosen features and their implementation. The latter means that if the recognition rates of both recognizers are almost the same then the feature sets are categorizing the activities in clearly distinguishable classes, which speaks for their right choice.

4.5. Integration

During the implementation of a Sound-based Recognition Framework one should not only think of it as a standalone system, but should prepare its integration into a bigger system. In this Thesis we prototype an integration as a Class Library (Section 4.5.1.) or as a running Service (Section 4.5.2.) into the AdAPT Project (Frey, 2013) (see Figure 4.9).

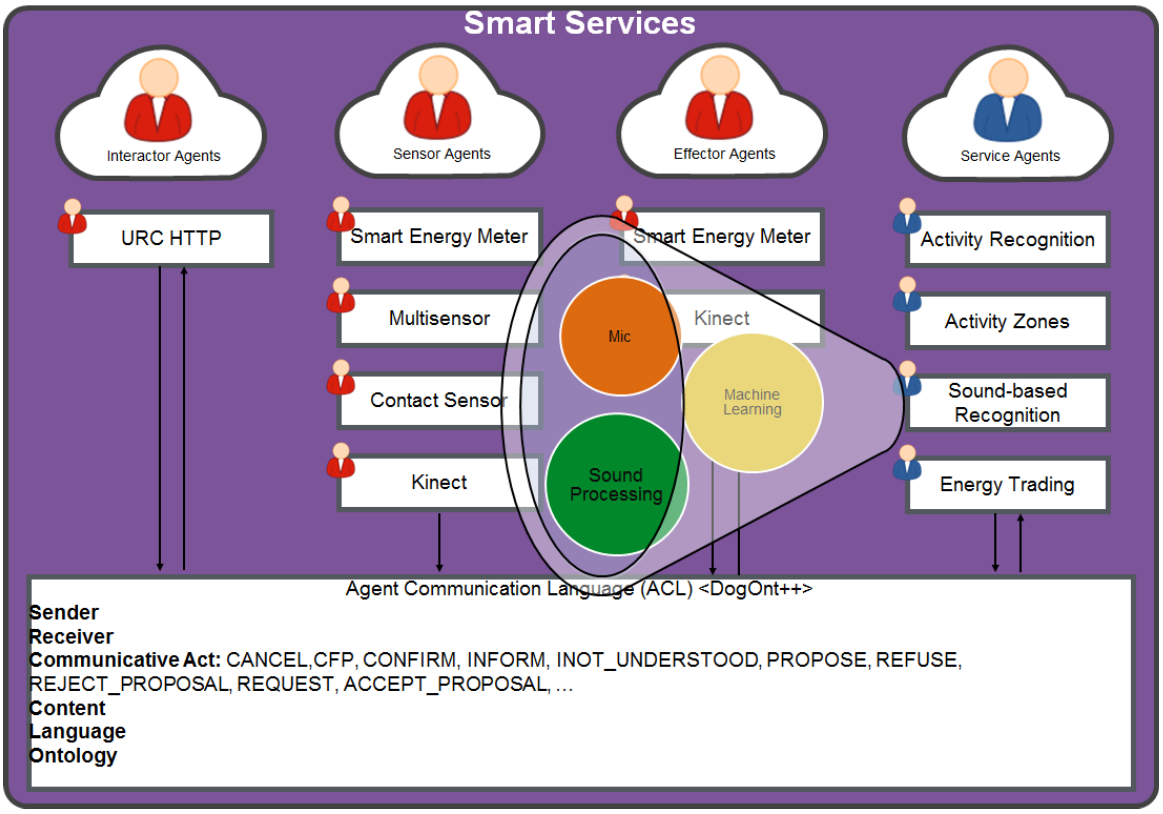


Figure 4.9: *Illustration of Multi-Agent-based Smart Service Platform used in AdAPT (Frey, 2013), where Sound-based Recognition with its components provide acoustic audio-based knowledge about a given environment.*

4.5.1. As a Class Library

Integrating the whole Sound-based Recognition Framework as a component into a bigger system, or providing its classes as self-deployable units, is a direct way to integrate it directly into a bigger system. Therefore, the framework itself and its classes, discussed in Subchapters 4.3. and 4.4., should be exportable as libraries to ease their importing. Integration as a component provides many benefits like real-time debugging and optimized runtime. However, it might increase the complexity of the system and its deployment.

4.5.2. As a Running Service

Developing a connection protocol to offer a running service provides a convenient way for external systems to access the knowledge of the Sound-based Activity Recognition Framework. For example a running recognition system might require supplement acoustic information, for cases, where it has doubts in recognizing some event. Furthermore, using the framework as a service has the advantage of easy integration. E.g. only through some interface, without bothering what is on the other side. Depending on the development intentions, the latter can be considered a flaw in cases, where one might want to have detailed view of the ongoing computations.

4.6. Privacy Issues

Since Ambient Intelligence is tightly embedded in everyday life, it brings fears to both users and developers about “a future in which all of our moves, actions, and decisions are recorded by tireless electronic devices, from the kitchen and living room of our homes to our weekend trips in our cars” (Weber, Rabaey, & Aarts, March, 2005), where “Ambient Intelligence, though often designed to enhance freedom and control, has the potential to limit freedom and autonomy as well” (Brey, 2005). So when designing a system, which monitors the sounds in a private life environment, one should always consider meeting the owner’s acceptance instead of forcing him to accept intrusive privacy policies in cases where he needs the system for some reason like health care.

In our current design case, the Sound-based Device Recognition Framework should erase all records after extracting the desired features. An option for storing the records could be acceptable only during the development period for testing purposes, like changing the sound setup and re-extracting the sound characteristics. Furthermore, in a deployment phase, it makes sense to provide the user with the choice to hide certain activities from the recognizer.

Another issue arises with centralized server architecture[26] and the eventual user reluctance to accept being monitored externally, despite promises of data trust. Further problems are the breakable security systems, which are often the target of third parties, which have great interest of acquiring personal information for reasons including legal enforcements (like the possibility to recognize listening of unauthorized music songs and forcing a payment for them). So a standalone home system capable of performing all designed tasks should be a major priority of development.

[26] If parts of the system are outside of users reach, like a cloud recognition service for multiple households

Chapter 5

Sound-based Device Recognition Framework

In this chapter we discuss in detail the implementation of a Sound-based Device Recognition Framework. We first describe its environment with its characteristics in Subchapter 5.1. Then we introduce the framework solution with its concrete architecture (Subchapter 5.2.) and the developed Graphical User Interface (Subchapter 5.2.), which will help us put implementation fragments in the overall development picture throughout the implementation details of the Sound Processing (Subchapter 5.4.) and the Machine Learning (Subchapter 5.5.) components. Finally in Subchapter 5.6., we carry out a demonstration, how we cover the process from activity to its device recognition.

5.1. Environment and Hardware Setup

Since the main goal of this study is to provide device recognition in the home, the architecture, implementation and the tests are centralized on this infrastructure. For the setup we have a home environment consisting of a single room (see Figure 5.1). For the sound monitoring a single microphone is used. We assume there is only one activity running at time, with a single user that performs it. However as mentioned in the introduction, during certain activities, there may be multiple devices running at the same time, like showering and shaving. We call such activities, which consist of simultaneous occurring activities, complex activities, and we will try to recognize the used devices during them.

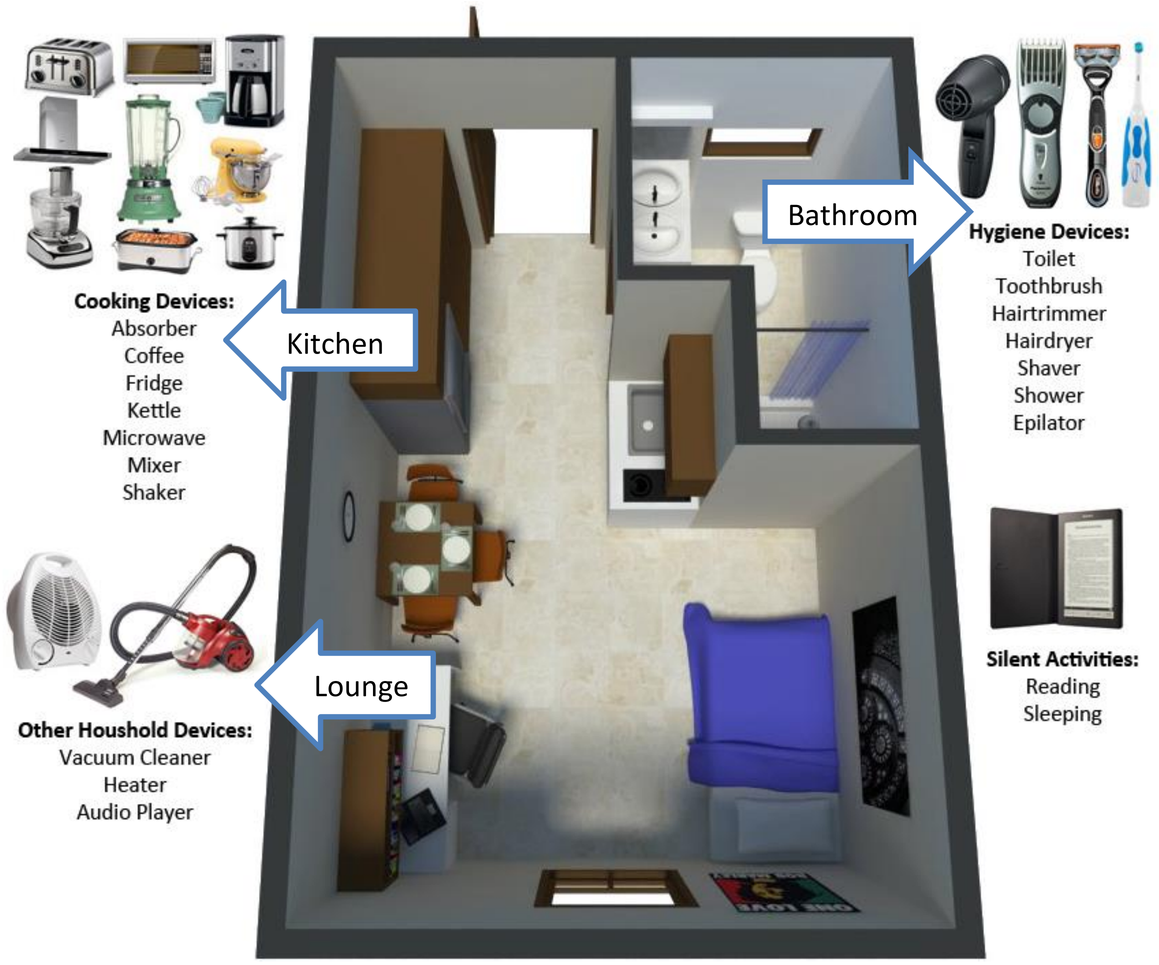


Figure 5.1: *Illustration of a typical one room home environment*[27] *with its typical devices and corresponding locations. One should note that we are interested in those devices, which produce sounds.*

The placement of the microphone can be static or wearable, with the first being less obtrusive and the second providing cleaner input. In terms of acoustics both placements provide similar respective features, as the source is related to the computer or the actuator. However switching between static and wearable setup, as well as other changes in the placement of the microphone in runtime might affect the recognition as well. The reason is that different placement and angles give different frequency responses, which are crucial for recognition. One can compare the displacement of the microphone in audio recognition setup to installing a distortion lens in front of a camera and trust its input data for test after a learning phase before distortion. It is good after every new setup to erase the old training data for accurate recognition.

[27] http://etalk.sgu.edu/contribute/housing/OnCampusHousingOption.htm

5.2. Framework Solution in VS2012

Our framework is written using C# under Visual Studio 2012. It allows easy transition between phone and desktop applications, which is an important feature and provides the possibility of transition to a smartphone application or creating a client/server architecture, aside from the conventional PC implementation. A drawback in the development under VS is the lack of libraries and examples for the specific task of sound transformation compared to other languages and platforms like Matlab, but the solid .NET framework, which is very useful in other tasks, together with the good debugger that Visual Studio offers are making up for the loss.

Before becoming familiar with all components of the system we will study their organization in the main framework project. At first we look at the abstract class diagram in Figure 5.2, showing the primary components of the framework, which are necessary to complete the process from activity to device recognition. On the top of the framework there is a graphical user interface to control all other elements and to provide an environment to connect different modules and test them (see Subchapter 5.3 for details). The two main classes are the Sound Class (Subchapter 5.4) and Machine Learning Class (Subchapter 5.5). For a detailed class organization see Appendix A.

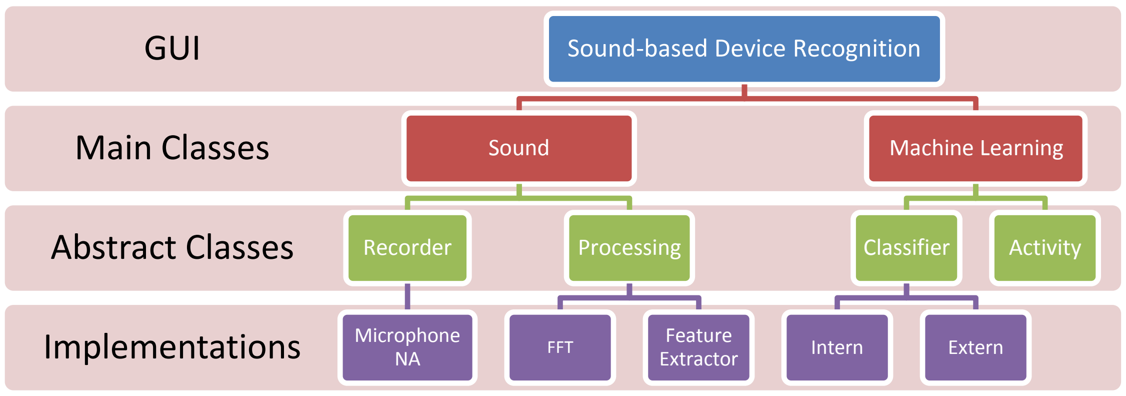
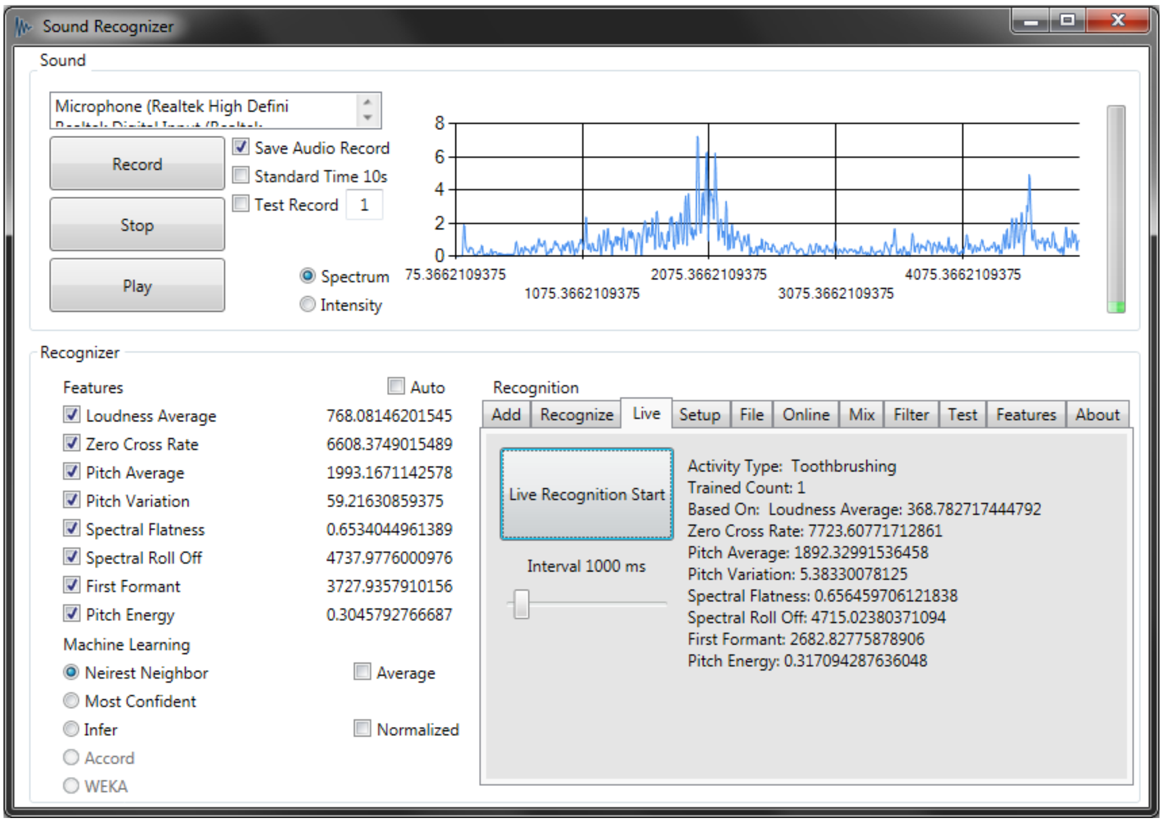


Figure 5.2: *Abstract class diagram of the primary part of Sound-based Device Recognition Framework. On top sits the GUI, which connects both Sound and Machine Learning classes and gives operating power to the user. Both main classes have abstract subclasses for their main capabilities and concrete implementations of those abstract classes*

5.3. Framework Graphic User Interface

Writing a GUI to accompany some software development process is nowadays a natural consequence and brings dozens of benefits. It provides fast access to all functions and combinations of them, and keeps all together in one window. It is also of big importance in visualizing some of the sound derivations or to keep live tracking of the features while testing. Afterwards, it is the best way to share the work and its results with the public. For this reason a GUI was developed in parallel to the main goals and served as a pointer to the current development progress. A snapshot of the GUI in live recognition mode can be seen in Figure 5.3.



Microphone (Realtek High Defini

[✓] Save Audio Record
[ ] Standard Time 10s
[ ] Test Record 1

[✓ Spectrum] Intensity

Features [ ] Auto

[✓] Loudness Average	768.08146201545
[✓] Zero Cross Rate	6608.3749015489
[✓] Pitch Average	1993.1671142578
[✓] Pitch Variation	59.21630859375
[✓] Spectral Flatness	0.6534044961389
[✓] Spectral Roll Off	4737.9776000976
[✓] First Formant	3727.9357910156
[✓] Pitch Energy	0.3045792766687

Machine Learning

[✓ Neirest Neighbor] Most Confident Infer Accord WEKA

[ ] Average
[ ] Normalized

Recognition

Interval 1000 ms

Activity Type: Toothbrushing
Trained Count: 1
Based On: Loudness Average: 368.782717444792
Zero Cross Rate: 7723.60771712861
Pitch Average: 1892.32991536458
Pitch Variation: 5.38330078125
Spectral Flatness: 0.656459706121838
Spectral Roll Off: 4715.02380371094
First Formant: 2682.82775878906
Pitch Energy: 0.317094287636048

Figure 5.3: *Snapshot of the GUI in Live Recognition Mode. On the upper part one can see the sound panel for sound related operations and settings. On the lower part the recognizer panel with its related operations and settings. Most important settings like features set and Machine Learning algorithm are always available to enable runtime changes. The rest of the operations are encapsulated in different tabs in the recognition tab control*

The GUI is divided in two panels, one for the sound related functions, such as recording or visualizing, and second for the recognizer related functions, like its setup and output. The GUI parts are also meant to be accessed consequentially. E.g. firstly setting up the sound and testing, whether it works. Then choosing the feature set for recognition, together with the Machine Learning algorithm. Finally starting recognition, with further advanced properties available.

In order to provide in runtime enabling / disabling of features, also switching between different Machine Learning algorithms, both Features and Machine Learning panels are outside the Recognition panel tab control, where everything else is encapsulated for its concrete usage.

5.4. Sound Class

The Sound Class encapsulates all classes and methods responsible for Sound Transformation (Section 5.4.1.). In particular Spectrum Derivation and Feature Extraction (Section 5.4.3.). Further important capabilities are Sound Preprocessing for Noise Cancellation and Equalization (Section 5.4.2.), Wave File Mixing for dealing with multiple activities at same time (Section 5.4.4.), and Wave Recording and File Parsing for enabling a flexible input. Less important functions of the sound unit in terms of recognition, but important for testing, include sound playback and visualization. Since the native sound related libraries provided by Microsoft are not very consistent in time in terms of their interfaces and usage, we use an additional library for basic audio features, like recording and stream handling, called NAudio[28].

In this subchapter we describe all steps from Preprocessing through to Filtering and finally extracting features as illustrated in Figure 5.4. At the end we introduce the mixing of sounds for recognition of multiple used devices in complex activities.

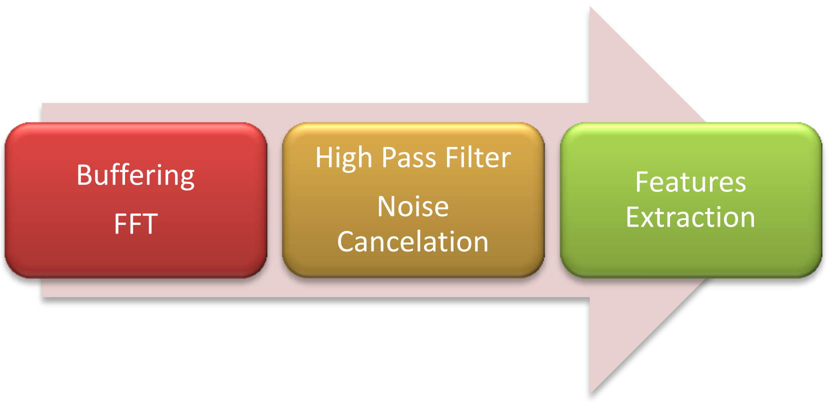


Figure 5.4: *Illustration of the main steps of the Sound Processing unit pipeline*

5.4.1. Sound Transformation

Here we will get details in Sound Transformation and in particular the performed short-time Fourier transform, but first we need to describe the buffering and its setup.

Buffering Setup

Besides setting up the hardware there is a need to setup the software and its recording parameters. When one chooses flexible recording settings, one still has to convert those unified metrics for the later transformation, which always results in data loss. So instead of bothering with conversion, we fix the recording parameters to unified supported constants, which satisfy our needs. Furthermore, our choice will help us to analyze the recorded data and the produced features for consistency and effectiveness.

[28] NAudio 1.7 Available at http://naudio.codeplex.com/

The sampling frequency of the recorded sound is set to the most common used 44100 Hz (also used in Compact Disks). It has been chosen because any hardware supports this rate. Not all of the microphones support stereo recording though, which might produce some deviations during the hardware setup as well, so the number of channels is reduced to one (mono recording). For the sound transformations it is useful to have high precision variables, so the bit depth is set to 32 bit signed integer per sample, which doubles the standard bit depth value of 16 and is uniformly supported as well. Overall the user setup is reduced to choosing the sound input and options for standard time recording, recording storage confirmation, and visualization modes (see Figure 5.5).

A further feature of the sound unit and its visualization is triggered by clicking the graphic, which makes a snapshot of all chart data points and saves them to the database in snapshot section as XLSX[29] and RAW[30] formats. The later can be used for in depth analyzing and visualization using different tools, like Microsoft Excel or Matlab. For example, all image examples in this thesis are imported in this way.

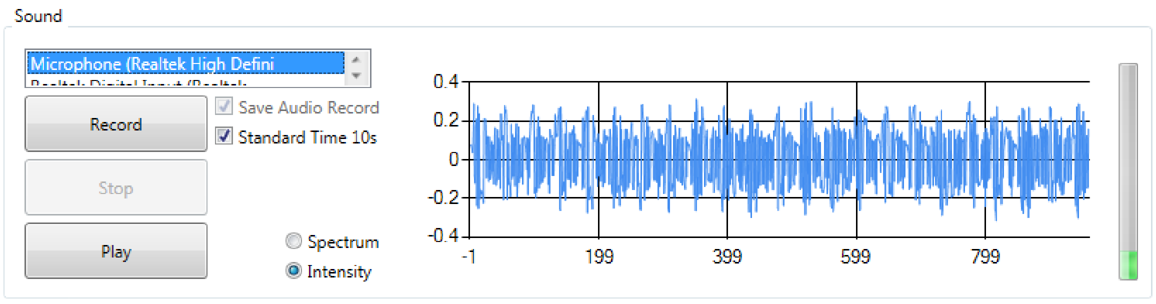


Figure 5.5: *Illustration of standard time recording of 10 seconds using microphone input and displaying the intensity in the graphic (Y-axis amplitude, X-axis time in deciseconds)*

To obtain better FFT precision we set the FFT buffer to 4096 samples. For record frames we've chosen a variable time length, which is at least twice the FFT buffer size. However to standardize data for tests there is a standard record length checkbox for 10 seconds.

Short-Time Fourier Transform Implementation

For the Short-Time Fourier Transform we use a C# translation of the C++ implementation by S.M. Bernsee (Bernsee, September 21, 1999) with the exact implementation adapted for C# by Mark Heath in NAudio. Since the algorithm provides imaginary valued output we map it to real numbers using Euclidean distance to obtain the absolute values of the complex numbers. We use overlapped buffering to overcome the loss of sound data at the edges of the windows (See Figure 5.6).

[29] Excel Microsoft Office Open XML Format Spreadsheet File
[30] With .txt file extension, starting with information header, followed by values separated by a tabulator

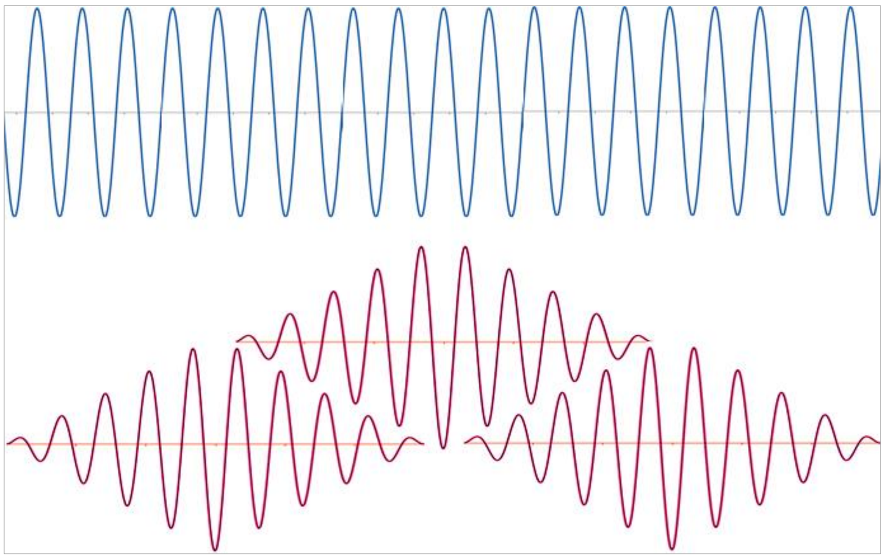
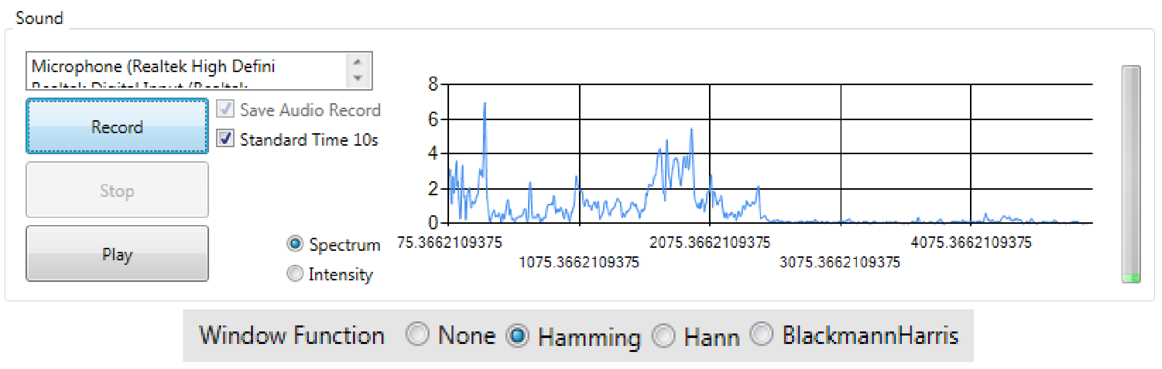


Figure 5.6: *Illustration of window overlapping to overcome buffer leaks after windowing. Line above is the source buffer, while the redline represents the three overlapping buffers sent for FFT.*

For a Window Function we use primary Hamming Window with two supplement choices – Blackmann-Harris and Hann Windows shown in Section 2.2.3. One can also skip usage of Window Function. After the transformation, the resulted spectrum is displayed in the sound related part (see Figure 5.7).



Sound

Microphone (Realtek High Defini

[✓ Save Audio Record] [✓ Standard Time 10s]

[✓ Spectrum] Intensity

Window Function: None [✓ Hamming] Hann BlackmannHarris

Figure 5.7: *Screenshot of spectrum visualization using Hamming window of frequencies between 80 and 5000 Hz (X-axis) with their relative energies (Y-axis)*

After making the Fourier transformation we extract the frequency range between 80 Hz and 5000 Hz for further recognition. This frequency range contains the most distinguishable features in the spectrum, according to our manual study and automated evaluation of different signals. Further reasoning for this choice is the unsteady behavior of the frequency response, which different microphones provide (see Figure 5.8). For further disturbances in frequency response of our chosen range we implement various Filters at spectrum level, as explained in the next section.

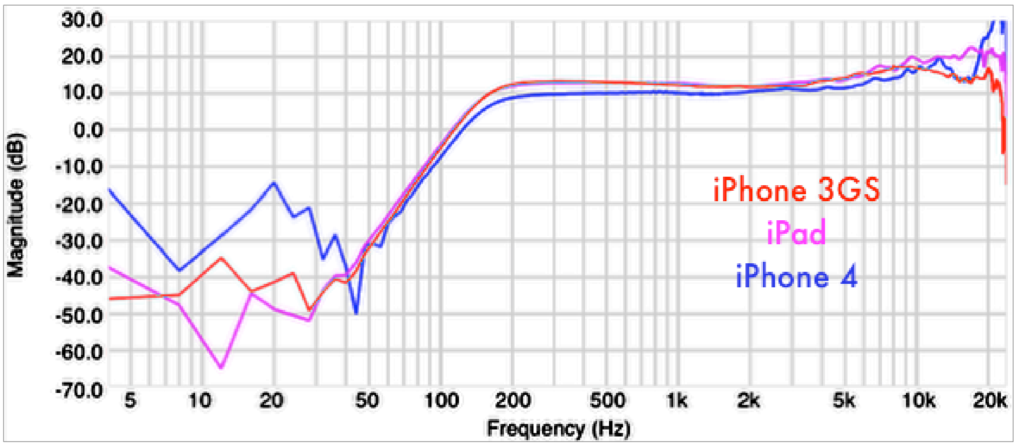
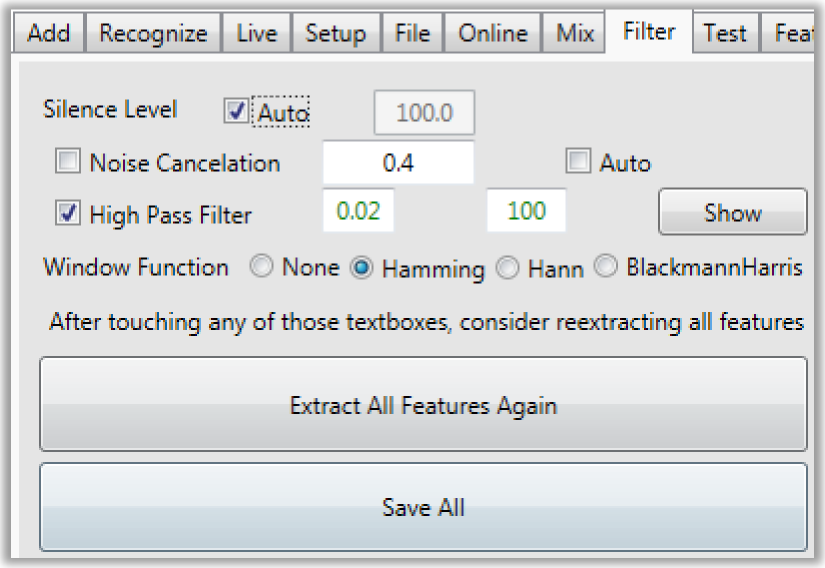


Figure 5.8: *Frequency response plot for build in microphones of different iPhone series*[31] *displays large deviations in lower and higher frequencies.*

5.4.2. Sound Filtering

Sound Filtering is important step to overcome some issues with microphone or environment noise, which often occur in practice. We will first describe the importance and implementation of a High-Pass Filter to overcome irregular deviations in frequency response for some microphones and then we introduce straight forward Noise Cancelation to eliminate low level and evenly distributed environment noise.



Silence Level [✓ Auto] 100.0

[ ] Noise Cancelation 0.4 [ ] Auto

[✓ High Pass Filter] 0.02 100

Window Function None [✓ Hamming] Hann BlackmannHarris

After touching any of those textboxes, consider reextracting all features

Figure 5.9: *Filter tab screenshot with options for, Noise Cancelation, High-pass Filter setting with visualization option, and Window Function choice. Note, that after making changes to some on those options there is usually a need to extract all features again from their source sound and save them*

[31] By Faber Acoustical LLC at http://blog.faberacoustical.com/2010/ios/iphone/iphone-4-audio-and-frequency-response-limitations/

High-Pass Filter

Most microphones are very prone to deviations in lower frequencies (see Figure 5.10 above), which made us implement a High-pass Filter to overcome those problems.

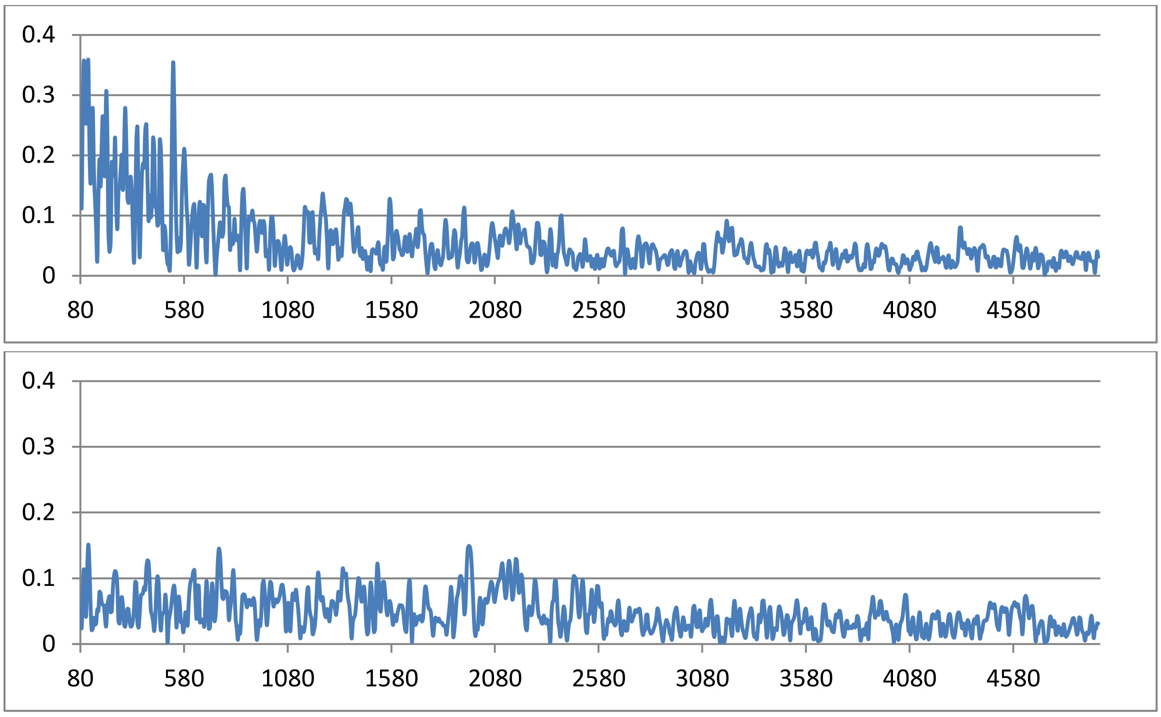


Figure 5.10: *Plots of the spectrum between 80 Hz and 5000 Hz of the microphone used for later for tests with and without High-pass Filter*

In this work we implement a discrete version of a so called First Order High-pass Filter using the following formula:

$$Spectrum[Index] = Spectrum[Index]\frac{1}{1 + \big((HighPassThreshold - Index) * Coefficient\big)}$$

The effect of applying the filter with $HighPassThreshold$ of 100 and Coefficient of 0.02 can be seen in Figure 5.10.

Noise Cancelation

As mentioned in Section 4.2.1., our recognition is in indoor environment with single inhabitant, so we assume that the single inhabitant and his actions are the main source of sound signals. The other sources are the devices performing periodic tasks, like air conditioner while normalizing the room temperature. In this development phase we assume, that there are no sound signals outside our home environment, which are louder than those occurring inside.

After changing the High-pass Filter, Noise Cancelation, or the Window Function, one should reconsider extracting all features, which use the spectrum domain, from the recorded test data again. Most affected features are Average Loudness, Pitch Energy, Spectral Flatness and Spectral Roll Off. For Zero Cross Rate it makes no change, while for the rest of the features one might perceive small deviation in the feature values, which still could make a big difference, especially for the silent activities.

5.4.3. Features Extraction

After transforming and filtering the sound it is time to extract features. First we will study different sound spectrums and then we will justify the decision of which features are worth implementing. Then we will explain the chosen features from in the context of device recognition with their implementation.

Acoustic Characteristics of Activities and Devices

In order to make a decision which features we should select we first studied the nature of the sounds produced by devices using speech and music analysis software (see Figure 5.11) (see Appendix D. and E. for further spectrograms of device sounds). The first notable difference from the mentioned domains was that our signals were most noise-like, similar to the environmental sounds studied by (Chu, Narayanan, & Kuo, 2009). So we had to consider a specific feature choice, different from the one used by speech and Music Recognition fields.

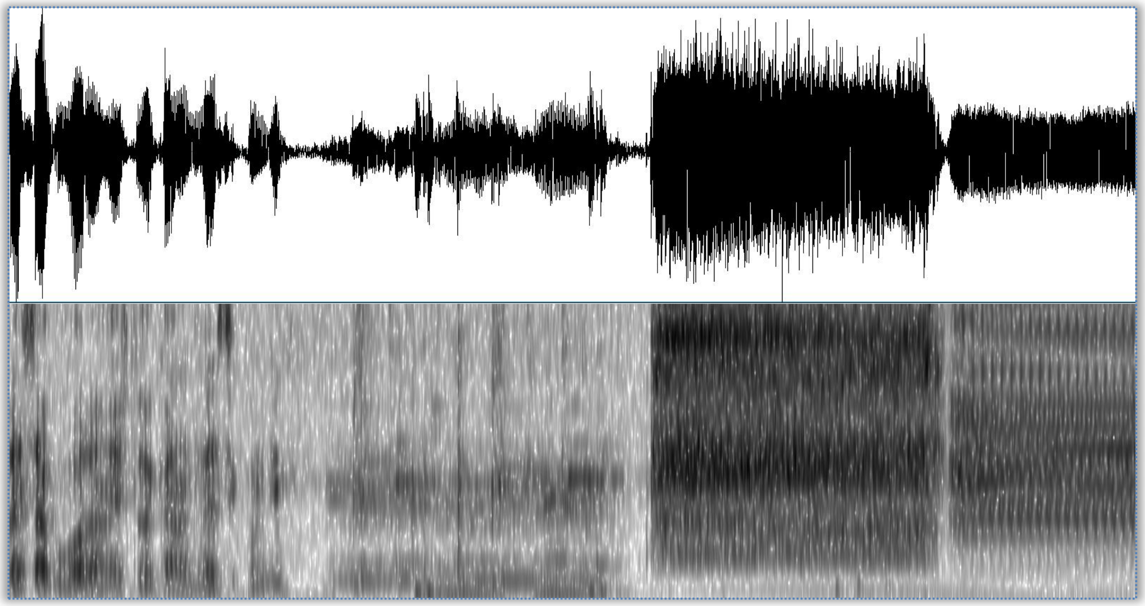


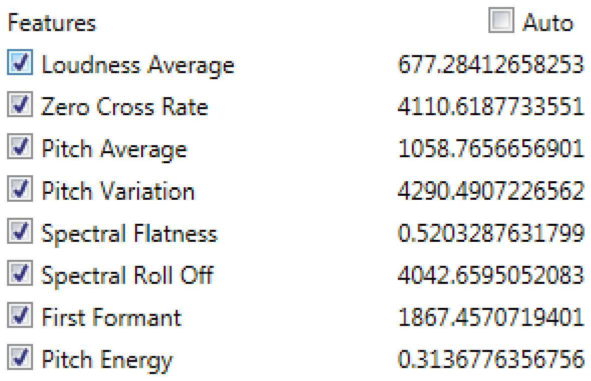
Figure 5.11: *Plot of 10 second recording of speech (first quarter), music (second quarter), epilator (third quarter) and hair trimmer (last quarter). Above we see the waveform of the recording and below its spectrogram between 80 Hz and 5000 Hz.*

A further remark we made is that, for activities performed with electrical devices it is typical that most of the defining part of the sound comes from its electrical motor, which is the actual sound source. Also interesting was that other activities performed with non-electrical device, like showering, had spectrograms similar looking to electrical ones.

Overall, we made the conclusion that most of the devices produce monotonic sounds, which are noise-like. Their further characteristics are not steady and are very implementation dependable. For example, calculating the pitch of a speaker is not a hard task compared to calculating pitch of a hair trimmer, as evident in the spectrogram in Figure 5.11. So after changing the feature derivation method, one should re-extract the updated feature from the training data to avoid classifier confusion.

Implemented Features

After getting familiar with the nature of sounds produced by devices, we introduce the implemented features in the Sound-based Device Recognition Framework (see Figure 5.12), also previously explained in Section 2.2.4.



Features	☐ Auto
✓ Loudness Average	677.28412658253
✓ Zero Cross Rate	4110.6187733551
✓ Pitch Average	1058.7656656901
✓ Pitch Variation	4290.4907226562
✓ Spectral Flatness	0.5203287631799
✓ Spectral Roll Off	4042.6595052083
✓ First Formant	1867.4570719401
✓ Pitch Energy	0.3136776356756

Figure 5.12: *Features panel screenshot displaying the enabled features for recognition together with their current value in the according local recognition metrics (see the list of features below for details over the metrics)*

We present the features this time in order of their addition to the Framework over the implementation process, which was influenced by our perception of sound and the conclusion from the last subsection. For example, the first perceivable feature of a sound is its loudness, so we chose to start with it. Then, in a mathematical perspective, zero crossings are one of the most important characteristics of a function, together with its maximums and minimums. Subsequently, we implemented a set of 8 features, for the task of audio-based device fingerprinting.

Loudness

We compute it by calculating the average cumulative energy of the spectrum over the recognition interval. Note that this is a relative measure and is very dependent of any Filters and especially Noise Cancelation algorithms.

Zero Crossing Rate

Zero Crossing Rate is the only feature derived from the time/amplitude domain (e.g. without processing the raw signal), after deciding to compute loudness after the filtering. To count zero crossings we check whether we have a zero crossing after each received sample. In order to compute the Zero Crossing Rate for a given time interval and number of zero crossings we use following formula:

$$ZCR = \frac{1}{t}\big(ZeroCrossCount(t)\big)$$

Where $ZeroCrossCount(t)$ is the number of zero rrosses over time period t. For example if we have zero cross count $ZCC = 5000\ zero\ crosses$ for time $t = 10\ s$, for our zero cross rate we obtain $ZCR = 500\ crosses/second$. Furthermore, for a simple periodic signal, this will mean a frequency $F = 250\ Hz$, which is the fundamental frequency F_0 as well (recall that two zero crosses build one period of the signal, so for the frequency F of a simple signal we have $F = ZCR\ /\ 2\ Hz$ for a ZCR measured in zero crosses per second). We use this loose relation later, when defining common metric for the different feature metrics.

Furthermore, in order to obtain bigger floating point precision and at same time to minimize floating point error, the exact computation of ZCR is different than straight translation of its formula.

Pitch Detection

To detect the Pitch we scan the spectrum for the highest value there. Its functionality was tested via frequency tests. The measure of the Pitch is in Hertz.

Again, since we are dealing with variable size buffers, which are discrete representation of the frequency band, we need to make couple of calculations to compute obtain the frequency value in Hertz using following formula:

$$F_0 = SampleRate\,\frac{MaxBinIndex}{FramesCount}$$

Where $MaxBinIndex$ is the index where the maximum value occurred in the buffer, which has size of $FramesCount$. Together $MaxBinIndex/FramesCount$ build up the relative position in frequency between 0 and $SampleRate$. Note that the value at the $MaxBinIndex$ might be different according to the chosen Window Function and the exact FFT computation. Furthermore, due to the discrete representation of the spectrum, the computation of the Pitch is exact with possible error of $\pm BinSize = SampleRate/FramesCount$.

Pitch Span

Straight pitch produces monotone sound. Varying pitch with steady average value sounds like whirring. This is also the only feature, which is highly dependent of the recognition buffer length, since the smaller the recognition buffer length, the smaller the chance to observe bigger pitch span. One can think of this feature also as a temporal feature. The future inclusion of further temporal features is also of high interest as stated later in the conclusion (7.2). The measure of the pitch span is in Hertz.

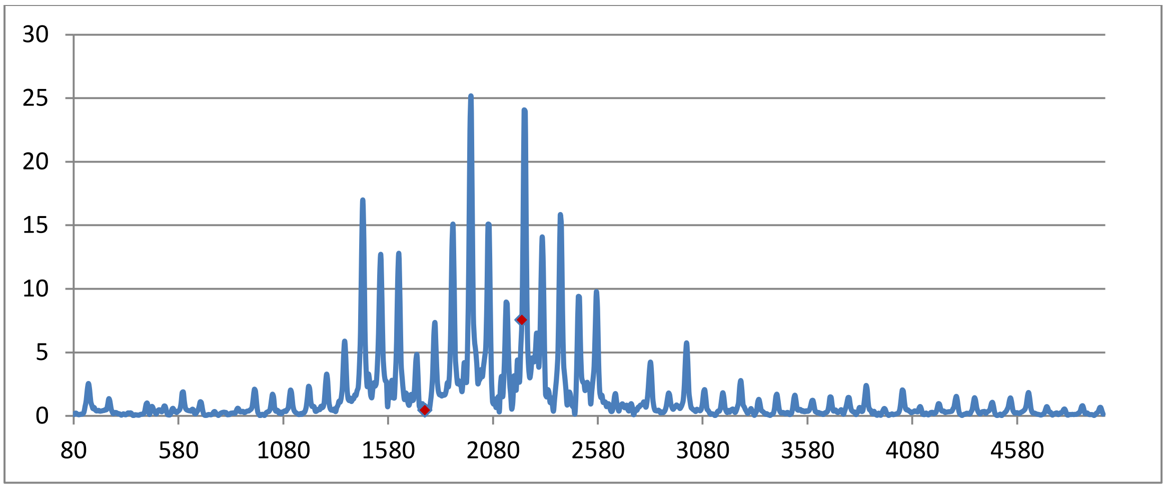


Figure 5.13: *Plot of spectrum of a hair trimmer illustrating that determination of the Pitch and the First Formant are hard task. For the current snapshot the Pitch would be calculated as 1975 Hz, while the First Formant, would be 2234 Hz. One should note that over time those peaks switch, making the pitch vary between the mentioned two values, which is the way we compute the Pitch Span. Red points represent the beginning and the end of the interval for computing the Pitch Energy.*

For some devices, like a vacuum cleaner (see Figure 2.12), the Pitch is steady and doesn't vary over time, while for some other devices, like a toothbrush, the pitch varies over time. In some cases, Pitch Span refers to the distance between the Pitch and the First Formant over time, due to ambiguity of automatic distinction between local maximums (see Figure 5.13).

Pitch Energy

This is the amount of energy as part of the whole energy, which surrounds the pitch in a 10% rectangular window (e.g. the 10% of the signal around the pitch as middle point). In Figure 5.13 Pitch Energy refers to computing the energy between two local peaks left from the pitch and two peaks right from it, which is between 1729 Hz and 2221 Hz as marked with red points.

First Formant

We compute the First Formant by finding the first local maxima in the spectrum after the pitch. We start the search after skipping some indexes from the spectrum array in order not to misinterpret parts near the Pitch as First Formant. The metric of the First Formant is in Hertz. As we can see in Figures 5.11 and 5.13, for some devices like a hair trimmer, this feature is hard to obtain and provides often ambiguous results.

Spectral Flatness

This is an important measure, which is very useful to distinguish meaningful sound from noise. The formula to compute Spectral Flatness is:

$$SpectralFlatness = \frac{\sqrt[N]{\prod_1^N Spectrum(index)}}{\frac{\sum_1^N Spectrum(index)}{N}}$$

Where N is the size of the spectrum buffer $Spectrum(index)$ is the bin value at the given $index$ position.

Again due to possible floating point precision problems occurring in computation of the product the exact computation of the Spectral Flatness is done via intermediate calculations of the product chunks by using the following mathematical formula

$$\sqrt[n]{xy} = \sqrt[n]{x}\sqrt[n]{x}$$

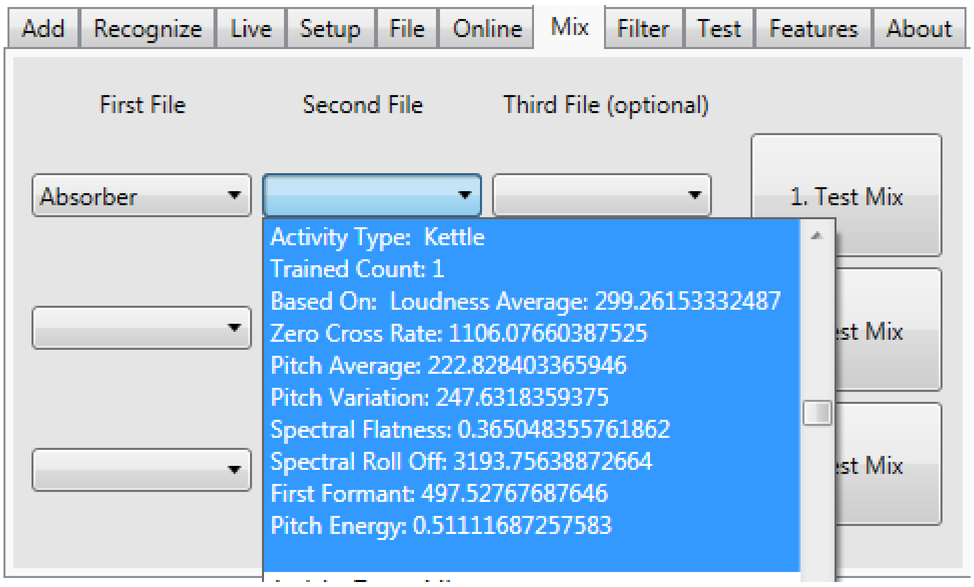
Where x and y represent two positive real numbers. We should also note that since some of the spectrum indexes might be zero values, we exclude from the overall calculation, because their presence might lead to division by zero in the intermediate computations.

Spectral Roll Off

As mentioned in Section 2.2.4., Spectral Roll Off is important measure about the energy distribution over the spectrum. For its computation, we determine the point where we have 95% of spectrums total energy. We measure the Spectral Roll Off in Hertz.

5.4.4. Sound Mixing

Mixing sounds for their recognition is a novel approach in the field of Sound-based Activity Recognition. It has been discussed in the field of Music Recognition for mixing different instruments in order to attempt their combined recognition (Wieczorkowska, Kolczyńska, & Raś, 2008). However the technique used here is slightly different and avoids volume normalization (see Section 2.2.2.), which is important for musical instruments, since they can play at different intensity, but mostly irrelevant for devices, which often have steady loudness. We are aware that defining which activities can occur simultaneously is a hard task. That’s why in our Framework we implement up to three mixing possibilities, mostly for research purposes. Each mix consists of at most three records (see Figure 5.14).



Activity Type: Kettle
Trained Count: 1
Based On: Loudness Average: 299.26153332487
Zero Cross Rate: 1106.07660387525
Pitch Average: 222.828403365946
Pitch Variation: 247.6318359375
Spectral Flatness: 0.365048355761862
Spectral Roll Off: 3193.75638872664
First Formant: 497.52767687646
Pitch Energy: 0.51111687257583

Figure 5.14: *Sound Mixing Tab, which enables mixing of two or three of the available activities.*

To create a mix of recordings we iterate through their values and add them. Often when mixing signals there occurs a problem called Audio Clipping (see Figure 5.15). It happens when the sum of the mixed values exceeds its range. A way to avoid clipping is by lowering the amplitude of the whole mix, but by doing this, one obtains erroneous information in some features like Loudness. We overcome the problem of clipping in our audio setup by adjusting the recording levels in the installation step. Furthermore, mixing sounds with lower overall amplitudes might introduce lower precision, due to rounding error of the chosen bit dept. However, our choice of 32 bit depth provides plenty of resolution even at lower recording levels.

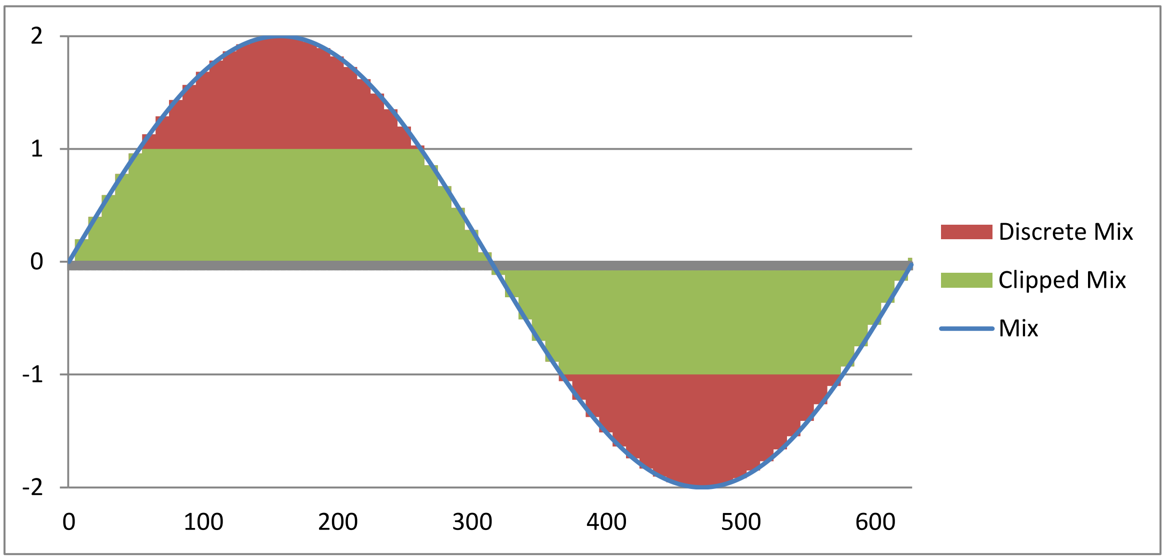


Figure 5.15: *Plot of single period of the function from Figure 2.6 added with itself (blue line). The discretization of its mix is out of the interval [-1,1], so all values out of the interval will be lost (Red Part) and we will obtain only the rounded discretization (Green).*

A further benefit of sound mixing for recognition can be seen if we compare it with video-based recognition, where one needs complex modelling in order to simulate or recreate how someone would look like when performing simultaneously multiple activities, in order to recognize them.

5.5. Machine Learning Class

Unlike the pipe process nature of the Sound Class, which constantly transforms the incoming sound signal, the Machine Learning Class evolves with time in terms of its knowledge and capability (see Figure 5.16). In this subchapter we introduce the Activity Class, which encapsulates information for the different types of activities and the devices used to perform them (Section 5.5.1.). Then we discuss automatic prioritizing and selecting features (Section 5.5.2.), as well as their metrics and normalization (Section 5.5.3.). Finally after introducing the different activities and their feature sets, we study the implemented Machine Learning algorithms for classification (Section 5.5.4.), together with the imported Machine Learning techniques (Section 5.5.5.).

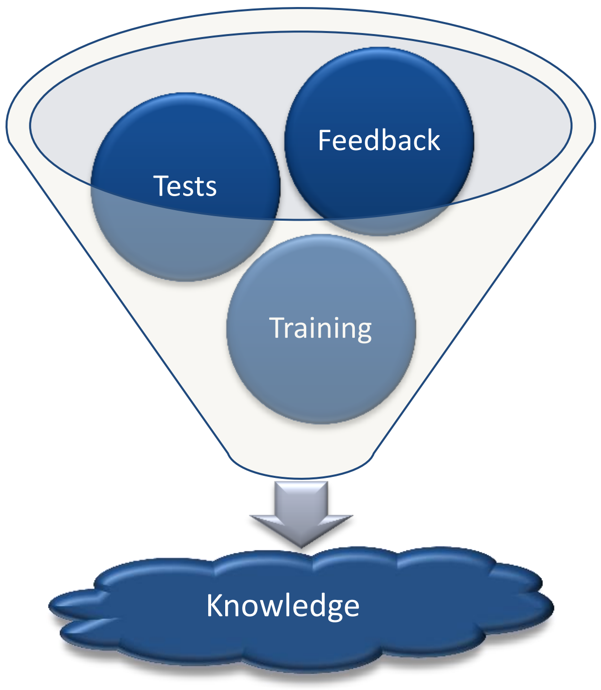


Figure 5.16: *Illustration of the evolving nature of the Machine Learning unit, which increases its knowledge with the occurring training and tests, and the feedback by the user*

5.5.1. Activity Class

In this section we define our activity class, which comprehends different types of activities, which are of interest in this thesis, due to the various device types used to perform them. Afterwards we explain the way we store the gathered information.

Activity and Device Types

In our Framework the Activity Type encapsulates all types of activities, which produce sounds for time intervals bigger than our smallest recognition window of 0.1 s. Since we are interested in recognizing the devices used in activities we introduce a Device Type for devices used to perform the according activity (see Appendix B for mapping between those two type). One should also note that, not all of the activities are performed by humans. For example, a heating element starts heating alone, when the temperature falls beyond given threshold. We define the state of device as active, if they disperse sounds. According to the heater example, it is on standby or inactive while measuring the temperature and active while heating.

To enable deeper knowledge about the used devices there is a checkbox to determine whether the activity is performed with electrical device. On this basis one can make statements in recognition whether the recorded activity is performed with electrical device. For example if we have five activities like brushing teeth with an electrical toothbrush, cleaning using a vacuum cleaner, coffee making using a moka pot, showering and speaking, we can separate those activities into three categories – performed with an electrical device (first two), performed with non-electrical device (second two)[32] and the last one performed without any device at all (see Figure 5.17). So after issuing recognition the recognizer assigns different probabilities for the occurrence of each activity, and by those probabilities, one can make statements as to which of the currently defined three activity categories this was. A sample formula to determine this could be, by looking, whether all electrical devices score probability over some threshold, like 0.5, while all other activities score probability below this threshold (see sample mathematical definition below).

$$ElectricalDevice = (P\{ED\} > 0.5) \wedge (P\{\neg ED\} < 0.5)$$

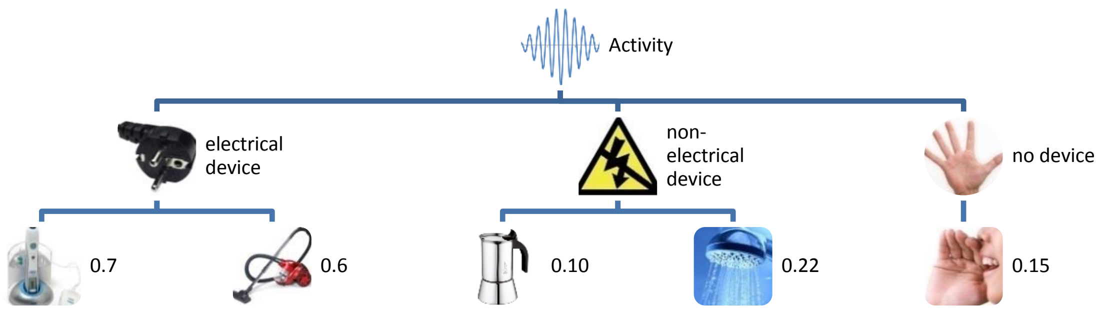
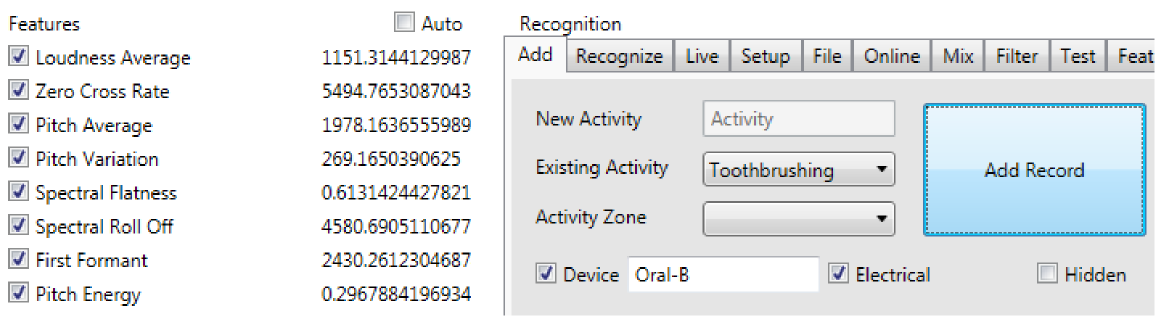


Figure 5.17: *Example of possible activity division into three categories – using electrical-, non-electrical-, and no-device. At the lowest level we see the concrete activities with their current occurrence probability. One can clearly make the conclusion from those probabilities, that the occurring activity is most likely performed with electrical device, because both listed electrical devices obtain much higher probability according to the classifier, while making a distinction between non-electrical device and no device at all is ineligible since both classes obtain similar probability.*

[32] Indeed, one usually heats the moka pot with a hot-plate, and the water for the shower with a boiler

Activity Database

The Sound-based Device Recognition Framework builds up its Activity Database with the user assistance via Supervised Learning, as explained in Section 2.3.1. So we rely on the user to supply the recognizer with sufficient information about his activities. We should note regarding the difference between activity and its devices, discussed in Section 2.1.1., that the user still labels an Activity Type and Device Type combination. This is so, because from user's perspective the actual activity is known, while from Sound-based Recognition perspective we cannot apply an inference from device to activity with confidence.



Features ☐ Auto

✓ Loudness Average	1151.3144129987
✓ Zero Cross Rate	5494.7653087043
✓ Pitch Average	1978.1636555989
✓ Pitch Variation	269.1650390625
✓ Spectral Flatness	0.6131424427821
✓ Spectral Roll Off	4580.6905110677
✓ First Formant	2430.2612304687
✓ Pitch Energy	0.2967884196934

Recognition

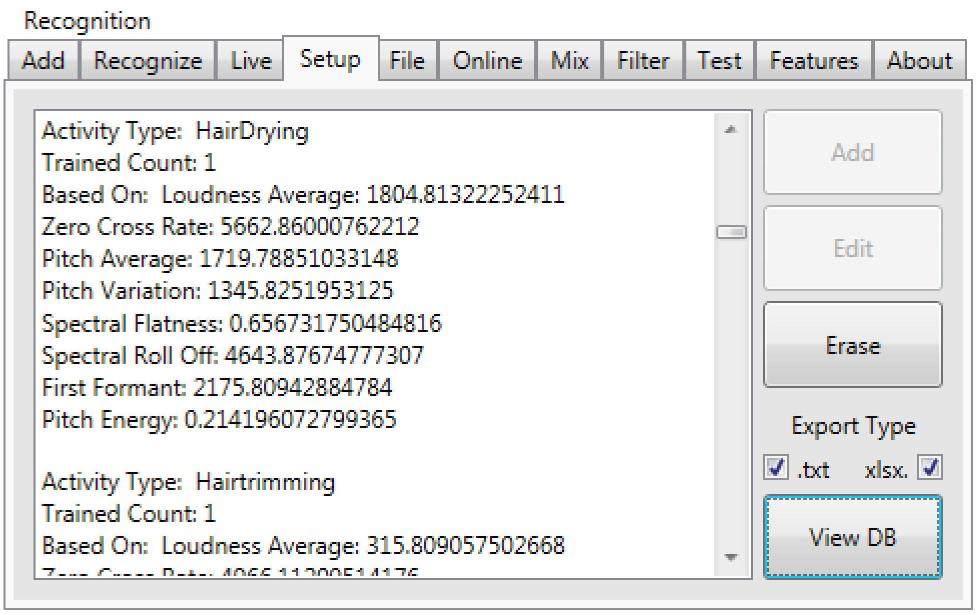
New Activity	Activity
Existing Activity	Toothbrushing
Activity Zone	

✓ Device Oral-B ✓ Electrical ☐ Hidden

Figure 5.18: *Snapshot of the process of labeling an activity after making a record and computing its features. The user first chooses the activity type from the provided list of activities or he defines it himself. Then he enters further information, whether there is a device used to perform this activity and enters information about the device.*

In the Frameworks user interface after recording some activity, there is given the possibility to label it with its information details such as used device to perform it (see Figure 5.18). To simplify the addition process the user can choose between listed activities.

After gathering different activity sounds and extracting their features, it makes sense to have them organized in some way (see Figure 5.19). In Speech Recognition there are collections of spoken texts and their transcription, usually called Speech Corpus. Similarly we gather labeled recordings together with extracted features, in a corpus called Activity Database. It stores all activities with their corresponding fields and stats in a single file, which is synchronized after changes. The file format is either raw text file or Excel Worksheet. In order to meet the Privacy criteria from Subchapter 4.6 one might consider encrypting those when it comes to eventual deployment.



```
Activity Type:  HairDrying
Trained Count: 1
Based On:  Loudness Average: 1804.81322252411
Zero Cross Rate: 5662.86000762212
Pitch Average: 1719.78851033148
Pitch Variation: 1345.8251953125
Spectral Flatness: 0.656731750484816
Spectral Roll Off: 4643.87674777307
First Formant: 2175.80942884784
Pitch Energy: 0.214196072799365

Activity Type:  Hairtrimming
Trained Count: 1
Based On:  Loudness Average: 315.809057502668
```

Figure 5.19: *Snapshot of the setup tab, where one can browse current activities and Add/Edit/Erase the selected (only single selection possible). Further setup capabilities include choosing the database storage type between raw text file or Microsoft Excel file, as well as the possibility to open the storage with the corresponding default software.*

In our frameworks GUI we provide database access with basic functions. Such include erasing activities, which are not performed anymore the way they were trained. While other advanced operations, such as editing the activities by altering their feature values, or adding activities with providing exact feature values, remain only for functionality purposes and are thus not enabled in run mode. One can also choose to view the database in a convenient way according to the selected export type, which is very handy for applying analytical operations on it.

5.5.2. Automatic Feature Selection and Priority Determination

We implement Automatic Feature Selection and Feature Priority Determination in a similar way. However they both have different usage scope – Feature Priorities are used only by the Most Confident Algorithm, discussed in Section 5.5.4.

In our implementation we iterate through the features check whether they remain static for different activities, thus making them unimportant for categorizing a concrete set of activities (see Algorithm 1).

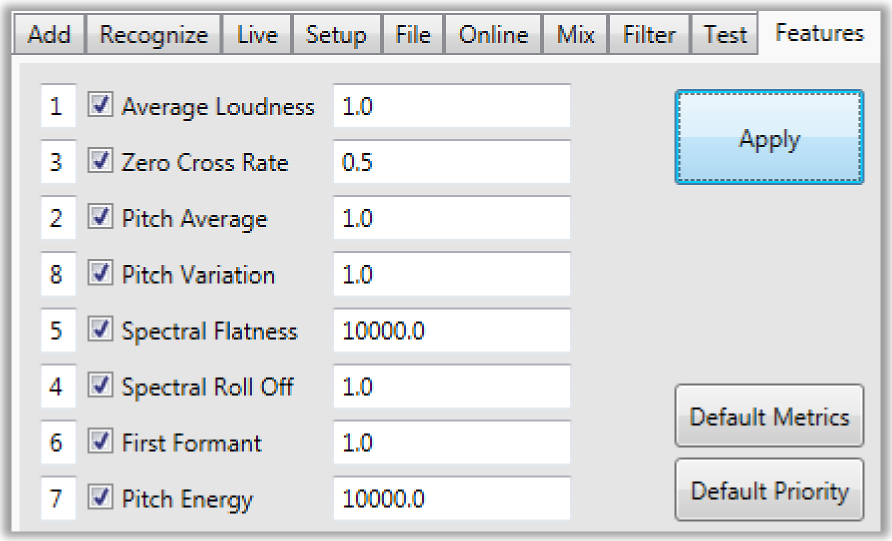
```
FeatureSelection (SpanThresholdCoefficient STC, DistanceThreshold DT)

Enable all features

For each Feature F

      Find the largest distance LD of F between all activities

      Calculate and store |LD – DT| in FC

      if (FC < STC)

            Disable F


if all features are disabled

      Set the feature F with largest corresponding FC

      if there are multiple such features

            Choose the one with less computation steps


end
```

Algorithm 1: *Pseudo code of Automatic Feature Selection*

Another way to implement feature selection is to look for features, which vary a lot between different records of same activity, thus providing erroneous information. However, for this case one needs bigger training data for each activity type, which is the reason why we stick to the first variation of feature selection.

5.5.3. Feature Metrics and Normalization

For manually implemented Machine Learning algorithms it is important to define mutual metrics for all the features in order to measure them. Another concern is that by converting features to common metrics, one also associates different importance or weight with the choice of the conversion coefficient. For our choice of metrics we use the knowledge that all our sound features share the same source. The latter provides information about the connection between the features, which can be obtained by studying their derivation method and their role in defining the sound. In the GUI one can define his own set of metrics different to the displayed default one like shown in Figure 5.20.



1	✓ Average Loudness	1.0
3	✓ Zero Cross Rate	0.5
2	✓ Pitch Average	1.0
8	✓ Pitch Variation	1.0
5	✓ Spectral Flatness	10000.0
4	✓ Spectral Roll Off	1.0
6	✓ First Formant	1.0
7	✓ Pitch Energy	10000.0

Figure 5.20: *Screenshot of the metrics tab with the metrics default values and the priorities default values*

One should note that priority is used only for the Most Confident Algorithm described in next section. It is about the priority of the feature alone, since we are aware of the possibility that combination of features with lower priorities might perform better than combination of same size consisting of features with higher priorities.

5.5.4. Implemented Classifying Algorithms

After introducing our activity database and the different features with their common metric, in this section we describe the two implemented classifying algorithms, both with intension of high efficiency recognition.

Nearest neighbor

As we first described in Section 2.3.2, we implement the Nearest Neighbor algorithm with single reference search. E.g., either having only one reference in the training set for some class, or averaging multiple references to obtain single reference value. We do this in order to enable rapid search for result, knowing that the number of classes is relatively small and relying on the feature choice to be suitable.

With a single reference set a possible behavior for false recognition might be to replace the activity stats with the new one, or to replace only those selected for the recognition, which appeared false. One can use the feedback as training data too and compute an average value for the reference values, or cleanup the largest deviations from an average value.

One-Dimensional

In the one dimensional case of Nearest Neighbor we look for the activity with feature statistics having smallest distance from the measured reference feature (see Algorithm 2).

```
NearestNeighbor1D (Feature F)

Create a sorted list L of activities according to their distance in statistics
from reference F.

return L      // first element of L is the nearest neighbor
```

Algorithm 2: *Pseudo code of one-dimensional Nearest Neighbor, which returns a sorted list of activities according to their distance in given feature from a measured reference*

In the runtime computation we exclude the runtime for returning a full sorted list neighbors according to their distance, because it is done only for debugging reasons. So the total runtime for the one-dimensional Nearest Neighbor is $O(n)$ for n number of activities. The latter could be optimized to $O(logn)$, with utilizing binary search and keeping the statistics sorted.

Multi-Dimensional

In the multi-dimensional Nearest Neighbor we run single dimensional search for all features according to common metrics (see Algorithm 3) (see Appendix C for a sample implementation).

```
NearestNeighborMD (Statistics S)

Convert all features in S to unified metrics and sum them in S'

Create an empty sorted list SL of activities according to distance from S

for each activity A

    Convert all features from A into unified metrics and sum them in F'

    Compute the distance D between F' and the features from S'

    Add D and A into SL

return SL    // first element of SL is the nearest neighbor
```

Algorithm 3: *Pseudo code of multidimensional Nearest Neighbor algorithm*

The runtime of multi-dimensional Nearest Neighbor $O(nm)$ for n number of activities and m number of enabled features.

Most Confident

As we will later see in the evaluation, some features are more trustworthy than others. In order to exploit this property, we developed further optimization of the Nearest Neighbor algorithm, which breaks it and returns an intermediate result according to applied confidence criteria. We call this type of algorithm Most Confident[33]. We considered a couple of possible implementations of such algorithm. The first one is to return intermediate result of NNMD after being confident enough. The second one is to

[33] Not to be confused with some name conventions, which use the word confidence for probability

return the NN1D according to only one prioritized confident feature. We chose the second variation, since the first one requires knowledge not only about the different features alone, but also about their combination. With deeper knowledge about the feature and their behavior in combinations with other features, one might implement a combination of the both mentioned variations.

In detailed view (see Algorithm 4), the implemented Most Confident algorithm runs a sequence of one-dimensional Nearest Neighbor algorithm. After each run it computes relative confidence check, whether the intermediate result is trustworthy. In case none of the intermediate results met the criteria, it returns simply the multi-dimensional case of the Nearest Neighbor algorithm built up with the intermediate computations.

```
MostConfident (Statistics S, ConfidenceThreshold T)

Sort the list of features from S according to their priority in LF

while (LF is not empty)

        Take first feature F from LF

        Run NearestNeighbor1D(F) and store recocnition result in RR

        // measure the relation of distances between F
        // and its two Nearest Neighbors (or first two elements of RR)
        Run CalculateConfidence(F, RR) and store confidence in C

        if (C > T)

                return first element of RR       // e.g. current result



// none of the features passed the confidence test
return multi-dimensional Nearest Neighbor computed using all stored RR
```

Algorithm 4: *Pseudo code of Most Confident algorithm which iterates through the different feature types according to their priorities and returns a result if being confident enough. If not the case it eventually returns the multidimensional version of Nearest Neighbor.*

This algorithm makes it possible to compute fast results in runtime of $O(n)$ for best case scenario of positive confidence check, where n is the number of activities. For taking average values the runtime will be $O(nm)$, where m is the maximum count of stats for some activity (taking the average of those would be $O(m)$, thus the total runtime of $O(nm)$). Worst case will be as the Multi-Dimensional Nearest Neighbor, which is $O(lnm)$, where l is the number of selected features.

One should note that there are further optimizations possible with intelligent choice of the data structures. Such include pre-computation of the average statistics during training to get rid of the m factor. With keeping the data sorted, one can reduce the n

factor to $logn$. With even a bigger redundancy, consisting of sorting the training data for every set of features, one can get rid of l factor too. Overall, we provide a possibility of worst case total result search time of $O(logn)$ and best case with direct search hit of $O(1)$.

5.5.5. Imported External Libraries

In both of the implemented algorithms we exploited the setup to boost the runtime of result search. However, there are numerous interesting Machine Learning tasks like building an incremental learning system, which smartly uses large sets of training data. The latter can be used to build up distribution models, out of which one can make precise predictions about some unknown input and its probability to be some of the known classes (recall Figure 5.17). For such advanced purposes we import Infer.NET (Minka, Winn, Guiver, & Knowles, 2012), a state of the art Machine Learning library. Further reason to include external library is to test the efficiency of the implemented algorithms.

Infer.NET

The Infer.NET library implements the Bayes Point Machine algorithm (Herbrich, Graepel, & Campbell, 2001)(introduced Section 4.4.) in a standard Supervised Learning setting and is trained via Expectation Propagation (Minka T. P., 2001).

Infer.NET works by compiling a model definition into the source code needed to compute a set of inference queries on the model (see Figure 5.21), therefore providing numerous integration advantages over other established libraries, such as those mentioned in Section 4.4.2. WEKA or Accord.NET, including runtime optimization, detailed debugging, code transparency, model exporting.

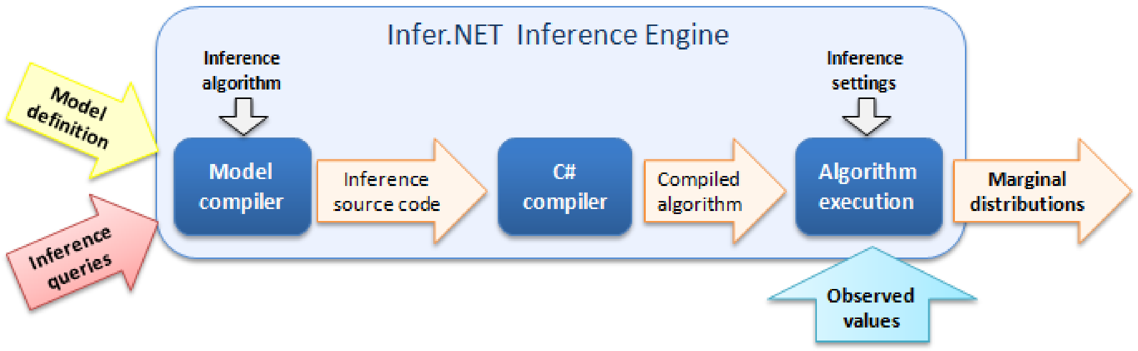


Figure 5.21: *Diagram summarizing the inference process of Infer.NET (Minka, Winn, Guiver, & Knowles, 2012)*

In our multi-class setting for device classification, every device class has an associated weight vector with standard Gaussian vector priors. The device class with a corresponding feature vector is defined by the arg-max of its score under each class. The score is defined as the inner product between the features of the data point and the weight vector plus some added noise. The factor graph corresponding to Bayes Point machine is shown in Figure 5.22.

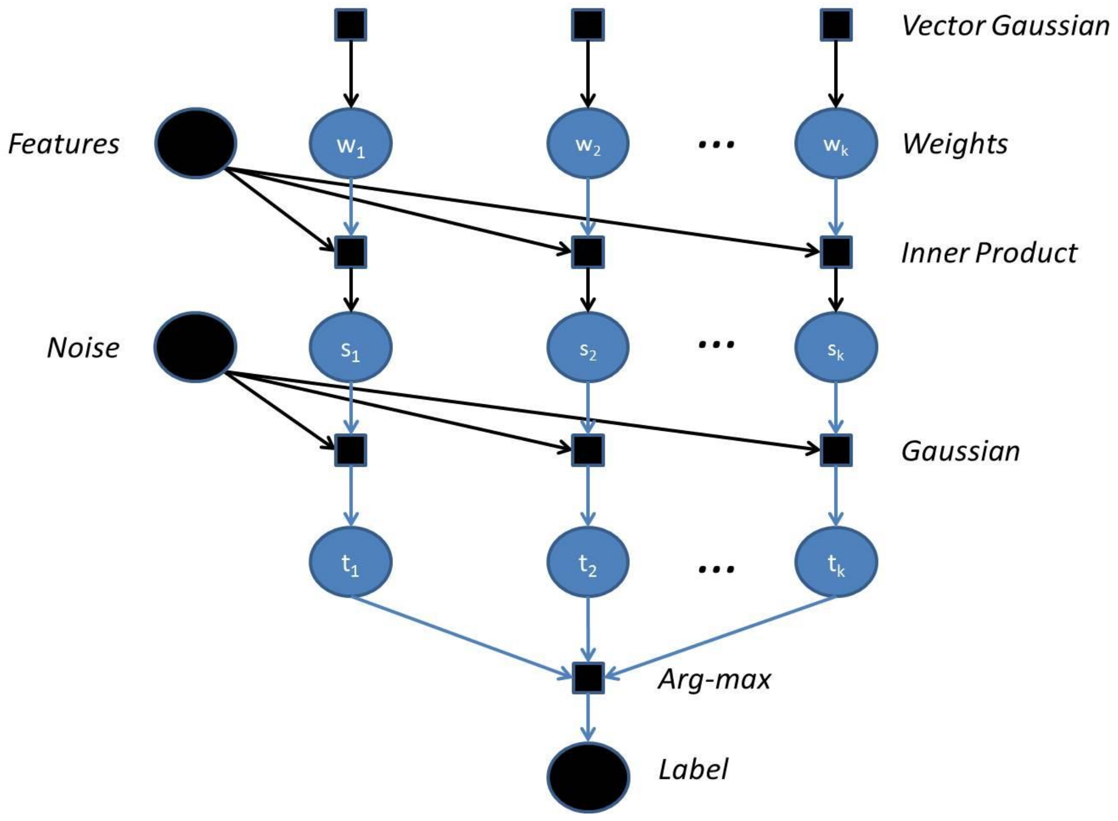


Figure 5.22: *Illustration of the multi-class classification with Bayes Point Machines (Minka, Winn, Guiver, & Knowles, 2012)*

Furthermore, Infer.NET is a perfect choice to satisfy the Adaptivity criteria described in Section 4.4.2. To build an incremental learning system we connect the feedback from the user and use it to obtain posterior distribution of learned data.

We should note that the integration of several classifiers has the potential of interesting opportunities for their usage. Besides combining features for recognition one might also consider combining the Machine Learning algorithms as well. According to (Provost & Fawcett, 2001), a wise choice of classifiers beats the recognition rate of any of them separately.

5.6. Process from Activity to its Device Recognition

After getting familiar with the Sound-based Device Recognition Framework and its inner mechanisms, in this subchapter we provide a revision of the process from activity to its device recognition introduced in Chapter 4 (see Figure 5.23).

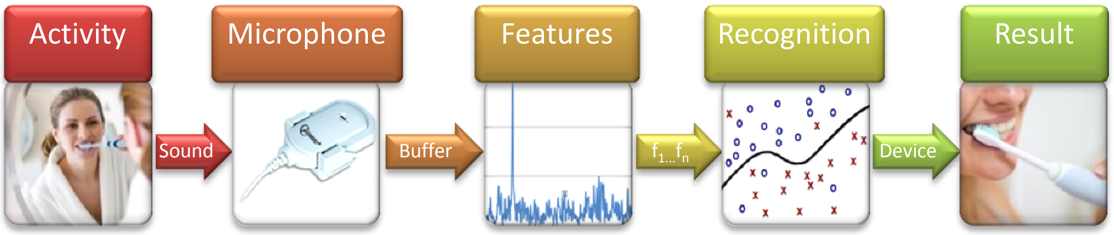
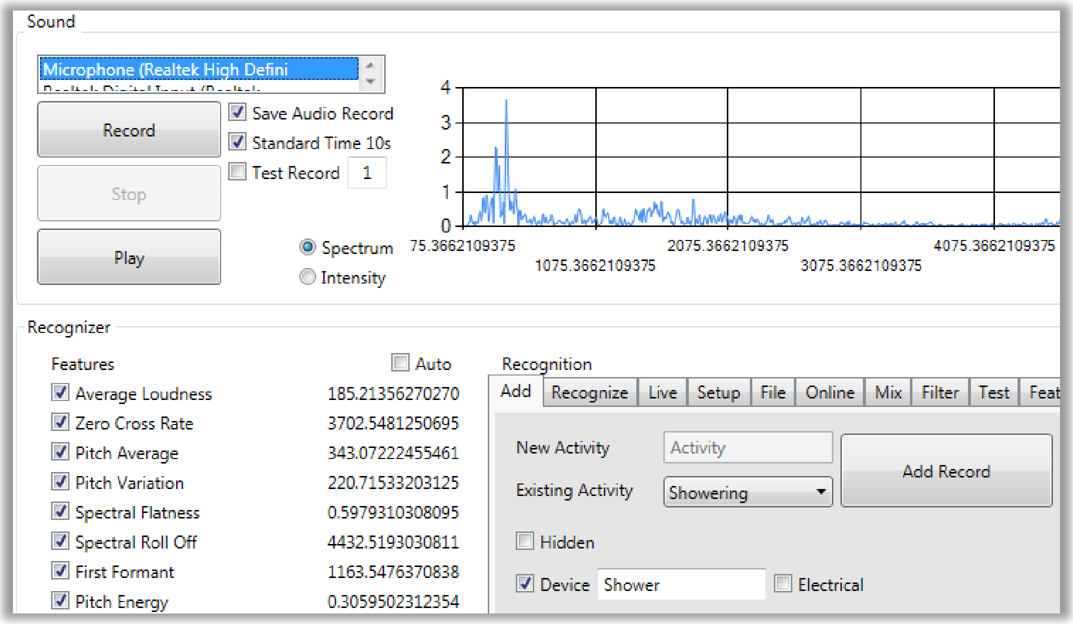


Figure 5.23: *Revision of the process from occurring activity to its device recognition based processing and classification of captured acoustic signals*

For this purpose the user first has to train the recognizer (Section 5.6.1.) and then we can use it for static recognition (Section 5.6.2), as well as for real-time recognition (Section 5.6.3.). At the end of this subchapter in Section 5.6.4., we explain also the interface for integration, which satisfies the concept from Subchapter 4.5.

5.6.1. Training

In the training phase the user labels his recorded activities (See Figure 5.24). This is a form of manual event detection, meaning that the user identifies his activity timing alone and provides the information to the recognizer.


Sound

Microphone (Realtek High Defini

Record | Stop | Play

☑ Save Audio Record
☑ Standard Time 10s
☐ Test Record 1

◉ Spectrum
○ Intensity

Recognizer

Features ☐ Auto

☑ Average Loudness	185.21356270270
☑ Zero Cross Rate	3702.5481250695
☑ Pitch Average	343.07222455461
☑ Pitch Variation	220.71533203125
☑ Spectral Flatness	0.5979310308095
☑ Spectral Roll Off	4432.5193030811
☑ First Formant	1163.5476370838
☑ Pitch Energy	0.3059502312354

Recognition

Add | Recognize | Live | Setup | File | Online | Mix | Filter | Test | Feat

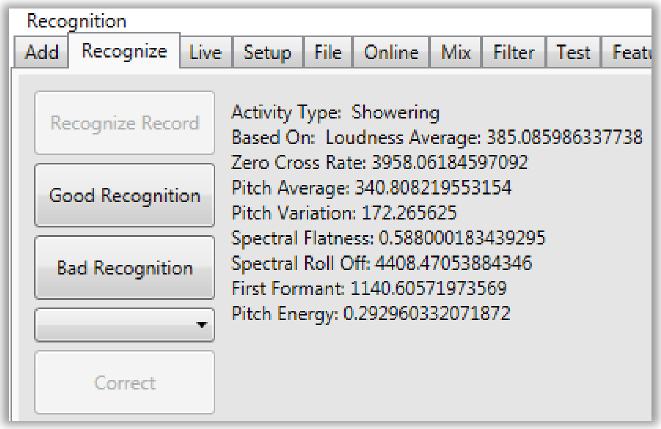
New Activity: Activity
Existing Activity: Showering
Add Record
☐ Hidden
☑ Device: Shower ☐ Electrical

Figure 5.24: *In training phase the user makes records and labels them*

If the added record already exist in the database, the recognizer uses it as further training information for the selected class.

5.6.2. Recognition

In the recognition phase the user asks the recognizer to make a guess about some recording as shown in Figure 5.25. After attempting recognition, the user can provide feedback to the recognizer and correct him if he decides to do so.

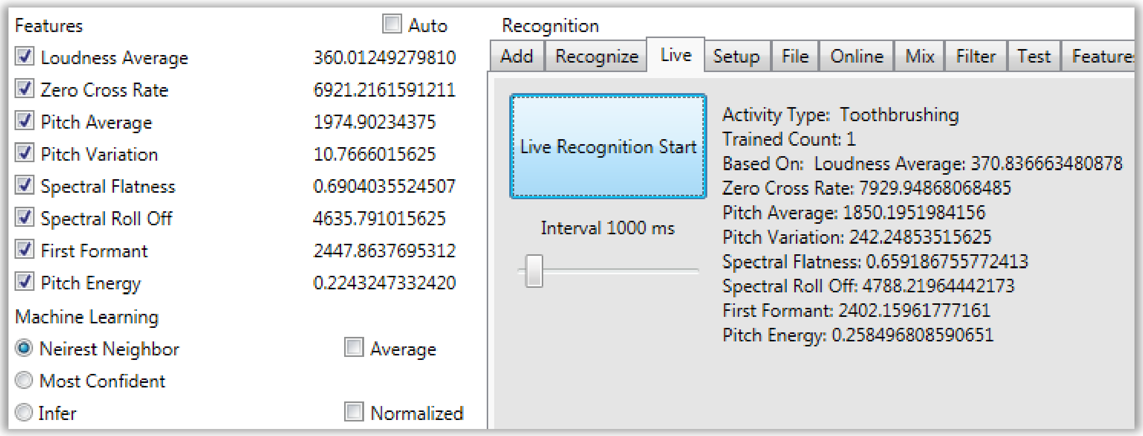


Activity Type: Showering
Based On: Loudness Average: 385.085986337738
Zero Cross Rate: 3958.06184597092
Pitch Average: 340.808219553154
Pitch Variation: 172.265625
Spectral Flatness: 0.588000183439295
Spectral Roll Off: 4408.47053884346
First Formant: 1140.60571973569
Pitch Energy: 0.292960332071872

Figure 5.25: *Snapshot of a recognition attempt. We can see on the label right the recognition result, and upon what features it is based. The user is given a chance to provide a feedback whether the recognition was good. If the recognition was bad, the user may also correct the recognizer.*

5.6.3. Real-Time Recognition

Real-time recognition is an important characteristic of every pattern recognition system (see Figure 5.26). Here we have a flexible system for real-time recognition, which enables changing the feature set for recognition with the Machine Learning algorithm while running. The recognition intervals are between 0.1 s and 10 s. Note, that the training data is usually recorded with 10 s intervals as mentioned. Furthermore, for time intervals below one second pitch span feature is automatically disabled, because of the impossibility to gather sufficient and reliable temporal information of this type in such small periods of time.



Features ☐ Auto

✓ Loudness Average	360.01249279810
✓ Zero Cross Rate	6921.2161591211
✓ Pitch Average	1974.90234375
✓ Pitch Variation	10.7666015625
✓ Spectral Flatness	0.6904035524507
✓ Spectral Roll Off	4635.791015625
✓ First Formant	2447.8637695312
✓ Pitch Energy	0.2243247332420

Machine Learning
[✓ Neirest Neighbor] ☐ Average
Most Confident
Infer ☐ Normalized

Interval 1000 ms

Activity Type: Toothbrushing
Trained Count: 1
Based On: Loudness Average: 370.836663480878
Zero Cross Rate: 7929.94868068485
Pitch Average: 1850.1951984156
Pitch Variation: 242.24853515625
Spectral Flatness: 0.659186755772413
Spectral Roll Off: 4788.21964442173
First Formant: 2402.15961777161
Pitch Energy: 0.258496808590651

Figure 5.26: *Live Recognition mode enabling feature and Machine Leaning algorithm selection, during a real-time recognition with time intervals between 0.5 s and 10 s.*

Viewed in depth, the real-time recognition is a non-stop interaction between the two main components of the software system – Sound Unit and Machine Learning Unit. The first one constantly buffering the incoming input from the environment and extracting different features, while the second one periodically pulls the current state of the features and issues recognition (see Figure 5.27).

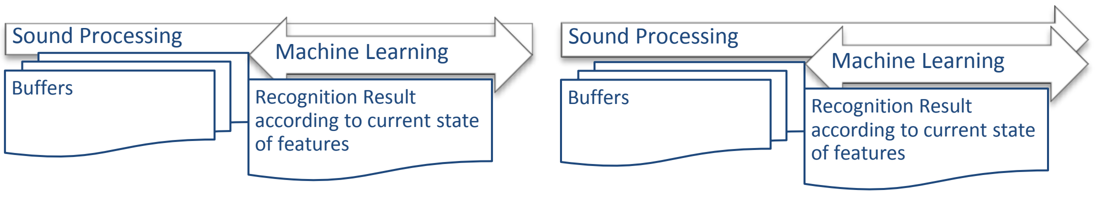

Sound Processing
Machine Learning
Buffers
Recognition Result according to current state of features
Sound Processing
Machine Learning
Buffers
Recognition Result according to current state of features

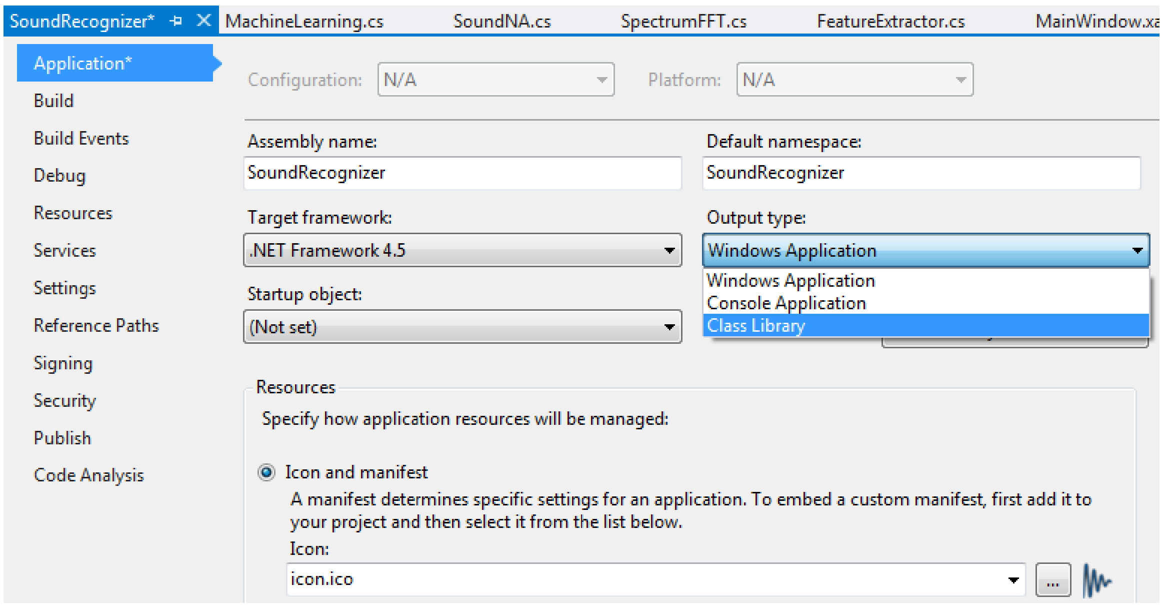
Figure 5.27: *Live recognition dataflow diagram showing the interaction between the Sound Processing unit, which is constantly buffering the recorded sound and extracts features, while the Machine Learning component starts recognition after some point of time*

5.6.4. Integration Interface

Here we describe our integration interface, which satisfies the described one in Subchapter 4.5. We first provide the integration possibilities as Class Library, then as a Running Service.

Class Library

In order to integrate the project to class library, one has to change its output type to a class library as shown in Figure 5.28.



Configuration: N/A
Platform: N/A
Assembly name: SoundRecognizer
Default namespace: SoundRecognizer
Target framework: .NET Framework 4.5
Output type: Windows Application
Startup object: (Not set)
Resources
Specify how application resources will be managed:
Icon and manifest
A manifest determines specific settings for an application. To embed a custom manifest, first add it to your project and then select it from the list below.
Icon:
icon.ico

Figure 5.28: *Changing the output type of the project to class library for integration*

After compiling to class library one can access the whole framework as a complete solution, as well as both of its namespaces for Sound Processing and Machine Learning separately (see Figure 5.29).

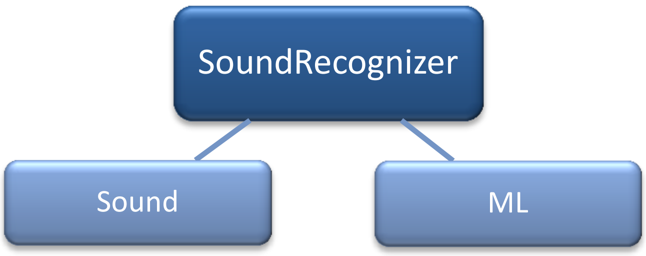


Figure 5.29: *Illustration of best exporting opportunities of the Framework and its namespaces – Sound Namespace for Sound related operations and ML Namespace for Machine Learning related operations.*

Service

Since the described Sound-based Device Recognition Framework encapsulates all the logic for Sound Processing and Machine Learning, we rely on the client providing sound recordings via TCP/IP a connection[34]. So the first step is to establish connection, after which the GUI acting as a server starts to listen for commands. There are two commands, which the user has to choose – one for training and one for recognition. They are defined as follows:

```
➔ Train <Activity, WAVE File>
← OK                // Confirmation


➔ Recognize <WAVE File>
← Activity          // Recognition Result
```

One should note that according to the definition of those commands, obtaining a real-time recognition requires a connection of at least 176400 bytes per second for the wave data plus the additional bytes for the commands and the WAVE File headers (recall Figure 2.8). One can loosen up such connection requirements if he considers a mixed integration. For example, by integrating the Sound Class into a device in order to send only the features to the Framework for recognition.

[34] Group of communications protocols used for the Internet and similar networks

Chapter 6

Evaluation

In the evaluation of the Sound-based Device Recognition Framework, we evaluate all important aspects of the developed system, such as its recognition accuracy, automatic mixing and automatic feature selection. We concentrate on the functionality of the framework at its current development stage, and show that our system provides a solid base for further developments. In Subchapter 6.1., we define our test environment and its setup, as well as our test corpus and software test module. Then, in Subchapter 6.2., we present our experimental results.

6.1. Test Setup

In our tests, we cover a large variety of devices that produce different noise in a home environment. In Section 6.1.1 we introduce these devices and we specify our hardware setup. In Section 6.1.2 we present our software module for automatic testing.

6.1.1. Test Environment Setup and Corpus

We perform our test in a home environment. We select a 25 class problem consisting of 20 devices, three complex activities with two devices, speaking and silence. In comparison, most of the related works, with similar test environment to ours from Subchapter 3.4., are usually dealing with classes of a size between 10 and 15 entities. Similarly to other works, we perform 6 records per device and use one record for reference, while the other 5 are for tests.

Zone-wise our environment can be separated into three activity zones – zone for personal hygiene, zone for preparing meals and zone for relaxing and socializing (see Figure 6.1). Zones can be separated either in separate rooms or locations in one big room. Furthermore, almost all tested activities are performed at their corresponding place, excluding vacuum cleaning (moving around the room), hedge cutting (also at different locations). Devices, which had strongly bounded locations were Fridge, Coffee Maker (Moka Pot), Kettle, Shaker (Blender), Washing Machine, Toilet, Absorber, Microwave, Heater and Music Centre.

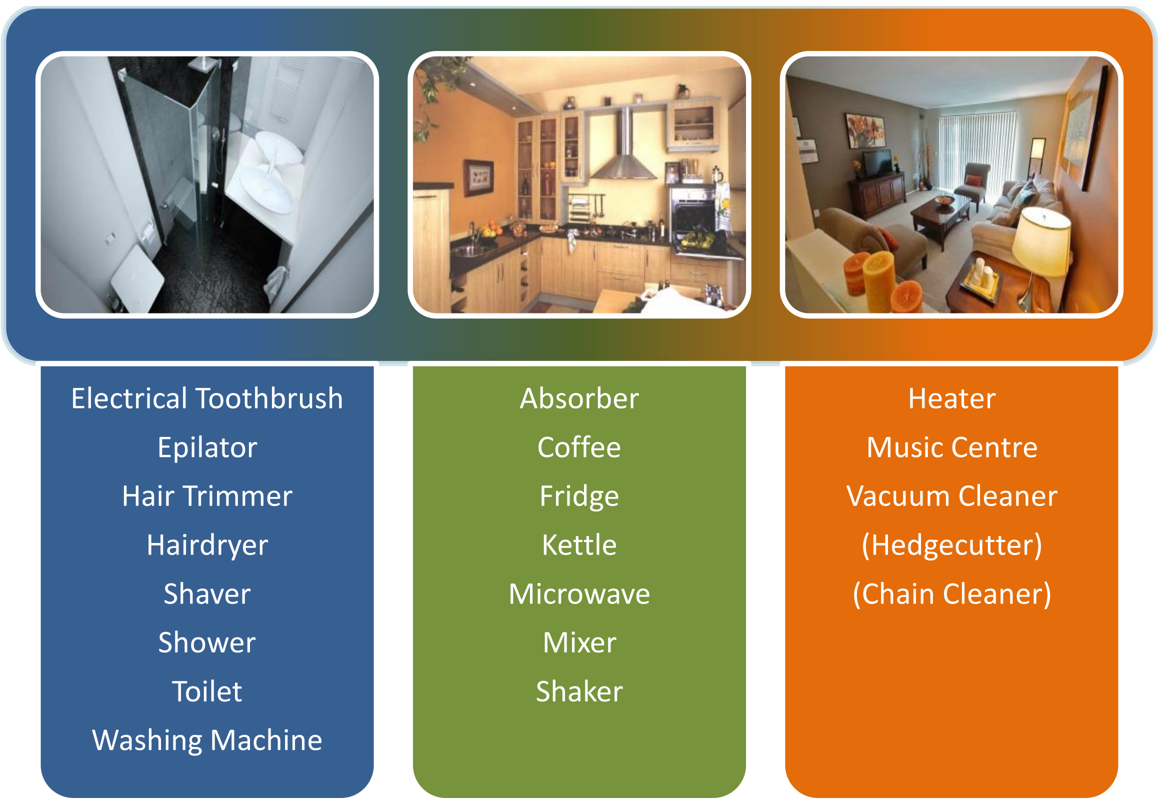


Figure 6.1: *Visualization of our simulated environment with three activity zones and their corresponding devices. Hedgecutter is in brackets, because it occurred outside, despite being captured indoors. Similar to that the Chain Cleaning occurred in the corridor.*

Besides the conventional devices used in a home, we tried to make the recognition problem harder by adding a Hedgecutting activity, which occurred outside of our test environment, but still was captured indoors. We also included a Chain Cleaning activity, in order to increase the number of non-electrical devices and the variety of the devices.

Mixed1 is a complex activity of simultaneous Showering and Toothbrushing with usage of respective Shower and electrical Toothbrush. Mixed2 is cleaning while blending at same time, with Vacuum Cleaner and Blender. Mixed3 is actually performed by a person cutting hedges from outside with a Hedgecutter and by a person trimming hair with Hairtrimmer.

The only activity without device was the speaking activity, which should be correctly identified for future developments of the framework, where it might include Speech Recognition components. In parallel, the music listening activity can be used to recognize the songs listened to with an external library, besides providing information about the usage of a HiFi system. Furthermore we added a silence entity despite being neither device nor activity itself, due to the fact that with an average microphone, like the one we used, the recordings of Fridge and Shaving were almost the same as those of silence in terms of their volume levels and spectrum shape. Furthermore in integration of the system into a bigger recognition system recognizing silence as a separate entity might present some vital information, which might be crucial for differentiating some entities, like mentioned in Section 2.2.4.

For our recordings we use a single microphone. We make 10 second mono records at 44100 Hz rate and use FFT Buffer size of 4096 samples. It is also important, that the sounds are actually recorded from activity that really occurs. E.g. no playback recordings were indexed. In out tests, we made all records with manual event detection using our framework.

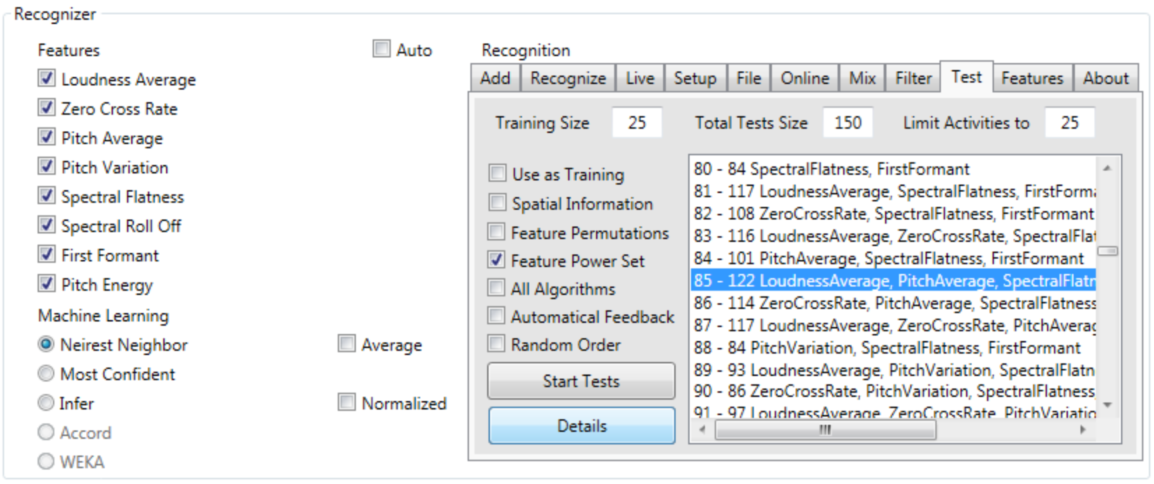
6.1.2. Software Test Module

In our software module for automatic testing we provide different opportunities to test various aspects of the system (See Figure 6.2), including:

- Test all combinations of features for a given test corpus via computing their power set. This provides important information about the best feature combination, as well as the worst. Furthermore, it can be used to identify the best performing derivation method.
- Test different permutations for given number feature set size in order to find out the best performing. Indeed, this is a subset of the feature power set of all features, but sometimes with larger test corpus it takes a lot of time to compute the power set of all features.
- Test different provided algorithms for finding the best algorithm.
- Provide automatic feedback, as if the user would. This is an option, which makes bin sense with less training data, where the single reference is vital.
- Provide spatial information to the recognizer, as if there are multiple microphones installed, or tracking.
- After tests the test module cleans all information from the Activity Database, but there is an option not to do so. Such an option is important to see how real-time recognition performs.
- If none of the options is selected, the test module runs the tests with the selected features and algorithm, as if the user ran them.

It is also worth noting, that before each test, besides setting up the Machine Learning parameter, one can change Sound Processing settings as well.

We perform our experiments in two steps. First we identify the feature set with the highest recognition accuracy. Second, we run detailed tests with the set in order to study in details its obtained results.



Features

[✓ Loudness Average]
[✓ Zero Cross Rate]
[✓ Pitch Average]
[✓ Pitch Variation]
[✓ Spectral Flatness]
[✓ Spectral Roll Off]
[✓ First Formant]
[✓ Pitch Energy]

Auto

Machine Learning

[✓ Neirest Neighbor]
Most Confident
Infer
Accord
WEKA

Average
Normalized

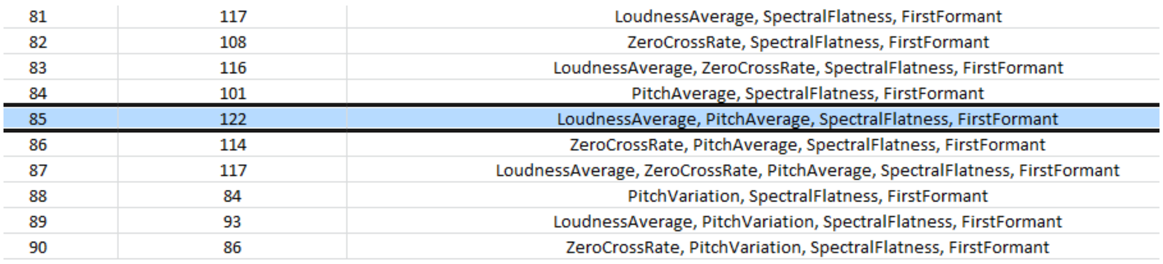
Training Size 25 Total Tests Size 150 Limit Activities to 25

Use as Training
Spatial Information
Feature Permutations
[✓ Feature Power Set]
All Algorithms
Automatical Feedback
Random Order

80 - 84 SpectralFlatness, FirstFormant
81 - 117 LoudnessAverage, SpectralFlatness, FirstForm
82 - 108 ZeroCrossRate, SpectralFlatness, FirstFormant
83 - 116 LoudnessAverage, ZeroCrossRate, SpectralFla
84 - 101 PitchAverage, SpectralFlatness, FirstFormant
85 - 122 LoudnessAverage, PitchAverage, SpectralFlatn
86 - 114 ZeroCrossRate, PitchAverage, SpectralFlatness
87 - 117 LoudnessAverage, ZeroCrossRate, PitchAvera
88 - 84 PitchVariation, SpectralFlatness, FirstFormant
89 - 93 LoudnessAverage, PitchVariation, SpectralFlatn
90 - 86 ZeroCrossRate, PitchVariation, SpectralFlatness
91 - 97 LoudnessAverage, ZeroCrossRate, PitchVariatio

Figure 6.2: *Test for power set of all features to identify the best combination. In the first column of the list we see the feature combination in a decimal notation, while its selected bits represent the enabled features. In the second column we see the number of correct recognitions. In the current snapshot we see Total Tests Size of 150 and Training Size of 25, meaning 125 performed tests. So 122 hits means recognition accuracy of 97.6%.*

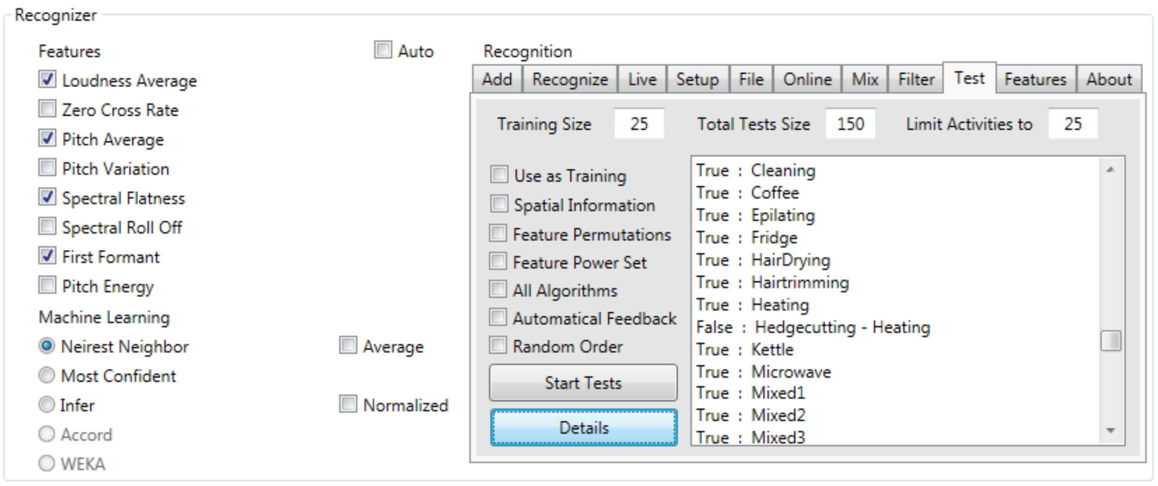
As mentioned, in our current tests we first test power set of all features to determine the best performing set (see Figures 6.2 and 6.3). If some features have the same recognition accuracy then we choose the minimized set of features for best result. In case of a conflict for the number of selected features, we choose those with minimal costs similarly to (Bolón-Canedo, Porto-Díaz, Sánchez-Maroño, & Alonso-Betanzos, 2014).



81	117	LoudnessAverage, SpectralFlatness, FirstFormant
82	108	ZeroCrossRate, SpectralFlatness, FirstFormant
83	116	LoudnessAverage, ZeroCrossRate, SpectralFlatness, FirstFormant
84	101	PitchAverage, SpectralFlatness, FirstFormant
85	122	LoudnessAverage, PitchAverage, SpectralFlatness, FirstFormant
86	114	ZeroCrossRate, PitchAverage, SpectralFlatness, FirstFormant
87	117	LoudnessAverage, ZeroCrossRate, PitchAverage, SpectralFlatness, FirstFormant
88	84	PitchVariation, SpectralFlatness, FirstFormant
89	93	LoudnessAverage, PitchVariation, SpectralFlatness, FirstFormant
90	86	ZeroCrossRate, PitchVariation, SpectralFlatness, FirstFormant

Figure 6.3: *Detailed view of the tests after being performed for further statistical operations.*

After identifying the best performing feature set, we run specific tests with it in order to obtain detailed information about each recognition attempt (see Figure 6.4).


Recognizer

Features

- [✓] Loudness Average
- [ ] Zero Cross Rate
- [✓] Pitch Average
- [ ] Pitch Variation
- [✓] Spectral Flatness
- [ ] Spectral Roll Off
- [✓] First Formant
- [ ] Pitch Energy

[ ] Auto

Machine Learning

- [✓ Neirest Neighbor]
- Most Confident
- Infer
- Accord
- WEKA

[ ] Average

[ ] Normalized

Recognition

Training Size 25 Total Tests Size 150 Limit Activities to 25

- [ ] Use as Training
- [ ] Spatial Information
- [ ] Feature Permutations
- [ ] Feature Power Set
- [ ] All Algorithms
- [ ] Automatical Feedback
- [ ] Random Order

```
True  : Cleaning
True  : Coffee
True  : Epilating
True  : Fridge
True  : HairDrying
True  : Hairtrimming
True  : Heating
False : Hedgecutting - Heating
True  : Kettle
True  : Microwave
True  : Mixed1
True  : Mixed2
True  : Mixed3
```

Figure 6.4: *Test environment snapshot while performing concrete test*

Again, we can see details about the recognition in an automatically generated sheet shown in Figure 6.5.

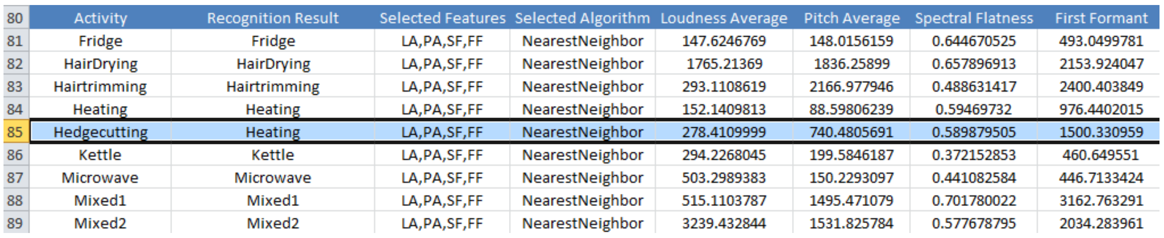

80	Activity	Recognition Result	Selected Features	Selected Algorithm	Loudness Average	Pitch Average	Spectral Flatness	First Formant
81	Fridge	Fridge	LA,PA,SF,FF	NearestNeighbor	147.6246769	148.0156159	0.644670525	493.0499781
82	HairDrying	HairDrying	LA,PA,SF,FF	NearestNeighbor	1765.21369	1836.25899	0.657896913	2153.924047
83	Hairtrimming	Hairtrimming	LA,PA,SF,FF	NearestNeighbor	293.1108619	2166.977946	0.488631417	2400.403849
84	Heating	Heating	LA,PA,SF,FF	NearestNeighbor	152.1409813	88.59806239	0.59469732	976.4402015
85	Hedgecutting	Heating	LA,PA,SF,FF	NearestNeighbor	278.4109999	740.4805691	0.589879505	1500.330959
86	Kettle	Kettle	LA,PA,SF,FF	NearestNeighbor	294.2268045	199.5846187	0.372152853	460.649551
87	Microwave	Microwave	LA,PA,SF,FF	NearestNeighbor	503.2989383	150.2293097	0.441082584	446.7133424
88	Mixed1	Mixed1	LA,PA,SF,FF	NearestNeighbor	515.1103787	1495.471079	0.701780022	3162.763291
89	Mixed2	Mixed2	LA,PA,SF,FF	NearestNeighbor	3239.432844	1531.825784	0.577678795	2034.283961

Figure 6.5: *Excel Export of the detailed test results for a chosen feature combination. We can see in the first column the actual activity compared to the recognition result in the second column. In the next two columns we see the selected features and the used Machine Learning algorithm for recognition, followed by the values for each selected feature.*

All tests and their results are stored in the database. This includes all intermediate computation steps, as well as the used Sound Processing and Machine Learning parameters. One can use the results on his own for performing further evaluations with Excel or Matlab. Such evaluations and visualizations are those which follow in the next sub chapters with concrete tests. Other evaluations, which are not of particular interest in this chapter`s performance evaluation, but represent interesting input for solving the introduced use cases in Subchapter 1.2., are evaluations of the users schedule, for creating his activity profile, based on different activities which he performs at a different time. One can also import the computed features from the intermediate steps in his own program and tryout different Machine Learning algorithms, or use the features as additional information for his existing recognition software (e.g. skipping to include the Sound-based Device Recognition Framework as library).

6.2. Test Results

In this subchapter we perform our tests which were introduced in Section 6.1.3., with hardware setup and environment explained in Section 6.1.2.

6.2.1. Identifying the Best Feature Set for the Tests

As mentioned in Section 6.1.1., our Test Corpus consists of 25 Class problem, out of which 20 Classes represent devices, 3 Classes are mixes of two devices, and the other two Classes are Speaker and Silence. We run the implemented multi-dimensional Nearest Neighbor algorithm with single training for the task of recognizing 125 activities consisting of 5 occurrences from each of the 25 classes. We test the power set of all features. This means, with our implemented 8 features we test 255 combinations, excluding the empty set. The results of best and worst performing feature combinations together with the average recognition results are shown in Table 6.1.

Feature Count	Best Set	Result	Worst Set	Result	Average Result
1	LA, (ZCR, PA)	52%	FF	30.4%	44.6%
2	LA, SRO	81.6%	SRO, FF	49.6%	67.49%
3	LA, SF, FF	93.6%	PV, FF, PE (PV,SRO,PE)	62.4%	76.69%
4	LA,PA,SF,FF (LA,SF,SRO,FF)	97.6%	PV,SRO,FF,PE	64%	79.97%
5	LA,PA,SF,SRO,FF	97.6%	ZCR,PV,SF,SRO,FF PA,PV,SRO,FF,PE	68.8%	80.66%
6	LA,ZCR,PA,SF,SRO,FF LA,ZCR,SF,SRO,FF,PE LA,PA,SF,SRO,FF,PE	94.4%	ZCR,PA,PV,SF,SRO,FF,PE (ZCR,PA,PV,SRO,FF,PE) (ZCR,PA,PV,SF,SRO,FF)	72%	80.06%
7	LA,ZCR,PA,SF,SRO,FF,PE	93.6%	PA,PV,SF,SRO,FF,PE (ZCR,PA,PV,SRO,FF,PE)	74.4%	79%
8	All Features	77.6%	All Features	77.6%	77.6%
		85.9%		62.4%	73.26%

Table 6.1: *Best, Worst and Average results for all different feature combinations of different set sizes. The results in brackets were up to 1 recognition close to the provided result*

With a single feature for recognition we obtained best results for Loudness Average (LA), directly followed by Zero Crossing Rate (ZCR) and Pitch Average (PA). They were also the first three implemented features. It is interesting, that the combination of those three was nowhere near to matching the performance of the winners in the next two categories.

For feature couple, we anticipate also an interesting result having Spectral Roll Off (SRO) in the best combination as well as in the worst combination. Similar to that the First Formant (FF) is constantly in all best results and worst results together – clear evidence that the combination of features is crucial for recognition, rather than having single strong features, supporting our claims from Section 5.5.4.

We can see also that Loudness Average (LA) performs well and could not be found in any of the worst results. This supports our pre-study of sounds in the theoretical background Chapter 2, where we claimed that electrical devices have a nice property of being bound to some loudness level. Also, doing most of the activities at their specific places also played an important role. For example deciding to regroup all devices and their locations, will have a tremendous effect on the LA feature.

For best recognition set we identified two combinations, which beat the 97% rate and one not far behind – <LA,PA,SF,FF> (97.6%), <LA,PA,SF,SRO,FF> (97.6%), and <LA,SF,SRO,FF> (96.8%). We identify the reason for these exceptional good results being the Sound Processing setup for the environment, as well as most of the devices being tested throughout the development, thus enabling the precise extraction of their characteristics.

Our average results between 4 and 7 feature sizes was about 80%, which is also a same feature count, where the best results peaked. We tested our automatic feature selection algorithm and it chose a set of 6 features to obtain 91.2% recognition accuracy. Thus we conclude that the feature count range between 4 and 7 features is the best performing.

In Figure 6.6 we can see a visualization of the results from Table 6.1. We can see that both best and average cases increase their accuracy for feature count up to 4 and 5, and from that point on we see a declining. Thus we observe that the increasing number of features doesn't necessarily mean better recognition, as mentioned in Section 2.2.4. However if one wants to be on the safe side, one should implement more features, because as we see the worst case recognition rate increases with each new feature addition.

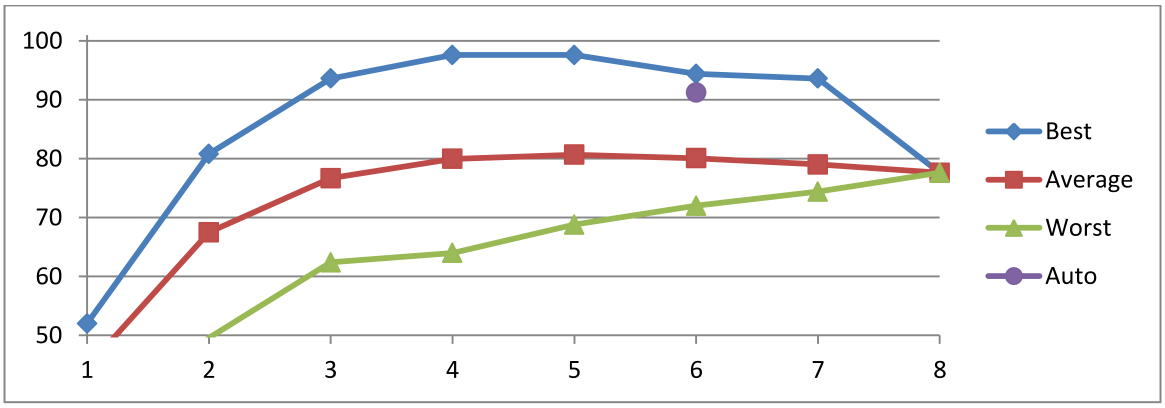


Figure 6.6: *A plot of best (blue) vs. average (green) vs. worst (red) results in terms of the different recognition rates (y-axis) according to the different feature set size (x-axis). The violet point represents the automatic feature selection, which selected 6 features and obtained 91.2% recognition accuracy.*

This evaluation can serve also as a test for correct identification of important features - e.g. threshold for those features and feature selection method justification.

6.2.2. Running Detailed Tests

In this section we provide a confusion table for the best feature sets identified in the last section, as well as a detailed test the of the performance with our automatic selected feature set by supplying the recognizer with special information about the activities. We should also note that the devices and their locations are according to Section 6.1.1.

Testing the Best Feature Set

In Table 6.2 we provide a confusion matrix of the device predictions of the two best performing sets, identified in the previous section, compared with the actual device.

Predicted / Actual Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1 Absorber	5	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2 ChainC	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3 Cleaner	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4 Coffee	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5 Epilator	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6 Fridge	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7 HairD	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8 HairT	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9 Heater	0	0	0	0	0	0	0	0	5	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10 HedgeC	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11 Kettle	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12 MicroW	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0
13 Mixed1	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0
14 Mixed2	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	1	0	0	0	0	0	0	0
15 Mixed3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0
16 Mixer	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0
17 Music	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0
18 Shaker	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0
19 Shaver	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0
20 Shower	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0
21 Silence	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0
22 Speech	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0
23 Toilet	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0
24 ToothB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0
25 Washer	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5

Table 6.2: *Confusion matrix for best results displaying the actual class of some device, and its prediction by the classifier (the two best results have same matrix)*

We can see surprisingly good results for the automatically mixed data for recognition. Also it was surprising that the Hedgecutter was misrecognized three times, while its combination with the Hairtrimmer was all correctly identified. The only misrecognized indoor device was the Shaker. However, it got confused with its combination with the Vacuum Cleaner, thus not being an entirely erroneous recognition.

Testing the Feature Set with Automatic Feature Selection

Our automatic feature selection algorithm chose a 6 feature set consisting of <LA,ZCR,PA,SF,SRO,PE>, which obtained recognition accuracy of 91.2%. Most of the wrong recognitions were of devices, which do not belong to the same Activity Zone according to Section 6.1.1., like the erroneous recognition of Hedgecutter as Absorber. So we conducted an experiment by supplying the recognizer with information about the location of the device. The experiment then achieved a recognition result of 97.6% (see Table 6.3).

Predicted / Actual Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1 Absorber	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2 ChainC	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3 Cleaner	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4 Coffee	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5 Epilator	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6 Fridge	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7 HairD	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8 HairT	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9 Heater	0	0	0	0	0	0	0	0	5	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10 HedgeC	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11 Kettle	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12 MicroW	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0
13 Mixed1	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0
14 Mixed2	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0
15 Mixed3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0
16 Mixer	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	5	0	0	0	0	0	0	0	0	0
17 Music	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0
18 Shaker	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0
19 Shaver	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0
20 Shower	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0
21 Silence	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0
22 Speech	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0
23 Toilet	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0
24 ToothB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0
25 Washer	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5

Table 6.3: *Confusion matrix for the automatic feature selection and its prediction by the classifier with the help of spatial information*

6.2.3. Test of Mixing

In order to test the mixing component we record two activities and their combination, and then we mix automatically the recorded activities and compare them according to the obtained features (see Figure 6.7).

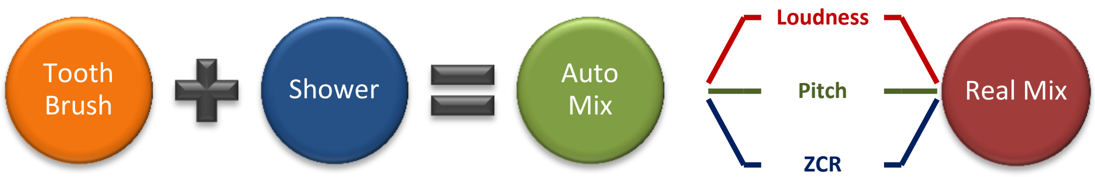


Figure 6.7: *Illustration of the mixing process of two device records and its feature-wise comparison with the real mix*

We use the tests from the previous section to compare 6 records of Toothbrushing, Showering, together with their automatic and real mixes, to illustrate in Figure 6.8 their similarity in terms of the first implemented and most robust single features. For spectrogram comparison of the tests see Appendix E.

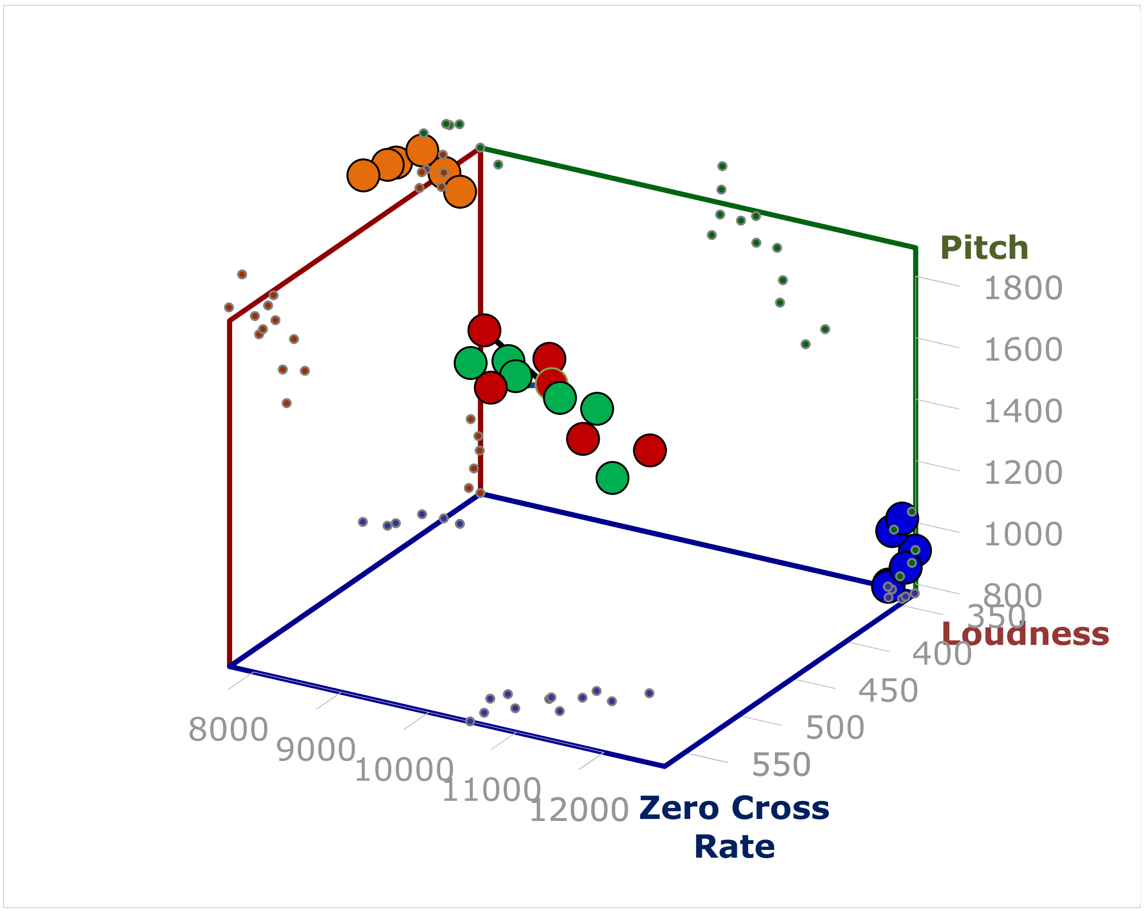


Figure 6.8: *Three-dimensional plot with real (Red) and automatic (Green) mix of Showering (Blue) and Toothbrushing (Orange) according to their of 6 experiment distribution over Loudness, Zero Crossing Rate and Pitch.*

We obtained similar results when mixing other devices, except while mixing Shaker and Vacuum Cleaner, which shared almost same features between the automatic mixing and real mixing, except the average pitch. E.g. in the mixed version the pitch of the shaker was perceived to be stronger, so it dominated and produced similar values throughout the tests. While in the real mixes of both devices we measured average pitch ranging between 586 Hz and 915 Hz with one occurrence of 1531 Hz. Such deviations can be explained by the occurrence of acoustic resonance and seriously affect pitch-based recognition.

6.2.4. Comparison between NNMD and Infer.NET

We also made a comparison between the Infer.NET classifier and our implementation of the multi-dimensional Nearest Neighbor for our test corpus. We selected the full feature set to check, whether Infer.NET can handle better multiple features. Our results showed that for testing with single training, both recognizers achieved the same recognition accuracy. The drawback of Infer.NET was that it ran about 10 times slower, which is understandable, considering the much larger number of computations it has to perform. However, with the increasing number of training data Infer.NET steadily increases its recognition rate, while our optimized implementation had virtually the same recognition rate (either a bit higher or a bit lower). So we anticipate a tradeoff, between runtime and recognition rate, where one might choose the best option for his setup.

Chapter 7

Conclusion & Outlook

In this Thesis we designed and developed the Sound-based Device Recognition Framework for classifying sounds occurring in a home environment. In the development process, we first studied the nature of the sounds produced by different devices. Then we incrementally determined a set of features, which distinguishes them from one another. We extracted these features by applying various Sound Processing techniques. For the final phase of classification, we optimized the Nearest Neighbor algorithm and imported the Bayes Point Machine. After implementing all steps of the full process from activity to recognition with multiple solution choices at each phase, we performed a detailed evaluation of the system by testing its multiple facets. Our tests showed robust recognition results and helped us to identify a sufficient set of features and Machine Learning algorithms to build a robust personalized recognizer. Thus, we consider this stage of development of the Sound-based Device Framework as complete. Furthermore, it provides a solid base for further developments.

7.1. Contributions

Despite its early development stage, there are many contributions in the field of Sound-based Device Recognition provided in this work. The study of specific characteristics of the sounds produced by devices alone has not been investigated in any related work. In contrast, we first performed a manual study of them and then made a full testing of all combinations to identify the best performing set. Most of those characteristics were overseen by the majority of related works for the case of general activity recognition, since they are not applicable for Speech Recognition, which is their conventional research starting point. However, according to our evaluation, combinations of our chosen features are definitely important for classification of activities in a home environment. We also adopted different Machine Learning techniques and optimized a couple of them for our purpose. We also introduced the only system in the field, designed to learn over time using a feedback from the user and to adapt its recognition settings such as automatically choosing its feature set. A further contribution from a software engineering standpoint is the flexible development of the Framework, which incrementally increases its capabilities carrying out a spiral development process.

7.2. Review of Research Questions

Here we review the research questions from Subchapter 1.6., which represent a further contributions of our work.

7.2.1. Could we reduce the complexity of a recognition system, while maintaining the recognition rate?

We have not only shown that we can significantly reduce the complexity of a recognition setup in a personalized setup, but also implemented different ways to do this in both Sound Processing and Machine Learning Classes.

7.2.2. Which are the relevant sound features for the task of device recognition, and can we choose these automatically?

We investigated the feature selection from many different perspectives. We first studied the feature selection manually with sound analyzing software and then we incrementally built the desired set of features in our Sound Processing Class. Then in our Machine Learning Class we implemented automatic feature selection. Both, the chosen feature set for implementation, as well as their automatic selection for recognition, performed well in our evaluation.

7.2.3. Could we automatically mix activities for their untrained recognition?

We obtained very good results in mixing automatically activities for their untrained recognition. Furthermore, we made a detailed comparison between automatically mixed records of some activities and their real simultaneous occurrence.

7.3. Outlook

There are several possible improvements that could be done in our Sound-based Device Recognition Framework in the near future. The first and most important is to develop automatic acoustic event detection. This will enable creating a smart continuous monitoring of the environment, without the drawback of fixed size frames, where one might buffer only partial acoustic information. This opens the door to modelling the environment using Dynamic Bayesian Networks, which relate states to each other over adjacent time steps. Among the most popular choices are the Hidden Markov Models, which represent a simplified version of a Dynamic Bayesian Network. They observe the state of the environment as the only information source to predict the next occurring state. The state transition probabilities are then computed via observations. In our case this will allow us to recognize activities as processes, instead of making partial momentarily recognitions. For example, by monitoring the sounds produced by some device over time, we could create a model to predict, what kind of activity was performed with the device.

Another way to confidently recognize activities is to go one step further in the direction of automatic event detection. Then, we could wisely extract meaningful audio intervals by applying envelope[35] modeling (Jensen, 1999). After extracting the features from the extracted sound intervals, one could seek for similarities with trained data using the Dynamic Time Warping algorithm[36] (Müller, Ellis, Klapuri, & Richard, 2011). For example, both described techniques could provide a meaningful differentiation between the activity of cleaning with a vacuum cleaner and leaving the vacuum cleaner running uncontrolled, beyond the recognition of the vacuum cleaner device itself.

For cases where one has multiple available sound inputs, one could adopt Sound Processing techniques to exploit their presence in terms of combining their information for obtaining in depth knowledge about the environment. It would also be interesting, whether one could exploit unconventional audio inputs as well. Such an unconventional audio input could be the musical instrument called Theremin, which produces different sounds according to the presence of humans and has successfully been tested for gesture recognition (Endres & Dimitrov, 2010).

7.4. Open Problems

Here we select three major complex problems that are still not investigated by any research collective. Each of those problems represents an interesting challenge in the domain of Sound-based Activity Recognition:

Environment with multiple inhabitants – it would be interesting to investigate an environment, where multiple persons are acting and producing different sounds. It is indeed a challenge to recognize them and their actions. One possible solution of the problem is after recording all sounds to mix them automatically and extract the features from the mix in the conventional way, as successfully incorporated in this thesis for multiple device recognition. A complicated scenario here might be the case, when dealing with architecture of centralized server and multiple clients, having thousands of records, where mixing all those records might not be practical, so one has to be able to make good automatic decisions, which activities could occur concurrently and at what place. There is a similar challenge in the Speaker and Speech Recognition field to discriminate, which person is talking, besides the actual Speech Recognition. However, there is often a good concern that both speakers usually don't speak at same time, which is not feasible in activity context.

[35] The evolution over time of the amplitude of a sound

[36] Algorithm for measuring similarity between two temporal sequences which may vary in time or speed. Finds numerous applications in Music Recognition in measuring similarity between music pieces, which are played with different tempo.

Rare activities – activities, which seldom happen like repairing a house or house accidents. A recognizer could not cover all those activities in a learning phase. However, an appropriate reaction should present, especially in case of an accident, one can consider high level abstraction, in which a central server processes a large database of rare activities listed with their features to run a lookup when the personalized recognizer fails to provide a recognition answer.

Transition to Unsupervised Learning - when receiving features, which are deviating from all current references, many times the same way, the system should be able to note, that there might be new activity happening. We could eventually tryout completely unsupervised activity recognition using activity models and common sense similar to (Wyatt, Philipose, & Choudhury, 2005) (Marszalek, Laptev, & Schmid, 2009).

Bibliography

Aztiria, A., Izaguirre, A., & Augusto, J. C. (2010). Learning patterns in ambient intelligence environments: a survey. *Artificial Intelligence Review, 34*(1), 35-51.

Beigi, H. (2011). *Fundamentals of Speaker Recognition.* New York, USA: Springer.

Belley, C., Gaboury, S., Bouchard, B., & Bouzouane, A. (2013). Activity Recognition in Smart Homes Based on Electrical Devices Identification. *Proceedings of the 6th International Conference on PErvasive Technologies Related to Assistive Environments* (pp. 7:1--7:8). Island of Rhodes, Greece: ACM.

Bernsee, S. M. (September 21, 1999). *The DFT "à Pied": Mastering The Fourier Transform in One Day.* DSP Dimension. Germany: DSP Dimension.

Boehm, B. W. (1988, May). A spiral model of software development and enhancement. *Computer, 21*(5), 61-72.

Bolón-Canedo, V., Porto-Díaz, I., Sánchez-Maroño, N., & Alonso-Betanzos, A. (2014, July). A framework for cost-based feature selection. *Pattern Recognition, 47*(7), 2481-2489.

Brey, P. (2005). Freedom and Privacy in Ambient Intelligence. *Ethics and Information Technology, 7*(3), 157-166.

Chu, S., Narayanan, S., & Kuo, C.-C. (2009, Aug). Environmental Sound Recognition With Time–Frequency Audio Features. *IEEE Transactions on Audio, Speech, and Language Processing, 17*(6), 1142 - 1158.

Cook, D. J., Augusto, J. C., & Jakkula, V. R. (2009, August). Ambient Intelligence: Technologies, Applications, and Opportunities. *Pervasive and Mobile Computing, 5*(4), 277-298.

Cooley, J. W., & Tukey, J. W. (1965). An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation, 19*(90), 297-301.

Daubechies, I. (1992). *Ten Lectures on Wavelets* (9th ed., Vol. 61). Philadelphia, Pennsylvania, USA: Society for Industrial and Applied Mathematics.

Dersch, W. C. (1962). *IBM Shoebox.* Advanced Systems Development Division Laboratory, Advanced Technology Group. San Jose, Calif: IBM.

Ellis, D. P., Whitman, B., & Porter, A. (October 24-28, 2011). Echoprint. *12th International Society for Music Information Retrieval Conference.* Miami: ISMIR.

Endres, C., & Dimitrov, S. (November 11-12, 2010). Using a Theremin for Micro-Gesture Recognition in an Automotive Environment. *Adjunct proceedings of the 2nd International Conference on Automotive User Interfaces and Interactive Vehicular Applications.* Pittsburgh, PA, USA: AutomotiveUI.

Feld, M. (2011). *A speaker classification framework for non-intrusive user modeling: speech-based personalization of in-car services.* Computer Science Institute, DFKI. Saarbrücken: Saarland University.

Frey, J. (July 18-19, 2013). AdAPT - A dynamic Approach for Activity Prediction and Tracking for Ambient Intelligence. *Proceedings of the 9th International Conference on Intelligent Environments* (pp. 254 - 257). Athens, Greece: IEEE.

Frey, J., Neurohr, C., & Brandherm, B. (July 2-4, 2014). EvA - Self Adaptable Event-based Recognition Framework for Three-Dimensional Activity Zones. *Proceedings of the 10th International Conference on Intelligent Environments.* Shanghai, China.

Frey, J., Stahl, C., Röfer, T., Krieg-Brückner, B., & Alexandersson, J. (2010). The DFKI Competence Center for Ambient Assisted Living. *Ambient Intelligence: First International Joint Conference*, 310-314.

Fujishima, T. (1999). Realtime Chord Recognition of Musical Sound: a System Using Common Lisp Music. *Proceedings of the International Computer Music Conference.* Beijing, China: International Computer Music Association.

Glinsky, A. (2000). *Theremin: Ether Music and Espionage.* Chicago: University of Illinois Press.

Gupta, S., Reynolds, M. S., & Patel, S. N. (2010). ElectriSense: single-point sensing using EMI for electrical event detection and classification in the home. *Proceedings of the 12th ACM international conference on Ubiquitous computing* (pp. 139-148). New York, NY, USA: ACM.

Guyon, I., & Elisseeff, A. (2003, 3 1). An introduction to variable and feature selection. *The Journal of Machine Learning Research, 3*, 1157-1182.

Hart, G. (1989, June). Residential energy monitoring and computerized surveillance via utility power flows. *Technology and Society Magazine, 8*(2), 12 - 16.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning* (2nd ed.). Springer.

Herbrich, R., Graepel, T., & Campbell, C. (2001, 9 1). Bayes point machines. *The Journal of Machine Learning Research, 1*, 245-279.

IBM Corporation, & Microsoft Corporation. (1991, August). Multimedia Programming Interface and Data Specifications 1.0. 56-65.

Istrate, D., Vacher, M., & Serignat, J.-F. (2008). Embedded Implementation of Distress Situation. Identification Through Sound Analysis. *The Journal on Information Technology in Healthcare, 6*(3), 204-211.

Jensen, K. (December 9-11, 1999). Envelope model of isolated musical sounds. *Proceedings of the 2nd COST G-6 Workshop on Digital Audio Effects.* Trondheim, Norway: NTNU.

K. H. Davis, R. B. (1952). *Automatic Digit Recognition (AUDREY).* Bell Laboratories.

Karbasi, M., Ahadi, S., & Bahmanian, M. (13-16 Dec 2011). Environmental sound classification using spectral dynamic features. *8th International Conference on Information, Communications and Signal Processing* (pp. 1 - 5). Singapore: ICICS.

Katz, S. (1983, 12). Assessing self-maintenance: activities of daily living, mobility, and instrumental activities of daily living. *Journal of the American Geriatrics Society, 31*(12), 721-727.

Ke, S.-R. a.-J., Hwang, J.-N., Yoo, J.-H., & Choi, K.-H. (2013, 6 5). A review on video-based human activity recognition. *Computers, 2*(2), 88-131.

Koller, D., & Sahami, M. (1996). Toward Optimal Feature Selection. *13th International Conference on Machine Learning* (pp. 284-292). Bari, Italy: Stanford InfoLab.

Kronland-Martinet, R., Morlet, J., & Grossmann, A. (1987, August). Analysis of sound patterns through wavelet transforms. *International Journal of Pattern Recognition and Artificial Intelligence, 1*(02), 273-302.

Lawton, M. P., & Brody, E. M. (1968). Assessment of Older People: Self-Maintaining and Instrumental Activities of Daily Living. *The Gerontologist, 9*(3), 179-186.

Lei, H., & Lopez-Gonzalo, E. (2009). Mel, Linear, and Antimel Frequency Cepstral Coefficients in Broad Phonetic Regions for Telephone Speaker Recognition. *Tenth Annual Conference of the International Speech Communication Association*, 2323-2326.

Lippmann, R. P. (1997). Speech recognition by machines and humans. *Speech communication, 22*(1), 1-15.

Lombriser, C., Amft, O., Zappi, P., Benini, L., & Tröster, G. (2011). Benefits of dynamically reconfigurable activity recognition in distributed sensing environments. *Activity Recognition in Pervasive Intelligent Environments, 4*, 265-290.

Lozano, H., Hernáez, I., Picón, A., Camarena, J., & Navas, E. (2010). Audio Classification Techniques in Home Environments for Elderly/Dependant People. *Computers Helping People with Special Needs*(6179), 320-323.

Mark Hall, E. F. (2009, 6). The WEKA Data Mining Software: An Update. *ACM SIGKDD Explorations Newsletter, Volume 11*(1), pp. 10-18.

Marszalek, M., Laptev, I., & Schmid, C. (20-25 June 2009). Actions in context. *IEEE Conference onComputer Vision and Pattern Recognition (CVPR)* (pp. 2929-2936). Miami, FL, USA: IEEE.

Microsoft Corporation. (2007, December). *Multiple Channel Audio Data and WAVE Files.* Microsoft Corporation.

Minka, T. P. (2001). Expectation Propagation for Approximate Bayesian Inference. *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence. UAI '01*, pp. 362-369. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Minka, T., Winn, J., Guiver, J., & Knowles, D. (2012). *Infer.NET 2.5.* Microsoft Research Cambridge.

Moorer, J. A. (1975). *On the Segmentation and Analysis of Continuous Musical Sound by Digital Computer.* Stanford University, Department of Computer Science. Stanford, CA: Stanford University.

Müller, M., Ellis, D. P., Klapuri, A., & Richard, G. (2011, February 4). Signal processing for music analysis. *IEEE Journal on Selected Topics in Signal Processing, 5*(6), 1088-1110.

Nebeker, F. (1998). *Fifty Years of Signal Processing: The IEEE Signal Processing Society and its Technologies 1948-1998.* IEEE, History Center. New Brunswick, NJ: The IEEE Signal Processing Society.

Passchier-Vermeer, W., & Passchier, W. F. (2000). Noise Exposure and Public Health. *Environmental health perspectives, 108*(1), 123-131.

Peeters, G. (2004). *A large set of audio features for sound description (similarity and classification) in the CUIDADO project.* Institut de Recherche et Coordination Acoustique/Musique, Analysis/Synthesis Team. Paris, France: IRCAM.

Pressman, R. S. (January 20, 2009). *Software Engineering A Practitioner's Approach* (7th ed.). McGraw-Hill Science/Engineering/Math.

Provost, F., & Fawcett, T. (2001, March). Robust Classification for Imprecise Environments. *Machine Learning, 42*(3), 203-231.

Russell, S., & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach* (3 ed.). Upper Saddle River13, New Jersey: Prentice Hall.

Scharenborg, O., & Cooke, M. P. (June 4 - 6, 2008). Comparing human and machine recognition performance on a VCV corpus. *Workshop on Speech Analysis and Processing for Knowledge Discovery.* Aalborg, Denmark: ISCA.

Sehili, M. A., Lecouteux, B., Vacher, M., Portet, F., Istrate, D., Dorizzi, B., et al. (2012). Sound Environment Analysis in Smart Home. *Ambient Intelligence: Third International Joint Conference*, 208-223.

Souza, C. R. (April 2012). *The Accord.NET Framework.* Federal University of Sao Carlos, Department of Computing.

Stäger, M. (2006). *Low-Power Sound-Based User Activity Recognition.* Swiss Federal Institute of Technology Zurich, Information Technology and Electrical Engineering. Zürich: ETH Zürich.

Stager, M., Lukowicz, P., & Troster, G. (2004). Implementation and Evaluation of a Low-Power Sound-Based User Activity Recognition System. *Proceedings of the Eighth International Symposium on Wearable Computers* (pp. 138--141). Washington, DC, USA: IEEE Computer Society.

Stager, M., Lukowicz, P., & Troster, G. (2007, June). Power and accuracy trade-offs in sound-based context recognition systems. *PerCom - Pervasive and Mobile Computing, 3*(3), 300-327.

Tapia, E. M., Intille, S. S., & Larson, K. (2004). Activity recognition in the home using simple and ubiquitous sensors. In A. Ferscha, & F. Mattern, *Lecture Notes in Computer Science* (Vol. 3001, pp. 158-175). Berlin Heidelberg, Germany: Springer.

Temko, A., Malkin, R., Zieger, C., Macho, D., & Nadeu, C. (2006). Acoustic Event Detection and Classification in Smart-Room Environments: Evaluation of CHIL Project Systems. *Cough*, 5-11.

Tzanetakis, G., Essl, G., & Cook, P. (2001). Audio Analysis using the Discrete Wavelet Transform. *Proc. WSES Int. Conf. Acoustics and Music: Theory and Applications.* Skiathos, Greece: AMTA.

Vinet, H., Herrera, P., & Pachet, F. (13th-17th October 2002). The CUIDADO Project (Content-based Unified Interfaces and Descriptors for Audio/music Databases available Online). *3rd International Conference on Music Information Retrieval.* Paris: ISMIR.

Vuegen, L. a., Karsmakers, P., & Vanrumste, B. (2013). Automatic Monitoring of Activities of Daily Living based on Real-life Acoustic Sensor Data: a preliminary study.

Fourth workshop on speech and language processing for assistive technologies (SLPAT) (pp. 113-118). Grenoble, France: Association for Computational Linguistics.

Wang, A. L.-C. (2003). An Industrial-Strength Audio Search Algorithm. *4th Symposium Conference on Music Information Retrieval*, 7-13.

Wang, J.-C., Lee, H.-P., Wang, J.-F., & Lin, C.-B. (2008, January 4). Robust Environmental Sound Recognition for Home Automation. *IEEE Transactions on Automation Science and Engineering, 5*(1), 25 - 31.

Wang, J.-F., Wang, J.-C., Huang, T.-H., & Hsu, C.-S. (2003, Dec 30). Home environmental sound recognition based on MPEG-7 features. *IEEE 46th Midwest Symposium on Circuits and Systems, 2*, 682 - 685.

Weber, W., Rabaey, J., & Aarts, E. H. (March, 2005). *Ambient Intelligence.* Springer.

Wieczorkowska, A., Kolczyńska, E., & Raś, Z. W. (2008). Training of Classifiers for the Recognition of Musical Instrument Dominating in the Same-Pitch Mix. *New Challenges in Applied Intelligence Technologies, III*(134), 213-222.

Wyatt, D., Philipose, M., & Choudhury, T. (May 9, 2005). Unsupervised activity recognition using automatically mined common sense. *Proceedings of the 20th national conference on Artificial intelligence. 1*, pp. 21-27. AAAI Press.

Young, S., Kershaw, D., Odell, J., Ollason, D., Valtchev, V., & Woodland, P. (2000). *The HTK Book (for HTK Version 3.1)* (5 ed.). Microsoft Corporation.

Yuk, D. (1999). *Robust Speech Recognition Using Neural Networks and Hidden Markov Models.* The State University of New Jersey, Graduate School-New Brunswick. New Brunswick: Rutgers.

Zhuang, X., Zhou, X., Huang, T. S., & Hasegawa-Johnson, M. (2008). Feature analysis and selection for acoustic event detection. *IEEE International Conference on Acoustics, Speech and Signal Processing, 2008 (ICASSP)* (pp. 17-20). Las Vegas, USA: IEEE.

Appendix

A. Class Organization

In Figure A.1 we see a full class diagram containing most important classes in the project and their corresponding place (in parallel to Subchapter 5.2).

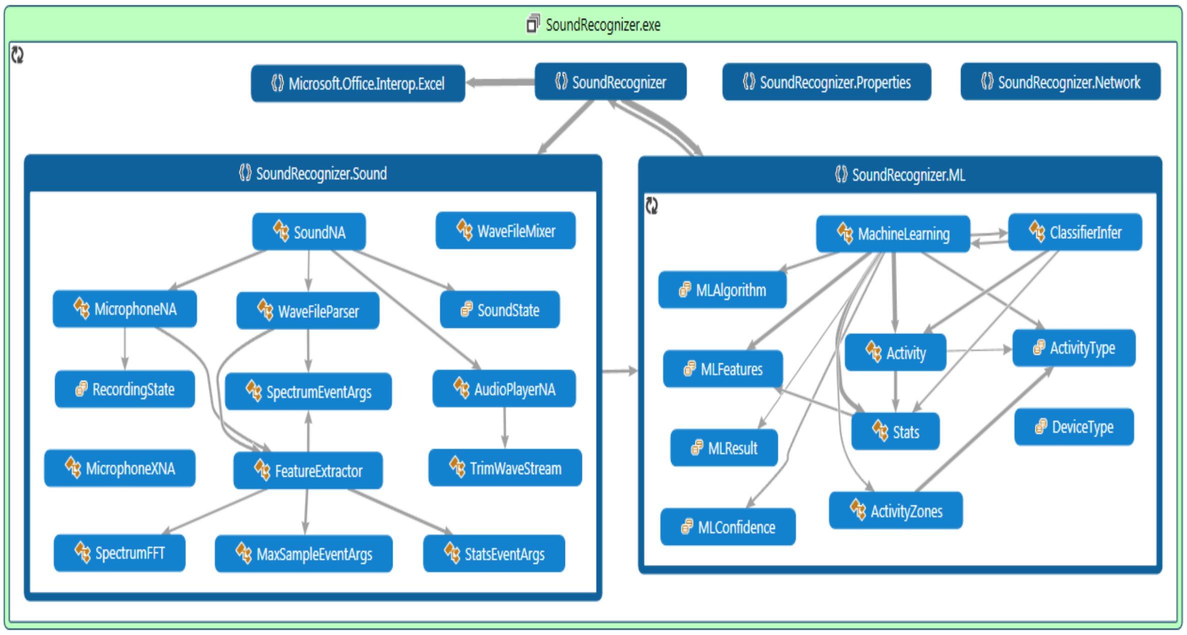


Figure A.1 *Full Class Diagram of the Sound-based Device Recognition Solution in Visual Studio 2012*

B. Activity Type and Device Type

When the user annotates some Activity Type we infer Device Type as follows:

```
public enum ActivityType
{
    Absorbing,          // -> Absorber
    BikeMaintenance,    // -> Chain Cleaner
    Cleaning,           // -> Vacuum Cleaner
    CoffeeMaking,       // -> Moka Pot
    Epilating,          // -> Epilator
    FridgeCooling,      // -> Fridge
    HairDrying,         // -> Hairdryer
    Hairtrimming,       // -> Hairtrimmer
    Heating,            // -> Heater
    Hedgecutting,       // -> Hedgecutter
    Microwaving,        // -> Microwave
    Mixing,             // -> Mixer
    MusicListening,     // -> Music Centre
    ShakerPreparation,  // -> Shaker
    Shaving,            // -> Shaver
    Showering,          // -> Shower
    Speaking,           // -> Speaker
    ToiletFlush,        // -> Toilet
    Toothbrushing,      // -> Electrical Toothbrush
    Washing,            // -> Washing Machine
    WaterBoiling,       // -> Kettle
    Test,               // -> Test Device
    Mixed1,             // -> <Device1,Device2,(Device3)>
    Mixed2,             // -> <Device1,Device2,(Device3)>
    Mixed3,             // -> <Device1,Device2,(Device3)>
    Unknown,            // -> {...}
}
```

Where $Unknown$ is fixed to unknown activities, like those which the user has not defined or decided to hide, $Test$ is for test purposes of unlisted activity types, $Mixed1, Mixed2, Mixed3$ are for three available mix choices.

C. Multi-dimensional Nearest Neighbor

Here we compute the Nearest Neighbor and return a sorted list with the nearest neighbors, according to Section 5.5.4. One should note that this is a simplified version, where logic for averaged statistics and common metrics is omitted.

```
public SortedDictionary<double, int> NearestNeighbourMD(Stats currentStats)
{
    SortedDictionary<double, int> distanceToFeature = new SortedDictionary<double, int>();

    for (int activityIndex = 0; activityIndex < activities.Count; activityIndex++)
    {
        double distance = 0.0;

        for (int featureIndex = 0; featureIndex < MLFeaturesCount; featureIndex++)
        {
            double reference = activities[activityIndex].stats.array[featureIndex]);
            double current = currentStats.array[featureIndex];

            if (!currentStats.ignore[featureIndex])
            {
                distance += Math.Pow(reference - current, 2.0);
            }
        }

        distanceToFeature.Add(Math.Sqrt(distance), activityIndex);
    }

    return distanceToFeature;
}
```

D. Test Records Spectrograms

Here we show 10 second spectrocrograms of all 22 test records:

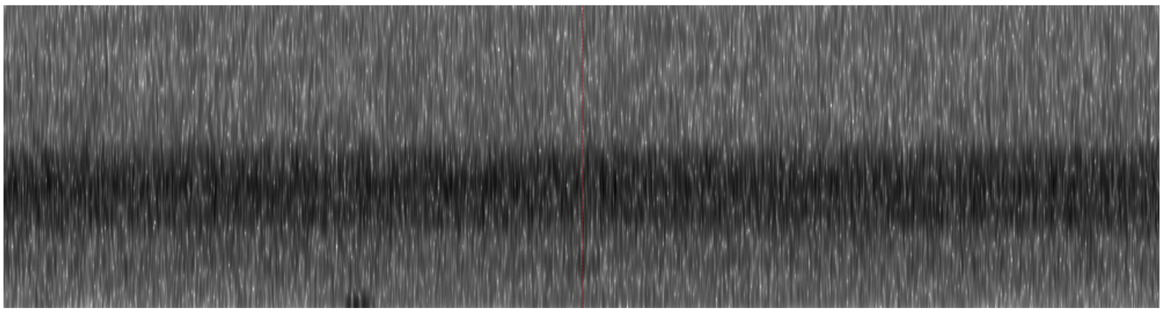


Figure D.1 *Absorber*

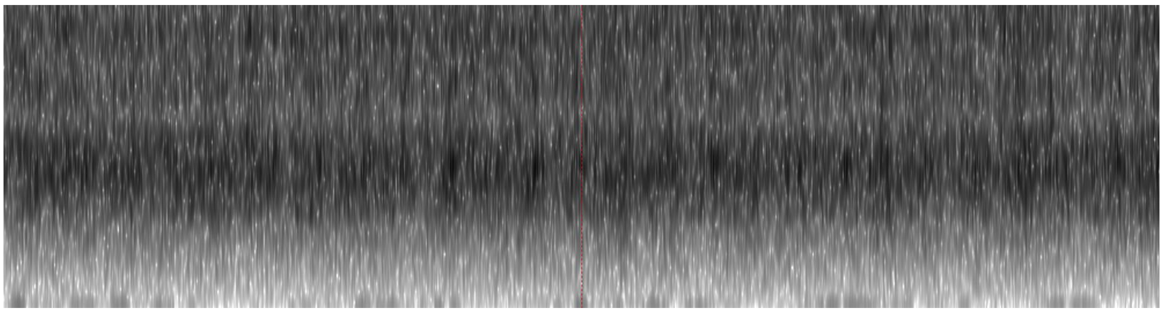


Figure D.2 *Chain Cleaner*

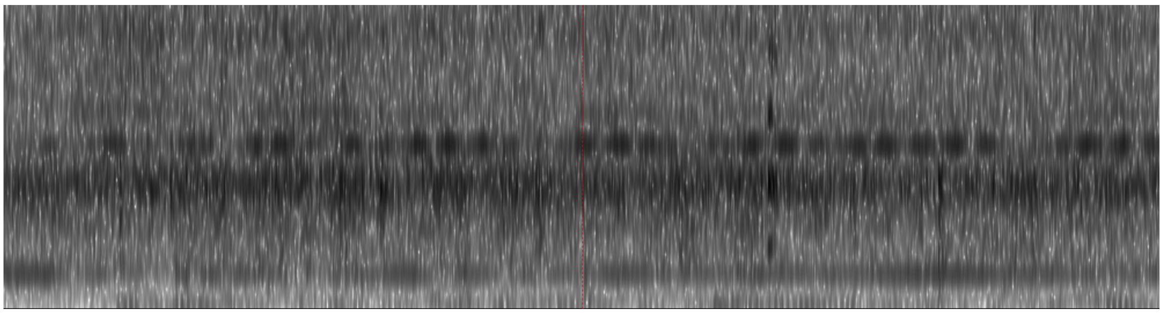


Figure D.3 *Vacuum Cleaner*

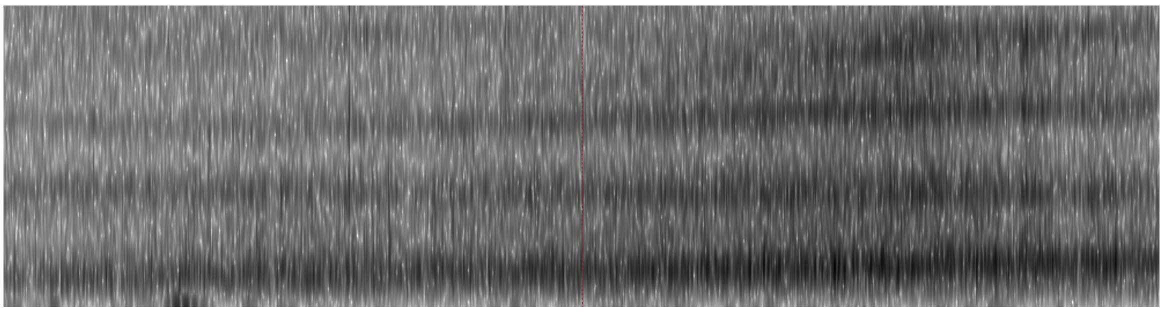


Figure D.4 *Moka Pot*

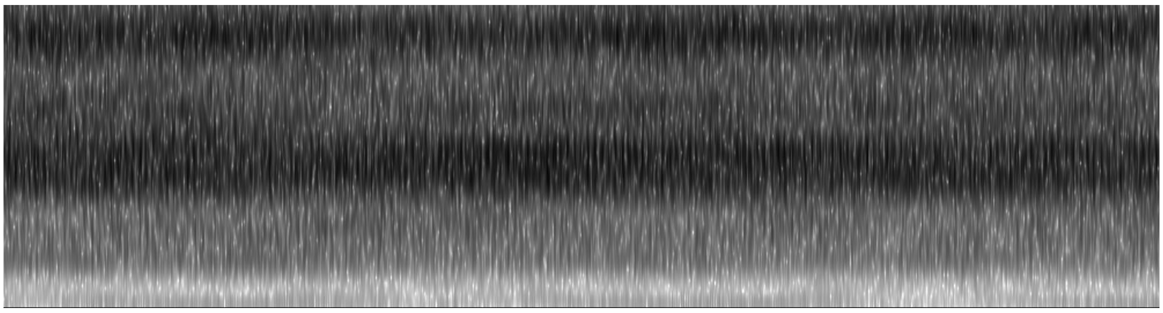


Figure D.5 *Epilator*

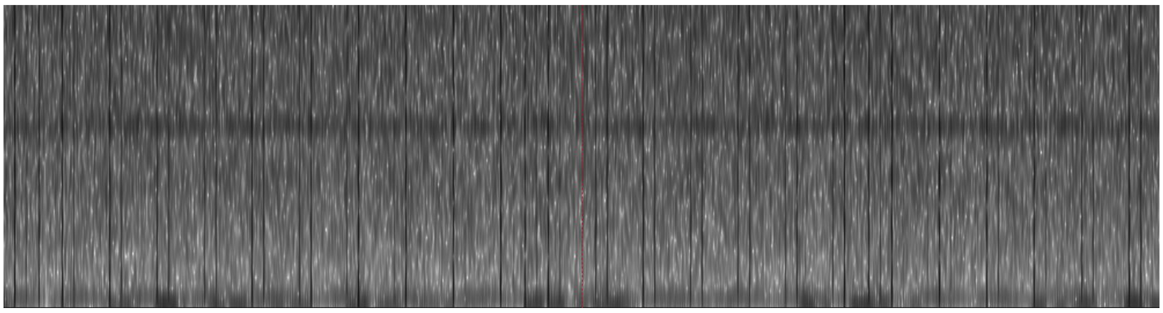


Figure D.6 *Fridge*

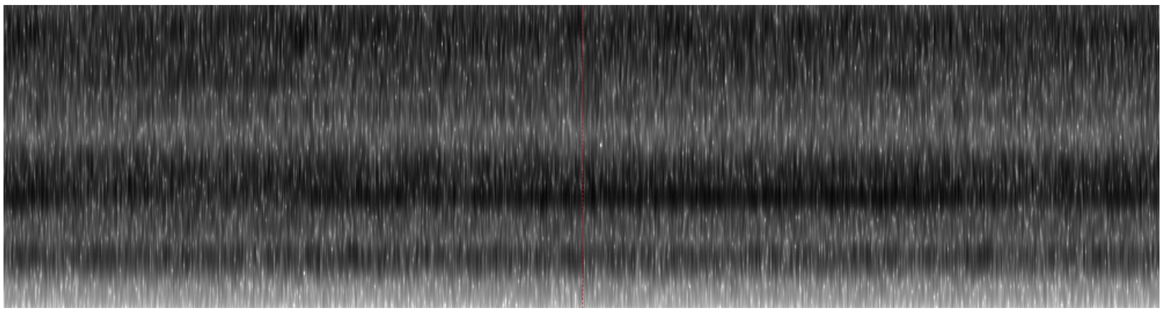


Figure D.7 *Hairdryer*

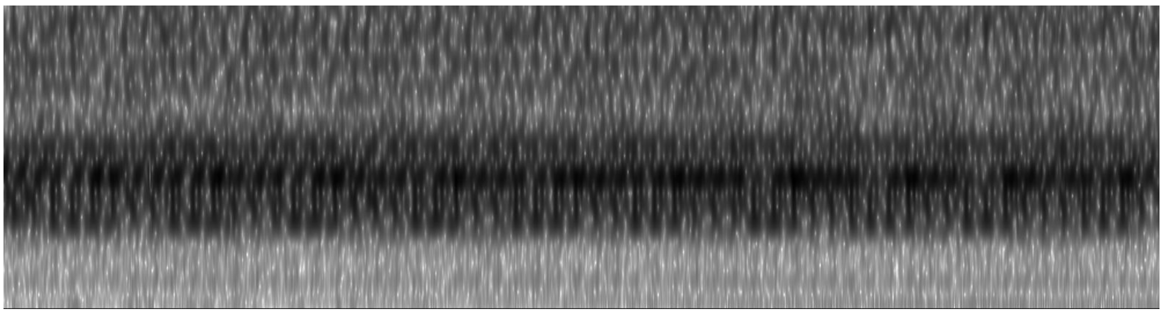


Figure D.8 *Hairtrimmer*

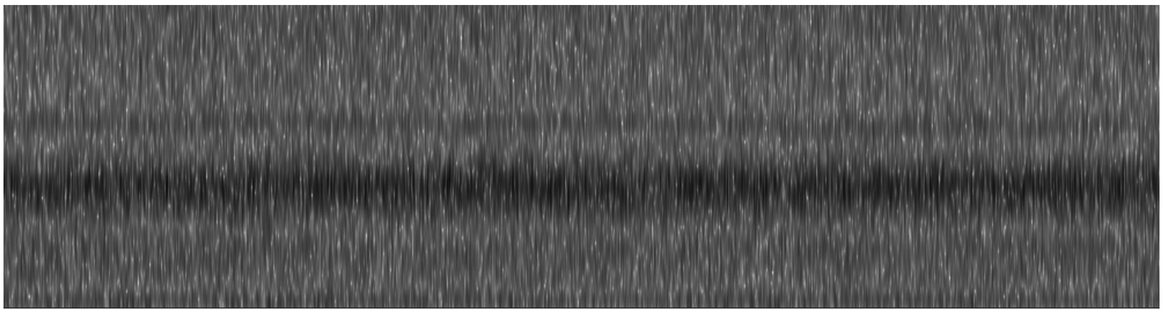


Figure D.9 *Heater*

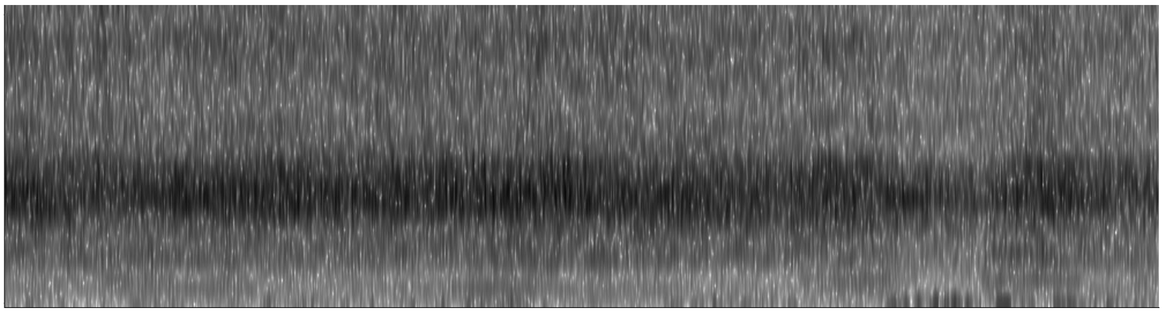


Figure D.10 *Hedgecutter*

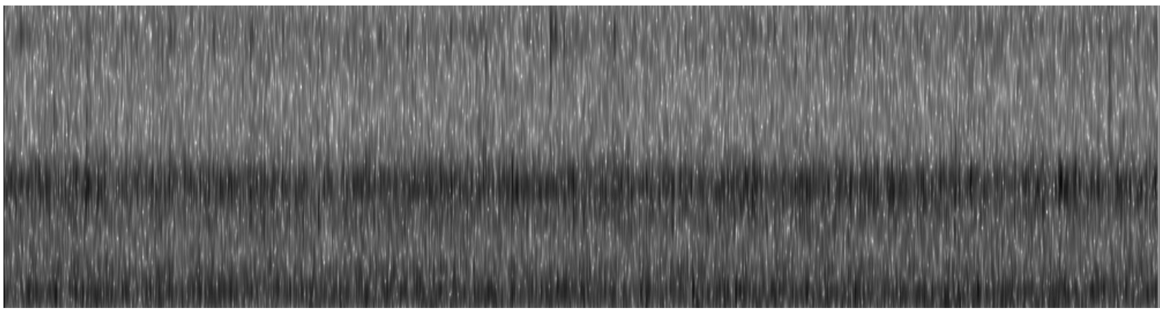


Figure D.11 *Kettle*

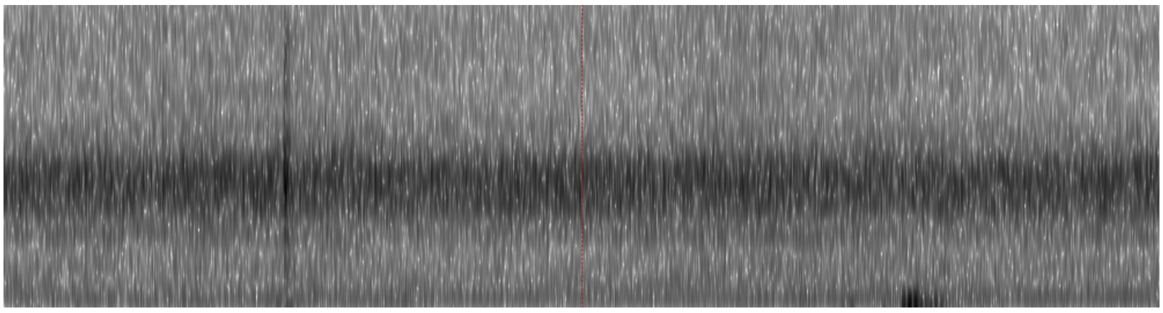


Figure D.12 *Microwave*

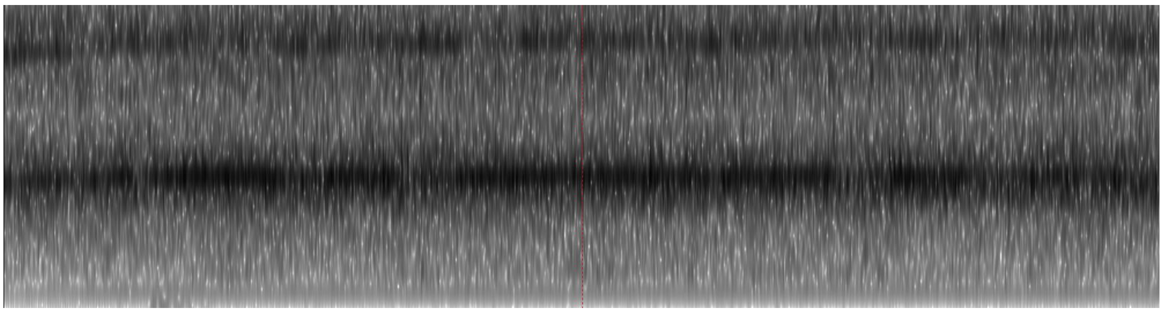


Figure D.13 *Mixer*

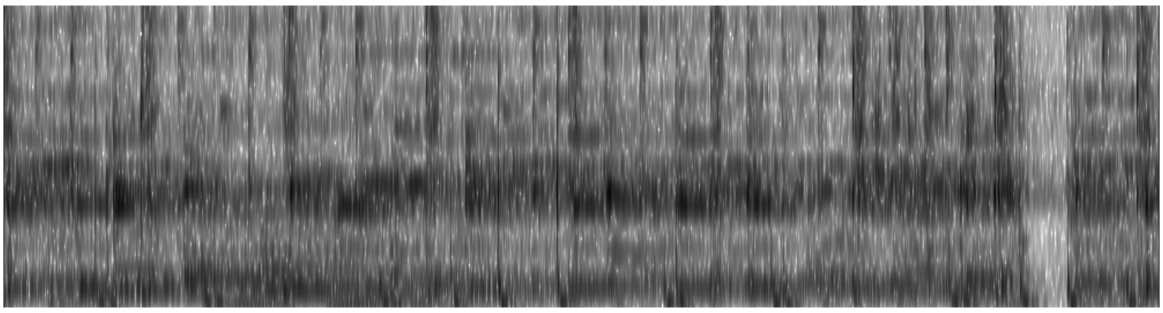


Figure D.14 *Music*

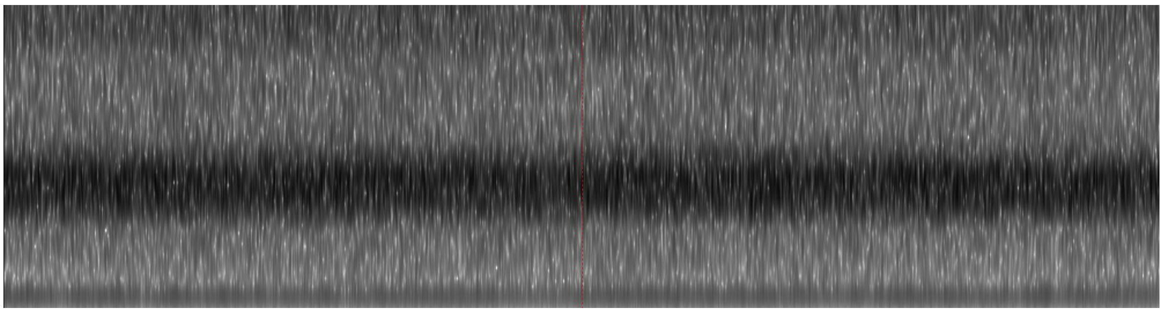


Figure D.15 *Shaker*

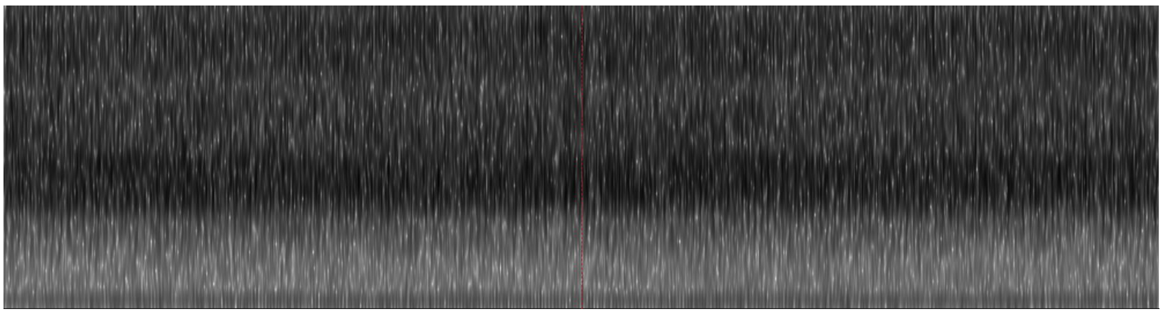


Figure D.16 *Shaver*

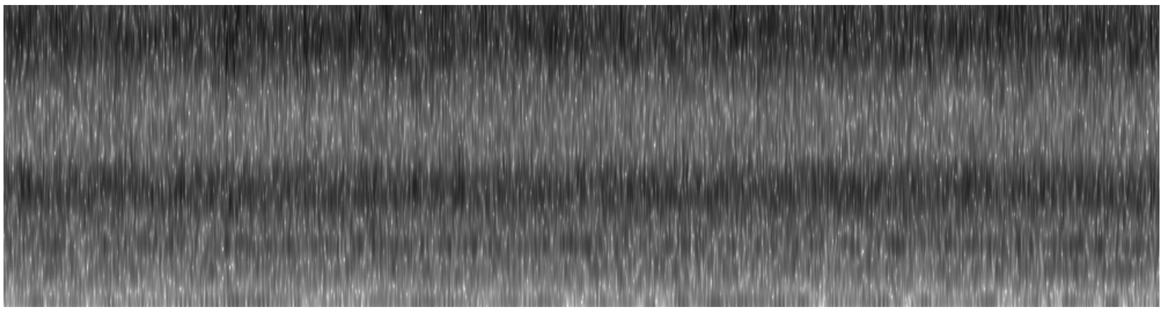


Figure D.17 *Shower*

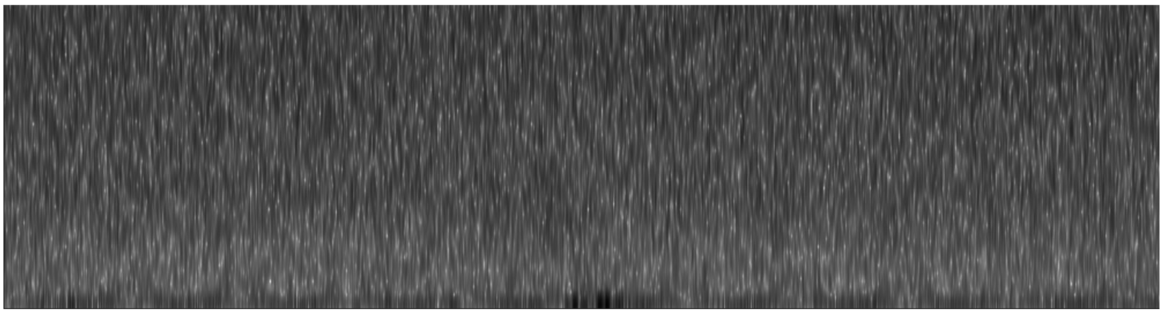


Figure D.18 *Silence*

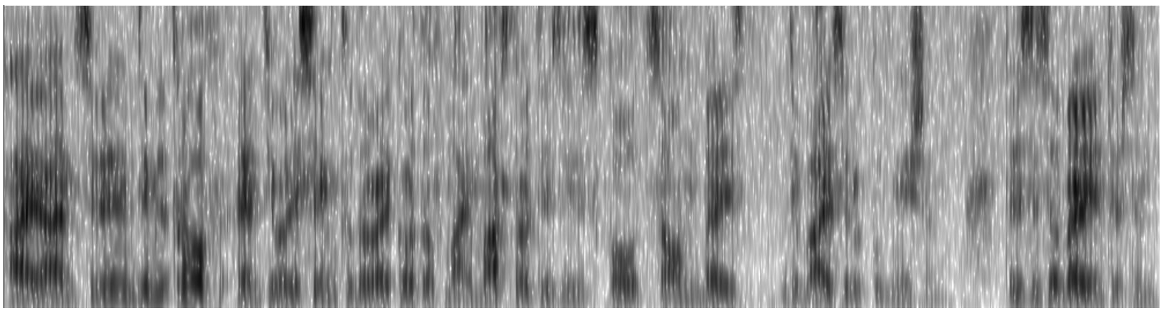


Figure D.19 *Speaking*

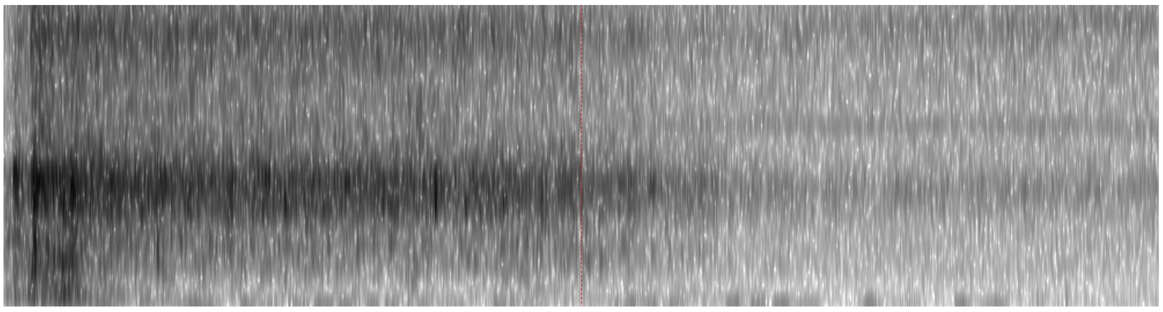


Figure D.20 *Toilet Flush*

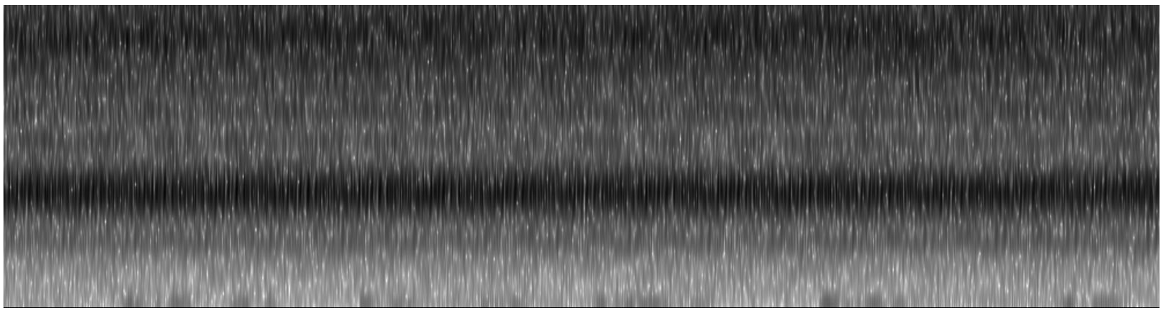


Figure D.21 *Toothbrush*

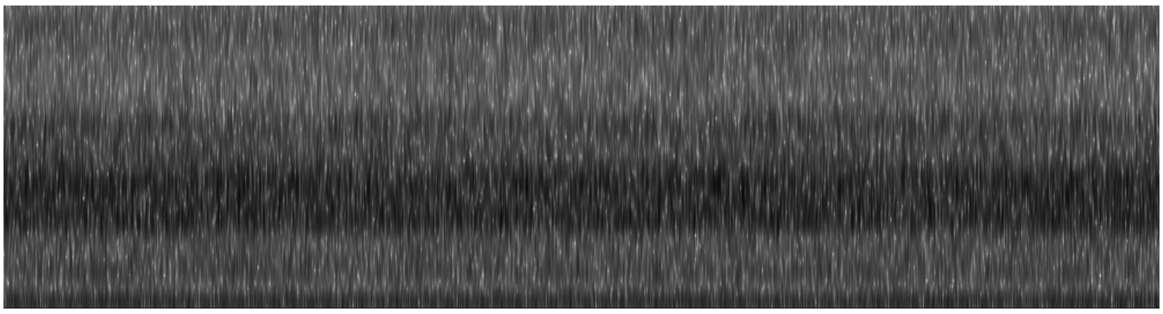


Figure D.22 *Washing Machine*

E. Mixed Records Spectrograms

Here we compare automatically generated mixes with their corresponding real records of complex activities.

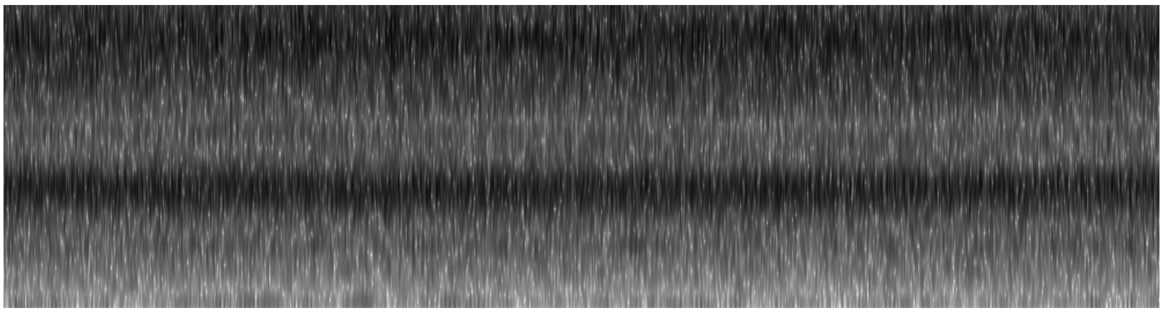


Figure E.1 *Original Mix Toothbrush and Shower (Mixed 1)*

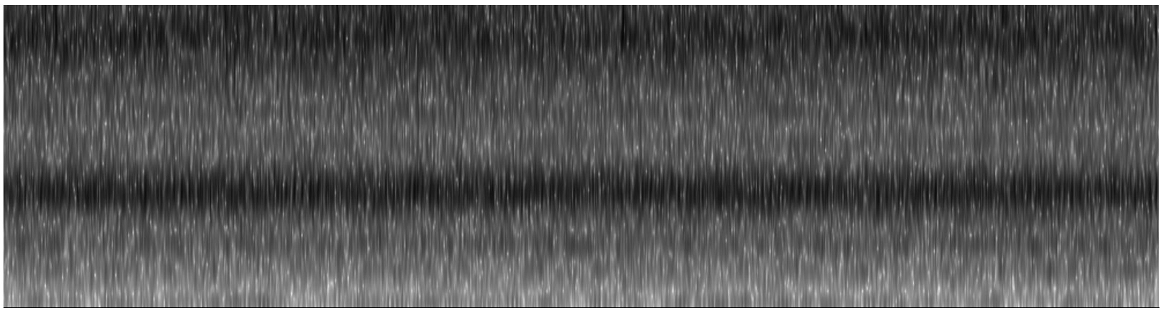


Figure E.2 *Automatic Mix Toothbrush and Shower (Mixed 1)*

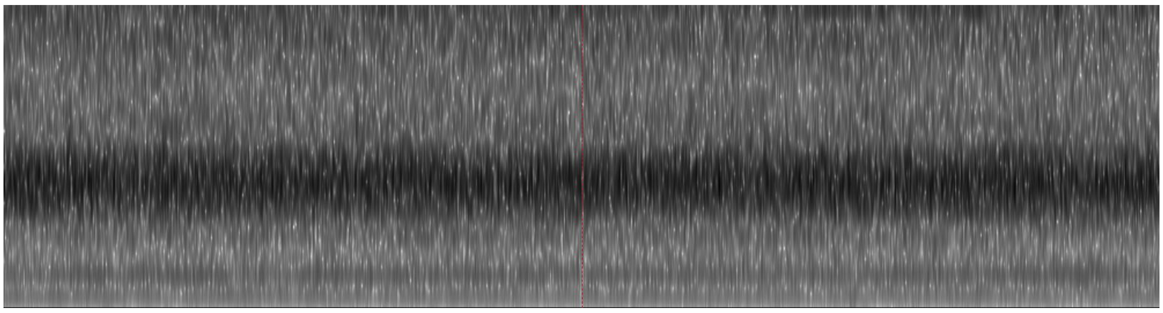


Figure E.3 *Original Mix Shaker and Vacuum Cleaner (Mixed 2)*

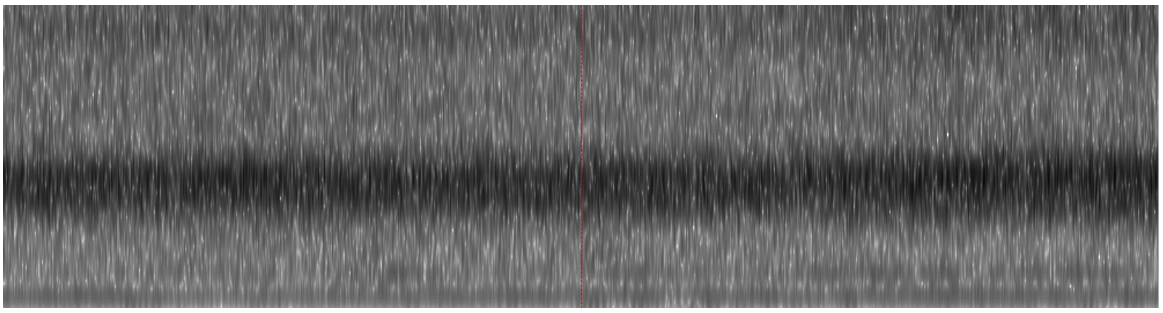


Figure E.4 *Automatic Mix Shaker and Vacuum Cleaner (Mixed 2)*

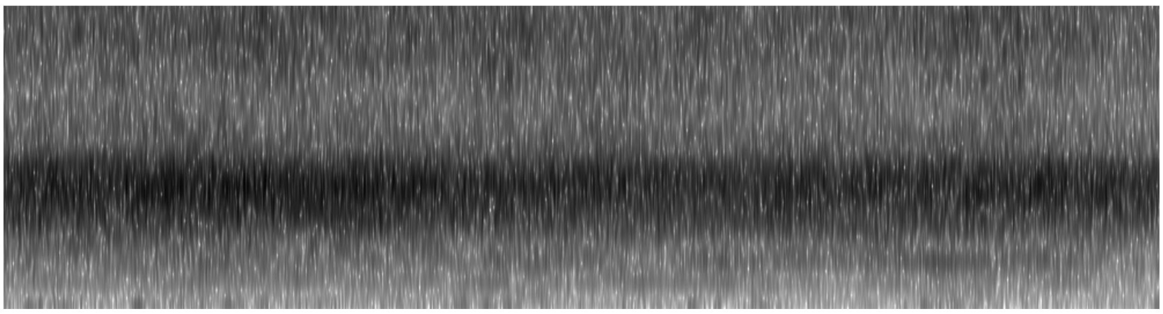


Figure E.5 *Original Mix Hedgecutter and Hairtrimmer (Mixed 3)*

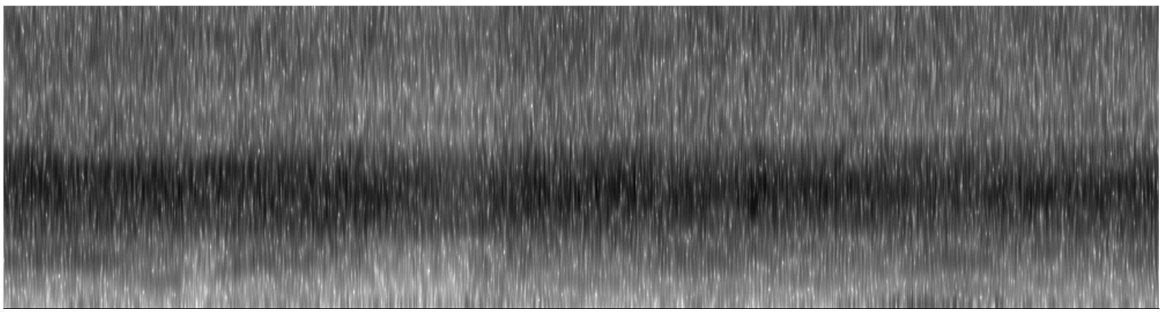


Figure E.6 *Automatic Mix Hedgecutter and Hairtrimmer (Mixed 3)*