ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

MIT Media Lab

Real-time Context Recognition

by

Mark Lawrence Blum

Submitted to the
Department of Information Technology and Electrical Engineering,
Swiss Federal Institute of Technology Zurich (ETH)
on October 24th, 2005,
in partial fulfillment of the requirements for the degree of

Master of Science

Abstract

The aim of wearable computing is to build intelligent machines that provide automatic and autonomous support in people's everyday lives. Accurate and comprehensive recognition of a user's context is an important step towards that goal. In this thesis a light-weight PDA-based wearable system is presented that classifies in real-time location, speech, posture and a number of activities, based on audio and acceleration sensing. After a pre-classification step, which is focused on the four categories, a common sense model is used to improve overall recognition accuracy in a post-classification step. The context information won is used in a diary application to trigger picture taking and audio recording of interesting moments of the wearer's life.

ETH thesis supervisor: Prof. Gerhard Tröster
MIT Media Lab supervisor: Prof. Alex (Sandy) Pentland

Foreword

ETH Zurich is a great school, and I enjoyed studying there. But my desire to experience and explore other cultures drove me abroad several times: First, a semester of studies at Chalmers University of Technology in Gothenburg, Sweden, then an internship with PSA Peugeot Citroën in Paris, and now, I've come back after six months of research at the MIT Media Lab in Cambridge, Massachusetts.

It was an interesting class by Professor Tröster that fostered my interest in wearable computing and context recognition. Many thanks to him for supporting my wish to do my Master's thesis abroad.

A big thank-you to Professor Pentland for being a kind and encouraging advisor and making it possible for me to come to the Media Lab. I appreciate the freedom and the opportunities I was given and enjoyed working in his group.

Then I would like to thank all the members of the Human Dynamics Group for listening to my cause and for their inputs, especially Jonathan Gips for solving many of my compile and run-time problems, Wen Dong for his mathematical expertise and the discussions on common sense modeling, Anmol Madan for teaching me all about the Zaurus and Ron Caneel for getting me going with the speech code.

Let me also cheer for my friends in Boston, who enriched my life and give me reason to come back: Tilke, Tracy, Philipp, Tom, Olivier, Dan and Yann. Thank you for the good times.

I am especially grateful to my parents, for their love and care, for paving the road for a privileged education and for unconditionally supporting my endeavors abroad. They never had the chance to go to university. I hope they can share with me the joy of graduation.

And last but certainly not least, I thank my charming girlfriend Simone, who spent three months with me in Boston. Her love and support, over long distance or short, gave me strength and helped me in some rough moments. We had a great time together!

Thesis goal

The fundamental goal of this thesis is to

- recognize certain dimensions of an individual's context
- using acceleration and audio sensors
- worn unobtrusively on few places on the body
- in a way suitable for real-world data.

This requires the assembly of a prototype hardware platform, collection of a large amount of user data, data analysis, construction of classification algorithms and implementation of real-time algorithms on the wearable.
The selected dimensions of context are *location*, *speech*, *posture* and *activities*. The labels within these categories are listed below. Labels within a category are regarded as mutually exclusive.

Location	**Speech**	**Posture**	**Activities**
office	no speech	unknown	no activity
home	user speaking	lying	eating
outdoors	other speaker	sitting	typing
indoors	distant voices	standing	shaking hands
restaurant	loud crowd	walking	clapping hands
car	laughter	running	driving
street		biking	brushing teeth
shop			doing the dishes

One focal point is to make use of "common sense dependencies" between the different dimensions of context. For instance, the activity typing is more likely to occur while we are sitting, rather than standing. Although sitting does not necessarily infer typing, typing does actually infer sitting. Such influences between different dimensions of context must be modeled appropriately and taken into account when conceiving a system that is to recognize such a diverse notion of context.

One particular motivation for this project is to use the recognized context to predict moments of interest in a user's everyday life. This can be used in an automated diary application that captures pictures or perhaps video and audio recordings of events interesting to the user.

The results of my work lead to the following thesis statements:

1. Ambient audio and two accelerometers, worn on hip and wrist, are sufficient to successfully classify posture and most of the labels proposed in the categories speech and activities.

2. Starting from a set of single classifiers for different types of context (location, speech, posture and activities), the overall accuracy in classifying a user's context can be improved by using correlations between those different types of context.

3. A comprehensive classification of a user's context helps predict moments of interest, which can be used to trigger audio and video recordings.

Table of Contents

Abstract **1**

Foreword **2**

Thesis goal **3**

Table of Contents **5**

List of figures **8**

List of tables **9**

1 Background and related work **11**

1.1 Healthcare 11
1.2 Interruptability 11
1.3 Diary applications 12
1.4 On-body versus environmental sensing 13
1.5 Sensor selection 13
1.6 Data collection and annotation 14

2 Wearable platform **15**

2.1 The PDA 16
2.2 Accelerometers 17
2.3 Audio and images 18
2.4 WiFi sniffing with Kismet 19
2.5 Battery life 19

3 Data acquisition and annotation **21**

3.1 Signal recording application 21
3.2 Accelerometer calibration 22
3.3 Interval-contingent experience sampling 22
3.4 Offline annotation tool 22
3.5 The data collected 24
3.6 Enhanced sensor configuration 24

4 Classification architecture and implementation **26**

4.1 Classification architecture 26
4.2 Implementation 27

5 Pre-classification: focus on each category 29
5.1 Classification methods 29
5.2 Posture 30
5.3 Activities 33
5.4 Speech 36
5.5 Location 38
5.6 Cross-validation datasets 40
6 Post-classification: Modeling common sense 42
6.1 Common sense dependencies 42
6.2 Initial position of post-classification 44
6.3 Static approach with joint probabilities 45
6.4 Static approach with pairwise joint probabilities 47
6.5 Dynamic approach with the influence model 51
6.6 Final comparison 55
7 Interest prediction 58
7.1 What are interesting moments? 58
7.2 Interest prediction algorithm 59
7.3 Experiment description 60
7.4 Results 61
8 Future work 63
8.1 Classification architecture 63
8.2 More data more systems 64
8.3 Common Sense modeling 64
8.4 New features and additional sensors 64
8.5 Other classifiers 65
8.6 Annotation tool 65
9 Summary 67
References 69
Appendix A Data Formats 71
A.1 Files generated by SignalRecorder 71
A.2 Files related to the offline annotation tool 73
A.3 Data representation in Matlab 74
Appendix B Matlab code 75
B.1 Main scripts 75
B.2 Functions 76
Appendix C C-Code 78

C.1 enchant-base 78
C.2 enchant-dev 78
C.3 inference 78
C.4 SignalRecorder 80

Appendix D Offline Annotation Tool 82

Appendix E Data collection and annotation tutorial 83

E.1 SignalRecorder 83
E.2 Copying the data to disk 83
E.3 Creating the Wave Clips 84
E.4 Offline Annotation Software 84

Appendix F Detailed comparison of four microphones 86

Appendix G Interest prediction experiment 89

G.1 Picture set A 89
G.2 Picture set B 94

List of figures

Figure 2-1: Sensor placement 15
Figure 2-2: Hardware setup with PDA, battery, ... 17
Figure 2-4: Camera and image examples with and without fisheye lens..... 19
Figure 3-1: Zaurus screenshots. System setup (left) and classifier outputs (right) 21
Figure 3-2: Sreenshot of annotation tool 23
Figure 4-1: Classification architecture 27
Figure 4-2: Software structure of real-time implementation 28
Figure 5-1: Feature space for posture 31
Figure 5-2: Feature space for activities..... 33
Figure 5-3: Feature space for speech 37
Figure 5-4: Feature space for location 39
Figure 6-1: Timing for post-classification 44
Figure 6-2: Pairwise conditional probabilities..... 48
Figure 6-3: Influence matrix learned by EM-algorithm 53
Figure A-1: Example of files generated by SignalRecorder and the annotation tool..... 73
Figure F-2: Comparison of spectrograms using four different microphones..... 87
Figure F-3: Comparison of FFTs using four different microphones 88

List of tables

Table 2-1: Technical specifications of the PDA 16
Table 3-1: Overview of the data acquired 25
Table 4-1: The four classification categories with labels 26
Table 5-1: Posture confusion matrix, Bayes classifier 31
Table 5-2: Posture confusion matrix, C4.5 classifier 32
Table 5-3: Activities confusion matrix, Bayes classifier with single Gaussian 34
Table 5-4: Activities confusion matrix, Bayes classifier with mixture of three Gaussians 34
Table 5-5: Activities confusion matrix, C4.5 classifier 35
Table 5-6: Activities confusion matrix, ergodic 2-state HMMs 36
Table 5-7: Speech confusion matrix, Bayes classifier 37
Table 5-8: Speaking / non-speaking with Bayes classifier 38
Table 5-9: Speech confusion matrix, C4.5 classifier 38
Table 5-10: Location confusion matrix, Bayes classifier 39
Table 5-11: Location confusion matrix, c4.5 classifier 40
Table 5-12: Cross-validation posture confusion matrix (Bayes classifier) 41
Table 5-13: Cross-validation activities confusion matrix (Bayes classifier) 41
Table 5-14: Cross-validation speech confusion matrix (Bayes classifier) 41
Table 5-15: Cross-validation location confusion matrix (Bayes classifier) 41
Table 6-1: Pre-classification accuracies 45
Table 6-2: Post-classification results using joint probabilities P(A,B,C,D) and correct labels 46
Table 6-3: Post-classification using joint probabilities P(A,B,C,D) and pre-classification decisions 46
Table 6-4: Post-classification results using P(A|B,C,D) only 47
Table 6-5: Post-classification results using P(A|B) etc. in a convex combination 49
Table 6-6: Post-classification results using P(A|B)^(1/k) 49
Table 6-7: Post-classification results using P(A|B)^(1/k) and certainty measures 50
Table 6-8: Post-classification using P(A|B)^(1/k) and certainty measures over three iterations 51
Table 6-9: Post-classification results using P(A|B)^(1/k) and certainty measures over three iterations, with correct location labels 51
Table 6-10: Post-classification results using the influence model over the last 3 timesteps 54

Table 6-11: Post-classification results using the influence model over the last 7 timesteps 54
Table 6-12: Location confusion matrix after pre-classification 55
Table 6-13: Speech confusion matrix after pre-classification 55
Table 6-14: Posture confusion matrix after pre-classification 55
Table 6-15: Activities confusion matrix after pre-classification 55
Table 6-16: Location confusion matrix after post-classification 56
Table 6-17: Speech confusion matrix after post-classification 56
Table 6-18: Posture confusion matrix after post-classification 56
Table 6-19: Activities confusion matrix after post-classification 57
Table 7-1: Assignment of interest points 59
Table 7-2: Number of biking images 61

1 Background and related work

The basic aim of wearable computing is to build intelligent machines that provide automatic and autonomous support in people's everyday lives. Accurate and comprehensive recognition of a user's context is an important step towards that goal. In the following, several applications of wearable computing and their dependence on context recognition are discussed.

1.1 Healthcare

A promising field for wearable computing lies in healthcare: Tracking health and wellness, from elder care to personal fitness. Extremely high costs of healthcare and an increasing shortage of caregivers are the main drivers. Opportunities are long-term monitoring, behavior and physiology trending, real-time proactive feedback and alert systems. For diagnosis and medical studies a history of continuous sensor readings and activity patterns is more accurate than a periodic questionnaire and can very well complement information won by verbal discussions with the patient. Medical professionals believe that one of the best ways to detect emerging conditions before they become critical is to look for changes in the activities of daily living (ADLs, [1]) such as eating, getting in and out of bed, toileting, grooming, shopping, and housekeeping. An overview of the group's current effort in healthcare is provided in [2], [3] and [4].

1.2 Interruptability

A potential of context-aware mobile devices is to proactively present information when the user needs it and is least interrupted. An increasing number of devices and applications are competing for the user's attention: phone calls, reminders, email notifications, voice messages, instant messaging services, news or changes in the stock market. Such interruptions quickly become annoying and can decrease work performance. It is thus a goal to minimize the perceived interruption burden of proactively delivered messages. Knowing about the user's state of activity can be used to reduce this burden and increase the value of receiving a notification. It has been shown that receptivity to interruptions is higher if they occur at activity transitions rather than at random times [5]. In [6] acceleration, audio and location sensing was used to assess personal and social interruptability. Rather than modeling the precise context a combination of tendencies was used to find the best notification modality e.g. vibrating or ringing.

1.3 Diary applications

Another promise of wearable computing, particularly in combination with today's mass data storage possibilities and ever increasing network bandwidth, are diary applications, also known as lifelogs:

> Wouldn't it be fantastic if a person could easily recall every moment of his or her life? Given that we live in such a technologically advanced society, why should a person ever have to forget what happened, when it was, where it happened, who was there, why it happened, and how he or she felt? If there was a system that could record everything a person sees, hears, and feels, how would that enhance our lives? We have enhanced our eye sight with prescription glasses, our hearing with hearing aids, and our timekeeping ability with watches. Enhancement of personal memories seems to be a natural next step [7].

What has been published in 1945 under the name of MEMEX (short for memory extender) is the first device for storing information that is electronically linked to a library and able to display books and films from the library and automatically follow cross-references from one work to another [8]. This "enlarged intimate supplement to memory" is a first form of what other people today call myLifeBits [9], Memories for Life [10] or What Was I Thinking [11]. A list of projects in this field can be found on the Continuous Archival and Retrieval of Personal Experiences (CARPE) website [12].

Most of these projects focus on organizing, categorizing and searching a massive load of random personal data. Techniques suggested are speech and image recognition, mostly in combination with common sense networks or other sorts of data mining as well as audio features that try to capture more the mood of a situation rather than the content. The key problem that is tackled lies in filtering information that is relevant and interesting to the user.

Now how about shifting this problem from offline analysis of collected data to online evaluation of a user's current situation? By including the context of a user in this evaluation, variables, which are not available offline, like current location, activity or social interaction, can be taken into account to better predict moments of interest. Audio and video recordings with a wearable could then be triggered specifically at those times, resulting in more "interest per recording".

One example that follows this approach is the eyeBlog system [13]. Eye contact sensing glasses report when people look at their wearer. The system will record video each time eye contact is established, resulting in automatically edited conversational video blogs.

To the best of knowledge, no work has been published that makes use of context classification with acceleration and audio sensors to automatically capture interesting moments of life.

1.4 On-body versus environmental sensing

Everyday activities can be broken down into two categories. Some are characterized primarily by the way the human body is being used for example, sitting, standing, walking, scrubbing, clapping hands and shaking hands. These activities are best recognized directly where they occur, using body-worn sensors. Examples for recent work using this approach are [6], [14], [15], [16], [17] and [18]. Other activities are clearly defined by the usage of certain objects or a sequence of objects for example grooming, cooking, and housekeeping. Instead of tracking body movement, sensors incorporated in the environment could track the movement of objects. This is done in [19], [20] and [21]. While this can perform extremely well under certain conditions, it is very costly and does not scale to settings outside these augmented environments. The focus of this work is on the former approach.

1.5 Sensor selection

There are a variety of sensing technologies that could be envisioned for context recognition for instance audio, video, photography, acceleration, light, air and body temperature, heat flux, humidity, pressure, heart rate, strain gauges, galvanic skin response, electrocardiograms or electromyograms. Streaming video is overly expensive on bandwidth and many physiology sensors require skin contact or special outfits. There is clearly a tradeoff between informative and unobtrusive sensing. Other points to consider are battery life and price.

In order to reach user acceptance a wearable must be small and unobtrusive, if possible even fashionable. There are several accessories that are nowadays widely accepted, such as belts, watches, necklaces, cell phones or pagers on a belt-clip. Miniaturization of hardware has made it possible to integrate sensors into such devices. While nobody would want to wear sensors strapped around all legs and arms, a belt and a wristband would be not more than what most people wear already.

This work focuses on a minimal set of sensors: audio plus two accelerometers. We believe that acceleration and audio sensing is sufficient to classify many interesting dimensions of context. This assumption is heavily supported by [18]. Several studies using physiology sensors were conducted in [3], the results of which show that most significant information was won from movement and audio. It has been shown that recognition rates for user activity only drop minimally when reducing the number of acceleration sensors to two, worn on hip and wrist [16].

1.6 Data collection and annotation

Activities are numerous and often not clearly defined. Already a simple activity like eating can be performed in several fashions by different subjects. Some use both hands holding forks and knives, some only use a fork in the right hand, and for eating a sandwich no utensils are used at all. Also, meals may be consumed at home, in a restaurant in the office or on the go, sometimes quietly without company other times in large groups including excited discussions. Furthermore, activities are not always mutually exclusive. Typing is usually done sitting. Brushing teeth could be done sitting, standing or walking. This shows that finding appropriate context labels and their right granularity is not an obvious task, especially if the system is to scale.
The "context structure" we propose consists of four categories: location, speech, posture and activity. These categories are concurrent because they describe different dimensions of the user's context. Within itself, each category comprises a number of mutually exclusive labels.

Another question is how to train classification models. Only few people suggest completely unsupervised learning methods (e.g. [22]). Supervised and semi-supervised learning requires a certain amount of labeled training data. In [6] and [17] an online labeling scheme was used, where the subject repeatedly selected an activity from a list on a PDA. The downside is evidently user distraction and impeding natural body movement. In [16] subjects were given a list of activities to perform and asked to note the starting and stopping times for each activity. No labeling was required for an experiment in [15], where the subject followed a strict protocol of subsequent activities. This does of course not scale to any general setting.
Our approach to data collection and annotation is interval-contingent experience sampling. During recordings of several hours, audio clips and an image are recorded once every minute. These are then presented to the subject offline with a selection of labels. A detailed description of this method follows in Chapter 3.3.

2 Wearable platform

The hardware platform used is based on low-cost sensors and leverages off commodity hardware. It consists of a PDA, two wireless accelerometers and the matching receiver. This provides the following sensing layout:

- Triaxial accelerometer on the left side of the hip (~90Hz, 10bit)
- Triaxial accelerometer worn on the wrist of the dominant hand (~90Hz, 10bit)
- Audio recorded from the wearer's chest (8kHz, 16bit)
- Images taken from the wearer's chest (1 per minute)
- WiFi access point sniffing with the PDA (every 100 seconds)

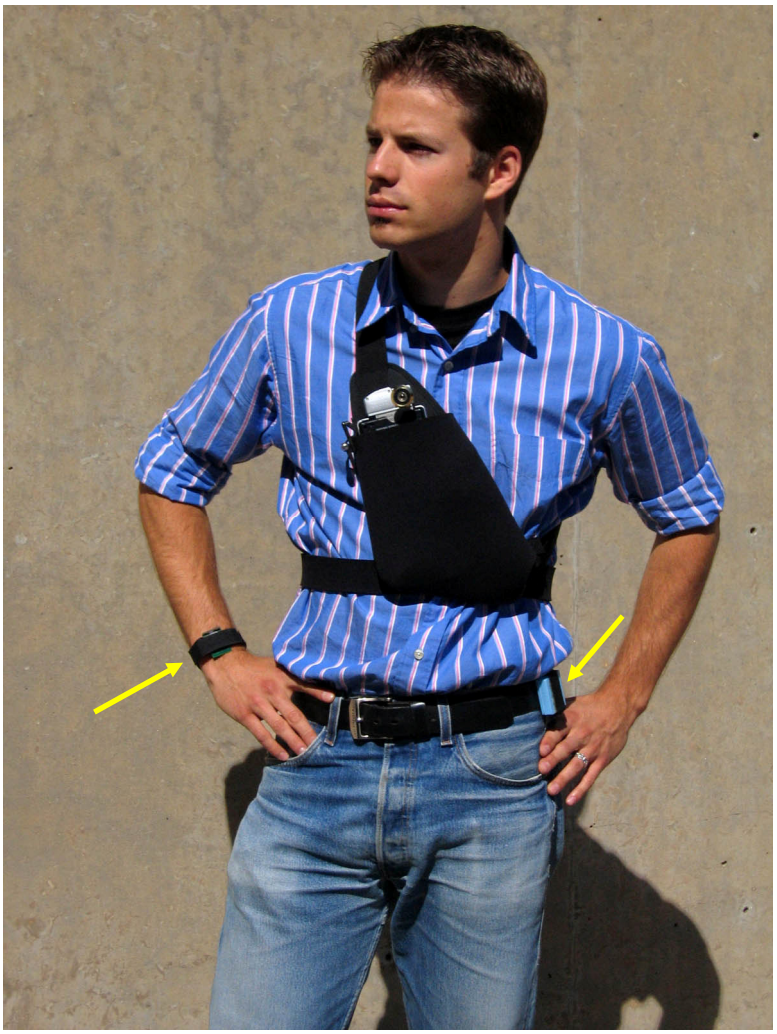


Figure 2-1: Sensor placement

2.1 The PDA

The Sharp Zaurus SL6000L is a very versatile and easy-to-use wearable. The PDA features a 400 MHz StrongARM processor, color touch screen, Audio I/O, Serial I/O, SD and CF card slots, built-in WiFi and runs the Linux operating system. SD cards of currently up to two gigabytes provide ample storage. The CF card enables a rich variety of peripherals to be attached, such as image or video cameras, Bluetooth wireless or even a head mounted display. The combination of touch screen and Qtopia allow nice and simple graphical user interfaces. There is also a vast user community, which becomes a valuable help while developing.

Sharp Zaurus SL6000L [23]
Linux + Qtopia operating system
StrongARM processor 400 MHz
64MB SDRAM
64MB Flash-ROM
4 inch transflective TFT 65k color touch screen
Resolution 480x640 Pixel
37 key QWERTY keyboard with slide cover
USB host interface
Serial interface
Infrared Interface IrDA (1.2)
built-in WiFi (IEEE 802.11b) WLAN interface
1 slot for SD (Smart Digital)
1 slot for CF (Compact Flash) Cards
Stereo Headset Jack
Shock-Resistant Casing
Lithium ion battery (Sharp EA-BL09 1500 mAh)
Dimensions: 156 x 80 x 21 mm
Weight: ca. 280g incl. battery

Table 2-1: Technical specifications of the PDA

Figure 2-2: Hardware setup with PDA, battery, microphone, camera and MITe-receiver

2.2 Accelerometers

The wireless accelerometers used in this project were developed by the House_n group at MIT. Their name *MITes* stands for MIT environmental sensors. They are designed around the nRF24E1 chip by Nordic VLSI Semiconductors, a 2.4GHz low power transceiver with built in microcontroller. In this project two triaxial MITes were used with the following features:

Sensors: ADXL202 (±2g)
Sampling-rate: 100Hz each
Quantization: 60 counts correspond to 1g
Dimension: 3.4 x 2.5 x 0.6cm (1.3 x 1.0 x 0.25in)
Weight: 8.1g (including battery)
Average battery life: 20.5h
Range: ~30m indoors and 220m outdoors

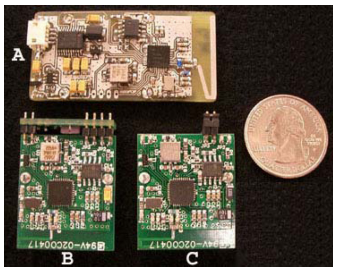


Figure 2-3: Accelerometer hardware
A) Receiver
B) 3D-MITe
C) 2D-MITe

The receiver board contains the same transceiver (nRF24E1), a RS232 level converter (MAX3218) and a voltage regulator (MAX8880) for external power supplies between +3.5 and +12v. The dimensions are 5 x 2.5 x 0.4cm (2 x 1 x 0.2in).
The receiver is designed to interface with a RS232 serial port, which works fine with a PC or laptop. The Zaurus PDA turned out to use a different serial standard, which required some alterations of the receiver board. The RS232 level converter was removed and the Rx/Tx pins were connected to the transceiver chip, with a logic inverter placed in between.
House_n designed a simple serial protocol for receiver configuration and data transfer, which was implemented in C++ and adapted to the software systems used on the Zaurus.
A detailed description of the hardware can be found in [24].

2.3 Audio and images

The Zaurus has a decent built-in microphone which can be sampled in mono at common rates from 8kHz to 44.1kHz. For this project the lowest rate of 8kHz was selected. Testing a variety of microphones showed that electret condenser microphones improve audio quality. Consequently after several recordings with the built in microphone the Sony ECM-T115 omnidirectional electret condenser microphone was used. A detailed comparison of the tested microphones can be found in Appendix F

The camera used is the Sharp camera Compact Flash card CE AG06, which features a 350’000 pixel CMOS imager. The driver software was partially available online.
Pictures can be captured in several resolutions. Although advertised by some vendors, we did not achieve the VGA resolution of 640x480 and settled for 480x480.
A downside of this camera is the bad wide-angle. Therefore, in order to capture as much information as possible we added a 180° door viewer.

Figure 2-4: Camera and image examples with and without fisheye lens

2.4 WiFi sniffing with Kismet

Kismet is a GPL open source software that uses the built in WiFi card to scan for available 802.11b networks. It is launched by the recording application in intervals of 100 seconds. This frequency was chosen to keep power consumption low. Network names and MAC addresses are extracted from a Kismet log-file and compared with entries in a look-up table to find location labels.

Apart from detecting WiFi networks, 802.11b can of course also be used to design multi-node, distributed wearable systems. Even high-bandwidth data such as full audio could be streamed to off-body resources, where available. This capability has not been used in this project, but is definitely a big potential for future applications.

2.5 Battery life

The internal battery of the Zaurus is not strong enough to power both the PDA and the MITes receiver. In order to increase runtime an external battery pack is used (Sony NP-

F550 7.2V). The 7.2 volts are regulated down to 5 volts, the expected input voltage of the Zaurus, using the LM7805C regulator. An inductive coil of 83uH was added in series to prevent spikes of high current that can cause the battery to lock in a short-circuit protection state.

With both the Zaurus and the external battery fully charged, the system operates for about 6 hours. This is in fact just the time it takes to fill the 1GB SD-card with data. The lifetime of the accelerometers is about 20 hours.

3 Data acquisition and annotation

In this chapter the process of recording and labeling data is presented. A detailed technical tutorial can be found in Appendix E

3.1 Signal recording application

A Qtopia application named SignalRecorder was developed to provide a simple graphical user interface for data acquisition. In a first step, a session name can be entered and sensors can be selected. By pressing the start-button SignalRecorder will take care of launching all sub-processes with the right parameters and start logging data onto the SD-card.

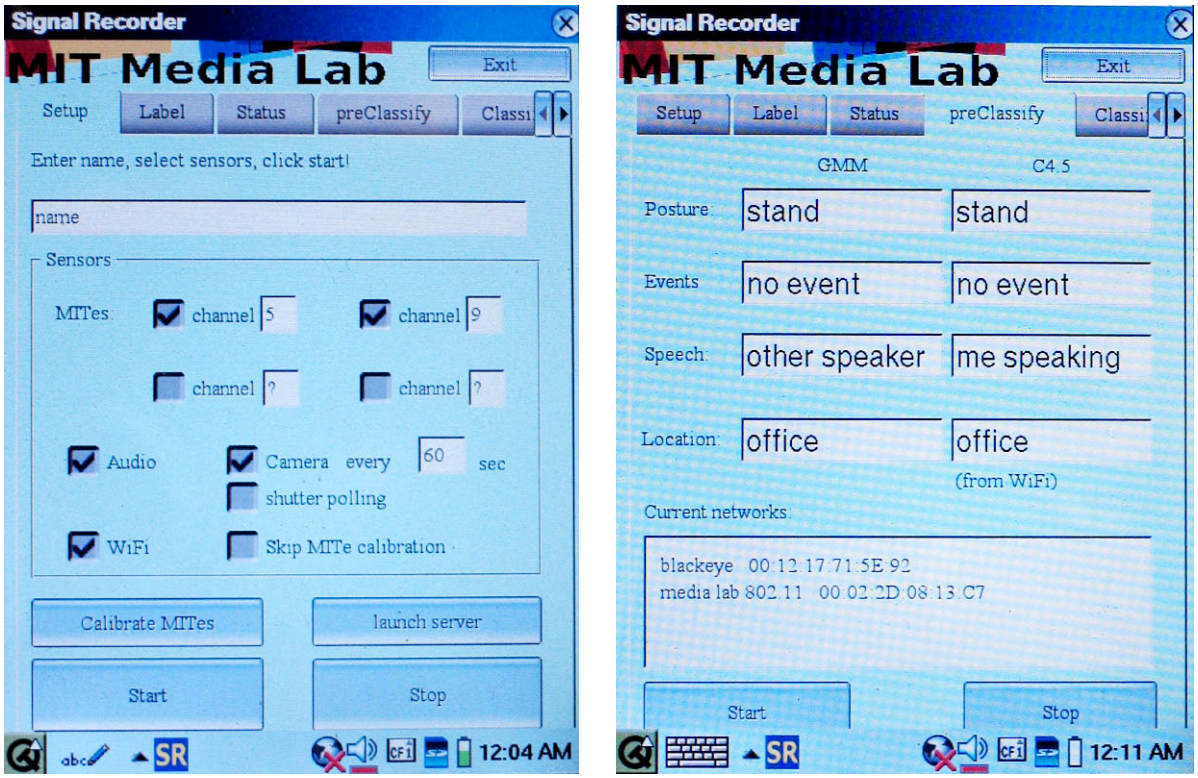


Figure 3-1: Zaurus screenshots. System setup (left) and classifier outputs (right)

At runtime, there is a possibility to give online labels by selecting activities from a list of radio buttons. Although implemented, this method was hardly used in this work.
During the whole recording session battery power and CPU load as well as the log-files are monitored periodically. Their sizes are displayed in a status window, and if in two consecutive readings any file does not increase in size, a warning is displayed. This was

considered very important in order to prevent the annoyance of wasting hours of recording time because of software crashes or malfunctioning sensors.
As classifiers were implemented, another window was added to the application to control the classifiers and display classification results in real-time.

3.2 Accelerometer calibration

The accelerometers used in this project need calibration. A program named MITeCalibrate was developed to measure the mean and the scale of acceleration values over all three axes. This is done by determining the values corresponding to +/- 1g for each axis, then translating the mean to zero and dividing by the difference equivalent to 1g. To find the 1g-values each accelerometer has to be rotated in each of its six extreme orientations and held still for a few seconds. Two rolling averages are computed per axis. One can only increase, the other decrease and the learning factor is indirectly proportional to the variance over a one-second window. Also the values are median filtered, which further reduces the risk of distorting the calibration by shaking.

3.3 Interval-contingent experience sampling

Subjects wear the system for several hours without interacting with it. Audio and acceleration signals are recorded continuously. The camera takes pictures once a minute and periodically WiFi networks are logged. After the recording session an offline annotation tool is used, which presents at a time an image, the corresponding sound clip and a list of labels to choose from. This naturalistic approach reflects best the every day life of a user and provides apart from the annotated data also statistics on the subject's activities. Obvious shortcomings are of course the one-minute granularity. A purely naturalistic protocol will not capture sufficient samples of certain activities like shaking or clapping hands. For these short activities, semi-naturalistic training is necessary.

3.4 Offline annotation tool

The annotation tool developed for this project is web-based and uses PHP and a MySQL database to manage annotation sessions. In a preprocessing step, for each image a corresponding audio clip is extracted from the audio log-file and saved as a wav-file. The clips are 20 seconds long and start at the time the image was taken. A list of image and audio pairs is then imported into a SQL table and associated with a session name. It is important to mention, that only the filenames are stored on the web server. The images and audio files are at all times located on the subject's hard drive or local network.

The labeling process is then as follows. For each clip, the best matching label in each category must be selected. If a clip is ambiguous, or the user is not sure, the "confident" checkbox is to remain unchecked. This is important for the quality of the labeled data. In case of a lot of training data, a few miss-annotations will not cause much harm. However, if the data is used for testing, the accuracy of the classifier will depend directly on the label the user decided on. Optionally a line of comment can be added.
When a label is inserted, the next clip will load and the selected labels for the previous clip are repeated as a suggestion. This speeds up the annotation significantly. During a long bike ride or in front of the computer, most clips are identical and can be clicked through rapidly. It is more difficult when many things happen, especially when speech is included. Then the whole 20 seconds need to be listened to in order do make sure the given labels are valid for the whole clip. In discussions it is very often not the case that only the user or somebody else is speaking throughout the clip. For those cases a label "Speech, but none of the above" was introduced so that the information on speaking/non-speaking can still be captured.

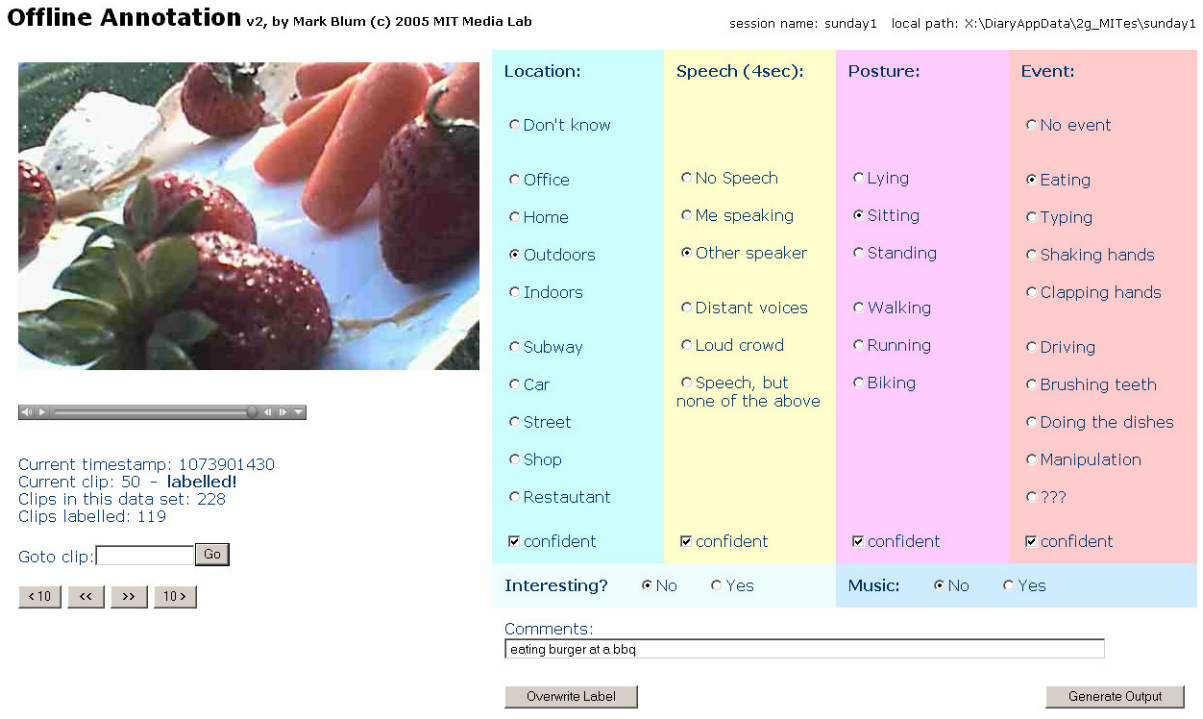


Figure 3-2: Sreenshot of annotation tool

The labels are stored in the database and can be retrieved or changed any time. Buttons are provided to navigate through the clips. After labeling all the clips, three output files are generated: One with the timestamps and the corresponding labels, a second with the comments, and a legend that maps numeric values into the labels as they were presented on the screen.

The time it takes to label data depends very much on its variety. In average it takes about one quarter of the recording time to label a dataset.

3.5 The data collected

As this wearable platform was developed from scratch, there were repeated changes in hardware and software that made the data collection an iterative process. At first we only had 10g accelerometers. In this configuration ten datasets of at least one hour were collected and used to train the first set of classifiers. Analysis of the data showed that acceleration levels were predominantly in the +/- 2g range, which suggested that through higher quantization of 2g accelerometers better quality data could be recorded.
A total of about 35 hours of raw data in 14 sessions was collected with 2g accelerometers. Two datasets were unusable because of wrong sensor placement and one because of hardware failures. The base for this work is about 24 hours of data from 11 sessions with myself as the subject. The dataset includes data from several locations recorded at different times of the day and reflects roughly the every-day life of a student. Unfortunately, due to a malfunctioning adapter plug, in 8 of those sessions the built-in microphone of the Zaurus was used instead of the Sony condenser microphone.

3.6 Enhanced sensor configuration

The decision was taken to include yet a different sensor package for future recordings. The new configuration includes accelerometers on hip, wrist and head, as well as audio from the chest and the wrist. This is to be the setup for the major data collection phase. So far 6 sessions have been recorded by 5 different subjects. Unfortunately, it is out of the scope of this thesis to elaborate on this new data.

	duration	instances
Location		
office	185mins	20
home	126.5mins	12
outdoors	16mins	19
indoors	92mins	8
restaurant	34mins	8
car	40mins	4
street	51mins	16
shop	20mins	4
Speech		
no speech	136mins	N/A
user speaking	31mins	N/A
other speaker	91mins	N/A
distant voices	20mins	N/A
loud crowd	12mins	N/A
laughter	8mins	1***
Posture		
unknown	9mins	1**
lying	13mins	1
sitting	886mins	54
standing	191mins	52
walking	30mins	21
running	4mins	1
biking	79mins	19
Activities		
no activity	1000mins	72
eating	92mins	10
typing	261mins	33
shaking hands	8mins	1*
clapping hands	6mins	1*
driving	33mins	4
brushing teeth	8mins	2
doing the dishes	12mins	2

Table 3-1: Overview of the data acquired

* Shaking and clapping hands is difficult to capture using the naturalistic approach. The training data in these cases were recorded at the lab.

** There are basically no occasions in everyday life for which posture would be anything other than the suggested labels. Nevertheless it is desirable to classify an *unknown* posture if sensor values clearly suggest so e.g. when the sensors are worn the wrong way. To emulate an unknown posture, random but realistic signals were generated.

*** Again, laughter is hard to capture. To acquire realistic laughing data, we edited several recordings carried out in and around the lab, in which we made use of jokes and funny videos to provoke laugher.

Note: the labels indoors and outdoors must be understood as in- or outdoors, but none of the other labels.

4 Classification architecture and implementation

4.1 Classification architecture

One goal of this work is to propose an architecture for comprehensive context recognition.
The four categories location, speech, posture and activities were chosen to reflect different dimensions of context, that all are well defined at any instance of time. The labels within the categories are mutually exclusive.

Location	Speech	Posture	Activities
office	no speech	unknown	no activity
home	user speaking	lying	eating
outdoors	other speaker	sitting	typing
indoors	distant voices	standing	shaking hands
restaurant	loud crowd	walking	clapping hands
car	laughter	running	driving
street		biking	brushing teeth
shop			doing the dishes

Table 4-1: The four classification categories with labels

The work in this thesis is in a very exploratory stage and it is still unclear what can be classified with the given sensing layout. The labels were chosen to reflect everyday situations, we believe are well classifiable.
This choice naturally has shortcomings. It is for instance hard to compare the speech label laughter with loud crowd, since both can actually be true at the same time. Also the temporal resolution poses problems. For a good recognition of eating, activity patterns over about 30 seconds are most meaningful. But then an activity like shaking hands might last only 3 seconds. These were lessons learnt and suggestions to improve this architecture are given in Chapter 8.

We decided to rely on acceleration for the categories posture and activities and on audio for the other two. In a pre-classification step four separate classifiers make a decision in their category. Then, in a post-classification step, a common sense model is proposed that combines the information from all four categories to output a final classification. This architecture is the base for the evaluation in the upcoming chapters and was implemented on the wearable in real-time.

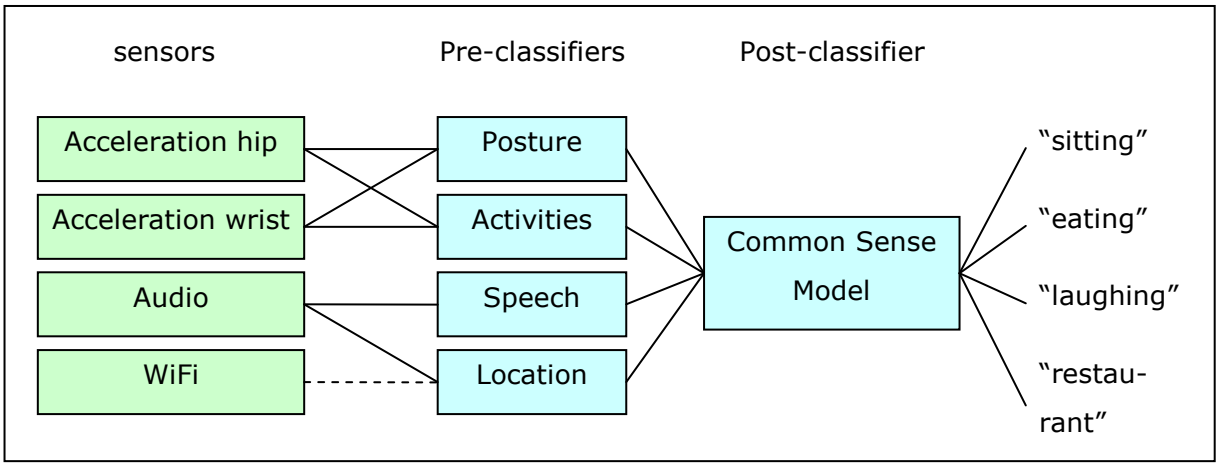


Figure 4-1: Classification architecture

Note: WiFi location classification is output on the wearable, but not used for any evaluation in this thesis.

4.2 Implementation

The implementation was done in C and C++ and is based on the MIThril 2003 software architecture, developed by the group. It is layered on top of standard Linux libraries and can be compiled for several processor types. The two main components used are described below, for more details see [25].

The Enchantment Whiteboard is a distributed client/server inter-preprocess communication system that provides a lightweight, modular framework for applications to communicate. Client processes can post information in the form of nodes on a whiteboard. All nodes have a unique node locator (UNL), which is in the form of `server-IP:/path`. Other clients can either poll nodes on a server, or subscribe to a whole set of postings to receive updates automatically.
For higher bandwidth signals, like the acceleration and audio data, the Enchantment Signal system offers point-to-point communications between clients, with signal "handles" being posted on the Whiteboard to facilitate discovery and connection. The Signal API abstracts away the need for signal producers to know how many, or even if, there are any connected signal consumers and who they are.
All these communication schemes are IP-based and thus network transparent.

The application that provides a graphical user interface and manages the signal reading and classifying process is SignalRecorder (see also Chapter 3 on data acquisition). This Qtopia application is merely the interface to sub-processes that do the actual classifica-

tion. It invokes processes for sensor reading, feature generation and classification using system calls and takes care of correctly terminating them when done. The whole classification procedure could be run from the command line.

The following figure depicts the software structure, showing processes and the flow of data though them:

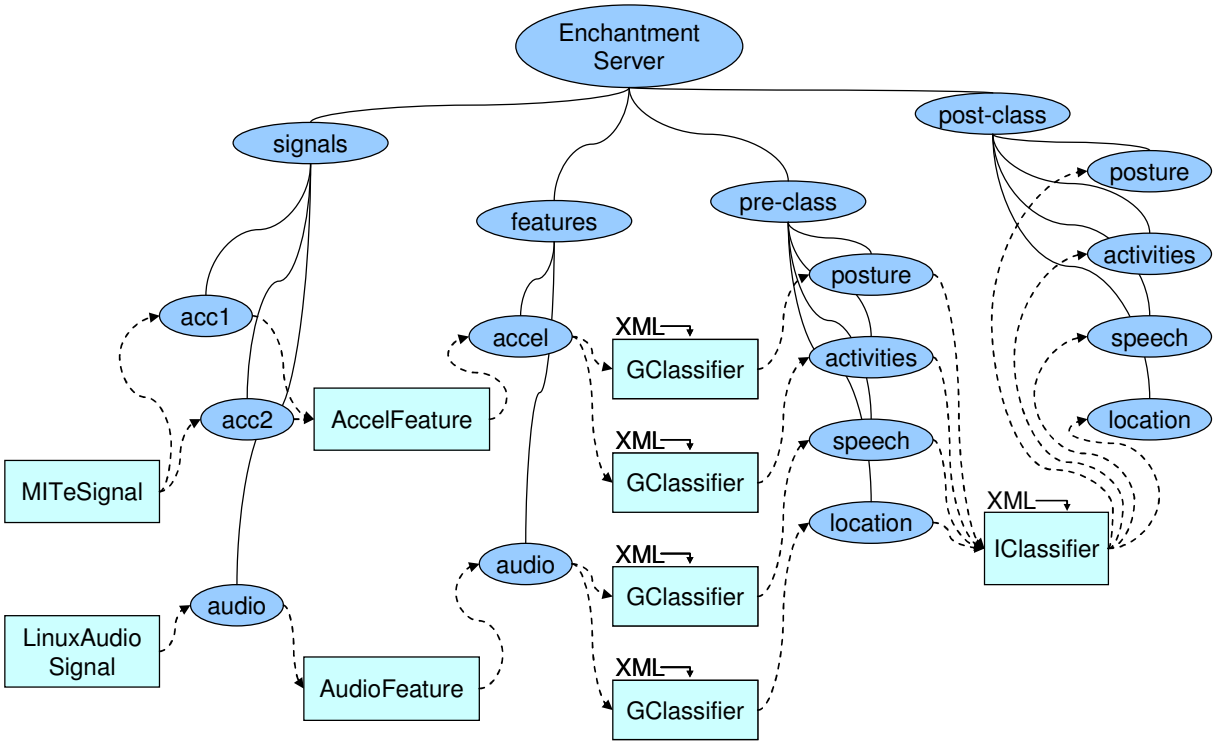


Figure 4-2: Software structure of real-time implementation

The light blue boxes represent single processes, the dark blue ovals nodes on the Enchantment whiteboard. The dashed lines symbolize the logical flow of data. While each process reads and posts signal handles on the server, the actual signals flow point-to-point.

In order to facilitate saving, loading and visualization of models, an XML-based interface was developed. Model parameters that are learned in Matlab are exported to file in a standard XML format, which can be loaded by the real-time classifiers.

A detailed description of the C-code can be found in Appendix C.

5 Pre-classification: focus on each category

This chapter explains the classification methods used for each category and gives a detailed evaluation of their performance.

5.1 Classification methods

In view of the selected architecture and considering that the system is to be run in real-time, the focus of the pre-classifiers was not on reaching highest possible accuracy for each category, but rather on speed and flexibility. Because of limited program memory, approaches that rely on saved test-data (like k-nearest-neighbors or histogram based Bayes classifiers) are not suitable for real-time. The classification methods studied were

- Naive Bayes classifier using Gaussian probability distributions
- Naive Bayes classifier using mixtures of Gaussians
- C4.5 decision trees
- Hidden Markov models

In all cases, classification was based on windowed features computed from the raw sensor data. Several features were tested. They are explained in the corresponding chapters further down. Linear discriminant analysis (LDA) was used to visualize the feature space. Classification methods using the projected features were tested but were found inferior.

For each category, features were calculated over the entirety of the data. These feature vectors were then split into training and testing sets of equal size by choosing every other vector for training and testing, respectively. This proceeding needs to be explained.

It can be reasoned that training and testing data should be from completely distinct datasets, in order to prove a classifiers generalizing capability. The reason we proceeded differently is the following. Data that is recorded and labeled using the experience sampling method is very diverse. There are for example 5 recordings of eating. In two of them the subject used knives and forks in both hands, in two of them only a fork in the right hand and one was eating a sandwich. Clearly, any classifier would perform badly, if it was trained exclusively on one way of performing an activity and tested on the other. In other words, the activities we are trying to classify are inherently so complex that the training data needs to represent the whole diversity of the data. This becomes less an issue if there are sufficient recordings in all variations of activities. For our work this was not the

case, rather more, the sessions were specifically recorded in different settings. This said, we believe interleaving samples is a valid approach for testing and comparing different classifiers. To add credibility some cross-validation data was recorded, which is evaluated at the end of this chapter.

The Bayes classifier is very simple, fast and intuitive and in the case of a single Gaussian PDF nearly immune against overfitting. Hence it serves as a good reference for other approaches. The performance of the Bayes classifier did not improve much using mixtures of Gaussians, except in the category activities. We thus mainly present the results for single Gaussian modeling.

The c4.5 decision trees seem to perform better in overall accuracy. However there is not much gain in average class accuracy. The algorithm used is implemented in C and originates from University of Regina, Canada [26]. The decision trees that are generated are very big. The tree learned on posture (17990 cases) contained 631 nodes before pruning, 541 nodes after. This gave reason to believe that the tree is overfitting the data. By modifying the source code, the pruning effort was increased. The resulting tree for posture is now down to 127 nodes with only minor losses in accuracy. It seems that classes with only few samples tend to suffer, compared with classes containing numerous examples. This hints that the pruning algorithm is unfair, or better said that it tries too much to maximize the overall accuracy. Until this classifier can be tested against more distinct data, it is unclear if the decision tree is still overfitting.

Hidden Markov models (HMMs) were tested on the category activities only. They are discussed further down.

5.2 Posture

Posture classification is based on acceleration only. The features used are means and variances of X, Y and Z axis of both accelerometers, resulting in a 12-dimensional feature vector. Other features that were tested but not selected are

- Sum of differences
- FFTs with and without the DC component over several singles axis and the magnitude, with and without Hanning windows.
- Eigenvalues
- Principle components of the covariance

All these features were evaluated using a Bayesian Classifier with a Gaussian PDF. The performance of the "complicated" features was remarkably poor compared to the simple

ones selected. Window sizes were varied over 128 samples (~1.4 seconds), 256, 384 to 512 samples (~5.7 seconds). Larger windows in general yielded better results, but also mean an increase in latency. A window size of 384 samples (~4.4 seconds) was selected.

Already the visualization of the feature space using linear discriminant analysis shows that most posture labels can be expected to be easily classified.

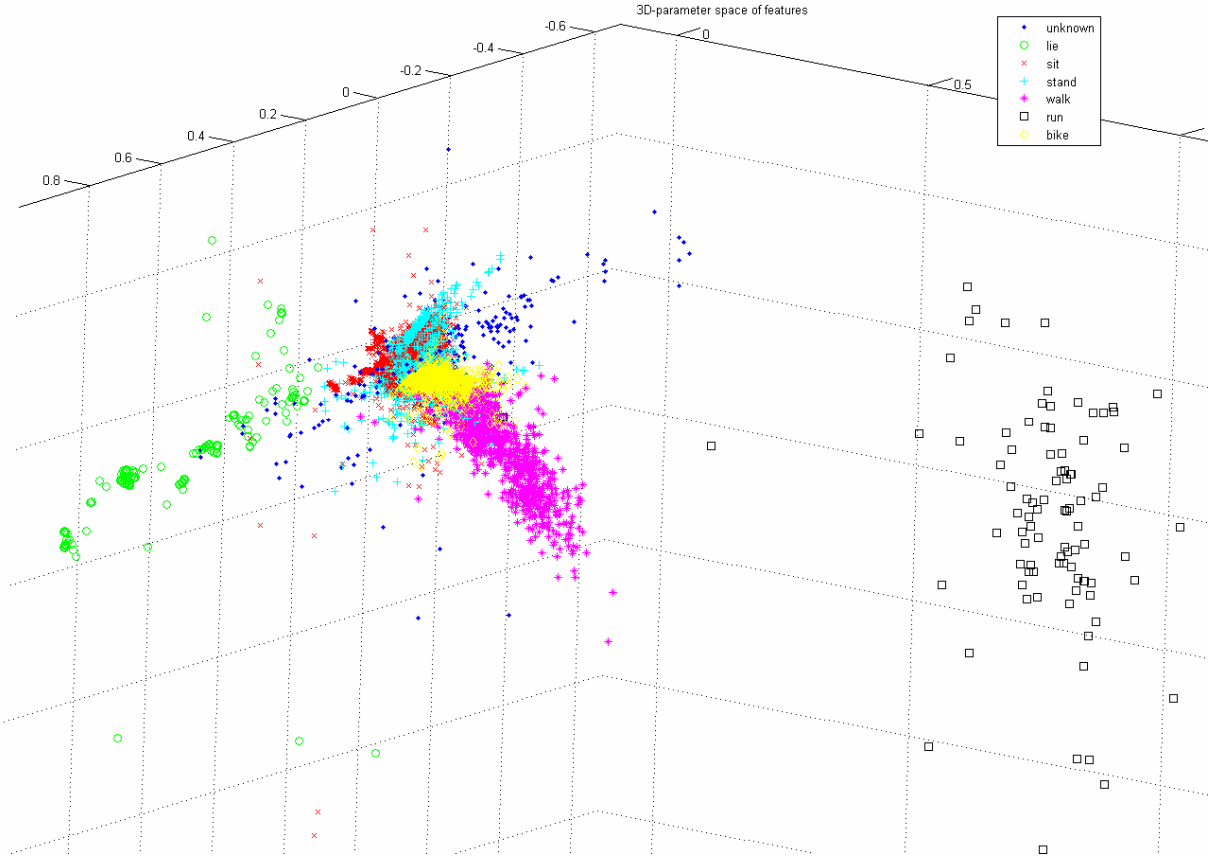


Figure 5-1: Feature space for posture

The resulting confusion matrix for the Bayes classifier is shown in the following table:

classified as -->	a	b	c	d	e	f	g	accuracy
a = unknown	53	1	5	2	0	0	0	87%
b = lying	1	89	2	0	0	0	0	97%
c = sitting	22	3	6241	174	2	0	27	96%
d = standing	8	0	304	924	43	1	100	67%
e = walking	0	0	6	16	182	0	6	87%
f = running	0	0	0	0	1	22	0	96%
g = biking	0	0	6	17	2	0	547	96%
							class average:	89.3%
							overall accuracy:	91.5%

Table 5-1: Posture confusion matrix, Bayes classifier

Six of the seven labels have an accuracy over 85%, which is very pleasing. Only the class standing shows major confusions, namely with sitting, walking and biking. We see two main reasons for this.

Firstly, it has to be said that the accuracy of the labels themselves is not 100%. The off-line annotation introduces labeling errors. The clips for posture (as also for events) were aggregated across several minutes. This means that if two consecutive clips have the same posture label, then all the data between the clips is labeled the same. This will of course mean that if somebody is basically standing, but quickly walks across the room to pick up something, this walking data can falsely end up with a standing label. The same thing is likely to happen for short pauses while walking.

This will however not account for all the standing/sitting and standing/biking confusions. For these cases we will have to bear with the fact that the positioning of the sensors does not allow accurate classification. Without information on the orientation of the thighs, it is hard to tell apart standing and sitting. Analysis of the classifications shows that sitting is actually best differentiated from standing by the orientation of the wrist. Sitting is assumed as soon as the hands are placed on a table, as in typing and eating. So if the subject is standing and holding something in his hand, a misclassification is likely. There is only little sitting/standing confusion though. This is thanks to a small but noticeable difference in angle of the hip accelerometer. If the hip is vertical, as in standing, the user often has his hand in a typing or eating position. In the case of sitting doing nothing, as on a couch for example, there is a much bigger tendency to lean back, which again prohibits classification as standing.

Here is the confusion matrix of the c4.5 decision tree. The overall accuracy is higher, but the same tendencies can be noticed as before, except for the unknown class, which is classified badly. This is not surprising though because it is randomly generated and there is no pattern to model, other than the fact that the samples don't fit any of the other categories. This can seemingly be better captured with the Bayesian approach.

classified as -->	a	b	c	d	e	f	g	accuracy
a = unknown	30	2	15	13	1	0	0	49%
b = lying	0	85	4	0	1	1	0	93%
c = sitting	0	0	6329	129	3	0	8	98%
d = standing	0	1	174	1187	9	0	7	86%
e = walking	0	0	5	26	174	1	1	84%
f = running	0	0	0	0	1	21	0	95%
g = biking	0	0	11	22	3	0	535	94%
							class average:	85.7%
							overall accuracy:	95.0%

Table 5-2: Posture confusion matrix, C4.5 classifier

5.3 Activities

As posture, activities are classified using acceleration only. This is of course a simplification, considered that we also have full audio. We do indeed believe that several activities would profit from including audio features in the classification process, especially typing, driving or doing the dishes. This will be part of future work. With the current approach, audio can still affect the final classification in the sense that the post-classifier will for example only output driving if the current location is classified as car.

The same acceleration features were tested as described previously. Findings were similar, no set of features performed significantly better than the means and variances. A difference though was perceived in the results with different window sizes. Larger windows seem to capture more of the dynamics, which are especially slow in the case of eating.
For ease of design, we decided to go with the same window size as the posture classifier. That way the features must only be computed once.
The following figure shows the feature space after LDA transformation:

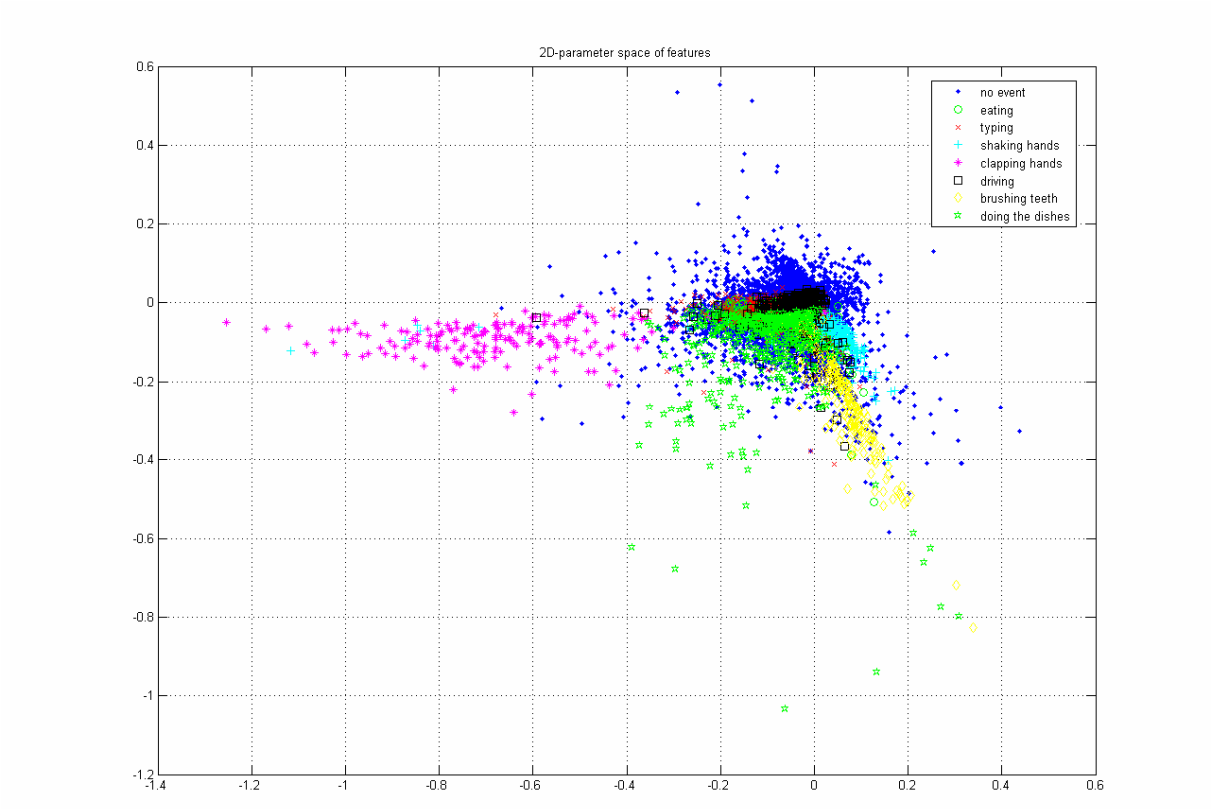


Figure 5-2: Feature space for activities

This plot is a reduction of the 12-dimensional into a two dimensional space and thus can only give hints on separability of the features. Nevertheless, very distinct movements such as clapping hands or brushing teeth can be well distinguished. Activities with small

variance values cluster in one cloud. Driving and shaking hands are compact clouds within, which should be fine, but eating and typing are pretty scattered. The samples of doing the dishes that are well outside the cloud are very likely the ones with scrubbing or where silverware was shaken to get the water off. Since "no event" comprises all kinds of activities other than the ones with other labels, it is not surprising that the cloud is big and surrounds most others (this plot does not quite show this because the blue dots were plotted first).

The following two tables show the confusion matrix of the Bayes classifier, first with one, then with a mixture of three Gaussian PDFs.

classified as -->	a	b	c	d	e	f	g	h	accuracy
a = no activity	4494	864	1578	11	2	306	21	24	62%
b = eating	28	486	145	0	0	10	1	1	72%
c = typing	95	56	1731	0	0	17	1	0	91%
d = shaking hands	8	0	0	47	2	0	0	1	81%
e = clapping hands	1	1	0	1	42	0	0	0	93%
f = driving	13	1	13	0	0	218	0	0	89%
g = brushing teeth	7	3	1	0	0	0	44	0	80%
h = doing dishes	36	0	2	3	0	0	1	44	51%
								class average:	77.5%
								overall accuracy:	68.6%

Table 5-3: Activities confusion matrix, Bayes classifier with single Gaussian

classified as -->	a	b	c	d	e	f	g	h	accuracy
a = no activity	5585	497	1005	5	1	173	11	23	77%
b = eating	84	490	95	0	0	0	0	1	73%
c = typing	177	46	1676	0	0	1	0	0	88%
d = shaking hands	8	0	0	48	1	0	0	1	83%
e = clapping hands	1	1	0	2	41	0	0	0	91%
f = driving	41	1	4	0	0	198	0	0	81%
g = brushing teeth	5	2	0	0	0	0	48	0	87%
h = doing dishes	43	0	2	0	0	0	0	41	48%
								class average:	78.5%
								overall accuracy:	78.5%

Table 5-4: Activities confusion matrix, Bayes classifier with mixture of three Gaussians

As expected, activities with distinct movements classify very well. Five of the eight activities are above 80% using a single Gaussian. However, there is much confusion with the garbage class "no activity", especially many false positives. Every third sample of that class is confused with typing. And many are confused with eating and driving. This shows that it is difficult to distinguish sitting actions where there is little hand movement involved.

If we recall that the window size used is about 4.4 seconds, it becomes understandable that many eating samples are confused. Eating is an activity that spans several minutes

and its characteristic movements often occur at a rate too low to be adequately captured by the window used. Viewed on a higher level, it should not be a problem to detect an entire meal.
The same can be said for doing the dishes. It is also likely in this case, that the small amount of data did not suffice to capture the complexity of this activity.

The mixture model is about 10% better in overall accuracy. This is solely because of a big improvement of the "no activity" class. There are less false positives, but apart from brushing teeth, all other classes suffered. This might possibly also be achieved simply by giving the "no event" class a higher prior probability. The assumption that a mixture of Gaussians would better model the eating class, because of its presumably dissimilar subsets of motions, was not confirmed. This is partly due to overfitting, for there is a considerable difference between training and testing error for the mixture model.

A very good job is done by the c4.5 decision tree:

classified as -->	a	b	c	d	e	f	g	h	accuracy
a = no activity	6972	105	141	11	3	40	5	19	96%
b = eating	112	542	14	0	0	0	0	0	81%
c = typing	252	23	1624	0	0	1	0	0	85%
d = shaking hands	20	0	0	32	1	0	0	3	57%
e = clapping hands	5	1	0	0	38	0	0	0	86%
f = driving	61	0	0	0	0	183	0	0	75%
g = brushing teeth	16	0	1	0	0	0	38	0	69%
h = doing dishes	33	0	1	0	0	0	0	51	60%
								class average:	76.2%
								overall accuracy:	91.6%

Table 5-5: Activities confusion matrix, C4.5 classifier

The tree seems to be able to model well eating and "no activity", the challenging classes for the Bayes classifier. Except doing dishes, all other classes perform worse, especially shaking hands, which is quite surprising. As mentioned above, this might be due to an unfair pruning algorithm.

The typical temporal patterns and characteristic of movements in many of these activities suggest modeling with hidden Markov models. Each class was modeled with one ergodic HMM containing 2, 3 or 4 latent states with continuous observations modeled by Gaussian distributions. The data was split into classification chunks of about the same length as above (~4.4 seconds). Each chunk was split into an equal number of slices. For each slice, the same features were calculated as above. The configurations tested were

- 10 samples per slice, 40 slices per chunk; 2, 3 and 4 latent states
- 32 samples per slice, 12 slices per chunk; 2, 3 and 4 latent states

The HMMs were trained with the Baum-Welch EM-learning algorithm using the same training data as before. For classification, the class with the highest log-likelihood value for the forward algorithm is used. Because some HMMs model certain activities better than others, a normalization step is necessary. Using a simple, iterative algorithm, log-likelihood offsets were learned that are to be added after applying the forward algorithm. This can be compared to assigning prior probabilities.
Both configurations tested performed about equally well. In the first configuration, which used shorter slices, the performance increased slightly with the number of states used. In the second configuration there was no significant difference. We thus chose to present the results for 2-state HMMs using 12 slices of 32 samples for each classification chunk:

classified as -->	a	b	c	d	e	f	g	h	accuracy
a = no activity	5002	585	736	5	3	185	66	222	74%
b = eating	68	523	29	0	0	2	0	1	84%
c = typing	179	65	1504	0	0	15	5	1	85%
d = shaking hands	1	0	0	46	2	0	0	4	87%
e = clapping hands	1	1	0	0	39	0	0	0	95%
f = driving	43	0	0	0	0	184	0	0	81%
g = brushing teeth	1	2	0	0	0	0	46	2	90%
h = doing dishes	9	0	1	0	0	0	0	69	87%
								class average:	85.4%
								overall accuracy:	76.8%

Table 5-6: Activities confusion matrix, ergodic 2-state HMMs

While HMMs are prone to overfitting, analysis of the learned models showed that in many cases, the states represent fairly well the different subsets within activities. For example, the two states in shaking hands typically correspond to the hand hanging down and the hand extended and shaking, respectively. We thus believe two-state HMMs to be a good approach for this category. The downside, however, are the more expensive calculations required.

5.4 Speech

Speech classification is based on a set of voice features previously assessed by our group, namely:

- Formant frequency
- Spectral entropy
- Value of maximum autocorrelation peak
- Number of autocorrelation peaks
- Energy

The properties of these basic features are discussed in [27]. They are calculated over 256-sample windows with 128 samples overlap, which corresponds to a feature-rate 62.5Hz. For this work, the means and variances of these features were calculated over a window of 4.8 seconds, resulting in a 10-dimentional feature vector.

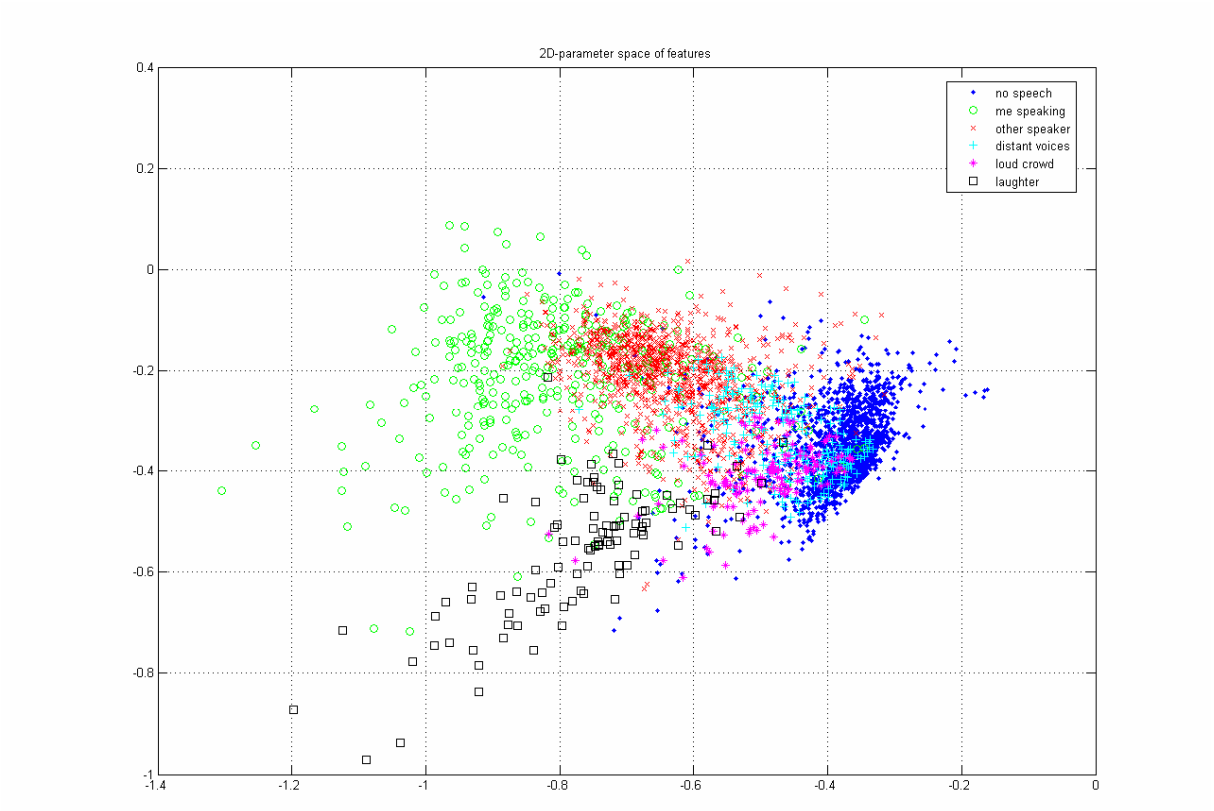


Figure 5-3: Feature space for speech

classified as -->	a	b	c	d	e	f	accuracy
a = no speech	785	4	21	4	8	3	95%
b = user speaking	7	104	65	0	9	2	56%
c = other speaker	26	6	493	10	21	0	89%
d = distant voices	76	0	41	6	2	0	5%
e = loud crowd	16	1	6	1	46	2	64%
f = laughter	3	4	6	0	3	37	70%
						class average:	63.0%
						overall accuracy:	80.9%

Table 5-7: Speech confusion matrix, Bayes classifier

The overall classification accuracy of 80.9% is descent, but mainly driven by the good classification of "no speech". The distinction of "user speaking" and "other speaker" only works in one direction. "Other speaker" seems to be the stronger class. It is understandable, though, that distinguishing speakers using ambient audio is a challenging task. The fact that "loud crowd" is mainly confused with "no speech" and not the other speech classes suggests that other features should be tested than the ones used, which are de-

signed to capture single voices. The complete misclassification of "distant voices" can be attributed to the unclear definition of the label and its inconsistent usage.

If the labels b to f are taken together as one class speech, we obtain an overall accuracy in speech recognition of 90.8%:

classified as -->	a	b	accuracy
a = no speech	785	39	95%
b = speech	127	858	87%
		overall accuracy:	90.8%

Table 5-8: Speaking / non-speaking with Bayes classifier

For this category there is no improvement with the c4.5 classifier. The results are very similar:

classified as -->	a	b	c	d	e	f	accuracy
a = no speech	769	5	21	12	16	0	93%
b = user speaking	15	105	59	1	2	4	56%
c = other speaker	34	26	466	7	20	1	84%
d = distant voices	54	1	37	26	5	1	21%
e = loud crowd	12	1	9	2	45	1	64%
f = laughter	1	1	2	1	8	38	75%
						class average:	65.6%
						overall accuracy:	80.1%

Table 5-9: Speech confusion matrix, C4.5 classifier

It must be said, again, that the accuracy values in the category speech do are not based on 100% correct labels. It is quite often the case that a 20-second clip will contain for example 15 seconds of user speech and some seconds of other speaker. Broken down to the 4.8-second window used, it is likely that several cases are mislabeled.

5.5 Location

We acknowledge the fact that location detection is an important part of context recognition. It is however not a focal point of this work for the following reason: For most cases, location can be easily and fairly precisely calculated using GPS outdoors and WiFi network detection indoors. Several publications in that field can be found in [28].
However, this does not directly apply to abstract labels like car, street or restaurant. We make an attempt in using audio to classify the eight selected labels, fully aware of the fact that it will be impossible to distinguish several of them.
Especially for home and office as well as several restaurants we can already us WiFi and a simple look-up table. This is implemented, but has not been taken into account for pre- nor post-classification approaches presented here.

For reasons of simplicity the same audio features and window sizes were used as for speech classification. The feature space is depicted below:

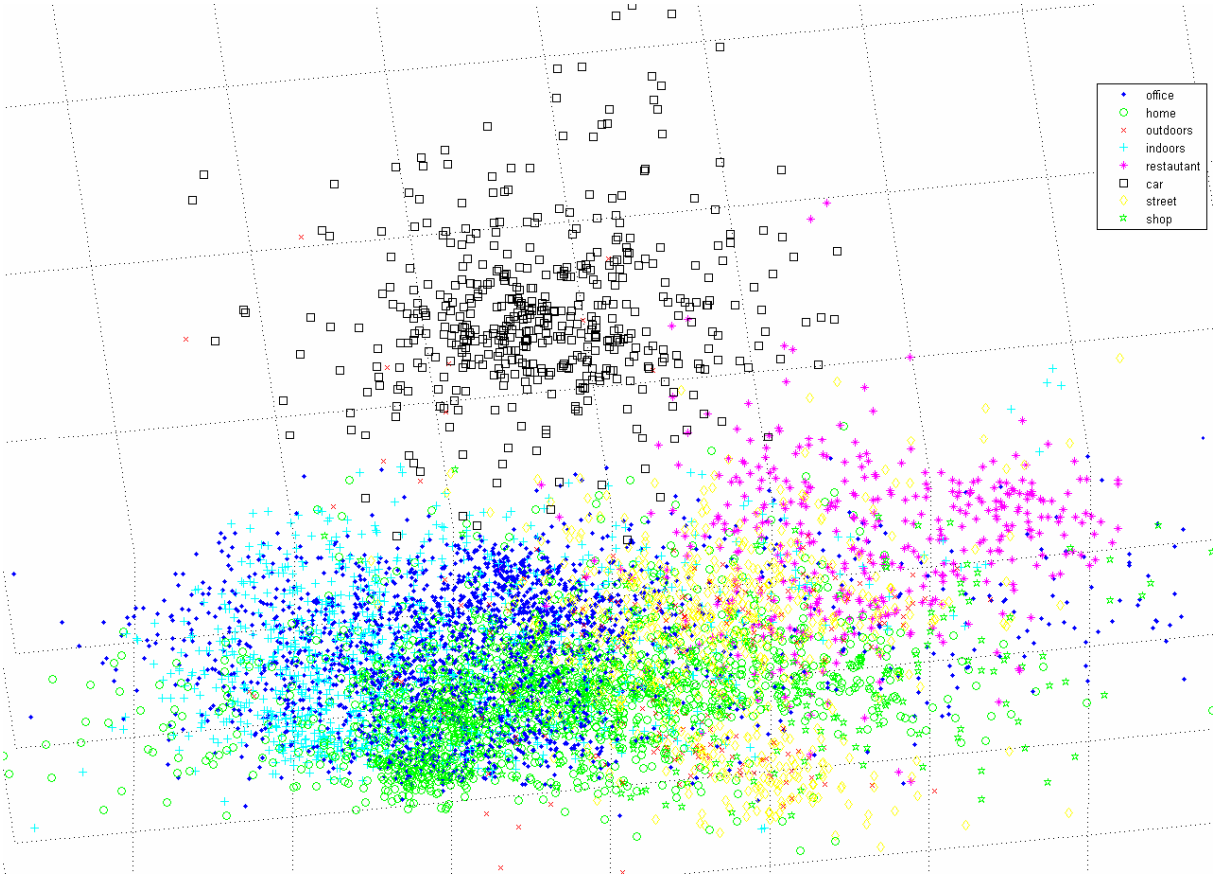


Figure 5-4: Feature space for location

All clouds are strongly scattered and most of them are intermingled. Only the label car can be completely separated.

classified as -->	a	b	c	d	e	f	g	h	accuracy
a = office	589	232	2	228	52	3	13	3	52%
b = home	89	563	3	43	57	1	12	2	73%
c = outdoors	33	10	9	0	3	8	35	0	9%
d = indoors	126	64	2	332	16	2	15	1	59%
e = restaurant	9	12	0	6	176	0	3	2	85%
f = car	1	0	0	0	0	240	3	0	98%
g = street	33	41	1	5	23	8	201	1	64%
h = shop	32	20	0	5	25	1	27	16	13%
								class average:	56.8%
								overall accuracy:	61.8%

Table 5-10: Location confusion matrix, Bayes classifier

As expected, the results in general are pretty poor, with an overall accuracy of 61.8% for the Bayes classifier. The classes outdoors and shop fail completely because the are mostly confused with office and home, the two classes with the most samples. Only car is

classified excellently. Restaurant is surprisingly good with 85%, street is somewhat disappointing with only 64%.
The c4.5 classifier performs again a bit better, but the comparison provides no new major insights.

classified as -->	a	b	c	d	e	f	g	h	accuracy
a = office	826	157	6	115	11	0	3	1	74%
b = home	187	519	10	36	4	1	9	2	68%
c = outdoors	7	20	24	1	4	5	31	4	25%
d = indoors	180	56	3	272	27	0	17	1	49%
e = restaurant	5	10	1	12	158	1	11	8	77%
f = car	0	1	3	0	0	238	2	0	98%
g = street	7	48	4	4	16	2	214	17	69%
h = shop	29	37	1	2	3	0	9	43	35%
								class average:	61.6%
								overall accuracy:	67.0%

Table 5-11: Location confusion matrix, c4.5 classifier

5.6 Cross-validation datasets

In order to further cross-validate the classification algorithms and verify their real-time implementation, three datasets with a total length of about 4.5 hours were recorded with running feature generators and classifiers.

Most real-time calculations are precise compared to the Matlab reference. Only the two basic audio features spectral entropy and especially formant frequency sporadically show errors greater than 1%. This is due to issues with fix-point FFTs on the wearable which cannot be solved without major losses in computation speed. These errors propagate through to the classifiers, which differ from the Matlab reference in roughly 10% of the time in the category speech.

The classification results presented here have been computed (in Matlab) according to same procedures as above, using the Bayes classifier with single Gaussian observations. They are comparable to the ones achieved using the interleaved training and testing data, which confirms the generalization capability of the classifier. Overall accuracy of posture is virtually the same, activities and speech even attained a higher accuracy. Location classification failed completely, which strongly suggests using at least a combination of audio and WiFi, preferably GPS.

classified as -->	a	b	c	d	e	f	g	accuracy
a = unknown	0	0	0	0	0	0	0	N/A
b = lying	0	243	0	0	0	0	0	100%
c = sitting	0	3	3349	12	12	0	2	99%
d = standing	1	0	322	1001	23	0	40	72%
e = walking	0	0	11	45	200	0	7	76%
f = running	0	0	0	0	0	0	0	N/A
g = biking	0	0	0	8	2	0	540	98%
							class average:	89.1%
							overall accuracy:	91.6%

Table 5-12: Cross-validation posture confusion matrix (Bayes classifier)

classified as -->	a	b	c	d	e	f	g	h	accuracy
a = no activity	3466	227	147	24	17	0	6	72	88%
b = eating	73	574	106	0	0	39	2	0	72%
c = typing	4	36	2455	0	0	0	0	0	98%
d = shaking hands	0	0	0	0	0	0	0	0	N/A
e = clapping hands	0	0	0	0	0	0	0	0	N/A
f = driving	0	0	0	0	0	0	0	0	N/A
g = brushing teeth	0	0	0	0	0	0	0	0	N/A
h = doing dishes	68	2	3	0	0	0	0	25	26%
								class average:	70.9%
								overall accuracy:	88.8%

Table 5-13: Cross-validation activities confusion matrix (Bayes classifier)

classified as -->	a	b	c	d	e	f	accuracy
a = no speech	601	3	10	3	16	3	94%
b = user speaking	1	16	22	0	6	2	34%
c = other speaker	8	4	91	0	4	1	84%
d = distant voices	39	0	1	0	0	0	0%
e = loud crowd	12	0	10	1	17	0	43%
f = laughter	0	0	0	0	0	0	N/A
						class average:	51.1%
						overall accuracy:	83.2%

Table 5-14: Cross-validation speech confusion matrix (Bayes classifier)

classified as -->	a	b	c	d	e	f	g	h	accuracy
a = office	84	400	0	17	1	2	1	3	17%
b = home	1	35	0	4	0	1	6	1	73%
c = outdoors	47	8	0	0	5	2	29	1	0%
d = indoors	24	6	0	6	52	0	2	1	7%
e = restaurant	13	4	0	0	3	0	4	0	13%
f = car	0	0	0	0	0	0	0	0	N/A
g = street	17	18	0	0	3	3	59	0	59%
h = shop	22	11	0	0	23	2	53	5	4%
								class average:	24.6%
								overall accuracy:	19.6%

Table 5-15: Cross-validation location confusion matrix (Bayes classifier)

6 Post-classification: Modeling common sense

By taking into account the dependencies between the four categories, the overall classification accuracy based on 3.5 hours of fully labeled data can be improved from 76% to 83%. In this chapter several approaches on how to extract and use this mutual information are discussed and evaluated.

6.1 Common sense dependencies

Common sense can be used to describe several facts that a computer will a priori never know. The simplest example is marginal probabilities:

- $P(shaking\ hands) < P(eating)$
- $P(restaurant) < P(office)$

We can reason that certain combinations of context are more likely than others. For example:

- $P\left(shaking\ hands,\ standing\right) > P(shaking\ hands,\ sitting) > P(shaking\ hands,\ walking)$
- $P(shaking\ hands,\ me\ speaking) > P(shaking\ hands,\ no\ speech)$

Some combinations can safely be called impossible:

- $P(lying, doing\ the\ dishes) \approx 0$
- $P(biking, typing) \approx 0$

Then, partial knowledge of a person's context can be used to infer other context. Quite safely we can state the following conditional probabilities:

- $P(car \mid driving) \approx 1$
- $P\left(sitting \mid typing\right) \approx 1$

The activity typing will usually infer the posture sitting, and a classifier that detects driving has implicitly also detected car. The reverse though is not true. Being in a car will suggest we are sitting, but not necessarily that we are driving.

The rules discussed so far did not depend on time. Including the dynamics of life, further assumptions can be made. The context at a given time will depend on the context of the time before:

- $P(location_t = street \mid location_{t-1} = shop) > P(location_t = street \mid location_{t-1} = office)$

In that sense, transition probabilities between time steps can be learned, within a category and across categories. On a higher level we can go even further and end up with complete stories. For instance, going out for dinner will very likely contain the following sequence of activities: Walking into a restaurant, sitting down, selecting a menu, ordering, eating, standing up and walking out.
There are also statistics on the average duration of activities. Brushing teeth will typically last 2-3 minutes, eating 10 to 20 minutes and being in the office is in the range of several hours. Apart from duration, the absolute time also has a strong influence. The static probabilities mentioned above can all be made dependant on time of day.

Some of the rules listed above are pretty obvious and well defined by common sense. We will never be biking and typing at the same time. Many rules though reflect special behavior and are different for each subject. Some people go to work by car, others by bike, on foot, or by bus. Some people brush their teeth before breakfast, some after; others don't eat breakfast at all.

The experience sampling method was preferred to recording single activities because it yields information about the user's behavior. Statistics can be won on joint, conditional and marginal probabilities as well as duration and absolute timing of activities. Major problems, though, are completeness and generalization. A valid representation of behaviors requires a lot of data, sampled over several days at all times of the day. A concrete problem that could come up is for example the following. If in the past, a user exclusively recorded sleeping at home, the system will have learned to use sleeping to infer being at home. Should the person suddenly take a nap in the office or outside in a park, then the system will probably still detect lying down, but then might use that to infer the location home.

Because it takes a lot of training data to extract these rules, it is desirable that the system have the possibility to incrementally improve its common sense model by unsupervised learning. Coming back to the example, if the location classification is overruled for a long period of time by the common sense system, it seems plausible that the model is updated, trusting the location and posture classifiers as they are in this case quite certain about their decisions (home and office both are likely to have WiFi).

We believe that if behaviors are learned and modeled correctly, classification results can be expected to improve dramatically. With an increasing amount of labels this approach will become even more valuable, but also more difficult. It will be vital that the efforts of context recognition be combined with those in common sense modeling. An interesting approach on how to model common sense is LifeNet [29], which extracts knowledge from huge web-editable database (Open Mind Common Sense, [30]) and represents it as an undirected graphical model. The efforts of the Media Lab in common sense can be found in [31].

6.2 Initial position of post-classification

The basis for the work in this chapter is all the clips from the 24 hours of data used in the previous chapter which have confident labels for all four categories. This boils down to 631 20-second clips, or 3.5 hours of data. Each clip was classified using the Bayes classifier described in the previous chapter. That step is referred to as pre-classification, and the output is a probability vector for each category. The next step, discussed in this chapter, is referred to as post-classification. Any post-classification scheme will use the probability vectors as input observations and will decide on a final classification for each category.

The window sizes and time steps are not equal for all classifiers. Therefore, each clip contains several pre-classifications in each of the categories. All clips were concatenated and in order to model real-time behavior, in each pre-classification instance, the most recent classifications of the other three categories were repeated to form a 4-tuple.

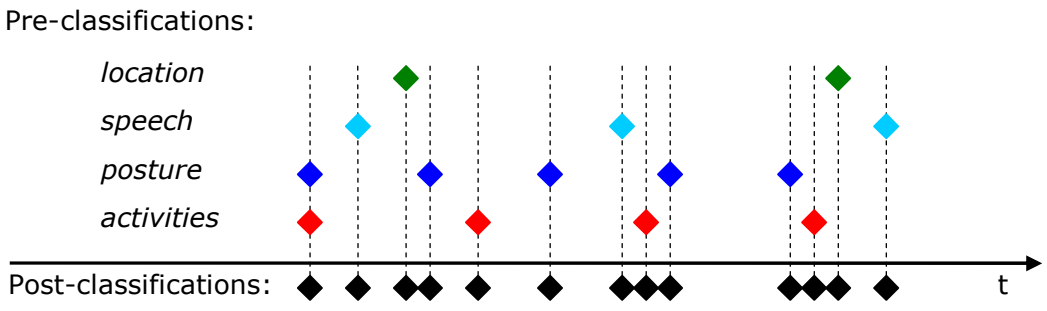


Figure 6-1: Timing for post-classification

In other words, if there were 10 pre-classifications for speech and location and 20 pre-classifications for posture and activities (this roughly corresponds to the difference in

window size), then there will be 10+10+20+20=60 post-classifications. The labels for post-classification also always correspond to the latest pre-classification label.
This timing concept makes sure the system always takes into account the latest classification results and has the potential to correct classifications in all categories even before new observations are available.

The different schemes are compared by post-classification accuracy. For a direct comparison to the pre-classifiers, the number of classification errors and accuracy was recomputed according to the presented timing scheme. The accuracies that are to be improved compute to

location:	62.10%
speech :	78.20%
posture:	91.60%
activities:	72.60%

Table 6-1: Pre-classification accuracies

6.3 Static approach with joint probabilities

In this approach we use the statistical information from the labeled data to learn joint, conditional and marginal probabilities. To facilitate discussion the following annotation scheme shall be defined:

$$P(A) \equiv P(location_t = A)$$
$$P(B) \equiv P(speech_t = B)$$
$$P(C) \equiv P(posture_t = C)$$
$$P(D) \equiv P(activities_t = D)$$

Without loss of generalization the computations are only described for post-classification of the location category. The other three categories are assumed to follow the same principles. The problem we are trying to solve is mathematically the following:

$$\tilde{A} = \underset{A}{\operatorname{argmax}}\, f\left(Obs_A, Obs_B, Obs_C, Obs_D\right),$$

where Obs_N is the observed probability vector of category N. Instead of using all four probability vectors we simplify and use only the whole observation vector of the category we are post-classifying; for the others we trust in the hard decision of the pre-classifier:

$$\tilde{A} = \text{argmax } f\left(Obs_A, \hat{B}, \hat{C}, \hat{D}\right) \text{ where } \hat{N} \equiv \text{argmax}(Obs_N) .$$

For the first approach, the post-classification probabilities are re-estimated according to the following:

$$P(A \mid B, C, D, Obs_A) = \frac{P(A, B, C, D, Obs_A)}{P(B, C, D, Obc_A)} \approx \frac{P(A, B, C, D)}{P(B, C, D)} P(A \mid Obs_A)$$

$P(A, B, C, D)$ was calculated simply by counting all combinations of 4-tuples of the labeled data and then normalizing them. To justify this approach in general, the following table shows the post-classification results if in each case the labels were used instead of the pre-classifier estimates for the three complementary categories (B, C and D in our notation convention).

	pre-classification errors	post-classification errors	improved	worse	unimproved
location:	6102 (38%)	3461 (21%)	2981	340	3121
speech :	3517 (22%)	2248 (14%)	1596	327	1921
posture:	1356 (8%)	682 (4%)	786	112	570
events :	4418 (27%)	655 (4%)	3940	177	478
total:	15393 (24%)	7046 (11%)	9303	956	6090

Table 6-2: Post-classification results using joint probabilities P(A,B,C,D) and correct labels

Improved: The absolute number of classification instances where post-classification corrected a pre-classification error

Worse: The instances where new errors were introduced by post-classification

Unimproved: The cases where pre-classification errors could not be corrected

The improvement from 24% errors to 11% errors is remarkable. However, these numbers serve not more than as an upper bound for the static approach. Using the labels as the given conditions is cheating, because the three complementary classifiers will of course never be faultless. The next table illustrates the realistic approach, where only the decisions of the pre-classifiers were used:

	pre-classification errors	post-classification errors	improved	worse	unimproved
location:	6102 (38%)	5612 (35%)	2250	1760	3852
speech :	3517 (22%)	3257 (20%)	654	394	2863
posture:	1356 (8%)	1236 (8%)	406	286	950
activities :	4418 (27%)	2330 (14%)	3360	1272	1058
total:	15393 (24%)	12435 (19%)	6670	3712	8723

Table 6-3: Post-classification using joint probabilities P(A,B,C,D) and pre-classification decisions

There is still a significant improvement. But only 2/3 of the previously corrected cases were improved this time and the number of worse classifications has more than tripled. This is due to the noise in the pre-classifier decisions. If the 3-tuples B, C and D do not correspond to the labels, the conditional probability $P(A \mid B, C, D)$ can be very wrong which results in "miss-corrections".

Another interesting finding is how well the four categories in our dataset could be classified using exclusively the hard decisions in the complementary categories. In other words, can A be guessed from B, C and D *only*? Formally,

$$\tilde{A} = \operatorname{argmax} P(A \mid \hat{B}, \hat{C}, \hat{D})$$

Using again first the correct labels we find the following:

	pre-classification errors	post-classification errors	improved	worse	unimproved
location:	6102 (38%)	5071 (31%)	3891	2860	2211
speech :	3517 (22%)	3543 (22%)	1713	1739	1804
posture:	1356 (8%)	2082 (13%)	834	1560	522
activities :	4418 (27%)	1548 (10%)	3804	934	614
total:	15393 (24%)	12244 (19%)	10242	7093	5151

Table 6-4: Post-classification results using P(A|B,C,D) only

The overall accuracies are surprisingly high. How can the performance compare to the cases where Obs_A is used? Firstly, it needs to be said that the same experiment using the pre-classifier decisions yields an error rate of 41%. Then, the 19% reached here are deceiving. The value is dominated by labels like "no speech", "sitting" or "no event" which account for most of the cases. The accuracies for less frequent labels suffer heavily under this approach. Brushing teeth for example is entirely classified as doing the dishes, because both activities are at home, standing and not speaking, but there were fewer samples of brushing teeth. Analog, running was entirely classified as walking.

6.4 Static approach with pairwise joint probabilities

While using the joint probabilities $P(A, B, C, D)$ to model the data is very precise, it scales badly in the number of categories and labels. For n categories with m labels each, the space grows to m^n. It is worth noting that for the studied data, of the possible 8 x 6 x 7 x 8 = 2688 combinations only 129 were featured. Because entries greater zero are needed for all expected realistic combinations, a lot of diverse training data will be required. For the same reason this approach is also prone to overfitting.

A model that is more tractable and less prone to overfitting is the following, using only pairwise conditional probabilities.

$$P(A \mid B, C, D, Obs_A) \approx f\left(P(A \mid B), P(A \mid C), P(A \mid D)\right) \cdot P(A \mid Obs_A)$$

Again, $P(A \mid B)$, $P(A \mid C)$ etc. can be calculated by counting labels, this time as 2-tuples. Now only $(n \cdot m)^2$ values are used. In this case, $(8 + 6 + 7 + 8)^2 = 841$ (8 + 6 + 7 + 8)^2 = 841, of which 281 are greater zero. The distributions are visualized in Figure 6-2. A can be read on the abscissa, B on the ordinate.

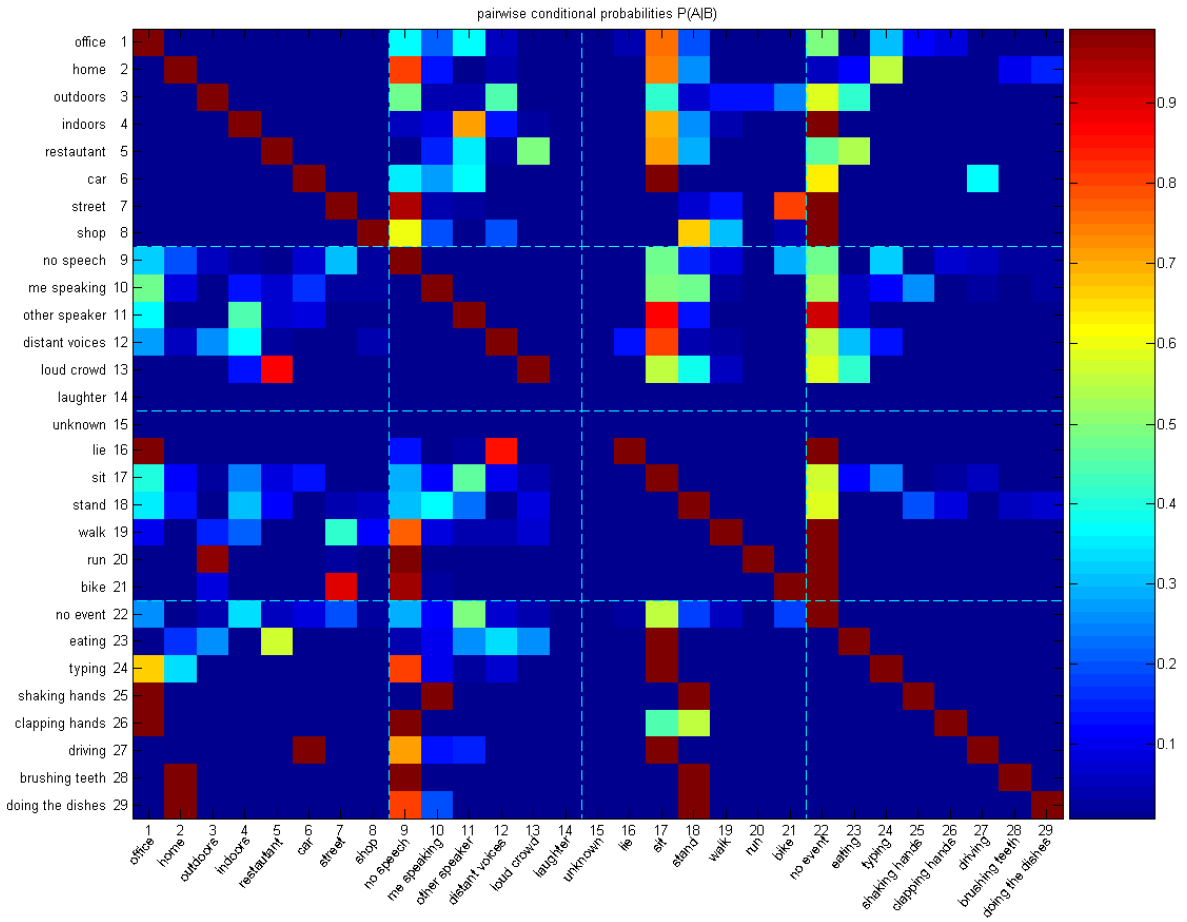


Figure 6-2: Pairwise conditional probabilities

We can see for example that the activity driving clearly implies the location car. However, the probability $P(no\ event \mid car)$ is actually greater than $P(driving \mid car)$, which is correct since the data contained more samples of the subject being a passenger, rather than a driver.

Different functions for combining the three conditional probabilities have been tested. The first is a convex combination:

$$P(A \mid B,C,D,Obs_A) \approx \left(\frac{1}{3} P(A \mid B) + \frac{1}{3} P(A \mid C) + \frac{1}{3} P(A \mid D) \right) \cdot P(A \mid Obs_A)$$

The classification results are as follows:

	pre-classification errors	post-classification errors	improved	worse	unimproved
location:	6102 (38%)	5391 (33%)	1733	1022	4369
speech :	3517 (22%)	3901 (24%)	220	604	3297
posture:	1356 (8%)	1602 (10%)	262	508	1094
activities :	4418 (27%)	3949 (25%)	2280	1811	2138
total:	15393 (24%)	14843 (23%)	4495	3945	10898

Table 6-5: Post-classification results using P(A|B) etc. in a convex combination

The fact that the three conditional probabilities are equally valued and added, prevents that one of them can exert much influence. The number of unimproved errors is high. To give each of them more power the following approach was tested:

$$P(A \mid B,C,D,Obs_A) \approx P(A \mid B) \cdot P(A \mid C) \cdot P(A \mid D) \cdot P(A \mid Obs_A)$$

This time the opposite was observed. Because each pair has a direct influence on the outcome, if only one of the three conditions is based on a wrong pre-classification decision post-classification will fail. The error rate amounted to 25% and more than half the class accuracies were below 50%, several at 0%.

The next approach tries to balance to two previous ones. Again the pairwise conditional probabilities are multiplied, but each of them is raised to the power of an exponent smaller one:

$$P(A \mid B,C,D,Obs_A) \approx P(A \mid B)^{\frac{1}{k}} \cdot P(A \mid C)^{\frac{1}{k}} \cdot P(A \mid D)^{\frac{1}{k}} \cdot P(A \mid Obs_A) \ , \quad (k > 1)$$

That way, the damage of a using a faulty pre-classification is lowered. Best performance was achieved with $k = 4$:

	pre-classification errors	post-classification errors	improved	worse	unimproved
location:	6102 (38%)	6162 (38%)	2372	2432	3730
speech :	3517 (22%)	3315 (21%)	745	543	2772
posture:	1356 (8%)	1947 (12%)	457	1048	899
activities :	4418 (27%)	2859 (18%)	3298	1739	1120
total:	15393 (24%)	14283 (22%)	6872	5762	8521

Table 6-6: Post-classification results using P(A|B)^(1/k)

The number of improvements is high, but also the number of newly introduced errors. This gives cause to think about why for instance the posture classification becomes worse. A notion of certainty needs to be introduced to prevent the post-classifier from worsening labels that are actually very well classified. Normalizing the post-classification output probabilities seemed a valid approach.

$$c(A_i) \equiv \frac{P(A_i \mid B, C, D, Obs_A)}{\sum_i P(A_i \mid B, C, D, Obs_A)}$$

$c(A_i)$ is a measure of certainty for post-classification results. This could be compared against a threshold. Better is a comparison with a measure for the certainty of the pre-classifiers. A classifier can be trusted when its false positive rate is very low. We thus use one minus the false positive rate as the pre-classification certainty measure. The prediction of the post-classifier shall only be used if $c(\tilde{A}) > 1 - fp$. It turned out that for some particular cases $c(\tilde{A})$ was close to one, but post-classification was sill introducing errors. This suggested another rule according to which pre-classifications can only be overruled if their false positive rate is above a certain threshold. The following statement summarizes these thoughts formally:

$$\tilde{\tilde{A}} = \begin{cases} \tilde{A} & c(\tilde{A}) > 1 - fp \ \ \& \ \ fp > fp_{\min} \\ \hat{A} & \text{else} \end{cases}$$

The results are significantly better than before, without certainty measures:

	pre-classification errors	post-classification errors	improved	worse	unimproved
location:	6102 (38%)	5495 (34%)	2240	1633	3862
speech :	3517 (22%)	3037 (19%)	662	182	2855
posture:	1356 (8%)	1668 (10%)	190	502	1166
activities :	4418 (27%)	1740 (11%)	3271	593	1147
total:	15393 (24%)	11940 (19%)	6363	2910	9030

Table 6-7: Post-classification results using P(A|B)^(1/k) and certainty measures

The number of improvements dropped only a little, while the number of introduced errors dropped by half.
Several different methods on how to further improve the certainty measures were investigated, some also including Obs_A as it can be reasoned that the probability vector can indicate a classifiers certainty for the particular decision. For instance:

$$c(A_i) = w \cdot Obs_A(i)^{\alpha} + (1-w) \cdot (1-fp)^{\beta} \qquad w \in [0..1]$$

The improvements were only minor and did not really impress, given the simplicity of the special case with $w = 0$ and $\beta = 1$.

Classification accuracy has been considerably improved in each category. With these better predictions, why not go through the same process again? Indeed, after iterating twice, the error rate drops to 17%. We show the results after three iterations, more did not yield any further improvement.

	pre-classification errors	post-classification errors	improved	worse	unimproved
location:	6102 (38%)	4662 (29%)	2199	759	3903
speech :	3517 (22%)	3022 (19%)	639	144	2878
posture:	1356 (8%)	1431 (9%)	222	297	1134
activities :	4418 (27%)	1705 (11%)	3302	589	1116
total:	15393 (24%)	10820 (17%)	6362	1789	9031

Table 6-8: Post-classification using P(A|B)^(1/k) and certainty measures over three iterations

This is the best result achieved with the static approach.

Since the location classifier still has an error rate of 29%, and we previously argued that location recognition can be expected to become very precise using GPS and WiFi, we would like to show how the results are expected to improve assuming perfect location classification. With the same post-classification scheme as above, including the three iterations and certainty measures, by using the location labels instead of the pre-classification decision we achieve an overall classification error of 14.8%

	pre-classification errors	post-classification errors	improved	worse	unimproved
location:	6102 (38%)	4031 (25%)	2272	201	3830
speech :	3517 (22%)	2964 (18%)	664	111	2853
posture:	1356 (8%)	1103 (7%)	297	44	1059
activities :	4418 (27%)	1455 (9%)	3258	295	1160
total:	15393 (24%)	9553 (15%)	6491	651	8902

Table 6-9: Post-classification results using P(A|B)^(1/k) and certainty measures over three iterations, with correct location labels

6.5 Dynamic approach with the influence model

The static approaches discussed are simple, efficient and effective in the sense that they do capture a lot of common sense information. Still, it is a goal to capture also temporal dependencies within and between the different categories. This section deals with the

influence model, a scalable version of coupled HMMs, which is currently studied intensively in our group.

The influence model is a novel approach for studying multi-agent dynamics. The computational difficulty of studying multi-agent dynamics is the astronomical number of combined states: Suppose we have 100 interacting agents in a system (which is typical of an organization), and each agent can take 2 states. It follows that the whole system will have 2^{100} combined states. The influence model copes with the exploding number of states by linearly combining the contributions of the interacting agents to each other. This model can typically improve the inference precision of individual agents by about 10% to 20% by combining observations on other agents.
A detailed description of the model is beyond the scope of this thesis. The mathematical background is explained in [32] and a typical application of the model can be found in [33].

For the application in our case as a post-classifier, the agents correspond to the four classification categories. The model thus includes four chains with as many latent states as labels in each category and uses the pre-classification probability vectors as continuous observations. The model is trained with 100 iterations on the entirety of the data using the labels as observed latent states. The influence matrix resulting from the EM-learning process is visualized in Figure 6-3.

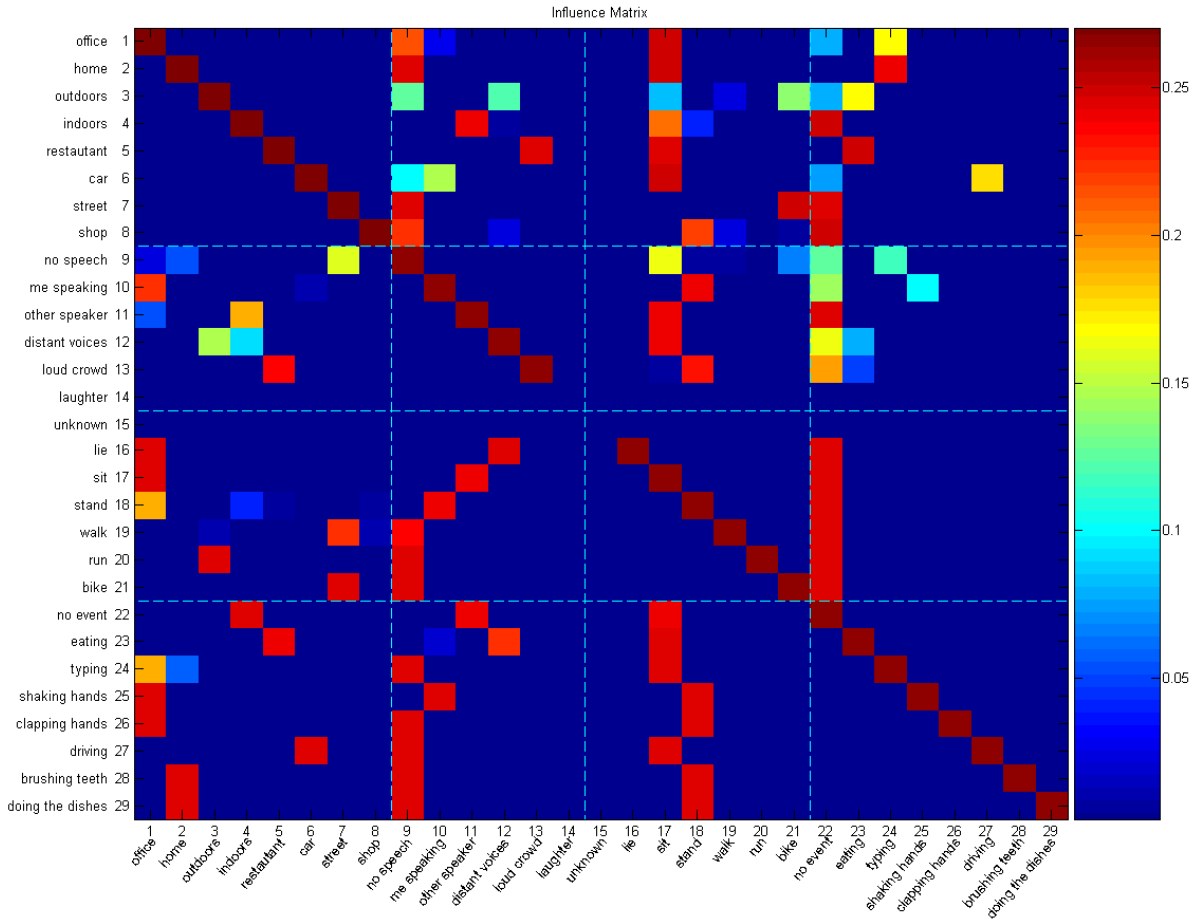


Figure 6-3: Influence matrix learned by EM-algorithm

The matrix must be interpreted as follows: Rows represent the states at time t, columns the states at time $t+1$. The influence executed by the state X_t on state Y_{t+1} is found in $row(X)$ and $col(Y)$. We see that the influence matrix resembles the previously discussed pairwise conditional probabilities. Again, driving clearly implies car, but car implies driving *or* doing nothing. However, is this what we want? We must expect (and want) the influence matrix to differ from the previous model for two reasons.

Firstly, the influence matrix describes transition probabilities that are related to the dynamics of the system. We would thus like to capture some information like "biking is likely to transition to standing, and then walking". Seen as a whole, context changes are rare though, because each bigger "chunk of context" contains several windows. This "stickiness" of the data results in very high self transition probabilities and reduces the influence of actual transitions from one state to another. As a result, the values in the influence matrix resemble the static conditional probabilities from above.

Then secondly, the influence matrix is the result of an EM-learning process. By definition it will converge to something that will maximize the overall likelihood. As a result, many transitions that are in the data will end up in values barely greater zero. For instance, as

the previous model showed nicely, sitting can imply several different locations. In the influence matrix sitting mainly implies office (other entries are close to zero), which is the most likely.

Let us now see how the influence model performs as a post-classifier. To comply with real-time conditions, the classification is based on the last k cases. For $k=3$ we get the following results:

	pre-classification errors	post-classification errors	improved	worse	unim-proved
location:	6102 (38%)	4552 (28%)	2438	888	3664
speech :	3517 (22%)	2986 (19%)	821	290	2696
posture:	1356 (8%)	1260 (8%)	147	51	1209
activities :	4418 (27%)	2818 (17%)	2009	409	2409
total:	15393 (24%)	11616 (18%)	5415	1638	9978

Table 6-10: Post-classification results using the influence model over the last 3 timesteps

By increasing k we would expect better results because more temporal information is used. However the opposite is the case. The performance decreases slightly until we obtain for $k \geq 7$ the following:

	pre-classification errors	post-classification errors	improved	worse	unim-proved
location:	6102 (38%)	4878 (30%)	1839	615	4263
speech :	3517 (22%)	2861 (18%)	927	271	2590
posture:	1356 (8%)	1289 (8%)	158	91	1198
activities :	4418 (27%)	2849 (18%)	1693	124	2725
total:	15393 (24%)	11877 (18%)	4617	1101	10776

Table 6-11: Post-classification results using the influence model over the last 7 timesteps

There is improvement in all categories and the overall classification accuracy has increased from 24% to 18%. This result does not outperform the static post-classifying approaches above, but is very encouraging considering that the model was not fine tuned and there are no certainty measures used, which turned out to be valuable in the static approaches.
The way it looks, there is not enough information on meaningful transitions in the training data. The influence model seems to get most of its gain from modeling conditional probabilities. It might well be the case that with more data, the influence model will better represent the common sense dependencies we are looking for. One idea is to shorten long periods of recurring states for the training process to emphasize transitions.

The influence model has been implemented in real-time and was integrated into the enchantment architecture. Details are in Appendix C.

6.6 Final comparison

To conclude this chapter we repeat the detailed results for pre- and post-classification. The starting positions were the following:

classified as -->	a	b	c	d	e	f	g	h	accuracy
a = office	2645	1638	0	1110	18	0	12	20	49%
b = home	210	1210	0	84	12	0	30	4	78%
c = outdoors	266	38	86	0	16	38	250	0	12%
d = indoors	942	322	0	2184	34	0	50	0	62%
e = restaurant	54	52	0	16	1088	0	0	12	89%
f = car	0	0	0	0	0	1344	22	0	98%
g = street	260	222	8	6	66	68	1426	6	69%
h = shop	76	66	0	4	0	0	70	24	10%
								class average:	58.4%
								overall accuracy:	62.1%

Table 6-12: Location confusion matrix after pre-classification

classified as -->	a	b	c	d	e	f	accuracy
a = no speech	6075	20	124	40	22	0	97%
b = user speaking	94	1176	966	0	108	18	50%
c = other speaker	290	44	4837	114	241	0	88%
d = distant voices	752	0	428	46	20	0	4%
e = loud crowd	144	6	68	12	458	6	66%
f = laughter	0	0	0	0	0	0	N\A
						class average:	60.7%
						overall accuracy:	78.2%

Table 6-13: Speech confusion matrix after pre-classification

classified as -->	a	b	c	d	e	f	g	accuracy
a = unknown	0	0	0	0	0	0	0	N/A
b = lying	0	192	0	0	0	0	0	100%
c = sitting	76	18	9706	404	0	0	44	95%
d = standing	0	0	436	2449	18	0	176	80%
e = walking	6	0	24	42	558	0	14	87%
f = running	0	0	0	0	4	92	0	96%
g = biking	0	0	18	60	16	0	1756	95%
							class average:	91.9%
							overall accuracy:	91.6%

Table 6-14: Posture confusion matrix after pre-classification

classified as -->	a	b	c	d	e	f	g	h	accuracy
a = no activity	6756	1150	2224	4	0	282	6	10	65%
b = eating	80	711	300	0	0	42	0	0	63%
c = typing	18	40	2506	0	0	0	0	0	98%
d = shaking hands	50	0	0	554	0	0	0	0	92%
e = clapping hands	0	0	0	0	470	0	0	0	100%
f = driving	12	0	14	0	0	478	0	0	95%
g = brushing teeth	26	18	0	0	0	0	118	0	73%
h = doing dishes	112	0	18	8	0	0	4	98	41%
								class average:	78.2%
								overall accuracy:	72.6%

Table 6-15: Activities confusion matrix after pre-classification

The best post-classification was achieved using pairwise conditional probabilities and a certainty measure. The confusion matrices show a significant improvement:

classified as -->	a	b	c	d	e	f	g	h	accuracy
a = office	4019	513	3	785	17	35	71	0	74%
b = home	216	1262	0	35	4	0	33	0	81%
c = outdoors	148	139	92	8	18	33	256	0	13%
d = indoors	1111	248	1	1889	42	0	241	0	53%
e = restaurant	61	34	1	22	1096	6	2	0	90%
f = car	122	64	14	0	0	1150	16	0	84%
g = street	65	26	14	2	27	8	1920	0	93%
h = shop	85	36	0	16	0	0	84	19	8%
								class average:	62.1%
								overall accuracy:	71.1%

Table 6-16: Location confusion matrix after post-classification

Location classification, which was based on audio only, improved from 62.1% to 71.1% overall accuracy. The average per class accuracy improved in five of eight cases.

classified as -->	a	b	c	d	e	f	accuracy
a = no speech	6114	36	115	0	16	0	97%
b = user speaking	65	1657	540	0	100	0	70%
c = other speaker	330	44	4898	23	231	0	89%
d = distant voices	766	0	455	18	7	0	1%
e = loud crowd	176	6	112	0	400	0	58%
f = laughter	0	0	0	0	0	0	N\A
						class average:	63.0%
						overall accuracy:	81.2%

Table 6-17: Speech confusion matrix after post-classification

Again, both overall accuracy and average class accuracy is improved for speech classification. A much better classification is done of the user speaking. The label distant voices still cannot be classified, another hint for missing consistency in applying that label.

classified as -->	a	b	c	d	e	f	g	accuracy
a = unknown	0	0	0	0	0	0	0	N/A
b = lying	0	143	37	12	0	0	0	74%
c = sitting	0	0	9789	432	0	0	27	96%
d = standing	0	0	385	2540	7	0	147	82%
e = walking	0	0	49	134	451	0	10	70%
f = running	0	0	0	0	4	92	0	96%
g = biking	0	0	66	105	16	0	1663	90%
							class average:	84.7%
							overall accuracy:	91.1%

Table 6-18: Posture confusion matrix after post-classification

For the category posture, there is no gain in post-classification. The overall accuracy is the same, but several classes lost in average accuracy. The reason for this is that mis-

classifications in the other three categories (it must be recalled that we start from a 62% accuracy in location classification) introduced new errors.

classified as -->	a	b	c	d	e	f	g	h	accuracy
a = no activity	9904	178	346	4	0	0	0	0	95%
b = eating	461	560	112	0	0	0	0	0	49%
c = typing	158	22	2384	0	0	0	0	0	93%
d = shaking hands	50	0	0	554	0	0	0	0	92%
e = clapping hands	0	0	0	0	470	0	0	0	100%
f = driving	111	0	0	0	0	393	0	0	78%
g = brushing teeth	33	11	0	0	0	0	118	0	73%
h = doing dishes	199	0	7	9	0	0	4	21	9%
								class average:	73.6%
								overall accuracy:	89.4%

Table 6-19: Activities confusion matrix after post-classification

With an increase from 72.6% to 89.4%, the category activities has the largest improvement in overall accuracy. This is mainly due to much higher accuracy for the garbage class "no activity". There are thus much less false positives for all activities. However, there are more false negatives. This can be desirable or not, depending on the application.

The jump in overall classification accuracy from 74% to 83% supports the approach of using common sense models to improve context classification. It can be argued that the statistics were extracted from the testing data and that this weakens the demonstrated results. We are aware of that fact and plan to do more testing on different data. The use of the entire data available was in our case necessary to attain a valid common sense model.

7 Interest prediction

As mentioned in the introduction, a longed for application in wearable computing is an automated diary that records and categorizes video and audio clips of a user's everyday life. We believe that context recognition can be used to determine interesting moments in a wearer's life and suggest an application that decides in real-time whether a recording is worthwhile storing or not.
In this chapter a scheme is presented for assessing the level of interest in a given moment on the basis of the context classification presented in the previous chapters. The scheme was implemented and an experiment was conducted that justified this approach.

7.1 What are interesting moments?

Obviously, not all 24 hours of a person's day are equally interesting. About a third of our time we are sleeping, the vast part of daytime is often spent at an office desk and long periods of time can be spent driving, sitting in a bus, reading a book or watching TV. These activities can of course be interesting and should make part of a diary. However, memorable things usually very often happen when these reoccurring patterns are interrupted.
This said, the task can be broken up into two parts: Firstly, we wish to capture samples of a person's everyday life. Secondly, we would like to document more extensively extraordinary events, like an excursion or a party. This separation is on a very high and abstract level. In order to determine an interruption of everyday life we first need a very good idea of what a user's everyday life is like. This requires data over a long period of time, which we currently do not have for this work. An example that shows, however, that learning everyday lives of people is possible is the work of Nathan Eagle in reality mining [34].
The notion of interesting moments used here is on a lower level. It is based on several simple assumptions:

- There is uninteresting context such as typing, driving, or lying down.
- There is moderately interesting context such as speech, restaurant or eating.
- There is explicitly interesting context such as laughter, shaking hands and clapping hands.
- Long stretches of uninteresting context like a 15 minute bike ride need only be captured once, because numerous images will not increase the amount of information.

- Changes in context indicate possibly interesting interruptions, or new activities.

An algorithm that combines these aspects is described in the next section.

7.2 Interest prediction algorithm

An algorithm was implemented that calculates the current level of interest based on the context classification. If that level exceeds a certain "interest threshold", the system detects a moment of interest. It will capture am image and store it together with the current context information.
The algorithm combines three measures:

1. The accumulated static interest, based on an interest map
2. Interest bonus for state transitions
3. Time since the last moment of interest

The static interest is the sum of interest points that correspond to the current classification of location, speech, posture and activities. The interest map below shows the mapping between labels and interest points.

	Interest points		Interest points
Location		**Posture**	
office	0	unknown	0
home	0	lying	0
outdoors	1	sitting	0
indoors	1	standing	1
restaurant	1	walking	1
car	0	running	3
street	1	biking	0
shop	1		
		Activities	
Speech		no activity	0
no speech	0	eating	2
user speaking	2	typing	0
other speaker	2	shaking hands	5
distant voices	1	clapping hands	5
loud crowd	2	driving	0
laughter	5	brushing teeth	0
		doing the dishes	0

Table 7-1: Assignment of interest points

By default the interest threshold is set to 5. This means, that as soon as e.g. shaking hands is detected, a picture is taken.

Then, to detect context transitions, the classifications over the last one minute are stored and a super-state is computed. The super-state for each category corresponds to the label which was classified most during that minute. Each time there is a change in super-state in any context category, a transition bonus of 0.5 points is added.

Finally, in order to make sure pictures are taken every once in a while even when the interest level is below its threshold, the time since the last picture is taken into account. Every second, 1/120 of a point is added. This is equivalent to one point every 2 minutes or 5 points, and thus a picture, every 10 minutes.

Each time a moment of interest is detected, the two counters for transition bonuses and time elapsed since last picture are reset to zero. In addition a hold-off period of 5 seconds will make sure pictures are not taken in masses for instance in the case of several seconds of laughter.

The most obvious result of this algorithm is the fact that pictures are taken at a low frequency when the user is not engaged in anything interesting over a long period of time and a higher frequency during interesting activities.
The numeric values were chosen as such, that in a typical recording, the average frequency of images taken is approximately one every 1-2 minutes. This varies, as mentioned, from one picture every 10 minutes for a user working on his computer in the office to several pictures per minute during a discussion in a restaurant over lunch.

7.3 Experiment description

An experiment was conducted to assess the algorithms performance. In one of the sessions previously used in Chapter 5 to cross-validate the classification algorithms, two sets of pictures were recorded. Set A contains the "interesting" pictures that were initiated by the described algorithm; set B took pictures as usual once every minute. During the 3 hours, a total of 114 pictures were taken for set A, and 178 pictures were taken for set B. To make the two sets comparable, every third picture in set B was dropped. The two sets of pictures were printed and displayed at the lab with voting slips that could be placed in an urn. The concept of the experiment was briefly explained, and people were asked which set they found more interesting and why. The same was done by means of email. The complete set of images can be found in Appendix G.

7.4 Results

In this experiment the algorithm clearly did a better job in distinguishing interesting moments. From a total of 28 votes received, 26 were for set A and only 2 for set B. About two thirds of the people mentioned the ratio of laptop pictures, about half mentioned the surplus of images with people in set A and some found that set B had too many repetitive pictures e.g. biking. In the following some of the results are discussed and explained.

There are 15 laptop pictures in set A versus 47 in set B. It should be noted that the ration of laptop pictures was only 3 to 17 before the lunch but 12 to 30 after lunch, mainly because my office mate was in a discussion with a colleague. This case suggests a measure to determine if the recognized speech actually involves the user.
The lunch scene was clearly better documented in set A (30 images) than in set B (11 images). What is particularly nice is that at the end of the lunch I shook hands with three people, and in two cases an image was taken (images 43 and 45). This can be guessed from image 45, but in image 43 the person was not in the field view anymore, probably due to the latency of the classifiers. The audio 5 seconds prior to the picture does reveal the person's name though.
Image 103 of set B shows a short discussion with my office mate. The same discussion was documented with two pictures (99 and 100) in set A.
Images 109 through 114 in set B document a nap at the lab. In set A this was possible with only one image.
Right after that nap I accidentally got involved in a discussion with people in the lab. There was a lot of laughter which resulted in the algorithm taking 9 pictures of that 4 minute conversation instead of only 3 in set B.
Interesting is also the number of bike ride pictures:

	Set A	Set B
lunch → supermarket	1	2
supermarket → home	0	3 to 4
home → shop	2	2
shop → lab	2	4

Table 7-2: Number of biking images

In three of four cases set A needed less or equal pictures to document the ride. However in one case the algorithm clearly failed. A picture was taken on the way out of the supermarket, thus resetting the transition counters. The four minutes of biking (2 points) plus the transitions shop to street & walking to biking (0.5 points each) and the static interest of street/outdoors (1 point) were not enough to pass the threshold.

It also needs to be said, that 7 pictures in the supermarket were initiated by the misclassification of clapping hands. Such false positives do of course affect the results directly.

Overall, the results are very pleasing and suggest that this approach, simple as it is, can increase the "amount of interest" in recorded pictures. Also it can be customized to a user's preferences by assigning different values to interest points and by adjusting the interest threshold. Something that remains to be studied is how this approach can scale downwards to taking only a handful of pictures per day. Will the most interesting moments still be captured? For that goal it will be important to incorporate behavioral patterns on a higher level, as discussed above.

8 Future work

8.1 Classification architecture

The architecture chosen worked out for the most part. We believe the approach using a post-classifier to combine the information of specialized pre-classifiers to improve the final classification is a good one. There are a few shortcomings though with the pre-classification schemes used.

Some of the labels are not really mutually exclusive. This problem is experienced during annotation and shows up in the classification results. Posture and activity labels were chosen well in this respect. But there are issues in location and speech. What should be done if two people are in a noisy restaurant discussing something and laughing? Is that a loud crowd, me speaking or laughter? Also, the location labels outdoors and indoors are ambiguous. It is almost impossible to distinguish home from indoors using audio.

An idea would be to split up location into two separate categories, one based on WiFi, the other on audio and perhaps later also vision. The WiFi labels would be something like office, home, shop, restaurant, unknown network and no network. The second, concurrent location category called "scene" would contain the labels indoors, outdoors, street and car. By mapping home, office etc. to indoors, there would be less ambiguity for the audio based classifier, which would increase its performance. This could be done by connecting the two location classifiers, or simpler by the post-classifier.

To better model speech situations it might be useful to split the labels into binary decisions: speech / non-speech. In case of speech: user speaking / other speaker and laughter / no laughter. That way, each classifier can be tailored on a simpler task using specialized features.

This leads to another problem, the difference in durations of certain classes. Activities like eating or doing the dishes have a longer time horizon than shaking hands or clapping hands. Yet, they are currently compared by the same classifier using the same window size, which is basically dictated by the activity with the shortest duration. Again, a separation of the category might be a solution: one category called "lengthy activities" for eating, typing, driving, doing the dishes and no event, and binary categories for shaking hands, clapping hands and brushing teeth. The final decision could still be viewed as one mutually exclusive category: Most of the time the decision for activities is taken from "lengthy activities" and in case one of the binary classifiers fires that decision would be overruled. This approach will again permit the classifiers to be customized.

One thing that has not been tackled at all is the abstraction of higher level activities. Initially planned labels for the category speech that have been removed are conversation, giving a talk and attending a talk. These labels again span a longer time frame and are better not classified based on low-level audio features. Rather more, they are defined by different patterns of user speaking / other speaker. Similarly, high level activities such as attending a concert, having a meeting, commuting to work or shopping are based on patterns and combinations of the underlying classification instances.
A next generation of algorithms will want to approach these higher levels of context by using hierarchical models that build on the current low-level classifications.

8.2 More data more systems

Expanding the system to new labels will require acquisition of additional user data. This will become more important for training higher level classifiers. It will also be vital to combine data recorded by several users to asses the generalization capability of the classifiers. For this it will be necessary to replicate the system. This will also open up the possibility of distributed experiments with several users interacting.

8.3 Common Sense modeling

With bigger datasets available, common sense modeling can be tested more effectively. Post-classification is not yet part of the current implementation. Studying the effect of post-classification in real-time should be interesting and might yield further insights. As the number of labels increases one must think of self-learning models or at least a process for incremental up-dates.
We see big potential in using the influence model, the manner of application must be further studied though.

8.4 New features and additional sensors

Using the current sensing configuration there is still a potential for improvement. Especially for location, well selected audio features should already make a big difference. Also, visual features should be tested. They might help distinguish indoor from outdoor locations.
Then the classification of activities could be improved using audio.
Acceleration features have been tested quite extensively for the current configuration. However, if as suggested specialized classifiers for certain activities are built, particular features with optimized window sizes will perform better.

Might we want to test additional sensors? Obviously more information is a potential but means more computation and can be a source of errors. The strength of this platform is its simplicity and the sensor placements were chosen so that once the PDA disappears, for instance into a belt buckle, the system will actually be wearable by non-cyborgs. It is thus not in the spirit of this work to suggest many new sensor placements, like accelerometers on thighs for instance, although that would definitely improve distinguishing sitting from standing.
What might be interesting though is some more information from the wrist sensor to better classify activities. Concretely, a microphone on the wrist would help locating audio sources by comparing intensities of wrist and chest audio, which might help attribute sounds to a related activity such as clicking of keys to typing. Further sensors that could provide valuable information from the wrist would be a strain gauge, GSR and EKG. The former could serve as an indication for grabbing objects, the latter two could tell us something about a wearer's physiology – a context dimension not yet studied so far.

One cheap "sensor", which is kind of obvious but has not yet been included in this work, is time. We expect a significant gain in post-classification through time conscious common sense models. Also, time of day and day of week are central ingredients for modeling higher level behaviors.

8.5 Other classifiers

In the range of classifiers there are still a few things to try out. First, all classifiers studied were trained exclusively on labeled data. The experience sampling annotation method does not provide labels that are valid over the entire data set. A lot of unlabeled data remains, which can be included for semi-supervised learning.
As mentioned previously, it is unclear how the selected c4.5 decision tree algorithm performs pruning and to what extent that influences accuracy and overfitting. An analysis of the tree structure might yield more insights. Another implementation of c4.5 can be found in the WEKA toolbox [35].
An additional well-known classification method that has not been used is Support Vector Machines (SVMs). There are implementations available in C and in Matlab.

8.6 Annotation tool

A nice enhancement of the offline annotation tool that would be easy to implement is "label suggestions". Classifications that are either done on the wearable in real-time or in a preprocessing step offline could be loaded into the SQL database. That could at the same time speed up the labeling process and provide practical insights into which situa-

tions are misclassified. As a byproduct the clips that needed correcting could be tagged and used specifically to improve trained models.
The annotation tool should also be adjusted to provide a distinct presentation of clips that were classified as interesting.

9 Summary

The goal of this thesis was to build a wearable real-time context recognition system. The hardware platform presented is based on an off-the-shelf PDA and includes two wireless accelerometers. This provides the following sensing layout: ambient audio from the chest, acceleration from the hip and the wrist of the dominant hand, WiFi network detection and images taken from the wearer's chest.

The categories – location, speech, posture and activities – were chosen to represent many diverse aspects of a user's context, the labels in each category to represent situations in everyday life. In order to build models that scale out of a laboratory setting, naturalistic user data was collected by means of interval-contingent experience sampling. During recording sessions no user interaction is required. After recording, pictures that have been captured once a minute are presented along with the corresponding audio clip and a list of labels for each category. Assigning a confidence level to each category permits reducing the amount of false labels. A total amount of about 35 hours of data was collected, of which about 24 hours were suitable for the study.

The classification process is split up into two parts. In a pre-classification step a probability vector is computed for each category. In a post-classification step the information from all four categories is combined and evaluated by a common sense model to come up with a final decision. Posture and activity classification rely on acceleration features, speech and location classification on audio features. Pre-classifiers tested were: Naive Bayes with single Gaussians as well as Gaussian mixture models, c4.5 decision trees and hidden Markov models. The achieved accuracy values range from 91% to 95% for posture, 68% to 92% for activities, 80% to 81% for speech and 62% to 67% for location (without WiFi).

Several approaches for modeling common sense dependencies between categories are presented. A promising static approach uses pairwise conditional probabilities and a promising dynamic approach is the influence model. The best model led to an increase in overall classification accuracy from 76% to 83%.

Finally, the focus was set on a diary application. Most prior work in the field of automated diaries has tackled the problem of categorizing and searching user data offline. The novel approach presented here uses information on the user's context to evaluate the current situation in real-time. An algorithm is suggested that predicts the current level of interest

based on certain statistics of a user state. If a moment of interest is detected, an image is taken and audio is recorded. This process was done in a three-hour recording session. The images taken by the algorithm were compared against the same number of images taken at regular intervals. An overwhelming majority of people voted for the interest prediction algorithm!

References

[1] W.A. Rogers, B. Meyer, N. Walker, A. D. Fisk, "Functional limitations to daily living tasks in the aged: a focus groups analysis", Human Factors, 40:111–125, 1998

[2] A. Pentland, "Healthwear: Medical Technology Becomes Wearable", IEEE Computer, May 2004

[3] M. Sung, "Non-Invasive Wearable Sensing Systems for Continuous Health Monitoring and Long-Term Behavior Modeling", Ph.D. Thesis MIT, Department of Electrical Engineering, October 2005

[4] M. Sung, A. Pentland, "Minimally Invasive Physiologic Sensing for Human-Aware Interfaces", HCI International, Third International Conference on Universal Access in Human-Computer Interaction, July 2005

[5] J. Ho, S. S. Intille, "Using context-aware computing to reduce the perceived burden of interruptions from mobile devices", in Proceedings of CHI 2005

[6] N. Kern, B. Schiele, "Context-Aware Notification for Wearable Computing", Proceedings of the IEEE International Symposium on Wearable Computing (ISWC 2003)

[7] http://bourbon.usc.edu/iml/recall/

[8] Vannevar Bush's MEMEX (1945) http://www.theatlantic.com/doc/194507/bush

[9] J. Gemmell, G. Bell, R. Lueder, S. Drucker, C. Wong, "MyLifeBits: Fulfilling the Memex Vision", ACM Multimedia '02, December 1-6, 2002, Juan-les-Pins, France, pp. 235-238

[10] A. Fitzgibbon, E. Reiter, " 'Memories for life': Managing information over a human lifetime", UK Computing Research Committee Grand Challenge proposal, 2003

[11] S. Vemuri, W. Bender, "Next-generation personal memory aids", BT Technology Journal, Vol 22, No 4, October 2004.
Project: http://web.media.mit.edu/~vemuri/wwit/wwit-overview.html

[12] http://www.sigmm.org/Members/jgemmell/CARPE

[13] C. Dickie, R. Vertegaal, D. Chen, D. Fono, D. Cheng, C. Sohn, "Augmenting and Sharing Memory with eyeBlog", Proceedings of 1st ACM Workshop on Continuous Archival and Retrieval of Personal Experiences. NYC: ACM Press, 2004

[14] K. Van Laerhoven, O. Cakmakci, "What shall we teach our pants?", Fourth International Symposium on Wearable Computers, pages 77–83, IEEE Press, 2000

[15] P. Lukowicz, J.A. Ward, H. Junker, M. Stäger, G. Tröster, A. Atrash, T. Starner, "Recognizing workshop activity using body worn microphones and accelerometers", Pervasive Computing: Proc. of the 2nd Int'l Conference. (2004) 18–22

[16] L. Bao, S. S. Intille, "Activity recognition from user-annotated acceleration data", Pervasive Computing: Proc. of the 2nd Int'l Conference. (2004) 1–17

[17] N. Kern, B. Schiele, A. Schmidt, "Multi-sensor activity context detection for wearable computing", European Symposium on Ambient Intelligence, 2003

[18] D. Wyatt, M. Philipose, T. Choudhury, "Unsupervised Activity Recognition using Automatically Mined Common Sense", Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI 2005)

[19] S. S. Intille, K. Larson, J. S. Beaudin, J. Nawyn, E. Munguia Tapia, P. Kaushik, "A living laboratory for the design and evaluation of ubiquitous computing technolo-

gies", in Extended Abstracts of the 2005 Conference on Human Factors in Computing Systems . New York, NY: ACM Press, 2004.

[20] E. Munguia Tapia, S. S. Intille, K. Larson, "Activity recognition in the home setting using simple and ubiquitous sensors", Proceedings of PERVASIVE 2004, vol. LNCS 3001, A. Ferscha and F. Mattern, Eds. Berlin Heidelberg: Springer-Verlag, 2004, pp. 158-175

[21] J. Lester, T. Choudhury, N. Kern, G. Borriello, B. Hannaford, "A Hybrid Discriminative-Generative Approach for Modeling Human Activities", Proceedings of International Joint Conference on Artificial Intelligence (IJCAI 2005)

[22] Krause, A., Smailagic A., Siewiorek D., Farringdon J., "Unsupervised, Dynamic Identification of Physiological and Activity Context in Wearable Computing", ISWC 2003

[23] http://www.handheld-linux.com/wiki.php?page=SL6000 and http://www.sharpusa.com/products/ModelLanding/0,1058,1255,00.html

[24] E. M. Tapia, N. Marmasse, S. S. Intille, K. Larson, "MITes: Wireless portable sensors for studying behavior", Proceedings of Extended Abstracts, Ubiquitous Computing, 2004

[25] R. DeVaul, M. Sung, J. Gips, A. Pentland, "MIThril 2003: Applications and Architecture", Proceedings IEEE ISWC, October 2003.

[26] http://www2.cs.uregina.ca/~hamilton/courses/831/notes/ml/dtrees/c4.5/tutorial.html

[27] S. Basu, "Conversational Scene Analysis", PhD thesis MIT, Dept. of Electrical Engineering and Computer science, 2002.

[28] http://www.placelab.org/publications/

[29] P. Singh, W. Williams, "LifeNet: a propositional model of ordinary human activity", Proceedings of the Workshop on Distributed and Collaborative Knowledge Capture at K-CAP 2003

[30] P. Singh, T. Lin, E. Mueller, G. Lim, T. Perkins, W. Zhu, "Open Mind Common Sense: Knowledge acquisition from the general public", First International Conference on Ontologies, Databases, and Applications of Semantics for Large Scale Information Systems. Irvine, CA, 2002

[31] http://csc.media.mit.edu

[32] W. Dong, A. Pentland, "Using the Influence Model to Capture System Interactions", submitted to JMLR special topic in learning in large probabilistic environment, June 2005

[33] W. Dong, A. Pentland, "Influence Modeling for Interaction Analysis", The MIT Media Laboratory Technical Report # 589, April 2005

[34] N. Eagle, "Machine Perception and Learning of Complex Social Systems", Ph.D. Thesis MIT, Program in Media Arts and Sciences, June 2005

[35] http://www.cs.waikato.ac.nz/ml/weka/

Appendix A Data Formats

A.1 Files generated by SignalRecorder

All SignalRecorder log-files are in ASCII and carry timestamps. The naming convention is

`sessionName_hostName_hour_minutes_seconds.signalType`

This generated filestem is saved in `outputFile.txt` and can thus be used in pre-processing scripts. The general log-file format is

```
Timestamp1 data data data data …
Timestamp2 data data data data …
Timestamp3 data data data data …
…
```

.audio

The audio signal, which is generated by LinuxAudioSignal, is read using SignalViewer with the terse option. The signal is 16bit at 8kHz and typically a 1024 element vector. Example:

```
1124494208.477360 88 40 –17 6 60 35 –43 –24 … 77
1124494208.732085 139 117 68 75 123 160 92 … 342
1124494208.733533 –27 9 –12 –46 –21 30 –22 … –121
…
```

.accel1/2

The two accelerometer signals are generated by MITeSignal. They are also 16 bit integers (data type only, 10bit are used), sampling rate is 100Hz by specification, but 90Hz is typical. There are two separate signals, again read by SignalViewer in terse format, and thus two log-files. Each sample is on a new line with first X, Y and then Z axis:

```
1124494207.049833 177 287 233
1124494207.059965 178 288 233
1124494207.069824 178 290 232
…
```

.wifi

The WiFi log-file is written by SignalRecorder itself. The format is timestamp, network name, MAC address. If no networks are detected, the output is “none”, if the name is missing, “<no ssid>”

```
1124494752.899091 media lab 802.11 00:60:1D:1D:21:7E
1124494878.446115 <no ssid> 00:20:A6:52:1A:F6
1124494878.446115 MIT 00:01:F4:7B:01:3B
1124494878.446115 MIT 00:01:F4:7B:03:6B
1124495003.027343 none
1124495128.556541 MIT 00:20:A6:52:3E:C0
1124495128.556541 SMARTSIGHT 02:49:2D:CA:6D:B4
…
```

.audioFeat .accelFeat

These are feature log-files, only output when SignalRecorder is running classifiers. They are both created by SignalViewer in terse format, reading from signals generated by AudioFeatures and AccelFeatures, respectively.

.speech.class .speech_c45.class etc.

These are classification files written by GClassifier and C45Classifier, which are both invoked by SignalRecorder, again only in classification mode. The files have the usual format; the data is a probability vector. No labels are written to file, but the classifications can be reconstructed by comparing the index of the maximum probability against the label names in the GMM model XML-file or the c45 names-file used.

.labels

Usually empty. Would contain timestamped labels in case the online labeling screen is used.

.interest

In classification mode, SignalRecorder will output the moments of interest (MoI), which occur when the calculated interest level of the interest prediction algorithm exceeds the preset threshold (per default 5). The format is

timestamp superStates(location speech posture event) currentState(l s p e) interestLevel

```
1073014068.449118 1 0 3 0 0 2 4 0 6.18
1073014153.650418 5 0 4 0 5 2 4 0 5.71
1073014287.239184 1 0 3 0 1 4 3 0 5.62
1073014408.391800 0 2 3 0 5 2 3 0 5.01
…
```

Config_MITes.txt

This file is written by MITesConfig and contains the calibration values for the accelerometers. One line has the format

meanX meanY meanZ OneG_equivalent_X OneG_equivalent_Y OneG_equivalent_Z

The first line is the hip, the second line the wrist accelerometer. Example:

```
235.500000 284.500000 228.999939 60.500000 57.500000 60.999939
295.000000 310.000000 274.500000 60.000000 61.000000 55.500000
```

This file is used in Matlab (and in real-time by AccelFeature) to scale the values according to:

$$X_{normed} = \frac{X_{raw} - X_{mean}}{X_{One\ g\ equivalent}}$$

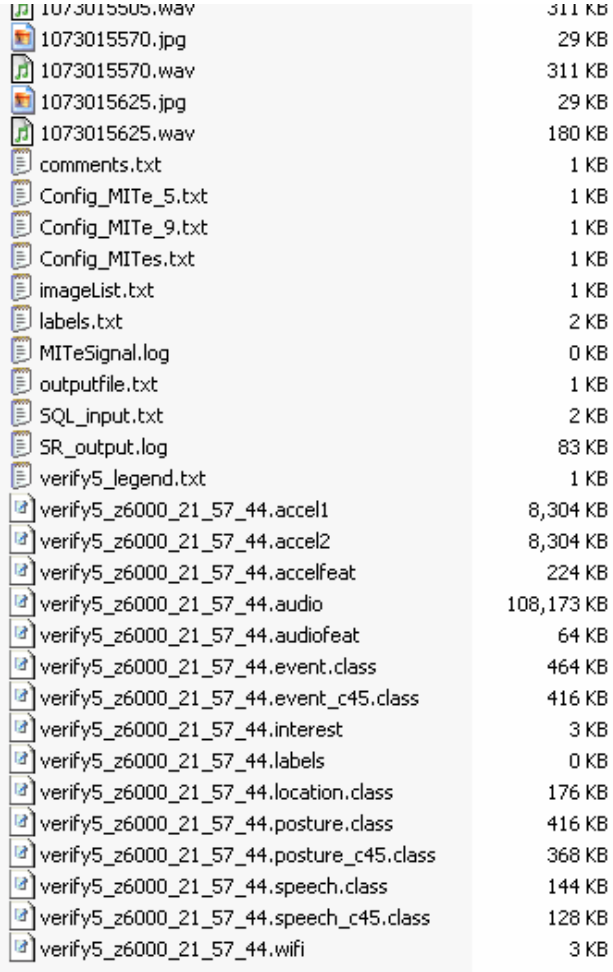


Figure A-1: Example of files generated by SignalRecorder and the annotation tool

A.2 Files related to the offline annotation tool

`SQL_input.txt`

This file is generated by WavClipExtractor and contains a list of jpg- and wav-files. It serves as the input for generating the SQL database table used for annotation. Example:

```
1073012385.jpg,1073012385.wav
1073012445.jpg,1073012445.wav
…
```

`labels.txt, legend.txt`

Generated when labeling is done. The format for the labels is the following:

Timestamp Labels(l s p e) confidence(l s p e) interesting? music?

```
1073014185 3 1 4 0 1 1 1 1 0 0
1073014245 4 6 3 0 1 0 1 1 0 0
```

So in the first line, all labels were confident, in the second line, the speech label was not. It is important to know that the labels written here start at 0. The numbers will correspond to the entries in `legend.txt`. In Matlab indices must start at 1. So there is an offset between the labels in Matlab and in the C-code! This is not very nice and might want to be redefined. Perhaps using whole strings instead of numbers would make it more flexible and easier to read.

comments.txt

The comments entered during annotation end up in this file.

```
1073014425 Running to catch the bus!
```

A.3 Data representation in Matlab

SignalLoader is the tool to convert the data from the format described above into organized Matlab structs, we call them clips. The main features of SignalLoader are described in Appendix E. A clip is defined as a continuous chunk of data that has the same labels. If the annotation was done correctly, the labels should be valid for the entire clip. A clip contains:

- Audio data as a vector in the 2-byte format int16
- The first audio timestamp
- The last audio timestamp
- The audio sampling rate computed from the information above
- Acceleration data in the same form as the .accel files above. However, the numbers might or might not have been scaled using the calibration values.
- If so, the calibration values are added to the clip
- The (smallest) number of acceleration samples
- The filestem string that specifies where the clip came from

```
                label: [7.0192e+003 1 1 2 2 1 1 1 1 0 0]
           audioStart: 7.0193e+003
            audioStop: 7.0390e+003
                audio: [159744x1 int16]
            audioRate: 8.0471e+003
                 acc1: [1739x4 double]
                 acc2: [1749x4 double]
        numAccSamples: 1739
              dataset: 'E:\...\mall1_z6000_12_34_07'
              normMat: [2x6 double]
        audioFeatures: []
 audioFeaturesLocation: [4x11 double]
   audioFeaturesSpeech: [4x11 double]
    accFeaturesPosture: [8x13 double]
      accFeaturesEvent: [8x13 double]
classificationLocation: [4x11 double]
  classificationSpeech: [4x9 double]
 classificationPosture: [8x10 double]
   classificationEvent: [8x11 double]
```

If feature calculation or classification is done, several fields may be added to the clip. Features follow again the format timestamp data data etc. Classifications have the following format, where `p` is a probability vector:

```
timestamp1 p1 p2 ... pn decision label
timestamp2 p1 p2 ... pn decision label
…
```

Appendix B Matlab code

The Matlab code developed is supplied on the CD and is available for download on the MIT Media Lab's wearables page: http://www.media.mit.edu/wearables/code.html

B.1 Main scripts

`CreateTrainingAndTestingClips.m`

Script that loads several data-sets. The data loaded is the base for pre-classification evaluation.

`trainAll_testAll.m`

This script trains and tests all four categories using a Bayes classifier with single Gaussian PDFs. At the end, a test-set for which all categories were "confidently" labeled is pre-classified. It serves as the basis for post-classification analysis.

`trainAll_testAll_withGMMs.m`

Trains and tests all four categories using a Bayes classifier with single Gaussian as well as a mixture of Gaussian PDFs.

`trainAll_testAll_withLDAandGMMs.m`

The same as above, except that the features are transformed using Linear Discriminant Analysis.

`verifyFeaturesAndClassifications.m`

Script that will load all files generated in a SignalRecorder session and verify the features and classifications computed in real-time against the Matlab computations.

`testWithValidationClips.m`

This script loads several cross-validation datasets and classifies them using a previously trained Bayes classifier.

`PreClassify.m`

Applies previously learned Bayes classifiers to previously loaded clips.

`PostClassificationEvaluation.m`

In this script several post-classification approaches are evaluated, including the ones discussed in Chapter 6.

`PostClassify.m`

Contains the best post-classification scheme.

`trainHMMs.m`

Script that trains HMMs for the category activities.

B.2 Functions

`AccelFeature.m`

Computes acceleration features.

`AccFeatures.m`

An earlier version of AccelFeature, which returns the features without timestamps.

`classifyBNET.m`

Old Gaussian Classifier, see GClassifier.

`computeFeatures.m`

Only used for trying out.

`computeLDA.m`

Computes the linear discriminant analysis of features and plots the reduced 3D or 2D feature space.

`Evaluate.m`

Computes accuracy and confusion matrix from a decision / label vector pair.

`EvaluateAllClassifications.m`

A wrapper for EvaluateClassifications.

`EvaluateClassifications.m`

Computes accuracies, confusion matrix and plots of classified clips.

`EvaluatePostClassifications.m`

Computes accuracies of post-classified clips and compares with pre-classification results. See PostClassifier.m for usage.

`export_C45.m`

Creates the files needed (.names .data .test) for the c4.5 program, which only runs in C.

`findLabels.m`

Clip management tool. It shows how many instances and seconds of each label is among a set of clips. It also returns the clip indices for a selected label.

`GClassifier.m`

Naive Bayes classifier with single Gaussian PDF or Gaussian mixture models. Takes features and a bnet and returns a timestamped probability vector.

`myPlot.m`

Helper function to create nice plots.

`playClip.m`

If the clip contains audio, the sound is played using waveplay.

`plotClips.m`

Clip visualization tool. Plots signals, labels and clip origin.

`plotInfluences.m`

Plots a nicely labeled image of an influence matrix or a conditional probability matrix

`SignalLoader.m`

Tool to import data and created labeled clips. There are many options on how this should be done. They are explained in the function header.

`trainBNET.m`

Takes feature vectors and labels to train Bayes classifiers. Single Gaussian observations as well as mixtures of Gaussians are supported using the BNT Toolbox. A faster manual implementation also offers single Gaussian modeling without the toolbox.

`XML_export_GMM.m`

Exports a Gaussian bnet to a file in XML format that plugs directly into GClassifier C-code.

`XML_export_IM.m`

Exports an influence model bnet to a file in XML format that plugs directly into IClassifier C-code.

Appendix C C-Code

The C-code is structured in four modules (four directories on the supplied CD). A description of each module follows, and at the end of the chapter a short tutorial will demonstrate how to compile everything and correctly copy files onto the Zaurus.
The sources can be downloaded from http://www.media.mit.edu/wearables/code.html

C.1 enchant-base

The module enchant-base contains the Enchantment library. It was initially developed by the MIT Wearables Group and is now maintained by a company called AWare Technologies. The library sources are packed in a zip-file, the current version is 0.9.2. The module also includes some other tools, still maintained by the group.

C.2 enchant-dev

The module enchant-dev contains a variety of tools designed for the Zaurus. The ones used are listed here:

`AudioFeatures.cpp`

This code computes the five voice features: energy, spectral entropy, formant frequency, maximal and number of autocorrelation peaks. From those features several statistics are calculated (not verified) as well as voiced/non-voiced and speaking/non-speaking states. In addition the means and variances of these features are computed over BIG_WINDOWSIZE samples. These are the features used for Speech and Location Classification.

`linux_audio.c`

Provides drivers for audio recording and playing.

`LinuxAudioSignal.c`

Generates 16 bit integer audio signals of specified rate using the Linux audio device and the Enchantment signal system.

`MITeSignal.cpp`

This is the serial driver for MITes (MIT environmental sensors), accelerometers developed by the Media Lab group house_n. The MITe-Receiver, which is connected to the serial port, is configured to listen on the channels requested. Acceleration values are posted as Enchantment Signals

`MITeCalibrate.cpp`

This Program is used to calibrate accelerometers. Input are one or two (hardcoded) acceleration signals, output is a text-file containing calibration values.

C.3 inference

The module inference contains programs and classes for context recognition.

`AccelFeature.c`

Program that computes features from acceleration signals. Supported feature types are mean, variance, 64-Window FFT.

`C45Classifier.c`

This is the Enchantment interface to the external c4.5 code. It takes care of reading feature signals, consulting the c4.5 functions, and posting the results as signals or nodes on the whiteboard. File I/O is also supported for offline use.

`Chain.cpp`

Class that models a chain of the influence model. Includes XML parsing.

`Classification.cpp`

The Classification class is used by GClassifier. It contains functionality to manage labeled probability vectors.

`Gaussian.cpp`

The Gaussian class is used by MixtureModel. It models a Gaussian probability density function. Includes XML parsing.

`GClassifier.cpp`

This program implements a Bayes classifier with single Gaussian PDF. It provides the Enchantment interface to the Gaussian classes, takes care of reading feature signals, calling classification functions, and posting the results as signals or nodes on the whiteboard. File I/O is also supported for offline use.

`hd_influence.c`

This is the implementation of the influence model, based on the Gnu Scientific Library (GSL). It serves as a library to IClassifier.

`InfluenceModel.cpp`

The class models the influence model, used by IClassifier. It includes XML parsing.

`MixtureModel.cpp`

The MixtureModel class is used by GClassifier. It contains several Gaussians. Includes XML parsing.

`psc.c`

This is an example on how to use the influence model with continuous observations. Includes reference code for Matlab

`psd.c`

This is an example on how to use the influence model with discrete observations. Includes reference code for Matlab

`XMLObject.cpp`

Implementation of an XML parser, based on the 'expat' library.

C.4 SignalRecorder

The module qtopia/SignalRecorder contains the GUI application, drivers for the camera and some programs/scripts for processing the log-files before annotation.

`CameraIO.cpp, CameraIO.h`

Driver for the CF-card camera.

`Capture.cpp, Capture.h`

Program that offers a command line interface to the CF-card camera.

`linux_audio.c, linux_audio.h`

The same as in the module inference. Used with SignalRecorder to play sound upon request.

`make_all.sh`

`Makefile.Capture, Capture.pro`

`Makefile.SignalRecorder, SignalRecorder.pro`

`Makefile.WavClipExtractor`

There are three make files in this module. The SignalRecorder Makefile is quite complex and needs the .pro file. It needs to link several Qtopia libraries. Capture is less complex, but also depends on some Qtopia libraries, thus the same type of Makefile. WavClipExtractor only needs basic gcc. Make_all.sh takes care of compiling all at once.

`sharp_char.h`

Includes certain definitions for Zaurus drivers.

`SignalRecorder.cpp, SignalRecorder.h`

These are the main files of SignalRecorder. The main features are:

- Connections to the GUI
- Starting and stopping recording as well as starting and stopping classifiers: Upon request it will launch required processes with system calls and will kill them on termination.
- Sensor file size, battery and CPU logging
- Managing sensor calibration using MITeCalibrate
- Manages picture taking
- Polls the shutter button if requested
- Runs the WiFi sniffing procedure
- Manages online labeling
- Computes the interest prediction algorithm and logs moments of interest

`SignalRecorder.desktop, SignalRecorder.png`

Files needed for the application icon in Qtopia.

`main.cpp`

Runs the actual SignalRecorder application. Do not touch.

`SignalRecorder_base.ui`

This file contains the graphics. It could be edited by hand, but better using QtDesigner. SignalRecorder_base.ui.z5500 is for the Zaurus 5500, previously used.

`sigrec_stop.sh`

Script that kills all processes SignalRecorder started (in case it should fail to do so itself).

`WavClipExtractor.c`

Program that extracts 20-second wav-clips from a SignalRecorder audio log-file at times corresponding to the timestamps of jpg-images in the current directory.

`preProcessing`

This directory contains script for pre-processing the SignalRecorder log-files before loading them in the offline annotation tool or in Matlab.

Appendix D Offline Annotation Tool

This section features a technical description of the offline annotation tool. It is implemented in PHP/HTML and connects to a MySQL server. The Software package used is XAMPP for Windows Version 1.4.13 (http://www.apachefriends.org/en/xampp.html). Currently, this environment is set up on moria.media.mit.edu in C:\apachefriends\xampp. The directory htdocs builds the root of the server.

After extracting audio clips with `WavClipExtractor`, a file called `SQL_input.txt` should have been created. It contains a list of the form

```
1124385421.jpg,1124385421.wav
1124385479.jpg,1124385479.wav
…
```

The file `insert.php` takes `SQL_input.txt`, a local path and a session name e.g. *mySession* to create an annotation session. The SQL database used is called *annotate*. `Open_db.php` establishes the connection. In the table called *sessions* a new entry is made with session name and local path. Then, a new table is created, which is named after the session name, here *mySession*. `insert.php` uses a text-file import function to load each row of `SQL_input.txt` as a row in the new table. The session has been created.

The file `annotate.php` takes care of the labeling. If no session name is passed as an argument, the user is asked to specify one. Then, the first clip is presented. The jpg and the wav-filenames are loaded from *mySession* and are appended to the local path, which is loaded from the *sessions* table. The user can start labeling by selecting the radio buttons and checkboxes. On *Insert Label,* `annotate.php` will write the values of the radio buttons and checkboxes as integers into the table *mySession*, as well as the comment line. The next clip is loaded and, if it was not yet labeled, the previous selections are repeated, assuming that very likely the next clip can be labeled the same. However, if the clip already contains a label, the values are read from *mySession* and displayed on the web-page and the caption of *insert label* changes to *overwrite label*. The navigation buttons on the left do not write anything into the table, but will make `annotate.php` show the corresponding clip with the stored label if applicable.

Once all clips are labeled, or if *generate output* is clicked, three text-files will be generated, saved on the server in the directory *output/mySession* and presented for download. `Labels.txt` contains a timestamp and the labels as integers (see Appendix A for the format), `comments.txt` contains a timestamp and the comment line and `mySession_Legend.txt` is a copy of `currentLegend.txt`, which contains a mapping of the integer values to the label strings and is located on the server. This file should be updated in case any label names are changed in `annotate.php` or else there will be confusion in a later stage.
Using integers as labels is easier from a programming perspective because they can be stored as matrices, are easily imported to Matlab and there is no ambiguity due to capital letters or white-spaces. But the need for consulting a legend and always thinking in offsets is quite a burden for the user. This decision might want to be thought over.

The file `replace.php` can be used to replace the label entries of a particular session with new ones located in a text-file with the correct format. This procedure can be useful if some labels change their names or if new categories are created. `Labels.txt` can then be edited in Excel for example.

XAMPP offers many nice tools for server and database management. They are accessed from the server root, with user name *borglab* (password protected). The user name for phpMyAdmin is *root*.

Appendix E
Data collection and annotation tutorial

E.1 SignalRecorder

This is a Qtopia application that offers a GUI to manage the data acquisition. It is typically located on the Zaurus in /mnt/card/borglab/SignalRecorder and can be started from command line or from the desktop icon. On the first tab, a name can be entered which will be pre-pended to all output files. With checkboxes the sensors are selected. The Numbers next to the MITe checkboxes are their channel IDs, needed for the lower level protocol. If the WiFi box is checked, the network sniffing process will be included. If the camera box is checked, pictures are taken every minute automatically.

SignalRecorder takes care of starting all sub-processes needed for reading the sensors (e.g. LinuxAudioSignal and MITeSignal). The signals created by the latter are logged using SignalViewer in terse mode. Each signal is logged in a separate file, which is of the format
Timestamp data1 data2 data3 … e.g. 1075321569.123456 246 654 847.
The files are "rectangular" and can thus be imported as a matrix into Matlab directly using the load() command.

After pressing start, a prompt will pop up and ask for the MITes to be calibrated. After clicking 'ok' a calibration process is launched, which will run for 30 seconds and then present the results. During these 30 seconds the user must gently rotate all MITes in all directions. Each of the six perpendicular orientations (two extreme positions for all three axes) must be maintained for at least 3 seconds!

Then, the application switches to the Annotation tab. Here you can click the radio-buttons to set labels online. The result is a timestamp and a label number each time a new button is selected. With the "take foto" button you can capture an image, which will be saved in the format 'timestamp.jpg'. This button works regardless whether the camera checkbox on the previous screen was checked. The third tab monitors the sensors. It does that by periodically logging the size of all output files. If in two consecutive readings the file size has not increased, the corresponding sensor will turn from green to red until the file size increases again. Also a status LED (the second from the right) indicates SignalRecorder is running and the sensors are fine.

Then there is the pre-classification tab. Start and stop buttons take care of launching and killing feature generation and classification processes. If classifiers are running, SignalRecorder will display their outputs once per second. If the WiFi box was checked, this tab will also show the wireless networks detected. If an entry is found in the look-up table, the corresponding location is displayed. The classification processes use the signals created by the sensor reading processes, but they don't interfere with data recording.

When exiting or stopping the recording, SignalRecorder will kill all processes previously started. This usually works fine. But it can for unknown reasons happen, that some processes don't terminate. It is then possible that such processes fill up the SD-card, which will cause the Zaurus to slow down and possibly crash. So one should make sure there are no unwanted processes running before starting a serious data acquisition session.

E.2 Copying the data to disk

All output data is saved in /mnt/card/borglab/SignalRecorder/data. Downloading data of only a few minutes can be done using the Zaurus' WiFi-connection. For large files using an SD-card reader or an ethernet-connection with a Zaurus CF-card is more appropriate.

The data directory currently used is on the workstation esgaroth: /localdata/mark/Name. After downloading, the data should be removed from the SD-card to make space for the next user. This list contains an example of the files that should be downloaded:

- `Name_z6000_08_32_57.audio`
- `Name_z6000_08_32_57.accel1`
- `Name_z6000_08_32_57.accel2`
- `Name_z6000_08_32_57.labels`
- `10730154119.jpg`
- `10730154179.jpg`
- `10730154239.jpg`
- etc.

E.3 Creating the Wave Clips

The annotation software will play a 20 second audio clip to each recorded image. These clips need to be created using WavClipExtractor on a Linux station. Copy WavClipExtractor, your audio file and all images into a directory. Run `./WavClipExtractor –a Name_12_32_45.audio –t 20` to create 20 second audio clips for all .jpg-files in the directory. If only a specific number of clips shall be extracted use the option `–i imageList.txt`.
A comma-separated output file `SQL_input.txt` is generated containing a simple list of all images and wav-files.

E.4 Offline Annotation Software

The offline post-annotation is done using a web-interface with PHP and MySQL. It is located in moria.media.mit.edu/borglab (IP: `18.85.18.52)`. After creating a session the annotation tool will, clip by clip, load the image, play the audio-clip and present several sets of radio-buttons and a comment line, which are used to label the clip.
For privacy reasons, the images and audio-clips remain on the users local workstation at all times.
On the first page, `insert.php` will use `SQL_input.txt`, a specified path for the directory on your local machine and a session name to create an entry in the MySQL table 'sessions' and to create a new table in which each row represents a clip.
This means that the annotation process keeps state. All labels are stored in the session's table and a user may resume a terminated session.
Once all clips are labelled the text-files `labels.txt`, `comments.txt` and `Name_Legend.txt` are created and presented to download.

On esgaroth:/localdata/mark/ there is a script "preProcessData.sh" that will do all necessary pre-processing:

```
echo "preProcessData: extracting file name from outputfile.txt:"
export fileName=`cat outputfile.txt`
echo $fileName
if [ "$fileName" == "" ]; then
echo "preProcessData: Error, could not find outputfile.txt"
exit –1
fi

cp /localdata/mark/audio2wave.pl .
cp /localdata/mark/ascii2bin .

echo "preProcessData: Cropping files (wait 1 min)..."
/localdata/mark/Cropfile $fileName.accel1
/localdata/mark/Cropfile $fileName.accel2
/localdata/mark/Cropfile $fileName.audio
```

```
/localdata/mark/Cropfile $fileName.accelfeat
/localdata/mark/Cropfile $fileName.audiofeat
/localdata/mark/Cropfile $fileName.event.class
/localdata/mark/Cropfile $fileName.posture.class
/localdata/mark/Cropfile $fileName.location.class
/localdata/mark/Cropfile $fileName.speech.class
/localdata/mark/Cropfile $fileName.posture_c45.class
/localdata/mark/Cropfile $fileName.event_c45.class
/localdata/mark/Cropfile $fileName.speech_c45.class

echo "preProcessData: Calling WavClipExtractor (wait 2 min)..."
/localdata/mark/WavClipExtractor -a $fileName.audio

rm -f audio2wave.pl
rm -f ascii2bin

echo "preProcessData: finished!"
```

Appendix F
Detailed comparison of four microphones

In a test recording a text was read out loud and four different microphones were used.

1. the built-in microphone of the Zaurus SL6000L
2. a dynamic hands-free cell phone microphone
3. RadioShack 33-3013 omnidirectional electret condenser microphone with LR44 1.5V alkaline button cell battery
4. Sony ECM-T115 omnidirectional electret condenser microphone with CR2022 3V lithium cell battery

The spectrograms and FFTs show differences in quality. In contrast to the condenser microphones, the built-in and the dynamic microphone show disturbing frequencies below 100Hz. This is likely to interfere with certain audio feature computations e.g. formant frequency. The cell-phone microphone shows a very large disturbance at 60Hz. This is probably due to the frequency of the AC power supply that was plugged in during the recordings.

All recordings show a disturbance at 1952Hz, which must be attributed to the AD converter of the Zaurus.

An important characteristic lies in the higher frequencies when it comes to detecting the number of autocorrelation peaks, which is one of the used audio features. Because those peaks were best visual in the fourth recording, the Sony condenser microphone was selected.

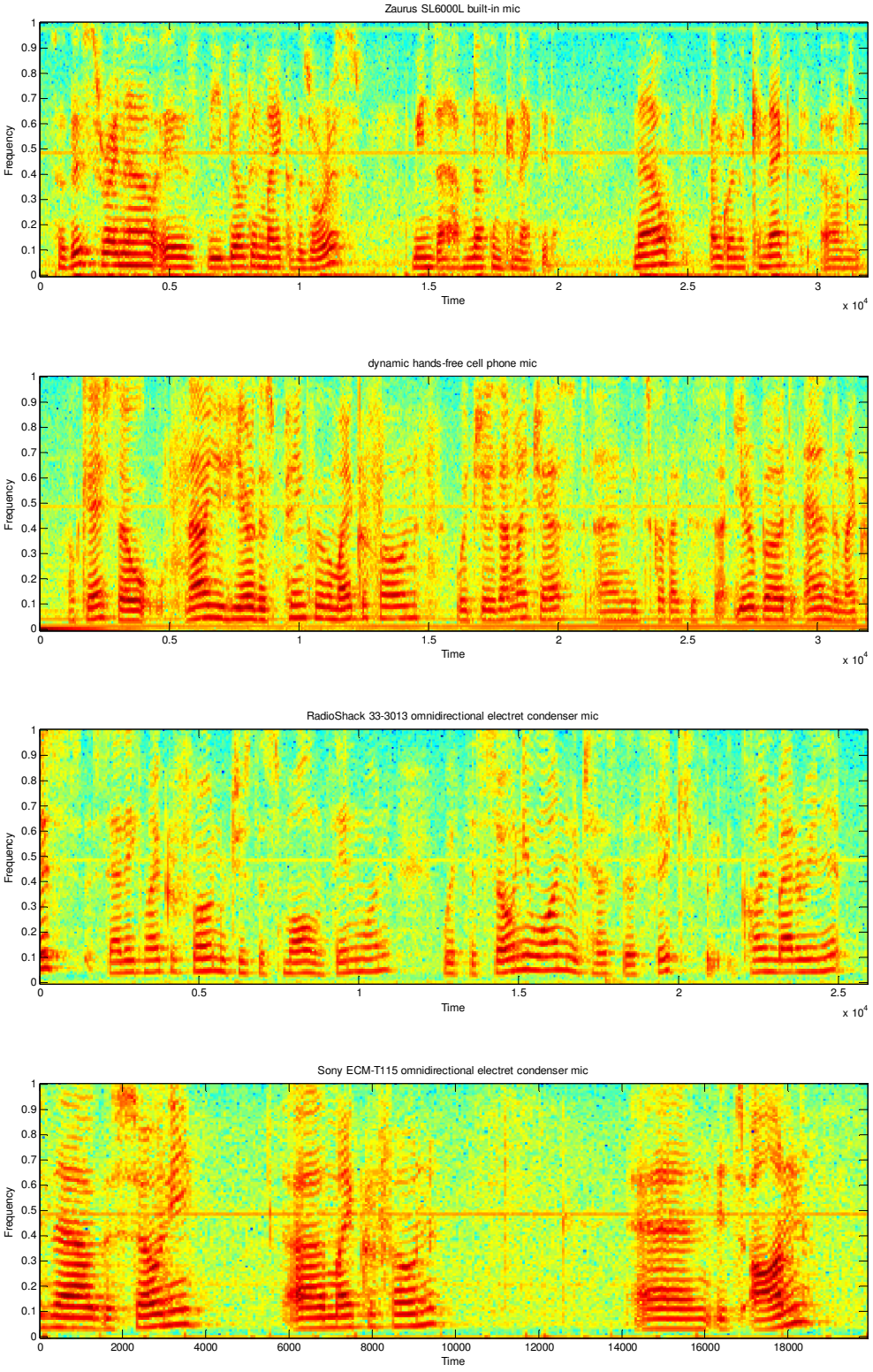


Figure F-2: Comparison of spectrograms using four different microphones

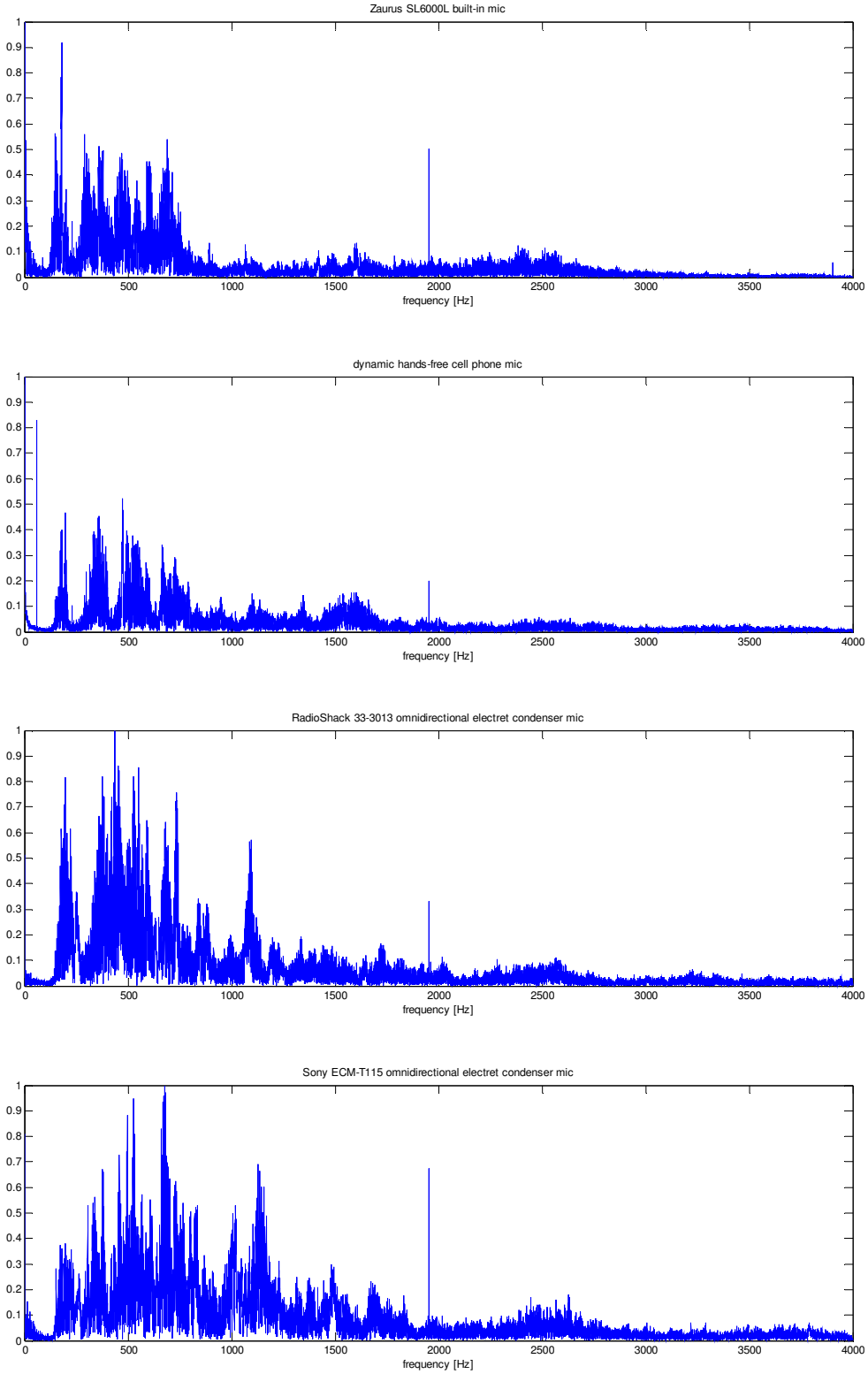


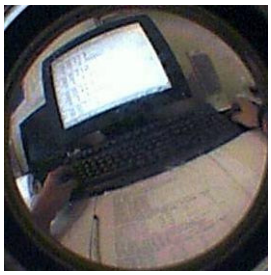
Figure F-3: Comparison of FFTs using four different microphones

Appendix G Interest prediction experiment

These are the pictures recorded in the 3-hour interest prediction experiment described in Chapter 7. The images of set A were taken by the interest prediction algorithm. In set B, pictures were taken at regular intervals.

G.1 Picture set A

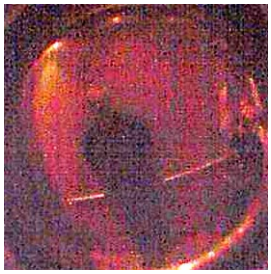

1


2


3


4


5


6


7


8


9


10


11


12


13


14


15


16


17


18


19


20


21


22


23


24


25


26


27


28


29


30


31


32


33


34


35


36


37


38


39


40


41


42


43


44


45


46


47


48


49


50


51


52


53


54


55


56


57


58


59


60


61


62


63


64


65


66


67

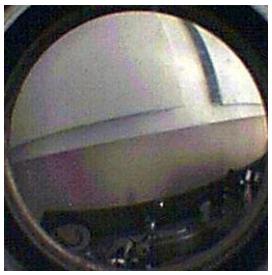

68

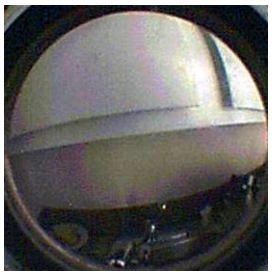

69


70


71


72


73


74


75


76


77

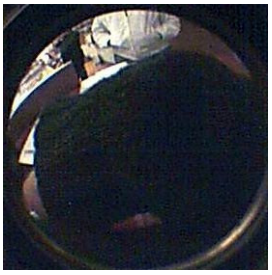

78


79


80

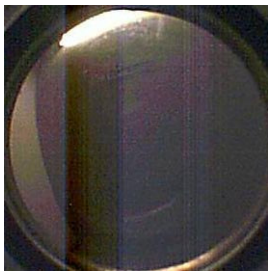

81


82


83


84


85

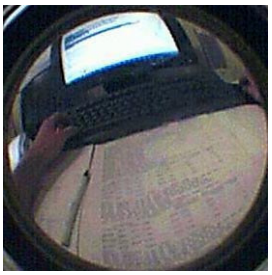

86

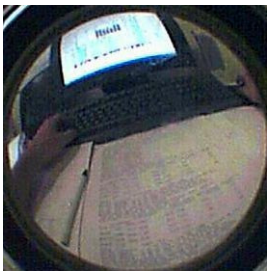

87


88


89


90


91


92


93


94


95


96

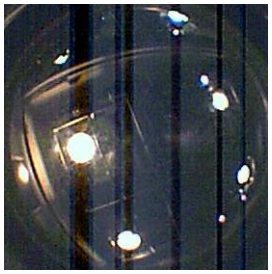

97


98


99


100


101


102


103


104


105


106


107


108


109


110


111


112


113

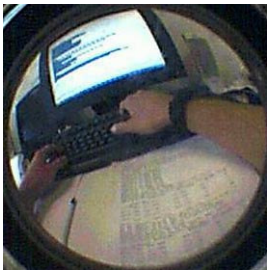

114

G.2 Picture set B


1


2


3


4


5

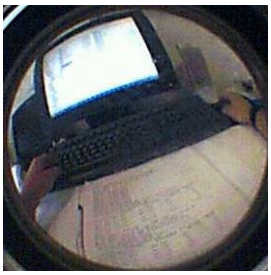

6

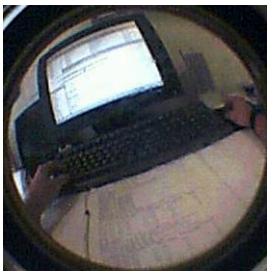

7


8


9


10


11


12


13


14


15


16


17


18


19


20


21


22


23


24


25


26


27


28


29


30


31


32


33


34


35


36


37


38


39


40


41


42


43


44


45


46


47


48


49


50


51


52


53

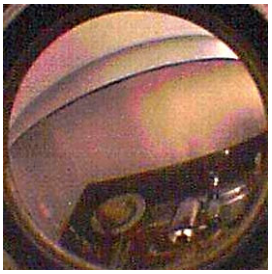

54


55


56


57


58


59


60


61


62


63


64


65


66


67


68


69

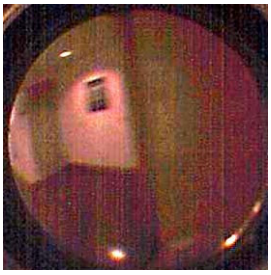

70


71


72


73


74


75


76


77


78

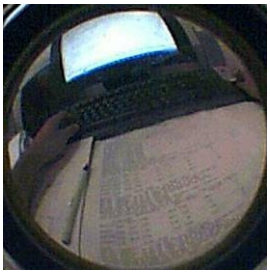

79


80

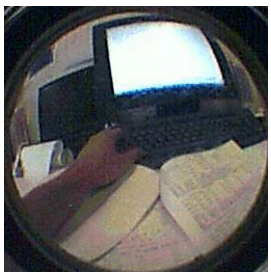

81

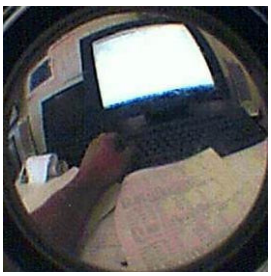

82


83


84


85

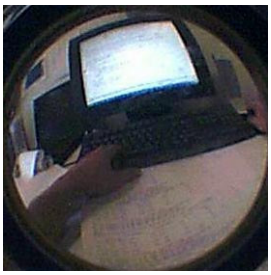

86

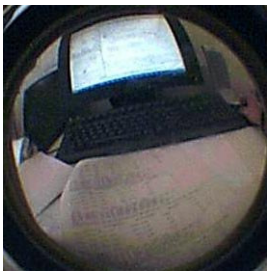

87


88


89


90


91


92


93


94


95


96


97


98


99


100


101


102


103

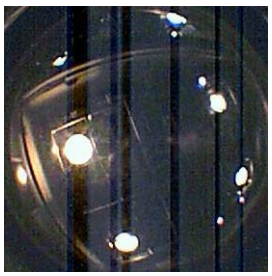

104

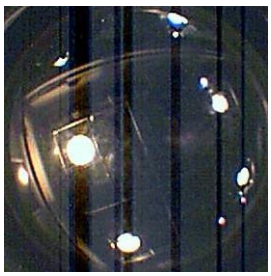

105

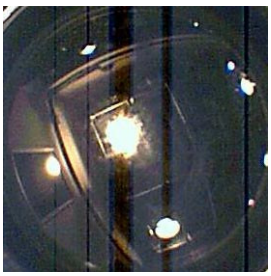

106

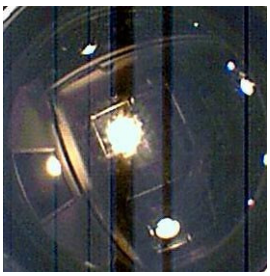

107

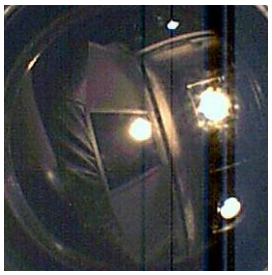

108


109


110


111


112


113


114


115


116


117