Low-Power Sound-Based User Activity Recognition

Mathias Stäger
Diss. ETH No. 16719

Diss. ETH No. 16719

Low-Power Sound-Based User Activity Recognition

A dissertation submitted to
ETH ZURICH

for the degree of
Doctor of Sciences

presented by
MATHIAS STÄGER
Dipl. El.-Ing. ETH
born 11th August 1975
citizen of Zizers GR

accepted on the recommendation of
Prof. Dr. Gerhard Tröster, examiner
Prof. Dr. Paul Lukowicz, co-examiner

2006

Mathias Stäger

Low-Power Sound-Based User Activity Recognition

Diss. ETH No. 16719, Zürich, 2006

© Mathias Stäger, 2006

Swiss Federal Institute of Technology Zurich (ETH Zürich)

Copies may be ordered online from `http://www.lulu.com`

Contents

Abstract **vii**

Zusammenfassung **ix**

1. Introduction **1**
- 1.1. Context Recognition and Activity Recognition 2
  - 1.1.1. Context-Awareness 2
  - 1.1.2. Activity Recognition 2
- 1.2. Use of Sound for User Activity Recognition 3
  - 1.2.1. Why Using Sound? 3
  - 1.2.2. Relevant Application Domains 5
- 1.3. Importance of Power Awareness 6
- 1.4. Related Work 8
  - 1.4.1. Audio Recognition in Wearable Computing 8
  - 1.4.2. Trade-offs in Wearable Computing 9
- 1.5. Research Objectives and Contributions 11
- 1.6. Outline 11

2. Approach **13**
- 2.1. Starting Position and Constraints 14
  - 2.1.1. System Architecture and Signal Flow 14
  - 2.1.2. Microphone Placement 16
  - 2.1.3. Number of Sounds 16
  - 2.1.4. Hardware Constraints 17
  - 2.1.5. Type of Sounds: Quasi Stationary Sounds 18
  - 2.1.6. Working with Isolated Events 19
- 2.2. Analyzing a Context Recognition System 20
  - 2.2.1. Need for Empirical Analysis 20
  - 2.2.2. Method Overview 21
- 2.3. Summary 22

3. Case Studies **23**
- 3.1. Experiments 24
  - 3.1.1. Data Collection I 25
  - 3.1.2. Data Collection II 26
- 3.2. Hardware 29
- 3.3. Summary 31

4. Algorithms for Sound-Based Activity Recognition 33
- 4.1. Introduction 34
- 4.2. Analysis of the Recognition Procedure 35
  - 4.2.1. Initial Assumption: Frame-Based Recognition . . . 35
  - 4.2.2. Sensor Placement and Sensor Selection 36
  - 4.2.3. Signal Acquisition 36
  - 4.2.4. Data Pre-processing and Segmentation 37
  - 4.2.5. Feature Extraction and Classification 40
- 4.3. Results of the Case Studies 43
  - 4.3.1. Feasibility of the Two Microphones Method 43
  - 4.3.2. Approach Validation with Data Set I 47
  - 4.3.3. Recognition Performance of Data Set II 51
- 4.4. Summary 59

5. Design for Power Efficiency 61
- 5.1. Introduction 62
- 5.2. Power and Execution Time Measurements 63
  - 5.2.1. Theoretical Considerations on the Power Model . . 63
  - 5.2.2. Power of Sensors and Signal Acquisition Stage . . 68
  - 5.2.3. Execution Time for Feature Calculation 69
  - 5.2.4. Execution Time for Classifiers 72
  - 5.2.5. Total Average Power Consumption 74
- 5.3. Trade-off Analysis 75
  - 5.3.1. Power Consumption vs. Recognition Rate Trade-off 75
  - 5.3.2. Classification Speed vs. Recognition Rate Trade-off 82
- 5.4. Summary 83

6. Discussion and Conclusion 85
- 6.1. Discussion 86
  - 6.1.1. Classifying a Large Number of Sounds 86
  - 6.1.2. Computation vs. Communication Trade-off 87
  - 6.1.3. Online Recognition Performance 89
  - 6.1.4. Relevance and Limitations of Design Method . . . 92
  - 6.1.5. Glimpse into the Future 93
- 6.2. Conclusions 97

A. Dealing with Class Skew in Context Recognition 99
- A.1. Introduction 100
- A.2. Two Class Problem 101
  - A.2.1. Illustration of the Problem 101
  - A.2.2. Receiver Operating Characteristic (ROC) 104
  - A.2.3. Area Under the Curve (AUC) 108
  - A.2.4. Case Study 108

A.3. Multi Class Problem 111
  A.3.1. Multi-class ROC 111
  A.3.2. Multi-class AUC 111
  A.3.3. Case Study Continued 112
  A.3.4. Theoretical Example 113
A.4. Discussion 113
A.5. Summary 114

B. Definitions **115**
B.1. Definition of Features 115
  B.1.1. Time-domain features 115
  B.1.2. Frequency-domain features 116
B.2. Definition of Classifiers 117
  B.2.1. Nearest class-center classifier (NCC) 118
  B.2.2. k-nearest neighbor (kNN) 118
  B.2.3. Naive Bayes Classifier 119
B.3. Classifier Fusion Methods 120
B.4. Sound Pressure and Sound Intensity 122
  B.4.1. Sound Pressure 122
  B.4.2. Sound Intensity 124
  B.4.3. Relationship between Sound Pressure and Intensity 124

Glossary **125**

Bibliography **129**

Acknowledgments **141**

Curriculum Vitae **142**

Abstract

In the near future, tiny computers and sensors integrated into our clothes will monitor our physical condition and activities to assist us in everyday tasks: be it suggesting simple improvements in a recipe while we cook or even preventing a maintenance worker from a hazardous mistake.

Typically, activities are recognized with motion sensors. In this work, we use sound as a novel modality for activity recognition. With a microphone mounted on the wrist, e.g. inside a watch, we are able to pick up sounds that are caused by the user or occur in close proximity to the user's hand. Thus, we can detect hand movements which generate a sound, like switching on an appliance or performing a noisy manual task.

Activity recognition systems are usually tuned to deliver high recognition rates. However, size and battery capacity of wearable systems impose limitations to the type and amount of sound processing that can be done. The challenge is to carefully select the parameters of the recognition process to achieve on the one hand high recognition rates and on the other hand low power consumption. Thus, our work describes advances towards the development of a power optimized recognition system: a system that is tuned during the training phase of the recognition process to represent a trade-off between power consumption and recognition accuracy.

We discuss advantages and limitations of the sound-based activity recognition approach. We show that reasonable scenarios exist where sound is a useful means to detect activities. We present case studies in which we recorded data with wrist worn microphones for four scenarios: operating kitchen appliances, performing manual tasks in a wood workshop, operating office appliances and being outdoors/using public transport. These recordings are used to benchmark various sound processing algorithms.

To make a low-power, sound-based activity recognition system feasible, we work with a frame-based method – in contrast to a continuous recognition. This requires segmentation procedures which partition the data stream into potentially interesting segments. One technique proposed here is based on the difference of the signal amplitudes of a wrist and a chest worn microphone. Furthermore, we analyze the recognition process and discuss how recognition accuracy is affected by various parameters like sampling frequency, sampling duration, number of frames, choice of features and classifiers. In addition to sound, we investigate another sensor modality: acceleration. Different methods to fuse data from two sensors are explored.

To estimate power consumption, we use a hardware platform containing a microcontroller, a microphone and accelerometers. Based on power measurements of selected operating points and a simplified model of the hardware platform, we synthesize its power consumption for various sampling frequencies, sampling durations and signal processing algorithms running on the microcontroller.

Finally, we combine the two conflicting metrics ‘recognition rate’ and ‘power consumption’ to a pareto plot. With the data from our case studies we demonstrate that the methods proposed in this work lead to improvements in battery lifetime by a factor of 2 to 4 with only little degradation in recognition performance. To conclude, a wrist worn sensor node recognizing 5 different kitchen appliances with 94% accuracy consumes as little as 0.55 mW. This allows to power it for 42 days with a 2 cm^3 lithium-ion battery.

Zusammenfassung

In naher Zukunft werden winzige, in Kleidung integrierte, Computer und Sensoren unseren Gesundheitszustand und unsere Aktivitäten überwachen, um uns bei alltäglichen Tätigkeiten zu unterstützen: sei es um Verbesserungsvorschläge an einem Kochrezept, das wir gerade ausprobieren, anzubringen oder um einen Servicetechniker vor einem gefährlichen Fehlgriff zu warnen.

Üblicherweise werden zur Erkennung von Aktivitäten Bewegungssensoren verwendet. In dieser Arbeit wählen wir einen neuartigen Ansatz und verwenden Geräusche, um Aktivitäten zu erkennen. Mit einem Mikrophon, das am Handgelenk befestigt ist, z.B. in einer Uhr, können wir Geräusche erfassen, welche durch den Benutzer verursacht werden oder in der näheren Umgebung seiner Hand auftreten. Dadurch können wir diejenigen Handbewegungen detektieren, die ein Geräusch auslösen, wie zum Beispiel das Einschalten eines Geräts oder eine laute manuelle Tätigkeit.

Systeme zur Erkennung von Aktivitäten sind üblicherweise auf eine hohe Erkennungsrate ausgerichtet. Allerdings beschränken Grösse und Batteriekapazität von tragbaren Computern die Art und Weise, wie Geräuscherkennung durchgeführt werden kann. Deshalb ist es eine Herausforderung, die Parameter des Erkennungsprozesses so zu wählen, dass einerseits eine hohe Erkennungsrate und andererseits ein geringer Leistungsverbrauch erzielt wird. In dieser Arbeit streben wir ein Aktivitäten-Erkennungs-System an, das einen Kompromiss zwischen Leistungsverbrauch und Erkennungsrate darstellt.

Wir diskutieren die Vorteile und Einschränkungen eines Aktivitäten-Erkennungs-Systems, das sich auf Geräusche stützt. Wir präsentieren Fallstudien, in welchen wir mit am Handgelenk getragenen Mikrofonen für vier Szenarien Daten aufgenommen haben: Ein- und Ausschalten von Küchengeräten, verschiedene manuelle Tätigkeiten im Bereich von Holzverarbeitung, Tätigkeiten im Büro und Geräusche die im Freien respektive in öffentlichen Verkehrsmitteln vorkommen. Die Aufnahmen werden verwendet, um Leistungsvergleiche zwischen verschiedenen Algorithmen, die Geräusche verarbeiten, durchzuführen.

Um ein Aktivitäten-Erkennungs-System mit einer niedrigen Leistungsaufnahme zu realisieren, arbeiten wir mit einer Methode, die einzelne 'Frames' oder Fenster analysiert – im Gegensatz zu einer kontinuierlichen Erkennung. Dazu werden Verfahren notwendig, die den Datenstrom in potenziell interessante Segmente aufteilen können. Eine Technik, die hier vorgestellt wird, basiert auf dem Lautstärkeunterschied zwischen einem Mikrofon am Handgelenk und an der Brust. Des Weiteren untersuchen wir den Erkennungsprozess und diskutieren dabei, inwiefern die Erkennungsrate von Parametern wie der Abtastfrequenz, der Abtastdauer und der Wahl der Erkennungsmerkmale und Klassifikatoren abhängt. Zusätzlich zum Mikrofon betrachten wir die Verwendung eines zweiten Sensors: eines Beschleunigungssensors. In diesem Zusammenhang beschäftigen wir uns mit unterschiedlichen Verfahren, um die Daten von den zwei Sensoren zusammen zu führen.

Um den Leistungsverbrauch abzuschätzen, verwenden wir eine Hardware, die einen Mikrokontroller, ein Mikrofon und Beschleunigungssensoren enthält. Basierend auf Stromverbrauchsmessungen von ausgewählten Operationsmodi und einem vereinfachten Modell der Hardware berechnen wir deren Leistungsaufnahme für verschiedene Abtastfrequenzen, Abtastlängen und signalverarbeitende Algorithmen.

Schlussendlich kombinieren wir die zwei Metriken 'Erkennungsrate' und 'Leistungsverbrauch' zu einer Pareto Grafik. Mit Hilfe unserer Fallstudien zeigen wir, dass die in dieser Arbeit vorgestellten Methoden die Batterielebensdauer um Faktor 2 bis 4 verlängern können bei beinahe gleichbleibender Erkennungsrate. Letztendlich verbraucht ein am Handgelenk getragenes System, das 5 verschiedene Küchengeräte anhand des Geräusches mit einer Wahrscheinlichkeit von 94% erkennt, nur 0.55 mW. Dadurch kann es für 42 Tage mit einer nur 2 cm^3 grossen Litium-Ionen Batterie betrieben werden.

1.1. Context Recognition and Activity Recognition

1.1.1. Context-Awareness

Wearable Computing [1] envisions a future in which a mobile computer is not just a smaller version of a desktop computer that allows us to do office work while being en-route. Instead, a wearable computer is an integral part of our everyday outfit, always operational and equipped to assist us in dealing with a wide range of situations. Applications may range from tourist guides [2], through remembrance agents [3] to health monitoring equipment [4].

A system that can perform all these tasks needs to be context aware. Context-awareness can be described as the ability of a system to model and recognize what the user is doing and what is happening around him, and to use this information to automatically adjust its configuration and functionality. Abowd defines context-awareness as follows:

> *"A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task" [5].*

For further literature studies on context-awareness we refer to [5–7].

While some of the context information can be retrieved from sensors not carried by the user, e.g. from beacons in the environment for location information, many applications require sensors to be placed on the user's body or in his clothes, e.g. accelerometers for motion recognition or sensors to monitor body functions like pulse and blood pressure. In this work we concentrate on the second method of information retrieval, i.e. the one with body-worn sensors.

1.1.2. Activity Recognition

The term *User Activity* can have various meanings. Very general, it is the state or quality of being active [8]. It can involve social activity as well as physical activity. In context recognition, activity recognition is usually associated with detecting human motion. However, the term is understood differently by different researchers: Polana and Nelson [9] use it only for periodic motions patterns like walking, running, cycling. Bobick [10] distinguishes between movements, activities and actions. In his terminology, for example, the *action* of heating milk in the microwave includes different *activities* like rinsing the cup, opening the door of the microwave, pressing the power button, etc. Each activity consist of a sequence of different *movements* like raising the arm, grabbing the doorknob, swinging the door around, etc. In this work, we use Bobick's definition of activity but broaden it to include simple actions like 'operating the microwave'.

Note that this simple action does not mention the milk from the previous example.

> *User Activity Recognition is the act of detecting the motion of a user. An activity is a sequence of simple movements of the human body and its limbs. Examples include simple activities like 'walking', through more complex activities like 'opening the door', to simple actions like 'getting a coffee'. In this work, we particularly focus on activities which require movements of the user's hand or arm.*

Activity recognition can be regarded as a sub-category of context recognition since it tries to detect the context of a user by means of his actions. For further literature studies on human motion recognition we refer to [11]. In a wider sense, the topics activity recognition and context recognition are part of the research field pattern recognition.

1.2. Use of Sound for User Activity Recognition

In this section, we motivate the use of sound for activity recognition. Based on everyday observations, we list examples and scenarios where sound could be a useful means to recognize user activity. Chapter 2 will then outline how to prove this hypothesis. In Chapter 2, we will also discuss the limitations of this approach as well as the circumstances under which sound-based activity recognition is feasible.

1.2.1. Why Using Sound?

Most researcher in the field of user activity detection use motion sensors like accelerometers or gyroscopes [12–16]. Alternatively, video analysis is used [17]. While computer vision corresponds closest to the way humans perceive their environment, it has many problems of its own. This includes sensitivity to light conditions and background clutter as well as large computational complexity.

Sound is rarely related to activity recognition. In case microphones are used, it involves speech or location detection. The later is based on the assumption that through an acoustic analysis of the environment the whereabouts of the user can be narrowed down to locations like a restaurant, a shop, a car, etc.

In this work, we introduce sound as an alternative source of information to detect user activity. In this case, "detecting user activity" is neither related to detecting if the user is speaking or not, nor does it mean we use sound to find out where the user is located. Instead, "user activity" is related to hand movements and the therewith associated or produced sounds. Many tools we use everyday produce a distinct sound. Even more, many of our actions are accompanied by a clear and distinct sound, be it typing on a keyboard or closing a door.

The amount of information contained in a sound signal is best illustrated by the fact that blind people can often get around using audio information alone, in many cases developing nearly a perfect understanding of the situation. Indeed, for human beings sound is the second most important source of information after vision. Therefore, it seems natural to use sound to recognize user activity. We see three advantages:

- Some activities or actions are easier to detect with sound than with other sensors. As a simple everyday example consider getting a coffee from a machine. In general, this action is not associated with a single characteristic gesture. Rather, it involves a series of hand movements such as putting the cup under the outlet and pressing a switch. These activities can be performed in a variety of ways and sequences (e.g. the water tank can be filled before the empty cup is put under the outlet or vice versa) and are thus difficult to recognize using both vision and wearable motion sensors – in particular from continuous, non-segmented data. On the other hand, most coffee machines tend to make a very distinct sound, which can be reliably recognized. Additionally, the duration of the sound determines the type of coffee (espresso, regular, double) the user has chosen: something which cannot determined with other sensors, except maybe with vision (provided that the user does not block the view) or with sensors integrated into the coffee machine.

- The sound produced by a machine or a tool is independent of the user. A user can perform an activity in a variety of ways: one or two handed, right or left handed, slow or fast, etc. An activity recognition system based on motion sensors needs to take this into account, while a sound analysis always yields the same result, no matter how the user performs the action. To some level, this is even true for sounds that do not stem from a machine.

- The third point concerns the data rates that have to be analyzed for different sensor modalities. Consider the order of magnitude estimates for the data rates, given in Table 1.1. Even though the data rates might differ if a specific application is considered, it is obvious that processing sound data is computationally less complex than video data but more complex than motion data. However, the higher computationally complexity compared to wearable motion sensors is compensated by the two aforementioned advantages.

Table 1.1. *Approximate data rates for different sensor modalities.*

Input Type	# Sensors	Sampling rate	Resolution	Data rate
Motion	10	100 Hz	8 bit	1 kB/s
Sound	1	10 kHz	8 bit	10 kB/s
Video	1	4608 kHz [a]	8 bit	4608 kB/s

[a] assuming 15 fps and VGA resolution

For obvious reasons, using sound to detect activities will not work with silent activities. Therefore, a user activity recognition system will always have to rely on other sensor modalities as well. This is not necessarily a drawback: information from motion, sound and vision sensors can be used to complement one another and consequently used to gain a complete understanding of the current context.

1.2.2. Relevant Application Domains

An analysis of several wearable application domains reveals that most scenarios contain sound sources which could be used for our approach.

Household Monitoring and Assistance It has been shown, that following user activity to monitor the elderly and ill, and assisting them to handle their daily chores, could improve their quality of life while reducing healthcare costs [18]. Relevant audio sources in this type of scenario can be found in the kitchen (kitchen appliances like mixer, microwave, coffee machine) or in the bathroom (toilet flush, water tap, brushing teeth, hairdryer, electric shaver). Other examples are telephones or vacuum cleaners. Sounds generated by the user itself by opening cupboards and doors, washing dishes, chopping vegetables, stirring something with a spoon or a fork, or rubbing on a surface to clean it, will also work.

Assembly and Maintenance Using wearable computers to provide online access to user manuals and remote experts in assembly and maintenance task is among most successful applications of wearable computing, e.g. [19–21]. In this scenario, the relevant sounds are produced by tools, machines and manual actions such as drilling, hammering or sawing.

Office Activity Understanding the activities of the user in an office environment can be important for a number of reasons including the desire to analyze and optimize workflow and online collaboration.

The main relevant audio events in the office scenario are the operating sounds of different equipment, ringing telephones, opening and closing sounds of drawers and doors or typing on a keyboard.

Outdoor Guidance Location and context based wearable guidance and assistance systems have been among the first wearable scenarios investigated, e.g. [2]. While the outdoor environment often tends to be cluttered with different non-relevant sounds, there are a number of things like passing cars or trains as well as weather related events (raindrops, strong wind) that have a clear, recognizable audio signature. Although this application domain is similar to scenarios in auditory scene recognition (see Sec. 1.4.1), the main purpose is not recognize a scene or location, but for example to warn a hearing impaired person of an approaching car.

Summarizing the above scenario discussion, it can be said that in all scenarios there are many user activities that are accompanied by a sound. The length of the sounds in questions varies from very short (e.g. the click of a switch, closing of a door) to more or less continuous (e.g. operating noises of an appliance).

1.3. Importance of Power Awareness

Wearable Computing encompasses a wide variety of systems ranging from mobile computers like PDAs, through smart badges to intelligent textiles. One important part of the wearable computing vision is that of miniaturized sensor networks seamlessly integrated in different parts of the user's outfit, including clothes and accessories such as jewelry, watches, keychains, glasses, etc.

Seamless integration in clothing and accessories invariably means that the sensor nodes have to be small and unobtrusive. For example, placing a sensor node inside a ring means that only a couple of cubic millimeters of space is available. In most cases, a wired connection to an external power source is not desirable since cables on the body are perceived as obtrusive and uncomfortable. Thus, a sensor node has to accommodate not only the sensors and electronics, but also the battery and/or a power generation system.

Table 1.2 provides an overview of how much power state of the art wearable energy-harvesting technologies can deliver. With the exception of heel-strike harvesting in electric shoes which can produce 5 to 800 mW for walking [22] or for solar cells larger than 3 cm^2 in bright light, available powers of miniaturized generators are generally well below 1 mW. Lithium-ion batteries have a volumetric energy density of 250–500 $\mathrm{mWh/cm}^3$ [8].

Table 1.2. *Comparison of energy-harvesting opportunities [22–24].*

Energy source	Power density
Ambient light[a] (outdoors)	0.3–15 $\mathrm{mW/cm^2}$
Ambient light[a] (indoors)	0.01–10 $\mu\mathrm{W/cm^2}$
Vibrations (piezoelectric conversion)	100–200 $\mu\mathrm{W/cm^3}$
Vibrations (electrostatic conversion)	5–50 $\mu\mathrm{W/cm^3}$
Thermoelectric[b]	60 $\mu\mathrm{W/cm^2}$
Ambient airflow[c]	1 $\mathrm{mW/cm^2}$

[a] includes solar cell efficiency, [b] with 5°C temperature differential, [c] 35 liters/min

This has to be set in contrast to the power consumption of sensors and processors: in continuous operation accelerometers or microphones consume 0.8–1.3 mW (cf. Table 5.1), processors in PDAs 0.5–3 W [25, 26] and microcontrollers 0.5–10 mW [27].

While a user usually accepts that he has to recharge his PDA or cell phone once a day or week, sensor nodes integrated into clothing should be fully autonomous – operating for months or years on a miniature battery. Thus, reducing the power consumption of the system should be a major design objective for wearable systems. In fact, without the ability to run sensor nodes at sub-milliwatt power levels, the concept of body worn sensor networks is not feasible.

Therefore, we postulate that in the design process for a wearable context recognition system not only the recognition rate should be considered but also the power consumption. As a consequence, our goal is not just to optimize recognition performance, but to gather as much useful information as possible while keeping the overall system power consumption to a minimum. In particular, we need to carefully consider what type of information can be extracted from what sensor with the least power consumption, and what algorithms are best suited for low power implementation. In addition, computation-communication trade-offs must be considered to determine which parts of the computation are to be performed locally on the sensor, and which must be outsourced to a central unit.

1.4. Related Work

1.4.1. Audio Recognition in Wearable Computing

Activity Recognition with sound: As mentioned in Sec. 1.2.1 sound is rarely used to detect user activity. In fact, besides the work we have previously published in [28–31], we are aware of only two groups that use sound for activity recognition[1]. Both use stationary microphones:

Istrate et al. have published several papers that deal with medical telemonitoring of the home by means of sound, e.g. [32, 33]. Microphones placed in every room of the apartment are used to spot events which need to cause an alarm (breaking of glass, falling of objects or screams) out of everyday sounds like ringing phones, footsteps, moving chairs or slamming doors. While their system detects the presence of a sound very accurately and classifies them with 90% accuracy, it is paid with high computational complexity and requires a PC to do the sound processing in real time. As a further drawback the sounds cannot be associated with a person (e.g. some of them could be caused by a pet) and therefore additional sensors like location sensors are required.

Chen et al. [34] use sound to monitor activities in the bathroom such as showering, washing hands, brushing teeth, urination and flushing the toilet. They justify the use of a stationary microphone with the bathroom environment, where users might take off their sensors, e.g. to take a shower.

Auditory Scene/Context Recognition: Auditory Scene Recognition and Auditory Context Recognition also relate audio events to activities. However, in those areas the term ‘activity’ is used in a broader sense than defined in Sec. 1.1.2. It includes activities which do not require any movements like watching TV or sitting in a restaurant.

Auditory scene recognition aims at recognizing distinct sound events through an acoustic analysis of the environment. The goal is an automatic segmentation, indexing, and retrieval of events. Auditory scene recognition is often used to discriminate TV programs like commercials, sports, news reports, and weather forecasts [35, 36] or different music styles and music instruments by classifying audio signals into categories like speech, music, song, environmental sound, speech with music background, silence, etc. [37–39].

---

[1] Here we use our definition of activity. Of course, one could argue that ‘speaking’ is also an activity and that there are many works on speaker recognition or speech detection. Furthermore, in auditory scene recognition, sound is used to detect the location the user is in. Of course, every location implicitly contains a set of activities that can or cannot be performed in it; e.g. in a restaurant it is most likely that one will drink or eat something, and not perform major physical exercises.

Auditory context recognition aims at recognizing the environment or the user's context. One possible application is the automatic adjustment of the noise canceling filters in hearing aids to account for different situations like clean speech, speech in noise, noise or music [40]. Kern et al. [41] and Clarkson et al. [42] used auditory context recognition to determine the best moment or situation to interrupt the user. Kern grouped the sounds into restaurants sounds, street noise, lecture, conversation, and a garbage class. Substantial work in classifying a large number of environmental sounds was done by Peltonen et al. [43]. They distinguished between 17 different scenes: street, nature, road, construction, bus, car, train, subway train, restaurant, supermarket, lecture, office, library, bathroom, church, railway station and subway station. Peltonen et al. were able to classify them with an accuracy of almost 70%. They also categorized the scenes into the higher level categories: outdoors, vehicles, public/social places, office/meeting rooms, home and reverberant. For the aforementioned 17 sounds, Eronen et al. [44] compared human ability to acoustically recognize different environments to machine learning performance. Not surprisingly, humans recognize scenes which contain a broad range of sounds better than a computer. Examples include restaurants, cafes or the concept of being outdoors.

Speech and speaker recognition: The most advanced research topic in audio recognition for wearable computing is speech and speaker recognition.

Speaker recognition or tracking focuses on distinguishing between different speakers. Research in this field varies from distinguishing between 'user vs. rest' [3] with a collar microphone, to attempts to separate and track different speakers in a room in a mobile setting with microphone arrays [45].

Speech recognition aims at identifying spoken works and phrases. Some examples of speech recognition in a wearable environment are given in [46, 47]. Especially earlier generations of wearable hardware outsourced the recognition process to a desktop PC, e.g. [47, 48]. Later work in speech recognition accounts for the limited performance of mobile processors [49] or even trades word accuracy for power [50].

1.4.2. Trade-offs in Wearable Computing

The design of a wearable computing system needs to account for various factors: power, performance, wearability, availability of sensors and communication channels, to name just a few. An example of a system design approach to power-aware mobile and wearable computers can be found in [51] and [52]. A more complex systematic high level approach for designing distributed wearable systems is described in [53, 54]. There, the authors focused on automated design methodologies for selecting the right set of processor, sensors, sensor location and transceiver for a wearable sensor system.

As for systems with a particular context recognition task, systematic design methodologies which explore the complete design space have not been presented so far. Usually, recognition performance is optimized while other factors like power consumption, memory requirements or communication bandwidth are considered as a constraint and not as an additional optimization parameter. Cakmakci et al. [55] present context recognition with the constraint of limited resources. Features for audio classification tasks are compared in [43] and [56], but power awareness is only mentioned on the side. Investigations on the influence of different system parameters (e.g. sampling rate, resolution) on context recognition accuracy have been made in [28, 41, 57]. There, power consumption concerns came into play by using the general assumption that 'lower sampling frequency equals less power' but a detailed investigation was missing.

On the other hand, many papers deal with power saving techniques on the hardware side without considering the complexity or performance of context recognition algorithms. For example, energy considerations at different hardware layers in miniaturized sensor nodes were investigated in the Smart Dust [58] or the µAMPS [59] project. A short review of power saving techniques will be presented at the beginning of Chapter 5.

First ideas of how to deal with the power and performance trade-off in context recognition systems were reported in our earlier work [29, 60] where the impact of sampling frequency and algorithmic complexity on recognition accuracy and power consumption was examined. The need for this trade-off is also acknowledged by others. In [61] a DSP for heartbeat detection trades number of input bits for recognition performance. Nedevschi et al. [50] present simulations for an speech recognition ASIC where they explored various trade-offs concerning recognition accuracy, energy, computational workload and memory size. In the field of context recognition, Krause et al. [62] present a trade-off analysis for a given hardware. They trade prediction accuracy for power consumption in a wrist-worn device that can discriminate between walking, running, sitting, standing and climbing or descending stairs with the help of accelerometers. Basically, Krause et al. use two parameters to achieve the trade-off: one is sampling frequency while always sampling for the same duration, the other one is duty cycle. In contrast to our sound-based activity recognition scenarios, they can assume that at any given time one of the considered activities occurs. Thus, they introduce different duty-cycling strategies like uniform or exponential spacing and spacing by learned transition probabilities; for example, if a person was walking for the last minute, it is most likely that he will continue to do so in the next second and hence the duty cycle can be lowered.

1.5. Research Objectives and Contributions

In our work we explore the novel approach to use sound to detect user activity. We discuss limitations and advantages of the approach. We prove that reasonable scenarios exist where sound is a useful means to detect activities – in some cases even superior to other sensor modalities. To prove our point, we present a detailed analysis of the recognition chain: from signal acquisition to classification result. To our knowledge, this has not previously been done in such a level of detail.

Moreover, context and activity recognition systems are usually tuned to deliver high recognition rate. However, size and battery capacity of wearable systems impose limitations to the type and amount of sound processing that can be done. The challenge is to carefully select the parameters of the recognition process to achieve on the one hand a high recognition rate and on the other hand low power consumption. Thus, our work describes advances towards the development of a power optimized recognition system: a system that is tuned during the training phase of the activity recognition process to represent a trade-off between power consumption and recognition accuracy. This is the second novelty.

To demonstrate our method for achieving such a trade-off, we present several realistic case studies. We benchmark sound processing algorithms on a wearable platform and discuss their suitability for both user activity recognition and low-power signal processing.

1.6. Outline

The remainder of this work is organized as follows:

- Chapter 2 explains our approach and gives an overview of the design process. Especially, we introduce constraints which help to break down the problem of sound-based activity recognition on a low power platform.

- Chapter 3 describes the data collection sessions that were performed as part of the case studies. Furthermore, we introduce our hardware platform on which the sound classification algorithms are supposed to run.

- Chapter 4 deals with algorithms for sound-based activity recognition that are able to run on a platform with limited processing power. We analyze the whole recognition chain from signal acquisition to classification result and show the influence various system parameters have on the recognition accuracy. We do this theoretically as well as practically with the help of the data from our experiments.

- Chapter 5 introduces the power aspect. The first part of the chapter illustrates the possibilities to reduce the power consumption of a general purpose sensor node. The second part of Chapter 5 demonstrates our approach to combine power consumption and recognition accuracy during the training phase of the context recognition system. We conclude by listing several parameters which represent the optimal trade-off for our hardware and our recognition task. Furthermore, we show the improvements achieved with our method compared to randomly selected parameters.

- In Chapter 6, we close the circle and discuss our results with respect to our initial assumptions made in Chapter 2. Finally, we conclude our work on sound-based activity recognition with a list of achievements and propositions for further research.

- Appendix A answers the question of how to deal with varying, unequal occurrence of individual classes. We apply different metrics from machine learning to our case study and compare their suitability to deal with class skew.

- Appendix B is thought as a reference and lists audio features, classifiers, classifier fusion methods and definitions in the field of acoustics.

2

Approach

This chapter explains our approach to sound-based activity recognition. In the first part of the chapter, we discuss our basic assumptions concerning the architecture of a wearable system, microphone placement, number and type of sounds to be detected and the type of hardware to be used. In the second part of the chapter, we argue why we favor an empirical approach to analyze a recognition problem. We present a short overview of the method that will be used in the subsequent chapters. This method allows to find a trade-off between recognition rate and power consumption during system training.

2.1. Starting Position and Constraints

2.1.1. System Architecture and Signal Flow

We envision the audio classification system embedded in a larger framework of a wearable context recognition system: A number of miniaturized sensor nodes are integrated into the users outfit, where they can best extract the relevant information, e.g. motion sensors on different limbs, external light sensors on the shoulders, etc. [3, 63]. The sensors provide context information to a central wearable computing node which is also responsible for sensor configuration and control (cf. Fig 2.1). Anliker et al. [53] show that the communication link determines the trade-off between wearability and power consumption: A wired system is less wearable than a wireless one, but consumes less power. As argued in Sec. 1.3, we chose a *wireless* communication link since it offers more flexibility in terms of placement of the sensor nodes.

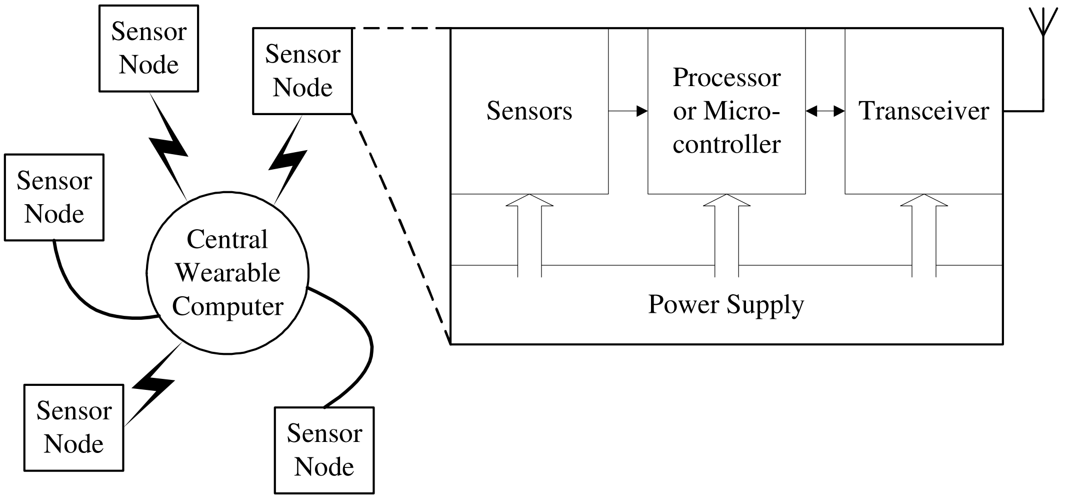


Figure 2.1. *System architecture of sensor network (left) and schematic of the Sensor Node (right).*

Furthermore, we assume that the sensor nodes are able to perform a limited amount of *local processing*. The type of processing that can be performed locally is illustrated in Fig. 2.2. One can either transmit raw sensor data, filtered and segmented data (e.g. only interesting segments), features or the classification result. Each additional step increases the average power consumption of the processor, but at the same time reduces the amount of data to be transmitted. Local processing is justified for several reasons:

- Wireless communication is generally more power consuming than computation [58, 64]. Therefore, computing a classification on the

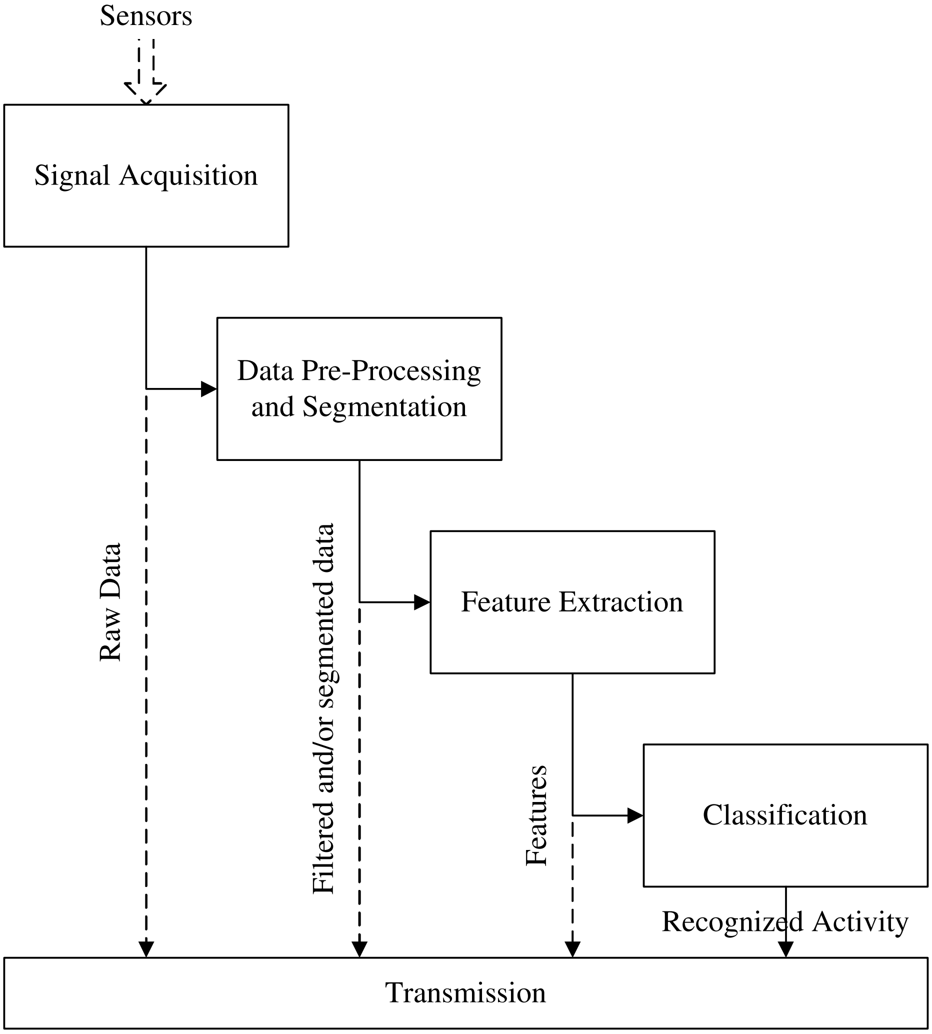


Figure 2.2. *Signal flow of the recognition process. The type of data to transmit depends on the available bandwidth, the processing capabilities of the processor and the available power (see text).*

node is more power efficient than transmitting the raw sensor data for off-line processing. This is particularly important for sound sensors due to their high sampling rate.

- Raw sensor data occupies more bandwidth than features or the classification result, thus increasing the collision probability with packets from other sensor nodes.
- In many applications there is only one single compact device (e.g. a watch) available for context recognition. Thus, calculations have to be done locally.

In this work, we focus on sensor nodes which perform the whole recognition process locally – from signal acquisition to classification. Thus, the challenge is to prove that sound-based activity recognition is feasible on such nodes.

Designing a sound-based user activity recognition system is a multi-objective optimization problem [53, 54], with an almost infinitive number of sounds to be recognized and numerous possibilities for sensor placement, configurations and platforms. Therefore, we introduce reasonable constraints and show how these constraints influence the user activity detection process. These constraints, described in the next four sections, involve mainly the microphone placement, the hardware and the number and type of sounds used. At the end of our work, in Chapter 6, we will review some of the constraints and discuss how our system would perform if they were eased.

2.1.2. Microphone Placement

Generally, important information is contained not just in the type of sound but also in the location of its source. In our case, the audio information is even more valuable if we can differentiate between sounds that occur in the immediate vicinity of the user (which are probably related to the user's activity), and sounds that occur farther away from the user.

Most activities involve some hand movements (e.g. switching on an appliance or performing a noisy manual task) and therefore a *wrist mounted microphone* predominantly picks up sounds that are caused by the user or occur in close proximity to the user's hand[1].

2.1.3. Number of Sounds

If local processing on the sensor node consists only of a pre-processing stage, e.g. a filter, then such a sensor node can be used for a very broad range of sounds since it does not require any prior knowledge of the type of sounds to be investigated. But in general, pre-processing does not reduce the amount of data to be transmitted. On the other hand, to perform high-level signal processing locally (e.g. sound classification, which is basically comparing the actual sound signature with signatures of known sounds), some a priori knowledge of the sounds is required in order to apply the optimal algorithms. Therefore, we assume that other sensors and background information are available and can be used to constrain the number of sounds. In most cases, other sensors (GPS, network location, infrared/ultrasonic beacons, RFID tags, inertial navigation, etc.) can restrict the user's whereabouts to a room, part of a room, or a particular

---

[1] Some sound-based activities might not be related to hand movements and may require different microphone positions. One example is analyzing eating patterns with the help of chewing sounds. As we have shown in [65], the optimal microphone placement for this kind of problem is in the inner ear, directed towards the eardrum.

outdoor location. A study of the scenarios in Sec. 1.2.2 showed that in most locations there are just a few (between 5 and 10) frequently occurring, relevant sounds. Thus, we focus on small groups of sounds in order to reduce the available search space.

In most everyday situations, people tend to spend considerable amount of time at a limited set of locations. When at work one would move predominantly between a few offices, the lab, and the cafeteria. Those places can be regarded as higher level context. Thus, it is possible to organize relevant sound groups into sets, with each set corresponding to a certain higher level context and being relevant during a different part of the day. As a consequence, at any given time an audio classification node contains only the signatures for the currently relevant sound group set. Whenever there is a change in high level context, the corresponding set is downloaded from the central wearable computer. Since in general such a change in high level context will happen only a few times a day, the reconfiguration of the sensor node is not relevant for the overall power consumption.

2.1.4. Hardware Constraints

Usually, three types of hardware are differentiated:

- dedicated hardware (e.g. ASIC)
- reconfigurable hardware (e.g. FPGA)
- general purpose hardware (processors)

Dedicated hardware achieves the highest performance and the lowest power consumption per operation of all three candidates. Its drawback, however, is its inflexibility. A primarily study to use dedicated hardware for sound-based activity recognition was performed in [28]. A good overview over circuit techniques for dedicated hardware architectures for sensor nodes in general is given in [59].

As the previous section has shown, the scenarios require for a certain flexibility, e.g. to adapt to a new environment with new types of sounds. Therefore, we prefer the last two types of hardware. A case study how reconfigurable hardware can be used in a wearable computing environment was conducted in [66]. In this work, we investigate the most commonly used architecture in the wearable field so far [67–73]: a miniaturized, low power node containing a set of sensors and a processor with clock frequencies in the MHz range.

Flexibility usually has to be paid with increased power consumption. The platform investigated in this work and described in detail in Sec. 3.2 presents a trade-off between flexibility and power consumption. The use of a Texas Instruments MSP430 microcontroller, instead of a processor with several hundred megahertz, lowers the power consumption, but at the same time imposes limitations on speed and the amount of data that can be handled: the processing capabilities of the microcontroller are not sufficient for continuously sampling and processing the high data rates for sound as indicated in Table 1.1. Instead, we are content with processing data on a frame by frame basis – even allowing a gap between frames. We also have to deal with limited power saving possibilities. Basically, the only available options are changing the processor frequency, switching between power saving modes of the processor, and disabling sensors and external components. One task of this work is to show what kind of sound-based activities can be recognized with what accuracy if only such a hardware platform is available.

2.1.5. Type of Sounds: Quasi Stationary Sounds

With regard to the hardware restriction and power limitations, we constrain ourselves to sounds and algorithms that do not require continuous operation of the microphones and can be used with a low duty cycle (approx. 5 to 20%). We call this class of sounds *dominant, quasi stationary sounds*.

By *dominant*, we mean that the sound in question is the loudest sound received by the system. Thus, the recognition does not have to deal with separating the relevant signal from background noise. This can be achieved with the right microphone placement (see Sec. 2.1.2 and 4.2.4).

Stationary refers to the temporal evolution and implies that the sound is essentially constant over time. This means that, except for windowing effects, the spectrum of the sound is identical in all time slots regardless of their position and length. Sound classification is thus reduced to pattern matching of the spectrum acquired from an arbitrary sample window. Neither signal segmentation nor time series analysis of the different phases of a sound (such as the phonemes of a spoken word) are required.

We speak of *quasi stationary* sounds because most relevant sounds have a negligible initial and terminal phase. They have, however, a long (at least about a second) main phase that can be described as an essentially stationary sound with added noise. The departure from strictly stationary sounds means that instead of having exactly identical spectra, different time windows from the same signal will have similar, but slightly varying spectral signatures.

The analysis of a number of scenarios in Sec. 1.2.2 has shown that many events occurring in the environment are accompanied by a loud sound that is clearly distinguishable from the background. In addition, the majority of such sounds fall within our definition of quasi stationary.

We mainly choose quasi stationary sounds since they do not require to find the start of a sound very accurately and therefore allow a low duty cycle of the hardware. As we have shown elsewhere [30], the algorithms used in this work also allow us to recognize very short sounds like opening and closing sounds of a drawer or the sound of a slamming door. However, the drawback is that the recognition process needs to run continuously to spot such short sounds.

2.1.6. Working with Isolated Events

At last, we assume that we apply our recognition algorithms onto isolated events. In other words, every time the recognition process is started, we can be sure that the user performs one of the activities from a previously defined set of activities. While this is a very efficient way to deal with the Null class (i.e. all irrelevant activities), since it simply avoids it, it is certainly not a real world scenario. However, we justify this assumption for the following reasons: Generally, the number of audio signals that can act as noise signals (i.e. covering mouth while coughing or sneezing, touching objects or clothes while walking, loud environmental noise, etc.) are too large and too unpredictable. Therefore, we do not intend to model or train a Null class based on a collection of noise signals. Instead, the approach is to detect all the relevant sounds first and classify them in a second step. Thus, the recognition system needs only to be tuned to classify the relevant sounds with high accuracy.

In terms of power consumption this process is advisable: it is not efficient to run the whole recognition process all the time. Instead, a segmentation of the input signal based on a low level signal analysis should be preferred. Different methods with various complexities have been proposed: In the simplest case, segmentation is realized with a threshold detector applied to the signal energy. In our case this approach works reasonably well since the hand, and hence the microphone, is so close to the occurring sound. In the course of this work we will discuss the threshold detector as well as an additional method based on the difference of the signal amplitudes of two microphones. More complex approaches as proposed by others are conceivable as well. A similarity search can identify segments in which the signal is very similar to trained sounds [74]. Vacher et al. [75] propose to cross-correlate two successive normalized windows. A change in the signal is indicated by the maximum of the cross-correlation falling below a threshold. Istrate et al. [32] use wavelets to detect short audio events and only consider the segment as relevant if the energy of selected wavelet transform coefficients exceeds a threshold. Furthermore, high level context information like the expected duration of a sound could be used to refine the classification result.

2.2. Analyzing a Context Recognition System

2.2.1. Need for Empirical Analysis

Besides elaborating on the interesting fact, that we can detect user activity based on sound analysis, our goal is also to show how a system can be designed that delivers high recognition rates but at the same time has a low power consumption. Obviously, most design choices favoring good recognition rates are likely to restrict power saving options. Generally, adding sensors will improve recognition but also increase the power consumption. Similarly, recognition rates tend to be improved by more advanced features and classifiers. On the other hand, such features and classifiers will be more computationally intensive and thus reducing the opportunities for using low-power modes.

In short, a power conscious design of a context recognition system is a complex multi-objective optimization problem. A particular difficulty in solving this problem lies in modeling the influence of the design choices on both recognition accuracy and power consumption. Concerning power consumption and recognition accuracy we note the following:

Power Consumption: Today, all methods aimed at modeling the power consumption of an algorithm rely on a detailed empirical characterization of the processor, e.g. [76]. In case an exact estimate is needed, either simulations [25, 77, 78] or power measurements of the target hardware running the algorithm must be performed. Methods based on simulations usually consider only the power consumption of the processor: sensors or related electronics are not simulated. Purely theoretical comparisons of the power consumption of different algorithms are largely an open research problem.

Recognition Accuracy: As far as recognition accuracy is concerned, for most relevant problems the probability density functions of the classes are not known, which is why training is required. Therefore, there is no way to theoretically model the influence of feature selection on recognition accuracy. Instead, the accepted approach is to record a representative data set, start with a reasonable choice of parameters (e.g. sampling rates, features, classifier settings) and repeatedly evaluate different variants of the system until an optimal configuration is found. The evaluation can either use abstract measures such as mutual information [54] or the actual recognition rates.

The above means that a combined power consumption and recognition rate optimization must heavily rely on an empirical approach since accurate models are lacking.

2.2.2. Method Overview

In this work, a systematic, empirical design process to optimize a context recognition system with respect to the 'recognition rate versus power consumption' trade-off is described. Basically, it *enhances* the application specific *training* already known from context recognition *with a power optimization.* This is a novel approach in the design of context recognition systems. The process takes into account a variety of parameters such as sensor choice, sensor operating parameters, feature choice and the use of different classifiers. We evaluate the impact of these parameters on the recognition accuracy as well as the power consumption.

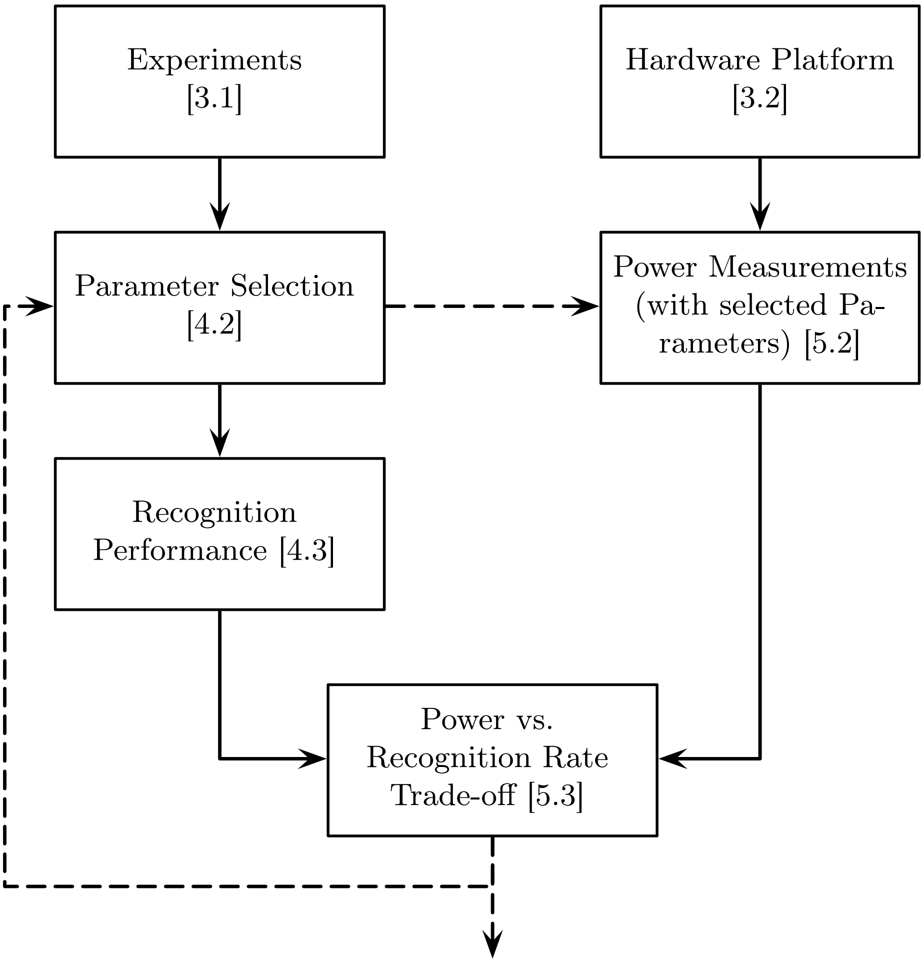

Experiments [3.1]
Hardware Platform [3.2]
Parameter Selection [4.2]
Power Measurements (with selected Parameters) [5.2]
Recognition Performance [4.3]
Power vs. Recognition Rate Trade-off [5.3]

Figure 2.3. *Empirical design process in context recognition systems. Numbers in brackets refer to the section in which each topic is discussed.*

The process in depicted in Fig. 2.3. We start with two basic requirements: a hardware platform and high quality data from an experimental setup (Chapter 3). The experimental data is used to calculate the recognition performance (Chapter 4), whereas the hardware is used to measure the power consumption (first part of Chapter 5). The results are combined to get a detailed experimental investigation of opportunities for power savings and recognition rate enhancements in a specific context recognition problem. From this evaluation a pareto-optimal system architecture can be derived for a given set of criteria (second part of Chapter 5). The parameters of this system will form the optimal trade-off between recognition accuracy and power consumption. Finally, the feedback loop in Fig. 2.3 indicates that parameters do not have to be selected with a brute-force method: non-promising variants can be discarded early in the process, which in turn speeds up the search procedure for suitable parameters.

2.3. Summary

Designing a sound-based user activity recognition system is a multi-objective optimization problem which requires to be analyzed with empirical methods. As virtually all context recognition systems require application specific training, it is reasonable to assume that such training can be combined with power consumption optimizations. To illustrate our method, we will present case studies: they consist of several experimental data collections and a hardware platform. To make a low-power approach feasible, we made the following assumptions:

- microphones mounted on the wrist
- use of a low-power sensor node with a microcontroller
- as far as possible, local execution of the recognition process
- only a few dominant, quasi stationary sounds present

Obviously, these assumptions reduce the complexity of the problem and impose constraints on the type of applications and recognition tasks that can be addressed. However, as we have argued, the assumptions still allow us to implement many of the scenarios from the application domains discussed in Sec. 1.2.2.

3

Case Studies

This chapter introduces the case studies conducted to support our empirical approach. The first part deals with several experiments carried out to collect real-world data on which we later tested the recognition algorithms. We describe two data collection sessions performed with high resolution equipment: one which consists of several scenarios but is recorded with just one user and covers only the stationary phase of the sounds, and a second one which involves just one scenario but was conducted with eleven test subjects equipped with a wrist-worn microphone and accelerometers. In the second part of the chapter, we present the hardware of our low-power sensor node. This sensor platform will be used in later chapters to measure power consumption and execution time of various recognition algorithms.

3.1. Experiments

To evaluate our approach, we collected data in several experiments. In our subsequent work, these data collections were used to evaluate the recognition algorithms. Since part of the design methodology includes finding the optimal signal parameters (e.g. bit resolution or sampling frequency), the experiments were recorded with high resolution equipment (16 bit) and high sampling frequencies (>40 kHz).

We collected two sets of data: the first set contained four scenarios with in total 19 different audio sources. The recordings were done by one user in a static way, i.e. a microphone was held close to a running appliance. In contrast, the second set consisted of only one scenario with 5 audio sources. But this time, it included 11 users which had to turn on and off the devices, while a wrist-worn microphone recorded the occurring sounds. The first set was used to get preliminary input data for the the signal processing algorithms and to prove that the algorithms work for a variety of sounds. The second set was used to get a more accurate estimation of the algorithms' performance and to investigate the use of additional sensors. Furthermore, the second set gave insight into the algorithms' capability to deal with non-stationary initial phases of the sounds.

All sounds belong to the class of quasi stationary sounds (cf. Sec. 2.1.5) and it was made sure that the stationary phase lasts for at least one second. Most sounds stem from machines, only a few sounds were produced by using a tool (cf. Table 3.1). Thus, the considered sounds cover only a subset of sounds that can be analyzed in the four application domains described in Sec. 1.2.2: mainly machine produced, quasi stationary sounds. As explained in Sec. 2.1.5, these sounds were chosen because they do not need continuous processing and thus are suited for low-power sound-based

Table 3.1. *Sounds of Data Collection I.*

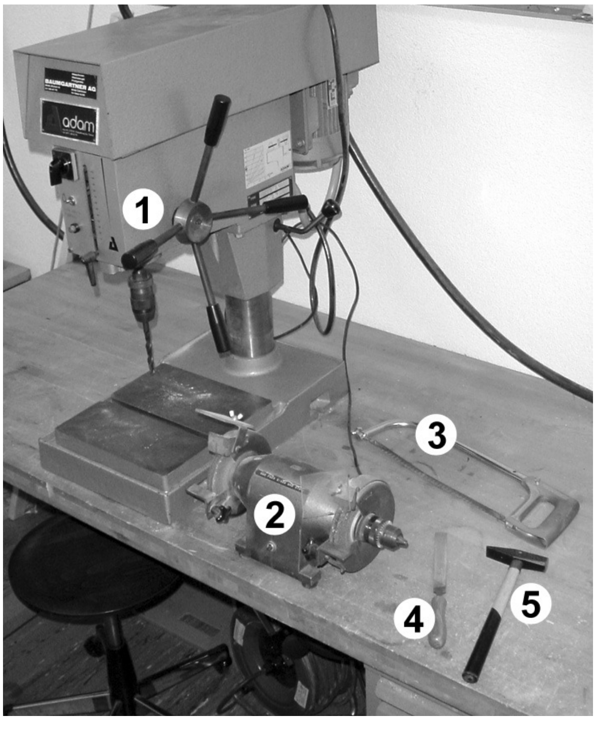
Sound group	Sounds
Kitchen	microwave, coffee grinder, coffee maker, hot water nozzle, water from water tap
Office	printer, copy machine, telephone, typing on keyboard, desk-ventilator
Workshop	sawing, drilling, hammering, grinding, filing
Outdoor	inside tram and bus, passing cars, raindrops on umbrella

user activity recognition. As proven elsewhere [30, 65], the recognition algorithms used in our work are also suited to recognize other type of sounds, e.g. very short sounds like closing of a drawer or chewing sounds.

3.1.1. Data Collection I

Scenarios: The sounds were chosen to represent the scenarios described in Sec. 1.2.2, while at the same being replicable in our lab environment. Table 3.1 lists the recorded sounds. For the Maintenance and Assembly scenario we have taken a set of sounds from a wood workshop (Fig. 3.1). This is a subset of the tasks used in [30, 31]. The Household scenario consist of the five kitchen appliances (Fig. 3.2): a microwave, a coffee maker, a nozzle that dispenses hot water and is integrated in the coffee machine, a coffee beans grinder and a water tap. We have also recorded sounds of several devices in one of our offices for the Office scenario. Finally, for the Outdoor Guidance scenario some typical sounds encountered on the way to our lab were used.

Recording Procedure: We were primarily interested in the quasi stationary phase of the sounds (cf. Sec. 2.1.5). Therefore, to record the sound of the appliances, they were turned on and then the occurring sound was



1. drilling machine
2. grinding machine
3. saw
4. file
5. hammer

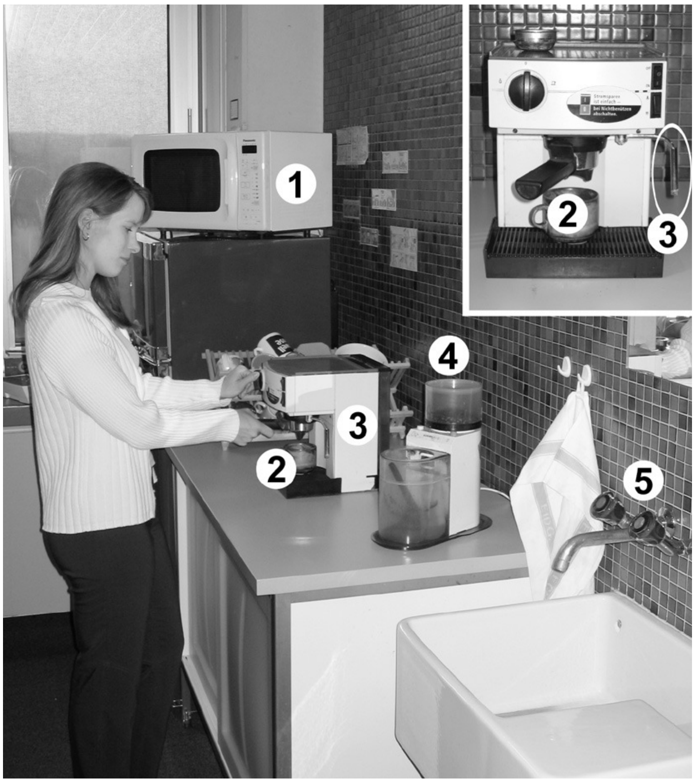
Figure 3.1. *Illustration of the workshop scenario with the devices and tools considered for the case study.*

recorded in a distance of 20 to 50 cm from the device – approximately where a user would keep his hand after turning the device on. For manual tasks in the workshop like filing or sawing, the microphone was mounted on the wrist. For environmental sounds like passing cars, raindrops on an umbrella or the noise of the desk-ventilator, a natural distance between the source and the microphone was kept. Recordings were done with 16 bit resolution and 48 kHz sampling frequency using a high quality microphone (type: Sony ECM-T145) and a DAT recorder.

Data Preprocessing: The recordings were manually cut so that they only include the relevant quasi stationary sounds (e.g. only cars passing by but no generic street noise). This resulted in about ten 10 to 30 seconds long segments for each sound.

3.1.2. Data Collection II

Scenario: For the second data collection set, the household scenario from the first data collection session was repeated. It consisted of operating the same 5 kitchen appliances: a microwave, a coffee maker, a coffee



1. microwave
2. coffee maker
3. hot water nozzle
4. coffee grinder
5. water tap

Figure 3.2. *Illustration of the household scenario with the appliances considered for the case study.*

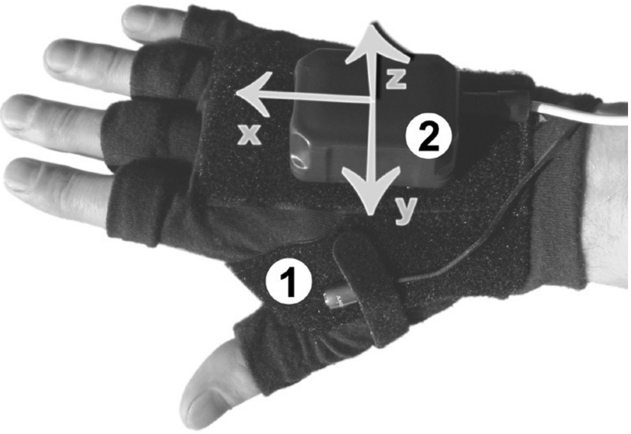


Figure 3.3. *Glove used for data collection II with microphone (1) and accelerometer (2) together with axes definition.*

beans grinder, a hot water nozzle and a water tap. In contrast to the first session, we used wrist-worn accelerometers and microphones to monitor the movement of the hand and record the occurring sound, respectively. Therefore, the recordings did not only include the sound of the appliance but also the hand movement and the ambient noise before the appliance was turned on. Furthermore, 11 test subjects were asked to participate in the experiment.

Recording Procedure: The eleven test subjects with body height from 1.65 m to 1.85 m had to switch the appliances on and off (with a reasonable time in between). Thus, we also included the initial and terminal phase of the otherwise quasi stationary sounds. However, our experiments were restricted to the sound of the appliance and the on/off motion. Other tasks like putting a cup under the coffee maker or opening and closing the microwave door were not performed. These tasks could be analyzed independently and be regarded as additional source of information. Recordings were done in two sessions per test subject. In total each test subject operated each appliance about 25 to 30 times, so that our data set contains over 300 repetitions for each appliance.

All test subjects wore a glove on their right hand. The glove had a 3-axis MT9 accelerometer from Xsens [79] and a electret condenser microphone (type Sony ECM-C115) mounted as shown in Fig. 3.3. Accelerometers were sampled at 100 Hz. The signals from the microphones were recorded at 44.1 kHz, 16 bit resolution with a commercial analog-to-digital converter (USB-Transit from M-Audio). The gain for the audio recordings was adjusted so that no clipping occurs.

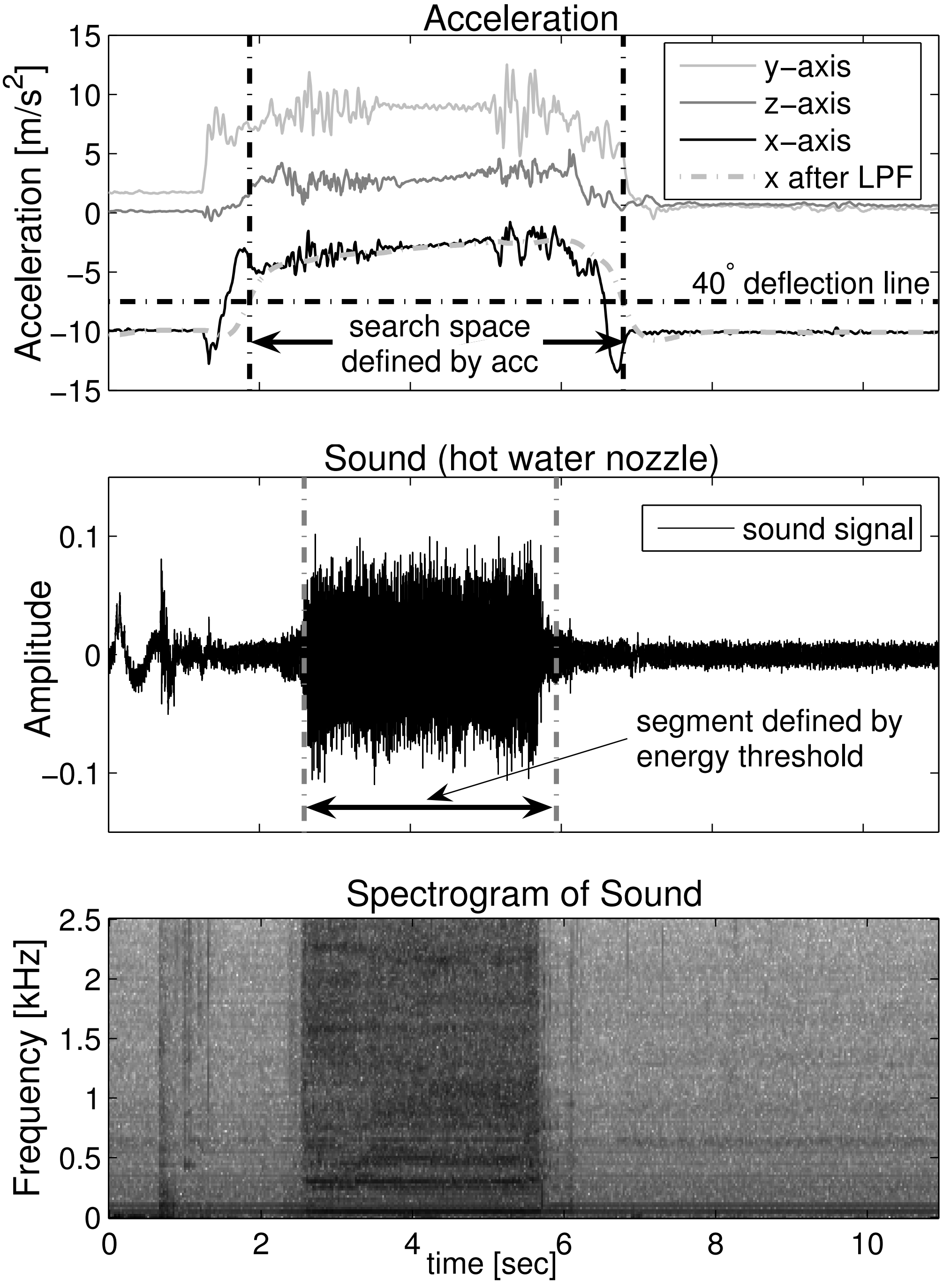


Figure 3.4. *Segmentation of data set II: Example of the 'Hot Water Nozzle' activity. Top: Acceleration signals show knob-turning to open and close the outlet. The x-axis signal is low-pass filtered and compared to the 40° threshold to define the search space for the sound signal. Middle: Within the search space, the signal energy compared to a threshold defines the interesting sound-segment. Note that the combination of accelerometers and microphone helps to exclude the audio noise at the beginning. Bottom: Spectrogram of sound.*

Data Preprocessing: The recordings described were pre-segmented by hand into pieces containing a complete action, e.g. doing nothing, raising arm, turning the appliance on, turning it off and lowering arm again. Then, both the sound and the acceleration signal were used to detect when an appliance is turned on and off. Fig. 3.4 illustrates the process: First, the x-axis of the accelerometer is used to spot interesting segments. Only deflections larger than 40° from the body axis are considered to be able to activate the appliance[1]. This helps to avoid confusions with loud input signals prior to the activation of the appliances. For this purpose, the x-axis signal was low-pass filtered with a 0.75 kHz, 2nd order butterworth filter and compared to a threshold.

Secondly, in segments in which the arm is raised, the short term energy of the sound signal is compared to an empirically determined threshold. This method detects the start point of an activity accurately since the microphone is close to the appliance when the user operates it. This method offers the advantage that it is robust to various starting positions of the arm.

3.2. Hardware

Although a variety of sensor nodes are available [68–73], we designed our own sensor platform in order to control all power consumption aspects. Originating from the *SoundButton* [29, 80], which contained just a microphone, the sensor platform – hereafter referred to as *Sensor Node* – was developed in several semester theses.

The version used in this work is described in detail in [67] and [81]. In Chapter 5, the Sensor Node will be used to get power consumption estimations and execution time measurements of various sound processing algorithms.

Fig. 3.5 and Fig. 3.6 show a picture of the Sensor Node and the system architecture, respectively. Overall, the system has a size of 41.5 × 27.5 mm^2, a thickness of 9 mm and weights 10 grams, including battery. It is composed of the commercially available off-the-shelf components listed in Table 3.2.

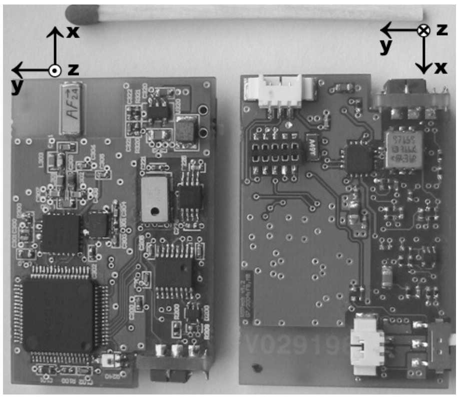


Figure 3.5. *Sensor Node*

---

[1] This is a rather conservative value. For most test subjects it was necessary to raise the arm 90° to press the various on/off buttons. But in the case of the microwave, which was placed on a low table, especially the taller test subject didn't have to raise their arm so high and therefore we chose a threshold of 40°.

Table 3.2. *Components of the Sensor Node.*

Component	Type	Manufacturer
Accelerometers	ADXL311	Analog Devices
Microphone	SP0103N	Knowles Acoustic
Light sensor	SFH3410	Osram
Microcontroller	MSP430F1611	Texas Instruments
RF transceiver	nRF2401E	Nordic Semiconductors
Battery	LPP 402025 CE	Varta Microbattery

Microcontroller: The MSP430 microcontroller with an integrated 12 bit Analog to Digital converter and 10 kByte data RAM is used as low-power processor. The 4.096 MHz clock for the microcontroller is generated by an internal digital controlled oscillator (DCO). The DCO is adjusted and stabilized by an external 32 kHz watch crystal[2].

Sensors: The Sensor Node contains three types of sensors: a MEMS microphone, two 2-axis accelerometers (one for the x and y axis, the second one for the z axis; cf. Fig. 3.5) and a light sensor, which was not used for this study. The supply voltage of the sensors can be individually toggled on and off by the microcontroller via an analog switch. Sensor signals are amplified and low-pass filtered using butterworth second order filters. Although the cut-off frequencies of the low-pass filters imply a minimum sampling frequency, we assumed that the filters could be changed in a future hardware revision. Therefore, for our subsequent power consumption analysis, we did not set a lower boundary for the sampling frequency. The maximum sampling frequency, however, is around 16 kHz[3].

Transceiver: The Sensor Node also contains a small and energy efficient 2.45 GHz transceiver. Due to a special burst transmit mode it requires only 57 nJ/bit for receiving and 26 nJ/bit for transmitting with −20 dBm output power [72], cf. Sec. 6.1.2.

Power Supply: The Sensor Node is powered by a lithium-ion battery with a capacity of 150 mAh and a size of $25 \times 20 \times 4\,\text{mm}^3$. The 3.7 V

---

[2] The design is a trade-off between size and power consumption in active and in low-power mode. If one wants to use a 4 MHz crystal oscillator instead of the DCO, one can either connect it to a second pair of oscillator-pins on the MSP (needs more space) or can use it instead of the 32 kHz watch crystal (but then the low-power mode will be clocked with 4 MHz instead of 32 kHz).

[3] The ADC of the MSP430 is capable of sampling frequencies of up to 100 ksps [82]. The value of 16 kHz stems from our requirement to shut down the MSP between two consecutive samples. Since the DCO is disabled in low-power mode, the sampling frequency is limited by the 32 kHz watch crystal which triggers the ADC.

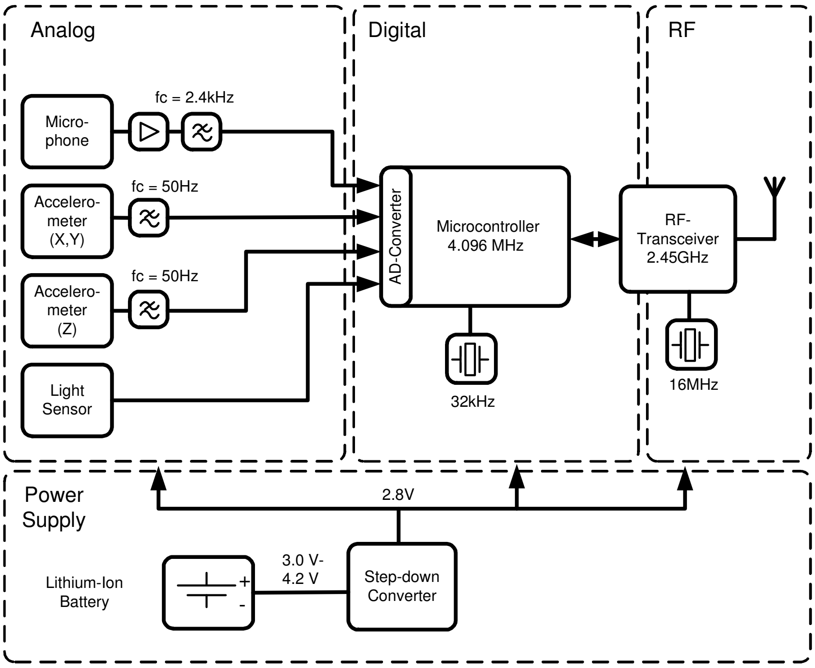


Figure 3.6. *System architecture of the Sensor Node [67, 81].*

nominal battery voltage is down-converted with a step-down converter to provide the microcontroller and the sensors with 2.8 V.

3.3. Summary

The empirical approach outlined in Sec. 2.2 relies on real-world data. For one, the recognition algorithms need to be tested for their performance on real data sets. Furthermore, power consumption and execution time of the algorithms depend on the hardware platform being used. Thus, we conducted several experiments to collect data sets and developed a low-power sensor node. The experiments were carried out in two sessions:

1. For each of the 4 scenarios in Sec. 1.2.2, one user recorded 4 to 5 typically encountered quasi stationary sounds.
2. 11 users operated 5 kitchen appliances as part of the household monitoring scenario. Audio and acceleration data was recorded in this experiment. Especially, the initial and terminal phase of the sounds produced by the appliances were recorded as well, since the test subjects had to turn them on and off. Additionally, we presented a simple procedure which allows to segment the data and find the relevant sound pieces.

The hardware represents a standard sensor architecture and contains a microcontroller, a microphone and accelerometers. Notably, it allows to switch on and off individual components.

4

Algorithms for Sound Based Activity Recognition

This chapter presents the recognition process for a sound-based user activity recognition system. The performance is validated with the experimental data from Chapter 3. By varying the parameters of the recognition process, a broad range of recognition rates is obtained. In subsequent chapters, these recognition rates will be combined with the power consumption of the recognition process to derive a pareto-optimal system architecture. The first part of this chapter gives a theoretical overview of our methods to acquire, segment and classify the sensor signals. Especially, we discuss a method which uses two microphones to segment the data stream. The second part of the chapter deals with the validation of our approach by means of pre-recorded data. Furthermore, strategies to increase the recognition performance are discussed.

4.1. Introduction

In a context recognition system, sensors typically generate a stream of data which needs to be processed and classified. As basis of our analysis, we use the signal flow diagram from Fig. 2.2 on page 14. The figure shows the main stages of a recognition system – sensors, signal acquisition, segmentation, feature extraction, classification and result transmission. As described in Sec. 2.1.1, we focus on a sensor node which performs the whole recognition process locally – from signal acquisition to classification. Thus, Fig. 2.2 can be simplified to Fig. 4.1.

In each stage of the recognition process, a variety of parameters such as sensor choice, sensor operating parameters, feature choice and the use of different classifiers influence the recognition performance as well as

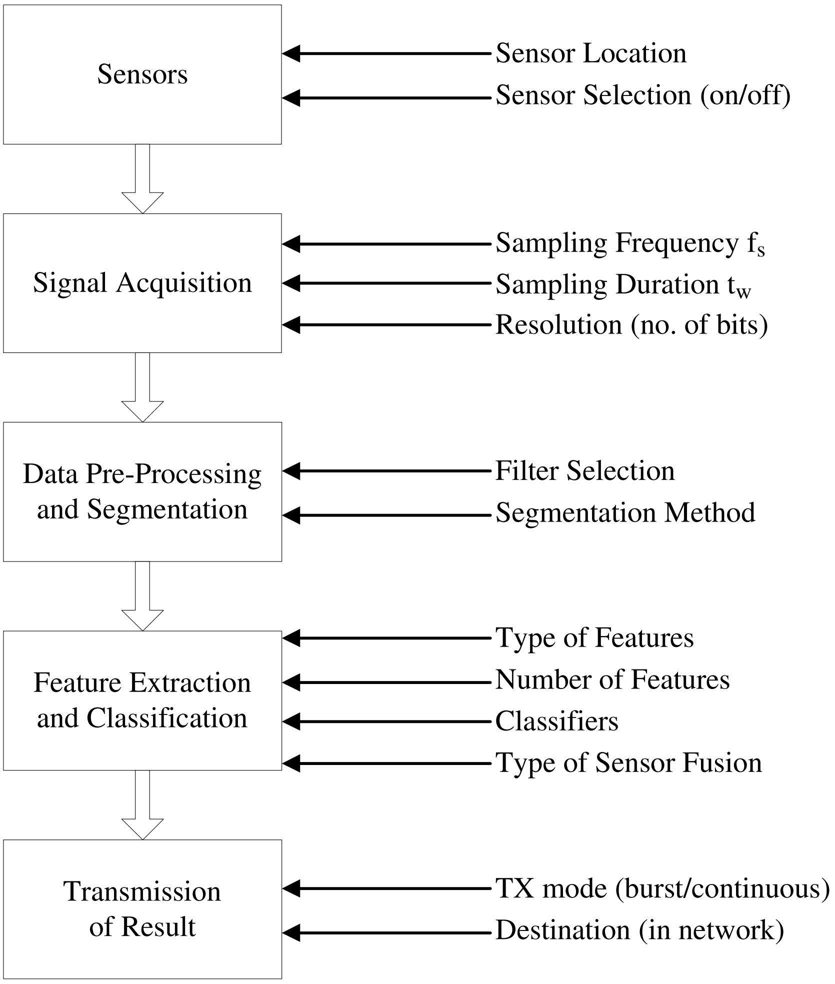


Figure 4.1. *Signal flow of the recognition process (left) and adjustable parameters (right) – cf. Fig. 2.2.*

the power consumption. The key design parameters considered in our methodology are depicted Fig. 4.1 on the right side.

In Sec. 4.2, we describe the stages from Fig. 4.1, state our approach and list potential algorithms to solve our recognition problem. Furthermore, we explain how different parameters influence the recognition rate in theory. Then, in Sec. 4.3, we apply the algorithms to the two experiments from our case study and compare their performance. Thus, this chapter covers the two boxes 'Parameter Selection' and 'Recognition Performance' from the design process depicted in Fig. 2.3.

The goal of this chapter is to show that we can indeed recognize user activities based on a sound analysis of the environment. By no means, we intend to list all possible algorithms that could perform this task. Rather, we show as a proof of concept that several algorithms perform reasonably well. Furthermore, this chapter serves as a preparation for later chapters to show that even with this small, straight-forward chosen set of algorithms a broad range of recognition rates can be achieved, which makes a careful selection necessary – particularly with regard to a trade-off between performance and power.

4.2. Analysis of the Recognition Procedure

The main problem with analyzing the different stages of the recognition process from Fig. 4.1 separately is that the choice of parameters in earlier stages depends on the algorithms used in later stages and vice versa. For example, depending on the features, different bit resolutions of the input signal might lead to the same result. In the end, recognition rates can only be given with a classifier: however, a bad classifier will undo all effort in the previous stages. Therefore, we cannot neglect this interdependency between different stages and we present a detailed analysis of the whole recognition process: from sensor selection to classification result.

4.2.1. Initial Assumption: Frame-Based Recognition

A classic approach to classification problems is the sliding window approach [11, 83]: in an observation window or 'frame' a set of features is computed. The window is then shifted by a few samples, so that it still overlaps the previous window. For continuous recognition, the feature values or the classification results from each frame are averaged to get a class prediction for a larger window [31]. However, continuous context recognition implies that the sensors are constantly working. It also involves high data rates which need to be constantly processed. Initial investigations ([84] and Fig. 5.5) based on 8 to 256 point FFTs with sampling frequencies from 0.5 kHz to 5 kHz showed that the microcontroller available on the hardware platform in Sec. 3.2 is not able to handle the amount of data needed to perform continuous sound-based activity recognition.

Since previous research [28, 29] has shown that continuous recognition is not mandatory for the activities considered in this work (cf. Table 3.1), we focus on optimizing recognition rate and power consumption on a frame basis. This approach works because the considered activities occur at a timescale of several seconds and the sounds show a *quasi stationary* behavior during that time [28]. Achieving high recognition rates on the basis of a single frame or a few frames makes continuous recognition superfluous. This is supported by the results in [62]: The authors showed that for simple motion based activities like walking, running, standing, sitting, etc., the classification accuracy did not drop significantly if only every second frame (of otherwise consecutive but non-overlapping frames) was processed.

4.2.2. Sensor Placement and Sensor Selection

In contrast to other works, which try to find an optimal sensor placement for a given task [53, 54], we consider the placement and type of the sensors as unchangeable. As stated in Chapter 1 and 2, we work with a wrist-worn microphone to detect 'noisy' activities caused by a hand movement. Indeed, since sound pressure is inversely proportional to the distance from its source (cf. Appendix B.4), a microphone mounted on the wrist can be used to detect sounds that occur in close proximity to the user's hand. In addition to the microphone, we investigate the use of wrist mounted accelerometers to support the decision making process. Of the two signal types, sound is regarded as the primary source of information since the sounds originate from the appliances and are thus independent of the user. In contrast, the movement to turn a device on might be user dependent. Furthermore, in Sec. 4.2.4 we will analyze the use of a second microphone to segment the data into interesting pieces.

4.2.3. Signal Acquisition

In the signal acquisition phase of Fig. 4.1, analog sensor signals are digitized with the help of an analog-to-digital converter. For further processing, the data is grouped in frames with duration $t_w = N/f_s$, with sampling frequency f_s and number of samples N.

As argued in [28] and [57], lowering the sampling frequency and bit resolution can reduce the power consumption of the system while still giving acceptable recognition rates. Generally, one tries to keep the sampling frequency of the input data as low as possible to avoid high frequency noise being added to the signal but at the same time high enough so that all the relevant information is preserved.

N and t_w are mainly limited by the memory of the microcontroller and by timing constraints of the application (i.e. the maximum allowable duration for recognition result calculation). The minimum N is determined by the features, e.g. calculating a zero-crossing-rate with just 2 points is useless. The bit resolution is limited by the AD-converter on one side, and on the lowest signal level on the other hand (assuming that no gain control is available). Further statements require a knowledge of the signals and are therefore given Sec. 4.3, which covers the data interpretation.

4.2.4. Data Pre-processing and Segmentation

This stage (cf. Fig. 4.1) typically consists of low-level operations such as smoothing or filtering the data. In our case, we skip the pre-processing for two reasons. Firstly, we assume that the signals have already been low-pass filtered by adequate analog filters and secondly, that this step can be combined with the feature extraction stage. So for example, applying a Hanning window prior to calculating FFT based features can as well be regarded as a feature calculation step.

Under segmentation we understand the partitioning of the data in interesting and uninteresting segments. A simple method to detect the start of an activity using a microphone and one accelerometer axis has already been presented in Sec. 3.1.2. Additionally, we introduce a method which uses two microphones.

RMS analysis with two microphones

As stated in Sec. 2.1.2, a wrist-worn microphone predominantly picks up sounds that are caused by the user or occur in close proximity to the user's hand. However, this method does not allow to distinguish between quiet sounds close to the hand and loud sounds farther apart. To estimate the distance to the sound source, the sound amplitude levels at two different (body) locations need to be compared.

For a free wave in air, the recorded sound amplitude is proportional to the sound pressure which in turn is inversely proportional to the distance from its source (cf. Appendix B.4). In general, two microphones A and B placed at different locations on the body will have different distances to the sound source. Assuming that the sound source is located at a distance r_A from the first microphone and $r_B = r_A + \delta$ from the second, the ratio of the short term RMS is:

$$\frac{rms_A}{rms_B} = \frac{r_B}{r_A} = 1 + \frac{\delta}{r_A} \tag{4.1}$$

with the root mean square rms of the raw microphone data defined as in equation (B.2) on page 115.

For sound sources located far away from both microphones (and thus from the user), r_A will be larger then δ (since δ cannot be larger than the distance between the body locations on which the microphones are placed). As a consequence, the quotient in (4.1) will be close to 1. On the other hand, if the source is very close to the first microphone we have in general $r_A < \delta$ and with it $rms_A/rms_B \gg 1$.

Thus, putting the first microphone on the wrist and the second one on the chest, we can use a large quotient as an indication that the sound was generated close to the user's hand. Therefore, the rms comparison provides a means to filter out background noise, i.e. audio sources that are located away from the user's hand but are picked up by the wrist microphone anyway. Nonetheless, there are disadvantages associated with this method:

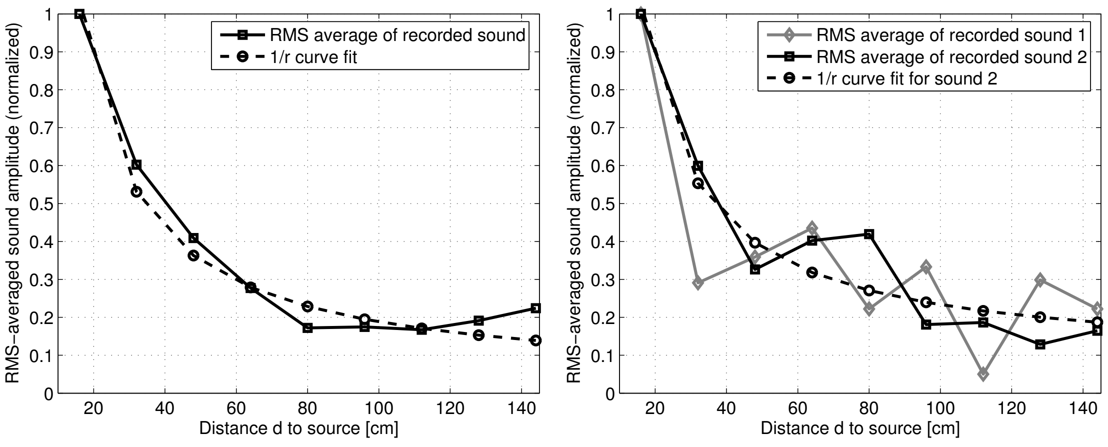
- As a premise for equation (4.1), the gain of the two microphones has to be same. This implies that techniques to control the gain automatically cannot be implemented unless the gain control of the two microphones can be coupled.

- The rms needs to be constantly calculated and transmitted to the second sensor node, so that the comparison in equation (4.1) can be performed. Assuming that a 16 bit rms is calculated over 50 ms non-overlapping windows, a bit rate of 320 bit/s is generated. Depending on the amount of local memory, transmitting in burst mode is not possible. E.g. with 5 kHz sampling rate and one transmission every second, 5000 samples need to be stored before a decision about a the usefulness of the data can be made.

- We assumed that the $1/r$ relationship (cf. equation (B.27)) is true in a real world scenario. However, measurements with a small loudspeaker[1] emitting a 1 kHz sine indicate that the microphone position (microphone not pointed directly at the loudspeaker but rotated by up to 5 degrees) and the environment (reflections on furniture) influence this behavior. Fig. 4.2(a) shows the ideal case. In most cases the results resembled the two examples depicted in Fig. 4.2(b). We observe that a higher signal amplitude might be received farther away from the source than closer to it (e.g. compare points at 50 and 60 cm).

---

[1] PC loudspeaker with membrane diameter 5 cm, placed at least 1 m away from each wall in a $4 \times 3.5\,\mathrm{m}^2$ living room.



(a) *ideal case: rms* $\propto 1/r$ **(b)** *environment influences ideal case*

Figure 4.2. *Measured sound amplitude (rms averaged) of a 1 kHz sine in function of the distance to the source. (a) ideal behavior expected for free field. (b) most measurements show local deflections from the ideal behavior due to reflecting surfaces and non-ideal microphone alignment.*

Further Segmentation Possibilities

It is worth mentioning that segmentation can be done at various points in the signal flow (cf. Fig. 4.1):

Hardware pre-processing: One possibility is to take advantage of hardware pre-processing prior to the signal acquisition stage. Under *hardware pre-processing* we understand low-power analog circuits which can wake up the processor. The segmentation method presented in Sec. 3.1.2 could for example be replaced by a tilt-sensor mounted at the right angle to detect raising of the arm [85] and a threshold detector which monitors the amplitude of the microphone signal and works similar to a voice activated switch. The advantage is that a recognition system only needs to react to events which are labeled as “potentially interesting” by the pre-processing engine and can be shut down afterwards and therefore save power. Especially, our frame-based recognition approach would benefit from hardware based segmentation since interesting segments can be detected without the need to turn on the microcontroller.

Classifier fusion: Another possibility, based on classifier fusion, was investigated by my colleague J. Ward in [31]: microphone and accelerometer data are classified continuously but independently. If both classifiers agree, it can be assumed that an interesting segment was found. The disadvantage of this approach is that it requires the recognition system to run continuously.

4.2.5. Feature Extraction and Classification

This stage (cf. Fig. 4.1) aims at finding features which best reduce the signal to context-relevant information. It also includes a feature subset selector which attempts to remove irrelevant and redundant features. The introduction of features and feature selectors leads to data reduction. Finally, a classifier associates the feature vector to a known class based on a specific decision rule learned during system training.

Features

In the past 10 to 15 years, many features for context recognition tasks have been proposed [11, 34, 39, 40, 43, 56, 86] – many of them originating from research in speech recognition. Of those, we considered the most commonly used ones – under the constraint that they can be reasonably implemented on the microcontroller. Table 4.1 lists the features; the mathematical definitions can be found in Appendix B.1. We did not include the standard feature of speech and music processing – mel-frequency cepstrum coefficients (MFCC) – even though this feature is sometimes used in auditory context recognition [32, 34, 43]. Firstly, we do not intend to emulate the human hearing process nor the speech process. Secondly, with the cepstrum coefficients (CEP) we have a representative of the cepstral methods in our feature list that performed just as well as MFCC for auditory context recognition in [43]. Since the complexity of CEP is high (requires two FFTs), it should be avoided for a low power implementation whenever possible. Furthermore, we assume the sounds to be stationary. Therefore, wavelet transformations were not used either.

For the microphone data, features in the temporal domain and in the frequency domain are considered. The frequency domain features are based on the magnitude of half of the number of the Fourier components $|F[0]| \dots |F[\frac{N}{2}-1]|$, retrieved from a N-point FFT. In our test runs, similar results were achieved with or without applying a Hanning window prior to the fourier transformation. This is due to the stationary nature of the sounds. Especially for $N \leq 32$, we even noticed a degradation in recognition rate when using a Hanning window. All features are scalar, except for FFTcomp (dimension: $\frac{N}{2}$), BER (dim: 4) and CEP (dim: 6). For the acceleration data, only the y and z-axis are considered since the x-axis is already used as a trigger to start the classification process (see Sec. 3.1.2).

Most features, i.e. zcr, mcr, fluc, BW, FC, FLUC-S, SRF and BER, offer the advantage that they are immune to linear scaling of the input data $\mathbf{x} \rightarrow \alpha\mathbf{x}$. Of the cepstrum coefficients only the first one is subject to input data scaling. Mean, std and rms scale linearly with the input data. The fourier coefficients FFTcomp also scale with the input data. There-

Table 4.1. *List of considered audio and acceleration features in time and frequency domain. See Appendix B.1 for mathematical definitions.*

		Feature	Abbreviation
sound	time	Zero-crossing rate	ZCR_{mic}
		Fluctuation of amplitude	FLUC_{mic}
	frequency	Fourier coefficients	$\mathrm{FFTcomp}_{mic}$
		Bandwidth	BW_{mic}
		Frequency Centroid	FC_{mic}
		Fluctuation of amplitude spectra	$\mathrm{FLUC\text{-}S}_{mic}$
		Spectral Rolloff Frequency	SRF_{mic}
		Band Energy Ratio (4)	BER_{mic}
		Cepstrum coefficients	CEP_{mic}
acceleration	time	Mean	mean_{accY}, mean_{accZ}
		Standard deviation	std_{accY}, std_{accZ}
		Fluctuation of amplitude	fluc_{accY}, fluc_{accZ}
		Root mean square	rms_{accY}, rms_{accZ}
		Zero Crossing Rate	zcr_{accY}, zcr_{accZ}
		Mean Crossing Rate	mcr_{accY}, mcr_{accZ}

fore, the input data $\mathbf{x}$ should be normalized. We found that normalizing with $\frac{1}{N}\sum_{i=1}^{N}|x[i]|$ works equally well as the usually used rms. This helps to lower the power consumption since the square and the root operation is omitted. Of course, normalizing with the sum of FFTcomp would also work [87].

Feature Selection

The feature selection process selects a set of features from the available features. The goal is to reduce the dimension of the feature space while preserving as much discriminatory information as possible. Optimal feature sets contain features that are highly correlated with the class and show a low intercorrelation between the individual features. Intuitively, a set with more features will achieve higher recognition rates. So for example, all multi-dimensional features investigated by Peltonen et al. [43] performed better than the scalar ones. On the other hand, a large feature set may contain redundant features and require more training data to train a general valid classifier. Since a complete overview is beyond the scope of this work, we refer to [11, 88–90] for further literature studies.

Here, we investigated two methods:

- Discarding features with little discriminatory ability: This includes methods based on mutual information (MI) [90] as well as correlation-based [91] feature selection methods. For data set I, we used a simple co-variance matrix to find correlated features. In data set II, we used the more sophisticated CFS (Correlation-based Feature Selection) method which also considers how well a feature represents the class [91] and is implemented in the Weka toolbox [92].
- Linear transformation of the feature vector: With this method there is no need to choose between two or more correlating features. Instead, the feature space is linearly transformed so that the resulting features are uncorrelated and/or best represents the class. To reduce dimensionality, the least significant dimensions can be dropped. We investigated two methods: Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) [93, 94]. We expect better results with the LDA since it takes the different classes into account whereas PCA only considers the properties of the features. In our experiments, the original feature set was reduced to a $M-1$ dimensional feature space with the help of a transformation matrix; where M is the number of classes.

However common those methods are, we see two disadvantages which need to be tackled in future work:

- Feature selection methods are based on the information content of the features but ignore the classifier. It has been shown that a feature set with a high information content according to Shannon's entropy [90] does not necessarily lead to an improved recognition rate [91].
- Methods that search the feature subspace for the best feature, evaluate features based on their ability to discriminate classes and their independence of other features. They do not take into account the computational complexity of the features.

We overcome the second problem by defining different feature sets with different number of features. The method that compiles the feature set assures that all sets perform well, while in a next step the set which represents the best power versus recognition rate trade-off is selected. Table 4.2 shows the resulting sets for the audio features: starting with a full set, subsets were defined iteratively according to the ranking of the features assigned by the CFS method. Compared to Table 4.1, FFTcomp$_{mic}$ is not included in the full set because all frequency domain features are based on it: our approach is to contrast the FFTcomp$_{mic}$ with the features

Table 4.2. *Audio-Feature Sets for Data Set II.*

Set	No.	Features						
F1	7	ZCR,	FLUC,	BW,	FC,	FLUC-S,	SRF,	BER
F2	5	ZCR,		BW,	FC,		SRF,	BER
F3	5	ZCR,	FLUC,			FLUC-S,	SRF,	BER
F4	3	ZCR,	FLUC,					BER
F5	2	ZCR,						BER
F6	2		FLUC,					BER

sets in Table 4.2. CEP_{mic} is not included in the full set because it is more complex to calculate than all the other features and did not improve the recognition rates in our experiments (see Sec. 4.3.2).

Classifiers

We evaluated different classifier types to classify the features: Naive Bayes, C4.5 decision tree [95], k-nearest neighbor (kNN) with k from 1 to 10 and nearest class-center classifier (NCC) – see also Appendix B.2. In the later case, the mean value of each class is used as a class center: A test point is assigned to the class associated with the minimum Euclidean distance to the class center. Since the sounds are assumed to be stationary for the observation duration, Hidden Markov Models (HMM) or other classifiers based on time series are not considered.

4.3. Results of the Case Studies

4.3.1. Feasibility of the Two Microphones Method

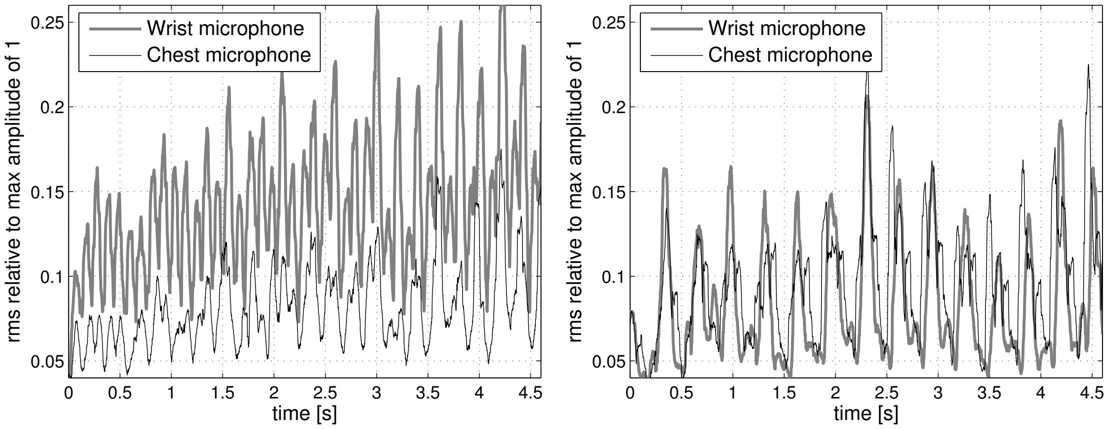
Visual Validation with Data Set I

To evaluate the feasibility of using two microphones to distinguish between sounds that occur in the immediate vicinity of the user and loud sounds that occur farther away from the him (see equation (4.1)), the workshop and the kitchen sounds were recorded using two mono microphones: one worn on the wrist and the other on the chest.

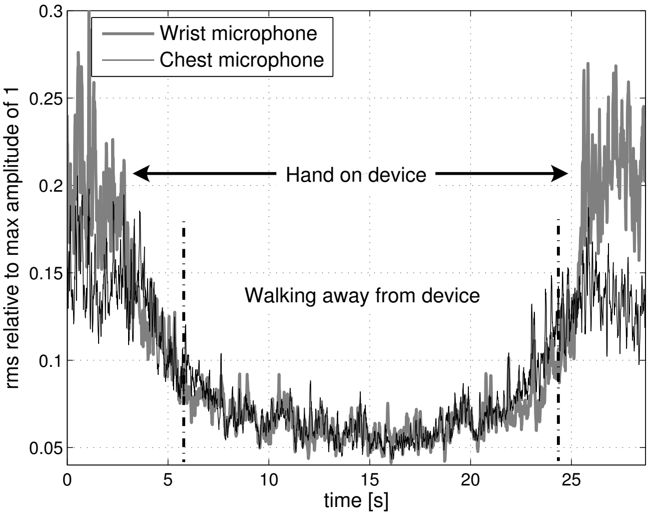
The two scenarios were selected because they are representative of two slightly different situations: (1) the user directly causing a sound through a certain motion of his hand (by using a tool e.g. sawing) and (2) the user being next to the appliance he is operating, possibly having his hand on the switch, activating/deactivating it (e.g. switching on the coffee grinder).

In all cases, the rms is calculated over a sliding window of 51.2 ms length with a sampling frequency of $f_s = 5\,\text{kHz}$.

The first type of situation is represented by the workshop sounds. Except for filing, we found that the rms of the wrist microphone is 1.5 to 2 times the rms of the chest microphone. As an example, a plot of the sliding-rms for the sound caused by smoothing a surface with sand paper is shown in Fig. 4.3(a). The curve reflects the periodic sanding motion with the minima corresponding to the changes in direction and the maxima coinciding with the maximum sanding speed in the middle of the motion. Since the user's hand is directly on the source of the sound the wrist rms is twice as large as the chest rms.



(a) *Workshop: Sanding* **(b)** *Workshop: Filing*



(c) *Kitchen: Coffee Grinder*

Figure 4.3. *rms of the wrist and chest microphone signals (sampled with 5 kHz), calculated over a 51.2 ms sliding window for three different sounds.*

By contrast, in the filing example in Fig. 4.3(b), the rms analysis does not work well for two reasons: first, the user's hand shielded the sound to the wrist microphone and second, the user was bent over the workbench bringing the chest microphone within the same distance to the sound as the wrist microphone. As a result, the wrist microphone rms is neither significantly nor consistently higher than that of the chest microphone (although a trend can be seen).

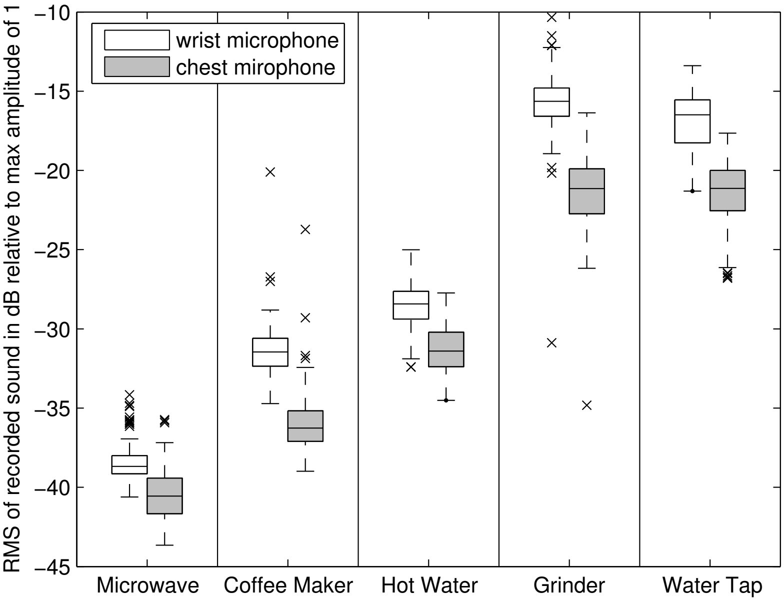
For the second type of situation, exemplified in the kitchenette scenario, we recorded a sequence starting with the user having his hand on the switch to activate the coffee grinder, then walking back approximately 3 m and approaching the device again to switch it off. Fig. 4.3(c) shows that the rms of the wrist microphone is large when the hand is on the switch, becoming equal with the chest rms as soon as the hand is removed and falling on both microphones as the user is moving away.

Statistics for Data Set II

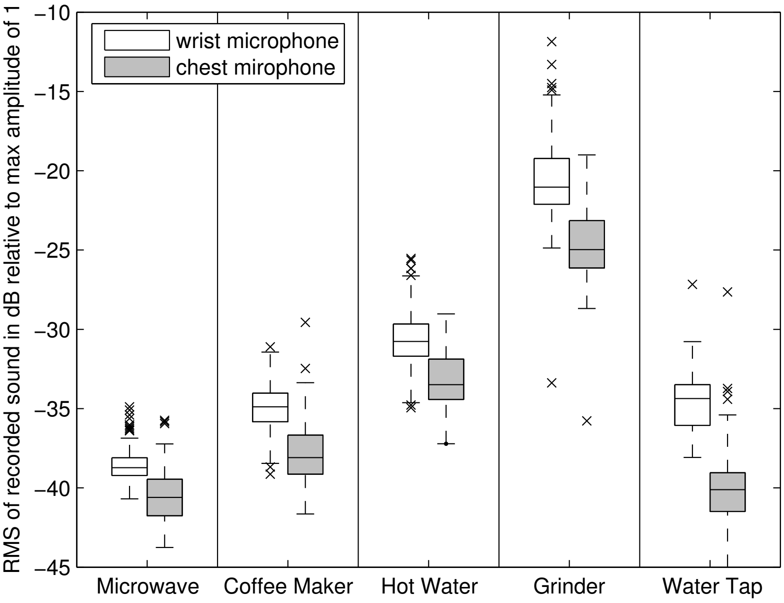
To support our thesis that the ratio of the rms from two distributed microphones can be used to spot activities related to hand movements, we analyze data set II. Fig. 4.4 displays the lower quartile, median, and upper quartile of all the rms values for all 11 test subjects and all repetitions. In analogy to the sound pressure level (SPL, cf. Appendix B.4), the rms is displayed as $20\log_{10}(rms(\mathbf{x}))$ where $\mathbf{x}$ are the sound amplitudes of a whole segment (cf. Fig. 3.4). We notice that except for the microwave, the rms of the wrist and chest microphone differ by 3 to 6dB. Fig. 4.4 is even somehow misleading, since it does not depict the segments of the wrist and chest microphone that belong together, i.e. that were recorded at the same time. In fact, for $f_s = 44.1$ kHz and looking at a whole segment, the rms of the chest microphone was only in 1.2% of all cases larger than the one of the wrist. If the rms is calculated just over a short frame of 25 to 50 ms duration instead of a whole segment, then this value increases to 8 to 10%. Therefore, Fig.4.5 depicts the ratio of the two rms according to equation (4.1) on page 37. We notice that in most cases the ratio is between 1.2 and 1.8 and therefore we conclude that a two microphone system works.

Furthermore, from Fig. 4.5 we observe that the difference between the wrist and the chest rms varies with sampling frequency especially for the coffee maker, the grinder and the water tap. Comparing Fig. 4.4(a) and (b) we notice that especially for the water tap (≈85% of the total energy above 5 kHz) and the grinder (≈60% of the total energy above 5 kHz) the energy in the high frequency bands is lost in the down-sampling process.

In an earlier work [30], we have shown that a two microphone system provides useful segmentation for the workshop sounds as well. It allows



(a) *with 44.1 kHz sampling frequency*



(b) *with 5 kHz sampling frequency*

Figure 4.4. *Dynamic range of the kitchen recordings in data set II for the wrist and the chest microphone and all 11 test subjects. The ordinate shows* $20\log_{10}(rms(\mathbf{x}))$ *where* $\mathbf{x}$ *are the sound amplitudes of a whole segment (cf. Fig. 3.4), with possible amplitude values from* -1 *to* $+1$*.The boxes show the lower quartile, median, and upper quartile values. The lines extending from each end of the boxes indicate outliers that are maximal 1.5 times the interquartile range beyond the corresponding quartile. The crosses represent outliers beyond this limit.*

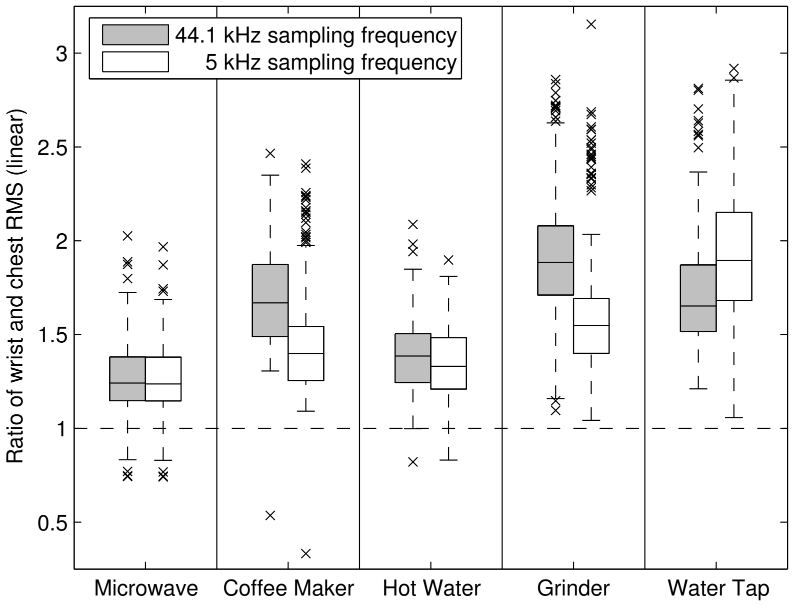


Figure 4.5. *Statistics of the ratio of the wrist and chest rms in data set II for two sampling frequencies. For each repetition of an activity, the two rms were calculated over a whole segment (cf. Fig. 3.4) and the ratio calculated according to equation (4.1). Values larger than 1 mean that the rms of the wrist microphone is larger than the one of the chest microphone. See Fig. 4.4 for the interpretation of the boxes.*

to partition a continuous data stream that contains even more sounds than considered in our case study. However, due to the continuous rms-calculation and the expected high communication overhead and the therewith associated higher power consumption, we will not use this method in our further considerations.

4.3.2. Approach Validation with Data Set I

Data set I was used to proof that the features and classifiers from Sec. 4.2 work for a broad range of applications. Furthermore, it was used to narrow down the search space for the final specification of the features and classifiers.

Performance Evaluation of Data Set I

Table 4.3 gives an overview of the 13 combinations of features and feature selectors we used to evaluate the different recognition methods. All 13 combinations were classified using a NCC, a kNN, a Naive Bayes, and a

Table 4.3. *Features and feature selectors applied to data set I.*

No.	Feature	Feature selector
1	128 FFTcomp	LDA
2	128 FFTcomp	PCA
3	128 FFTcomp	keep all
4	7 audio features	LDA
5	7 audio features	PCA
6	7 audio features	keep all
7	7 audio features	keep uncorrelated
8	7 audio features	keep uncorrelated + LDA
9	7 audio features + 6 CEP	LDA
10	7 audio features + 6 CEP	PCA
11	7 audio features + 6 CEP	keep all
12	7 audio features + 6 CEP	keep uncorrelated
13	7 audio features + 6 CEP	keep uncorrelated + LDA

The 7 audio features are: ZCR_{mic}, FLUC_{mic}, BW_{mic}, FC_{mic}, $\mathrm{FLUC\text{-}S}_{mic}$, SRF_{mic}, BER_{mic}

C4.5 decision tree classifier. The recorded data was split into a test and training set, out of which random frames with $f_s = 4.8\,\mathrm{kHz}$, $N = 256$ and $t_w = 53.3\,\mathrm{ms}$ were picked. All classes were assumed to occur with equal probability (cf. Appendix A).

Observations

Fig. 4.6 shows the resulting average recognition rates based on a 10-fold cross-validation. The 13 quadruples of bars correspond to the 13 different combinations in Table 4.3. For the kNN classifier, the maximum of 10 runs with k from 1 to 10 is displayed. We make the following observations:

- for all 4 scenarios, recognition rates $> 80\%$ can be achieved with the right combination of features, feature selectors and classifiers.
- in most cases, selecting a set of features in contrast to keeping all features does not degrade the recognition rate – the exception being the kitchen sounds (combination 6 vs. 7).
- apart from the NCC, which works only well in combination with LDA (combination 1, 4, 9), no classifier can be clearly favored.

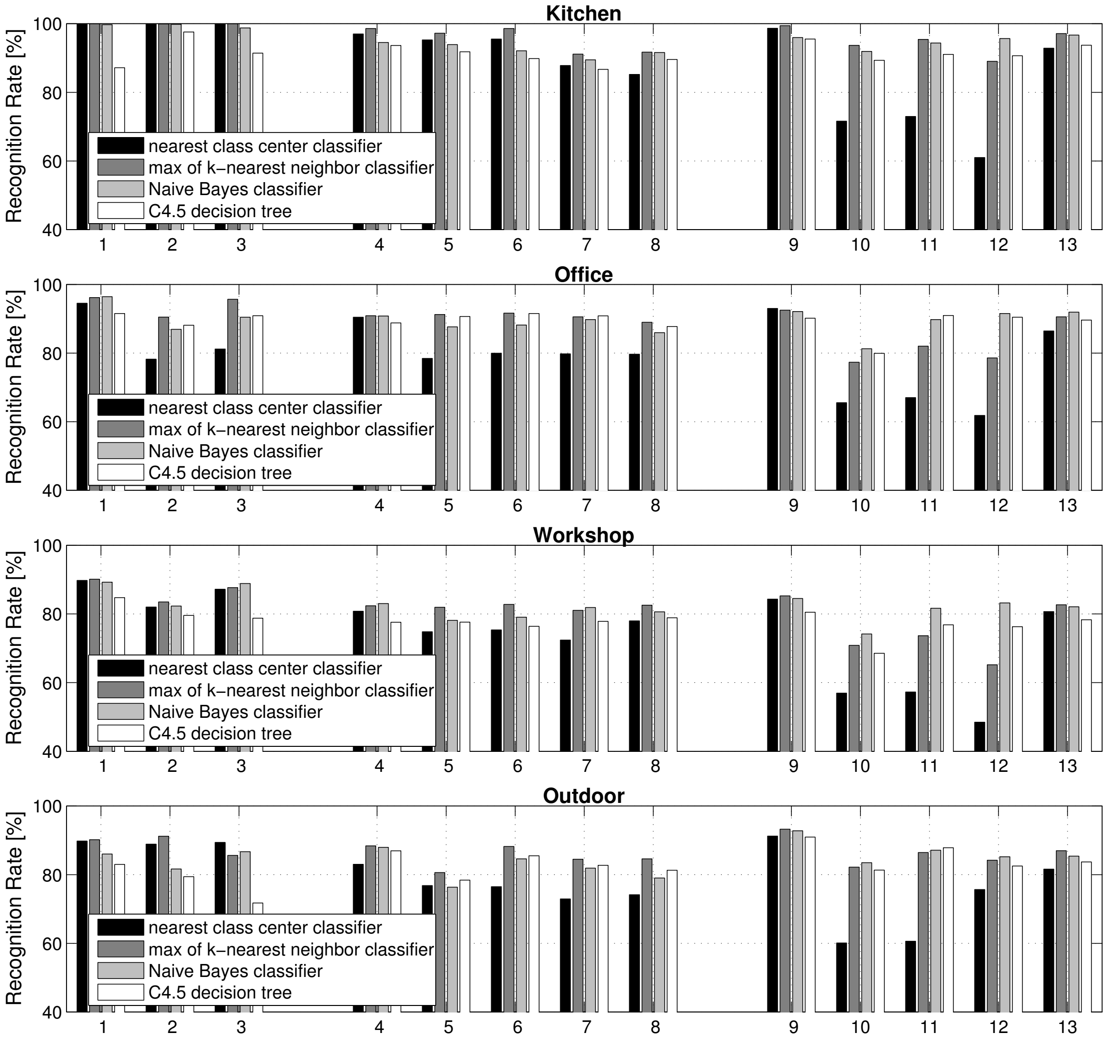


Figure 4.6. *Comparison of the overall recognition rate for different features, feature selectors and classifiers (cf. Table 4.3) for the kitchen, office, workshop and outdoor sounds, with* $f_s = 4.8\,kHz$, $N = 256$, $t_w = 53.3\,ms$.

- in most cases, PCA (combination 2, 5, 10) works worse than LDA (combination 1, 4, 9).
- there is no clear gain in using the CEP coefficients in addition to the 7 audio features (combination 9 to 13). For some combinations the recognition rate drops.
- due to the quasi-stationary nature of the sounds, the combination of fourier coefficients together with LDA performs exceptionally well.

We conclude that our approach to detect user activities with wrist worn microphones and simple frame-based context recognition algorithms is feasible for a broad range of scenarios.

Conclusions: FFTcomp/LDA vs. feature sets

We will continue to use different classifiers together with:

(a) the fourier coefficients combined with LDA, hereafter named FFTcomp/LDA method

(b) several sets of the 7 audio features, namely feature set F1 through F6 from Table 4.2

We continue with both variants, even though variant (a) – the combination of FFTcomp and LDA – performs better, because of following reasons:

- As argued in Chapter 1 and 2, recognition rate is not the only performance metric that should be considered in the design of a wearable system. As we will show in Fig. 5.4 on page 71, calculating the matrix multiplication of the LDA transformation (cf. page 42) for 5 sounds consumes 17% more power than calculating the 7 audio features (for a frame of length $N = 512$).

- As discussed in Sec. 2.1.1, the communication link needs to be taken into account as well. In Sec. 6.1.2 we will discuss the consequences of the 'communication vs. computation' trade-off on the power consumption. The impact of the communication link (between the sensor node and the central wearable computer) on the system design for variant (a) and (b) are summarized as follows:
    - a *bi-directional* link allows to store information locally on the sensor node. Thus, it makes an autonomous context recognition system (i.e. one that does all the classification online) feasible. For variant (a) the memory of the sensor node contains the transformation matrix[2] and information about the classifier (e.g. tree structure for C4.5, class centers for NCC, mean and variance of the probability density function for the Naive Bayes classifier); for variant (b) it contains information about which feature set to use and information about the classifier.
    - a sensor node with a *uni-directional* link from the node to the central wearable computer cannot adapt to new sounds. Therefore, it has to transmit its data before the classification stage is reached (cf. Fig. 2.2). The classification is left to the central wearable computer. Thus, a uni-directional link requires higher bandwidth than a bi-directional link. For variant (a) the data to transmit is the FFT components; for variant (b) the data is either a fixed set of audio features or the FFT components.

---

[2] The memory requirement of a transformation matrix for 5 sounds and 128 FFT components with 16 bit resolution is $(5-1) \times 128 \times 16\,\text{bit} = 1\,\text{kB}$ of RAM.

4.3.3. Recognition Performance of Data Set II

In this section, we test the performance of features and classifiers on our second data set. We operate with just 5 ‘parameters’: sampling frequency f_s, number of samples N in a frame, different feature sets, choice of classifiers and different frame averaging strategy (see below). Since all combinations of the parameters lead to more than 2’500 results, we show only a limited number of results which best represent the general case. However, to determine the optimal trade-off between recognition performance and power consumption in Sec. 5.3, all combinations were simulated. The other parameters of Fig. 4.1 on page 34 were determined in preliminary investigations and then fixed for the simulations of the 2’500 results.

Performance Evaluation

- Three strategies for obtaining a trade-off between power consumption and recognition accuracy will be discussed:
    1. processing one single sound frame out of one segment (segment defined by start and end of an activity, cf. Fig. 3.4)
    2. averaging features from several sound frames of one segment
    3. combining features from one sound frame with features from one acceleration frame
- Since most kitchen appliances show some non-stationary sounds for a short time after they are turned on, we skip the first sound samples of a segment. We set the start of the first frame $t_s = 0.5$ seconds after the segmentation algorithm detected the start point of an activity (cf. Fig. 3.4). Acceleration signals on the other hand are always analyzed directly after the device is turned on, because we want to capture as much of the movement as possible (e.g. turning the knob of the water tap or the hot water nozzle).
- All classes occur with equal probability and therefore just an overall accuracy is considered as performance metric (cf. Appendix A). The recognition rate thus denotes how well the system recognizes the five activities (microwave, coffee maker, hot water nozzle, coffee grinder, water tap) *on average.* Single activities might be recognized with better or worse accuracy.
- Unless otherwise noted, recognition rates are calculated using a C4.5 decision tree classifier with 10-fold cross-validation. In each fold, nine tenths of the 1545 recorded activities (from all test subjects and all classes) are used for training, the rest for testing. The training and test sets contain data from all users: this is called *user-adapted* case.

User-adapted vs. user-independent performance evaluation

Another performance evaluation is possible as an alternative to the user-adapted case: In the *user-independent* case, the recorded data from one subject is used only as test data while data from the remaining subjects is used as training data – hence the training is independent of the test-user.

We found that compared to the *user-adapted* case, the recognition rate in the *user-independent* case drops only 1 to 3% when using sound but drops more than 10% when using acceleration[3]. Table 4.4 supports our assumption from Sec. 1.2.1 that sound-based recognition is less dependent of the user than movement based activity recognition. The difference of 1 to 3% can be explained with different microphone orientations and varying background noise during the recordings.

Table 4.4. *Comparison of recognition rates in the user-adapted (U-A) and user-independent (U-I) case (see text) for data set II using a C4.5 classifier and sampling frequencies* f_s=*5 kHz and 10 Hz for microphone and accelerometer, respectively. Frame length is* t_w=*51.2 ms for mic and 1 sec for acc.*

Sensors	Features	U-A	U-I
mic	Feat. Set F1	83.6 %	81.8 %
	Feat. Set F5	77.9 %	75.0 %
acc	mean_{accY}, std_{accY}	78.3 %	67.5 %
	mean_{accY}, std_{accY}, mean_{accZ}, std_{accZ}	88.3 %	76.2 %
mic+acc	Feat. Set F1, mean_{accY}, std_{accY}	94.8 %	92.0 %
	Feat. Set F5, mean_{accY}, std_{accY}	91.6 %	88.2 %

Signal Acquisition Stage

In preliminary investigations, the limits for sampling frequency, number of samples, sampling duration and bit resolution were explored. For the case studies described in Sec. 3.1 we found that acceleration data can be processed with a minimal sampling frequency of 10 Hz and 8 bit resolution (which complies with the results in [57, 62]) but needs to be analyzed

---

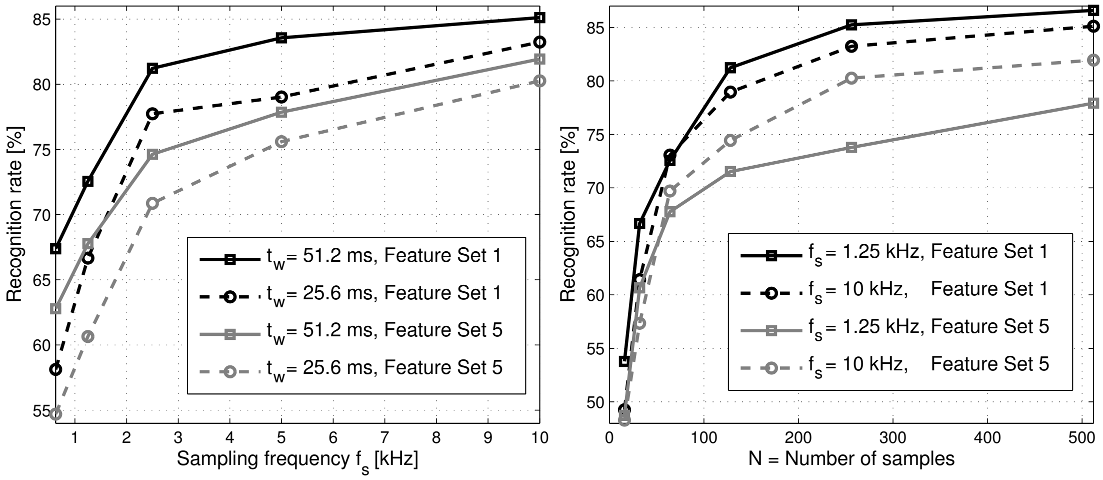
[3] When comparing percentages, the correct way of describing the difference of for example r_1=5% and r_2=10% is to say that they differ by 5 *percentage points* and that r_2 is 100% larger than r_1. However, throughout our work, we use the shorter and common notation that they differ by 5%.

for 1 second since about 10 sampling points should be used for feature calculation. Audio should be sampled with frequencies in the range of 1 to 10 kHz with at least 8 bit [28]. According to the available memory on the microcontroller, we set N to a maximum of 512 points, which would fill one tenth of the memory.

As an exemplary result, Fig. 4.7 shows the recognition rate as function of f_s, t_w, N and feature set F1 and F5 (cf. Table 4.2). The results are based on the analysis of one sound frame per activity (strategy no. 1 on page 51).

- Fig. 4.7(a) shows that the recognition performance increases with increasing sampling frequency and with a longer duration t_w of the frame. The decrease in recognition rate with a lower sampling frequency is on one hand due to the loss of high frequency fourier components and on the other hand due to the lower number of sampling points for a given t_w.

- Fig. 4.7(b) illustrates the influence of the parameters N, t_w and f_s. On the one hand, for a fixed number N of samples, lowering the sampling frequency increases the observation window t_w, which might result in a high recognition performance (Feat. Set 1). On the other hand, a low sampling frequency may cut off relevant information and therefore reduce the recognition rate (Feat. Set 5).



(a) *parameter:* f_s *and* t_w

(b) *parameter:* N *and* f_s

Figure 4.7. *Recognition rate in function of* f_s, t_w *and* N *for feature set F1 and F5 calculated over one frame with a C4.5 classifier.*

Performance of Features and Classifiers

In the following paragraphs, we will discuss the three strategies from page 51. A comparison of all three strategies is given in Fig. 4.11 on page 58 for different feature sets and for the FFTcomp/LDA approach.

1. Using one sound frame per segment: Fig. 4.8 shows a comparison of all features sets and the FFTcomp/LDA approach with different classifiers. As with data set I, FFTcomp/LDA together with a nearest class center classifier works best. Comparing feature set F1 to F6, we observe that sets with more features generally perform better, with a difference between the best and the worst set of 5 to 8%. Furthermore, we notice that the performance depends strongly on the classifier. Since feature set F1 to F6 are not transformed by LDA, the NCC classifier does not perform as good as the other classifiers. In all cases, the kNN classifier seems to be the best choice. The C4.5 classifier is the second choice in 4 out of 6 cases, followed by the Naive Bayes classifier in 2 out of 6 cases. The superior performance of kNN and C4.5 is confirmed in other works [12, 60].

However, the superior performance of the kNN classifier is only due to the large number of instances allowed to be kept for the nearest neighbor search. As Fig. 4.9(a) illustrates, the performance of the kNN drops below the one of the C4.5 classifier as soon as less than 550 randomly chosen training instances (i.e. 110 instances per class) are used. On the other hand, as shown in Fig. 4.9(b) the performance of the C4.5 classifier does not depend so strongly on the tree size. Reducing the number of leaves from a full tree to 15 leaves reduces the recognition rate by less than 5%. Therefore, we conclude that a C4.5 or a Naive Bayes classifier is the best

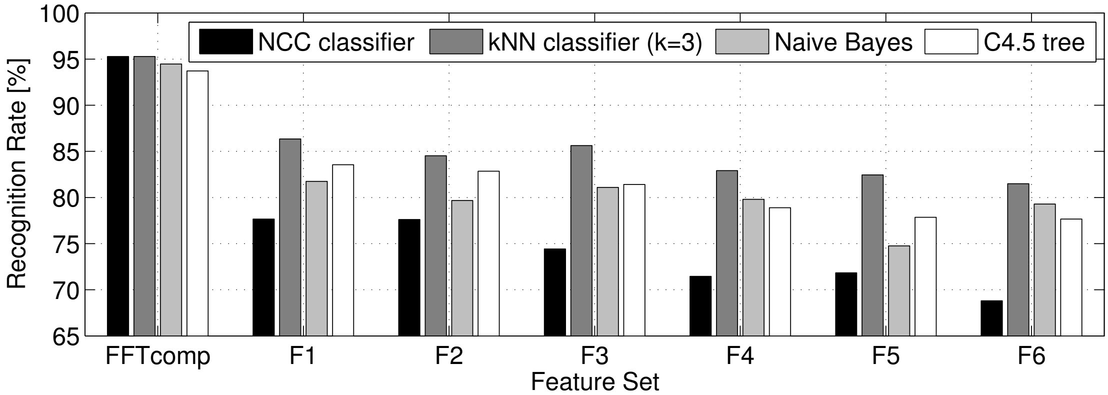
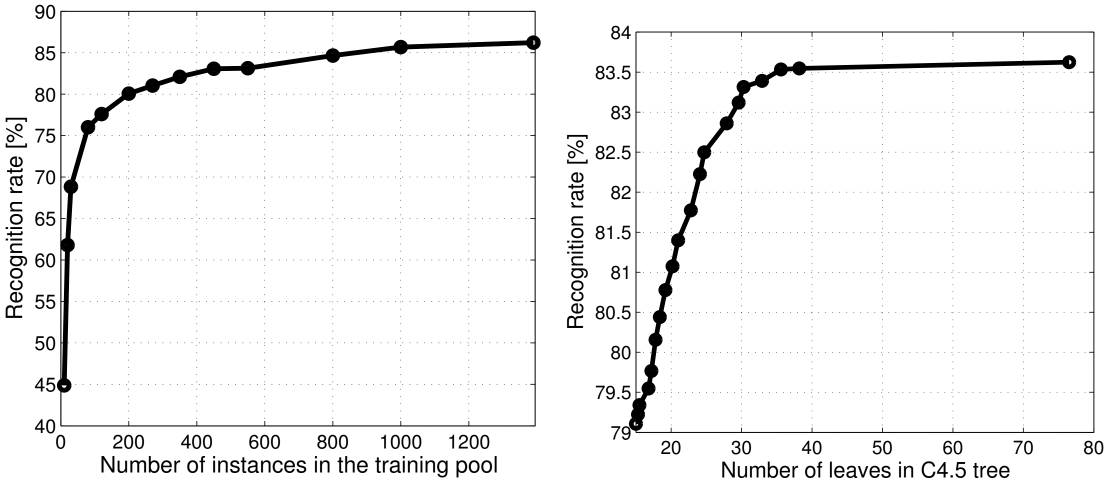


Figure 4.8. *Recognition rate for feature set F1 to F6 and for the FFT-comp/LDA approach evaluated with different classifiers. Recognition rates calculated over one sound frame with* f_s*=5 kHz,* t_w*=51.2 ms.*



(a) *kNN: recognition rate vs. number of instances in the training pool (k=3).*

(b) *C4.5: recognition rate versus number of leaves in the tree.*

Figure 4.9. *Recognition rate in function of the complexity of a classifier. Parameters: feature set F1 with* f_s*=5 kHz and* t_w*=51.2 ms.*

choice for a sensor node that uses the feature sets in Table 4.2 to recognize the kitchen scenario from Sec. 3.1.2.

2. Feature averaging over several sound windows t_w**:** In this strategy, we calculate the features of different sound frames and use the mean values of the features for classification. In the FFTcomp/LDA case, the Fourier components are averaged over different frames and then the LDA transformation is applied to the mean values.

On the one hand, the recognition rate depends on the number of frames over which the features are averaged: using more frames will increase the recognition rate (as long as the sound stays stationary) until a saturation is achieved. On the other hand, as illustrated in Fig. 4.10, the absolute value of the recognition rate depends also on the time interval between the frames. For the same number of frames, a wider spacing means that frames are taken over a broader range of the sound than with a smaller spacing. In most cases, we observe a higher recognition rate for a wider spacing: e.g. compare the point on the dashed black line at 1.9 sec (93% recognition rate) with the one on the solid black line at 1 sec (91% recognition rate). In both cases, the recognition rate is calculated over 7 frames but the recognition rate is higher for the curve with the larger interval. Exceptions exist, as shown with the black curves for averaging over 2, 3 and 4 frames.

A comparison of the 'feature averaging' strategy with the two other strategies is shown in Fig. 4.11 on page 58. In this figure we used 3 frames with the start points of the frames spaced $3t_w$=154 ms apart (and

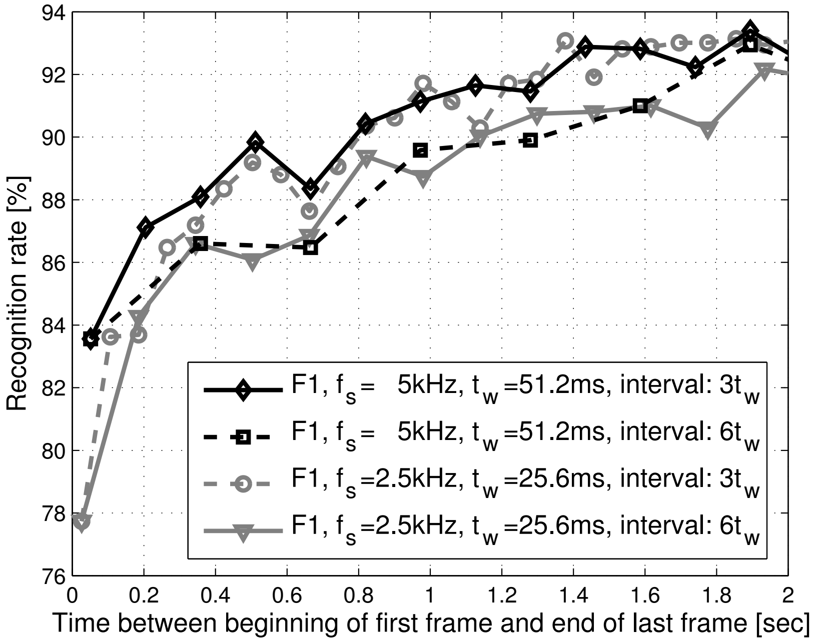


Figure 4.10. *Influence of number and interspacing of frames for 'feature averaging' strategy. Recognition rates are shown in function of the duration between the beginning of the first frame and end of the last frame, for different intervals between frames. The number of frames over which the features are averaged can be read off from the sum of markers on one line (counted from the left to the desired point; e.g. the rightmost recognition rate on the dashed black line is averaged over 7 frames).*

thus a duration from beginning of the 1^{st} frame to the end of the 3^{rd} of $7t_w$=358 ms). A gain of 5 to 10% can be achieved with this method. The difference between the best and the worst feature set is smaller than in the single frame case. Otherwise, the feature sets behave in the same way as in the single frame case. Note that in case of the feature set, the averaging strategy always stays below the combination of audio and acceleration signals. By contrast, in the FFTcomp/LDA case the averaging strategy is the better choice starting from a sampling frequency of 3 kHz. In the best case, we achieve 99% recognition rate.

3. Combining Sound with Acceleration: To combine sound with acceleration, feature sets for the acceleration data, similar to Table 4.2, need to be defined first. However, we found only one useful[4] combination

---

[4] Two criteria were used to define 'useful': firstly, a recognition rate of more than 70% and secondly, features which are easier to calculate were preferred (e.g. the combination of mean_{accY} and fluc_{accY} achieved 77.6% recognition rate, but this combination requires one division more than mean_{accY} and std_{accY} – cf. equation (B.1)).

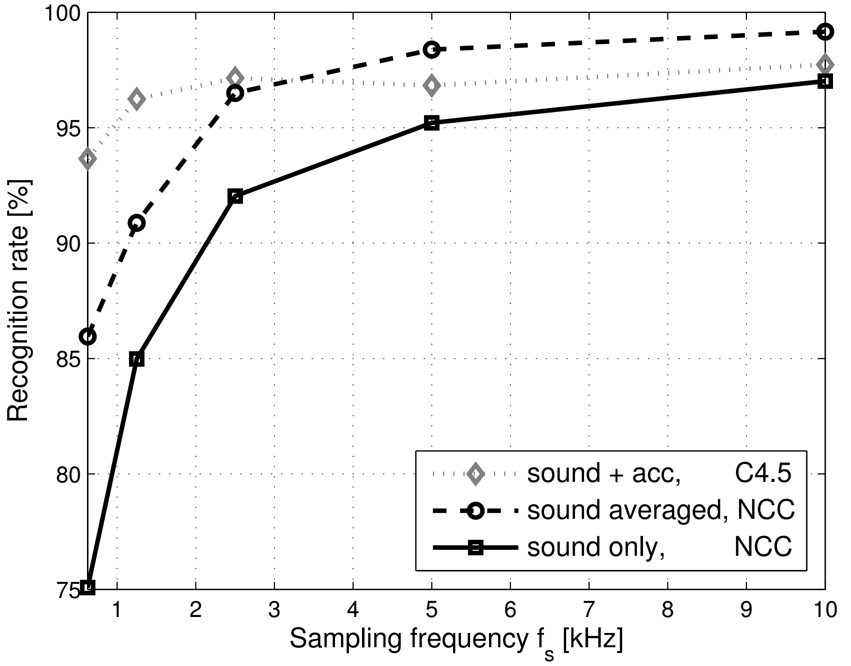
which works on a single axis: mean_{accY} and std_{accY} together (or the same features from the z-axis) achieve about 78% recognition rate. Adding more features to this set does not improve the recognition rate; using only one feature results in 50% recognition rate. Taking mean and std of both the y and the z-axis, improves the accuracy to 88% but has to be paid with 40% increase in power consumption compared to the use of just one axis because two accelerometers need to be on (cf. Sec. 5.2: assuming that they are always on and the microcontroller calculates only mean and std).

We use the *feature fusion* method to combine sound and acceleration. In this method, one classifier is trained with features from both sensors. The alternative is the *classifier fusion* method in which two classifiers are trained (one for sound and one for acceleration) and afterwards the two classification results are combined. Several approaches to combine the two classification results were investigated, namely 'Highest Rank', 'Average', 'Borda Count' and 'Logistic Regression' (see Appendix B.3 for a short description). As a further classifier fusion method, a 'probability matrix' approach was tested: in this case, the confusion matrix from the training is used as an indication of how reliable a classifier output is.

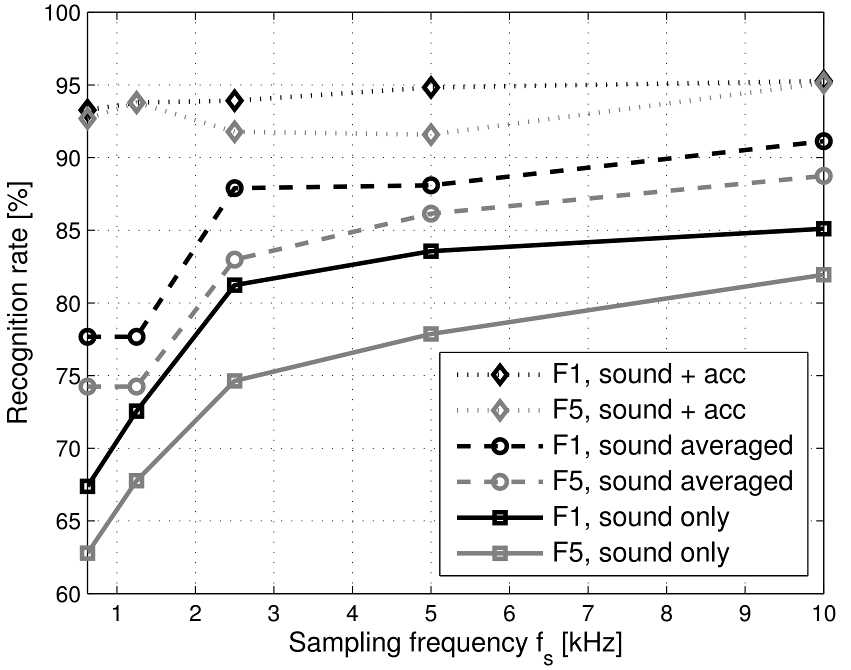
Table 4.5 compares the methods for different feature sets. Apart from the 'probability matrix' approach, which came close to achieving the same results as the feature fusion method, all other classifier fusion methods produced worse results than the feature fusion method. The reason, why

Table 4.5. *Comparison of different methods to fuse sound and acceleration data. Recognition rates are given for two different sound feature sets and different* f_s *and* N*. For acceleration:* f_s*=10 Hz,* N*=10 with features* mean_{accY} *and* std_{accY}*. Results are based on a C4.5 classifier and the performance is evaluated as described on page 51.*

Fusion Method	Set F1	Set F5	Set F1	Set F5
	f_s=2.5 kHz, N=64		f_s=5 kHz, N=256	
Sound only	77.7 %	70.9 %	83.6 %	77.9 %
Acceleration only	78.3 %	78.3 %	78.3 %	78.3 %
Feature Fusion	93.4 %	90.2 %	94.8 %	91.6 %
Highest Rank	83.2 %	81.5 %	87.3 %	84.7 %
Average of All Ranks	83.6 %	83.5 %	87.7 %	85.7 %
Average of Best Rank	84.8 %	84.0 %	88.2 %	86.2 %
Borda Count	84.3 %	84.9 %	87.5 %	84.3 %
Logistic Regression	84.3 %	86.2 %	88.7 %	85.2 %
Probability Matrix	89.3 %	87.6 %	91.7 %	88.3 %



(a) *with FFT components transformed by LDA*



(b) *with feature set F1 and F5*

Figure 4.11. *Comparison of the three strategies in function of microphone sampling frequency f_s. Single sound frame duration: $t_w = 51.2\,ms$.*

in our case ranking-based classifier fusion methods perform worse than the feature fusion method, is the small number of classifiers compared to the number of classes [96]. This, plus the fact that classifier fusion methods are more complex to calculate than a single classifier, motivated us to continue with the feature fusion method.

Fig. 4.11 shows that by adding two simple features from the acceleration signal (mean_{accY} and std_{accY}) to the sound features, the recognition rate can be increased by 10 to 20% and the overall recognition rate reaches 90 to 97%. Moreover, we observe that in this case the recognition rate is made less dependent on the sampling frequency of the microphone. We also discovered that for this strategy the choice of classifier is less critical – with the exception of the NCC classifier which delivers an inferior performance. Furthermore, we noticed that the recognition rate can be increased by additional 0.5 to 2% if more features from the acceleration signal, or the acceleration data of the z-axis are used.

4.4. Summary

Sound processing is usually associated with high data rates and high computational complexity. To make a low power, sound-based activity recognition system feasible, we work with a *frame-based* method – in contrast to a continuous recognition. This requires segmentation procedures which partition the data stream into potentially interesting segments. One technique proposed in this chapter operates with the difference of the signal amplitudes of a wrist and a chest worn microphone. With the help of the recordings from our case studies, we have shown that comparing the rms from the two microphones helps to spot activities related to hand movements in most situations.

Furthermore, by adapting various 'parameters' of the recognition process – i.e. using different sampling frequencies f_s, number of samples N, feature sets, classifiers and frame averaging strategies – we can achieve recognition rates of 75 to 95%. Two methods were analyzed in detail:

(a) a simple spectrum matching method based on LDA-transformed Fourier components

(b) calculating and classifying audio features

Method (a) outperforms method (b) by about 10% in recognition rate. Three strategies were used to evaluated the performance of the methods:

1. single-frame based sound recognition
2. multi-frame based sound recognition
3. combining single-frame features from sound and acceleration.

Strategy (3) works best with method (b), while (2) dominates for most situations with method (a). Fig. 4.11 on page 58 is representative summary of the results. As an alternative to (3), we also investigated classifier fusion strategies. However, the recognition rates were 5 to 10% lower than for the feature fusion strategy.

In addition to showing that sound-based activity recognition is feasible for a broad range of sounds, this chapter laid the foundation for our ‘performance versus power consumption’ trade-off which will be discussed in the next chapter: The combinations of parameters (like sampling rate, frame length, features, classifiers and strategies) not only cover a broad range of recognition rates, but of power consumption values as well.

5

Design for Power Efficiency

This chapter deals with the possibilities to reduce the power consumption of a general purpose sensor node. We introduce a model which allows to specify the power consumption of our sensor node by means of power and execution time measurements. The second part of the chapter illustrates our approach to combine power consumption and recognition accuracy during the training phase of the context recognition system. Our method leads to a pareto plot which combines the two conflicting design choices 'power consumption' and 'recognition rate' and allows to pick out an optimal operation point for a given application.

5.1. Introduction

The energy or power sources available to a wearable system have a direct impact on its *lifetime* and its *performance*. As illustrated in Sec. 1.3, especially for miniaturized, autonomous sensor nodes, power consumption is one of the key design issues. Thus, it is not astonishing that low power design and power awareness is a very active research field – see [97–101] for an overview. Activities range from CMOS design of low-power signal acquisition units [102] and special purpose processors (e.g. minimum energy FFT processor [103]) to the design of systems which incorporate dynamic voltage scaling [104].

In this work, we act on the assumption that the aforementioned low-power methods are not applicable for our sensor node. Thus, the possibilities for power reduction are limited:

- shutting down subsystems like sensors, ADC or network interface
- duty cycling and battery management
- minimizing instruction-level energy, i.e. low-power software design [105, 106]

We consider these techniques as a means to an end and do not intend to exploit them further than they already have. Our goal is a context recognition system which represents a trade-off between recognition accuracy and power consumption. Thus, we trade performance of a wearable system for its battery lifetime.

In contrast to some related work, the term ‘performance’ is not connected to something the user perceives directly, as for example ‘latency’ [51] or ‘video quality’ [107]. Rather, it denotes the recognition accuracy the system has been previously trained to achieve. In contrast to traditional context recognition systems, we do not trim the system to achieve the highest possible recognition rate, but try to operate it at a point which allows to get as much useful information as possible while keeping the overall power consumption to a minimum. Thus, the training of the recognition system is extended by a power analysis.

In the first part of this chapter, i.e. in Sec. 5.2, we describe the power measurements and execution time measurements of the hardware from Sec. 3.2. We avoid power estimations based on data sheets and literature studies as in [58, 108]. Instead, we rely on a more accurate method which includes modeling and measuring the hardware to synthesize the total power consumption. Then, in Sec. 5.3, we illustrate our trade-off analysis and explain what we can gain from it (compare also Fig. 2.3). In this chapter, we assume the hardware to be an autonomous node which performs all computation itself. Later on, in Sec. 6.1.2, we will discuss the additional design options in case a communication system is available.

5.2. Power and Execution Time Measurements

Power consumption measurements are only an intermediate step to our goal, the trade-off analysis between power consumption and recognition performance. Therefore, we (a) try to keep the measurements as simple as possible and as exact as necessary and (b) only present a selection of results here.

5.2.1. Theoretical Considerations on the Power Model

Today's methods to model the power consumption of a system cover all levels of abstraction: from transistor level [109], through architectural [77, 78] and instruction level [110, 111] to algorithm [76] and operating system level [25, 107]. The complexity of a model is a trade-off between its accuracy and the time and resources invested [112]. In this context, time and resources include not just the simulation process, but also the time to build a model and the availability of specifications for the hardware. Thus, many works concentrate on estimating the power consumption of processors (plus memory) for which cycle-accurate descriptions or simulators exists, e.g. for the StrongARM processor [113]. To simulate complex systems, additional estimations of the efficiency of the DC-DC converter or the capacitance of the interconnecting PCB lines are required [114].

However, for most sensor platforms such a detailed specification on architectural level does not exist, nor does a designer have the time to derive it from power measurements. Therefore, we propose a model that treats most of the hardware aspects as a black box. The model requires few measurements but is still accurate since the measurements are made on the target platform. Furthermore, we argue that the power consumption values do not need to be absolutely correct: if all values are off by the same factor or the same constant, the trade-off analysis between power consumption and recognition rate is still valid. In the remainder of this section, we will discuss our model and the assumptions on which it is built.

Battery model

A) ideal battery: We assume an ideal battery which has a constant voltage and capacity over the whole discharge cycle. The battery lifetime T_{bat} is therefore given by the capacity C, the nominal battery voltage V_{BatNom} and the average power P_{avg}

$$T_{bat} = \frac{C \cdot V_{BatNom}}{P_{avg}} \tag{5.1}$$

B) non-ideal battery: Since non-ideal batteries exhibit a voltage drop in function of the load over the course of their discharge, equation (5.1) is only a first order approximation accurate enough to get a rough indication of the battery lifetime.

Furthermore, as shown in [52, 115], $T_{bat,nonideal}$ depends rather on the peak power consumption than on the average power consumption. As a consequence, reducing the average power consumption by means of lowering the duty cycle is not as effective as by means of decreasing the maximum power consumption. However, the difference diminishes with smaller loads. In our case, the peak current at the battery terminal is less than 9 mA, C is 150 mAh and thus the load is smaller than $0.06C$. With such a slow discharge rate, this non-ideal property can be neglected and P_{avg} is a sufficient means to describe the battery lifetime [115].

Processor Model

C) architectural level: On an architectural level, the power consumption of a CMOS microprocessor can be expressed by

$$P_{tot} = P_{dyn} + P_{stat} = C_L V_{cc}^2 f_{\mu \mathrm{C}} + V_{cc} I_{leak} \tag{5.2}$$

where P_{tot} is the total power composed of a dynamic and a static component. V_{cc} is the supply voltage, I_{leak} the leakage current, C_L the total average capacitance being switched per clock cycle and $f_{\mu \mathrm{C}}$ the operating frequency [25]. If we denote t_{calc} as the time to execute a software routine, the total energy consumed by the routine is

$$E_{tot} = P_{tot} t_{calc} = C_{tot} V_{cc}^2 + V_{cc} I_{leak} t_{calc} \tag{5.3}$$

where C_{tot} is the total capacitance switched by executing the software routine [25]. Obviously, the total capacitance remains constant for any clock frequency $f_{\mu \mathrm{C}}$ and execution time t_{calc}. Consequently, the dynamic energy consumption is also constant (for constant V_{cc}). Thus, if we measure E_{tot} and t_{calc} for different clock frequencies $f_{\mu \mathrm{C}}$ at constant V_{cc}, we get a curve for E_{tot} that depends linearly on the execution time t_{calc} with the slope being proportional to $V_{cc} \cdot I_{leak}$.

For the MSP430F1611, the calculated[1] leakage current at $V_{cc} = 3\,\mathrm{V}$ is $I_{leak} = 7\mu\mathrm{A}$. In contrast to the total current of $500\mu\mathrm{A}$ at $f_{\mu \mathrm{C}} = 1\,\mathrm{MHz}$ and

---

[1] I_{leak} can be calculated with the help of two operating currents in active mode. According to [82]: $I_1 = 500\,\mu\mathrm{A}$ (at $f_1 = 1\,\mathrm{MHz}$) and $I_2 = 9\,\mu\mathrm{A}$ (at $f_2 = 4096\,\mathrm{kHz}$, both at $V_{cc} = 3\,\mathrm{V}$). The execution times are $t_{calc,i} = \alpha / f_i$ for $i = \{1, 2\}$ with α some constant. The total energy is $E_{tot,i} = V_{cc} \cdot I_i \cdot t_{calc,i}$. Now the leakage current can be calculated with equation (5.3):

$$I_{leak} = \frac{1}{V_{cc}} \cdot \frac{E_{tot,1} - E_{tot,2}}{t_{calc,1} - t_{calc,2}} = \frac{I_1/f_1 - I_2/f_2}{1/f1 - 1/f_2} \tag{5.4}$$

$V_{cc} = 3\,\mathrm{V}$ the static component, i.e. the leakage current, can be neglected in active mode and the total energy consumed by the routine is

$$E_{tot} = P_{tot} t_{calc} \approx C_{tot} V_{cc}^2 \tag{5.5}$$

Consequently, it is sufficient to measure the power and the execution time at one processor frequency.

D) instruction level: On the instruction level, the average power consumption of the microcontroller is given by $P_{\mu \mathrm{C}_{on}} = V_{cc} \cdot I$, where I is the average current. The average current I may be estimated by breaking the software code down into single instructions and using an *instruction level energy model* [105, 116]. This model has to be previously derived either from simulation based power analysis tools or from current measurements of the CPU. The current consumption between different instructions may vary significantly. However, Sinha et al. [25] have shown that the variation of the current consumption of whole programs is much smaller (38% variation between instructions vs. 8% between benchmark programs). Therefore, Sinha concluded that in a first order approximation "the current consumption of a piece of code is independent of the code and depends only on the operating voltage and the frequency of the processor". Indeed, our current consumption measurements for different software code on the MSP430 support this first order approximation. Therefore, we model I as a constant; the value measured on the Sensor Node is $I = 1.508\,\mathrm{mA}$ (see Table 5.1). Consequently, we calculate the energy consumption of an algorithm

$$E = V_{cc} \cdot I \cdot t_{calc} \tag{5.6}$$

by measuring the execution time t_{calc}.

System Model

E) duty cycling parameters: The average power consumption of a system that uses duty cycling and switches all components on and off at the same time is given by

$$P_{avg} = P_{active} \frac{t_{active}}{T_p} + P_{idle} \frac{T_p - t_{active}}{T_p} \tag{5.7}$$

with the periodicity T_p. Increasing T_p extends the idle phase (the low power phase) and decreases the average power consumption[2]. However,

---

[2] Assuming a periodic recognition process, T_p also indicates how often a classification result is delivered. Therefore, increasing T_p causes more events to be missed, which leads to a reduced recognition rate [62].

since we intend to compare different context recognition methods, we prefer to change the average power consumption by varying P_{active} and t_{active} rather than by adjusting T_p. Therefore, if not noted otherwise, the power consumption was calculated for a periodicity of $T_p = 1$ second.

F) asynchronous duty cycling: In case only one sensor is used, i.e. in our case the microphone, the average power consumption can be divided into four components:

1. microphone power consumption
2. microcontroller doing signal acquisition, i.e. AD-conversion and filling the memory with data. This occurs simultaneously to (1).
3. feature calculation and classification
4. idle phase during which the microphone, the ADC and most parts of the MSP430 are shut down: only a real-time clock, that triggers the next sampling phase, is active.

The *average total power consumption* P_{avg} is therefore given by

$$P_{avg} = P_{Mic} \cdot \frac{t_w}{T_p} + P_{SigAcq} \cdot \frac{t_w}{T_p} + P_{\mu\mathrm{C}_{on}} \cdot \frac{t_{calc}}{T_p} + P_{\mu\mathrm{C}_{idle}} \cdot \frac{T_p - t_w - t_{calc}}{T_p} \tag{5.8}$$

with t_w the sampling duration (the duration of a frame) and t_{calc} the time to calculate the features and the classification result. Measurements for (1) and (2) are covered in Sec. 5.2.2, those for (3) are described in Sec. 5.2.3.

If more than one sensor is used and the sampling duration for the sensor signals are different, P_{avg} includes more terms. For example, in Sec. 4.3.3 we assumed that the accelerometers are sampled for one second: for a short time all sensors are sampled, then the microphone can be turned off and the sound features can be calculated while the accelerometers are still being sampled.

G) Summary: Based on assumptions A to F, we define equation (5.8) as the model of our sensor node, with the parameters of the model derived from measurements on the hardware. We need to measure P_{Mic}, P_{Acc}, P_{SigAcq} (for different sampling frequencies), $P_{\mu\mathrm{C}_{on}}$, $P_{\mu\mathrm{C}_{idle}}$ – or the corresponding currents – and t_{calc} in order to calculate the average power consumption P_{avg} according to equation (5.8) and the battery lifetime T_{bat} according to equation (5.1). t_w depends on N and f_s via $t_w = N/f_s$. Given are periodicity $T_p = 1\,\mathrm{second}$, battery capacity $C = 150\,\mathrm{mAh}$ and nominal battery voltage $V_{BatNom} = 3.7\,\mathrm{V}$ (see Sec. 3.2).

Table 5.1. *Measured current consumption at 3.7 V for sensors and microcontroller modes.*

Phases (cf. page 66)	Sensor / Microcontroller Mode	Current [mA]
1) Sensors	microphone	0.216
	2-axis accelerometer	0.363
2) Signal Acquisition	sampling[2]: 1 ch[1] @ f_s [Mhz]	$4.0 \cdot f_s + 0.493$
	sampling[2]: 1 ch[1] @ f_s [Mhz] and 1 ch[1] @ $f_{s_{acc}}$=10 Hz	$4.0 \cdot f_s + 0.494$
	sampling[2]: 1 ch[1] @ f_s [Mhz] and 3 ch[1] @ $f_{s_{acc}}$=10 Hz	$4.0 \cdot f_s + 0.495$
	sampling[2]: 1 ch[1] @ $f_{s_{acc}}$=10 Hz	0.490
	sampling[3]: 1 ch[1] @ $f_{s_{acc}}$=10 Hz (µC always on)	1.874
3) Feature Calculation and Classification	µC always on ($f_{\mu C} = 4096$ kHz)	1.508
4) Idle phase	µC in low-power mode ($f_{\mu C} = 32$ kHz)	0.021

[1] ch = ADC channel: high sampling frequency for microphone, low sampling frequency for accelerometer. [2] µC is in low-power mode between two samples, i.e. only 32 kHz clock active. ADC voltage reference always on. [3] µC and ADC voltage reference always on.

5.2.2. Power of Sensors and Signal Acquisition Stage

Measurement results: Table 5.1 lists the results of the current consumption measurements of the sensors (and related electronics) and the most relevant microcontroller modes. The measurements were made with the hardware from Sec. 3.2 at the 3.7 V battery terminal. Thus, in contrast to data sheet estimations, the numbers in Table 5.1 include the efficiency of the step-down converter and other losses.

For the signal acquisition stage, Table 5.1 shows that the 10 Hz sampling frequency of the accelerometers (acc) contributes little to the power consumption if compared to the much higher sampling frequency f_s of the microphone (mic). In our measurements, we observed that the power consumption is linear with f_s. At $f_s = 2\,\text{kHz}$ the current is about 0.5 mA. It is mainly dominated by internal voltage reference of the AD-converter (0.47 mA) and the low-power mode of the microcontroller (0.02 mA) – for the microcontroller was put into low-power mode between two samples. The relationship between sampling frequency and number of samples is depicted in Fig. 5.1 for different cases. Power values are calculated with equation (5.8) using $T_p = 1\,\text{sec}$, $t_{calc} = 0$ and $t_w = N/f_s$ for the microphone. The curves that include the accelerometers were calculated under the assumption that the accelerometers are always on.

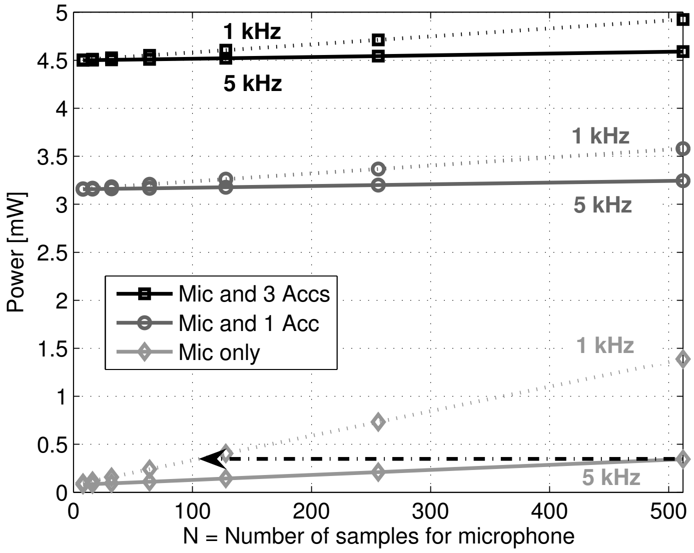


Figure 5.1. *Power consumption of sensors and microcontroller (without feature calculation). The 3 configurations (mic only, mic and 1 acc axis, mic and 3 acc axes) are depicted for 2 microphone sampling frequencies (f_s=1 kHz and f_s=5 kHz). Power values are calculated with equation (5.8) assuming that the microphone is sampled at f_s for N samples and then turned off, while the accelerometers are on for 1 second.*

Observations for signal acquisition stage: Three design rules for the signal acquisition stage can be derived from the measurements:

- For a given number of samples, choose the highest possible sampling frequency. E.g. at $N = 400$ the power consumption with 1 kHz sampling frequency is higher than with f_s=5 kHz. This rule is derived from the fact that a short on-time is favorable for a low power consumption.
- For a fixed on-time, a high sampling frequency has no effect on the power consumption. This is indicated by the arrow: e.g. for an on-time of $t_w = N/f_s = 512/5\text{kHz} = 102.4/1\text{kHz} = 102\text{ms}$ we get almost the same power consumption for sampling frequencies of 5 kHz or 1 kHz, respectively (the calculated difference is less than $10\,\mu$W). However, as will be shown in Sec. 5.2.3, a high number of input samples N will increase the processing time and therefore the overall power consumption.
- Using accelerometers results in a high power consumption even though they are only sampled with 10 Hz. This has three reasons: the high power consumption of the sensors itself, the power consumption of the voltage reference of the ADC, which is independent of the sampling frequency, and the relatively long on-period of the accelerometers[3].

5.2.3. Execution Time for Feature Calculation

Measurement results: Fig. 5.2 and Fig. 5.3 show the measured execution time of the FFT and the audio features running on the microcontroller with a clock frequency of $f_{\mu\mathrm{C}} = 4096$ kHz. Input and output resolution of the feature calculating routines are 16 bit, but internally 32 bits are used.

To compute the Fourier coefficients of N real-valued samples, a $K = \frac{N}{2}$ complex FFT is used which requires some rearranging of the output data [117]. As expected, the time to calculate the FFT is proportional to $K \cdot \log_2(K)$. The additional post-processing time in Fig. 5.2 includes the aforementioned rearrangement of the output data, and the calculation of the squares of the spectral magnitudes $|X[i]|^2$.

For the features, t_{calc} is proportional to the number of samples from which the feature is calculated (N or K depending on whether the feature

---

[3] The sensors cannot be shut down between two consecutive samples because the charge in the capacitors of the low-pass filters needs to the conserved. If the sensors are shut down, the capacitors discharge and need to be recharged when the sensors are turned on. This recharging time (approx. $t = 5\,\mathrm{RC}$) is longer than the time between two samples $t = 1/f_s$ if the filters are designed to fulfill Nyquist's theorem.

is a time domain or a frequency domain feature). The high complexity of the BER feature originates from the necessity to calculate FC first (see Appendix B.1). FC and FLUC-S have a long duration due to the square root operation in the calculation of the spectral magnitudes. Not depicted is the time to calculate mean$_{acc}$ and std$_{acc}$ from 10 input samples: there we measured $t_{calc} = 0.144\,\mathrm{ms}$.

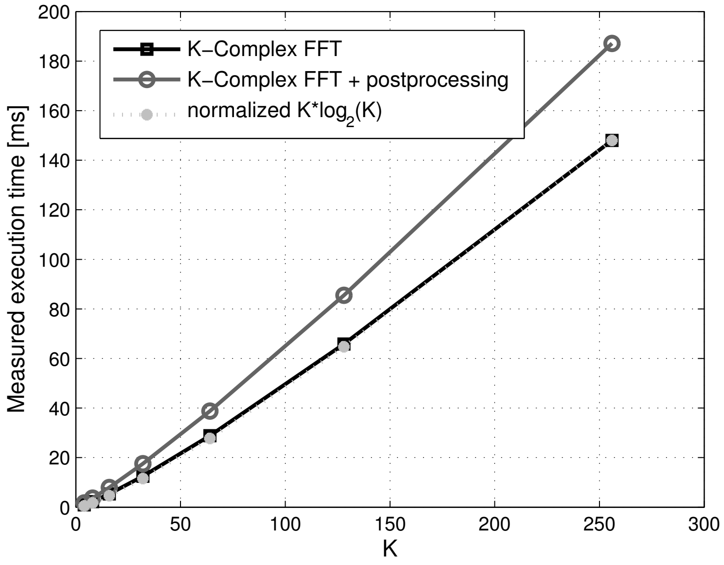


Figure 5.2. *Measured execution time for a K-point FFT. Microcontroller clock frequency is $f_{\mu\mathrm{C}} = 4096\,kHz$.*

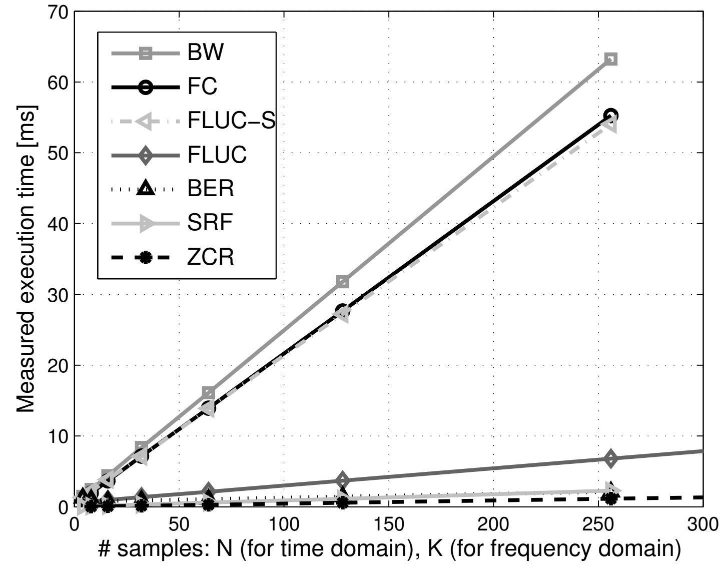


Figure 5.3. *Measured execution time for the audio features in Table 4.1 in function of number of input samples. Microcontroller clock frequency is $f_{\mu\mathrm{C}} = 4096\,kHz$.*

Fig. 5.3 ignores that certain features can be reused to calculate other ones; e.g. FC is needed to calculate BW. This re-usability of computed results is taken into account in Fig. 5.4 where the execution time of the six feature sets and the FFTcomp/LDA method is shown. For the LDA transformation matrix we assumed the existence of 5 classes or sounds, thus the matrix is of size $[\frac{N}{2} \times 4]$. Furthermore, as a simplification, we implied that the LDA transformation can be performed on the squared magnitudes of the Fourier components $|X[i]|^2$ instead of the the magnitudes $|X[i]|$, which would otherwise require a square root operation.

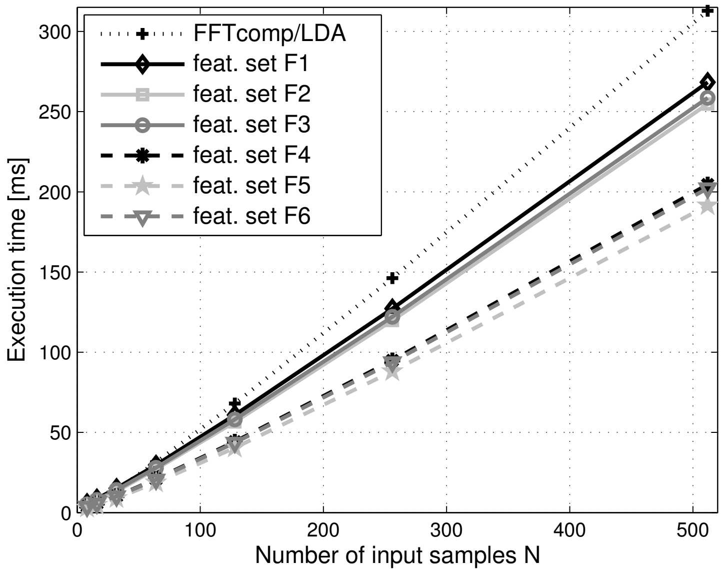


Figure 5.4. *Execution time in function of number of input samples for feature set F1 to F6 and the FFTcomp/LDA method. The LDA matrix multiplication assumes the existence of 5 classes, thus the matrix size is* $[\frac{N}{2} \times 4]$. *Microcontroller clock frequency is* $f_{\mu\mathrm{C}} = 4096\,kHz$.

Observation: Based on the previous measurements, we estimate the real-time processing capabilities of the microcontroller. More precisely, we are interested in the maximal allowed overlap of frames

$$max\ overlap = \left(1 - \frac{t_{calc}}{t_w}\right) \cdot 100\% \tag{5.9}$$

so that continuous sampling and processing of the data is possible without the need to discard the sampled data. For equation (5.9) we made two assumptions:

1. An old frame is processed while at the same time a new buffer is filled with samples.
2. The microcontroller can handle sampling and processing independently, so that t_{calc} is not affected by the sampling procedure.

An overlap value close to 100% means that for every new incoming sample the features can be computed. Zero overlap denotes that all samples are processed but that the frames do not overlap, i.e. that the frames are adjacent. An overlap of −100% (or $t_{calc} = 2t_w$) implies processing of only every second frame, i.e. that gaps the size of a frame arise in the continuous data stream. Fig. 5.5 displays the maximal allowed overlap for feature set F5 and the FFTcomp/LDA method, i.e. the two with the fastest and the slowest execution time, respectively. Due to the nature of the FFT, which requires $N = t_w \cdot f_s$ to be a power of 2, only the points indicated by the markers are valid. Comparing this result with Fig. 4.11 on page 58, we observe that generally frame overlap will not be possible for sampling frequencies which deliver a high recognition rate (i.e. $f_s > 2\,\text{kHz}$ for the FFTcomp/LDA method and $f_s > 3\,\text{kHz}$ for F5).

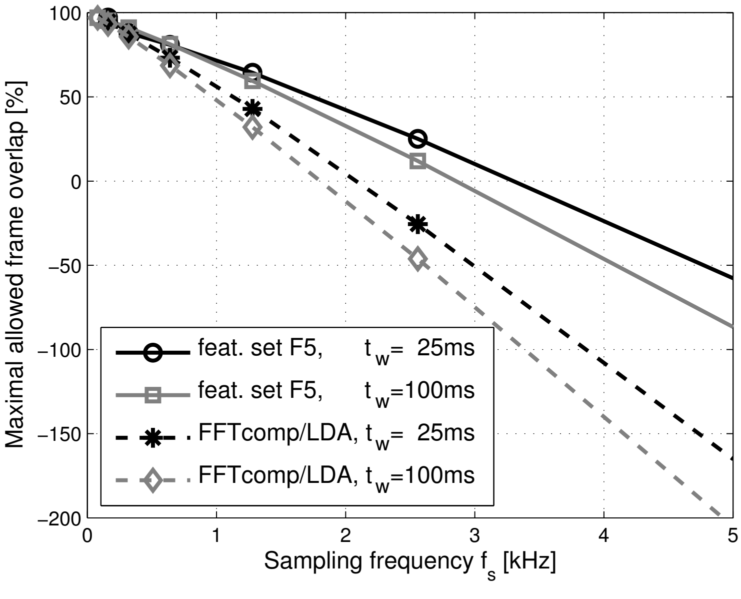
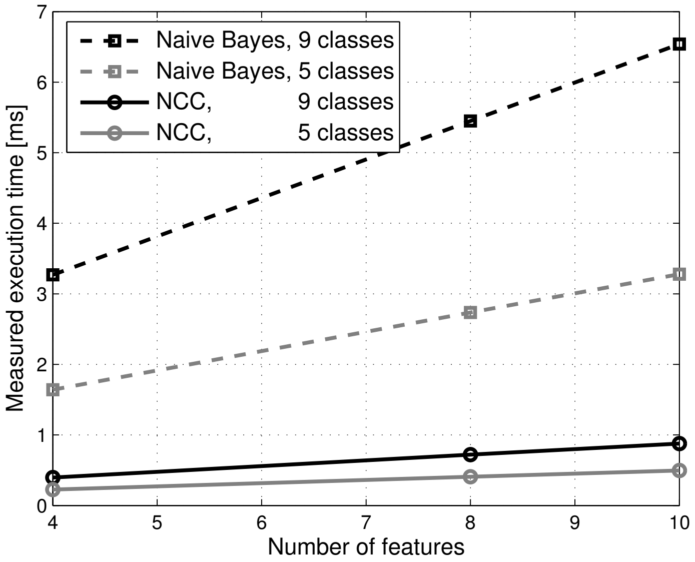


Figure 5.5. *Processing capability of the microcontroller in terms of percentage of frame overlap that allows to continuously sample the input signal at* f_s *and compute the features of a frame of duration* t_w. *Microcontroller clock frequency is* $f_{\mu\text{C}} = 4096\,\text{kHz}$.

5.2.4. Execution Time for Classifiers

Fig. 5.6 shows the measured execution time for a Naive Bayes, a NCC and a tree classifier. The time for calculating Naive Bayes and NCC depends linearly on the number of features and the number of classes (5 classes for our scenarios). Although Naive Bayes is implemented using an efficient method to calculate the product of the Gaussian distribution (by transforming them into a logarithmic space, cf. equation (B.18)), it is not as



(a) *Naive Bayes and NCC classifier*

h	t_{calc}
3	$15.12\,\mu$s
4	$18.55\,\mu$s
5	$21.54\,\mu$s

(b) *Tree classifier*

Figure 5.6. *Measured execution time for (a) Naive Bayes and nearest class center (NCC) classifier in function of number of features and number of classes and (b) decision tree classifier in function of the tree height h (max. number of leaves $= 2^h$). Microcontroller clock frequency is $f_{\mu\mathrm{C}} = 4096\,\mathrm{kHz}$.*

efficient as the NCC classifier, which on the other hand generally performs worse in terms of recognition rate.

Based on the observation on page 54 that the kNN classifier needs to store about 550 training instances to reach the same performance as a C4.5 classifier, the kNN classifier was not implemented. The execution time of a kNN classifier can be estimated based on the measurements of the NCC classifier. At their core, both classifiers calculate Euclidean distances (see Appendix B.2). Assuming that the execution time of the classifier routine is dominated by the distance calculations, the execution time is

$$t_{calc,kNN} = \frac{J \cdot L}{M \cdot L} \cdot t_{calc,NCC} = \frac{J}{M} \cdot t_{calc,NCC} \tag{5.10}$$

with M the number of classes, J number of training instances and L the dimensionality of the feature space. Thus, for $J = 550$ training instances and $L = 10$ features, the execution time of the kNN classifier is $t_{calc,kNN} \approx 54\,\mathrm{ms}$. Especially for large J (as with $J = 550$), the complexity of the sorting algorithm (cf. Appendix B.2.2) cannot be neglected. Thus:

$$t_{calc,kNN} = \frac{J \cdot L + J \cdot \log_2(J)}{M \cdot L + M} \cdot t_{calc,NCC} \tag{5.11}$$

For $J = 550$ training instances and $L = 10$ features we estimate the execution time of the kNN classifier to $t_{calc,kNN} \approx 94\,\text{ms}$.

According to the Table in Fig. 5.6, a decision tree classifier is the most power efficient of the three classifiers. For each node it requires a table lookup and a branching operation which can be calculated in less than $5\,\mu\text{s}$, with the total number of branching operations equal to the tree height h. With our data sets, the trained trees have never been higher than 10, i.e. $h \leq 10$. Thus, even large trees can be computed in less than $50\,\mu\text{s}$. This, plus the fact that the C4.5 algorithm provided high recognition results, was our motivation to continue solely with the C4.5 classifier.

5.2.5. Total Average Power Consumption

Combining the results from Sec. 5.2.2 to Sec. 5.2.4, the average total power consumption P_{avg} can be synthesized. Equation (5.8) is used as the model of our hardware, with the parameters of the model given in Table 5.1, Fig. 5.4 and Fig. 5.6.

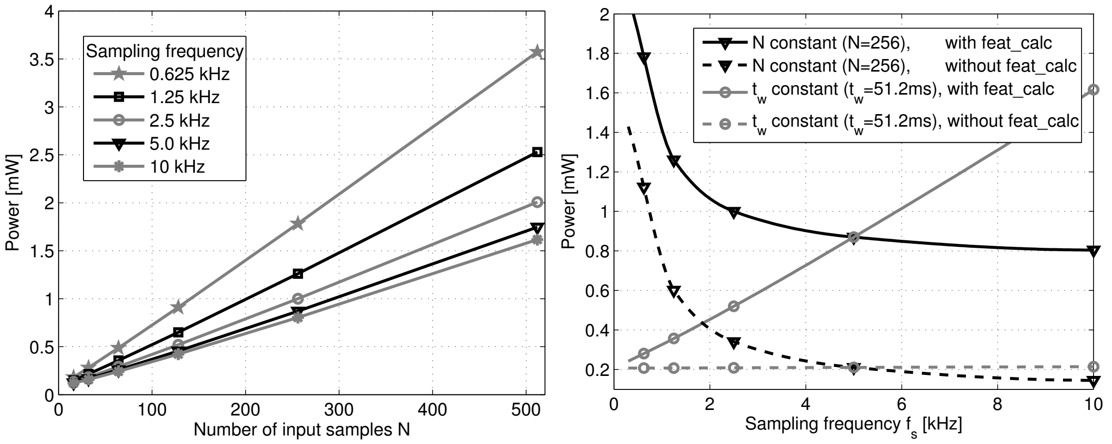
Fig. 5.7 shows one example of the total average power consumption for feature set F2 in function of f_s, N and t_w. We assume to use only the microphone, calculate the features over one sound frame of duration t_w and deliver one classification result every second. The following observations can be made:

- Fig. 5.7(a) shows that power consumption can be reduced by decreasing N due to the shorter calculation time of the features for smaller N.

- If N is given, a high sampling frequency should be considered to achieve low power consumption since this shortens the on-time of the sensors. On the other hand, if the length of the observation window t_w is given, a low sampling frequency should be chosen because this decreases t_{calc} (see Fig. 5.7(b)).

- Fig. 5.7(b) also shows that the power consumption required to calculate the FFT and features dominates the average power consumption. This is visible in the difference between the solid and the dashed lines. In case of constant N, the difference is constant since the microcontroller always requires the same time to process N samples. In case of constant t_w, the difference increases with increasing f_s since N is proportional to f_s.

This concludes our power consumption analysis. More results are depicted in the next section together with the prediction accuracy as part of our trade-off analysis.



(a) *P_{avg} versus N for various f_s.*

(b) *P_{avg} versus f_s for $N = 256$ or $t_w = 51.2\,ms$, with/without feature calculation (solid/dashed line).*

Figure 5.7. *Total average power consumption of the microphone with sampling rate f_s and the microcontroller calculating feature set F2 over a frame of duration $t_w = N/f_s$.*

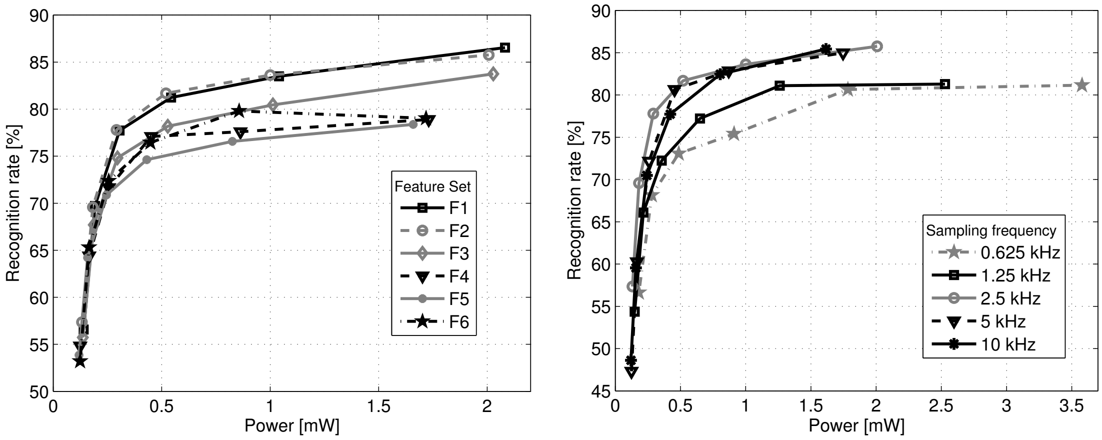
5.3. Trade-off Analysis

5.3.1. Power Consumption vs. Recognition Rate Trade-off

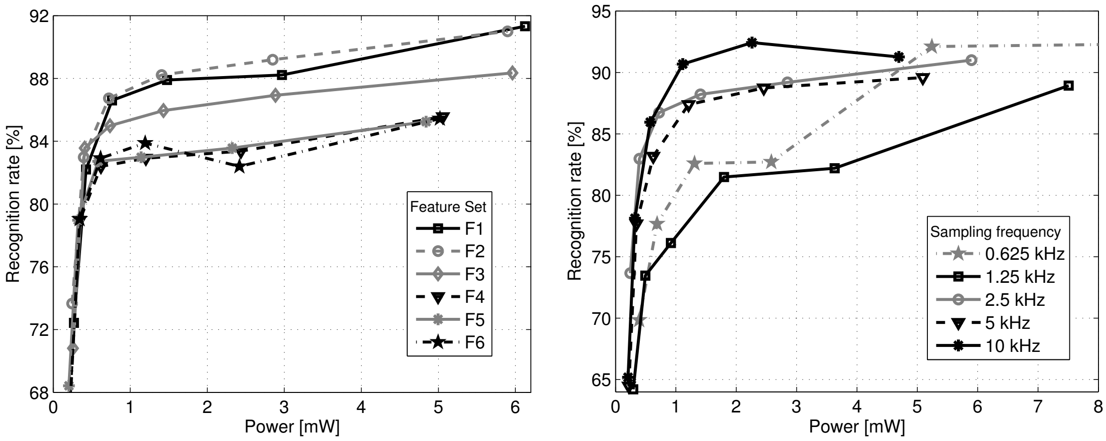
So far, the metrics power consumption and prediction accuracy have been analyzed separately. Generally, the effect that system parameters from Fig. 4.1 on page 34 have on those two metrics are conflictive. To give an example: results indicate that t_w and f_s should be small to achieve a low power consumption (cf. Fig. 5.7), but at the same time this results in a low recognition accuracy (cf. Fig. 4.7). To find an optimal operation point for a given application, we propose to incorporate power consumption concerns in the training of a context recognition system as it was illustrated in Fig. 2.3 on page 21.

Selected Results

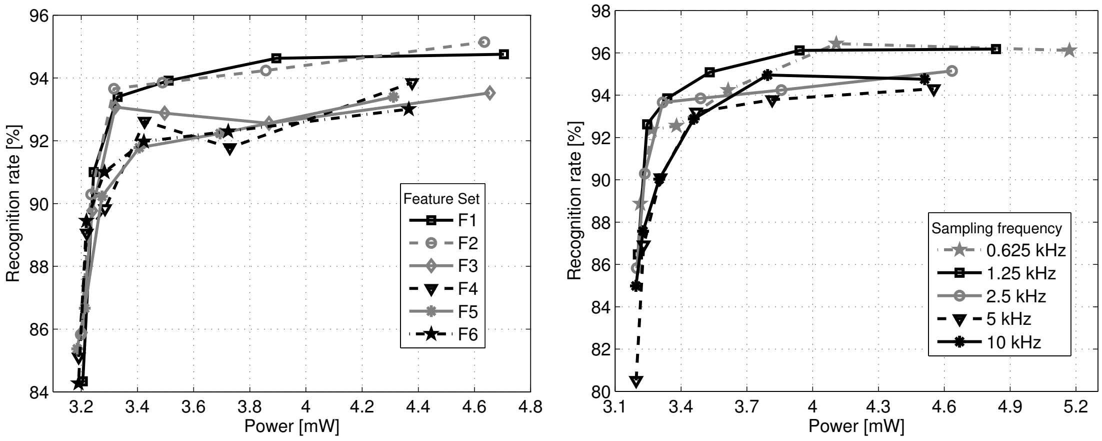
We demonstrate our approach by combining the recognition rate simulations from Sec. 4.3.3 with the measured power consumption values from Sec. 5.2. Fig. 5.8 show the resulting curves for sampling frequency $f_s = 2.5\,\text{kHz}$ (a,c,e) and feature set F2 (b,d,f). Fig. 5.8(a,b) present the 'one sound frame' strategy, (c,d) show the results where the microphone features are averaged over 3 frames and in (e,f) the microphone features are combined with the features from one accelerometer.



(a) *Mic only, with $f_s = 2.5kHz$* **(b)** *Mic only, with feature set F2*



(c) *Mic averaged, with $f_s = 2.5kHz$* **(d)** *Mic averaged, with feature set F2*



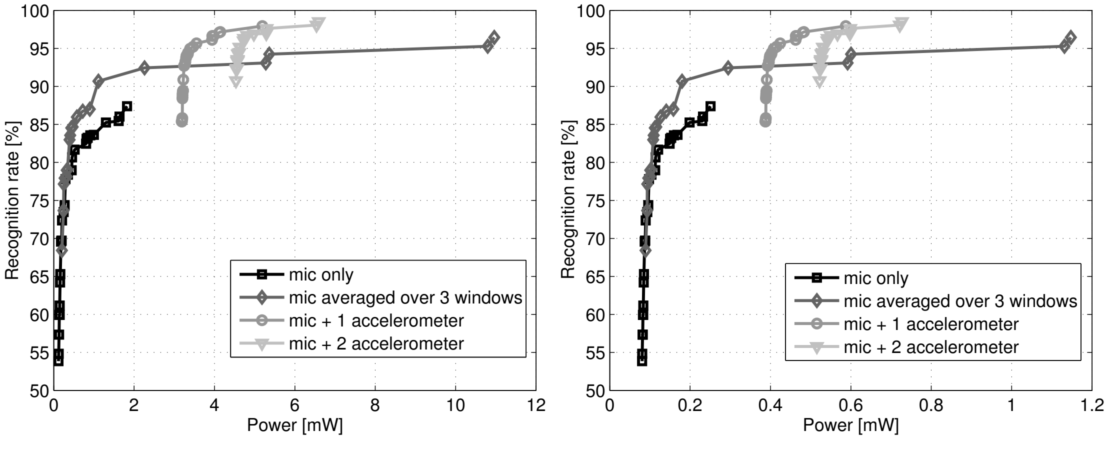
(e) *Mic + Acc, with $f_s = 2.5kHz$* **(f)** *Mic + Acc, with feature set F2*

Figure 5.8. *Selected results of the recognition rate versus power consumption trade-off for 3 strategies: audio features from 1 frame (top), audio features averaged over 3 frames (middle), combination of audio and acceleration features from one frame each (bottom).*

Each marker in the plot represents a calculated value and stands for a certain length of the observation window t_w and number of sampling points N. Optimal points with regard to the power consumption versus recognition rate trade-off are the ones towards the upper left corner. In most cases, the optimal point belongs to a high sampling frequency $f_s = 2.5 \ldots 10\,\text{kHz}$ and to feature set F1 or F2. It can be observed, that different feature sets with the same number of sampling points have little influence on the power consumption compared to the overall power consumption, but they do have an influence on the recognition rate.

Pareto Analysis

Combining the results for all parameter combinations and feature sets and selecting only the pareto points[4], leads to a curve representing the upper limit for the recognition rate and the power consumption. This is depicted in Fig. 5.9 for the three strategies (one sound frame, microphone features averaged over 3 windows, microphone features fused with features from one or two accelerometers, respectively), considering only the feature sets. In Fig. 5.9(a) the power consumption is shown for a periodicity of $T_p = 1$ second. As pointed out in Sec. 5.2.1, increasing T_p leads to a reduced power consumption as shown for $T_p = 10$ seconds in Fig. 5.9(b). The minimal power consumption is 76 μW: this is the power consumption of the hardware platform with the microcontroller running in low-power mode and all sensors turned off.



(a) *with $T_p = 1$ second* **(b)** *with $T_p = 10$ seconds*

Figure 5.9. *Pareto front for different recognition methods with periodicity T_p. Only feature sets F1 to F6 are considered here (cf. Fig. 5.10).*

---

[4] A point is called pareto point if it sees no other points in the north-west quadrant that originates from this point.

Fig. 5.9(a) shows that depending on the areas in the power vs. recognition rate plane, different strategies deliver the best trade-off. For example, starting from a power consumption of 3.25 mW it is better to use one accelerometer in addition to the microphone. Interestingly, most power consumption values that are reached by the 'one sound frame only' strategy are also covered by the feature averaging method – with the advantage of a higher recognition rate. The difference lies in the length of the sampling window: Averaging over several short windows results in a better recognition rate than using one single long window – with a similar power consumption. Using the second accelerometer on the hardware platform only increases power consumption without gaining much in recognition performance. This could be solved by changing the accelerometer orientation on the hardware, so that both the y and the z-axis belong to one accelerometer (cf. Sec. 3.2).

As we have shown, a linear transformation of the FFT components results in a higher recognition rate than the method with the feature sets (e.g. Fig. 4.8) but has to be paid with a higher power consumption (cf. Fig. 5.4). A comparison of the two methods in the power versus recognition rate plane is shown in Fig. 5.10. It is evident, that the FFTcomp/LDA method outperforms the feature set method for all three strategies. Furthermore, the FFTcomp/LDA method allows a pure sound-based activity recognition process, since it can even outperform the 'mic + acc' strategy.

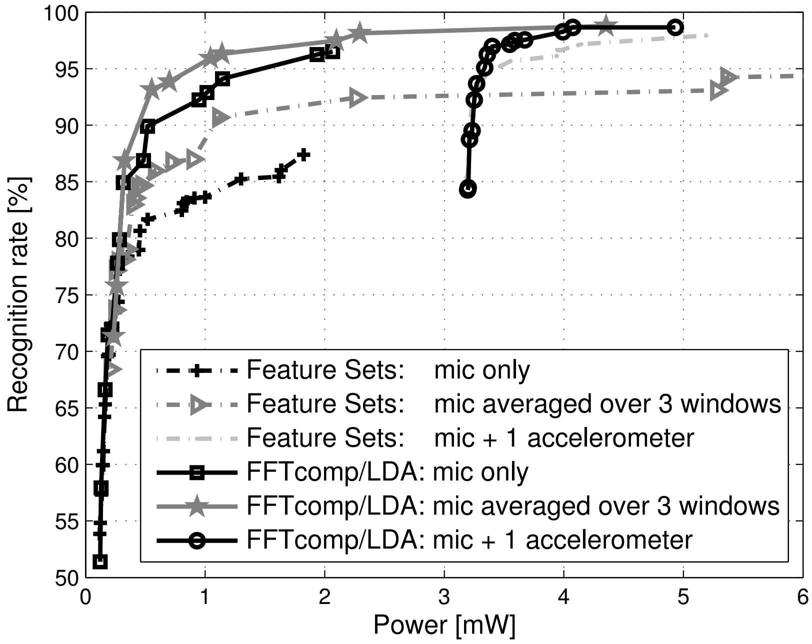


Figure 5.10. *Pareto front comparing the feature set and the FFT-comp/LDA method for the three strategies with* $T_p = 1$ *second.*

Gain of the Trade-off Analysis

In the end, the range in which we can vary parameters like sampling frequency, frame length, feature sets and recognition strategies, are limited by the wearable system and by the requirements of the application. Thus, although per definition every point on the pareto front represents the upper limit for the recognition rate for a given power consumption (and vice versa), the context recognition system may not be operated on every point of the pareto front.

Table 5.2 lists some of the parameters that can be restricted by boundary conditions. Obviously, the less stringent the conditions are, the easier it will be to find a set of parameters that match the boundary conditions. So, even though a visual inspection of Fig. 5.10 suggest that a pareto point in the north-west corner should be chosen, the requirements of the wearable system may dictate another choice. For example, a usage scenario which specifies a long battery life has an upper boundary for the power consumption. Likewise, a system that does not have enough memory to hold all the LDA transformation matrices needed for one location (cf. Sec. 2.1.3 and page 50) cannot use the FFTcomp/LDA method.

To cover a broad range of results, three choices will be presented. Besides comparing different points on the pareto front, we will also discuss the gain in recognition rate and battery lifetime achieved by choosing an optimal point in contrast to a random selected parameter set. By comparing Table 5.1 with Fig. 5.10 we notice that our frame based approach alone has reduced the power consumption by more than half in contrast to continuous recognition: if all components are kept continuously on, the system consumes 9.5 mW which equals to a battery lifetime of 59 hours (in the following section, we use the battery model from equation (5.1) in Sec. 3.2 with a battery capacity of $C = 150\,\mathrm{mAh}$ and nominal battery voltage $V_{BatNom} = 3.7\,\mathrm{V}$).

Table 5.2. *Boundary conditions for system parameters.*

Parameter	Example	Imposed by
Sampling frequency	$f_s \leq 5\,\mathrm{kHz}$	hardware
Acquisition window	$t_w \leq 200\mathrm{ms}$	application timing constraints (e.g. result ready in < 0.5 sec)
Sampling points	$N \leq 512$	hardware (memory), f_s, t_w
Recognition rate	$\geq 85\%$	application
Power consumption	‘as low as possible’	battery life, application
Periodicity	$T_p \approx 1$ sec	application, battery life
Recognition method	‘only feature sets’	hardware, system design

‘Feature set averaging over several audio frames’: With the example given in Table 5.2 one needs to choose the ‘feature averaging method’. To find the combination of feature set, sampling frequency and sampling duration that results in the optimal power consumption vs. recognition rate trade-off, Fig. 5.11 has to be considered. It shows the pareto curves of all features sets (Table 4.2) in function of the sampling frequency (e.g. the curve for $f_s = 2.5\,\mathrm{kHz}$ is the pareto curve of Fig. 5.8(c)). In this case, $f_s = 2.5\,\mathrm{kHz}$ is optimal and the point indicated by the arrow is chosen: it belongs to feature set F2 with $N = 64$ and $t_w = 25.6\,\mathrm{ms}$. Since it matches all the other boundary conditions in Table 5.2, it is a valid point. Overall recognition rate is 86.73% at a power consumption of 0.72 mW with $T_p = 1$ second. From Fig. 5.11 we also see that the potential in power savings is significant: the point on the pareto front at a power consumption of 3 mW is associated with a classification accuracy of 89.3%. This corresponds to a marginal increase in recognition rate by 3% while it shortens the battery lifetime by factor 4 (770 hours down to 185 hours). Moreover, picking a random point, i.e. one below the pareto front, may even lead to a reduction in recognition rate and battery lifetime at the same time. E.g. the point at 1 mW and 85% recognition rate shortens battery life by 216 hours and the recognition accuracy by 2%.

Table 5.3 shows the confusion matrix (CM) for the chosen (optimal) point. Especially the sounds from the coffee maker and the hot water nozzle are confused quite often. This was to be expected since they are both generated by the same device (see Fig. 3.2) which uses the same water pump for both functions. This behavior has already been reported in one of our earlier works [63].

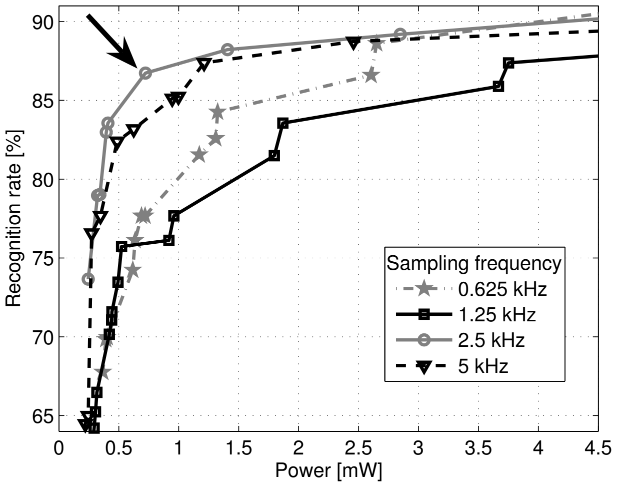


Figure 5.11. *Pareto plot (over all feature sets) for determining optimal* f_s *in case sound features are averaged over 3 frames.*

Table 5.3. *CM for sound (averaging), f_s=2.5 kHz, N=64, F2*

a	b	c	d	e	← classified as	Accuracy
284	3	18	1	3	a=microwave	91.91%
2	257	34	10	6	b=coffee maker	83.17%
32	27	233	14	3	c=hot water nozzle	75.40%
0	14	13	282	0	d=coffee grinder	91.26%
2	14	4	5	284	e=water tap	91.91%

Table 5.4. *CM for sound and acceleration, f_s=2.5 kHz, N=64, F2*

a	b	c	d	e	← classified as	Accuracy
287	7	9	6	0	a=microwave	92.88%
4	298	0	2	5	b=coffee maker	96.44%
5	0	276	26	2	c=hot water nozzle	89.32%
2	4	23	280	0	d=coffee grinder	90.61%
0	3	0	0	306	e=water tap	99.03%

Table 5.5. *CM sound (averaging), f_s=10 kHz, N=64, FFTcomp/LDA*

a	b	c	d	e	← classified as	Accuracy
302	0	3	1	3	a=microwave	97.73%
2	281	24	2	0	b=coffee maker	90.94%
7	15	269	14	4	c=hot water nozzle	87.06%
2	5	16	286	0	d=coffee grinder	92.56%
2	3	3	0	301	e=water tap	97.41%

'Combining sound feature-sets with acceleration': If a higher recognition rate for the coffee maker and the hot water nozzle is needed, the acceleration signals have to be considered as well. The two functions are activated differently: The coffee maker is switched on by pressing a button, whereas the hot water nozzle is operated by turning a knob. Table 5.4 lists the confusion matrix with the same parameters as for Table 5.3 with the difference that sound features are analyzed over only one window and acceleration features are used as well (mean_{accY} and std_{accY}). Overall recognition rate is 93.66% at a power consumption of 3.3 mW with $T_p = 1$ second, which leads to a battery lifetime of 168 hours.

As Fig. 5.8(e) and (f) illustrate, there are many sub-optimal points. By choosing one of these, one can easily be off by 0.5 to 1 mW in power consumption (about 1 or 2 days of battery lifetime) at a reduced accuracy.

'Averaging FFT components over several audio frames followed by LDA transformation': If possible, this approach should be chosen to detect the given group of activities. By averaging the fourier components over 3 frames, an average recognition performance of 93.14% can be achieved with as little as 0.55 mW, which corresponds to 42 days of 'continuously' analyzing 3 frames per second. Table 5.5 lists the confusion matrix for the selected pareto point which belongs to $f_s = 10\,\text{kHz}$ and $N = 64$. Astonishing here is that this translates to a frame duration of just 6.4 ms: A point that has never been considered as an optimal point during our separate recognition rate or power consumption analysis.

Furthermore, the pareto curve in Fig. 5.10 shows that to achieve high recognition rates, an excessive increase in power is required. For example the last two points on the pareto curve for this strategy are related to an accuracy of 98.1% and 98.8%, respectively. But this small increase of 0.7% has to be paid with 2 mW additional power consumption (from 2.29 mW to 4.35 mW, i.e. a reduction of the battery lifetime by almost 50%).

5.3.2. Classification Speed vs. Recognition Rate Trade-off

For our applications, we presumed that the sounds under investigation last for several seconds and thus classification speed is not an issue. However, in some cases, it is more important to be able to analyze as many frames

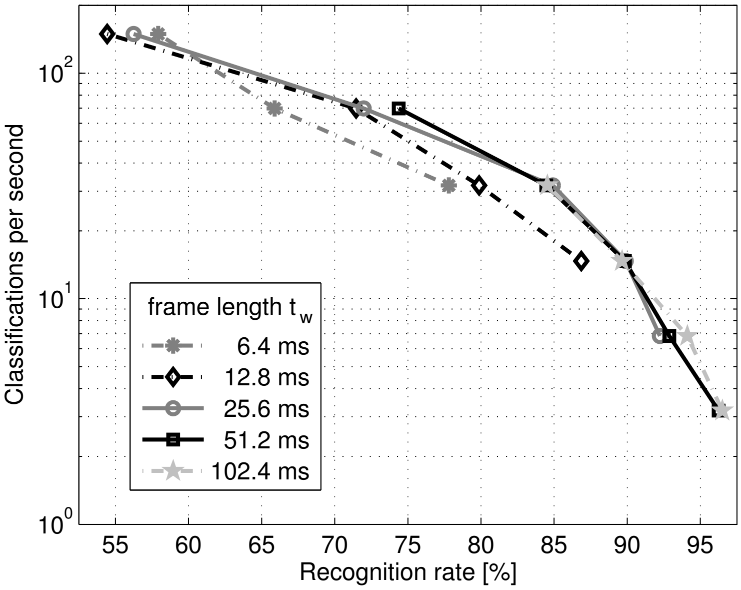


Figure 5.12. *Number of classifications per second versus recognition rate for the FFTcomp/LDA method assuming one sound frame is analyzed. Microcontroller clock is $f_{\mu\mathrm{C}} = 4096\,\text{kHz}$.*

as possible, e.g. to make sure that even sounds of short duration like loud closing of a door, are not missed.

In this case, most possibly a trade-off between recognition speed and recognition accuracy exists: For a frame of given duration t_w, increasing the number of samples will increase the recognition rate but also increase the execution time of the feature calculation. This is illustrated in Fig. 5.12 for the FFTcomp/LDA method. To achieve the highest recognition rate on a frame-wise basis, only 3 classifications per second are possible. Allowing the recognition accuracy to drop by 7% makes 15 classifications per second possible.

5.4. Summary

Since sensor nodes should run autonomously for days and months, a low power design is critical for such systems. For a predefined hardware, the options to achieve low power consumption are limited: in our case they are restricted to shutting down subsystems, duty cycling and efficient software design. We used a simplified model of our hardware to estimate the power consumption of the Sensor Node for various sampling frequencies, sampling durations, features and classifiers. The parameters of the model were derived directly from power consumption measurements of the hardware and execution time measurements of different software routines running on the microcontroller.

Next, a set of parameters which optimizes a context recognition system not only with regard to recognition rate but also in terms of power consumption has to be found. Our novel approach extends the standard design and training process used to optimize recognition performance in machine learning with a power optimization. It leads to a pareto plot which combines the two conflicting design choices ‘power consumption’ and ‘recognition rate’ and allows to pick out an optimal operation point for a given set of criteria. Our empirical design process allows to find parameter combinations which are not obvious from an analytical analysis.

In the case study with data set II, we found that optimizing the ‘parameters’ (sampling frequency f_s, sampling duration t_w, feature choice and classifiers) of the sound-based recognition process allows improvements in power consumption by a factor of 2 to 4 with only little degradation in recognition performance. Furthermore, the trade-off analysis does not stop at parameter optimization. The right selection of sensors or the number of observation windows is crucial as well and can further improve the recognition rate at an identical power consumption.

Three optimal points were selected to illustrate how well our system can classify the operation of kitchen appliances with the help of a wrist-worn microphone and accelerometers. We achieve the following recognition accuracies and power consumptions at a rate of 1 classification per second:

- 87% consuming 0.72 mW (battery lifetime 770 hours or 1 month) using sound only and a set of audio features.
- 94% consuming 0.55 mW (battery lifetime 1'009 hours or 42 days) using sound only and a linear transformation of the Fourier components.
- 94% consuming 3.3 mW (battery lifetime 168 hours or 1 week) using sound and acceleration features.

6

Discussion and Conclusion

This chapter discusses our achievements with respect to our initial assumptions. Especially, we will illustrate how the recognition algorithms deal with a large number of sounds. Furthermore, we present initial results of our algorithms running completely online on the hardware platform. Moreover, we review the relevance and the limitations of our design process and discuss the power consumption of future generations of the Sensor Node based on the International Technology Roadmap for Semiconductors. Finally, we conclude our work on sound-based activity recognition with a list of achievements and propositions for further research.

6.1. Discussion

In Chapter 2 we have introduced constraints concerning the sensor architecture, the hardware, the microphone placement, and the number and type of sounds used. While we argue that some of the constraints are well justified (e.g. wrist-worn microphones for user activity detection or the use of a general purpose hardware platform), others might leave open questions. So, for example, how well would our algorithms perform if more than just 5 sounds have to be distinguished? This will be discussed in Sec. 6.1.1. Furthermore, in Sec. 6.1.2 we will review the assumption that it is more power efficient to perform the classification locally in contrast to transmitting the raw sensor data to a wearable computer.

Moreover, we have assumed that our approach of recording the experiments with high quality equipment and lowering the quality of the data as part of the design process is comparable to performing the recognition procedure online on the hardware. Sec. 6.1.3 shows the recognition performance of the hardware platform if the on-board microphone and fixed-point computations are used. Last but not least, we review the relevance and the limitations of our design process in Sec. 6.1.4 and discuss the power consumption of future generations of the Sensor Node based on the International Technology Roadmap for Semiconductors in Sec. 6.1.5.

6.1.1. Classifying a Large Number of Sounds

Based on the arguments in Sec. 2.1.3, we have optimized our algorithms for a small number of sounds. The following paragraphs show how our recognition process scales with the number of sounds.

Increasing the number of sounds will lead to a decrease in recognition rate, since a limited number of features allows to discriminate only a limited number of sounds. This can easily be explained with the probability density function of the features: Adding more classes does not change their probability density functions but generally places them closer together in the feature space so that they are harder to separate. To avoid this, more features can be used – and thus creating more ‘space’ in the feature space.

To evaluate how much the recognition rate will drop, all 19 sounds from data set I (i.e. all the sounds from Table 3.1) were mixed together and thus the classification task was to distinguish between 19 different sounds. The result is shown in Fig. 6.1. The performance evaluation was done in the same way as for Fig. 4.6 on page 49. The 13 quadruples of bars correspond to the 13 different combinations in Table 4.3. Especially interesting is combination 1 and 6, which corresponds to the FFTcomp/LDA approach and Feature Set F1, respectively.

We observe that the recognition rates, which in case of 5 sounds per group were in the range of 80% to almost 100%, now drop to 70% to 90%. Furthermore, we notice that the FFTcomp/LDA approach now definitely performs better and achieves 85% recognition rate. This, however, is due to the different dimensionality: While feature set F1 spans a 10 dimensional feature space for any sound group, the LDA transformation in the FFTcomp/LDA approach reduces the feature space to $M-1$ dimensions, where M is the number of sounds. If, for a fair comparison with Fig. 4.6, only the 4 most dominant dimensions are considered, then the recognition rate drops to 67% (C4.5) or 72% (kNN) in case 19 sounds are used. Similarly, we observe that combination 9 and 11 perform rather well, which again can be accredited to a larger feature set.

To sum up, increasing the number of classes from 5 to 19 results in a reduced accuracy by 10 to 20%. To compensate, one needs to include more features in the feature sets or use the FFTcomp/LDA approach since it scales with the number of classes. Obviously, both variants will lead to an increase in power consumption. Based on our observations, we expect the recognition process to work with more sounds as well, but with a gradual decrease in recognition rate if more sounds are included.

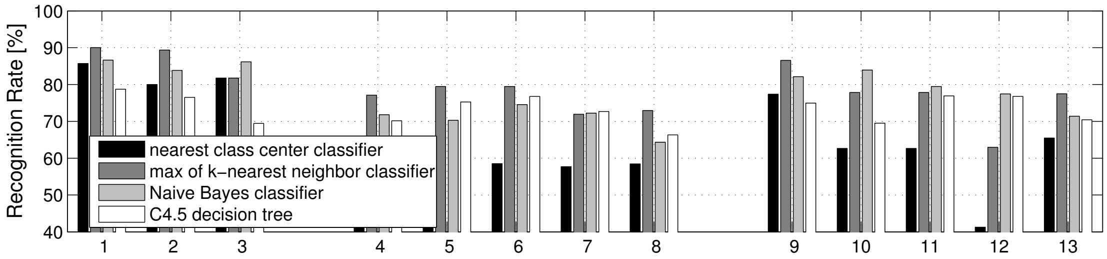


Figure 6.1. *Recognition rates for classifying all 19 sounds from data set I, with* $f_s = 4.8\,kHz$*,* $N = 256$*,* $t_w = 53.3\,ms$*.*

6.1.2. Computation vs. Communication Trade-off

Throughout our work we have assumed that the hardware platform is an autonomous node which performs all computation itself. In a non stand-alone system, the communication strategy will have an impact on the power consumption (cf. Fig. 2.1). Generally, it is presumed that wireless communication is more power consuming than computation [52, 64]. To support this statement, one usually compares the energy to transmit one bit across the wireless link with the energy to execute an instruction on the local processor [48]:

$$E_{TX,bit} = \frac{P_{TX}}{DataRate} \qquad E_{\mu \mathrm{C},instr} = \frac{P_{\mu \mathrm{C}} \cdot CPI}{f_{\mu \mathrm{C}}} \tag{6.1}$$

where CPI is the number of clock cycles per instruction for the processor. If an algorithm requires m instructions to reduce the data by n bits, then it is energy-wise more efficient to do local processing before transmitting data if

$$m \cdot E_{\mu \mathrm{C},instr} < n \cdot E_{TX,bit} \tag{6.2}$$

For the MSP430F1611

$$E_{\mu \mathrm{C},instr} = \frac{2\,\mathrm{mA} \times 3\,\mathrm{V}}{4\,\mathrm{MHz}/(1\,cycle/instr)} = 1.5\,\mathrm{nJ/instr} \tag{6.3}$$

The nRF2401 transceiver requires

$$E_{TX,bit} = \frac{8.8\,\mathrm{mA} \times 3\,\mathrm{V}}{1000\,\mathrm{kbps}} = 26.4\,\mathrm{nJ/bit} \tag{6.4}$$

according to the data sheet. This value is optimistic because several effects are neglected: First of all, the MSP430 cannot deliver data rates of 1 Mbps. However, the nRF2401 transceiver has a ‘ShockBurst’ modus which allows the microcontroller to fill a register of the transmitter at much lower speed. During that time just the MSP430 and the register of the nRF2400 draw current. Once the register is filled, the RF frontend is activated to transmit the data with 1 Mbps over the wireless link and hence for a short time 8.8 mA is drawn. Depending on the speed of the microcontroller and the output power at the RF frontend, we measured $E_{TX,bit}$ in the range of

$$E_{TX,bit} = 115 \ldots 290\,\mathrm{nJ/bit} \qquad \text{(measured)} \tag{6.5}$$

This complies with Kohvakka et al. [72] who measured a much higher power consumption of the nRF2401 compared to data sheet calculations[1]. Based on those numbers, we determine the computation – communication trade-off in two examples:

Example 1: *Online Classification vs. offline Classification:.* We assume that in both cases the features are calculated locally. Execution time for a tree based classifier was roughly $t_{calc} = 25\,\mu$s (cf. Sec. 5.2.4) at $f_{\mu \mathrm{C}} = 4096\,$kHz. Thus, $m = 102$ instructions; the classification process consumes 154 nJ.

The classification stage reduces the number of bits by $n = 4 \cdot 16 - 8 = 56$ bits, assuming a 4-dimensional feature space (FFTcomp/LDA), 16 bit features and 8 bit classification result. It would require at least 56 bits $\cdot 115\,\mathrm{nJ/bit} = 6.4\,\mu\mathrm{J}$ to transmit these bits. Therefore, local classification is advisable.

---

[1] For comparison: $E_{TX,bit}$ of a Bluetooth radio ranges between 100 nJ/bit [58] to 270 nJ/bit [53].

Example 2: *Raw data transmission vs. local feature calculation and classification:*. Here, we consider the best and the worst case (cf. Fig. 5.4): $t_{calc,F5} = 192\,\mathrm{ms}$ and $t_{calc,FFTcomp/LDA} = 313\,\mathrm{ms}$ for $N = 512$ input samples and $f_{\mu\mathrm{C}} = 4096\,\mathrm{kHz}$. Therefore, local feature calculation and classification consumes between 1.2 mJ and 1.9 mJ.

The reduction in numbers of bits is $n = 512 \cdot 16 - 8 = 8184$ bits. Transmitting 8184 bits requires between 0.9 mJ and 2.4 mJ. In this case, it is not obvious whether local computing or wireless communication should be favored.

As the second example illustrates, a first order approximation of the energy consumption is not sufficient to determine the computation – communication trade-off. More aspects need to be considered. A complete investigation is beyond the scope of this work; here we just list some aspects which in the end let us favor local processing. In terms of power, the first order approximation has the following drawbacks:

- protocol overhead and networking issues like packet loss or collisions is ignored (e.g. longer transmit duration might lead to more collisions)
- power consumption of the receiver is not included, neither are transmitter power up/down transients.

Even if in terms of energy efficiency, the computation vs. communication trade-off is decided in favor of transmitting raw data for processing on the wearable computer (cf. Fig. 2.2), other factors need to be looked at:

- availability of channel bandwidth to send raw data
- availability of the central wearable computer (to receive and process the data).
- impact on the power consumption of the central wearable computer if it has to process the data from one or several sensor nodes.

6.1.3. Online Recognition Performance

For our empirical design process, we have assumed that comparable recognition rates can be achieved by recording the experiments with high quality equipment and lowering the quality of the data as part of the design process and by performing the recognition procedure directly on the hardware. Thus, so far we have not dealt with limited microphone sensitivity, reduced accuracy of fixed-point calculations and system noise usually present in a hardware platform. In the following section, we evaluate the overall influence of the hardware implementation on the recognition accuracy.

Frame Based Recognition: A series of experiments were recorded with our hardware[2] in the wood workshop and the kitchen. Similar to the recording procedure for data collection I, described in Sec. 3.1.2, we recorded sounds with several hand positions. Overall, about 12'000 frames were recorded with the hardware and the features calculated in real-time on the hardware's microcontroller.

Table 6.1 lists the frame-wise recognition rates using feature set F5 and a C4.5 decision tree classifier for the kitchen and the workshop sounds. As a comparison, the recognition rates for data set I and II are listed. The difference of about 30% between set I/HW and set II for the coffee maker and the hot water nozzle might be contributed to a different coffee machine used for the two experiments. Thus, a direct comparison is only possible between set I and HW. As for the comparison between data set I and the SoundButton, we observe that the recognition rates are generally 5 to 15% lower for the hardware.

Table 6.1. *Frame based recognition rates for data set I and II and the hardware prototype (HW). Results are calculated with feature set F5, a decision tree classifier,* $f_s = 4.8\,kHz$ *(*$5\,kHz$ *for set II) and* $N = 256$*.*

(a) *Kitchen Sounds*

Sound	Set I	Set II	HW
microwave	94 %	90 %	93 %
coffee maker	97 %	65 %	95 %
hot water nozzle	91 %	64 %	83 %
coffee grinder	78 %	77 %	70 %
water tap	89 %	92 %	81 %
Average	90 %	78 %	84 %

(b) *Workshop Sounds*

Sound	Set I	HW
sawing	76 %	66 %
drilling	94 %	82 %
hammering	78 %	86 %
grinding	83 %	41 %
filing	62 %	56 %
Average	79 %	66 %

There are some results which are not fully understood so far. For example, the bad recognition rate of the grinding machine (41% vs. 83%) in the workshop sounds or the good recognition rate of hammering (86% vs. 78%). We believe, that these results are mainly due to the recorded sound levels: some of the sounds recorded with the prototype showed very small amplitudes (3 bits and less) which makes sound discrimination almost impossible. Or in other words: although the features are in principle immune to scaling of the input vector, the fixed-point implementation of the algorithms showed a tendency to classify sounds in 'loud' and 'quiet'

---

[2] We used a first prototype called 'SoundButton' from [29, 80] for these experiments. The performance is comparable to the one of the hardware presented in Sec. 3.2 because the differences in layout and hardware components between the SoundButton and the Sensor Node are marginal.

sounds. Therefore, we suggest to include an automatic gain control for the microphone in future hardware revisions.

Event Based Continuous Recognition: To illustrate the performance of the system in a real life scenario, a subject wearing the SoundButton was asked to randomly pick 20 activities from the list of 'kitchen' tasks and perform them at random times within a 10 minute period. Feature set F5 was calculated online on the hardware at a rate of 5.6 frames per second and transmitted together with the raw microphone data to a PC. Then we processed the data as follows: Firstly, we identified interesting segments using the ratio of rms from the microphone signals of two devices – one mounted on the user's wrist, the other on the user's chest (cf. Sec. 4.2.4). Then, we applied a frame by frame recognition to these segments and classified the features, which had been calculated on the microcontroller, with a tree based classifier. Finally, we performed segment-wise classification using a majority decision over all frames in a segment.

The results of the first 4 minutes are shown in Fig. 6.2. It depicts the hand labeled ground truth, the frame based recognition and the activities that are detected as the result of the majority decision over all frames in a segment (segmented with the help of the rms ratio between the two microphones). For the whole 10 minutes, we counted 18 correctly labeled events, 2 substitutions (a wrongly classified event), 2 insertions (detection of an event although none was there) and 0 deletions.

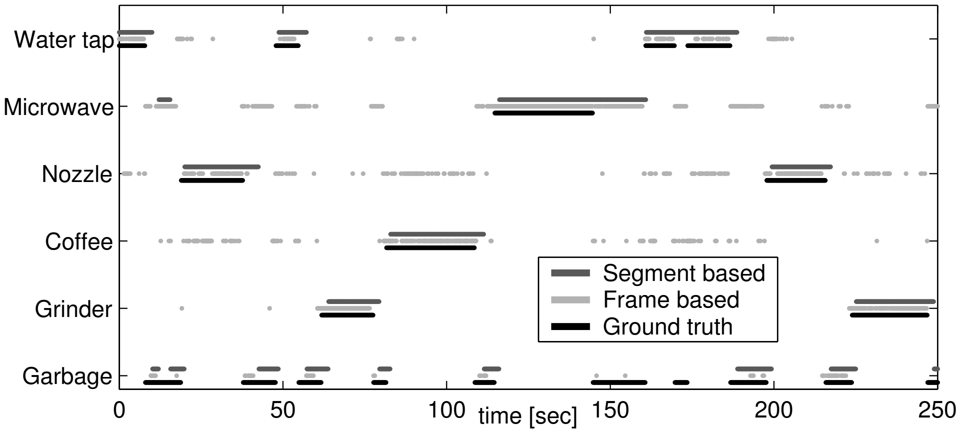


Figure 6.2. *Continuous recognition with the hardware platform using a wrist and chest microphone for segmenting the data stream, feature set F5 and a C4.5 classifier for frame based classification, and a majority decision over all frames of one segment to achieve event based recognition.*

6.1.4. Relevance and Limitations of Design Method

The problem addressed by our design methodology – reducing the power consumption of a context recognition system – is widely recognized to be one of the key challenges of the field. In principle, the general concept of an iterative, empirical optimization method that incorporates power consumption issues into the training process of the activity recognition system is applicable to any context recognition system.

As argued in Sec. 1.2.2, the recognition of interactions with appliances through sound analysis is relevant for a wide range of applications. Furthermore, the sensors used in our case study are prototypical for a generic combination of ‘complex’ and ‘simple’ sensors often found in wearable systems [53, 54]: One sensor, in our case the microphone, captures signals which contain a lot of information and can be used to spot complex activities such as “getting a coffee” but require high sampling rates and computing power. On the other hand, the second sensor delivers less information and therefore may allow ambiguous interpretations of the reading but is easier to process. Other sensor combinations are for example video and accelerometers [118] or accelerometers and a light sensor [63].

In summary, it can be said that the case studies used to illustrate our method are certainly not toy examples and the lessons learned are applicable to a range of other scenarios and hardware platforms. This is backed up by other studies which have shown that design methodologies can improve battery lifetime with almost no degradation in recognition performance [60, 62]. Nonetheless, we acknowledge that there are clearly limits to what can be generalized from a single study and further system examples need to be considered.

In technical terms, the main limitation of the methodology presented in this work is the lack of an automated design method which automatically selects the parameters for an optimal recognition rate vs. power consumption trade-off during the training phase of the system. Therefore, the following items need to be incorporated:

- automatic power estimations tools which include the power consumption of the algorithms, the sensors and AD-converter
- methods which evaluate the performance of features and their power consumption
- design tools which determine the optimal parameters automatically, e.g. with genetic algorithms [119]

6.1.5. Glimpse into the Future

Although in the computer area some technology forecast proved to be wrong[3], we dare to glimpse into the future and forecast sound-based activity recognition systems of the future. Especially, we are interested in the validity of the presented methods and results, 10 years from now, in 2015.

ITRS Roadmap: The International Technology Roadmap for Semiconductors (ITRS) [121] identifies the technological challenges and needs which the semiconductor industry will be facing over the next 15 years. The problem with extrapolating the power consumption of our hardware platform are manyfold:

- in terms of logic circuits, the ITRS focuses on microprocessors (meaning processors with several GHz processor frequency) made in a single process technology. Manufactures of microcontrollers, on the other hand, use proprietary processes specifically developed for low power operation, often combining several different processes that are specialized for different sections of the µC [122].
- manufacturable solutions for achieving the semiconductor speed and density postulated by the ITRS are not known for years beyond 2009.
- power consumption of sensors (e.g. MEMS microphone) cannot be estimated based on ITRS.

Extrapolating Power Values: We will investigate 4 systems which are described in detail in Table 6.2.

1. Sensor Node from Sec. 3.2 implemented in 2005
2. Sensor Node as in 1. but assuming a lower operating voltage
3. Sensor Node in 2015 with the same functionality (i.e. number of transistors) and same standby power consumption as in 2005 but otherwise scaled with process technology
4. Sensor Node in 2015 with increased functionality compared to 2005; leakage and dynamic power consumption scaled with process technology

---

[3] Remember the famous quote: “640 kB ought be enough for everybody” by Bill Gates or “I think there is a world market for about 5 computers” by IBM chairman Thomas Watson back in 1943 – more examples of technology forecasts that did never fulfill in [120].

We made the following assumptions to estimate the power consumption of the systems:

- power consumption is based on calculation of the 'Averaging FFT components over several audio frames followed by LDA transformation' strategy for the parameters in Table 5.5 in page 81 ($f_s = 10\,\text{kHz}$, $N = 64$).
- only microphones are used. Accelerometers are omitted.
- the average total power consumption is calculated with equation (5.8) with $T_p = 1\,\text{seconds}$.
- the power consumption of the MSP430 is used as a base (cf. Table 5.1). The power consumption for future processors is calculated with the help of scaling factors derived from the 'ITRS Process Integration, Devices, and Structures (PIDS) 2005' roadmap for the LSTP (low standby power logic) category.
- processor power consumption scales according to equation (5.2) and depends only on the logic circuits, i.e. memory power consumption is assumed to scale linearly with logic circuits.
- current consumption of the microphone, the ADC reference voltage generator and the ADC[4] scale only by a factor 0.5 in 10 year.
- The efficiency η of the step-down converter was modeled with:

  $$\eta = \alpha - \beta \cdot (i_L)^{-\gamma} \qquad \text{in \% with } i_L : \text{load current in mA}$$

  with $\alpha = 88$, $\beta = 4.2$, $\gamma = 0.78$ fitted to the efficiency curve of the Texas Instruments TPS62220 step-down converter used on the Sensor Node. For the 2015 Sensor Node we choose: $\alpha = 95$, $\beta = 5$, $\gamma = 0.78$. The minimum efficiency is always set to 70%.

Table 6.2 displays the key parameters of the four systems and indicates how the parameters were derived from ITRS. Fig. 6.3 shows the calculated power consumption (including step-down converter efficiency) and battery lifetime (assuming the model from equation (5.1) and 24 h operation) of the four systems. The power consumption is split into the 4 phases described in connection with equation (5.8) and Table 5.1: Microphone, Signal Acquisition, Feature Calculation (in this case FFTcomp/LDA) and Idle/Standby phase.

---

[4] According to the 'System Drivers' chapter [121], the figure of merit for an ADC – measured in GHz/W – improves by a factor of 2 every three years. However, this applies mostly to sampling frequencies of several MHz to GHz. As shown in Sec. 5.2.2, in our case the current consumption of the ADC reference voltage generator dominates the power consumption of the ADC.

Table 6.2. *Key parameters for comparison of Sensor Node System 1 and 2 (2005) with 3 and 4 (2015).*

Hardware part	Parameter	System 1[a]	System 2[b]	System 3[c]	System 4[c]
System	Supply voltage [V]	2.8	1.8	1.5	1.5
Microphone	current consumption [mA]	0.211	0.211	$0.211 \cdot \frac{1}{2}$	$0.211 \cdot \frac{1}{2}$
ADC	current consumption[d] [mA]	0.57	0.57	$0.57 \cdot \frac{1}{2}$	$0.57 \cdot \frac{1}{2} + 0.13$
Processor	scaling : V_{dd}	-	1	0.833	0.833
	scaling[e]: C_{gate}	-	1	0.15	0.15
	scaling[f]: static power	-	1	1[g]	0.21
	scaling[h]: number of transistors	-	1	1	30
	active current [mA]	1.7	1.02	0.12	3.78
	standby current [μA]	20	20	20	128
	feature calculation time[i] [ms]	77	77	77	7.7
Battery	capacity[k] [mW/g]	150	150	400	400

[a] Sensor Node from Sec. 3.2 in 2005. Values differ from Table 5.1 since here they are measured at output of step-down converter.

[b] never built since accelerometers require 2.4 V supply voltage, but possible without accelerometers and using today's technologies.

[c] scaling factors for logic circuits (processor) in 2015 are based on ITRS [121] with System 2 as a reference (see footnote e to h).

[d] sampling frequency $f_s = 10\,\text{kHz}$. Higher current for System 4 due to higher leakage current.

[e] C_{gate} is total gate capacitance per transistor, calculated with $C_{gate} = C_{g,total} \cdot 3 \cdot L_g$, see PIDS Table 41a and 41b [121].

[f] per transistor, calculated with $P_{stat,dev} = V_{dd}(I_{sd,leak} + J_{g,limit} \cdot L_g) \cdot 3 \cdot L_g$, see PIDS Table 41a and 41b [121].

[g] for System 3 same leakage current as in 2005 is assumed.

[h] number of transistors scaled with Moore's law: density increase by 41% per year [123]:
System 4 has same chip size as in 2005, System 3 has $\frac{1}{30}$ chip size of 2005.

[i] $f_{\mu\text{C}}$ constant at 4 MHz, $N = 64$ samples. System 4: t_{calc} is decreased by factor 10 due to increased number of transistors.

[k] 2005: battery from Sensor Node; 2015: from ITRS; weight kept constant at 3.7 g

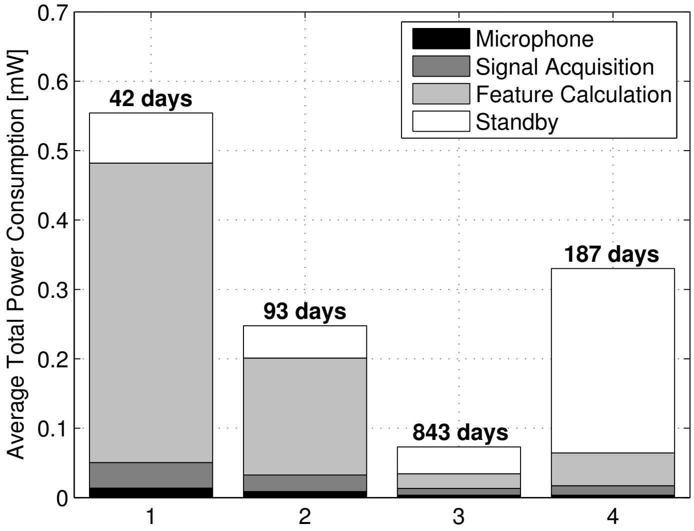


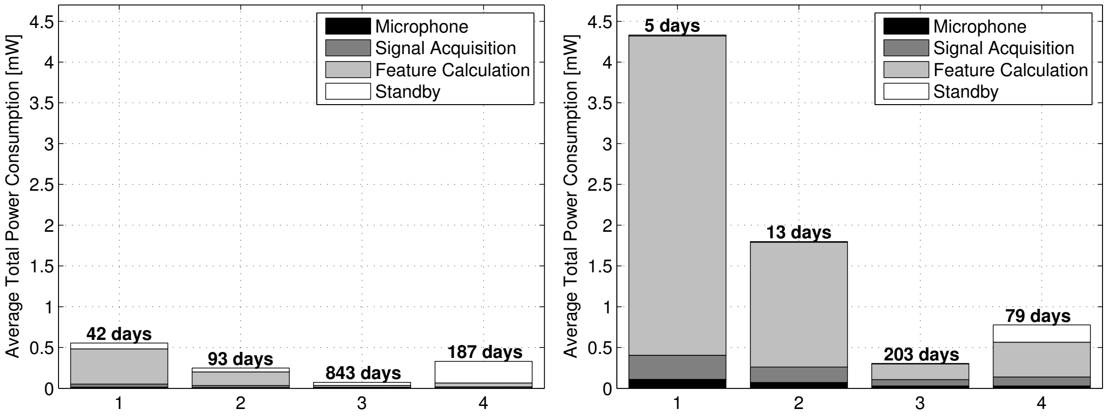
Figure 6.3. *Extrapolated power consumption and battery lifetime for the four systems in Table 6.2*

We make the following observations:

- Even today a reduction of the power consumption by a factor 2 is possible by lowering the supply voltage to 1.8 V.
- In 10 years, a sub-100 μW sound-based user activity recognition system is feasible (System 3). However, this requires that technology scaling is used to make the chips smaller and to consume less power in overall (and not just per transistor). System 3 could provide one classification per second, 24 hours a day, for almost $2\frac{1}{2}$ years.
- System 4 reflects the trend in processor manufacturing: at more or less constant chip size, maximum speed and number of transistors are increased but at the same time total power consumption as well (as seen with current trends in processors for PDAs and PCs – compare [124]). System 4 offers more processing capability than needed for the algorithms used in this work. High leakage currents cause the total power consumption to be higher than for the optimized system of 2005 (System 2). Due to an increased battery capacity, the lifetime of System 4 is still twice as long as of System 2.

Having a low power system at hand does not render the design methodology presented in this work obsolete. This is illustrated in Fig. 6.4. On the left, the same graph as in Fig. 6.3 is shown but with a different ordinate scaling. Fig. 6.4(b) depicts the power consumption and battery lifetime for the four systems of Table 6.2 on the assumption that $N = 512$ samples are processed with $f_s = 10\,\text{kHz}$ (the rightmost point on the FFTcomp/LDA pareto plot – marked with grey stars – in Fig. 5.10). This result in a longer on-time of the ADC and a longer feature calculation time t_{calc} compared to the configuration with $N = 64$ samples and $f_s = 10\,\text{kHz}$.

We observe that with the help of our design methodology, the battery lifetime of System 1 and 2 is decreased by a factor of 7.8 and 7.3, respectively. For the future Systems 3 and 4 the decrease in battery lifetime is still a factor of 4.2 and 2.3, respectively. Due to higher leakage currents for System 4 (which are constantly present), the battery lifetime for system 4 decreases not as much as the lifetime for System 3. Nevertheless, based on Fig. 6.4 we conclude that our design method is both valid and useful in the future.



(a) *with* $N = 64$ **(b)** *with* $N = 512$

Figure 6.4. *Impact of number of sampling points N on power consumption and battery lifetime for the four systems in Table 6.2.*

6.2. Conclusions

The objectives of this work were twofold:

1. to show that sound is a useful senor modality to detect user activity
2. to show that sound-based user activity recognition is possible on hardware platforms which offer only limited computational power

We believe, that this work is a different approach to the field of wearable activity detection which is usually done with motion sensors. The achieve our goal, we made the following contributions:

- We analyzed scenarios in which sound-based activity detection is useful. Furthermore, we discussed under which constraints low-power sound-based activity recognition is feasible.

- We modeled the recognition chain from sensors to classifiers to show that with little computational complexity reasonable recognition rates can be achieved. Such a detailed analysis is rare in related works. Most works just concentrate on one topic (e.g. features).
- We introduced a novel way to segment the data stream by comparing the signal amplitude of a wrist-worn and a chest-worn microphone.
- We recorded sounds for 4 different scenarios and applied our recognition algorithms to them to prove that our method works for a wide range of scenarios.
- As a major contribution, a design method was introduced to determine the power consumption versus recognition rate trade-off during system training. To our knowledge, this is the first report that illustrates how energy limited context recognition systems can benefit from a detailed analysis of not only the recognition procedure, but also the power consumption.
- To prove our design method, we recorded data in a second experimental setup which involved 11 users. Furthermore, we presented power consumption measurements of an existing sensor hardware and implemented features and classifiers on a microcontroller to perform sound classification.

We believe that our work will open new ways for user activity recognition and context recognition systems:

- First of all, we have shown that sound-based user activity is possible. But whether it is useful for daily applications needs to be proven in field studies where emphasis is put on a natural environment. While loud background noises were eliminated during our experiments, in an realistic setting we may encounter interfering noise, e.g. from a radio, a TV or from people talking.
- Furthermore, emphasis needs to be put on low-power sound-segmentation methods. The performance of these is dependent on the ratio of occurrence between ‘sound-based’ activities and other sounds/noises. Thus, the segmentation methods need to be tested in real world scenarios where other sounds can occur or the user is allowed to perform other activities than the ones the context recognition system should recognize.
- Our work could open up new ways to train context recognition system: by including its power consumption. Future research could deal with including power consumption into the feature subset selection or with the design of automatic tools to speed up the search process.

A

Dealing with Class Skew in Context Recognition [*]

As research in context recognition moves towards more maturity and real life applications, appropriate and reliable performance metrics gain importance. This chapter focuses on the issue of performance evaluation in the face of class skew (varying, unequal occurrence of individual classes), which is common for many context recognition problems. We propose to use ROC curves and Area Under the Curve (AUC) instead of the more commonly used accuracy to better account for class skew. We present a theoretical analysis of their advantages for context recognition and illustrate their performance on a real life case study.

---

[*] This chapter is mainly based on Ref. [125]

A.1. Introduction

As context-aware systems become more mature and aim to move from laboratory examples towards real life applications, the need for appropriate performance measures becomes stringent. For one, such measures are required for an objective comparison between different research results. Second, they are needed to optimize system architectures with respect to specific requirements and trade-offs demanded by different applications.

Class Skew in Context Recognition: Very often context recognition deals with varying, unequal occurrences of individual classes. This is called *class skew*. It can be caused either by naturally imbalanced classes or by varying user preferences. Performance metrics should consider this effect in order to avoid misleading interpretations of the result. In health care, for example, one often tries to detect *rare* events like collapses, heart attacks, seizures or other dangerous incidences. In other context recognition domains, an engineer might train a system to recognize certain events without knowing the user's preferences. Place and scene recognition based on environmental clues is such an example: the time spent at an individual location might differ from user to user and not match the percentages used during the training of the system. Monitoring daily activity also has to deal with class skew since no two activities take the same amount of time: e.g. sleeping takes much longer than eating, but both are probably equally important. Thus, in many context recognition applications it is not desirable to tune classifiers to recognize often occurring events better than rare ones.

Accuracy and Class Skew: The performance measure most commonly used in context recognition research is the overall *accuracy*. It is defined as the total number of correctly classified instances divided by the total number of instances. We argue that it is not well suited for context recognition applications with class skew. Consider a simple two class problem with a naive classifier that always says "class 1 occurred". If, in reality, both classes occur 100 times, the accuracy is 100/200 = 50%. If class 1 occurs 10'000 times and the other class occurs only 100 times, the accuracy is 10000/10100 = 99%. The same classifier produces a different accuracy for different class skews.

Note that for fields such as speech or character recognition, which also use accuracy, the above is an advantage rather than a problem. In both these fields it makes sense to have better accuracy for more common words. Additionally, the relative probabilities of classes (words, characters) are known and, in the long run, constant. Thus, the fact that accuracy reflects on how well the classifier is adapted to the class distribution is a desirable feature. This is not the case for many context recognition tasks where the class distribution is neither known nor constant, and fine tuning for better recognition of more common classes is not always desirable.

Contributions and Outline: We postulate that metrics which are less sensitive to class skew such as *R*eceiver *O*perating *C*haracteristic (ROC) and *A*rea *U*nder the *C*urve (AUC) should be used to evaluate and design context recognition systems. To make our case we will first introduce some common performance metrics and discuss on a theoretical basis how and why they are subject to class skew (Sec. A.2.1). We will then briefly describe *ROC* and *AUC* and explain why these metrics are better suited to deal with class skew than accuracy (Sec. A.2.2 and A.2.3). The data from the case study in Chapter 3 will then be used to show how the different metrics perform in a real-life scenarios. This includes both a two class and a multi-class problem (Sec. A.2.4 and A.3). Finally, open issues are outlined in Sec. A.4.

ROC analysis was originally developed for radar detection theory [126] but is also widely used as a method to measure performance in diagnostic systems [127–129], medical imaging [130–132] and machine learning [133, 134]. Even though the metrics considered in this chapter have been in existence for years, they have up to now hardly been used in the field of ubiquitous and wearable computing. [135, 136] are a few rare examples. Worth mentioning is also [137], where the authors use ROC curves to deal with unknown costs for interruptability at learning time.

A.2. Two Class Problem

A.2.1. Illustration of the Problem

Metrics: We assume to observe two classes H_0 and H_1 which occur with the *a priory probabilities* $P_0 = P(H_0)$ and $P_1 = P(H_1)$, respectively. Each class is represented by one or more features R with a probability density function (pdf) $f_{r|H_i}(R|H_i)$. The task of a classifier is to 'guess' the true

Table A.1. *Confusion Matrix.*

	$\hat{H}_1$	$\hat{H}_0$
H_1	**T**rue **P**ositives	**F**alse **N**egatives
H_0	**F**alse **P**ositives	**T**rue **N**egatives

class H_i based on the observation of R. In other words: it maps each instance (or observation) R to a hypothesis $\hat{H}_i$. The decision is correct if $\hat{H}_i = H_i$. For each observed instance there are four possible outcomes, as illustrated by the confusion matrix in Table A.1. Each field contains the number of instances, that are classified as $\hat{H}_i$, given H_i. Based on Table A.1 we list some very well known metrics:

$$tpr = recall = \frac{TP}{TP+FN} \text{ (true positive rate)} \tag{A.1}$$

$$fpr = \frac{FP}{FP+TN} \text{ (false positive rate)} \tag{A.2}$$

$$precision = \frac{TP}{TP+FP} \tag{A.3}$$

$$accuracy = \frac{TP+TN}{TP+FN+FP+TN} \tag{A.4}$$

$$norm\_acc = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{FP+TN}\right) \tag{A.5}$$

Since these metrics stem from detection theory which was originally developed for detection of adverse airplanes on radar, *tpr* is also called *hit rate* and *fpr* also *false alarm rate*. Additionally, $miss\ rate = 1 - tpr = \frac{FN}{TP+FN}$ is defined. The normalized accuracy *norm_ acc* tries overcome class skew by first calculating the recognition rate for each class individually.

To illustrate how these metrics perform in the presence of class skew, we use the example from the introduction: we consider a classifier A, which always guesses “class 1 occurred”. A second classifier B always says “class 0 occurred”. Both classifiers are presented with three different class distributions: α: both classes occur 100 times, β: class 1 occurs 10’000 times but class 0 only 100 times and γ: class 1 100 times and class 0 10’000 times. Table A.2 lists the calculated metrics. As far as accuracy is concerned, we observe that depending on the class distribution, every accuracy can be achieved and that tuning a classifier to achieve a high accuracy would favor often occurring classes.

Table A.2. *Metrics for Classifier A and B, class distribution α, β, γ.*

Metrics	A.α	A.β	A.γ	B.α	B.β	B.γ
$tpr =$	1	1	1	0	0	0
$fpr =$	1	1	1	0	0	0
$precision =$	0.5	0.99	0.01	–	–	–
$accuracy =$	0.5	0.99	0.01	0.5	0.01	0.99
$norm\_acc =$	0.5	0.5	0.5	0.5	0.5	0.5

Source of Class Skew: In context recognition, there are two ways class skew can be introduced into the recognition system:

1. A given classifier is presented with different class distributions.
2. At various times, a classifier is trained with different class distributions.

The first kind of class skew was illustrated in the example in Table A.2. A classifier A, which has been previously 'trained', i.e. has fixed rules of how to make a decision, is presented with class distributions α, β or γ. Clearly, accuracy is subject to class skew and changes even though the classifier does not. Normalized accuracy $norm\_acc$ is constant for all three class distributions. As we will show in Sec. A.2.2, normalized accuracy is immune to the first kind of class skew. Judging from Table A.2, it even seems to be constant for different classifiers. This however is due to the fact that classifier A and B use a similar decision criterion.

To understand the second kind of class skew and to show how ROC analysis deals with it, we use Bayesian decision theory [93] to briefly introduce a simple model of classifier training. First, we assume that the classifier to be trained is presented with a representative training set of features R. From this set, a classifier tries to estimate the pdf $f_{r|H_i}(R|H_i)$ of the features and the a priori probabilities P_i for class H_i. Based on those estimates, a classifier calculates the a posteriori probability $P(H_i|R)$ that a new observation R belongs to class H_i. Finally, the hypothesis $\hat{H}_i$ that belongs to the class with the highest $P(H_i|R)$ is chosen. Mathematically this is expressed by the MAP (maximum a posteriori) criterion[1]

$$P(H_1|R) \underset{H_0}{\overset{H_1}{\gtrless}} P(H_0|R) \tag{A.6}$$

It can be shown that the MAP criterion also minimizes the total probability of error [138]. Using Bayes' theorem and rearranging equation (A.6) results in

$$\frac{f_{r|H_1}(R|H_1)}{f_{r|H_0}(R|H_0)} \underset{H_0}{\overset{H_1}{\gtrless}} \frac{P_0}{P_1} = \eta \tag{A.7}$$

which is often referred to as the *likelihood ratio test*. Equation (A.7) corresponds to making a decision based on the computation of the likelihood ratio for the observed value R and then comparing it to a threshold η.

It illustrates the second kind of class skew: for different a priori probabilities P_i, i.e. for different class distributions, the threshold η needs to be chosen differently. If the classifier is trained a second time with a different

---

[1] Whereby equal costs for a wrong classification ($C_{01} = C_{10} = 1$) and no costs for a correct detection ($C_{00} = C_{11} = 0$) are assumed.

class distribution, it will choose a different threshold. Similarly, if a classifier is presented with a different class distribution than in the training phase, the threshold is not optimal anymore. In the confusion matrix, a different threshold will result in a changed ratio of instances in the first and second column.

To illustrate this further, we assume normally distributed pdfs[2], i.e. $f_{r|H_0}(R|H_0) \sim N(0, \sigma^2)$ and $f_{r|H_1}(R|H_1) \sim N(m, \sigma^2)$, and solve equation (A.7). It can be shown that the optimal threshold is:

$$\delta = \frac{m}{2} + \frac{\sigma^2}{m} \ln \frac{P_0}{P_1} \tag{A.8}$$

Hypothesis $\hat{H}_0$ is chosen if $R < \delta$ and $\hat{H}_1$ if $R > \delta$. Only if both symbols are equally probable, i.e. $P_0 = P_1$ then $\delta = \frac{m}{2}$. Fig. A.1 shows the pdfs of the observed instances R, the threshold δ and the areas for a wrong classification.

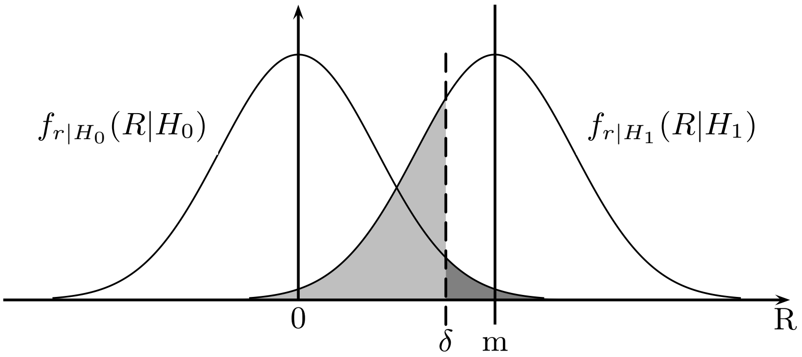


Figure A.1. *Probability density function $f_{r|H_i}(R|H_i)$ and threshold δ.*

A.2.2. Receiver Operating Characteristic (ROC)

Detection theory has long known ROC curves as a means to compare different detectors and to depict the trade-off between between *true positive rate* and *false positive rate* (ROC curves plot *tpr* against *fpr*, cf. Fig. A.2). Besides this capability, their advantage is their independence of class skew. A detailed description of ROC graphs for classifier evaluation can be found in [134] and [139–141].

---

[2] In detection theory this example is known as 'MAP detector for On-Off Keying in AWGN channel'. There, a binary source which produces the symbols $H_0 = 0$ (Off) and $H_1 = m$ (On) is considered. After the symbols are altered by a probabilistic transition mechanism the resulting symbols R which have then a probability density function (pdf) of $f_{r|H_i}(R|H_i)$ are observed. In an additive white gaussian noise (AWGN) channel, the pdfs will be normally distributed.

Independence of Class Skew: To show that ROC curves are not susceptible to class skew, we will analyze how they overcome the two different kind of class skew described in Sec. A.2.1. First, consider the case where a trained classifier is presented with different class distributions. The class distribution is the ratio of the number of instances in the first row to the ones in the second row of the confusion matrix in Table A.1. If the class distribution changes, but the detector always applies the same threshold η, the ratio of true and false decisions within one row does not change. Since *fpr* and *tpr* are metrics that are each based on *one* row, they are independent of the first kind of class skew. For the same reason, normalized accuracy stays constant as long as only the ratio of the classes in the test set is changed. On the other hand, a metric based on a column of Table A.1 is subject to class skew, as Table A.2 proves for accuracy and precision.

The reason why ROC curves are not subject to the second kind of class skew, is that they are parametric plots of *fpr* against *tpr* using the threshold η (and hence the class distribution) as parameter. ROC curves cover the whole of prior probabilities P_i, i.e. η is varied from 0 to ∞, while the curve is plotted from point (1,1) to (0,0). If the class distribution of the training set changes, the operating point on the ROC curve changes, but not the curve itself.

We illustrate this further in case of the MAP detector with normally distributed pdfs (see equation (A.8) and Fig. A.1). The probability of a FP and TP is given by

$$fpr = \int_{\delta}^{\infty} f_{r|H_0}(R|H_0)dR = Q\left(\frac{\delta}{\sigma}\right) = Q\left(\frac{\ln\eta}{d} + \frac{d}{2}\right) \tag{A.9a}$$

$$tpr = \int_{\delta}^{\infty} f_{r|H_1}(R|H_1)dR = Q\left(\frac{\delta - m}{\sigma}\right) = Q\left(\frac{\ln\eta}{d} - \frac{d}{2}\right) \tag{A.9b}$$

with $Q(x) = \frac{1}{\sqrt{2\pi}} \int_{x}^{\infty} \exp(-\frac{x^2}{2})dx$, $d = \frac{m}{\sigma}$, and $\eta = \frac{P_0}{P_1}$. Fig. A.2 shows the ROC curves for different values of d. For any given d, the statistics of fpr and *tpr* vary as $\eta = \frac{P_0}{P_1}$ is varied.

For most classifiers such as Naive Bayes, neural networks or decision trees, ROC curves can be generated without having to train the classifier with different class distributions. These ranking or scoring classifiers produce at their output the probability of an instance belonging to class H_i. A threshold is used to generate a binary decision: if the classifier output is below the threshold, the classifier chooses hypothesis $\hat{H}_0$, else $\hat{H}_1$. Each

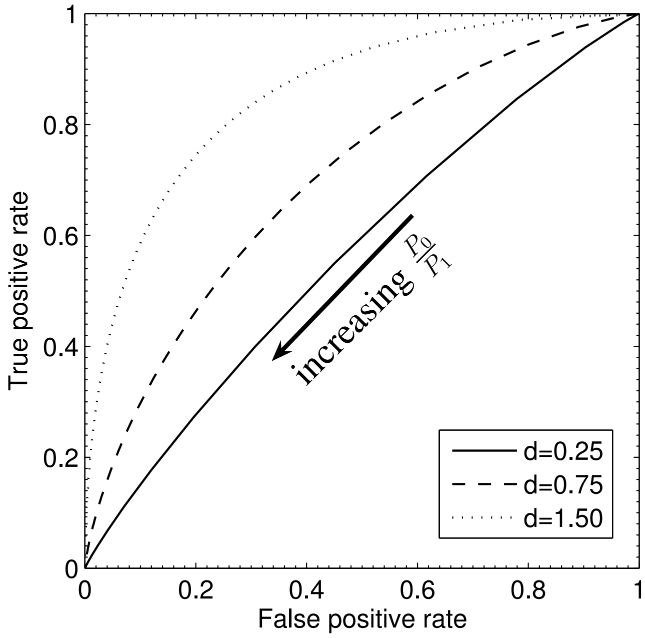


Figure A.2. *ROC graph for the MAP detector with normally distributed pdfs and different signal to noise ratios* $d^2 = \frac{m^2}{\sigma^2}$.

threshold produces a confusion matrix and therefore a point in the ROC space. Consequently, if we vary the threshold, we get a whole ROC curve[3].

It can be shown, that varying the threshold at the output of a ranking classifier is equivalent to varying the class probabilities P_0 and P_1. In case the classifier produces a posteriori probabilities $P(H_i|R)$, equation (A.6) can be rewritten to[4]:

$$P(H_1|R) \underset{H_0}{\overset{H_1}{\gtrless}} k \cdot P(H_0|R) \tag{A.10}$$

with k related to the threshold at the classifier's output. This changes the term η in equation (A.7) to $\eta = k \cdot \frac{P_0}{P_1}$. Therefore, varying the threshold at the classifier's output, i.e. varying k, can be seen as simulating a change in class distribution.

---

[3] Many classifiers which per se do not produce scores like decision tree or ruled generators, can easily converted into scoring classifiers, cf. [140]. If a classifier produces only a binary decision, a system engineer would need to simulate different underlying class distributions to get a ROC curve.

[4] Most ranking classifiers do not produce a posteriori probabilities but scores which express the degree to which an instance R is a member of H_1. Usually, there exists a monotonic function which allows to transform scores into a posteriori probabilities. This monotonic transformation will yield the same ROC curve [126] and therefore all ROC curves generated by varying the threshold at the output of a classifier are immune to class skew.

Comparison of Classifiers: As mentioned before, a ROC graph also provides a means to compare different detectors or classifiers. Generally, the points on the diagonal from (0,0) to (1,1) represent a classifier that randomly guesses a class. Any classifier that performs better lies in the upper triangle, with the optimum at (0,1). Fig. A.2 shows that the detectors with increasing d (i.e. increasing signal to noise ratio $d^2 = \frac{m^2}{\sigma^2}$) perform better: i.e. for a given *fpr* the *tpr* increases.

An important property of ROC curves is that the slope at a particular point is equal to the threshold η (and therefore the class distribution) required to achieve the *tpr* and *fpr* of that point [138]:

$$\frac{d\,tpr}{d\,fpr} = \frac{P_0}{P_1} = \eta \tag{A.11}$$

This property is useful if one classifier does not outperform another classifier so clearly as in Fig. A.2. Consider the example in Fig. A.3. Compared to classifier 1 and 2, classifier 3 is never optimal. But the decision between classifier 1 and 2 depends on the expected class distribution. Classifier 1 is optimal for a class distribution with more occurrences of class H_0 than H_1; classifier 2 for one biased towards H_1. As an example, two slopes for a class distribution of $H_0 : H_1 = 5 : 1$ and $1 : 5$, respectively, are plotted. Graphically, we see that the line with slope 0.2 needs to be shifted down until it touches the curve of classifier 1 and therefore classifier 2 is the

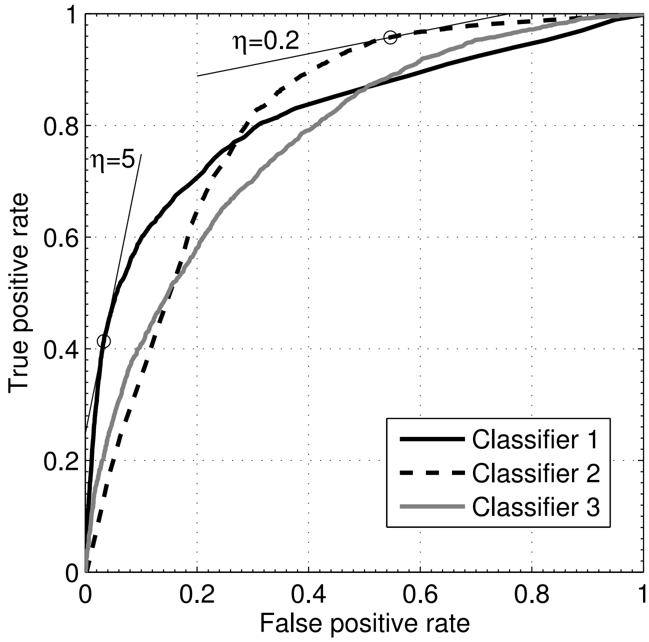


Figure A.3. *ROC curves allow comparison of classifiers: Depending on the class distribution* $\eta = \frac{P_0}{P_1}$ *either classifier 1 or 2 performs better.*

better choice for $\eta = 0.2$. Mathematically, the lines with slopes $\frac{P_0}{P_1}$ have the following equation [141]:

$$tpr = \frac{P_0}{P_1} fpr + \frac{accuracy - P_0}{P_1} \tag{A.12}$$

Therefore, given a fixed slope, a classifier is better if the corresponding tangent has a higher *tpr*-axis intercept point than another classifier. An algorithm, called *ROC Convex Hull Method*, which systematically explores this property is described in [139].

Cost for Wrong Decisions: ROC graphs allow an easy integration of cost for a correct or wrong decision. We denote C_{ij} the cost for for choosing $\hat{H}_i$ if H_j is produced. Instead of the MAP criterion in equation (A.6), Bayes criterion [138] is used which minimizes the average expected costs. The *likelihood ratio test* is then

$$\frac{f_{r|H_1}(R|H_1)}{f_{r|H_0}(R|H_0)} \underset{H_0}{\overset{H_1}{\gtrless}} \frac{P_0(C_{10} - C_{00})}{P_1(C_{01} - C_{11})} \tag{A.13}$$

which in terms of ROC curves is simply a modification of the class ratio $\frac{P_0}{P_1}$ and therefore ROC curves are not altered by varying costs. Alternative representations are for example discussed in [142] and [143].

A.2.3. Area Under the Curve (AUC)

Very often, it is more convenient to have one number representing the performance of the classifier than a whole graph. In connection with ROC graphs the area under the ROC curve (AUC) is often used since it is also independent of class skew. AUC is a measure that compares the average performance of a classifier. More specifically, “the AUC of a classifier is equivalent to the probability that the classifier will rank a randomly chosen instance from H_1 higher than a randomly chosen instance from H_0” [140]. In other words, a classifier with a higher AUC means that the classifier performs better on average, although in certain regions of the *fpr*–*tpr* plane another classifier with lower AUC might perform better.

This can be seen in Fig. A.3: classifier 3 outperforms classifier 1 in certain regions (for high *fpr*) but in average classifier 1 is better since it has a larger AUC. In this case, $AUC_{c1} = 0.81$, $AUC_{c2} = 0.82$ and $AUC_{c3} = 0.78$. For a random classifier (diagonal line between (0,0) and (1,1)) AUC is 0.5 and any other classifier should have $AUC > 0.5$.

A.2.4. Case Study

We used our data set II described in Sec. 3.1.2 to illustrate how ROC curves and AUC perform compared to accuracy in a realistic scenario.

Again, the task was to classify different activities by analyzing the sound of the kitchen appliances. Since the recorded sound is almost independent of the user, we use only sound (in contrast to sound and acceleration data) to compare performance metrics under different class skews. In doing so, we can be sure that differences in the results are not caused by accidentally choosing a 'wrong' set of users who perform an activity differently than another user group (see also Sec. A.4).

Obviously, if two classes are perfectly separable, i.e. if the pdfs in Fig. A.1 do not overlap, a reasonable classifier will always produce a recognition accuracy of 1 – independent of the class skew. Therefore, to show the effects of class skew, the goal was to achieve a low separability between the two classes. From our previous work [63], we expected some confusion between the sounds from the coffee maker and the hot water nozzle since they are both generated by the same water pump. Furthermore, we used a set of parameters (sampling frequency, feature set, etc.) that would not give the highest possible separability between the two classes[5]. Otherwise, we would have been able to achieve $AUC = 0.94$. However, using a non-optimized parameter set is in no way unrealistic nor does it mean to tune our data to favor certain metrics: On one hand, there are many applications where classes are not perfectly separable – no matter how good the parameter set is chosen. On the other hand, limited resources on a wearable device might inhibit the use of performance optimized parameters.

From the recorded data, different training and test sets were compiled and classified with a Naive Bayes classifier. We used different class skews to simulate different user preferences: some prefer coffee and others tea. Differences in the distribution of the classes in the training and the test set could occur in a scenario where a 'coffee drinker' hands his training system to a 'tea drinker'.

Table A.3 lists the individual accuracy of the two classes H_0: coffee maker and H_1: hot water nozzle. Furthermore, the overall *accuracy*, normalized accuracy *norm_ acc* and AUC are listed. Fig. A.4 shows the corresponding ROC curve calculated for the different class skews.

We observe, that only ROC curves – and consequently AUC as well – are independent of class skew. On the other hand, accuracy is susceptible to class skew. That is because accuracy is based on a single point on the ROC curve: i.e. a classifier that is trained with a $H_0 : H_1 = 1 : 3$ set, chooses a threshold that is optimal for this class distribution. As shown in Sec. A.2.1, this threshold is not optimal anymore for another class distribution. In our example accuracy ranges from 66% to 85%: by simply changing the frequency of occurrence of the classes, accuracy can be increased. But this does not necessarily mean that the system performs

---

[5] Here we used $f_s = 2.5\,\text{kHz}$, $N = 64$ and features SRF, 2^{nd} and 4^{th} BER.

Table A.3. *Comparison of metrics for the two-class problem.*

Training $H_0 : H_1$ Testing $H_0 : H_1$	1:1 1:1	1:3 1:3	1:3 3:1	3:1 3:1	3:1 1:3
accuracy for H_0	0.83	0.69	0.68	0.91	0.92
H_1	0.75	0.90	0.91	0.60	0.57
accuracy	0.79	0.85	0.74	0.83	0.66
norm_acc	0.79	0.80	0.80	0.76	0.75
AUC	0.87	0.88	0.87	0.87	0.87

better, nor that it is a fair metric to compare different systems. Normalized accuracy is resistant to class skew as long as the class distribution in the training set does not change. Since *precision* is subject to class skew (Sec. A.2.2), the common *precision-recall* graphs are also subject to class skew – as illustrated in Fig. A.5.

This case study also shows, that due to the real valued nature of the input signals and the limited data set, small deviations from the ideal behavior exist. For example, *AUC* should be constant for all class distributions although we calculated values from 0.871 to 0.877. Furthermore, the accuracy for H_1 should be the same in the last two columns of Table A.3, but they differ by almost 3 percentage points.

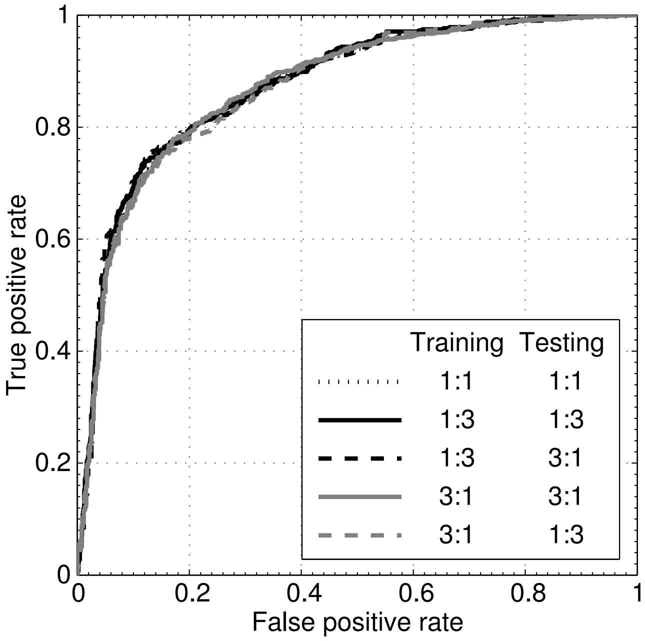


Figure A.4. *ROC curves of the case study: immune to class skew.*

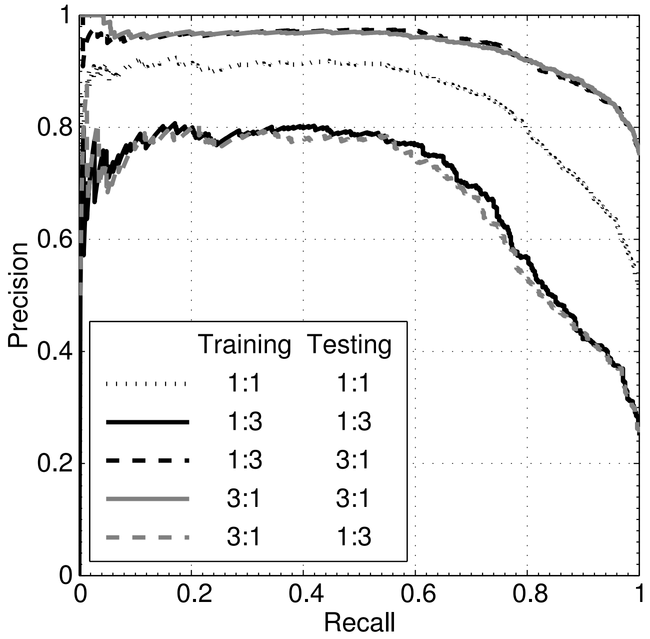


Figure A.5. *Precision-Recall curves of the case study: dependent on class distribution and therefore subject to class skew.*

A.3. Multi Class Problem

A.3.1. Multi-class ROC

If we have M classes, we get an $M \times M$ confusion matrix with M fields for correct classifications and $M^2 - M$ fields for errors [144, 145]. In general, this leads to a ROC surface with $M^2 - M$ dimensions [146], which even with 3 classes is a 6-dimensional surface and therefore not practical to depict.

A.3.2. Multi-class AUC

The generalization of the two class AUC is the Volume Under the ROC Surface (VUS). However, since this metric is rather tedious to calculate, two other means have been proposed. Both are based on projecting the *classification results* down to a set of two-dimensional ROC curves and averaging the AUC for these curves. Therefore the analysis of a multi-class problem is reduced to a two class problem:

- **1-vs-rest:** From the set of all classes C, class c_i is chosen as the positive class H_1 and all other classes are assumed to be the negative class H_0. Then, the $AUC(c_i)$ of these two 'classes' is calculated. This process is repeated for every class $c_i \in C$, i.e. $M = |C|$ times. The $AUC(c_i)$ values are weighted by c_i's prevalence $p(c_i)$ and averaged to get a multi-class AUC [140].

$$MAUC_1 = \sum_{i=0}^{M-1} AUC(c_i) \cdot p(c_i) \tag{A.14}$$

Although easy to calculate, the drawback is that this metric is sensitive to class skew, since the negative class includes several classes.

- **1-vs-1:** From the set of classes C, two classes c_i and c_j are picked out from the classification result and the pairwise $AUC(c_i, c_j)$ is calculated. Then, the multi-class AUC is calculated as the unweighted average over all possible $M(M-1)$ pairs [147]:

  $$MAUC_2 = \frac{1}{M(M-1)} \sum_{\{c_i, c_j\} \in C} AUC(c_i, c_j) \tag{A.15}$$

  This metric is independent of class skew.

A.3.3. Case Study Continued

Similar to Sec. A.2.4 we compared different metrics for a multi-class problem in Table A.4. This time, a Naive Bayes classifier was trained and tested with three different classes: 'c_0: coffee maker', 'c_1: hot water nozzle' and 'c_2: water tap'. The results are similar to those in Table A.3. It can be seen that both accuracy and normalized accuracy are susceptible to class skew, although the normalized accuracy is not affected by class skew as long as the ratio of the classes in the training set is constant. From the multi-class metrics, only $MAUC_2$ stays the same for all class skews.

As mentioned in Sec. A.2.4, the effects of class skew cannot be observed in perfectly separable classes. Therefore we used a parameter set that did

Table A.4. *Results of the case study for the 3-class problem.*

Training $c_0 : c_1 : c_2$	1:1:1	1:1:1	1:1:5	1:1:5
Testing $c_0 : c_1 : c_2$	1:1:1	1:1:5	1:1:1	1:1:5
accuracy for class c_0	0.71	0.69	0.62	0.63
c_1	0.55	0.56	0.53	0.53
c_2	0.92	0.92	0.95	0.95
accuracy	0.73	0.84	0.70	0.84
norm_acc	0.73	0.72	0.70	0.70
$MAUC_1$	0.89	0.96	0.88	0.95
$MAUC_2$	0.89	0.88	0.88	0.88

not achieve the highest possible separation[6]. Using an optimized set, we were able to achieve $MAUC_2 = 0.96$.

A.3.4. Theoretical Example

To emphasize the aforementioned results, we simulated different class skews for training and testing in an theoretical example. A Naive Bayes classifier was trained and tested with normally distributed random numbers, i.e. $f_{r|H_i}(R|H_i) \sim N(i,1)$. Table A.5 summarizes the results. Again, we see that all metrics but $MAUC_2$ are subject to class skew.

Table A.5. *Comparison of metrics for the multi-class problem in a theoretical example: Naive Bayes classifier and classes normally distributed:* $f_{r|H_i}(R|H_i) \sim N(i,1)$.

Training $c_0 : c_1 : c_2$ Testing $c_0 : c_1 : c_2$	1:1:1 1:1:1	1:1:1 1:10:1	1:10:1 1:1:1	1:10:1 1:10:1
accuracy for class c_0	0.70	0.71	0.05	0.04
c_1	0.38	0.38	0.99	0.99
c_2	0.69	0.69	0.04	0.04
accuracy	0.59	0.43	0.36	0.84
norm_acc	0.59	0.59	0.36	0.36
$MAUC_1$	0.77	0.66	0.77	0.66
$MAUC_2$	0.77	0.77	0.77	0.77

A.4. Discussion

In the previous sections we have shown that only ROC graphs, AUC or in case of a multi-class problem MAUC$_2$ are immune to class skew. Two assumptions were made:

1. the same kind classifier is used and it picks the same features for different class distributions
2. the pdf of the features stays constant for different class distributions

The first assumption might not always be correct. Assume that a classifier is presented several data streams (i.e. observed features are vectorial $\mathbf{R}$), each of which represents different characteristics of the recognition problem. If the classifier is allowed to distribute different weights to the data streams, it might do so differently for two different class distributions.

---

[6] We used $f_s = 2.5\,\text{kHz}$, $N = 64$ and features ZCR, BW, SRF, 2$^{\text{nd}}$ and 3$^{\text{rd}}$ BER.

Whether the second assumption holds depends on the source of the data. Imagine a user learning sign language. He will improve by using the sign language *more often* and therefore perform the signs in a *smoother* way. In this case, especially if the features are badly chosen, the pdf of the underlying data will change as the class distribution changes over time. In other cases, the pdf of the data to classify does not depend on class skew. In auditory scene recognition, for example, the features are calculated from environmental sound. These features are independent of how long a user stays in each location and therefore the second assumption is fulfilled.

This discussion shows that even though ROC graphs and AUC provide means to handle class skew, a system designer still has to be careful about the implications a change in the class distribution imposes on the system's performance. Even more care needs to be taken if *accuracy* or *normalized accuracy* is to be used, e.g. to perform a quick feasibility study or to overcome the computing time for multi-class AUC. In this case, an interpretation of the result needs to take into account the class distribution in the training and testing sets. Moreover, one has to be aware that another class distribution will lead to a different result.

A.5. Summary

We propose to use ROC analysis to design context recognition systems and to use AUC or multi-class AUC to represent the performance of a system. We have showed in theory and with the help of a case study that these metrics are independent of class skew. As we have argued in the beginning, many applications in context recognition will have varying class skew. Although ROC analysis is widely used in other research areas, in the field of wearable computing accuracy is still the dominant metric to express the performance of a system – even though it heavily depends on the distribution of the classes in the training and test phase.

B

Definitions

B.1. Definition of Features

B.1.1. Time-domain features

- **Zero-crossing Rate zcr** is defined as the number of zero crossings within a frame.
- **Mean-crossing Rate mcr** is defined as the number of mean crossings within a frame.
- **Fluctuation of Amplitude fluc** gives an indication of how much the signal oscillates around the mean value in relationship to the mean value.

$$fluc = \frac{mean(\mathbf{x})}{stdev(\mathbf{x})} \tag{B.1}$$

- **Root mean square rms** is defined as

$$rms = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x^2[i]}. \tag{B.2}$$

B.1.2. Frequency-domain features

Let $X[i]$ denote the i^{th} frequency component of the N-point FFT of $\mathbf{x}$. For a real valued input $\mathbf{x}$, $X[i]$ and $X[N-i]$ are conjugate complex. Therefore, if only magnitudes of the fourier components $|X[i]|$ are considered, the spectrum can be represented with just the first $K = \frac{N}{2}$ components[1].

- **Fourier components FFTcomp** If the fourier coefficients are used as features we mean the magnitude of the first $K = \frac{N}{2}$ components

$$FFTcomp = \big|X[i]\big| \text{ for } i = 1 \dots K \tag{B.3}$$

- **Fluctuation of amplitude spectra FLUC-S** is similarly defined as fluctuation in the time domain:

$$FLUC - S = \frac{mean(\big|\mathbf{X}\big|)}{stdev(\big|\mathbf{X}\big|)} \tag{B.4}$$

- **Frequency Centroid FC** represents the balancing point of the spectral power distribution:

$$FC = \frac{\sum_{i=1}^{K} i \cdot \big|X[i]\big|}{\sum_{i=1}^{K} \big|X[i]\big|} \tag{B.5}$$

  This feature correlates with the zero-crossing rate feature [39].

- **Bandwidth BW** is defined as the range of frequencies that the signal occupies. More precisely, it is the the square root of the power-weighted average of the squared difference between the spectral components and the frequency centroid:

$$BW = \sqrt{\frac{\sum_{i=1}^{K} (i - FC)^2 \cdot \big|X[i]\big|^2}{\sum_{i=1}^{K} \big|X[i]\big|^2}} \tag{B.6}$$

- **Spectral Rolloff Frequency SRF** is the 'frequency' h below which resides TH percent of the total power.

$$SFR = max\Big(h\Big|\sum_{i=1}^{h} \big|X[i]\big|^2 < TH \sum_{i=1}^{K} \big|X[i]\big|^2\Big) \tag{B.7}$$

  In accordance with [43] we chose the threshold $TH = 0.93$.

---

[1] Note that we use MATLAB notation and start vectors with index 1.

- **Band Energy Ratio BER** is the ratio of the energy in a certain frequency band to the total energy. In this work we use 4 logarithmically divided subbands as in [43]. The BER in the m^{th} subband is defined as:

$$BER_m = \frac{\sum\limits_{i=a_m+1}^{a_{m+1}} \left|X[i]\right|^2}{\sum\limits_{i=1}^{K} \left|X[i]\right|^2} \quad \text{for } m = 1 \dots 4 \tag{B.8}$$

  where $a_m = 2^{m+log_2(K)-5}$. Logarithmic subband are usually preferred because they represent the human hearing process more accurately. However, for a classification task, linearly spaced subbands would also be possible [148].

- **Cepstrum Coefficients CEP** We use the term cepstrum as it was originally introduced by Bogert et al. [149] and define it as the Fourier transform of the logarithm of the spectrum of a signal[2]. The real valued cepstrum coefficients are defined as:

$$CEP[k] = \frac{1}{N} \sum_{n=1}^{N} \ln\Big(\left|X[n]\right|\Big) \cdot e^{-j\frac{2\pi(k-1)(n-1)}{N}} \tag{B.9}$$

  In this work we used the first six cepstrum coefficients: $k = 1 \dots 6$ One of the interesting properties of the cepstrum is that any periodicities in a spectrum will be sensed as specific components in the cepstrum. Therefore, the cepstrum is well suited for echo or vocal-pitch detection [150].

B.2. Definition of Classifiers

Let $\mathbf{y}$ denote the L dimensional input vector for the classifier and $y[l]$ the sample number l of the input vector $\mathbf{y}$, with $l = 1 \dots L$. In our case, $\mathbf{y}$ is either the L dimensional feature space (with $L = 5 \dots 10$, dependent on the feature set, cf. Table 4.2) or the result of the LDA transformed Fourier components (with $L = M - 1$ and M the number of classes, cf. Sec. 4.2.5).

---

[2] Many texts state that the cepstrum is the '*inverse* Fourier transform of the logarithm of the spectrum'. This is not the definition given in the original paper, but unfortunately widespread [8].

B.2.1. Nearest class-center classifier (NCC)

With the help of a training set $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_j, \ldots \mathbf{y}_J\}$ the center vectors $\mathbf{m}_i$ for each class c_i are calculated:

$$\mathbf{m}_i = \frac{1}{J_i} \sum_{\mathbf{y}_j \in c_i} \mathbf{y}_j \qquad \text{for } i = 1 \ldots M \tag{B.10}$$

with J the total number of vectors in the training set and J_i the number of training vectors for class c_i with $J = \sum_{i=1}^{M} J_i$.

A test point $\mathbf{y}$ is assigned to the class associated with the minimum Euclidean distance to the class center. The Euclidean distance to center vector $\mathbf{m}_i$ is calculated with

$$d_i = \left|\mathbf{y} - \mathbf{m}_i\right| = \sqrt{\sum_{l=1}^{L} \left(y[l] - m_i[l]\right)^2} \tag{B.11}$$

For the implementation on the microcontroller the square root was omitted to save computation time. The predicted class $\hat{c}_j$ is

$$\hat{c}_j = \underset{i=1\ldots M}{\operatorname{argmin}}(d_i) = \underset{i=1\ldots M}{\operatorname{argmin}} \left|\mathbf{y} - \mathbf{m}_i\right| \tag{B.12}$$

The NCC classifier requires $M \cdot L$ distance calculations. In our implementation the argmin calculation requires at most $M \cdot (M-1)/2$ comparisons, in the best case $M - 1$ comparisons.

B.2.2. k-nearest neighbor (kNN)

Given a training set $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_j, \ldots \mathbf{y}_J\}$ consisting of J prototype patterns or vectors of dimension L, a test point $\mathbf{y}$ is classified as class $\hat{c}_i$ if most of the k closest prototype patters are from class c_i [151]. 'Closest' refers to the Euclidean distance d_j between the test point $\mathbf{y}$ and a vector of the training set $\mathbf{y}_j$:

$$d_j = \left|\mathbf{y} - \mathbf{y}_j\right| \tag{B.13}$$

Thus, the kNN classifier needs to find the k smallest distances d_j. The class c_i to which most of the k smallest distances belong, is the predicted class $\hat{c}_i$.

The complexity of the distance calculations is $\mathrm{O}(J \cdot L)$. Sorting algorithms to sort the J distances d_j have a complexity between $\mathrm{O}(J \cdot \log_2(J))$ and $\mathrm{O}(J^2)$ [8]. For an overview of speed-optimized kNN classifiers we refer to [151].

B.2.3. Naive Bayes Classifier

The Naive Bayes Classifier assumes that each feature y_n is conditionally independent of every other feature y_m for $n \neq m$. This means that

$$P(y_n|c_i, y_m) = P(y_n|c_i) \tag{B.14}$$

The maximum a posteriori (MAP) decision rule is used to classify a feature vector: given an observed feature vector $\mathbf{y}$, the class $\hat{c}_j$ that is most probable is chosen:

$$\hat{c}_j = \underset{i=1\dots M}{\operatorname{argmax}} \left[P(c_i|\mathbf{y})\right] \tag{B.15}$$

Using Bayes' theorem and the assumption of conditional independence results in

$$\hat{c}_j = \underset{i=1\dots M}{\operatorname{argmax}} \left[\frac{P(c_i)}{P(\mathbf{y})} \cdot \prod_{n=1}^{L} P(y_n|c_i)\right] \tag{B.16}$$

We assume a Gaussian distribution for the conditional probabilities:

$$P(y_n|c_i) = \frac{1}{\sigma_{n,i}\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{y_n - \mu_{n,i}}{\sigma_{n,i}}\right)^2} \tag{B.17}$$

where the mean $\mu_{n,i}$ and the standard deviation $\sigma_{n,i}$ for each combination of feature and class is estimated from the training set.

The denominator $P(\mathbf{y})$ in equation (B.16) is constant for a all classes and can be omitted. Furthermore, applying a monotonically increasing function – like a logarithm and a multiplication with 2 – to equation (B.16) does not influence the argmax function. Thus, we can eliminate the exponential function in equation (B.17) and rewrite equation (B.16) as

$$\hat{c}_j = \underset{i=1\dots M}{\operatorname{argmax}} \left[\ln\left(2 \cdot P(c_i) \cdot \prod_{n=1}^{L} \frac{1}{\sigma_{n,i}\sqrt{2\pi}}\right) - \sum_{n=1}^{L}\left(\frac{y_n - \mu_{n,i}}{\sigma_{n,i}}\right)^2\right] \tag{B.18}$$

In this equation, the $\ln(\cdot)$ term can be pre-computed based on the training instances. An M dimensional vector is required to store the pre-computed $\ln(\cdot)$ terms. Additionally, two $L \times M$ matrices are required to store $\mu_{n,i}$ and $1/\sigma_{n,i}$.

To evaluate equation (B.18), a Naive Bayes classifier needs to calculate $M \cdot L$ subtractions, $M \cdot L$ multiplications (with $1/\sigma_{n,i}$) and $M \cdot L$ squares; furthermore M subtractions from the $\ln(\cdot)$ term are required. Compared with a NCC classifier (cf. equation (B.12)), classifying a feature vector with a Naive Bayes classifier is more complex, especially due to the additional $M \cdot L$ multiplications (see also Fig. 5.6).

B.3. Classifier Fusion Methods

Classifier fusion is necessary if the output of two or more classifiers need to be combined to give one single result. Classifier fusion is a well-studied field: for further literature we refer to [96, 152–156]. Most classifiers can provide a level of confidence they have for each class. We denote the confidence level from classifier Z for class c_i with $C_Z(i)$. However, these confidence levels may be incomparable among classifiers. One approach is to convert the confidences into a posteriori probabilities. The drawback is the computational complexity and the required large amount of training data. Therefore, most fusion strategies assign ranks to the classes in a linear ordering, based on the sorted confidence levels. For each classifiers Z, a class c_i receives a rank $R_Z(i)$ – with the highest rank belonging to the most likely class. Then, a score function calculates a score $S(i)$ over all classifiers for each class c_i based on the ranks. The class with the highest score wins:

$$\hat{c}_j = \underset{i=1\ldots M}{\operatorname{argmax}}(S) \tag{B.19}$$

Subsequent definitions will be explained with the following example: for a 4 class problem, classifier A provides the following confidence levels C_A=[$C_A(1)$, $C_A(2)$, $C_A(3)$, $C_A(4)$]=[0.35, 0.41, 0.10, 0.14] for one instance. Classifier B outputs C_B =[0.4, 0.15, 0.45, 0] for the same instance. The ranks are then calculated as R_A=[2, 3, 0, 1] and R_B=[2, 1, 3, 0] for classifier A and B, respectively.

Highest rank: The score of a class is equal to its highest rank:

$$S(i) = max\big(R_A(i), R_B(i)\big) \tag{B.20}$$

For our example, the scoring vector is S=[2, 3, 3, 1]. In our case, ties are solved by looking at the confidence levels. Thus, our implementation actually calculates $S(i) = max\big(C_A(i), C_B(i)\big)$. In the example class 3 wins.

Borda Count [96]: The Borda count for a class is the sum of the number of classes ranked below it:

$$S(i) = \sum_{\forall Z} R_Z(i) \tag{B.21}$$

The Borda count is a generalization of the majority vote and treats all classifiers and ranks equally. In our example, the scoring vector is S=[4, 4, 3, 1]. Again, we solved ties by comparing the confidence levels of the classes with the highest scores. Thus, in the example class 2 wins.

Logistic Regression [96]: Logistic Regression is a generalization of the Borda Count. The score is a weighted sum of ranks:

$$S(i) = \alpha_i + \sum_{\forall Z} \beta_{i,Z} \cdot R_Z(i) \tag{B.22}$$

The weights $\beta_{i,Z}$ indicate the significance of the classifier Z and are estimated using the training set. If the number of classes is large and/or the amount of available training data is small, the same weight for all classes can be used:

$$S(i) = \alpha + \sum_{\forall Z} \beta_Z \cdot R_Z(i) \tag{B.23}$$

Average of All Ranks [83]: This measure is based on the confidence levels. The metric assumes that the confidence levels are comparable among classifiers and averages the confidence levels of all ranks:

$$S(i) = \sum_{\forall Z} C_Z(i) \tag{B.24}$$

For our example, the scoring vector is S=[0.75, 0.41, 0.55, 0.29] and thus class 1 wins.

Average of Best Rank [83]: Similar to the 'Average of All Ranks', this metric compares the average ranking of the top choices of the classifiers.

$$S(i) = \begin{cases} \sum_{\forall Z} C_Z(i) & \text{if } c_i \text{ is ranked top by at least one classifier} \\ 0 & \text{if } c_i \text{ is never ranked top} \end{cases} \tag{B.25}$$

In our example: class c_2 and c_3 are ranked top by classifier A and B, respectively. Therefore, the scoring vector is S=[0, 0.41, 0.55, 0] and thus class 3 wins.

Probability Matrix: Like 'Logistic Regression', this method uses a training phase to get a prior knowledge of the quality of a classifier. In contrast to the other methods presented here, it uses neither rank nor confidence levels but operates on the decision of the classifier. For each classifier, a confusion matrix is produced by testing the training set with the classifier. An element $n_{ij}^{(Z)}$ of the confusion matrix denotes that $n_{ij}^{(Z)}$ samples of the true class c_i have been assigned to class c_j by classifier Z.

The conditional probability that an instance x belongs to the true class c_i if classifier Z assigns class c_j to it, is given by:

$$P\big(x \in c_i | Z(x) = c_j\big) = \frac{P\big(x \in c_i\big) \cdot P\big(Z(x) = c_j | x \in c_i\big)}{P\big(Z(x) = c_j\big)} = \frac{n_{ij}^{(Z)}}{\sum_{i=1}^{M} n_{ij}^{(Z)}}$$

Thus, we can build a *Probability Matrix* containing the elements $pm_{ij}^{(Z)} = P\big(x \in c_i | Z(x) = c_j\big)$, where $pm_{ij}^{(Z)}$ is the probability that the true class is c_i if classifier Z detects c_j. If two classifiers disagree on their output, i.e. classifier A chooses class c_k, while classifier B chooses class c_l, the decision is based on the Probability Matrix according to:

$$pm_{lk}^{(A)} \underset{c_k}{\overset{c_l}{\gtrless}} pm_{kl}^{(B)} \tag{B.26}$$

For example, if classifier A detects class 4 (but from training we know that with a high probability the true class could also be class 1, i.e. $pm_{14}^{(A)}$ is high) and classifier B detects class 1 (but with low probability that the true class is any other class, i.e. $pm_{41}^{(B)}$ is low) then $pm_{14}^{(A)} > pm_{41}^{(B)}$ and we assume that the true class was class 1.

B.4. Sound Pressure and Sound Intensity

B.4.1. Sound Pressure

Acoustical sound waves apply a force to air particles. *Sound pressure* p is defined as the fluctuating pressure superimposed on the static pressure (in air: atmospheric pressure) by the presence of sound. Sound pressure is a scalar quantity. The unit is Pascal; it is equal to a force F of one newton applied over an area A of one square meter [8].

The amplitude of sound pressure from a point source decreases in the free field proportional to the inverse of the distance r from that source:

$$p \propto \frac{1}{r} \tag{B.27}$$

For a plane progressive wave the sound pressure p is:

$$p = \rho \cdot c_s \cdot v \tag{B.28}$$

with ρ: the density of air, in kg/m^3 (1.204 kg/m^3 at 20℃),
c_s: the speed of sound, in m/s (343.4 m/s at 20℃),
v: the sound velocity or particle velocity, in m/s.

Usually, the term *sound pressure* refers to the root-mean-square *averaged* pressure deviation caused by a sound wave passing through a fixed point.

$$p_{rms} = \sqrt{\frac{1}{T}\int_0^T p^2(t)\,dt} \tag{B.29}$$

Sound pressure level (SPL), L_p, is a decibel scale based on a reference sound pressure p_0 calculated as:

$$L_p = 20\log_{10}\left(\frac{p_{rms}}{p_0}\right) \tag{B.30}$$

The reference sound pressure $p_0 = 2 \cdot 10^{-5}$ Pa corresponds to the threshold of human hearing and equals to a pressure variation of less than a billionth of atmospheric pressure (atmospheric pressure at sea level: 1013 mbar = $1.013 \cdot 10^5$ Pa).

Microphones and most sound measurement equipment convert pressure variations into a voltage which is proportional to the sound pressure. Therefore changes in sound pressure level can be calculated by:

$$\Delta L_p = 20\log_{10}\left(\frac{V_1}{V_2}\right) \tag{B.31}$$

where V_1 and V_2 are the measured voltage amplitudes. Similarly, our eardrums are sensitive to the sound pressure. Microphones are usually specified in *SPL* at 1 meter distance. Table B.1 states the specified SPLs and the sensitivity of the microphones that were used for the experiments and on the hardware platform. For comparison: a conversation measured at a distance of 1 m has a SPL of 40 to 60 dB; for a jet a SPL of 110 to 140 dB can be measured in 100 m distance [8].

Table B.1. *Specified SPLs and sensitivity of Sony ECM-C115 and Knowles Acoustics SP0103 microphone.*

Microphone	Sony ECM-C115	Knowles Acoustics SP0103
Noise Level	39 dB SPL	35 dB SPL
Maximum Sound Pressure [a]	110 dB SPL [c]	100 dB SPL [d]
Sensitivity [a,b]	10 mV/Pa [e]	79 mV/Pa [f]

[a] at 1 kHz, [b] open circuit output voltage level, [c] THD = 3%, [d] THD < 1%, [e] output impedance: 1.9 kΩ, [f] output impedance: 100 Ω

B.4.2. Sound Intensity

Sound intensity I is the time-averaged product of particle velocity v and sound pressure p.

$$\overrightarrow{I} = \frac{1}{T} \int_0^T p \cdot \overrightarrow{v} \, dt \tag{B.32}$$

Sound intensity is a vector quantity. It describes as a function of frequency the direction and the amount of net flow of acoustic energy at a position in a sound field [157].

The amount of sound intensity $|I|$ is the acoustic power P_{ac} per unit area A measured in watts per square meter (W/m^2). Neither eardrums nor microphones can convert sound intensity to voltage modulation[3].

For a spherical sound source in the free field, the amount of sound intensity as a function of distance r is:

$$|I_r| = \frac{P_{ac}}{A} = \frac{P_{ac}}{4\pi r^2} \tag{B.33}$$

Sound intensity is inversely proportional to the square of the distance from a point source: $|I| \propto 1/r^2$.

Sound intensity level, L_I, is the logarithmic measure of the sound intensity relative to the standard threshold of hearing intensity $I_0 = 10^{-12}\mathrm{W/m^2}$.

$$L_I = 10 \log_{10} \left(\frac{|I|}{I_0} \right) \tag{B.34}$$

B.4.3. Relationship between Sound Pressure and Intensity

Usually, there is no direct relationship between sound pressure and sound intensity [157]. However, for a free progressive wave in air (e.g. a plane wave traveling down a tube, or a spherical wave traveling outward from a source in free air) the relationship is given by:

$$|I| = \frac{p_{rms}^2}{\rho \cdot c_s} \tag{B.35}$$

As a consequence, sound intensity level L_I and sound pressure level L_p are only the same in case of a free progressive wave in air. In this case:

$$L_I = 10 \log_{10} \left(\frac{|I|}{I_0} \right) = 20 \log_{10} \left(\frac{p_{rms}}{p_0} \right) = L_p \tag{B.36}$$

---

[3] Sound intensity measurement equipment exists: the difficulty lies in measuring the particle velocity. Luckily, the particle velocity can be related to the pressure gradient using Euler's equation: $\frac{\partial v}{\partial t} = -\frac{1}{\rho} \frac{\partial p}{\partial r}$. The pressure gradient can be measured with two closely spaced microphones [157, 158].

Glossary

Symbols

A	area
C	capacity (Ah)
c_s	speed of sound (m/s)
c_i	class number i
$\hat{c}_j$	hypothesis; recognized class
$C_Z(i)$	confidence level of classifier Z for class c_i
d	distance
$f_{\mu\mathrm{C}}$	microcontroller main clock frequency
$f_{r\|H_i}(R\|H_i)$	pdf of features R for class H_i
f_s	microphone sampling frequency
I	sound intensity (W/m^2)
J	number of training instances or vectors
K	number of frequency domain samples $\left(K = \frac{N}{2}\right)$
L	dimensionality of feature space
L_I	sound intensity level in dB
L_p	sound pressure level in dB
M	number of classes
N	number of input samples (time domain)
p	sound pressure
P_0, P_1	a priory probability for class H_0 and H_1
P_{ac}	acoustic power radiated by a source
P_{avg}	average power
p_{rms}	sound pressure: root-mean-square average
r	distance
ρ	density of air (kg/m^3)
R	feature vector

$R_Z(i)$	rank of class c_i from classifier Z
$S(i)$	score of class c_i in classifier fusion methods
t	time
t_{calc}	execution time of a software routine; time to calculate features and classification
T_p	periodicity; number of classifications per second
t_s	start time of a frame relative to start of segment
t_w	observation window, frame length
v	sound velocity or particle velocity (m/s)
V	voltage
V_{BatNom}	nominal battery voltage
$x(t)$	continuous function
$\mathbf{x}$	vector of input samples
$x[i]$	sample number i of input vector
$\mathbf{X}$	vector of frequency components
$X[i]$	i^{th} frequency component of the FFT of $\mathbf{x}$

Abbreviations

A	ampere
acc	accelerometer
ADC	analog to digital converter
ASIC	application specific integrated circuit
AUC	area under (ROC) curve
BER	band energy ratio
BW	bandwidth
CEP	cepstrum coefficients
CFS	correlation-based feature selection
CM	confusion matrix
DAT	digital audio tape
DCO	digital controlled oscillator
DSP	digital signal processor

ECM	electret condenser microphone
FC	frequency centroid
FFT	fast fourier transformation
FFTcomp	magnitude of the fourier components
fluc	fluctuation (of amplitude)
FLUC-S	fluctuation (of amplitude spectra)
FN	false negatives
FP	false positives
FPGA	field-programmable gate array
fpr	false positive rate
fps	frames per second
GPS	global positioning system
HMM	hidden markov model
Hz	hertz
ITRS	international technology roadmap for semiconductors
kB	kilo byte (1’000 bytes)
kNN	k-nearest neighbor classifier
ksps	kilo samples per second
LDA	linear discriminant analysis
LPF	low-pass filter
LSTP	low standby power logic (ITRS categorization)
MAP	maximum a posteriori
MAUC	multi-class AUC
mcr	mean-crossing rate
MEMS	micro-electro-mechanical system
MFCC	mel-frequency cepstrum coefficient
MHz	mega hertz
MI	mutual information
mic	microphone
ms	milli-seconds
µC	microcontroller
NCC	nearest class center classifier

PC	personal computer
PCA	principal component analysis
PCB	printed circuit board
PDA	personal digital assistant
pdf	probability density function
PIDS	process integration, devices, and structures
RAM	random access memory
RFID	radio frequency identification
RMS	root mean square
ROC	receiver operating characteristic
SPL	sound pressure level
SRF	spectral rolloff frequency
THD	total harmonic distortion
TN	true negatives
TP	true positives
tpr	true positive rate
V	volt
VGA	video graphics array (640×480 pixels)
W	watt
zcr	zero-crossing rate

Bibliography

[1] M. Billinghurst, and T. Starner. Wearable devices: new ways to manage information. *IEEE Computer*, 32(1):57–64, Jan. 1999.

[2] C. Randell, and H. L. Muller. The well mannered wearable computer. *Personal and Ubiquitous Computing*, 6(1):31–36, Feb. 2002.

[3] N. Kern, B. Schiele, H. Junker, P. Lukowicz, and G. Tröster. Wearable sensing to annotate meeting recordings. *Proc. of the 6th IEEE Int'l Symposium on Wearable Computers (ISWC)*, pages 186–193, Oct. 2002.

[4] P. Lukowicz, U. Anliker, J. Ward, G. Tröster, E. Hirt, and C. Neufelt. AMON: A wearable medical computer for high risk patients. *Proc. of the 6th Int'l Symposium on Wearable Computers (ISWC)*, pages 133–134, Oct. 2002.

[5] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles. Towards a better understanding of context and context-awareness. *Proc. of the 1st Int'l Symposium Symposium on Handheld and Ubiquitous Computing (HUC)*, pages 304–307, Jun. 1999.

[6] B. Schilit, N. Adams, and R. Want. Context-aware computing applications. *Proc. of the IEEE Workshop on Mobile Computing Systems and Applications*, pages 85–90, Dec. 1994.

[7] J. Pascoe. Adding generic contextual capabilities to wearable computers. *Proc. of the 2nd Int'l Symposium on Wearable Computers (ISWC)*, pages 92–99, Oct. 1998.

[8] Wiktionary and Wikipedia. Online open content dictionary and free encyclopedia, 2006. URL: `http://www.wikimedia.org`.

[9] R. Polana, and R. Nelson. Recognizing activities. *Proc. of the 12th IAPR Int'l Conf. on Computer Vision and Image Processing*, volume 1, pages 815 – 818, Oct. 1994.

[10] A. F. Bobick. Movement, activity, and action: The role of knowledge in the perception of motion. *Phil. Trans. Royal Society London B, 352*, pages 1257–1265, 1997.

[11] H. Junker. *Human Activity Recognition and Gesture Spotting with Body-Worn Sensors*, volume 158 of *Series in Microelectronics.* Hartung–Gorre Verlag Konstanz, 2005.

[12] L. Bao, and S. S. Intille. Activity recognition from user-annotated acceleration data. *Proc. of the 2nd Int'l Conf. on Pervasive Computing*, pages 1–17, Apr. 2004.

[13] H. Junker, P. Lukowicz, and G. Tröster. PadNET: Wearable physical activity detection network. *Proc. of the 7th Int'l Symposium on Wearable Computers (ISWC)*, pages 244–245, Oct. 2003.

[14] K. van Laerhoven, N. Kern, H.-W. Gellersen, and B. Schiele. Towards a wearable inertial sensor networks. *In Proc. of the IEE Eurowearable*, pages 125–130, Sep. 2003.

[15] C. Randell, and H. Muller. Context awareness by analysing accelerometer data. *Proc. of the 4th Int'l Symposium on Wearable Computers (ISWC)*, pages 175–176, Oct. 2000.

[16] M. Sekine, T. Tamura, T. Fujimoto, and Y. Fukui. Classification of walking pattern using acceleration waveform in elderly people. *Proc. of the 22nd Annual Int'l Conf. of the IEEE Engineering in Medicine and Biology Society*, volume 2, pages 1356–1359, Jul. 2000.

[17] T. B. Moeslund, and E. Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81 (3):231–268, 2001.

[18] M. Chan, E. Campo, and D. Esteve. Monitoring elderly people using a multisensor system. *Proc. of 2nd Int'l Conf. on Smart Homes and Health Telematic (ICOST)*, pages 162–169, 2004.

[19] L. Bass, D. Siewiorek, A. Smailagic, and J. Stivoric. On site wearable computer system. *Conference companion of Conf. on Human Factors in Computing Systems*, pages 83–84, 1995.

[20] J. H. Garrett, and A. Smailagic. Wearable computers for field inspectors: Delivering data and knowledge-based support in the field. *Artificial Intelligence in Structural Engineering*, volume 1454 of *Springer Lecture Notes in Artificial Intelligence*, pages 146–164, 1998.

[21] D. Siewiorek, A. Smailagic, L. Bass, L. Siegel, R.Martin, and B. Bennington. Adtranz: A Mobile Computing System for Maintenance and Collaboration. *Proc. of the 2nd Int'l Symposium on Wearable Computers (ISWC)*, pages 25–32, Oct. 1998.

[22] J. A. Paradiso, and T. Starner. Energy scavenging for mobile and wireless electronics. *IEEE Pervasive Computing*, 4(1):18–27, Jan.–Mar. 2005.

[23] S. Roundy, D. Steingart, L. Frechette, P. Wright, and J. Rabaey. Power sources for wireless sensor networks. *Proc. of the 1st European Workshop on Wireless Sensor Networks (EWSN)*, pages 1–17, Jan. 2004.

[24] S. Roundy, P. K. Wright, and J. M. Rabaey. A study of low level vibrations as a power source for wireless sensor nodes. *Computer Communications*, 26(11):1131–1144, Jul. 2003.

[25] A. Sinha, N. Ickes, and A. P. Chandrakasan. Instruction level and operating system profiling for energy exposed software. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 11(6):1044–1057, Dec. 2003.

[26] R. Lee, and R. Nathuji. Power and performance analysis of pda architectures. Technical report, Advanced VLSI Computer Architecture Group, MIT, Dec. 2000. URL: `http://www.cag.csail.mit.edu/6.893-f2000/project/lee_final.pdf`.

[27] Texas Instruments. Microcontroller homepage, 2006. URL: `http://www.ti.com/hdr_p_micro`.

[28] M. Stäger, P. Lukowicz, N. Perera, T. von Büren, G. Tröster, and T. Starner. SoundButton: Design of a Low Power Wearable Audio Classification System. *Proc. of the 7th IEEE Int'l Symposium on Wearable Computers (ISWC)*, pages 12–17, Oct. 2003.

[29] M. Stäger, P. Lukowicz, and G. Tröster. Implementation and Evaluation of a Low-Power Sound-Based User Activity Recognition System. *Proc. of the 8th IEEE Int'l Symposium on Wearable Computers (ISWC)*, pages 138–141, Nov. 2004.

[30] P. Lukowicz, J. A. Ward, H. Junker, M. Stäger, G. Tröster, A. Atrash, and T. Starner. Recognizing workshop activity using body worn microphones and accelerometers. *Proc. of the 2nd Int'l Conf. on Pervasive Computing*, pages 18–22, Apr. 2004.

[31] J. A. Ward, P. Lukowicz, and G. Tröster. Gesture spotting using wrist worn microphone and 3-axis accelerometer. *Proc. of Smart Objects and Ambient Intelligence Conference (sOc-EUSAI)*, pages 99–104, Oct. 2005.

[32] D. Istrate, E. Castelli, M. Vacher, L. Besacier, and J.-F. Serignat. Information extraction from sound for medical telemonitoring. *IEEE Transactions on Information Technology in Biomedicine, Accepted for future publication*, 2006.

[33] M. Vacher, D. Istrate, L. Besacier, and J. F. Serignat. Sound detection and classification for medical telesurvey. *Proc. of 2nd Conf. on Biomedical Engineering (BIOMED)*, pages 395–398, Feb. 2004.

[34] J. Chen, A. H. Kam, J. Zhang, N. Liu, and L. Shue. Bathroom activity monitoring based on sound. *Proc. of the 3rd Int'l Conf. on Pervasive Computing*, pages 47–61, Apr. 2005.

[35] Z. Liu, Y. Wang, and T. Chen. Audio feature extraction and analysis for scene classification. *Journal of VLSI Signal Processing (Special issue on multimedia signal processing)*, pages 61–79, Oct. 1998.

[36] Z. Liu, J. Huang, Y. Wang, and T. Chen. Audio feature extraction and analysis for scene classification. *Proc. of IEEE 1st Workshop on Multimedia Signal Processing*, pages 343–348, Jun. 1997.

[37] T. Zhang, and C.-C. Jay Kuo. Audio content analysis for online audiovisual data segmentation and classification. *IEEE Transactions on Speech and Audio Processing*, 9(4):441–457, May 2001.

[38] E. Scheirer, and M. Slaney. Construction and evaluation of a robust multifeature speech/music discriminator. *Proc. of IEEE Int'l Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, pages 1331–1334, Apr. 1997.

[39] D. Li, I. Sethi, N. Dimitrova, and T. McGee. Classification of general audio data for content-based retrieval. *Pattern Recognition Letters*, 22 (5):533–544, 2001.

[40] M. Büchler, S. Allegro, S. Launer, and N. Dillier. Sound classification in hearing aids inspired by auditory scene analysis. *EURASIP Journal on Applied Signal Processing*, 18:2991–3002, 2005.

[41] N. Kern, A. Schmidt, and B. Schiele. Recognizing context for annotating a live life recording. *Personal and Ubiquitous Computing*, 2005.

[42] B. Clarkson, N. Sawhney, and A. Pentland. Auditory context awareness in wearable computing. *Workshop on Perceptual User Interfaces*, Nov. 1998.

[43] V. Peltonen, J. Tuomi, A. Klapuri, J. Huopaniemi, and T. Sorsa. Computational auditory scene recognition. *IEEE Int'l Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, pages 1941–1944, May 2002.

[44] A. Eronen, J. Tuomi, A. Klapuri, S. Fagerlund, T. Sorsa, G. Lorho, and J. Huopaniemi. Audio-based context awareness – acoustic modeling and perceptual evaluation. *IEEE Int'l Conf. on Acoustics, Speech, and Signal Processing (ICASSP '03)*, volume 5, pages 529–532, 2003.

[45] U. Anliker, J. F. Randall, and G. Tröster. Speaker separation and tracking system. *EURASIP Journal on Applied Signal Processing*, Article ID 29104, 14 pages, 2006.

[46] G. D. Abowd, A. K. Dey, R. Orr, and J. Brotherton. Context-awareness in wearable and ubiquitous computing. *Virtual Reality*, 3(3):200–211, 1998.

[47] A. Smailagic, and D. Siewiorek. System level design as applied to CMU wearable computers. *Journal of VLSI Signal Processing Systems*, 21(3): 251–263, Jul. 1999.

[48] S. Park, I. Locher, A. Savvides, M. Srivastava, A. Chen, R. Muntz, and S. Yuen. Design of a wearable sensor badge for smart kindergarten. *Proc. of the 6th Int'l Symposium on Wearable Computers (ISWC)*, pages 231–238, Oct. 2002.

[49] R. Krishna, S. Mahlke, and T. Austin. Architectural optimizations for low-power, real-time speech recognition. *Proc. of the Int'l Conf. on Compilers, Architecture and Synthesis for Embedded Systems*, pages 220–231, 2003.

[50] S. Nedevschi, R. Patra, and E. Brewer. Hardware speech recognition for user interfaces in low cost, low power devices. *Proc. of the 42nd Conf. on Design automation*, pages 684–689, Jun. 2005.

[51] A. Smailagic, D. Reilly, and D. P. Siewiorek. A system-level approach to power/performance optimization in wearable computers. *Proc. IEEE Computer Society Workshop on VLSI (WVLSI)*, pages 15–20, Apr. 2000.

[52] T. L. Martin, D. P. Siewiorek, A. Smailagic, M. Bosworth, M. Ettus, and J. Warren. A case study of a system-level approach to power-aware computing. *ACM Transactions on Embedded Computing Systems (TECS)*, 2 (3):255–276, Aug. 2003.

[53] U. Anliker, J. Beutel, M. Dyer, R. Enzler, P. Lukowicz, L. Thiele, and G. Tröster. A systematic approach to the design of distributed wearable systems. *IEEE Transactions on Computers*, 53(3):1017–1033, Aug. 2004.

[54] U. Anliker, H. Junker, P. Lukowicz, and G. Tröster. Design methodology for context-aware wearable sensor systems. *Proc. of the 3rd Int'l Conf. on Pervasive Computing*, pages 220–236, Apr. 2005.

[55] O. Cakmakci, J. Coutaz, K. van Laerhoven, and H.-W. Gellersen. Context awareness in systems with limited resources. *Workshop on Artificial Intelligence in Mobile Systems, Lyon, France*, pages 21–28, Jul. 2002.

[56] S. Ravindran, D. Anderson, and M. Slaney. Low-power audio classification for ubiquitous sensor networks. *IEEE Int'l Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 4, pages 337–340, May 2004.

[57] H. Junker, P. Lukowicz, and G. Tröster. Sampling frequency, signal resolution and the accuracy of wearable context recognition systems. *Proc. of the 8th IEEE Int'l Symposium on Wearable Computers (ISWC)*, pages 176–177, Nov. 2004.

[58] L. Doherty, B. Warneke, B. Boser, and K. Pister. Energy and performance considerations for smart dust. *Int'l Journal of Parallel and Distributed Systems and Networks*, 4(3):121–133, Dec. 2001.

[59] B. H. Calhoun, et al. Design considerations for ultra-low energy wireless microsensor nodes. *IEEE Transactions on Computers*, 54(6):727–740, Jun. 2005.

[60] N. B. Bharatula, M. Stäger, P. Lukowicz, and G. Tröster. Empirical Study of Design Choices in Multi-Sensor Context Recognition Systems. *Proc. of the 2nd Int'l Forum on Applied Wearable Computing (IFAWC)*, pages 79–93, Mar. 2005.

[61] R. Amirtharajah, and A. P. Chandrakasan. A micropower programmable DSP using approximate signal processing based on distributed arithmetic. *IEEE Journal of Solid-State Circuits*, 39(2):337–347, Feb. 2004.

[62] A. Krause, M. Ihmig, E. Rankin, S. Gupta, D. Leong, D. Siewiorek, and A. Smailagic. Trading off prediction accuracy and power consumption for context-aware wearable computing. *Proc. of the 9th IEEE Int'l Symposium on Wearable Computers (ISWC)*, pages 20–26, Oct. 2005.

[63] P. Lukowicz, H. Junker, M. Stäger, T. von Büren, and G. Tröster. WearNET: A distributed multi-sensor system for context aware wearables. *Proc. of the 4th Int'l Conf. on Ubiquitous Computing (UbiComp)*, pages 361–370, Sep. 2002.

[64] G. J. Pottie, and W. J. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58, May 2000.

[65] O. Amft, M. Stäger, P. Lukowicz, and G. Tröster. Analysis of chewing sounds for dietary monitoring. *Proc. of the 7th Int'l Conf. on Ubiquitous Computing (UbiComp)*, pages 56–72, Sep. 2005.

[66] C. Plessl, R. Enzler, H. Walder, J. Beutel, M. Platzner, L. Thiele, and G. Tröster. The case for reconfigurable hardware in wearable computing. *Personal and Ubiquitous Computing*, 7(5):299–308, Oct. 2003.

[67] N. B. Bharatula, M. Stäger, P. Lukowicz, and G. Tröster. Power and Size Optimized Multi-Sensor Context Recognition Platform. *Proc. of the 9th IEEE Int'l Symposium on Wearable Computers (ISWC)*, pages 194–195, Oct. 2005.

[68] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. *The 4th Int'l Conf. on Information Processing in Sensor Networks (IPSN/SPOTS)*, Apr. 2005.

[69] J. Hill, and D. Culler. Mica: A wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):12–24, Nov./Dec. 2002.

[70] L. Holmquist, et al. Building intelligent environments with Smart-Its. *IEEE Computer Graphics and Applications*, 24(1):56–64, Jan./Feb. 2004.

[71] G. Q. Maguire, M. T. Smith, and H. W. P. Beadle. Smartbadges: A wearable computer and communication system. *The 6th Int'l Workshop on Hardware/Software Co-design, Invited Talk*, Mar. 1998.

[72] M. Kohvakka, M. Hannikainen, and T. D. Hamalainen. Wireless sensor prototype platform. *Proc. of the 29th Annual Conf. of the IEEE Industrial Electronics Society (IECON '03)*, volume 2, pages 1499–1504, Nov. 2003.

[73] Homepage of the TEA project. Technology for Enabling Awareness, 2000. URL: `http://www.teco.edu/tea/`.

[74] O. Amft, H. Junker, and G. Tröster. Detection of eating and drinking arm gestures using inertial body-worn sensors. *Proc. of the 9th Int'l Symposium on Wearable Computers (ISWC)*, pages 160–163, Oct. 2005.

[75] M. Vacher, D. Istrate, L. Besacier, E. Castelli, and J. F. Serignat. Smart audio sensor for telemedicine. *Proc. of Smart Objects Conference (SOC)*, pages 222–225, May 2003.

[76] N. Julien, J. Laurent, E. Senn, and E. Martin. Power estimation of a C algorithm based on the functional-level power analysis of a digital signal processor. *Proc. of 4th Int'l Symposium on High Performance Computing (ISHPC)*, pages 354–360, May 2002.

[77] W. Ye, N. Vijaykrishnan, M. T. Kandemir, and M. J. Irwin. The design and use of simplepower: a cycle-accurate energy estimation tool. *Proc. of the 37th Conf. on Design Automation*, pages 340–345, 2000.

[78] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. *Proc. of the 27th Annual Int'l Symposium on Computer architecture (ISCA)*, pages 83–94, Jun. 2000.

[79] Homepage Xsens. miniature inertial motion sensors, 2006. URL: `http://www.xsens.com`.

[80] M. Schröder. SoundButton: Design eines Low-Power Wearable Audio Classification Systems. Master's thesis, Electronics Laboratory, ETH Zurich, 2004.

[81] C. Gutscher, and P. Hefti. Wearable Sensor Package 2.0. Semester thesis, Electronics Laboratory, ETH Zurich, 2005.

[82] Texas Instruments. *Documentation for the TI mixed signal microcontroller MSP430F1611*, 2005. URL: `http://www.ti.com`.

[83] G. Ogris, T. Stiefmeier, H. Junker, P. Lukowicz, and G. Tröster. Using ultrasonic hand tracking to augment motion analysis based recognition of manipulative gestures. *Proc. of the 9th IEEE Int'l Symposium on Wearable Computers (ISWC)*, pages 152–159, Oct. 2005.

[84] D. Roggen, N. B. Bharatula, M. Stäger, P. Lukowicz, and G. Tröster. From sensors to miniature networked SensorButtons. *Proc. of the 3rd Int'l Conf. on Networked Sensing Systems (INSS)*, Jun. 2006.

[85] K. Van Laerhoven, and H.-W. Gellersen. Spine versus porcupine: A study in distributed wearable activity recognition. *Proc. of the 8th IEEE Int'l Symposium on Wearable Computers (ISWC)*, pages 142–149, Nov. 2004.

[86] M. Cowling. *Non-Speech Environmental Sound Classification System for Autonomous Surveillance.* PhD thesis, Griffith University, Gold Coast Campus, 2004.

[87] B. Clarkson, and A. Pentland. Extracting context from environmental audio. *Proc. of the 2nd Int'l Symposium on Wearable Computers (ISWC)*, pages 154–155, Oct. 1998.

[88] N. Kwak, and C.-H. Choi. Input feature selection for classification problems. *IEEE Transactions on Neural Networks*, 13(1):143–159, Jan. 2002.

[89] M. Dash, and H. Liu. Feature selection for classification. *Intelligent Data Analysis*, 1(3):131–156, Aug. 1997.

[90] R. Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks*, 5(4):537–550, Jul. 1994.

[91] M. A. Hall, and L. A. Smith. Practical feature subset selection for machine learning. *Proc. of the 21st Australian Computer Science Conference*, pages 181–191, Feb. 1996.

[92] I. H. Witten, and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* Morgan Kaufmann, 1999.

[93] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification.* Wiley, New York, 2000.

[94] S. Balakrishnama, A. Ganapathiraju, and J. Picone. Linear discriminant analysis for signal processing problems. *Proc. of IEEE Southeastcon*, pages 78–81, 1999.

[95] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[96] T. Ho, J. Hull, and S. Srihari. Decision combination in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(1):66–75, 1994.

[97] M. Pedram, and J. M. Rabaey, editors. *Power Aware Design Methodologies.* Kluwer Academic Publishers, Boston, 2002.

[98] J. M. Rabaey, and M. Pedram, editors. *Low Power Design Methodologies.* Kluwer Academic Publishers, 1996.

[99] Y.-H. Lu, and G. De Micheli. Comparing system level power management policies. *IEEE Design and Test of Computers*, 18(2):10–19, Mar./Apr. 2001.

[100] A. Acquaviva, L. Benini, and B. Ricco. An adaptive algorithm for low-power streaming multimedia processing. *Proc. of Design, Automation and Test in Europe*, pages 273–279, Mar. 2001.

[101] J. L. nd Pai H. Chou, N. Bagherzadeh, and F. Kurdahi. A constraint-based application model and scheduling techniques for power-aware systems. *Proc. of the 9th Int'l Symposium on Hardware/Software Codesign (CODES)*, pages 153–158, Apr. 2001.

[102] M. D. Scott, B. E. Boser, and K. S. J. Pister. An ultralow-energy adc for smart dust. *IEEE Journal of Solid-State Circuits*, 38(7):1123–1129, Jul. 2003.

[103] A. Wang, and A. P. Chandrakasan. A 180-mV subthreshold FFT processor using a minimum energy design methodology. *IEEE Journal of Solid-State Circuits*, 40(1):310–319, Jan. 2005.

[104] J. Pouwelse, K. Langendoen, and H. Sips. Dynamic voltage scaling on a low-power microprocessor. *Proc. of the 7th ACM Int'l Conf. on Mobile Computing and Networking (Mobicom)*, pages 251–259, Jul. 2001.

[105] K. Roy, and M. C. Johnson. Software design for low power. *Nato Advanced Study Institutes Series on Low Power Design in Deep Sub-micron Electronics*, pages 433–460, 1997.

[106] V. Tiwari, S. Malik, and A. Wolfe. Compilation techniques for low energy: An overview. *Proc. of Symposium on Low-Power Electronics*, 1994.

[107] J. Flinn, and M. Satyanarayanan. Energy-aware adaptation for mobile applications. *Proc. of the 17th ACM Symposium on Operating Systems Principles*, pages 48–63, Dec. 1999.

[108] N. B. Bharatula, S. Ossevoort, M. Stäger, and G. Tröster. Towards wearable autonomous microsystems. *Proc. of the 2nd Int'l Conf. on Pervasive Computing*, pages 225–237, Apr. 2004.

[109] C. X. Huang, B. Zhang, A.-C. Deng, and B. Swirski. The design and implementation of PowerMill. *Proc. of the Int'l Workshop on Low Power Design*, pages 105–110, Apr. 1995.

[110] V. Tiwari, S. Malik, A. Wolfe, and T. Lee. Instruction level power analysis and optimization of software. *Journal of VLSI Signal Processing Systems*, 13(2):1–18, Aug. 1996.

[111] J. Russell, and M. Jacome. Software power estimation and optimization for high performance, 32-bit embedded processors. *Proc. of the Int'l Conf. of Computer Design (ICCD)*, pages 328–333, Oct. 1998.

[112] L. Benini, R. Hodgson, and P. Siegel. System-level power estimation and optimization. *Proc. of the Int'l Symposium on Low Power Electronics and Design (ISLPED)*, pages 173–178, Jun. 1998.

[113] A. Sinha, and A. Chandrakasan. Jouletrack - a web based tool for software energy profiling. *Proc. of the 38th Int'l Conf. on Design Automation*, pages 220–225, Jun. 2001.

[114] T. Simunic, L. Benini, and G. De Micheli. Cycle-accurate simulation of energy consumption in embedded systems. *Proc. of the 36th ACM/IEEE conf. on Design automation*, pages 867–872, 1999.

[115] T. L. Martin, and D. P. Siewiorek. Non-ideal battery properties and low power operation in wearable computing. *Proc. of the 3rd Int'l Symposium on Wearable Computers (ISWC)*, pages 101–106, Oct. 1999.

[116] V. Tiwari, S. Malik, and A. Wolfe. Power analysis of embedded software: a first step towards software power minimization. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2(4):437–445, Dec. 1994.

[117] H. V. Sorensen, D. L. Jones, M. T. Heideman, and C. S. Burrus. Real-valued fast fourier transform algorithms. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(6):849–863, Jun. 1987.

[118] H. Brashear, T. Starner, P. Lukowicz, and J. Junker. Using multiple sensors for mobile sign language recognition. *Proc. of the 7th IEEE Int'l Symposium on Wearable Computers (ISWC)*, pages 45–52, Oct. 2003.

[119] T. Blickle, J. Teich, and L. Thiele. System-level synthesis using evolutionary algorithms. *Design Automation for Embedded Systems*, 3(1):23–58, Jan. 1998.

[120] P. E. Ross. 5 commandments: The rules engineers live by weren't always set in stone. *IEEE Spectrum*, 40(12):30–35, 2003.

[121] ITRS. International technology roadmap for semiconductors, 2005. URL: `http://public.itrs.net`.

[122] A. M. Holberg, and A. Saetre. Innovative techniques for extremely low power consumption with 8-bit microcontrollers. Technical report, Atmel Corporation, Feb. 2006. URL: `http://www.atmel.com/dyn/resources/prod_documents/doc7903.pdf`.

[123] M. J. Wolf, H. Reichl, J. Adams, and R. Aschenbrenner. *The World of Electronic Packaging and System Integration*, chapter : From Packaging to System Integration - The Paradigm Shift in Microelectronics, pages 94–103. ddp goldenbogen, Jan. 2004.

[124] Chapter 'System Drivers' of ITRS, pages 6–8, 2003. URL: `http://public.itrs.net/Files/2003ITRS/SysDrivers2003.pdf`.

[125] M. Stäger, P. Lukowicz, and G. Tröster. Dealing with class skew in context recognition. *Proc. of the 6th Int'l Workshop on Smart Appliances and Wearable Computing (IWSAWC)*, Jul. 2006.

[126] J. Egan. *Signal Detection Theory and ROC Analysis.* Academic Press, New York, 1975.

[127] C. E. Metz. Basic principles of ROC analysis. *Seminars in Nuclear Medicine*, 8(4):283–298, Oct. 1978.

[128] J. Swets. Measuring the accuracy of diagnostic systems. *Science*, 240 (4857):1285–1293, Jun. 1988.

[129] M. Greiner, D. Pfeiffer, and R. Smith. Principles and practical application of the receiver-operating characteristic analysis for diagnostic tests. *Preventive Veterinary Medicine*, 45(1):23–41, May 2000.

[130] J. Swets. ROC analysis applied to the evaluation of medical imaging techniques. *Investigative Radiology*, 14(2):109–121, Mar./Apr. 1979.

[131] C. E. Metz. ROC methodology in radiologic imaging. *Investigative Radiology*, 21(9):720–733, Sep. 1986.

[132] K. Woods, and K. Bowyer. Generating ROC curves for artificial neural networks. *IEEE Transactions on Medical Imaging*, 16(3):329–337, Jun. 1997.

[133] A. P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, Jul. 1997.

[134] F. Provost, and T. Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42(3):203–232, Mar. 2001.

[135] J. Mäntyjärvi, M. Lindholm, E. Vildjiounaite, S.-M. Mäkelä, and H. Ailisto. Identifying users of portable devices from gait pattern with accelerometers. *Proc. of the IEEE Int'l Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, pages 973–976, Mar. 2005.

[136] S. Thiemjarus, B. P. L. Lo, K. V. Laerhoven, and G. Z. Yang. Feature selection for wireless sensor networks. *Proc. of the 1st Int'l Workshop on Wearable and Implantable Body Sensor Networks*, Apr. 2004.

[137] A. Bernstein, P. Vorburger, and P. Egger. Direct interruptability prediction and scenario-based evaluation of wearable devices: Towards reliable interruptability predictions. *Proc. of the 1st Int'l Workshop on Managing Context Information in Mobile and Pervasive Environments*, May 2005.

[138] H. L. Van Trees. *Detection, Estimation and Modulation Theory: Part I.* Wiley, New York, 2001. ISBN 0-471-09517-6.

[139] F. Provost, and T. Fawcett. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. *Proc. of the 3rd Int'l Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 43–48, Aug. 1997.

[140] T. Fawcett. ROC Graphs: notes and practical considerations for researchers. Technical Report previous: HPL-2003-4, HP Laboratories, Palo Alto, CA, USA, Mar. 2004.

[141] N. Lachiche, and P. Flach. Improving accuracy and cost of two-class and multi-class probabilistic classifiers using ROC curves. *Proc. 20th Int'l Conf. on Machine Learning (ICML)*, pages 416–423, Jan. 2003.

[142] P. Domingos. MetaCost: a general method for making classifiers cost-sensitive. *Proc. of the 5th Int'l Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 155–164, Aug. 1999.

[143] C. Drummond, and R. C. Holte. Explicitly representing expected cost: an alternative to ROC representation. *Proc. of the 6th Int'l Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 198–207, Aug. 2000.

[144] T. Lane. Extensions of ROC analysis to multi-class domains. *ICML Workshop on Cost-Sensitive Learning*, Jul. 2000.

[145] A. Srinivasan. Note on the location of optimal classifiers in n-dimensional ROC space. Technical Report PRG-TR-2-99, Oxford University Computing Laboratory, England, 1999.

[146] P. A. Flach. The many faces of ROC analysis in machine learning. ICML Tutorial: The 21st Int'l Conf. on Machine Learning, 2004. URL: `http://www.cs.bris.ac.uk/~flach/ICML04tutorial/index.html`.

[147] D. J. Hand, and R. J. Till. Simple generalisation of the area under the ROC curve for multiple class classification problems. *Machine Learning*, 45(2):171–186, Nov. 2001.

[148] M. Büchler. *Algorithms for Sound Classification in Hearing Instruments.* PhD thesis, ETH Zurich, 2002.

[149] B. P. Bogert, M. J. R. Healy, and J. W. Tukey. The quefrency alanysis of time series for echoes: cepstrum, pseudo-autocovariance, cross-cepstrum, and saphe cracking. *Proc. of the Symposium on Time Series Analysis*, pages 209–243, 1963.

[150] A. V. Oppenheim, and R. W. Schafer. From frequency to quefrency: a history of the cepstrum. *IEEE Signal Processing Magazine*, 21(5):95–106, Sep. 2004.

[151] S. Warfield. Fast k-nn classification for multichannel image data. *Pattern Recognition Letters*, 17(7):713–721, 1996.

[152] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.

[153] L. Xu, A. Krzyzak, and C. Y. Suen. Methods of combining multiple classifiers and their application to handwriting recognition. *IEEE Transactions on Systems Man and Cybernetics*, 22:418–435, 1992.

[154] D. Ruta, and B. Gabrys. An overview of classifier fusion methods. *Computing and Information Systems*, 7(1):1–10, Feb. 2000.

[155] L. I. Kuncheva. A theoretical study on six classifier fusion strategies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2): 281–286, 2002.

[156] K. Chen, L. Wang, and H. Chi. Method of combining multiple classifiers with different features and their applications to text-independent speaker recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 11(3):417–445, 1997.

[157] Science & Engineering Encyclopedia, 2006. URL: `http://www.diracdelta.co.uk`.

[158] Brüel & Kjær. Sound intensity, 1993. URL: `http://www.bksv.com/pdf/Sound_Intensity.pdf`.

Acknowledgments

I would like to express my gratitude to my advisor, Prof. Dr. Gerhard Tröster, for his confidence in me, his support and for giving me leeway in my research. Special thanks go to Prof. Dr. Paul Lukowicz for many fruitful discussions and his countless ideas for my work. I greatly appreciate his support during the preparation of scientific papers and the opportunity to work with him for six months at his research institute.

I would like to thank all my colleagues at the ETH for the pleasant and inspiring research atmosphere and their valuable support. Special thanks go to my colleagues Urs Anliker, Holger Junker, Thomas von Büren and Marc von Waldkirch who were part of the original Wearable Computing Lab team and accompanied me over the years. Further thanks go to my colleagues Oliver Amft, Patrick de la Hamette, Evelyne Isenschmid, Ivo Locher, Clemens Lombriser, Thomas Stiefmeier and Marco Wirz, who all enriched my life at the ETH.

Finally, I would like to express my gratitude to Verena – without her encouragement and patience this work would not have been possible.

Zürich, April 2006 Mathias Stäger

Curriculum Vitae

Personal Information

Stäger Mathias
Born August 11th, 1975, Aarau, Switzerland
Citizen of Zizers GR, Switzerland

Education

2000–2006	Ph.D. studies in Information Technology and Electrical Engineering (Dr. sc. ETH Zürich) at the Swiss Federal Institute of Technology (ETH), Zurich, Switzerland
1995–2000	M.Sc. studies in Electrical Engineering (Dipl. El.-Ing. ETH) at ETH Zurich, Switzerland
1991–1995	Mathematisch-Naturwissenschaftliches Gymnasium at the Alte Kantonsschule Aarau, Switzerland
1982–1991	Primary and secondary school in Buchs AG, Switzerland

Professional Experience

2000–2006	Research and teaching assistant at the Electronics Laboratory, ETH Zurich, Switzerland
2005	Research Internship at the Institute of computer systems and networks at the University for Health Sciences, Medical Informatics and Technology (UMIT), Hall in Tirol, Austria
1998	Research Internship at Ascom Systec AG, Division 'Applicable Research & Technology', Mägenwil, Switzerland: Channel equalization for HIPERLAN receiver
1996	Internship at Müller Martini, Zofingen, Switzerland: Course in basic engineering skills

Low-Power Sound-Based User Activity Recognition

Tiny computers and sensors integrated into our clothes are likely to have a major impact on our life in the future. By monitoring our physical condition and activities they will allow to provide us assistance with everyday tasks or warn us of dangerous situations: They will recognize that we are running to catch a train and might tell us that we don't need to hurry because the train is delayed as well. They will follow the work steps of a maintenance worker and warn him in case he is about to make a hazardous mistake.

For such sensor nodes to be of any use, they need to recognize our activities with a high accuracy. Furthermore, they should be fully autonomous – operating for months or years on a miniature battery. Unfortunately, systems tuned to achieve high recognition accuracy are likely to consume a lot of power. This work presents an empirical design methodology to optimize a context recognition system with respect to a trade-off between power consumption and recognition performance rather than straightforward maximization of the recognition rate.

Activities do not necessarily have to be recognized with the help of motion sensors. Many tools we use everyday produce a distinct sound, be it an electric shaver, a coffee machine or a photocopier. Even more, many of our actions are accompanied by a clear and distinct sound, be it typing on a keyboard, brushing teeth or closing a door. Thus, in this work we use sound as a novel modality for user activity recognition. As we will demonstrate for several scenarios, sounds that are caused by the user or occur in close proximity to the user's hand can be picked up with a wrist-worn microphone.

About the Author

Mathias Stäger received the Dipl.-Ing. (M.Sc.) degree in electrical engineering from ETH Zurich, Switzerland, in 2000. He joined the Wearable Computing Laboratory of the Department of Information Technology and Electrical Engineering at ETH Zurich in Summer 2000 as a research and teaching assistant. His research was focused on context recognition, low-power wearable systems and wireless body sensor networks. In 2006, he received the Dr. sc. ETH (Ph.D.) degree from ETH Zurich.