Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Laura Hendrikje Friderike Knetter

Identification and logging of alarm records through feature detection based on MPEG7 Low-Level-Audio-Descriptors

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Laura Hendrikje Friderike Knetter

Identification and logging of alarm records through feature detection based on MPEG7 Low-Level-Audio-Descriptors

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Technische Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. rer. nat. Wolfgang Fohl
Zweitgutachter: Prof. Dr. Ing. Andreas Meisel

Eingereicht am 22. März 2012

Thema der Bachelorarbeit

Identifizierung und Protokollierung von Lautsprecher-Warndurchsagen mittels Merkmalsextraktion auf der Basis von MPEG7 Low-Level-Audio-Deskriptoren

Stichworte

Audio, digitale Signalverarbeitung, MPEG7, Low-Level Deskriptor, Pattern-Matching

Kurzzusammenfassung

DESY konstruiert und baut zur Zeit einen neuen Linear-Teilchenbeschleuniger in Hamburg, den European XFEL. Daher wurden die Sicherheitsvorkehrungen für alle Beschleuniger überprüft. Es wird als notwendig empfunden eine Audio Erkennung für die Warndurchsagen zu den bestehenden Sicherheitssystemen hinzuzufügen.

Die Audio Erkennung wird in dieser Arbeit als Machbarkeitsstudie einer Erkennung auf Basis von Low Level Audio Deskriptoren aus dem MPEG7 Standard getestet. Somit erhält man eine kompaktere Spracherkennung als beispielsweise mit einem Algorithmus basierend auf einem Hidden Markov Modell.

Title of the paper

Identification and logging of alarm records through feature detection based on MPEG7 Low-Level-Audio-Descriptors

Keywords

Audio, digital signal processing, MPEG7, Low-Level Descriptor, pattern matching

Abstract

DESY is currently building a new linear accelerator, the European XFEL. Due to this event safety precautions for all accelerators were revised and the need of an audio recognition for the warning signals was stated.

This work is testing the possibility of a low level speech recognition for the warnings based on MPEG7 Low Level Audio Descriptors.

Acknowledgements

I would like to thank a number of people who supported me during this work. First of all my supervisor Dr. Wolfgang Fohl for his patience, advice, assistance and input of new ideas.
Especially many thanks to Stefan May for an extraordinary morning by showing me the accelerator and to the rest of the DESY team for providing the topic.
Furthermore a lot of thanks to all my fellow students for the supporting help over the last three and a half years.
And last, but definitely not least, thanks a lot to my father for laying the key interest in computer science, my mother for trying to understand what I am doing and my aunt and uncle for all the support they gave.

Contents

1 Introduction

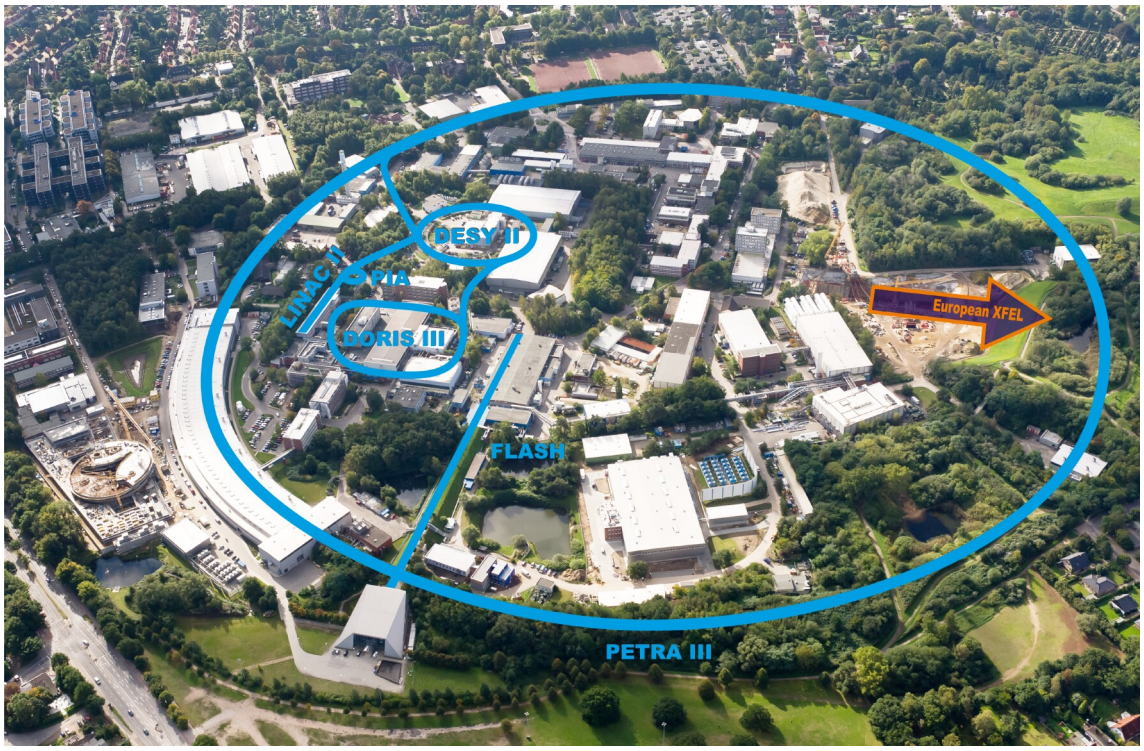

Figure 1: Air photograph of DESY grounds with particle accelerators (2010), Source: [DES12]

The DESY [DES12] - Deutsches Elektronen-Synchrotron - is a national research center, member of the Helmholtz Association, which investigates the structure of matter by developing and using accelerators and detectors for photon science and particle physics.[DES09] It is mainly located in Hamburg-Bahrenfeld as shown above 1 and at the moment a new linear accelerator, the European XFEL [Gmb12], is built.

This project led to a consideration of the safety precautions build into the accelerators with the result that an audio recognition for the warning signals shall be build.
As the progress of the construction of the accelerator is today just so far that the 3km tunnel for the accelerator is nearly bored through and the completion date set to 2015 it is a clear fact that precise hardware restrictions and specifications are not yet available.
Therefore this work is a feasibility analysis whether it is possible to develop a lightweight speech classification system for recognition of warning signals.

Figure 2: Tunnel XTL, European XFEL, on 06.12.2011, Source: [Gmb12]

Physical experiments on atomic level usually provide a certain danger for a human body. Therefore it has to be ensured that a decent safety distance to the actual experiment area is observed.
The motivation of this work occurs from the pictured circumstances: It is to improve the existing safety equipment in and around the accelerators by adding an audio recognition to the established procedures.
Between two physical experiments in the accelerator maintenance work and preparation must be done. Therefore people are inside it. Those people have to get out of the accelerator before an experiment is started, because otherwise they would get (more or less severe) injured.
Before a experiment a searching troupe walks through the tunnel looking after that people get outside.
After this searching troupe a automatic door is activated, once activated ensures that you can only can get back into the tunnel with a special key. Where when one of those keys (there are six.) is missing, the accelerator cannot be switched on.
Inside the accelerator tunnel every twenty meters there are bright shining warning lamps and a emergency switch off button which holds the whole procedure.

The focal points of this work are:

- The development of an algorithm to recognize warning signals
- its testing
- and its possible verification.

2 Outline of The Problem

2.1 Related Work

Much research on speech recognition in various fields of applications and on digital signal processing in reference to audio has been done. [And04, Dim08, RT10, web11, Zöl02, Zöl96] This led to the conclusion that the world of speech recognition has a very wide range. It spans from more or less successful recognizing normal everyday language on one edge, as the implied personal assistant 'Siri'[Inc12] from Apple is intended to be, to using a fixed amount of commands on the other edge, like some newer TV devices use.

As until today no generic efficient algorithm which covers most usual scenarios of speech recognition exists. Therefore for each problem it is necessary to develop a different solution with the help of previous ones out of this wide range.
The advantage of this particular scenario is that it not only has a fixed and small amount of signals which can be equaled to 'commands', but that they are always delivered with the same voice. This makes the algorithm rarely vulnerable to timbral effects.

2.2 Specific Solution

Considering the small amount of signals which have to be recognized it is obvious from the beginning on that no big speech recognition, as an algorithm based on Hidden-Markov-Models [Wik12] for example would be, is needed.
Still it has to be a reliable and nearly 100% correct model in order to be used in such a crucial environment.

Since later real physical conditions might impact the signals it is necessary to take some kind of metadata of the audio signals and not the actual signal to avoid greater influences caused by interferences on the signal such as the audio volume or any kind of noise.
That is where the MPEG7 audio descriptors come to account, because with them it is easier to for example ignore volume differences.
It has to be ensured that the chosen idea to take MPEG7 Descriptors as a basis for the recognition is likely to fit the circumstances. Therefore it is possible to first try an upcoming algorithm on the whole signals. For the real application later it is needed to have a streaming input, so that the system is able to respond in about a second from the beginning of the warning, rather then waiting ten seconds until it is finished and can be processed then.

3 Basic Background Knowledge

In this chapter some basic facts will be explained in order to provide a certain knowledge to understand the development. They will be referred to in the following chapters.

3.1 Provided Audio Signals

All audio signals that are provided from the DESY come in .wav-format and have a sample rate of 48 kHz and a duration of approximately 10 seconds.
Generally there are two types of warning signals. One type is played before a searching troupe will get into the accelerator to get all people who were doing maintenance work or some kind of preparation out of it, e.g. *'Interlock-Absuche. Bitte verlassen Sie das Experimente-Gebiet. Interlock search. Please leave the experimental area.'* . The other one is played immediately prior to the beginning of an actual physical experiment, e.g. *'Achtung! Petra wird eingeschaltet. Attention! Petra will be switched on.'* . After this warning an experiment can still be cancelled by a person within the tunnel in pressing one of the emergency off buttons.
The second type of those warning signals is played much louder than the first type and can hardly be failed to hear, even from outside the accelerators hall.
An overview of those signals is provided in appendix A.

For testing purposes there is another set of signals included (Recognizable by the prefix 'RR_' in the filename). They are basically the same as the original signals from DESY, but played back via home stereo equipment and re-recorded with a professional-class condenser microphone. These signals provide a impression of what interference could do to the recognition.
To test the reliability of the streaming input there need to be created more test signals, e.g. with a slightly shifted start point (added zero samples to the begin of the file) or a signal with music instead of speech or a different speech signal. Each of the last two prepared to fit in the silence scheme of a specific reference signal.

3.2 MPEG7

3.2.1 General

MPEG7 - formally known as "Multimedia Content Description Interface" - is a general standard (ISO/IEC FDIS 15938-4:2001) to describe multimedia data [Mar12]. It can be used just for visual content (such as pictures), for audio content (speech and music) or for both combined as in films. Mainly used here is the audio part, but first some general aspects. MPEG7, created by MPEG [MPE], Moving Picture Experts Group, is part of a large group

of standards. It consists of different parts which together form a metadata describing system. These parts are: Description Schemes, Descriptors, Datatypes, Description Definition Language and System Tools. Next the focus will be drawn on some special descriptors. For further information on MPEG7 in general consider the provided literature at the end of the work, e.g. [SS02], [Pee04] or [MPE].
As the MPEG7 standard is an interface no actual implementation is given. A Matlab library which will be explained later on is used.

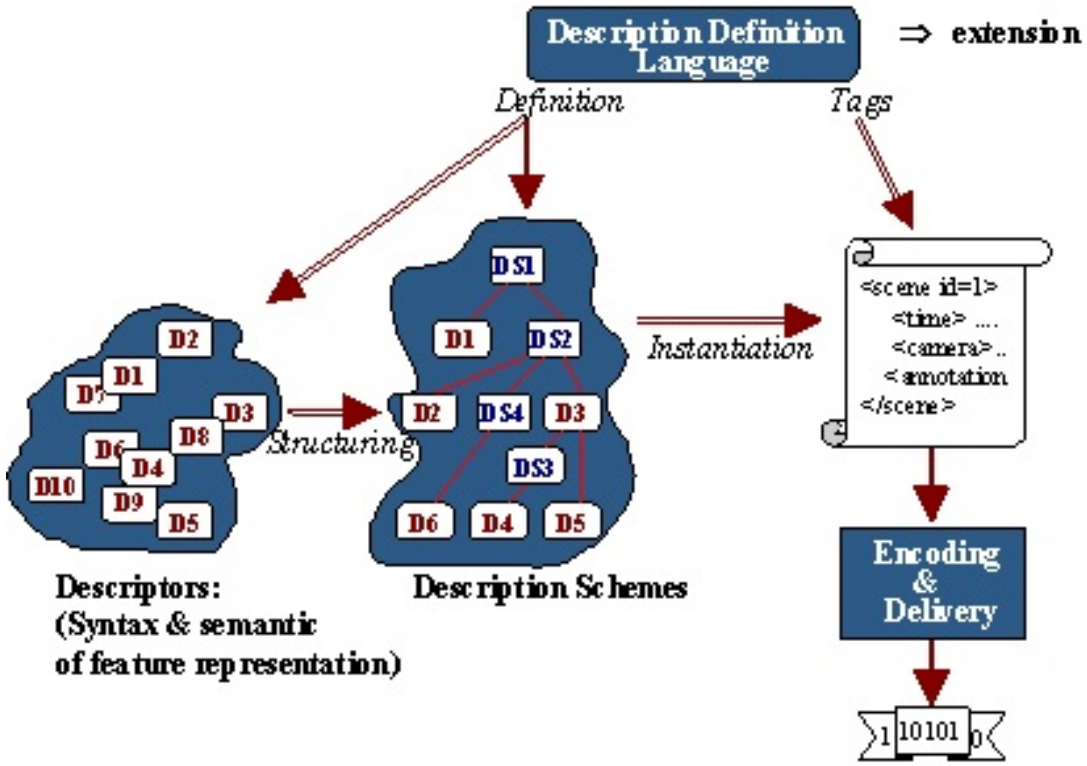

Figure 3: Overview of MPEG 7 generally, Source: [Mar12]

This diagram 3 shows the main elements of MPEG7. The Low-Level Audio Descriptors refer to the Descriptor area seen on the left side.

3.3 Low Level Audio Descriptors

3.3.1 Applicable to All Low Level Audio Descriptors

Low Level Audio Descriptors are descriptors based on the waveform or the spectrum of an audio signal and concentrate on different special aspects of those. Therefore they take the physical shape of the signal in concern and no metadata.

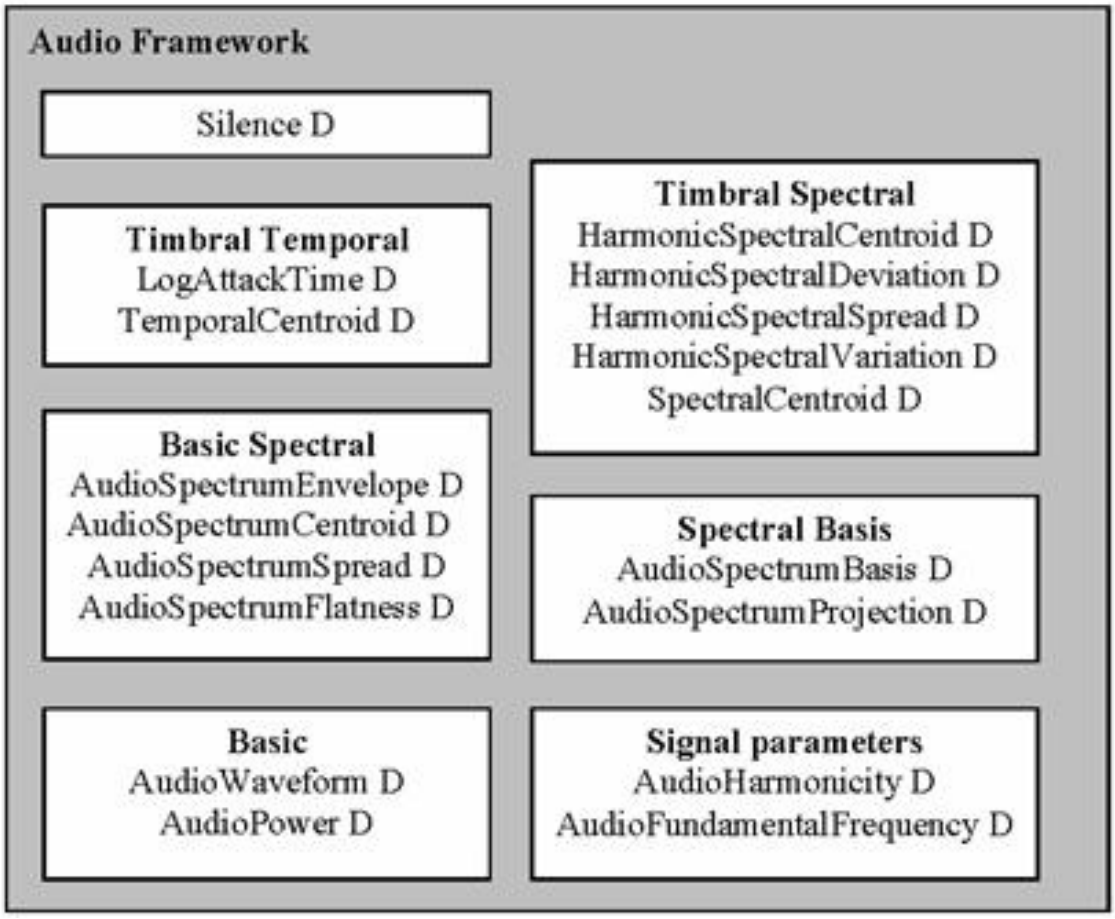

Figure 4: Overview of MPEG 7 audio framework, Source: [Mar12]

Figure 4 gives an overview about the audio framework of MPEG7 and its audio descriptors divided into logical categories. All of these descriptors can be found in the ALL-XM library [Cas04].

3.3.2 Descriptors Especially in Use Here (ASE, SilenceD, Spectrum Centroid)

- ASE

  The Audio Spectrum Envelope Descriptor is based on the power spectrum of an audio signal. It uses a logarithmic frequency axis. It is useful to create a spectrogram or as general-purpose descriptor in comparison.
  It shows a very compact description of the signal's spectral content and mirrors the approximately logarithmic response of the human ear.
  This is done by a time-series of logarithmic, sub-band descriptions of the short-term audio signal spectrum, cf. [SS02].
  Mostly it is displayed in logarithmic scale so it can be referred as decibel.
  Here it is used as a base descriptor for the recognition and for the Silence Descriptor.
  Graphic 5 shows an example audio signal (Petra warning) in waveform, its spectrogram and its Audio Spectrum Envelope, calculated by the descriptor of the library [Cas04].

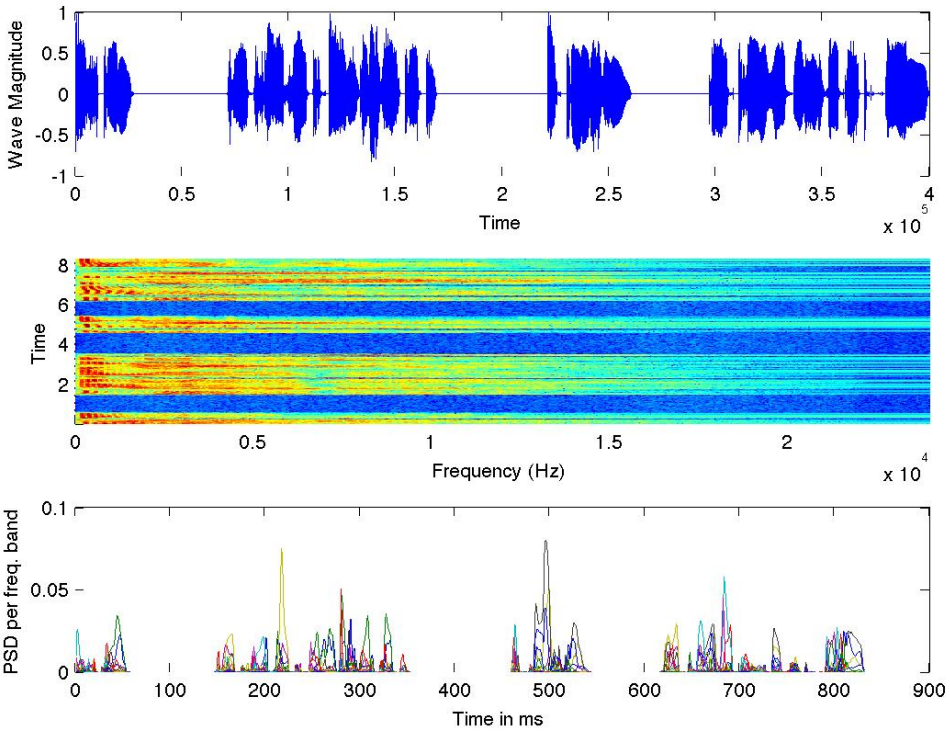

Figure 5: An example audio signal, its spectrogram and its corresponding Audio Spectrum Envelope

- SilenceD

  By description the most basic descriptor of all. It shows where an audio signal has a significant amount of no (recognizable) sound in a time period. Using the Silence Descriptor one gets to the physical structure of the given audio signal and it can be used for segmentation. So each silence segment shall have a start time and a duration.
  That part was a bit changed for the convenience of fitting into the rest of the recognition algorithm. Basically this implementation of the Silence Descriptor shows if there is silence in every time block of the audio signal used on which is shown in the graphic below 6. For reading convenience not the actual audio signal is shown as a reference, but the average spectral magnitude of ASE which is just a different way to describe an audio signal and which happens to be in the same time span as the Silence Descriptor. Concrete information is provided in the .m-file added to the work, see in appendix B.

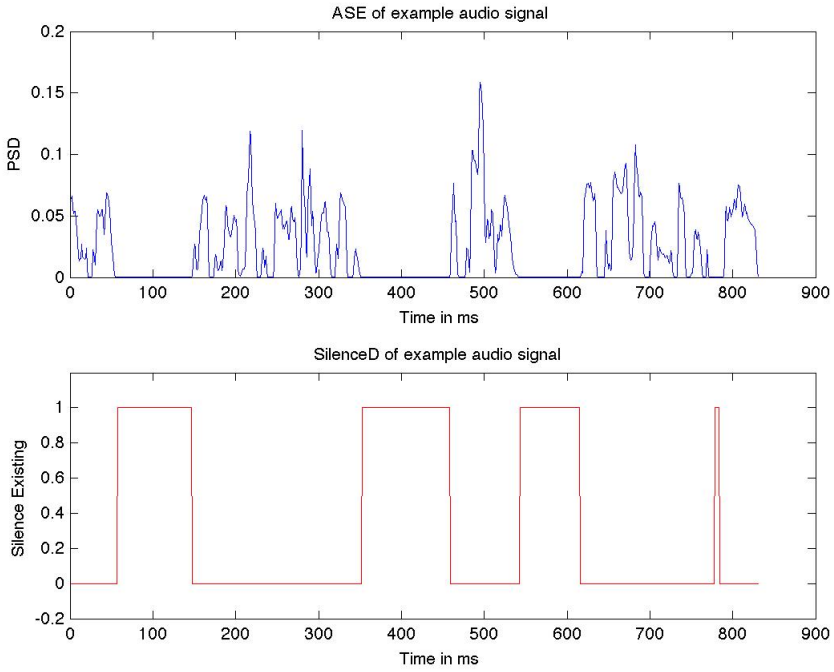

Figure 6: Average spectral magnitude of ASE and corresponding Silence Descriptor of the example audio signal

- Spectrum Centroid

  It describes the center of gravity of the log-frequency of a power spectrum. It gives a measure, if the signal is dominated by high or low frequencies.
  During the development process it was shown that even silence has a spectrum centroid, which is considerable when thinking of silence as described above. The problem of this is that this centroid values lies within an area where they could be also evolved of an actual audio signal. This can be seen in the graphic 7 below. Which is why a little advanced version of the descriptor is used where the blocks where there is silence (referring to the Silence Descriptor) are set to zero. That later leads to a better result when cross correlating the values of this descriptor of different audio signals.

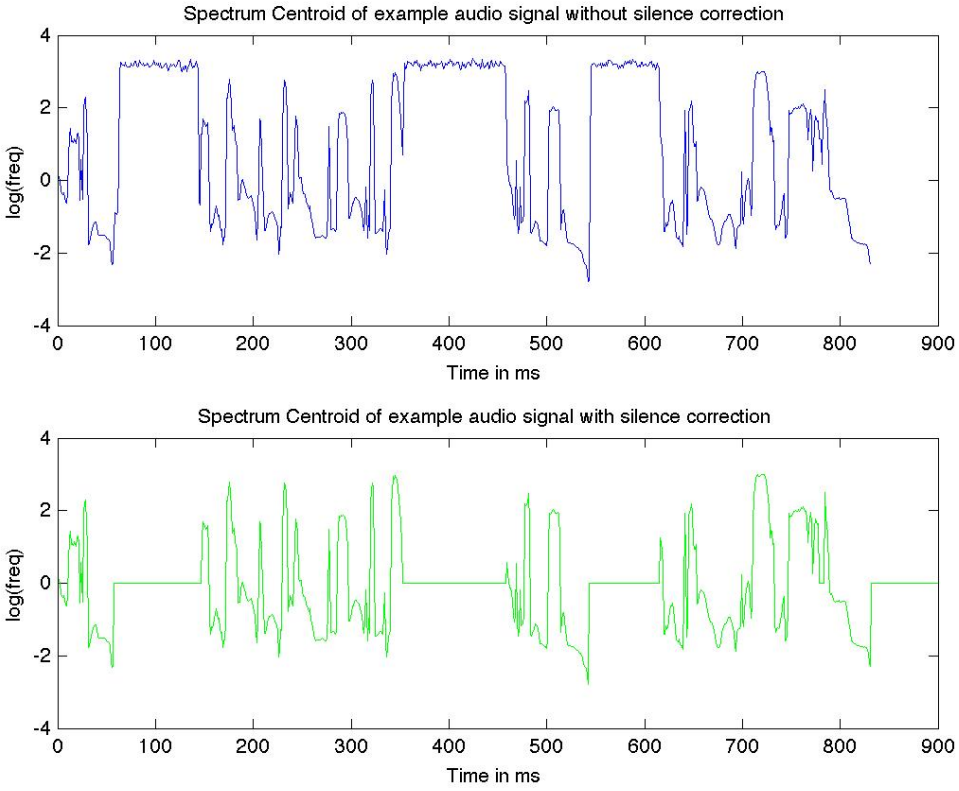

Figure 7: Spectrum Centroid of the example audio signal. Above without silence correction. Down with silence correction.

3.4 All-XM MATLAB library

All-XM is a Matlab [Mat] library for the MPEG7 Audio Descriptors. It provides a .m-file for every LLD and optional XML-Output for each of it. The creators of this library tried to follow the outline of the interface given by the standard which is why it sometimes does not follow normal Matlab coding style.
Unfortunately the Silence Descriptor which this recognition is mostly based on is not implemented. So it is a custom-made descriptor which may not support other projects. For further details about the single descriptors see in the basics chapter 3.3.2.
The version of the library which is used is included on the attached CD.

4 Investigation of Several Audio Descriptors

As chapter 3 described the descriptors in use it is obvious which ones the used are, but it has to be explained why these are chosen:

Since the speaker equipment shall be also used for general announcements (no warnings) it is necessary to provide an easy way to distinguish if the incoming signal is a warning or not. This is thought of to be done by adding a pilot tone to the warning signals. With a frequency of 19kHz it cannot be heard by humans, but lies inside the range of the descriptors which is why it has to be filtered out before the actual recognition starts.

To decide which of the Low Level Descriptors should be used for the recognition the type of the signals has to be considered. They are limited to speech only and delivered in the same voice, so any timbral descriptors would not be useful. Those were to choose when dealing with more musical material.
Since speech naturally has pauses between the single words, it is obvious to use the Silence Descriptor (3.3.2) to cut the signals in language blocks.
But just the Silence Descriptor is not a reliable criteria for all signals. So others have to be added. As it should be kept simple, the Spectrum Centroid is likely to be considered as another criteria, because even slightest differences in the signals (e.g. just a different name of the accelerator) provide a clear change in the value of the descriptor which becomes even greater in a cross correlation. This is proved by the conclusion matrix for the Spectrum Centroid criteria, seen in figure 9.

Once decided which descriptors shall be used for the recognition it has to be proved that they are able to distinguish the signals correctly. That was done using cross correlation. Following the normally defined formula for a cross correlation of two signals:

$$(s1 \star s2)[k] = \sum_{m=-\infty}^{\infty} s1^*[m]s2[k+m] \tag{1}$$

Here the bounds of the sum can be limited to a close neighbourhood to zero, since a larger shift cycle would not create more precise results.

$$(s1 \star s2)[k] = \sum_{m=-N}^{N} s1^*[m]s2[k+m] \tag{2}$$

N represents the maximum temporal shift of one signal against the second.

Most interesting is the value of the peak nearest to a zero shift.
So the resulting number of the feature space is the quadratic of the quantity of the reference signals for every feature that is included.
One test set was formed between the original signals and one test set between the original signals and the rerecorded signals.
The results of these cross correlations were then reshaped and put into a confusion matrix which can be seen down below.
On both axes the different signals are listed in alphabetical order (see listing 1 in Appendix A), so that we expect a straight line of correct indicated signals along the principal diagonal.

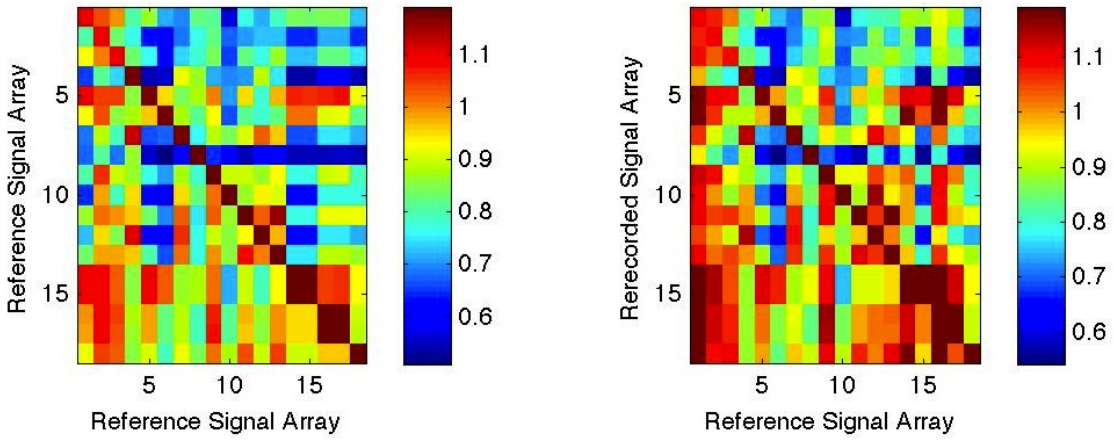

Figure 8: Confusion matrices I: Silence Descriptor criteria. Left the reference and right the rerecorded signals.

Figure 8 shows that the Silence Descriptor does recognize all signals correctly (straight line in the principal diagonal), but seems not to provide enough information to distinguish between the different signals. That is logical considering the fact tat the signals mostly just differ in the name of the accelerator and so the speech pauses in the rest of the signal are nearly the same.

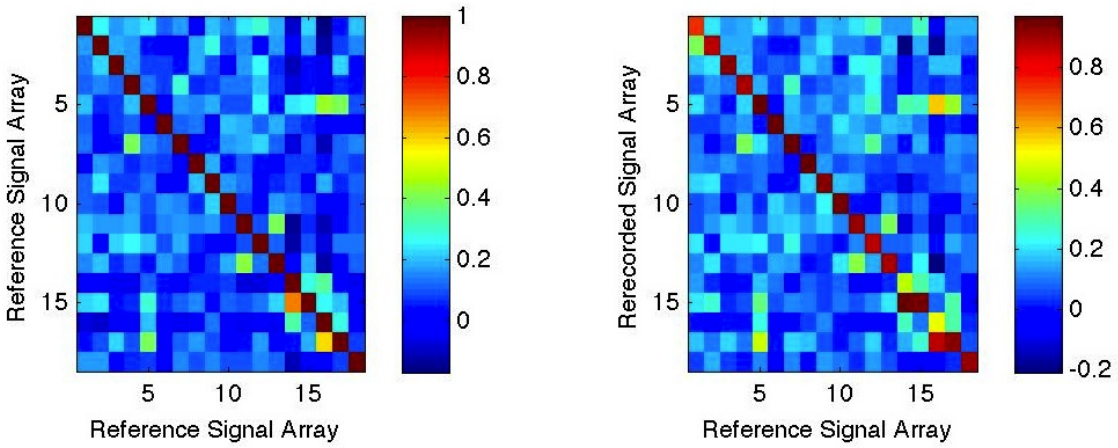

Figure 9: Confusion matrices II, Spectrum Centroid criteria. Left the reference and right the rerecorded signals.

This graphic 9 shows that the Spectrum Centroid criteria as itself provides nearly enough information to recognize all signals correctly, but might have problems with the signals which have a pilot tone. This can especially seen in the rerecorded against reference matrix on the right-hand side. The pairs (14,15) and (16,17) should be consulted here. Of course one has to keep in mind that here the complete signals are taken for calculation.

The Silence Descriptor criteria and the Spectrum Centroid criteria both on itself are not sufficient to distinguish all signals correctly. Especially when thinking about the real time aspect that later is needed the Spectrum Centroid criteria on its own would at first not provide enough information to consider a special signal as a candidate.

So it is reasonable to use a combined criteria out of them both. In the graphic below 10 this is shown as an easy test with a 50/50 weight. This provides especially for the rerecorded signals a better result for the signals with pilot tone. (Pairs (14,15) and (16,17)).

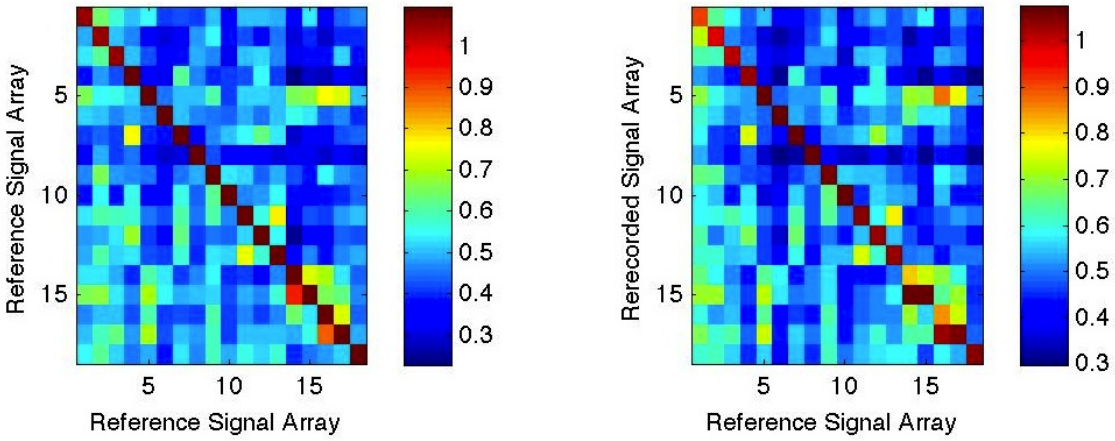

Figure 10: Confusion matrices III, 50/50 combined criteria. Left the reference and right the rerecorded signals.

That proves that these two descriptors together provide enough information once the signal is determined as a warning by a positive pilot tone criteria.

5 MATLAB Prototype Development

5.1 Pilot Tone Evaluation

The pilot tone added to the signal shall just act as an indicator for a warning. It has no other use than this and needs to be erased for the rest of the recognition as it would have to much impact on the values of the used LLDs. This impact of the pilot tone can be seen clearly by considering a spectrogram for the rerecorded signals with pilot tone against a spectrogram of the same reference signal with pilot tone, as shown in the figure 11 below. Surely the cause of this scattered pattern lies in the way the signals are rerecorded, but it shows the effect. This is a cause for its evaluation got realized by use of the ASE.

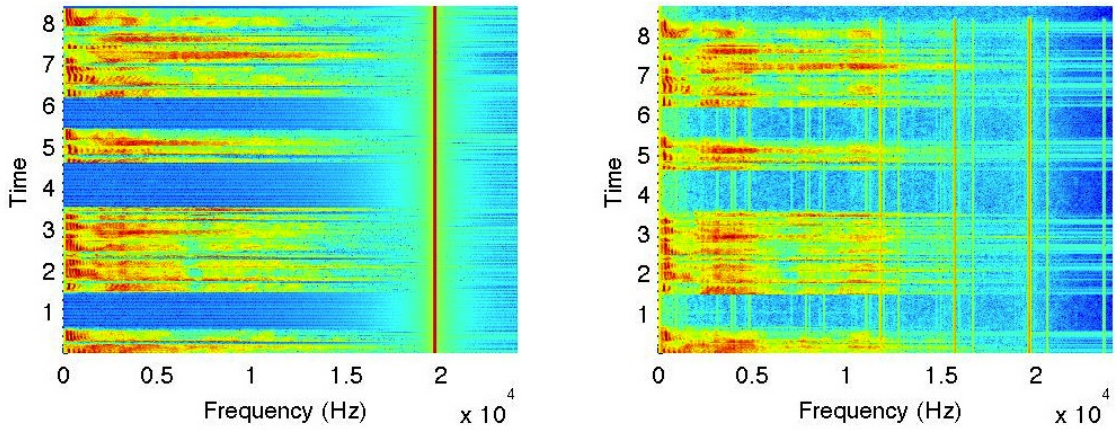

Figure 11: Spectrogram of Petra warning with pilot tone impact, seen at the rerecorded signals (right-hand side)

5.2 Silence Descriptor

As the ASE is already calculated for the pilot tone criteria it can be used for the Silence Descriptor as well. This gives the advantage that the silence can be evaluated from the spectrum representation of the signal, already split into different frequency bands. Having hold of these bands the pilot tone frequency can easily be excluded, so it has no effect for the silence evaluation which would otherwise be fatal since no actual silence could be gathered due to the impact of the pilot tone frequency being always dominant

This figure 12 shows that the threshold level must have set an much higher when a pilot signal is included and it would interfere with some speech parts of the signals.
To detect silence there must be next to none frequency outcome at all frequency bands. So

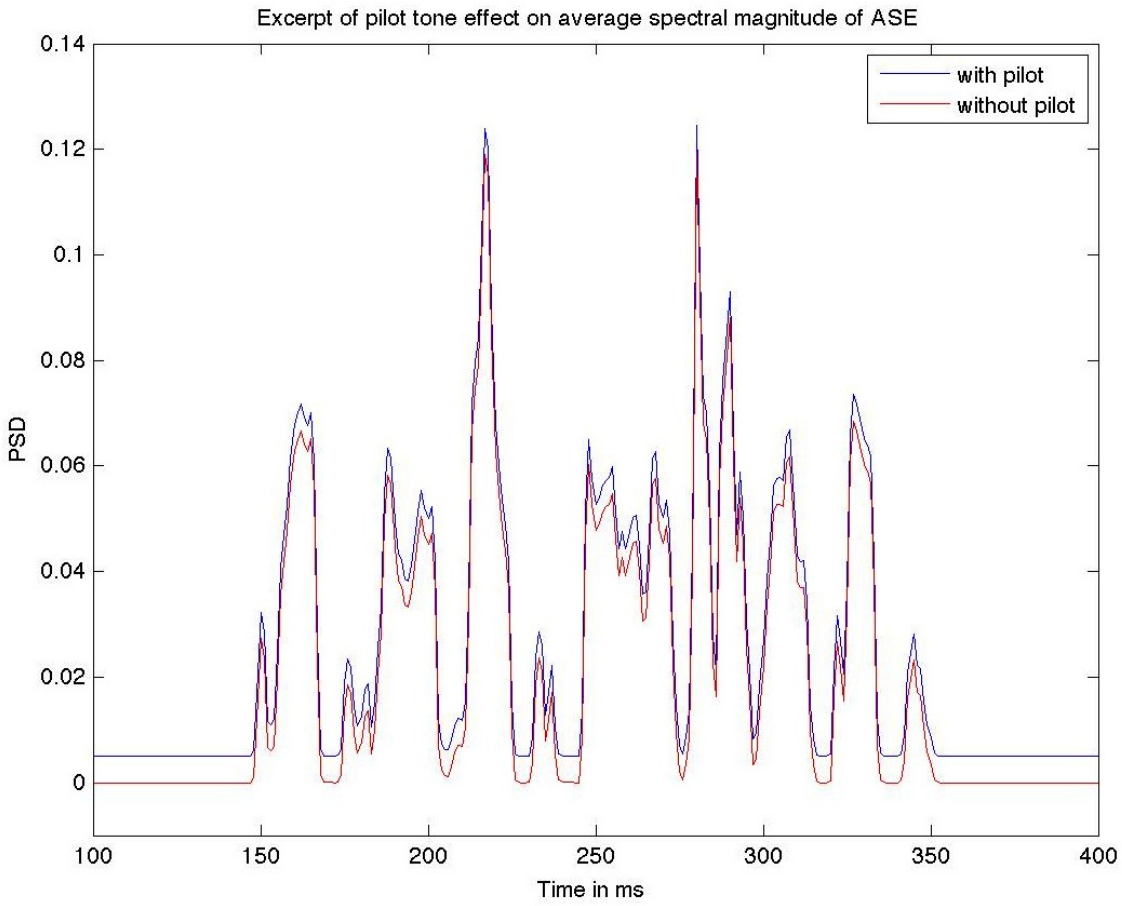

Figure 12: Excerpt of average spectral magnitude of ASE of signal with pilot tone and without pilot tone

the ASE is summed up and a marker is set to true at the points where the resulting PSD is smaller then a threshold.
Since the Silence Descriptor uses the values of the ASE as a basis they have a common time increment which is 10ms, the default value for the ASE Descriptor.

5.3 ASE and Spectrum Centroid

For both, the Audio Spectrum Envelope and the Spectrum Centroid, the functions of the All-XM library are used. Only a wrapper to exclude unnecessary parameters (such as the not needed XML-Output) was added.
Also the time increment of the Spectrum Centroid is set to default 10ms which results in no timing problems when compare the descriptors.

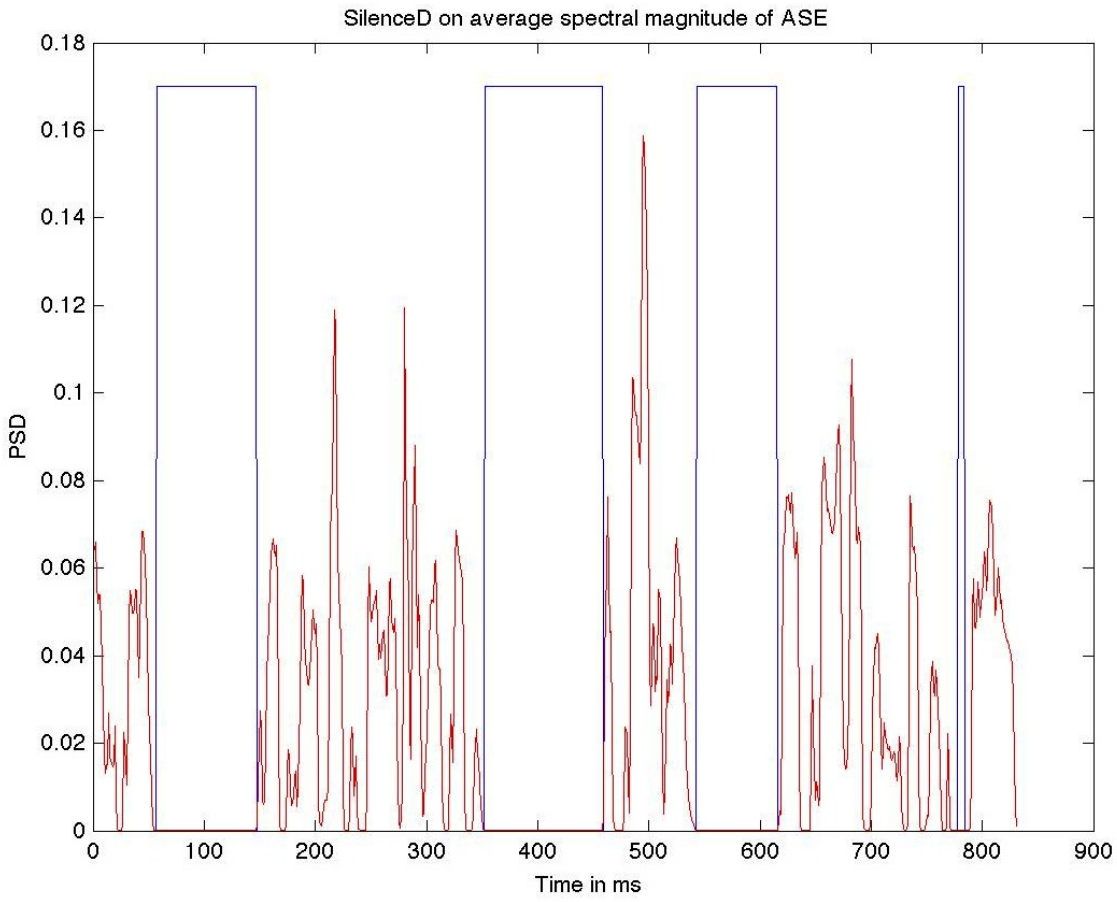

Figure 13: SilenceD laid on top of average spectral magnitude of ASE

As already said in chapter 3.3.2 the Spectrum Centroid is modified in the silence segments to exclude a false impact from silence areas.

5.4 Signal Recog and Single Signal Recog

With the help of the Silence Descriptor the signals are divided into language blocks, silence- and non-silence- segments (so sound) alternating which forms the typical pattern of speech. This is done by the Matlab function 'getLanguageBlocks' which can be seen in the attached file in appendix B.
These language blocks are used for a kind of Final State Machine (FSM) which is drafted in figure 14 below.

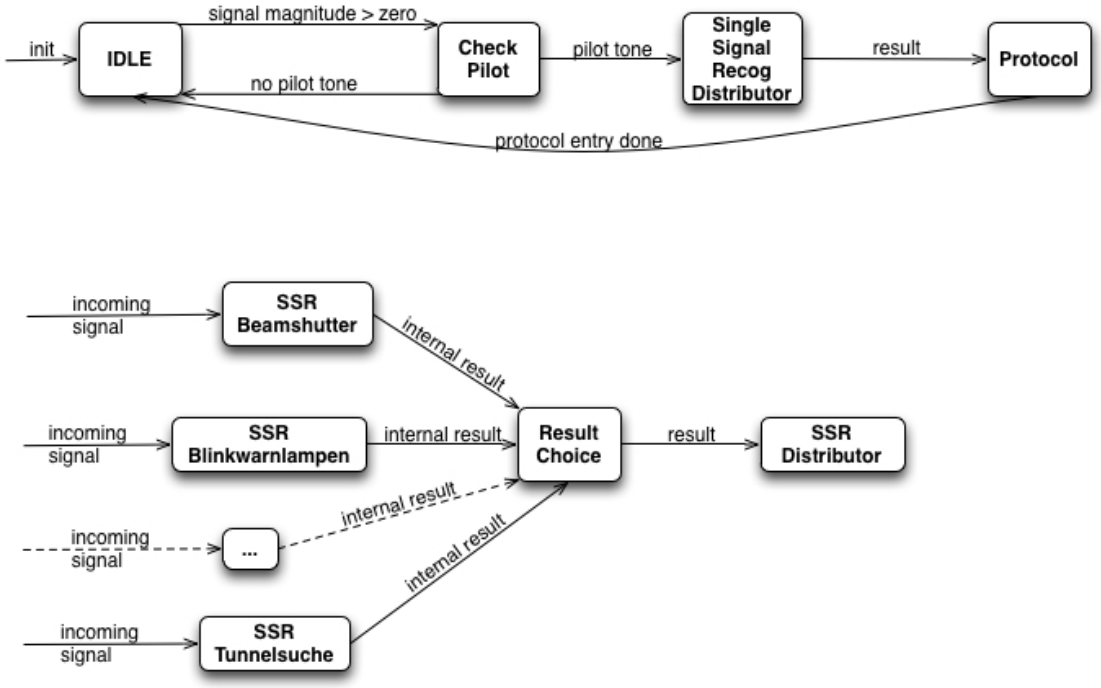

Figure 14: Signal Recog and the depending Single Signal Recog Distributor

This FSM is not really implemented as a FSM in this Matlab Test implementation. It is represented through two state counter, one for positive matches (*stateNo*) and one for mistakes (*ErrorState*). But it helps to grip the idea of the concept.

The FSM is realized with two parts - the Signal Recog (SR) and the Single Signal Recog (SSR) (as the names in Matlab are). Where Single Signal Recog is the sub FSM to the Signal Recog.

The SR is triggered when a signal amplitude greater then zero is observed. If also a pilot tone is on the signal it hands it down to the SSRs. The Plural implies it, the Single Signal Recogs are state machines for one single reference signal.
The incoming signal is compared against the saved language blocks of the particular reference signal and if the whole signal is recognized that result will be mentioned to the super FSM.
When the language block signals it could be a potential match additionally the values of the SilenceD and the Spectrum Centroid criteria (cross correlation values) are compared, too.
The SSR is drafted in figure 15.
At the end of each recognition cycle a line stating its result is added to the protocol file.

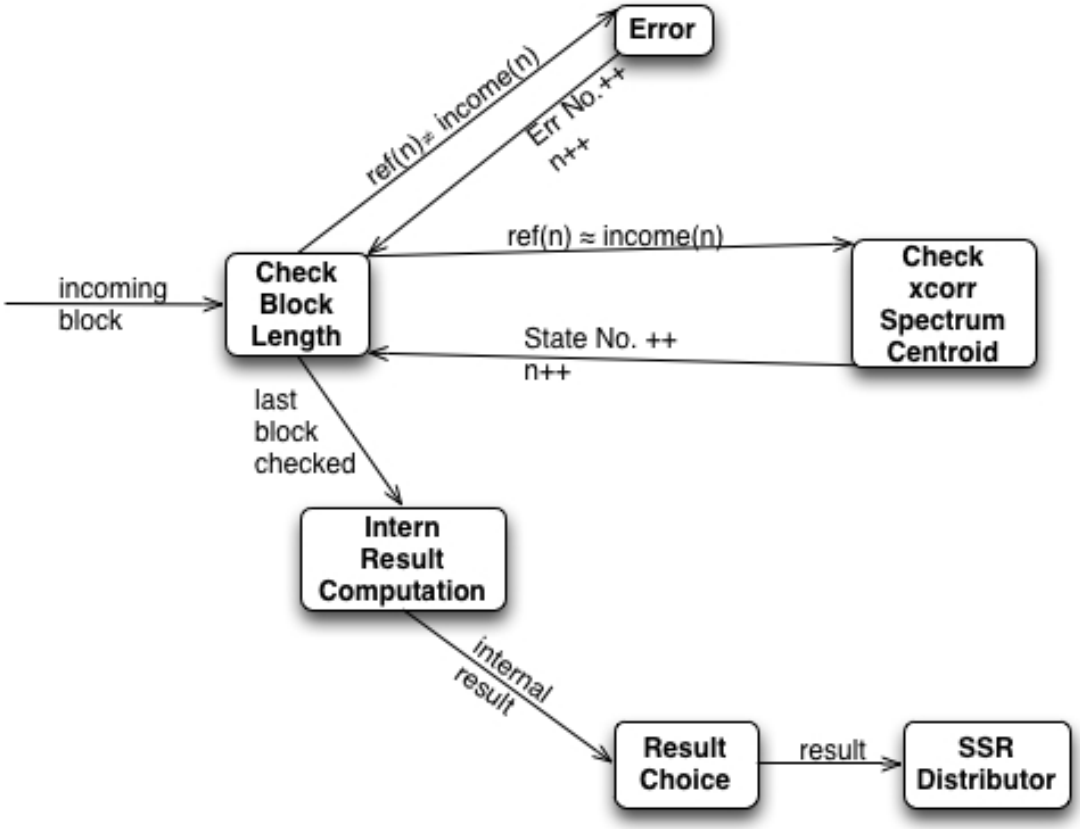

Figure 15: Single Signal Recog

Defined how the recognition works when the whole signal is known we must now take a further step: The recognition shall later be in real time, so we will have a streaming input signal. This leads to a problem with real real time. That is not possible due to the fact that the descriptors do need a much larger number of samples to work on than one might think when thinking about usual block sizes. So for a first try it is decided to use a buffer and wait until it is filled with about one second of audio material. The first non-silence signal triggers the recognition. The incoming blocks will be put into a cross correlation with the reference signals. For the first block it is done with all the reference signals and later just the blocks who are likely to be correct. Which ones those are can be decided by taking the peaks of the cross correlation. So a possible candidate will create a peak pattern with peaks shifted by around one block size. If there is just one candidate left, it has to be followed until the end of the signal to decide whether it was played totally correct or not.
Surely the beginning of an incoming signal is not exactly in time with the reference signal. So some kind of dynamic time warping needs to be included in the recognition.
Possibly an approach similar to the one outlined in Simon Dixxon's DAFX conference paper [Dix05] will be useful for dealing with this problem.

6 Conclusion

As shown the developed algorithm is capable of recognizing the provided audio material. Surely some more signals could be added if necessary as long as they have an added pilot tone of the right frequency.
Due to the lack of hardware it could not be examined how the algorithm reacts in the real environment and its effects.
The testing material is constructed and possibly in reality some more problems which cannot be thought of can occur. Therefore the employees of DESY have to be asked what could possibly happen to the signal. Then a more realistic testing environment can be constructed.

There are some factors for optimization:
- The chosen block size for the buffer of the incoming signal.
- Use of a reduced cross correlation function instead of the Matlab library function
- Re-implementing the MPEG7 library function to minimize overhead

By the time the concrete hardware for the accelerator is decided the recognition can be implemented for that special system.
Then the algorithm has to be proved in reality once it is implemented. Also it needs to be co-checked from the people who work at DESY if there are any further requirements or if some adjustments have to be made.
It might be necessary to include for example an explicit order of signals in the recognition process to be checked simultaneously to the recognition.

Further workings have to include the implementation of this developed algorithm for the then defined hardware configuration as well as testing the algorithm on the actual hardware system.
And also, as said before, a collaboration with the employees at DESY is needed.
However it has also made to be clear that it is only a solution for this special scenario and will most probable not fit for others.

List of Figures

List of Tables

Abbreviations

ASE	Audio Spectrum Envelope
dB	Dezibel
DESY	Deutsches Elektronen-Synchrotron
FSM	Final State Machine
HAW	Hochschule für Angewandte Wissenschaften
HW	Hardware
Hz	Hertz
kHz	kilo Hertz
LLD	Low Level Descriptor
MPEG7	Multimedia Content Description Interface
SR	Signal Recog
SSR	Single Signal Recog

References

[And04] Tobias Andersson. Audio classification and content description. Master's thesis, Luleå Tekniska Universitet, 15.03.2004.

[Cas04] Michael Casey. All-xm library. http://mpeg7.doc.gold.ac.uk/mirror/index.html, 2004.

[Cas12] Michael Casey. Mpeg-7 multimedia software resources. http://mpeg7.doc.gold.ac.uk/, 31.01.2012.

[DES09] DESY. 50 years of desy - insight starts here. http://pr.desy.de/sites2009/site_pr/content/e113/e48/column-objekt55756/lbox/infoboxContent55758/DESY50_anniversabrochure_E_eng.pdf, September 2009.

[DES12] DESY. Desy - deutsches elektronen-synchrotron. http://www.desy.de/, 2012.

[Dim08] Giuseppe Dimattia. An automatic audio classification system for radio newscast. Master's thesis, Departament de Teoria del Senyal i Comunicacions de Terrassa, http://upcommons.upc.edu/pfc/bitstream/2099.1/4863/1/GiuseppeDimattia.pdf, March 2008.

[Dix05] Simon Dixxon. Live tracking of musical performances using on-line time warping. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.117.5005rep=rep1type=pdf, 2005.

[GI12] GI. Mpeg-7 - gesellschaft für informatik e.v. http://www.gi.de/no_cache/service/informatiklexikon/informatiklexikon-detailansicht/meldung/mpeg-7-52.html, 31.01.2012.

[Gmb12] European X-Ray Free-Electron Laser Facility GmbH. European xfel. http://www.xfel.eu/en/, 2012.

[Inc12] Apple Inc. Apple website siri. http://www.apple.com/iphone/features/siri.html, 2012.

[Mar12] José M. Martínez. Mpeg-7 overview. http://mpeg.chiariglione.org/standards/mpeg-7/mpeg-7.htm#E11E8, 31.01.2012.

[Mat] Mathworks. Mathwork's website. http://www.mathworks.de, 31.01.2012.

[MPE] MPEG. Mpeg home page. http://mpeg.chiariglione.org/, 31.02.2012.

[Pee04] Geoffroy Peeters. Aes25 mpeg-7 workshop introduction, 17.6.2004.

[RT10] Ye Li Ran Tao, Zhenyang Li. Music genre classification using temporal information and support vector machine. http://www.asci.tudelft.nl/media/proceedings_asci_conference_2010/asci2010_submission_41.pdf, 2010.

[SS02] Phillipe Salembier and Thomas Sikora. *Introduction to MPEG-7: Multimedia Content Description Interface*. Number 0471486787. John Wiley & Sons, Inc., New York, NY, USA, 1. edition edition, 2002.

[web11] Dafx conference. http://www.dafx.de, 01.11.2011.

[Wik12] Wikipedia. Hidden markov model. http://en.wikipedia.org/wiki/Hidden_Markov_model, 2012.

[Zöl96] Udo Zölzer. *Digitale Audiosignalverarbeitung*. Informationstechnik. Prof Dr.-Ing. Norbert Fliege, 1. edition, 1996.

[Zöl02] Udo Zölzer. *Digital Audio Effects (DAFX)*. ISBN 978-0471490784. John Wiley and Sons, 1. edition edition, 2002.

A Appendix A

A.1 MPEG7 Time Format

The time format in MPEG7 can represent two different types: media time and generic or world time. Both types are represented equally, but to the world time a time zone is added. For this work only the media time is important.
The media time data types consist of two parts, a start time point (*mediaTimePoint data type*) and a duration (*mediaDuration data type*), so together they represent a time period.
The syntax of the mediaTimePoint data type is:

-YYYY-MM-DDThh:mm:ss:nFN.

The different variables mean:
Y - year, can include negative value for BC.
M - months
D - days
T - separator
h - hours
m - minutes
s - seconds
nFN - 1/N is a fraction of one second and *n* the number of those fractions. *F* - separator.

So a certain time point in history (e.g. the end of the world, as described by the Mayas - 21.12.2012 15:00h) can be written as follows:

```
<MediaTimePoint>2012-12-21T15:00:00:0F0</MediaTimePoint>.
```

The following syntax is used by the mediaDuration data type:

(-) PnDTnHnMnSnNnF.

P is a separator which indicates the start of a duration. n is a count followed by the specified unit being counted. The units are as described above for the mediaTimePoint data type, where N specifies the counted fractions and F the fractions of one second.
For example a musical piece which lasts 3 minutes, 44 seconds and 27/30 of a second is written as:

```
<MediaDuration>PT3M44S27N30F</MediaDuration>.
```

The usual time period used in the descriptors, e.g. for the hopsize, is 10ms, which is written as PT10N1000F.

A.2 Reference Signals provided by DESY

No.	Name
1	Beamshutter-48m-equ-komp
2	Blinkwarnlampen-1648-m-equ-komp
3	DESY2-1648m-equ-komp
4	DESY2-Mag-1648m-equ-komp
5	DORIS-Str-FG-1648m-equ-komp
6	FLASH-48m-equ-komp
7	HF-FG-1648m-equ-komp
8	Interl.-Abs.-Exp-1648m-equ-komp
9	Interl.-Tuer.-Tst-1648m-equ-komp
10	Interlockabs-1648m-equ-komp
11	Linac2-1648m-equ-komp
12	Linac2-Gun-1648m-equ-komp
13	Magnetstromfreigabe-48m-equ-komp
14	PETRA-Str-FG-1648m-equ-komp-pilot
15	PETRA-Str-FG-1648m-equ-komp
16	Strahl-1648m-equ-komp-pilot-19600
17	Strahl-1648m-equ-komp
18	Tunnelsuche-1648m-equ-komp

Table 1: Reference Signals provided by DESY

All specific accelerator or experiment signals have the following form: "Achtung! _NAME_ wird eingeschaltet. Attention! _NAME_ will be switched on."

Beside those there are a few different signals examined in this recognition. They read as follows:
"Die gelben Blinkwarnlampen laufen nur zur Probe. The yellow blinking lamps are running for test purposes only."
"Interlockabsuche. Bitte verlassen Sie das Experimente-Gebiet. Interlock search. Please leave the experimental area."
"Interlockabsuche. Bitte verlassen Sie das Gebiet. Interlock search. Please leave the area."
"Die Interlock Türen werden nur zu Testzwecken gesetzt. The interlock doors will be set for test purposes only."
"Tunnelsuche. Bitte velassen Sie den Tunnel. Tunnel search. Please leave the tunnel."

B Appendix B - Matlab Files

Here the interesting parts of the Matlab files. All needed files can be found on the attached CD.

B.1 'getLanguageBlocks'

```
%{
    divide the signal into speech pattern, silence and sound alternating
    L. Knetter, November 2011
%}


function langLengths = getLanguageBlocks(silence)

    [XX YY] = size(silence);
    lanLengths = [];
    langLengths = [];
    maxBlockLength = 20;
    silenceLengths_norm = zeros(maxBlockLength,1);
    speakLengths_norm = zeros(maxBlockLength,1);

    % get edges of silence periods
    tmp = find(silence == 1);
    tmp2 = diff(tmp);
    silenceLengths = find(tmp2 >1);
    silenceLengths_norm(1:length(silenceLengths)) = silenceLengths;

    tmp = find(silence == 0);
    tmp2 = diff(tmp);
    speakLengths = find(tmp2 >1);
    speakLengths_norm(1:length(speakLengths)) = speakLengths;

    for i = 1:length(speakLengths_norm)
        if ~(speakLengths_norm(i) == 0 || silenceLengths_norm(i) == 0)
            if(speakLengths_norm(i) < silenceLengths_norm(i))
                if(i == 1)
                    lanLengths = [lanLengths; [1 speakLengths_norm(i)]];
                end
                lanLengths = [lanLengths; [speakLengths_norm(i) silenceLengths_norm(i)]];
            else
                if(i == 1)
                    lanLengths = [lanLengths; [1 silenceLengths_norm(i)]];
                end
                lanLengths = [lanLengths; [silenceLengths_norm(i) speakLengths_norm(i)]];
            end
        else
            if (speakLengths_norm(i) == 0)
                lanLengths = [lanLengths; [silenceLengths_norm(i) XX]];
                break;
            end
            if(silenceLengths_norm(i) == 0)
                lanLengths = [lanLengths; [speakLengths_norm(i) XX]];
                break;
            end
```

```
        end
    end


    X = lanLengths(:,1);
    Y = lanLengths(:,2);
    for k = 1:length(X)
            langLengths = [langLengths; Y(k) - X(k)];
    end

end
```

B.2 SilenceD

```
%{
    silenceD.m
    Input: Audio Spectrum Envelope of audio signal
    Output: - @param silenceExisting: bool array if silence or not (1 means silence ↩
        existent)
            - @param NoOfSilences: Number of silence peiods in the signal

    L.Knetter, August-November 2011; edited 16.1.2012
%}

function [silenceExisting, NoOfSilences] = SilenceD(auSigASE) %param auSigASE - ↩
    corresponding ASE to an audiosignal

    silenceThreshold = 9e-6;

    % delete pilot tone
    auSigASE = auSigASE(:,1:end-2);

    ASEsum = sum(auSigASE,2);

    time = find(ASEsum < silenceThreshold);
    timespots = diff(time) > 1;
    silenceEndTime = time(timespots);
    NoOfSilences = length(silenceEndTime) + 1;



    [X Y] = size(auSigASE);
    silenceExisting = zeros(X,1);
    silenceExisting(time) = 1;

    % eliminate 1-3 samples silence
    count = 0;
    memo = [];
    for i = 1:length(silenceExisting)
        if(silenceExisting(i) == 1)
            count = count+1;

            if(count<=4)
                memo = [memo; i];
            end

        else
            if(count<=4)
```

```
                for j=1:length(memo)
                    silenceExisting(memo(j)) = 0;
                end
            end
            count = 0;
            memo = [];
        end
    end
end
```

B.3 SR - Signal Recog

```
%{
    SignalRecog - Super state machine which recognizes all reference signals.

    L. Knetter, November 2011
%}

% @param langBlocks: output of getLanguageBlocks()
function SSRFeats = SignalRecog(filentrys, feats, name_check, langBlocks_check, ↩
    specCentroid_check, silence_check)


    SSRFeats = [];
    for i = 1:filentrys

        SSRFeat = SingleSignalRecog(feats(i).langBlocks,langBlocks_check,...
                        feats(i).name,name_check,feats(i).spectralCentroid,...
                        specCentroid_check,feats(i).silence,silence_check);
        SSRFeats = [SSRFeats; SSRFeat];

    end


end
```

B.4 SSR - Single Signal Recog

```
%{
    SingleSignalRecog - State machine to recognize one reference signal.

    L. Knetter, November 2011
%}

% @param langBlocks: output of getLanguageBlocks()
function SSRFeature = SingleSignalRecog(langBlocks_ref, langBlocks, refName, testName, ↩
    specCen_ref, specCen, sil_ref, sil)

    maxBlocksCount = 20;
    langBlocks_ref_norm = zeros(maxBlocksCount,1);
    langBlocks_norm = zeros(maxBlocksCount,1);
    langBlocks_ref_norm(1:length(langBlocks_ref)) = langBlocks_ref;
    langBlocks_norm(1:length(langBlocks)) = langBlocks;
```

```
MAXLAG = 10;

autoCorr_ref = xcorr(specCen_ref, MAXLAG, 'unbiased'); %Autocorrelation is Massstab
signalEnergy = max(autoCorr_ref);

xcorr_SpecCen_fkt = xcorr(specCen_ref, specCen, MAXLAG,'unbiased');
xc_specCen_krit = xcorr_SpecCen_fkt / signalEnergy;
xc_specCen_kriterium = max(xc_specCen_krit);

autoCorrLang_ref = xcorr(langBlocks_ref, MAXLAG, 'unbiased'); %Autocorrelation is ↩
    Massstab
signalEnergyLang = max(autoCorrLang_ref);

xcorr_Lang_fkt = xcorr(langBlocks_ref, langBlocks, MAXLAG,'unbiased');
xc_Lang_krit = xcorr_Lang_fkt / signalEnergyLang;
xc_Lang_kriterium = max(xc_Lang_krit);

autoCorrSil_ref = xcorr((1-sil_ref), MAXLAG, 'unbiased'); %Autocorrelation is ↩
    Massstab
signalEnergySil = max(autoCorrSil_ref);

a=zeros(1200,1);
b=zeros(1200,1);
a(1:length(sil_ref))=(1.-sil_ref);
b(1:length(sil))=(1.-sil);
xcorr_Sil_fkt = xcorr(a, b, MAXLAG,'none');
xc_Sil_krit = xcorr_Sil_fkt / signalEnergySil;
xc_Sil_kriterium = max(xc_Sil_krit);

xc_combined_kriterium = (xc_Sil_kriterium *0.5)/1000 + xc_specCen_kriterium *0.5;
stateNo = zeros(maxBlocksCount+1,1);
stateNo(1) = 1;
tolerance = 5;
ErrorState = [];

for i = 1:length(langBlocks_norm)
    if(langBlocks_norm(i) >= (langBlocks_ref_norm(i))-tolerance && langBlocks_norm(i)↩
         <= (langBlocks_ref_norm(i)+tolerance))
        if(xc_specCen_kriterium >= 0.50 && xc_specCen_kriterium <= 1.00)
        stateNo(i+1) = stateNo(i)+1;
        ErrorState = [ErrorState; NaN ];
        end
    else
        stateNo(i+1) = stateNo(i)+100;
        ErrorState = [ErrorState; stateNo(i)];
    end
end
stateline = find(diff(stateNo)>1, 1);
if(isempty(stateline))
    Name = refName;
%{
elseif(length(stateline)<=1)
    if(xc_specCen_kriterium >= 0.50 && xc_specCen_kriterium <= 1.00)
        Name = refName;
    else
        Name = 'No correct signal.';
    end
    %}
else
    Name = 'No correct signal.';
    %Name = 'ERROR';
end
```

```
    SSRFeature = SSRFeatures(Name,testName,refName,ErrorState,xc_specCen_kriterium,↩
        xc_combined_kriterium,xc_Sil_kriterium);

end
```

B.5 Recog Test

```
%{

    Testen des Erkennungsautomaten:

    1. Signale einlesen (Namen)
    2. ASE bestimmen
    3. Merkmale extrahieren (silence)
    4. LanguageBlocks bestimmen
    5. diese in den automaten geben
    6. automat ruft subautomat f?r jedes referenzsignal auf
    7. automat bestimmt aus den ergebnissen der subautomaten, welches signal vorhanden.

%}

%% 1.

    [filelist, filentrys] = filescan(audio_path, '*.wav');
    % !ATT 'filelist' is a cell array!

filelist = filelist(1:end-2); %exclude problem with WT (only stereo signal) and the noise↩
    .wav which is no signal
filelist(19) = [];
filelist(6) =[];
filentrys = filentrys-4;

    [RR_filelist, RR_filentrys] = filescan(rerecord_path, '*.wav');

RR_filelist = RR_filelist(1:end-2); %exclude problem with WT (only stereo signal) and the↩
     noise.wav which is no signal
RR_filelist(6) =[];
RR_filentrys = RR_filentrys-3;
out='both filelists read'
%% 2. 3. 4.

% ASEs should be a multidimensional array otherwise access not possible
% later onwards
xmax = 1000;
ymax = 53;
ASEs = zeros(xmax,ymax,filentrys);
feats = [];

for i = 1:filentrys
    file = filelist{i};
    tmp = getASE(file);
    [X Y] = size(tmp);
    out='vor ASE'
    if(Y == ymax)
        ASEs(1:X,:,i) = tmp;
    else
        echo on
```

```
        'Fehler bei ASE Bildung.'
        file
        i
        echo off
        return;
    end
    out='hinter ASE. feat beginn'
    id_i = i+100;
    tmp = extractMPEG7Features(file, ASEs(:,:,i), id_i);
    feats = [feats; tmp];
    out='feat ende.'

end
out='features ref signals'
%---------- RR signals ----------------
RR_ASEs = zeros(xmax,ymax,filentrys);
RR_feats = [];
%langBlocks = [];

for i = 1:RR_filentrys
    RR_file = RR_filelist{i};
    tmp = getASE(RR_file);
    [X Y] = size(tmp);
    out='vor ASE'
    if(Y == ymax)
        RR_ASEs(1:X,:,i) = tmp;
    else
        echo on
        'Fehler bei ASE Bildung.'
        file
        i
        echo off
        return;
    end
    out='hinter ASE. feat beginn'
    tmp = extractMPEG7Features(RR_file, RR_ASEs(:,:,i),i+200);
    RR_feats = [RR_feats; tmp];
    out='feat ende.'
    %{
    tmp = getLanguageBlocks(feats(i).silence);
    langBlocks = [langBlocks; tmp];
    out='langblock ende'
    %}
end
out='features RR_xxx signals'

%% 5. 6.


SSRFeats = [];
for j = 1:filentrys

    SSRFeatstmp = SignalRecog(filentrys,feats,feats(j).name,feats(j).langBlocks,feats(j).↩
        spectralCentroid,feats(j).silence);
    SSRFeats = [SSRFeats; SSRFeatstmp];

end

Name = reshape(SSRFeats,18,18);

out='SSRFeats ref signals'

%------- RR signals against ref signals -------
```

```
RR_SSRFeats = [];
for j = 1:RR_filentrys

    RR_SSRFeatstmp = SignalRecog(filentrys,feats,RR_feats(j).name,RR_feats(j).langBlocks,↩
        RR_feats(j).spectralCentroid,RR_feats(j).silence);
    RR_SSRFeats = [RR_SSRFeats; RR_SSRFeatstmp];

end

RR_Name = reshape(RR_SSRFeats,18,18);

out='SSRFeats RR_xxx signals'

%% Test of streamed Insignal
%{
p=wavread(Petra);
l=wavread(Linac);
rp=wavread(RR_Petra);

resu = [];
diff = [];

blocksize = length(p)/1000;
idx = 1:blocksize;
oldidx=1;
wavwrite(p(idx),48000,'blocktmp.wav');
for i=1:1000
    if(i>1)
    idx = oldidx:oldidx+blocksize;
    wavwrite(p(idx),48000,'blocktmp.wav');
    end
    [resutmp difftmp] = streamedInRecog('blocktmp.wav','blocktmp.wav');
    resu = [resu; resutmp];
    diff = [diff; difftmp];
    oldidx=oldidx+blocksize;
end

plot(resu);

%}
```

C Content of CD-Rom

- Bachelor thesis "Identification and logging of alarm records through feature detection based on MPEG7 Low-Level-Audio-Descriptors" (PDF)
- Matlab .m-files
- used library (zip-file)
- some publications cited in this work

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §22(4) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 21. März 2012	
Ort, Datum	Unterschrift