FP7-ICT-2013-C TWO!EARS Project 618075

Deliverable 3.2

Software Architecture

WP3 *

November 28, 2014

* The TWO!EARS project (`http://www.twoears.eu`) has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 618075.

Project acronym:	TWO!EARS
Project full title:	Reading the world with TWO!EARS

Work packages:	WP3
Document number:	D3.2
Document title:	Software architecture
Version:	1

Delivery date:	30th November 2014
Actual publication date:	30th November 2014
Dissemination level:	Public
Nature:	Report

Editor:	Guy Brown
Author(s):	Ning Ma, Ivo Trowitzsch, Jalil Taghia, Christopher Schymura, Dorothea Kolossa, Thomas Walther, Hagen Wierstorf, Tobias May, Guy Brown
Reviewer(s):	Jonas Braasch, Dorothea Kolossa, Bruno Gas, Klaus Obermayer

Contents

1 Executive summary

The goal of the Two!Ears project is to develop an intelligent, active computational model of auditory perception and experience in a multi-modal context. At the heart of the project is a software architecture that optimally fuses prior knowledge with the currently available sensor input, in order to find the best explanation of all available information. Top-down feedback plays a crucial role in this process. The software architecture will be implemented on a mobile robot endowed with a binaural head and stereo cameras, allowing for active exploration and understanding of audiovisual scenes. Our approach recasts a conventional "blackboard system" in a modern machine learning framework. In the blackboard system, knowledge sources cooperate to solve a problem; in this case, the problem is to identify the acoustic sources that are present in the environment and ascribe meaning to them.

This deliverable documents our progress on the design and development of the Two!Ears software architecture. We have developed a blackboard system with three layers. In the first layer, the acoustic input is pre-segmented using information about pitch, onsets/offsets, interaural coherence and amplitude modulation. This allows sound sources of interest to be separated from the acoustic background. Visual information gathered from cameras (or initially from a 3D simulation) are also segmented. Since sound sources of interest in the environment will not be static, we have also developed a framework for nonlinear tracking of sound sources which models their underlying motion dynamics. In the second layer of the expert system, audio-visual events are labelled to indicate their attributes. So far, we have focused on attributes relating to the spatial location of sound events and their source type (e.g., 'female voice' or 'telephone ring'). Correctly labelling these attributes in noisy and reverberant acoustic environments is a challenging problem; we have developed techniques for improving the robustness of spatial location estimation (using multi-condition training and head movements) and source type classification (using noise-adaptive linear discriminant analysis). We have also developed an i-vector approach to speaker recognition which gives very promising results. In the third layer of the blackboard system, events are interpreted to derive a meaningful description of the auditory scene. We aim to achieve this within a graphical model framework, based on the open-source software toolkit GMTK. To date, we have demonstrated how sound source localisation can be cast within this framework.

2 Introduction

The main objective of this work package is to implement an expert system using a graphical-model-based architecture on top of the monaural and binaural models provided by WP2. The blackboard system is able to interpret complex acoustic environments containing multiple sound sources, via the use of a multi-layered architecture connected by feedback pathways. In the first layer, the auditory scene is pre-segmented into different streams that correspond to sound sources and the acoustic background. In the second (event-expert) layer, information about foreground streams is used to define auditory events. These are semantically interpreted and annotated, creating a first symbolic scene description. In the third layer, symbolic information about multiple auditory events is combined in a graphical model architecture. Sounds events are considered in context in order to disambiguate their interpretation and assign meaning to the auditory scene. The flexible structure of the blackboard architecture also allows for the inclusion of knowledge from other modalities (vision and motor control).

2.1 Structure of this report

The main purpose of this report is to describe the initial implementation of the Two!Ears software architecture. The report is structured in three main sections, which document our progress against tasks in the work plan for work package three (WP3) as described below:

Software architecture (Task 3.1). This section describes recent progress on the implementation of a blackboard-based software architecture for auditory scene understanding. Following an overview of the blackboard architecture and a discussion of the motivation for it, we describe the developments that have been made to the blackboard system after the initial design and implementation described in the 6-month report (Deliverable D3.1). These focus on more sophisticated ways to dynamically request signals and store data, and an improved scheduler. A summary of the knowledge sources currently implemented within the blackboard system is given.

Pre-segmentation and tracking (Task 3.2). Here we describe approaches to pre-segmentation and tracking in the first layer of the blackboard. Techniques for fore-

ground/background classification, onset/offset detection, segmentation based on periodicity and amplitude modulation, and nonlinear tracking have been developed. We also report preliminary work on visual processing, which has been enabled by the implementation of a visual simulator. Planned work on audio-visual speaker identification is briefly described.

Formation of auditory objects (Tasks 3.3, 3.4 & 3.5). This section describes progress on the second layer of the blackboard, which concerns source models and predictors for assigning attributes to sound events. We describe a system for sound localisation that is robust to reverberation and uses human-like head movements to resolve front-back confusions. Machine learning techniques have been applied to identify different source types (e.g., 'female speech', 'fire'), using both discriminative classifiers and probabilistic models. Also, an i-vector approach has been developed for determining the identify of different speakers.

2.2 Major achievements in this period

The main achievements over the first 12 months of this work package can be summarised as follows:

- **Specification, design and implementation of a flexible, dynamic and maintainable blackboard architecture for auditory scene understanding**. The blackboard system is tightly integrated with the outputs of other work packages; it interfaces with the auralization environment developed in WP1, is integrated with the peripheral processing and feature extraction stages of WP2, and provides an interface for implementing feedback mechanisms that are addressed in WP4.
- **Methods for pre-segmenting the acoustic input**, based on pitch, onsets/offsets and the amplitude modulation spectrum.
- **A framework for nonlinear tracking of sound sources**, which can be integrated with the blackboard architecture and allows the motion dynamics of a source to be modelled as dynamical system.
- **Implementation of a 3D virtual test environment**, the Bochum Experimental Feedback Testbed (BEFT), which mimics feedback-relevant parts of the Two!Ears system and thus allows for early testing of feedback mechanisms.
- **A machine-hearing system for sound localisation**, implemented within the blackboard architecture, that employs head movements and multi-condition training to achieve robust performance in reverberant environments.

- **Building a flexible pipeline for model training**, which is able to efficiently create multi-conditional auditory scene data via the WP1 acoustic front-end and WP2 auditory front-end, and facilitates experimentation with different feature creation and model training methods via exchangeable modules. This pipeline will allow for rapid creation and testing of robust methods for auditory scene description.
- **Classifiers for source type**, based on a discriminative approach (support vector machines) and a probabilistic modelling approach (Gaussian mixture models). Source identification performance has been optimised using feature selection (by mixture of factors analysis), and a novel noise-adaptive linear discriminant analysis (NALDA) technique.
- **Speaker identification techniques** based on Gaussian mixture models with a universal background model (GMM-UBM), and using an i-vector approach to avoid undue influence of the recording channel and room acoustics.

3 Software architecture

This chapter documents the design of the Two!Ears software architecture and describes the motivation for the approach taken.

3.1 Overview of software architecture

The goal of the Two!Ears project is to develop an intelligent, active computational model of auditory perception and experience in a multi-modal context. In order to do so, the system must be able to recognise acoustic sources and optical objects, and achieve perceptual organisation of sound in the same manner as human listeners do. Bregman (1990) has referred to the latter phenomenon as auditory scene analysis (ASA), and to reproduce this ability in a machine system a number of factors must be considered:

- ASA involves diverse sources of knowledge, including both primitive (innate) grouping heuristics and schema-driven (learned) grouping principles;
- Solving the ASA problem requires the close interaction of top-down and bottom-up processes through feedback loops;
- Auditory processing is flexible, adaptive, opportunistic and context-dependent.

The characteristics of ASA are well-matched to those of *blackboard* problem-solving architectures. A blackboard system consists of a group of independent experts ('knowledge sources') that communicate by reading and writing data on a globally-accessible data structure ('blackboard'). The blackboard is typically divided into layers, corresponding to data, hypotheses and partial solutions at different levels of abstraction. Given the contents of the blackboard, each knowledge source indicates the actions that it would like to perform; these actions are then coordinated by a scheduler, which determines the order in which actions will be carried out.

Blackboard systems were introduced by Erman *et al.* (1980) as an architecture for speech understanding, in their Hearsay-II system. In the 1990s, a number of authors described blackboard-based systems for machine hearing (Cooke *et al.*, 1993, Lesser *et al.*, 1995, Ellis, 1996, Godsmark and Brown, 1999). All of these systems were in most respects 'conventional'

blackboard architectures, in which the knowledge sources employed rule-based heuristics. In contrast, the TWO!EARS architecture aims to exploit recent developments in machine learning, by combining the flexibility of a blackboard architecture with powerful learning algorithms afforded by probabilistic graphical models.

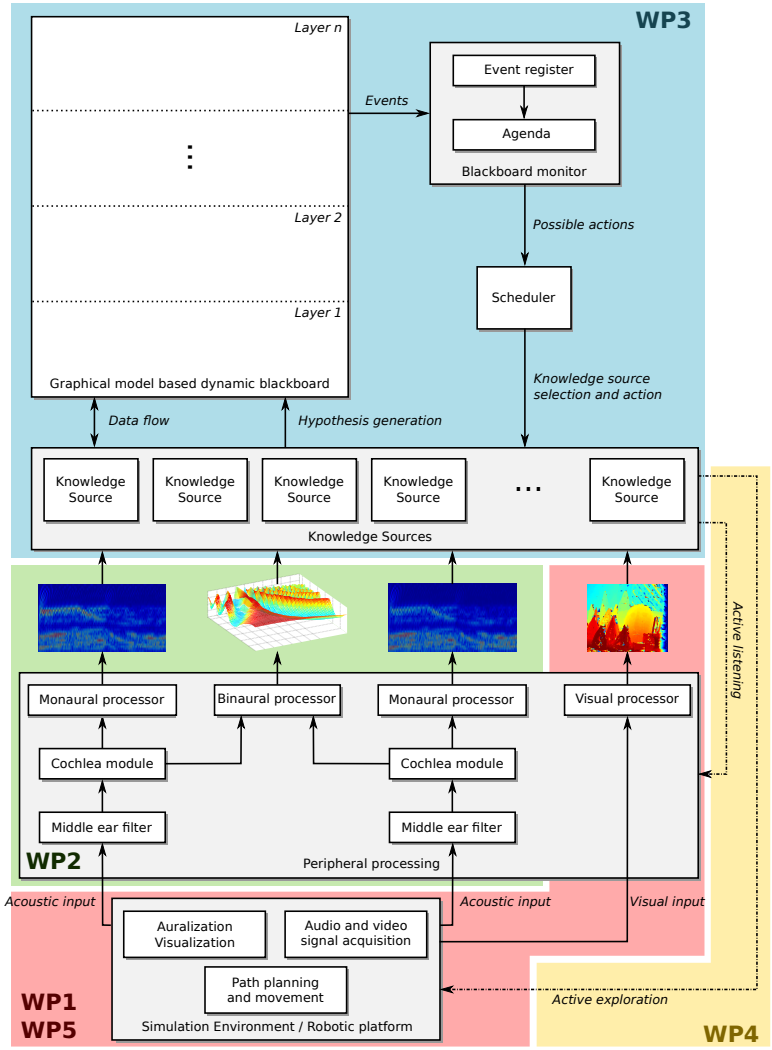


Figure 3.1: Overview of the general software architecture across all work-packages.

The general structure of the complete TWO!EARS software architecture is shown in Figure 3.1. Since Deliverable D3.1, major progress has been made towards integration of all related software modules across work packages. The current system includes the auralization environment developed by work package one (WP1), which is interfaced with the peripheral processing and feature extraction stage of work package two (WP2). The

blackboard system, developed by WP3, acts as the core of the integrated framework and provides additional interfaces to include feedback mechanisms investigated within work package four (WP4) and a robotic interface to communicate with work package five (WP5). Recent development on the software architecture is focused on integrating simulated visual processing, based on the (MORSE, 2014) simulator provided by WP5, into the framework. This will provide capabilities necessary to simulate acoustic and visual scenes that will be investigated within the TWO!EARS application scenarios.

In the following sections, we describe in more detail the progress on the design and implementation of the integrated framework as well as individual knowledge sources.

3.2 Progress on development of the TWO!EARS system

The TWO!EARS computational framework is targeted as the front-end for a great variety of applications, providing an architecture that integrates experience formation and active behaviour from a set of different functional modules. These modules can work on different levels of abstraction, independently from each other or in "collaboration", in a bottom-up or top-down manner. A key feature of this system should be its ability to evolve during (and after) the project lifetime, so that easy modification, exchange and/or extension of modules can be achieved within a scalable architecture.

To achieve the above mentioned goals, recent software development of the TWO!EARS architecture has focused on implementing more dynamic behaviour, construction and communications.

3.2.1 TWO!EARS system architectural considerations

In order to implement this integrative and system-wide view, some core attributes of the system have been established as follows.

Building a flexible system The system we develop is a *platform*, i.e. it provides functionality to execute other functionality. While the "target" functionality is clear – auditory and multimodal experience formation, scene understanding and exploration – it involves many different problems, each with many possible solutions. We therefore design the TWO!EARS system with extension in mind, trying not to constrain possible functionality of modules.

In particular, the blackboard system allows the "plugging-in" of different *knowledge sources*. Knowledge sources are modules that define their own functionality, to be executed in

the organized frame of our system. They define by themselves which data they need for execution and which data they produce – the blackboard system provides the tools for requesting and storing this data, but does not care about the actual contents (while the knowledge sources do not need to care about where and how data is stored). It is also important that the blackboard system has no static knowledge of what types of knowledge sources are available. So long as knowledge sources follow a certain implementation scheme, independent of their actual functionality they can register *dynamically* (i.e. at runtime) as a module in the blackboard system. Thus, a library of knowledge sources can be built during this project that can be extended arbitrarily, without need to modify the blackboard system. Implementors of new modules need only be concerned with implementing their functionality.

The Two!Ears architecture has been designed and implemented using an object-oriented approach. Accordingly, the "implementation scheme" knowledge sources must adhere to is provided in the form of an abstract class (see Section 3.3.1). Additionally, to enable creation of new knowledge sources that depend on auditory signals without needing to hard-code a signal request into the Two!Ears system, an "auditory front-end dependent knowledge source superclass" has been developed (see Section 3.3.3).

Building a dynamic system Key to providing the described flexibility is to neither hard-code lists of usable knowledge sources nor the interactions between them. Hard-coded (or static) lists and dependencies would be overly restrictive – the system must be open to dynamic change.

At the same time, flexibility for extension is not the only cause for needing a dynamic system. The Two!Ears system is intended to be an *active* system that does not only work in a signal processing bottom-up manner, but also in a "cognitive" top-down manner. Modules must therefore be allowed to change the system setup at runtime. This means that it is essential for our system to be equipped with functionality for *dynamic* module instantiation, registration and removal. This also implies the need for on-the-fly rewiring of the communication links between modules.

Building a usable system The Two!Ears architecture is a platform for integrating the modules produced by different partners of the Two!Ears consortium. Also, the platform will be open to the public, empowering further research. It is thus very important to make the system *usable* and easy to configure for different tasks, and we have put great emphasis on writing clean program code: to keep classes and methods small and understandable, clearly separate responsibilities between classes, make use of inheritance and abstract classes, and encapsulation. As much of the "system machinery" as possible is hidden from the user while providing an open and easy-to-use functional interface.

3.2.2 Dynamic system construction

To ensure an easy-to-use system, we implemented as the main class a wrapper that integrates the different main components, and hides their connections where possible. This main class is called **BlackboardSystem**, since the blackboard is the central component of our platform. This is an excerpt of its definition:

```
class BlackboardSystem
    properties
        blackboard;
        blackboardMonitor;
        scheduler;
        robotConnect;
        dataConnect;
    methods
        BlackboardSystem()
        setRobotConnect( robotConnect )
        setDataConnect( connectorClassName )
        buildFromXml( xmlName )
        addKS( ks )
        createKS( ksClassName, ksConstructArgs )
        numKSs()
        run()
```

The "engine" of our system is distributed across the `BlackboardSystem`, `Blackboard`, `BlackboardMonitor` and `Scheduler` classes, with the `BlackboardSystem` class holding instances of the latter three. These four classes each have genuine responsibilities: the `BlackboardSystem` integrates the framework parts, responsible for constructing and setting up the system. The blackboard is the central storage of *functional* data and knowledge sources. It holds a data map that saves arbitrary knowledge source data along time, together with methods to add and recall data from within knowledge source code. Additionally, the knowledge sources themselves are put into blackboard storage by the `BlackboardSystem`.

The `BlackboardMonitor` is responsible for creating bindings on demand between knowledge sources, by instantiating event listeners. It keeps track of these bindings and maintains the agenda of knowledge sources. The `Scheduler` is the executive component of the system. While the `BlackboardMonitor` keeps control of the knowledge sources in the agenda, the `Scheduler` decides about the order of those knowledge sources to be executed. It does that based on the attentional priorities of the knowledge sources. Figure 3.2 shows the system class diagram.

Several core functionalities are provided through the `BlackboardSystem` class:

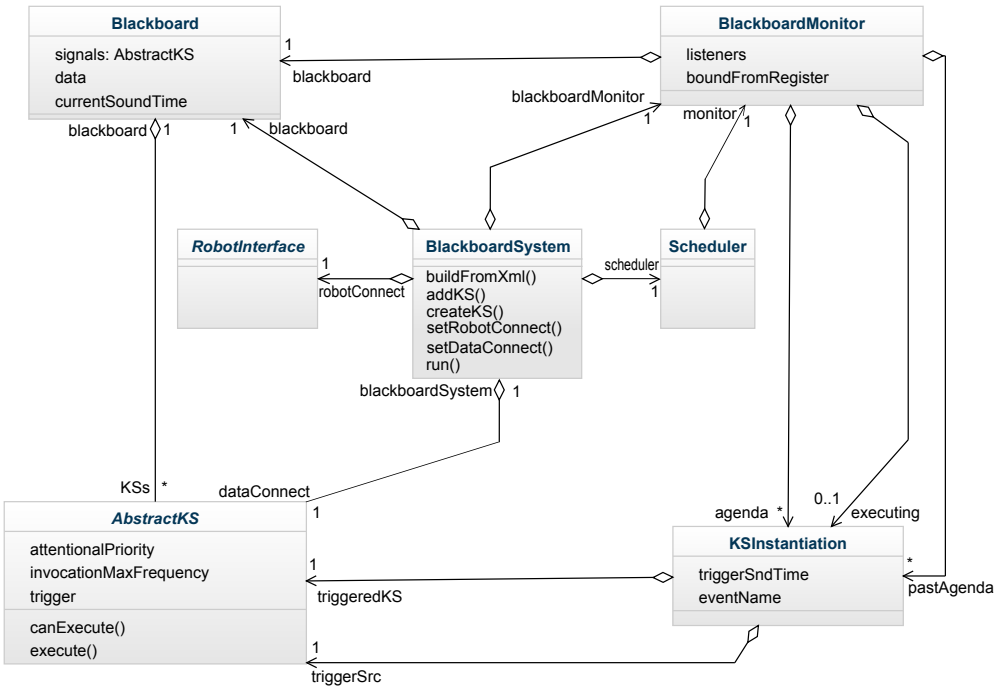


Figure 3.2: Class diagram of the whole blackboard system. The BlackboardSystem class is the integrative system component holding the other modules and giving access to system functionality.

- Connecting the robot/binaural simulator (see Deliverable D6.1) to the blackboard system. The connected object must implement the robot interface for delivering audio "ear" signals and commanding movement and head rotation. The blackboard system and all its components including the knowledge sources get access to the audio stream and robot actions through this connection (`setRobotConnect` method).

- Setting the type of the module that integrates with the auditory front-end (AFE; see Deliverable 2.2), instantiating it and connecting it to the blackboard system. This module is a knowledge source itself, responsible for processing the ear signals into cues (such as interaural time differences) needed by other knowledge sources. The AFE itself is connected to the robot/binaural simulator in order to obtain the ear signals. See also Section 3.3.2 (`setDataConnect` method).

- Instantiating and adding knowledge sources. These knowledge sources must inherit from `AbstractKS` (see Section 3.3.1) or `AuditoryFrontEndDepKS` (see Section 3.3.3) to be able to be interfaced and run by the system. Knowledge sources that inherit from `AuditoryFrontEndDepKS` automatically get connected with the AFE by the system in order to place their signal/cue requests.

```
<blackboardsystem>
    <dataConnection Type="AuditoryFrontEndKS"/>

    <KS Name="baby" Type="IdentityKS">
        <Param Type="char">baby</Param>
        <Param Type="char">6687829ce1a73694a1ce41c7c01dec1b</Param>
    </KS>
    <KS Name="femaleSpeech" Type="IdentityKS">
        <Param Type="char">femaleSpeech</Param>
        <Param Type="char">6687829ce1a73694a1ce41c7c01dec1b</Param>
    </KS>
    <KS Name="idDec" Type="IdDecisionKS">
        <Param Type="int">0</Param>
        <Param Type="int">1</Param>
    </KS>

    <Connection Mode="replaceOld" Event="AgendaEmpty">
        <source>scheduler</source>
        <sink>dataConnect</sink>
    </Connection>
    <Connection Mode="replaceOld">
        <source>dataConnect</source>
        <sink>baby</sink>
        <sink>femaleSpeech</sink>
    </Connection>
    <Connection Mode="replaceParallel">
        <source>baby</source>
        <source>femaleSpeech</source>
        <sink>idDec</sink>
    </Connection>
</blackboardsystem>
```

Figure 3.3: an example of an XML-configured blackboard system. Two identity knowledge sources are connected to the AFE, and triggering an identity decision knowledge source.

Adding and instantiating knowledge sources can take place both before or *while* running the system; it can be done from "outside" the system *or from inside knowledge sources*. This enables the development of top-down controlling knowledge sources from higher "cognitive" experts running in the system (`addKS` or `createKS` method).

- The start-up configuration of the system can completely be defined by an XML file; the system is then constructed before running by loading this file. Of course this configuration can be changed dynamically while executing the system. The XML description needs at least a `dataConnection` node specifying the type of the AFE module; then, it can also contain `KS` nodes with parameters to construct knowledge

sources, and `Connection` nodes that specify event bindings between knowledge sources. See Figure 3.3 for an example (`buildFromXml` method).

- Starting the execution of the system. This triggers the blackboard system to request data from the robot/binaural simulator connection, and subsequent action by the AFE and all knowledge sources that are connected. The system will not stop execution before the robot/binaural simulator sends an ending signal (`run` method).

3.2.3 Dynamic blackboard memory

The `Blackboard` class holds the central data repository of the platform. It stores the knowledge sources and any shared data, in particular the output of the knowledge sources (e.g., estimates of the location of a sound source). It is accessible to all knowledge sources; and it not only stores current data, but keeps track of the history of this data in order to enable knowledge sources to work on time series data.

Importantly, the `Blackboard` is flexible about data categories, which do not have to be hard-coded into the system. Knowledge sources "decide" on their own and at runtime what to add and what to request. Thus, the system does not need to be changed in order to implement new knowledge sources that work with new data categories. Of course, knowledge sources can only read data categories that are actually stored in the blackboard by other knowledge sources (or themselves).

The following listing shows an excerpt of the Blackboard interface:

```
class Blackboard
        KSs;
        signals;
        currentSoundTime;
    methods
        Blackboard()
        addData( dataLabel, data, append, time )
        getData( dataLabel, reqSndTime )
        getLastData( dataLabel, time )
        getNextData( dataLabel, time )
        getDataBlock( dataLabel, blockSize_s )
```

Prominently featured are methods to add and access data:

- `addData` lets knowledge sources add data to the blackboard storage. The data category has to be named in `dataLabel`, `data` hands over the actual data to store. `append` is an optional flag indicating whether to overwrite or append data at the same time step (there might, for example, be several source identity hypotheses per

timestep, but only one source number hypothesis might be allowed). `time` specifies the time point under which this data shall be stored. It is optional and, if not set, defaults to the current time.

- `getData` lets knowledge sources read data from the blackboard storage. `dataLabel` indicates the data category requested, `reqSndTime` the time point of interest. `getLastData`, `getNextData` and `getDataBlock` are special cases of `getData` for retrieving the last data, the next data after a particular point in time, or a whole data block of length `blockSize_s`.

The following is an example from the implementation of the `IdDecisionKS` class:

```
idHyps = obj.blackboard.getData( ...
             'identityHypotheses', obj.trigger.tmIdx ).data;
%...
%find the most likely identity hypothesis -> maxProbHyp
%...
obj.blackboard.addData( ...
    'identityDecision', maxProbHyp, false, obj.trigger.tmIdx );
```

Additionally, the blackboard is used as a storage for pointers to signals from the auditory front-end requested by knowledge sources inheriting from `AuditoryFrontEndDepKS`. The actual memory in which these signals are stored for recall is implemented in the auditory front-end through cyclic buffers (see Deliverable D2.2).

3.2.4 Dynamic blackboard interactions

Knowledge sources can communicate information through the flexible blackboard storage. However, adding data to the blackboard does not trigger other knowledge sources to be executed. Such interaction – triggering knowledge source execution – is done through an event system. Specifically, knowledge sources do not actually trigger execution of other knowledge sources (since they are decoupled and have no "knowledge" of each other), but knowledge sources *make a request to be triggered* upon the firing of particular *events*.

Each knowledge source has a standard event it can trigger, `KsFiredEvent`, inherited from `AbstractKS`. Beyond that, every knowledge source class is free to define as many additional events as reasonable for its task. Knowledge sources cause the events themselves through a call to `notify` as in the following example, in which the knowledge source induces an event and attaches a `BlackboardEventData` object holding the time that it was triggered:

```
notify( 'KsFiredEvent', BlackboardEventData(obj.trigger.tmIdx) );
```

The blackboard system a priori is totally ignorant of which events exist (clear responsibilities principle, open to extension). It also does not monitor any events by default, until knowledge sources request to be triggered by an event. This request is done through the method `bind` provided by the `BlackboardMonitor` class, whose interface is (partially) listed in the following excerpt:

```
class BlackboardMonitor
    properties
        pastAgenda;
        executing;
        agenda;
    methods
        BlackboardMonitor()
        bind( sources, sinks, addMode, eventName )
```

The `bind` method connects the `sinks` knowledge sources to event `eventName` (optional, defaults to `KsFiredEvent`) caused by the `sources` knowledge sources. `addMode` specifies how the `BlackboardMonitor` shall handle adding the triggered knowledge sources into the agenda. It understands the following modes, illustrated in Figure 3.4:

add Add the triggered knowledge source to the end of the agenda, regardless of whether or not there is already a (not yet executed) knowledge source instantiation of this sink in the agenda from a former triggering.

replaceOld Replace old knowledge source instantiations of this sink in the agenda with the new one. Only instantiations of the sink triggered by the same source and same event are replaced. This is an important mode for knowledge sources where processing current data is more important than processing all data.

replaceParallel Replace knowledge source instantiations of this sink from the same time point of "parallel" sources in the agenda with the new one. Only instantiations of the sink triggered at the same time and by the same event are replaced. This mode avoids sinks being unnecessarily executed several times with the same information.

replaceParallelOld Replace old or current knowledge source instantiations of this sink triggered by "parallel" sources in the agenda with the new one. Only instantiations of the sink triggered by the same event are replaced. This mode is a combination of the replaceOld and replaceParallel modes.

It should be noted that the `addMode` only affects triggered knowledge source instantiations in the *agenda*, i.e. those that are not executed yet. As soon as a knowledge source is executed, it is removed from the agenda (first in `executing`, afterwards in `pastAgenda`).

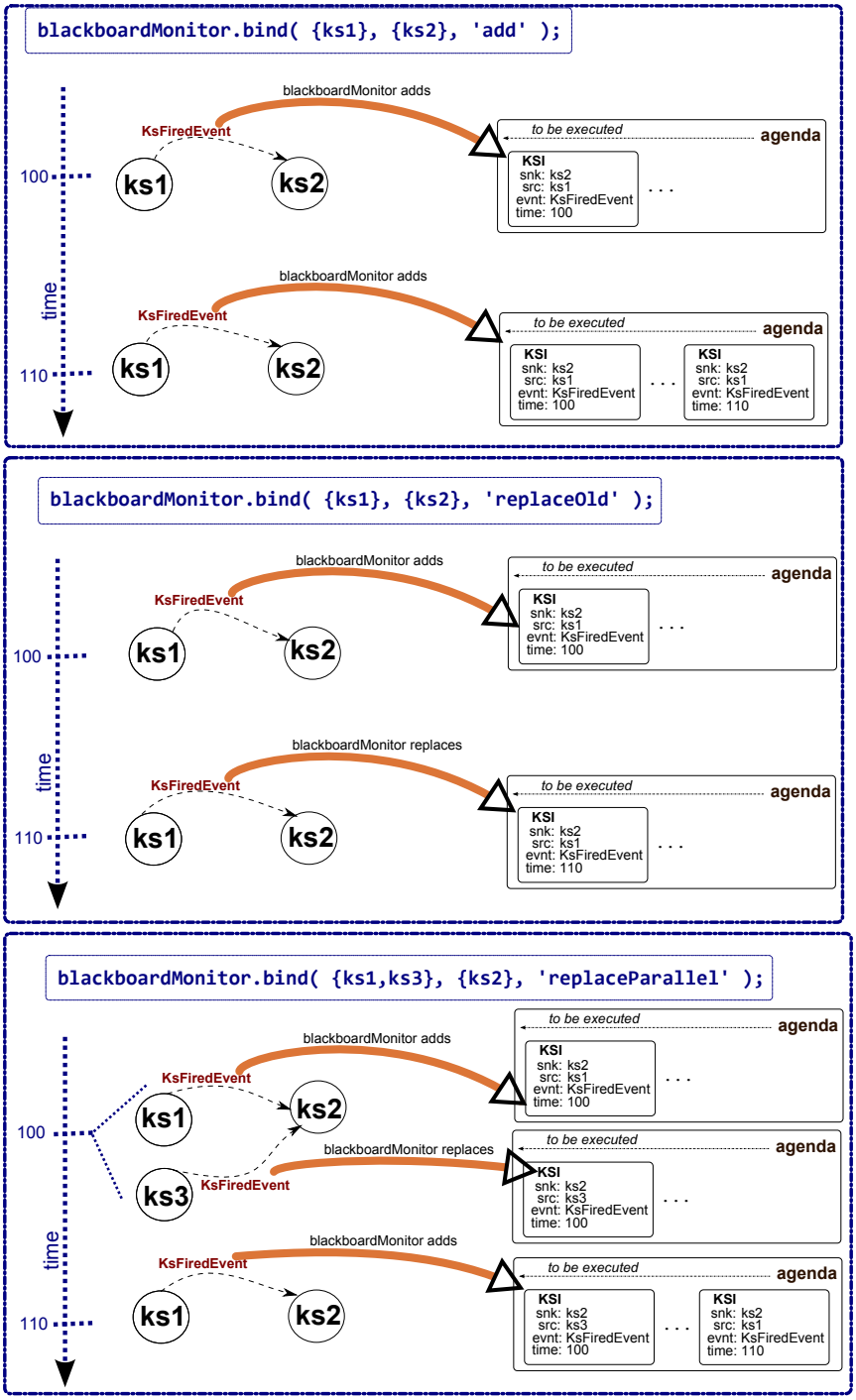


Figure 3.4: The different possibilities of event binding between knowledge sources with the blackboard system.

3.2.5 Dynamic blackboard scheduler

The scheduler is the component of the blackboard system that actually executes the knowledge sources – but first, it *schedules* them, that is, it decides the order in which knowledges sources waiting in the agenda get executed. This order is rescheduled after every execution of a knowledge source, since the conditions determining the order may have changed, or new knowledge sources may be present in the agenda that are more urgent.

The following factors influence the order of execution of knowledge sources:

- Knowledge sources have a property called *attentional priority*. Knowledge sources with higher priority get executed before knowledge sources with lower priority. This priority can be set by the knowledge source itself, by other knowledge sources or from outside the system. The `BlackboardMonitor` provides a method for setting focus on a knowledge source (increasing its priority), along with the option to propagate this higher priority down along the dependency chain of this knowledge source. The dependency chain is determined by the event bindings.
- Knowledge sources must implement a method `canExecute`, that returns whether or not the knowledge source can execute at this moment, and which is called by the scheduler if the knowledge source is first on the scheduling list. If it cannot execute, the knowledge source can decide whether to remain in the agenda or be removed from it.
- Knowledge sources define a maximum invocation frequency, that cannot be exceeded. It is a *maximum* frequency, because knowledge sources get not necessarily executed periodically, since they are triggered by events, but not by timers. The scheduler checks whether the last execution time was long enough ago before considering the knowledge source for execution. Until then, it remains in the agenda.

This listing shows the relevant parts of the interface with respect to influencing the scheduling:

```
class BlackboardMonitor
    methods
        focusOn( ks, propagateDown )
        resetFocus()

class AbstractKS
    properties
        invocationMaxFrequency_Hz;
    methods (Abstract)
        canExecute()
        execute()
```

```
    methods
        focus()
        unfocus()
```

Relationship to attention Since we are trying to implement an active auditory system in TWO!EARS, *attention* is a concept of particular interest. Scheduling modules by means of priorities seems could be viewed as one part of directing attention to specific tasks, since it means assigning more computing resources to preferred knowledge sources.

3.2.6 Integration of work across work packages

The blackboard system currently integrates modules from different work packages, in particular the acoustical scene auralization environment, the auditory front-end, and functional modules for auditory object formation. Since the acoustical scene auralization environment is implemented and connected to the blackboard system through the robot interface, instead it would be possible to connect the robot to the system, which then can – controlled by knowledge sources – actively explore the environment. Furthermore, by providing a modular system that is easy to configure, the blackboard can now be used for simulating and testing many different auditory scenarios, modules and tasks.

3.2.7 Prolog interface

The current blackboard system is capable of representing problems and tasks via a flexible probabilistic framework. This can be considered as a powerful extension of conventional blackboard designs, which are restricted to a rule-based processing paradigm (e.g., Erman *et al.*, 1980). However, even though a probabilistic representation provides a great amount of flexibility, specific problems can be more efficiently represented by a set of rules, applied to a corresponding knowledge base. This is especially important if additional knowledge provided by human experts should be integrated into the system.

Therefore, an integration of the PROLOG (**Pro**gramming in **Log**ic) programming language into the blackboard system is currently under development. PROLOG is widely used in several fields of artificial intelligence research and provides a flexible way to set-up and query large knowledge bases by a set of prespecified rules.

As opposed to imperative or object-oriented languages like C or PYTHON, PROLOG is considered a declarative programming language. This means that, instead of specifying how a certain goal should be achieved, the PROLOG interpreter derives a solution from a

query, describing the goal itself as a hypothesis that should be confirmed. Therefore, it is necessary to provide a knowledge base, containing a set of facts and rules which describe the possible solution space. The PROLOG interpreter then performs a depth-first-search (Russell and Norvig, 2003, Ch. 3) within the solution space to either confirm or reject the hypothesis that was described by the query.

The interface that is currently implemented is based on SWI-Prolog (Wielemaker *et al.*, 2012), an open-source PROLOG programming environment and interpreter. Recent developments focus on the integration of this environment into MATLAB, allowing direct access to the interpreter from within the TWO!EARS software architecture. For the next reporting period it is planned to provide an easy to use wrapper class which encapsulates all necessary SWI-Prolog functions and makes them available to the blackboard system.

3.3 Progress on implementation of knowledge sources

In this section we describe progress on software implementation of individual knowledge sources within the TWO!EARS framework. The techniques employed by the knowledge sources are discussed in more details in the following chapters.

3.3.1 Abstract knowledge source

The abstract knowledge source (`AbstractKS` class) is the base class for all knowledge sources in the blackboard system. The listing below shows the parts most relevant to development of new knowledge sources:

```
class AbstractKS
    properties
        blackboard;
        blackboardSystem;
        invocationMaxFrequency_Hz;
        trigger;
    events
        KsFiredEvent
    methods (Abstract)
        canExecute()
        execute()
    methods
        focus()
        unfocus()
```

There are different aspects of functionality in this interface.

Data access Knowledge sources have a handle to the `blackboard`. Through this handle, data can be placed on and retrieved from the blackboard.

System setup Through the handle `blackboardSystem`, knowledge sources get access to the methods for adding and removing other knowledge sources, and also access to the `BlackboardMonitor`.

Execution properties The property `invocationMaxFrequency_Hz` specifies how often this knowledge source is allowed to be executed. The methods `focus` and `unfocus` give access to the attentional priority of the knowledge source, which influences its relative importance when competing for computing resources with other knowledge sources. See Section 3.2.5 for a description of scheduling.

Execution conditional The abstract method `canExecute` must be implemented by the inheriting knowledge source. It is called by the scheduler when the knowledge source is next in the schedule before actually executing. If this method returns false, execution will not be performed. The second output argument of this method indicates whether the knowledge source should remain in the agenda or be removed.

Execution The main functionality of any knowledge source is implemented in the method `execute`. A knowledge source gets executed by the scheduler if its maximum invocation frequency would not be exceeded and its `canExecute` method returns true. In this method, a knowledge source gets access to its `trigger`, a structure that contains information about the triggering event, the triggering source, and an argument the trigger source placed for usage by sinks.

Events Knowledge sources can define their own individual events. However, each class already inherits a standard event from `AbstractKS`, `KsFiredEvent`. Events can be triggered by knowledge sources via `obj.notify(`*eventname*`,` *attachedData*`)`.

3.3.2 Auditory front end (AFE) knowledge source

This knowledge source integrates the auditory front-end from WP2 into the blackboard system. The AFE itself is a self-contained module (see Deliverable 2.2 for more details) and this section focuses on its integration within the Two!Ears framework.

The AFE knowledge source is connected to the blackboard and the robot interface by registering itself in the system via `BlackboardSystem.setDataConnect`. Upon construction, the AFE `dataObject` and `managerObject` are instantiated and connected to the robot interface ear signals stream. The maximum invocation frequency of the `AuditoryFrontEndKS` is set to infinity. Execution mainly consists of getting the latest chunk of ear signals data,

processing it through the AFE, and notifying a `KsFiredEvent`.

Other knowledge sources can register requests with the AFE indirectly, through inheriting from the `AuditoryFrontEndDepKS` class (see Section 3.3.3), and binding to it's KsFiredEvent.

3.3.3 Auditory signal dependent knowledge source superclass

Whenever a knowledge source needs signals, cues or features from the auditory front-end, it should subclass from the `AuditoryFrontEndDepKS` class. Any knowledge source added to the blackboard through the BlackboardSystem `addKS` or `createKS` methods, register these AFE signal requests automatically with the AFE.

Setting up the requests Inheriting knowledge sources need to put their requests in their `constructor`, in particular in the call to the superconstructor: `obj@AuditoryFrontEndDepKS( requests )`. `requests` is a structure with a field `r` and `p`, standing for "request" and "parameters", respectively. Each of these fields is a cell array; request `r{reqIdx}` has to be accompanied by parameters `p{reqIdx}`.

Accessing signals These requested signals can then be accessed by the knowledge source via the inherited `getReqSignal( reqIdx )` method. `reqIdx` refers to the same indexes used as in the request structure.

A more elaborate description of the request parameter structure and the signal objects can be found in Deliverable 2.2. See Figure 3.5 below for an example of how to request signals from the AFE.

3.3.4 Localisation knowledge sources

Four knowledge sources work together to generate hypotheses of sound source azimuths: `Location` knowledge source, `Confusion Detection` knowledge source, `Confusion Solving` knowledge source, and `Head Rotation` knowledge source. Section 5.1 provides more details about the localisation models. In this section we focus on implementation of the knowledge sources and their usage within the blackboard framework.

Location knowledge source

Class `LocationKS` implements knowledge about the statistical relationship between spatial cues and azimuth locations. This knowledge source (KS) requires signals from the AFE

```
classdef LocationKS < AuditoryFrontEndDepKS
%...
    function obj = LocationKS()
        requests.r{1} = 'ild';
        requests.p{1} = genParStruct(...
            'f_low',80,'f_high',8000,...
            'nChannels',32,...
            'rm_decaySec',0,...
            'ild_wSizeSec',20E-3,...
            'ild_hSizeSec',10E-3,'rm_wSizeSec',20E-3,...
            'rm_hSizeSec',10E-3,'cc_wSizeSec',20E-3,...
            'cc_hSizeSec',10E-3 );
        requests.r{2} = 'itd_xcorr';
        requests.p{2} = requests.p{1};
        obj = obj@AuditoryFrontEndDepKS( requests );
        %...
    end

    function execute(obj)
        ildsSObj = obj.getReqSignal( 1 );
        itdsSObj = obj.getReqSignal( 2 );
        % ...
    end
```

Figure 3.5: An example of using inheritance from the AuditoryFrontEndDepKS to request acoustic cues from the AFE.

and thus inherits from the `AuditoryFrontEndDepKS` (Section 3.3.3, Figure 3.5) and needs to be bound to the `AuditoryFrontEndKS`'s `KsFiredEvent`. The `canExecute` precondition checks the energy level of the current signal block and localisation takes place only if there is an actual auditory event. After execution, a `LocationHypothesis` containing a probability distribution of azimuth locations is placed on the blackboard (category "`locationHypotheses`") and the event `KsFiredEvent` is notified.

binds to	`AuditoryFrontEndKS.KsFiredEvent`
writes data category	`locationHypotheses`
triggers event	`KsFiredEvent`

Confusion Detection knowledge source

The `ConfusionDetectionKS` checks new location hypotheses and decides whether there is a confusion. A confusion emerges when there are more valid locations in the hypotheses than assumed auditory sources in the scene. In case of a confusion, a `ConfusedLocations`

event is notified and the responsible location hypothesis is placed on the blackboard in the `confusionHypotheses` category. Otherwise, a `PerceivedLocation` object is added to the blackboard `perceivedLocations` data category, and the standard event is triggered.

binds to	`LocationKS.KsFiredEvent`
reads data category	`locationHyptheses`
writes data category	`confusionHyptheses` or `perceivedLocations`
triggers event	`ConfusedLocations` or `KsFiredEvent`

Confusion Solving knowledge source

The `ConfusionSolvingKS` solves localisation confusions by predicting the location probability distribution after head rotation, and comparing it with new location hypotheses received after head rotation is completed. The `canExecute` method will wait for new location hypotheses; when there is one, it will check whether the head has been turned, otherwise it will not execute. The confusion is then solved by using the old and the new location hypothesis, and a `PerceivedLocation` object is placed on the blackboard.

binds to	`ConfusionDetectionKS.ConfusedLocations`
reads data category	`confusionHypotheses`, `headOrientation` and `locationHypotheses`
writes data category	`perceivedLocations`
triggers event	`KsFiredEvent`

Head Rotation knowledge source

The `HeadRotationKS` has knowledge on how to move the robotic head in order to solve confusions in source localisation. If there is no other head rotation already scheduled, the KS uses the `robot` interface to turn the head.

binds to	`ConfusionDetectionKS.ConfusedLocations`
reads data category	`confusionHypotheses`, `headOrientation`
writes data category	`headOrientation`

3.3.5 Identification knowledge sources

This section focuses on implementation of sound identification knowledge sources within the blackboard framework. See Section 5.2 for more details of the models used to estimate the type of auditory events.

Identity knowledge source

Objects of class `IdentityKS` implement source models, by incorporating an instance of a model (`IdModelInterface` class) with knowledge about the connection of acoustical cues and certain sound source types. Many Identity KSs can be used concurrently. Currently for each sound class to be identified there exists an object of class `IdentityKS`. The model object of `IdentityKS` can in theory employ any kind of models, such as a linear support vector machine, or a Gaussian mixture model. The `IdentityKS` needs access to AFE signals, thus it is a subclass of `AuditoryFrontEndDepKS` (see Section 3.3.3). The model object holds the signal request structure.

The KS predicts, based on the incorporated source model, whether the currently received auditory stream includes an auditory object of the sound type it represents.

binds to	`AuditoryFrontEndKS.KsFiredEvent`
writes data category	`identityHypotheses`
triggers event	`KsFiredEvent`

Identity decision knowledge source

The identity knowledge source checks new identity hypotheses. It then decides which of them are valid, by comparison and incorporating knowledge about the number of assumed auditory objects in the scene.

binds to	`IdentityKS.KsFiredEvent`
reads data category	`identityHypotheses`
writes data category	`identityDecision`
triggers event	`KsFiredEvent`

3.4 Conclusion

In this chapter, we have presented the Two!Ears software architecture that was designed and developed during the first year of the project. We have shown that the current framework is capable of serving as a flexible computational model that allows testing and evaluation of auditory perception and experience in a multi-modal context. The core-component of the software framework, namely the blackboard system, was extended in a way that allows the dynamic creation of task-related configurations. Furthermore, we have presented the embedding of the blackboard system into the software architecture across work packages, combining acoustic scene auralization, auditory processing, feature extraction and possibilities for high level feedback through interactions with the robot

interface. Additionally, we have introduced further extensions that are currently under development, namely the audio-visual interface and logical rule learning via the Prolog programming language.

4 Pre-segmentation and tracking

The chapter reports progress in Task 3.2, which concerns pre-segmentation and tracking. Techniques have been developed for foreground/background classification, onset/offset detection and nonlinear tracking. Currently, these techniques are evaluated in isolation; work in the next period will consider how they are best combined within the blackboard architecture. We also report preliminary work on visual processing, which has been enabled by the implementation of a visual simulator. Planned work on audio-visual speaker identification is also briefly described.

4.1 Methods for audio segmentation

Sound perception studies of human listeners suggest that there are processes in the auditory system which segregate the acoustic evidence into perceptual streams based on their characteristics. This allows listeners to selectively attend to particular streams of interest (Bregman, 1990). Inspired by perceptual studies, the pre-segmentation stage of the Two!Ears architecture implements algorithms that pre-segment the auditory scene into different auditory streams corresponding to foreground sound sources and noisy background. We use the term *fragments* to refer to regions in the spectro-temporal domain where the energy is dominated by a single acoustic source.

In this section we investigate several pre-segmentation methods based on various grouping cues. Although most of the methods work on individual cues, they can be integrated to provide more robust pre-segmentation. Cue integration will be investigated in the next period of the Two!Ears project.

4.1.1 Grouping based on periodicity

It is believed that listeners' ability of auditory scene analysis is underlain by many auditory grouping cues. Among many, grouping by periodicity appears to be one of the most robust grouping cues, and has been widely employed in machine-hearing systems (Wang and Brown, 2006). In this section we investigate pre-segmentation models based on periodicity.

Autocorrelogram

The periodicity of sound is well represented by the autocorrelogram (ACG), or simply *correlogram*. The correlogram is a three-dimensional volumetric function, mapping a frequency channel of an auditory periphery model, temporal autocorrelation delay (or lag), and time to the amount of periodic energy in that channel at that delay and time. If the original sound contains a signal that is approximately periodic, such as voiced speech, then each frequency channel excited by that signal will have a high similarity to itself delayed by the period of repetition. This can be emphasised by summing the ACG over all frequency channels, producing a 'summary ACG' (see bottom panels in Figure 4.1). The position of the largest peak in the summary ACG corresponds to the pitch of the periodic sound source. Primarily because it is well-suited to detecting signal periodicity, the correlogram is widely considered as the preferred computational representation of early sound processing in the auditory system (Slaney and Lyon, 1990, Meddis and Hewitt, 1991, Assmann and Summerfield, 1990, Meddis and Hewitt, 1992).

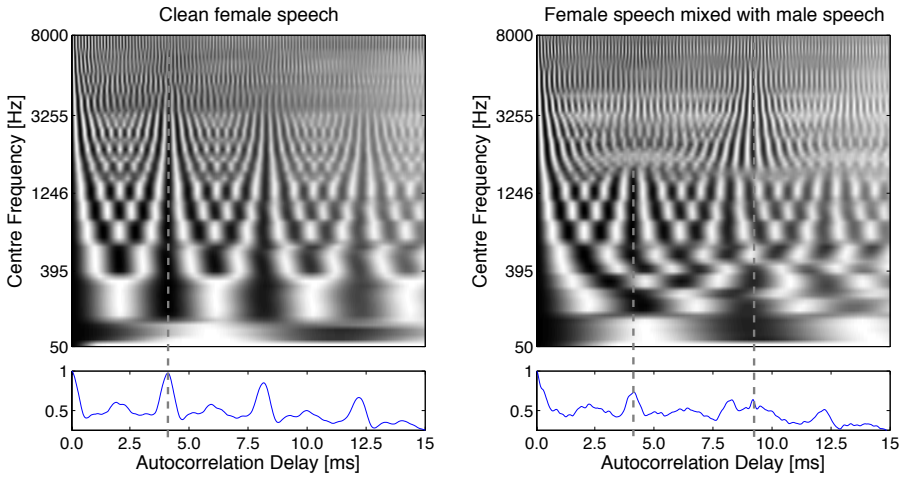


Figure 4.1: Correlograms computed at different times for a clean speech signal uttered by a female speaker. Each correlogram has been normalised and plotted as an image. The corresponding summary ACG is shown at the bottom of each correlogram. Dotted lines indicate the pitch period of a sound source.

Simultaneous grouping

Simultaneous grouping involves organising time-frequency (T-F) components of acoustic mixtures across frequency. For a periodic sound source all autocorrelation channels exhibit large peaks corresponding to the fundamental frequency (F0), forming vertical stems in the correlogram centred on the delays corresponding to multiple pitch periods. Meanwhile,

because each filter channel also actively responds to the harmonic component that is closest to its centre frequency (CF), each channel tends to repeat itself at an interval of approximately 1/CF, giving a succession of peaks at approximately the period of the CF in the correlogram. This produces symmetric tree-like structures appearing at intervals of the pitch period in the correlogram. When only one harmonic source is present, the stem extends across the entire frequency range (see the left panel in Figure 4.1). When a competing sound source is also present, some ACG channels may be dominated by the energy that has arisen from the competing source, causing a gap in the stem of the structure corresponding to the target source's pitch (see the right panel in Figure 4.1). Such ACG structures can be extracted and used to form spectral groups from a simultaneous sound mixture (Summerfield *et al.*, 1990, Ma *et al.*, 2007).

Sequential grouping

Fragments can be formed by sequentially linking spectral groups based on pitch continuity. Spectral groups that are likely to belong to the same source tend to have pitch estimates that form a smooth temporal pitch contour. A simple rule-based tracker is used to form potentially overlapping pitch track segments that extend through time (Ma *et al.*, 2007). Each pitch track is then used to recruit a spectro-temporal fragment. In each time frame, the frequency channels that have autocorrelogram peaks corresponding to the pitch track value are recruited into that track's fragment. When more than one pitch track is simultaneously active, channels are assigned to tracks according to which pitch track best explains the autocorrelogram peaks. This process is illustrated in Figure 4.2.

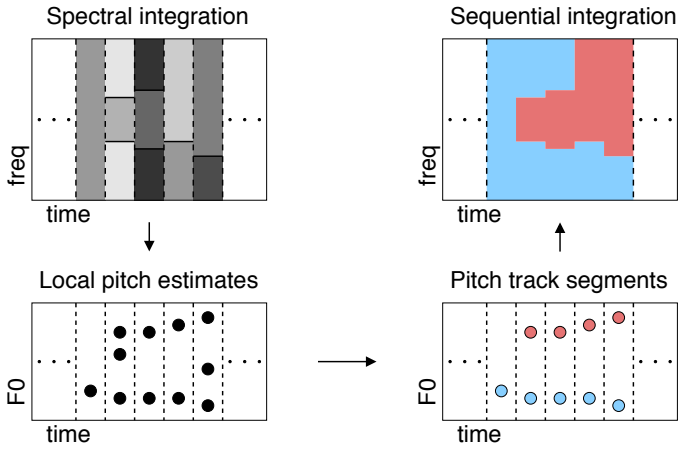


Figure 4.2: Upper-left panel: different shades of grey represent different spectral groups in each frame. Lower-left panel: dots are local pitch estimates for the spectral groups. Lower-right panel: two pitch track segments are produced by linking the local pitch estimates. Upper-right panel: two fragments are formed corresponding to the two pitch track segments.

As an example of the described method, Figure 4.3 shows fragments generated for a mixture of male/female speech signals. Here the male speaker and the female speaker are equally loud and overlap in time. In panel (b) the 'oracle' (ground-truth) segmentation of the two speakers is indicated by using different colours. Panels (c–d) show estimated pitch tracks of the two overlapping speakers. Panel (e) shows the final fragments after sequential grouping. It should be noted that in the pre-segmentation stage the fragments simply correspond to acoustic events and their identities are not known.

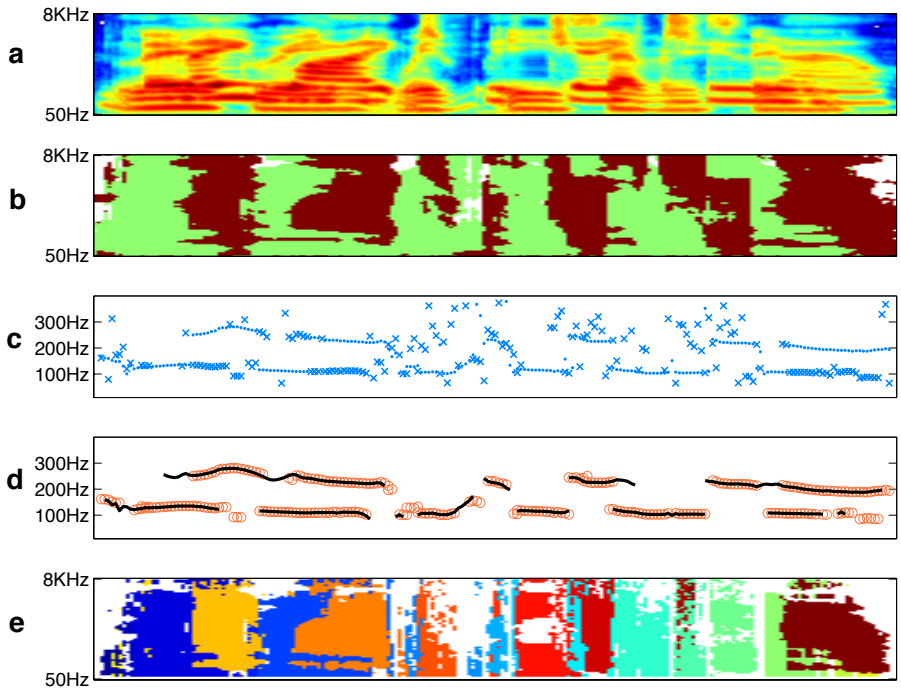


Figure 4.3: (a) Cochleagram of a male/female speech mixture (SNR=0 dB). (b) The 'oracle' segmentation. Brown region: pixels where the value in the mixture is close to that of the female speaker; green region: the mixture value is close to that of the male speaker; white: low energy regions. (c) Pitch estimates of simultaneous sources. Dots represent pitch estimates of the stronger source in each frame and crosses represent the weaker source. (d) Circles are pitch tracks produced by the multipitch tracking algorithm; solid lines show pitch tracks estimated from pre-mixed signals. (e) Fragments after sequential grouping.

Integration with other grouping cues

Grouping by periodicity has been shown to be a robust cue in the presence of multiple sources and reverberation. Other cues, such as source location, are often severely degraded in such adverse listening environments. Therefore fragments generated based on periodicity provide a useful framework to integrate other cues. For example, Christensen *et al.* (2009)

showed that integration of localisation cues within a fragment identified by periodicity relations can improve localisation performance significantly in reverberant conditions. In (Ma *et al.*, 2013), location cues and pitch cues are integrated within a fragment-based model to improve automatic speech recognition with reverberation and multiple sound sources. We plan in the near future to further investigate integration of different grouping cues based on the current fragments model within the Two!Ears framework.

4.1.2 Grouping based on amplitude modulation features

One of the most striking abilities of the human auditory system is the capability to focus on a desired target source and to segregate it from interfering background noise. Despite substantial progress in the field of computational auditory scene analysis (CASA) over the past decades, machine-based approaches that attempt to replicate human speech recognition abilities are still far away from being as robust as humans against the detrimental influence of competing sources and interfering noise.

Assuming *a priori* knowledge about the energy of the target source and all interfering sources in individual time-frequency (T-F) units, the concept of the Ideal binary mask (IBM) has been introduced, where the time-frequency representation of noisy speech is classified into either target-dominated or interference-dominated T-F units (Wang, 2005). This classification is commonly derived by comparing the signal-to-noise ratio (SNR) in individual T-F units to a local criterion (LC). T-F units with an SNR above the predefined LC threshold are considered reliable and subsequently labeled as 1. All remaining T-F units are assumed to be dominated by noise and therefore labeled as 0. The resulting IBM can be interpreted as the ideal segregation and many studies have shown its potential for a wide range of applications, including speech intelligibility in noise (Brungart *et al.*, 2006, Li and Loizou, 2008, Kjems *et al.*, 2009, Wang *et al.*, 2009), automatic speech recognition (Cooke *et al.*, 2001) and speaker identification (May *et al.*, 2012a,b). However, the IBM is not available in practice and, therefore, its estimation in realistic scenarios is one of the key challenges of CASA, e.g. in connection to applications in hearing aids and communication devices.

Several previous studies have employed amplitude modulation spectrogram (AMS) features with linearly-scaled modulation filters (Kollmeier and Koch, 1994, Tchorz and Kollmeier, 2003, Kim *et al.*, 2009, Han and Wang, 2012, Wang *et al.*, 2013, Healy *et al.*, 2013, May and Dau, 2013, 2014), which is not consistent with psychoacoustic data on modulation detection and masking in humans (Bacon and Grantham, 1989, Houtgast, 1989, Dau *et al.*, 1997a,b, Ewert and Dau, 2000). As demonstrated by Ewert and Dau (2000), the processing of envelope fluctuations can be described effectively by a second-order band-pass filterbank with logarithmically-spaced center frequencies. In the present study, the ideal segregation of noisy speech, as represented by the IBM, was estimated by only exploiting

AMS features. In contrast to previous studies that employed linearly-scaled modulation filters, an auditory-inspired modulation filterbank with logarithmically-scaled modulation filters was used here, and its influence on speech segregation performance was investigated. Moreover, the influence of spectro-temporal integration on speech segregation was analyzed by combining information present in neighboring T-F units. Specifically, the size and the shape of the spectro-temporal integration window in the classification stage were varied and analyzed in terms of their impact on speech segregation. The speech segregation system was trained with AMS features that were extracted for a limited set of low SNRs, but evaluated over a wide range of SNRs to analyze the ability of the system to generalize to unseen SNRs.

The speech segregation system

The estimation of the IBM was accomplished in two stages: First, the amplitude modulation spectrogram features were used to train a two-class Bayesian classifier, which estimated the *a posteriori* probability of speech and noise presence in individual T-F units. Second, these probabilities were considered across a spectro-temporal window of adjacent time and frequency units and the final mask estimation was obtained by comparing the probability of speech with the probability of noise presence. Both stages are described in more detail below.

Amplitude modulation spectrogram features

The noisy speech signal was sampled at a rate of 16 kHz and normalized according to its long-term root mean square (RMS) value. Two different representations of the AMS features were compared: a "linear" representation based on linearly-scaled modulation filters and a "logarithmic" representation where the center frequencies of the modulation filters were spaced logarithmically, inspired by findings from auditory modeling.

Linearly-scaled AMS features

The linear AMS feature representation was similar to that described in earlier studies (Tchorz and Kollmeier, 2003, Kim *et al.*, 2009, Han and Wang, 2012, Wang *et al.*, 2013, Healy *et al.*, 2013, May and Dau, 2013, May and Gerkmann, 2014, May and Dau, 2014). The noisy input was segmented into overlapping frames of 4 ms duration with a shift of 0.25 ms. Each frame was Hamming windowed and zero-padded to a length of 128 samples and a 128-point fast Fourier transform (FFT) was computed. The FFT magnitudes were multiplied by 25 bandpass filters, in the following referred to as "frequency channels", with center frequencies equally spaced on the mel-frequency scale between 80 Hz and 8000 Hz.

The envelope in each frequency channel was then extracted by full-wave rectification, resulting in an auditory spectrogram-like representation.

Each frequency channel of the auditory spectrogram was further divided into overlapping segments of 32 ms duration with a shift of 16 ms. Each segment was Hamming windowed and zero-padded to a length of 256 samples and a 256-point FFT was applied to compute a modulation spectrogram for each frequency channel. Finally, the modulation spectrogram magnitudes were multiplied with 15 triangular-shaped modulation filters that were linearly-spaced between 15.6 Hz and 400 Hz. Because the modulation spectrogram had a frequency resolution of 15.6 Hz, each triangular filter contained modulation information derived from 3 adjacent FFT bins.

Logarithmically-scaled AMS features

Each frequency channel of the auditory spectrogram was processed by a second-order low-pass filter with a cutoff frequency of 4 Hz and 8 second-order band-pass filters with center frequencies spaced logarithmically between 8 and 1024 Hz, altogether representing a modulation filterbank. The band-pass filters were assumed to have a constant-Q factor of 1 inspired by auditory modeling and speech intelligibility prediction studies (Ewert and Dau, 2000, Jørgensen and Dau, 2011, Jørgensen *et al.*, 2013). The output of each modulation filter was integrated within segments of 32 ms duration with a shift of 16 ms to produce the final set of 9 logarithmically-scaled AMS features for each frequency channel.

Normalization

Machine-learning based segregation systems are typically trained with features that are extracted for a specific acoustic scenario, e.g. for a particular set of SNRs that were included in the training stage. The problem with these systems is that performance rapidly deteriorates as soon as a mismatch occurs between the acoustic conditions used for training and those used for testing. To alleviate the influence of the overall signal level on the AMS feature distribution, a normalization strategy was employed in the present study with the aim of improving the robustness of the system to mismatches between the SNRs in the training and the testing conditions. More specifically, the temporal envelope of the output of each frequency channel was normalized by its median prior to extracting the AMS features. The subband envelope signal is distributed between zero and an upper limit, leading to an asymmetric and skewed distribution. Therefore, a median-based normalization was chosen here.

Segregation stage

The segregation stage consisted of a Gaussian mixture model (GMM) classifier that is trained for each individual frequency channels, representing the AMS feature distributions of speech- and noise-dominant T-F units (Kim *et al.*, 2009, May and Dau, 2013). Given the trained GMM models for speech and noise, denoted by λ_1 and λ_0, as well as the AMS feature vector $\mathbf{X}(t,f)$ for a given time frame t and frequency channel f, the *a posteriori* probabilities of speech and noise presence were given by:

$$P(\lambda_{1,f}|\mathbf{X}(t,f)) = \frac{P(\lambda_{1,f})\,P(\mathbf{X}(t,f)\,|\lambda_{1,f})}{P(\mathbf{X}(t,f))} \tag{4.1}$$

$$P(\lambda_{0,f}|\mathbf{X}(t,f)) = \frac{P(\lambda_{0,f})\,P(\mathbf{X}(t,f)\,|\lambda_{0,f})}{P(\mathbf{X}(t,f))}, \tag{4.2}$$

where the two *a priori* probabilities $P(\lambda_{0,f})$ and $P(\lambda_{1,f})$ were determined by counting the number of feature vectors during training. Subsequently, the IBM was estimated by comparing the *a posteriori* probabilities of speech and noise presence for each individual T-F unit:

$$\mathcal{M}(t,f) = \begin{cases} 1 & \text{if } P(\lambda_{1,f}|\mathbf{X}(t,f)) > P(\lambda_{0,f}|\mathbf{X}(t,f)) \\ 0 & \text{otherwise.} \end{cases} \tag{4.3}$$

Information integration across time and frequency

Instead of using the output of the Bayesian classifier directly to estimate the IBM according to Eq. 4.3, the *a posteriori* probability of speech presence $P(\lambda_{1,f}|\mathbf{X}(t,f))$ was considered as a new feature spanning across the spectro-temporal integration window $\mathcal{W}(t,f)$ and representing the centered T-F unit

$$\bar{\mathbf{X}}(t,f) := \{P(\lambda_{1,v}|\mathbf{X}(u,v)) : (u,v) \in \mathcal{W}(t,f)\}, \tag{4.4}$$

with the window function $\mathcal{W}(t,f)$ defining the amount of spectro-temporal integration with respect to adjacent time and frequency units. Similar to Healy *et al.* (2013), this new feature vector $\bar{\mathbf{X}}(t,f)$ was learned by a second classifier for speech-dominant and noise-dominant T-F units. Depending on the size of the integration window $\mathcal{W}(t,f)$, the dimensionality of this new feature vector could be quite large. Therefore, a support vector machine (SVM) classifier was employed here, capable of dealing with high-dimensional data and requiring only little amount of training data.

Model evaluation

The segregation system was evaluated with 60 randomly selected sentences from the Danish Hearing in Noise Test (D-HINT; Nielsen and Dau, 2011) that were different from those used during the training stage. Each sentence was mixed with the seven different background noises listed in Tab. 4.1 at $-5, 0, 5, 10, 15$ and 20 dB SNR.

In order to evaluate the speech segregation performance, the percentage of correctly identified T-F units was computed by comparing the estimated binary mask with the IBM. Specifically, HIT - FA was reported (HIT; percentage of correctly classified speech-dominant T-F units) minus the false alarm rate (FA; percentage of erroneously identified noise-dominant T-F units) because this metric has been shown to correlate with human speech intelligibility (Kim *et al.*, 2009).

Table 4.1: Types of background noises.

Noise type	Description	Duration
ICRA1	Stationary speech-shaped noise	120 s
ICRA7	Non-stationary six persons babble	1200 s
PSAM 8Hz	Sinusoidal amplitude-modulated pink noise	∞ s
Traffic	Cars, trams, trucks and trains passing by	360 s
Music	Classical music	570 s
Destroyer	Destroyer operations room noise	235 s
Factory	Factory floor noise inside a car factory	235 s

Effect of spectro-temporal integration

Figure 4.4 shows the speech segregation performance obtained with the linear ("lin AMS"; open symbols) and the logarithmically-scaled AMS features ("log AMS"; filled symbols), respectively, as a function of the SNR (panel a), and for the different types of background noises (panel b). The logarithmically-scaled AMS features produced a considerably higher classification accuracy than the linear AMS features, although only 9 rather than 15 modulation filters were exploited. This performance difference was consistent across a wide range of SNRs and was about 10 %. The spectro-temporal integration stage was based on a causal plus-shaped integration window that spans 3 adjacent time frames and 9 frequency channels. A performance increase compared to the results without integration for both the linear and logarithmic AMS feature representations was obtained. This improvement in terms of speech segregation performance was close to 13 % and most prominent at low SNRs. Moreover, the integration stage seems particularly beneficial for non-stationary background noises, e.g. for the ICRA7 and the factory noise, as shown in panel b of Fig. 4.4.

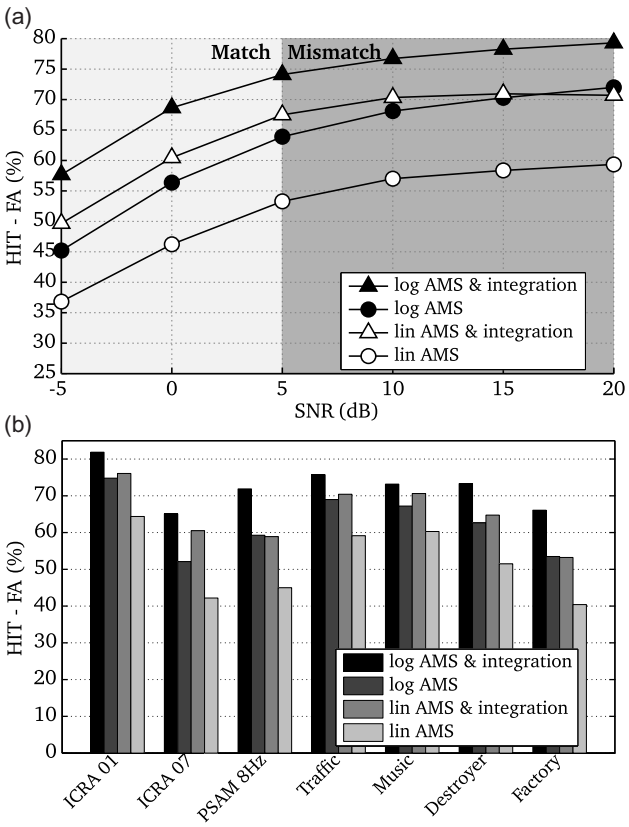


Figure 4.4: Classification results of individual T-F units for various segregation systems as a function of (a) the SNR averaged across all background noises and frequency channels, (b) the background noise averaged across all SNRs and frequency channels.

Visualization of modulation-based speech segregation

An illustration of the modulation-based speech segregation is shown in Fig. 4.5. The IBM is presented in panel (a) for speech mixed with factory noise at 0 dB SNR. It can be seen that the distribution of speech-dominant T-F units in the IBM is quite compact. Panels (b) and (c) present the estimated IBMs using the linear and the logarithmically-scaled AMS features, respectively, with the benefit of spectro-temporal integration represented in panels d and c, respectively. In addition to the estimated IBM patterns, the average HIT-FA rates for each frequency channel are provided in the right part of each panel. When the IBM was estimated on the basis of individual T-F units (panels b and c), following Eq. 4.3, some speech-dominated T-F units, particularly at higher-frequency channels, were erroneously classified as background noise. In addition, several noise-dominated T-F units were classified as being speech-dominated. In general, the logarithmic AMS features achieved a higher classification accuracy compared to the linear AMS features, especially

at higher frequencies. The spectro-temporal integration stage in panels (d) and (c) reduced theses outliers, and, more importantly, recovered many target-dominant T-F units at higher-frequency channels. Still, the linear AMS features missed many speech-dominant T-F units at higher frequency channels, which cannot be recovered by the spectro-temporal integration stage.

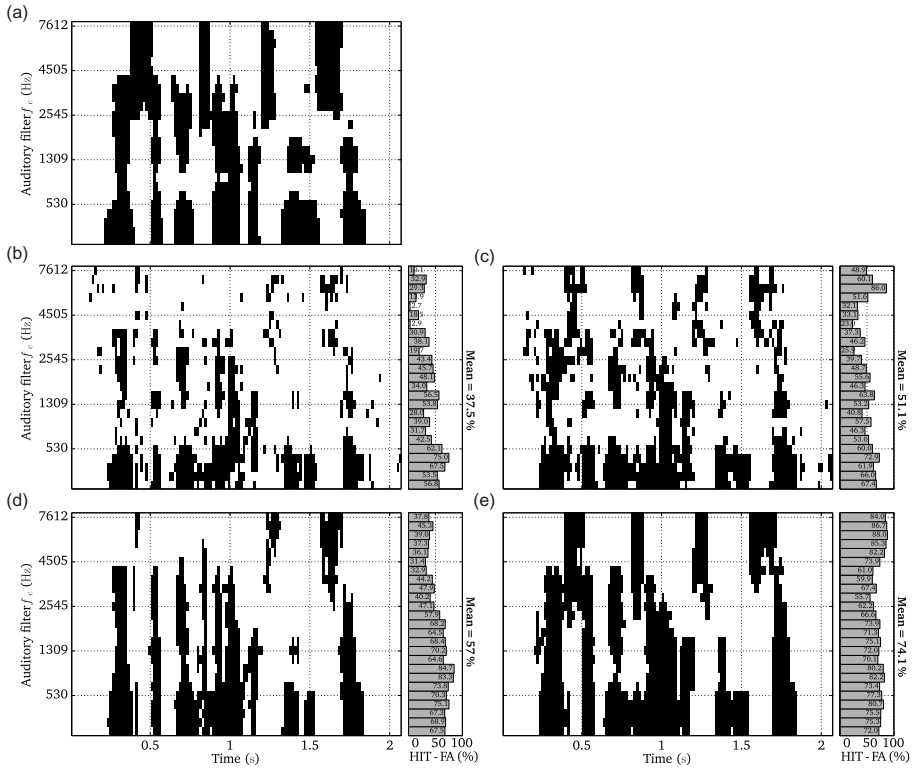


Figure 4.5: Ideal binary mask estimation and the frequency-dependent HIT-FA rates for a utterance mixed with factory noise at 0 dB SNR. (a) IBM, (b) Estimated IBM using linear AMS features, (c) Estimated IBM using logarithmic AMS features, (d) Estimated IBM using linear AMS features and spectro-temporal integration and (e) Estimated IBM using logarithmic AMS features and spectro-temporal integration.

4.1.3 Grouping based on onsets and offsets

According to Bregman (1990), common onsets and offsets across frequency are important grouping cues that are utilized by the human auditory system to organize and integrate sounds originating from the same source. Onsets can be detected by measuring the frame-based increase in energy of the ratemap representation. This detection is performed based on the logarithmically-scaled energy, as suggested by Klapuri (1999). Similarly to

onsets, the strength of offsets can be estimated by measuring the frame-based decrease in logarithmically-scaled energy.

The information about sudden intensity changes, as represented by onsets and offsets, can be combined in order to organize and group the acoustic input according to individual auditory events (Hu and Wang, 2007). Based on the respective onset and offset strength, a binary decision about onset and offset activity is formed, where only the most salient information is retained. In order to achieve this, temporal and across-frequency constraints are imposed for both the onset and offset information. Motivated by the observation that two sounds are perceived as separate auditory events when the difference in terms of their onset time is in the range of $20 - 40$ ms (Turgeon *et al.*, 2002), onsets or offsets are fused if they appear within a pre-defined *time context*. If two onsets or offsets appear within this time context, only the stronger one will be considered. This time context can be adjusted separately for onsets and offsets, respectively. Moreover, the minimum across-frequency context can be controlled. To allow for this selection, individual onsets and offsets which are connected across multiple time-frequency units are extracted using MATLAB's image labeling tool `bwlabel`. The binary onsets and offset map will only retain those onsets and offsets which consists of at least a pre-defined minimum of connected time-frequency units.

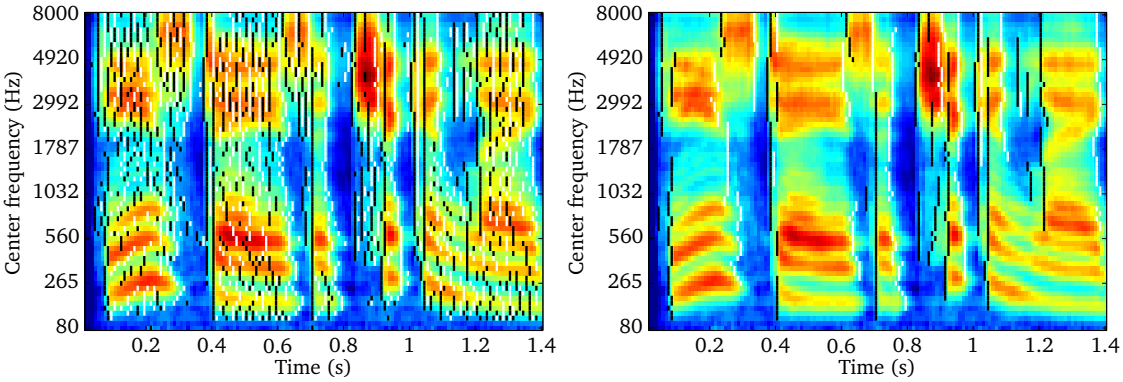


Figure 4.6: Detected onsets and offsets indicated by the black and white vertical bars. The left panels shows all onset and offset events, whereas the right panel applies temporal and across-frequency constraints in order to retain the most salient onset and offset events.

To illustrate the benefit of selecting onset and offset information, a ratemap representation is shown in Fig. 4.6, where all detected onsets and offsets (without applying any temporal or across-frequency constraints) are indicated by black and white vertical bars (left panel). It can be seen that the onset and offset information is quite noisy. When only retaining the most salient onsets and offsets by applying temporal and across-frequency constraints (right panel), the remaining onsets and offsets can be used as temporal markers, which clearly mark the beginning and the end of individual auditory events.

As suggested by Hu and Wang (2007), a multi-scale analysis using multiple time constant should be considered in the future to further improve the robustness of onset and offset detection.

4.1.4 Progress on Bioinspired Methods: Spike Coding

We investigated the possible application of spike coding methods for the pre-segmentation task. Using spike coding methods, a sound event is decomposed into discrete acoustic elements using time-shiftable kernel functions (Smith and Lewicki, 2005). Due to the discrete nature of the decomposition, the acoustic elements are called spikes, and methods for extracting spikes are known as spike coding methods. Each spike is a three-dimensional vector consisting of time, frequency and amplitude.

Spikes lie on a measurable feature space, and we can indeed define discrete measures for them. For example, one of the common assumptions is that the time, frequency and amplitude lie in the interval between 0 and 1. Let $\boldsymbol{x} = (x_a, x_t, x_f)^\top$ and $\boldsymbol{y} = (y_a, y_t, y_f)^\top$ denote two spikes, where x_a, x_t, and x_f are the amplitude, time and frequency features of the spike $\boldsymbol{x}$. Then we can define the distance between these two spikes as

$$d(\boldsymbol{x}, \boldsymbol{y}) = w_a(x_a - y_a)^2 + w_t(x_t - y_t)^2 + w_f(x_f - y_f)^2,$$

where $0 \leq w_a, w_t, w_f \leq 1$ and $w_a + w_t + w_f = 1$. With this definition, the distance between two sound events $\boldsymbol{s}_1 = \{s_{1n} \mid 1 \leq n \leq N\}$ and $\boldsymbol{s}_2 = \{s_{2n} \mid 1 \leq n \leq N\}$ is given by the normalized sum of the distances between the corresponding spikes, s_{1n} and s_{2n}, as:

$$\mathcal{D}(\boldsymbol{s}_1, \boldsymbol{s}_2) = \frac{1}{N} \sum_{n=1}^{N} d(\boldsymbol{s}_1, \mu(\boldsymbol{s}_1)),$$

where μ is a mapping function between the spikes of $\boldsymbol{s}_1$ and $\boldsymbol{s}_2 = \mu(\boldsymbol{s}_1)$. The mapping function can be determined by the Hungarian algorithm (Munkres, 1957), which assigns spikes from one sound to spikes from the other sound such that the total distance between both sounds is minimized.

Spike coding methods have been applied to the identification of sound events as described in Adiloglu *et al.* (2012), and a similar approach will be employed in future work. However, it is not clear that such methods can be used for pre-segmentation, and therefore their application is not considered further here.

4.1.5 Nonlinear tracking

The sound source localization framework introduced in Deliverable 3.1 is currently limited to static scenarios. Localization in dynamic environments, including moving sound sources and the movement of the robot itself, requires extensions of the current framework by nonlinear tracking methods. These methods rely on a state-space representation of the underlying problem, specified as a dynamical system

$$\boldsymbol{x}_k = f(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}) + \boldsymbol{v}_k, \tag{4.5}$$

$$\boldsymbol{y}_k = h(\boldsymbol{x}_k) + \boldsymbol{w}_k, \tag{4.6}$$

where $\boldsymbol{x}_k$ and $\boldsymbol{y}_k$ are the hidden state and measurement vectors, $\boldsymbol{u}_k$ is the control input to the robots actuators, $\boldsymbol{v}_k$ and $\boldsymbol{w}_k$ are vectors describing the process and measurement noise, $f(\,\cdot\,)$ and $h(\,\cdot\,)$ are linear or nonlinear mapping functions and k is the discrete time index. Equations (4.5) and (4.6) can be represented as a graphical model, shown in Fig. 4.7, allowing an efficient integration into the probabilistic framework of the blackboard system. The graphical model representation of the dynamical system reveals that it fulfills the Markov property. Thus, the current state only depends on its predecessor and the previous input. An estimation of the state trajectory can be achieved by performing inference on the graphical model. The current state can be inferred by marginalizing over all previous observations and inputs:

$$p(\boldsymbol{x}_k|\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k, \boldsymbol{u}_0, \ldots, \boldsymbol{u}_{k-1}) = \int p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{k-1}, \boldsymbol{u}_0, \ldots, \boldsymbol{u}_{k-2}) p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) \, \mathrm{d}\boldsymbol{x}_{k-1} \tag{4.7}$$

Popular techniques for solving the inference problem (4.7), like the well known Kalman Filter (KF) introduced in Kalman (1960), are usually restricted to linear dynamical systems. However, the task of tracking one or multiple sound sources implies a nonlinear relationship between the state and the measurement vector as will be derived later. Hence, a more advanced state estimation framework that handles nonlinearities is needed in this case. The current work in the field of nonlinear tracking within Two!Ears focuses on solving the state estimation task by integrating an Unscented Kalman Filter (UKF) approach, proposed by Julier and Uhlmann (2004), into the estimation framework. The UKF is an extension of the linear KF by introducing a statistical estimation technique into the filtering operation, thus realizing a computationally tractable inference procedure for nonlinear dynamical systems. This allows the approximation of a nonlinear function that is dependent on a random variable, by performing a linear regression between a specified number of samples drawn from the prior distribution of this random variable.

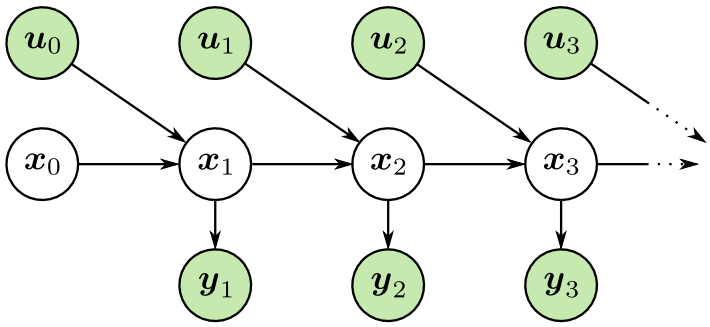


Figure 4.7: Graphical model representation of the general dynamical system described by Eq. (4.5) and Eq. (4.6). Shaded nodes denote observable variables and all other nodes represent hidden random variables. All random variables shown here are continous.

Process model describing source dynamics. By assuming a single, dynamic sound source in the horizontal plane, the hidden state vector can be represented in Cartesian coordinates

$$\boldsymbol{x}_k = \begin{bmatrix} x_k & \dot{x}_k & y_k & \dot{y}_k \end{bmatrix}^T,$$

where x and y represent the source position in the x-y-plane with respect to the head at time-instance k. $\dot{x}_k$ and $\dot{y}_k$ denote the first-derivatives of the corresponding state variables and can thus be characterized as direction-specific velocities.

A challenge in tracking the position of a sound source is, that the source motion dynamics, which are modeled via the function $f(\,\cdot\,)$, are generally unknown. In particular, possible source motion can be covered by a wide range of dynamics, ranging from completely static sources (e.g. music from a radio, fan noise) to highly dynamic motion (e.g. driving cars). To account for this, the underlying state-space model has to provide a flexible way to model a variety of source dynamics.

A model that has been proven to work well in the field of speaker tracking is the Langevin model introduced in Vermaak and Blake (2001):

$$\underbrace{\begin{bmatrix} x_k \\ \dot{x}_k \\ y_k \\ \dot{y}_k \end{bmatrix}}_{\boldsymbol{x}_k} = \underbrace{\begin{bmatrix} 1 & Te^{-\beta_x T} & 0 & 0 \\ 0 & e^{-\beta_x T} & 0 & 0 \\ 0 & 0 & 1 & Te^{-\beta_y T} \\ 0 & 0 & 0 & e^{-\beta_y T} \end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix} x_{k-1} \\ \dot{x}_{k-1} \\ y_{k-1} \\ \dot{y}_{k-1} \end{bmatrix}}_{\boldsymbol{x}_{k-1}} + \underbrace{\begin{bmatrix} T & 0 \\ 1 & 0 \\ 0 & T \\ 0 & 1 \end{bmatrix}}_{\boldsymbol{B}} \underbrace{\begin{bmatrix} v_{x_k}\sqrt{(1-e^{-2\beta_x T}} \\ v_{y_k}\sqrt{(1-e^{-2\beta_y T}} \end{bmatrix}}_{\boldsymbol{v}_k} \tag{4.8}$$

Here, T is the time interval between two consecutive measurements, β_x and β_y are friction coefficients and v_{x_k} and v_{y_k} are zero-mean, Gaussian distributed random variables with variances σ_x^2 and σ_y^2, respectively. The matrix $\boldsymbol{A}$ is denoted as the transition matrix and $\boldsymbol{B}$ is the noise input matrix. The model parameters that were suggested in Vermaak and Blake (2001) for the application of tracking human speakers are $\beta_x = \beta_y = 10\,\mathrm{s}^{-1}$ and $\sigma_x^2 = \sigma_y^2 = 5\,\mathrm{ms}^{-1}$. As the TWO!EARS framework is not restricted to localization and tracking of human speakers, the model parameters have to be adapted online, depending on the source type that is currently detected. This task can be considered as a bridge between source identification described later in section 5.2 and source localization, which will be a primary focus of further investigations within WP3.

Integration of head movements. A major advantage of using a Cartesian-coordinate based process model like (4.8) is, that the motion dynamics of a source can be modelled as a linear dynamical system with Gaussian process noise. However, a remaining aspect that still has to be considered is the dynamics of head movements. As the current look direction is best described as an angular value ψ_k, a head-centered coordinate transform of the state-space is necessary each time a head movement is performed.

By restricting head movements to the horizontal plane, the necessary coordinate transform reduces to a rotation of the cartesian coordinate system

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}, \tag{4.9}$$

where x' and y' are the cartesian coordinates after a counter-clockwise rotation of the original coordinates x and y by the angle ψ. The transformation (4.9) can be directly applied to the previously introduced state-space model (4.8), by projecting the source coordinates x_k and y_k onto a new head-centered coordinate system. However, it is also necessary to perform the coordinate transform for the source velocities. By taking into account the velocity of the source with respect to the x-axis, Eq. (4.8) yields the relations

$$\dot{x}_{k-1} = \frac{1}{e^{-\beta_x T}} \dot{x}_k \tag{4.10}$$

and

$$\dot{x}_{k-1} = \frac{1}{e^{-\beta_x T}} (x_k - x_{k-1}). \tag{4.11}$$

Using the coordinate transform (4.9) and substituting (4.10) into (4.11) yields the fol-

lowing relationship between the original and projected source velocity along the x-axis:

$$\dot{x}_k = Te^{-\beta_x T}\cos(\psi_{k-1})\dot{x}_{k-1} - Te^{-\beta_x T}\sin(\psi_{k-1})\dot{y}_{k-1}$$

The relationship describing the projected velocity along the y-axis can be derived in a similar way. The resulting projections can be described by the transformation matrix

$$\boldsymbol{T}(\psi_{k-1}) = \begin{bmatrix} \cos(\psi_{k-1}) & 0 & -\sin(\psi_{k-1}) & 0 \\ 0 & \cos(\psi_{k-1}) & 0 & -\sin(\psi_{k-1}) \\ \sin(\psi_{k-1}) & 0 & \cos(\psi_{k-1}) & 0 \\ 0 & \sin(\psi_{k-1}) & 0 & \cos(\psi_{k-1}) \end{bmatrix}. \tag{4.12}$$

When defining the look direction ψ_{k-1} as the control input u_{k-1} to the dynamical system, the head rotation can be included into the state-space representation of the source dynamics via the transformation matrix (4.12). Figure 4.8 shows an example of how this coordinate transform works in a dynamic scenario. Assuming a matrix notation as introduced in (4.8), this yields the modified process model

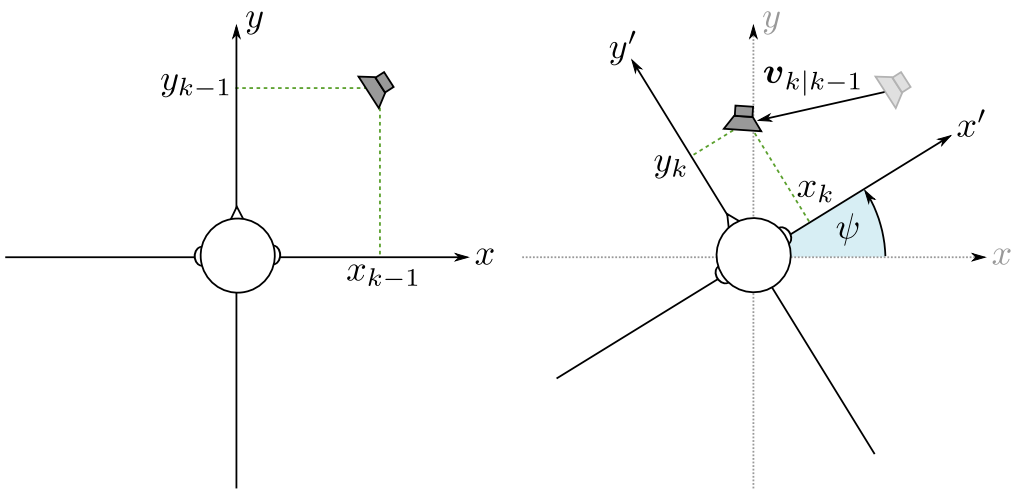
$$\boldsymbol{x}_k = \boldsymbol{T}(u_{k-1})\boldsymbol{A}\boldsymbol{x}_{k-1} + \boldsymbol{B}\boldsymbol{v}_k. \tag{4.13}$$

Measurement models. As can be seen so far, a linear process model can be derived for describing the motion of a source with respect to the head. However, the derivation of a proper measurement model (4.6) requires a mapping of the state vector $\boldsymbol{x}_k$ onto **ITDs!** (**ITDs!**), **ILDs!** (**ILDs!**) and probably additional features. This relationship is nonlinear in nature, as **ITDs!** and **ILDs!** generally correspond to azimuth angles, which implies a nonlinear mapping from cartesian to polar coordinates

$$\begin{aligned} d_k &= \sqrt{x_k^2 + y_k^2} \\ \varphi_k &= \operatorname{atan2}(y_k, x_k), \end{aligned}$$

where d_k is the distance between the source and the center of the head, φ_k is the azimuth angle and $\operatorname{atan2}(y_k, x_k)$ is the arctangent function with two arguments.

The derivation of a measurement model involves a selection of auditory cues, that are



(a) Previous timestep (b) Current timestep after head rotation

Figure 4.8: Illustration of the coordinate transform, performed between two consecutive timesteps if a head rotation by an angle of ψ degrees is conducted and the source is moving. The vector $\boldsymbol{v}_{k|k-1}$ corresponds to the movement trajectory of the source between time instants $k-1$ and k.

related to the distance and the azimuth angle of the source. Whereas **ITDs!** and **ILDs!** imply a natural correspondence to the latter, they have been shown to be independent of the source distance, at least if far-field conditions are assumed as described in Shinn-Cunningham (2000). Further investigations within the upcoming reporting period will focus on deriving appropriate measurement models, which link auditory cues to angular positions and distances. This includes the selection of relevant auditory cues besides **ITDs!** and **ILDs!**, especially for the estimation of source distances. Previous work in this direction has already been conducted by Shinn-Cunningham (2000), Lu and Cooke (2010, 2011) and Georganti *et al.* (2013). Additionally, the probabilistic localization framework introduced in the previous reporting period and its extensions described in section 5.1 will be integrated into the nonlinear tracking framework. This effort should lead to the derivation of an accurate measurement model, capable of modeling both spatial cues within one integrated framework.

4.2 Methods for visual processing

4.2.1 Simulation environment

As the elements of the Two!Ears system that are concerned with feedback will be developed over a time span of almost two years, many of these elements are not immediately available for constructing and testing of feedback procedures. This is particularly true for methods that deal with more abstract and/or more complex functions, like active exploration and multi-modal feedback methods. Both of these feedback paths will require the Two!Ears system to be endowed with sophisticated visual processing methods, e.g., for visual object detection/recognition, audio-visual speaker identification, or vision-based collision avoidance.

To be able to perform early feedback experiments, and test techniques for visual processing, we set up a virtual test environment (VTE) which mimics the feedback-relevant parts of the Two!Ears system and enables the visualization of simulated environments.

This allows project partners to test their own feedback-related ideas in the virtual environment, take advantage of the visual data provided by the VTE and set up cooperation with WP4's feedback routines early. By that strategy, potential issues might be detected and eliminated long before final system assembly. While experimenting with the feedback loops in the virtual environment, environmental variables and labels will be identified that are definitely needed for reliable feedback, thus allowing for algorithmic streamlining.

A first VTE realized in the Two!Ears context is the Bochum Experimental Feedback Testbed (BEFT) (**?**), on which we report in the following. For completeness, note that 4.2.2 and 4.2.3 are directly correlated to excerpts from Deliverable 4.1, part D.

4.2.2 BEFT

BEFT integrates a virtual 3D visualization environment based on the OGRE 3D rendering engine (OGRE, 2014), and hosts a mobile front end - currently a CAD model of the PR2 robot with a KEMAR dummy head mounted on a rotational axis. The testbed allows further scene components to be read in directly from XML files. This way, simulated entities such as people, walls, terrain, and so on, can easily be added to a scenario.

The entities convey physical parameters, such as distance and azimuth (w.r.t. the robot), or percentage of occlusion. Based on these parameters, BEFT simulates category labeling

for each entity: "hand-crafted" degradation functions (**?**) weaken a-priori knowledge of the entities' true categories in order to emulate uncertainty in category estimation caused by sensor noise or algorithmic issues.

According to the estimated parameters and the inferred category labels, the virtual mobile front end can be actuated (e.g., via active exploration mechanisms, see Deliverable 4.1, part D) in order to update and enhance the parameter/label estimates and refine the robot's internal world model.

Note that BEFT was intended to operate on the cognitive (rather than signal) level, allowing for very early feedback testing and multi-modal analysis, by skipping Two!Ears's signal processing and pre-segmentation stages that were "under construction" at that time. However, BEFT's 3D display capabilities and its abilities to handle robot control based on the Robot Operating System (ROS, 2014) middleware are clearly limited. The first issue might hamper simulation of visually challenging scenarios; the latter problem, however, would cause major re-work of algorithms tested in BEFT in order to port these methods to physical robot devices operating in real-world scenarios.

Figure 4.9: MORSE simulation of a KEMAR head and torso

As this is clearly unacceptable, the MORSE robotics simulator (MORSE, 2014) will inherit from BEFT and become the standard VTE for visual simulation in Two!Ears. Note that MORSE is based on the BLENDER 3D modeling/simulation software (Blender Foundation, 2014), using Bullet (2014) to enable physically plausible behavior of the simulated environment.

As MORSE has the ability to operate different robotic middlewares, e.g., ROS, porting issues can be minimized by enabling the Two!Ears framework to control/read out the virtual robotic front-end (motors, cameras, microphones, etc.) using exactly the same methods that will be employed to control/read out the physical device in later project stages. Further, MORSE comes with multiple pre-defined industrial components (sensors, actuators, controllers, and robotic platforms) which can be assembled into complete robots using straightforward Python™ scripting. For a more detailed overview of MORSE and its integration into the Two!Ears framework, Deliverable 4.1, part D, and Deliverable 5.1 are recommended.

4.2.3 Progress on visual processing

BEFT, combined with the MORSE simulator, will allow the study of feedback mechanisms in virtual scenarios. Several visual functions need be integrated into the Two!Ears architecture, as shown in Figure 3.1, in order to label objects detected in the physical world. The objective here is to integrate state-of-the-art methods that will provide visual knowledge to be fused with acoustic knowledge. Several steps are considered:

- For visual acquisition, the KEMAR head must be equipped with a visual sensor in order to acquire both appearance-based and 3D data.
- From acquired data, a pre-segmentation step allows the detection of visual targets, as regions of interest (ROI) in images. Such a ROI is characterised from some *a priori* knowledge provided to the system; an index k is assigned to every detected target X. Depending on the segmentation method, the X_k target could be characterised by histograms H_k, or could be associated with a semantic label S_k.
- A visual target is then tracked in the image sequence, potentially involving panning motions of the KEMAR head. As multiple targets could be detected and tracked in the same sequence, the tracking function must preserve the target indices by applying semantic, temporal and spatial consistency constraints.
- At every step, all visual knowledge available for targets is propagated in the global architecture towards the interpretation and fusion layer. The semantic interpretation could be made here, involving more complex methods than in the lower layer.
- Depending on the context, the fusion results, and on the audio-visual target characteristics, an active strategy could be decided and specific orders could be sent to the lower layer. For example, if the detection comes from audio data, the robot and/or the vision modules could be controlled to confirm the presence of the target in a predicted area.

To date, the visual modules have been analysed and the first specifications have been provided for some functions. Ultimately, the implementation will depend on the scenarios selected for the final demonstrations.

Visual acquisition

Several sensors could be selected for the visual acquisition with the following requirements:

- It must provide both appearance-based and 3D data. Appearance is required for the target detection and labelling process, while 3D geometrical data (typically position, velocity, and bounding box) will give observations required for the fusion process in order to update the target vector state.
- The accuracy of the 3D measurements will determine the accuracy of the target localisation. This must be consistent with the accuracy assumed for target localisation using the robotic platform on which the KEMAR head will be mounted, typically 10cm.
- For the 3D range, targets will have to be detected in a domestic environment, e.g. 5-7m maximum range.
- Finally, the sensor resolution and the sensor field of view depend on the demonstration scenarios, e.g. on the size of the objects to be detected, labelled and located.

The ideal sensor configuration would be bio-inspired, combining panoramic vision (from a fish-eye or an omnidirectional camera) and focalised vision (from a camera equipped with a long focal length or zoom). However, such a hybrid visual system is not available within the consortium, so the selected camera will be equipped with a fixed focal length lens, providing a horizontal field of view of approximately 60°.

Monocular vision, combined with structure-from-motion functions, could give 3D information assigned to every detected target, typically its position and velocity with respect to the KEMAR head. Previously, LAAS-CNRS have worked on visual SLAM (simultaneous localization and mapping) and MOT (mobile object tracking), using *a contrario* reasoning in order to detect a mobile target with respect to a camera embedded on a potentially mobile robot. But using only one camera is not sufficient to completely determine the position of a mobile target (Sola, 2007).

Passive stereovision will allow estimation of the full 3D position of objects more robustly than with only one camera. In the Two!Ears project, we can assume that an object to be detected will always be in the stereo view field, so that a dense stereovision algorithm could be executed in order to first build a disparity map from rectified images, and then a

3D image. LAAS has used a standard correlation-based stereovision algorithm for robot navigation for more than 20 years; additionally, many dense stereovision methods are made available on the MiddleBury web site (Scharstein and Zeliski, 2012).

Regarding the stereo rig configuration, it is known that the depth resolution decreases with the baseline (the distance between the optical centres) and with the horizontal focal length. With a stereo rig integrated on the KEMAR head, the baseline could be around 9cm. In Figure 4.10 it is shown that using a large field of view (96°) and a standard resolution camera (1 mega pixel), a 3D point could be reconstructed under 4-5m (disparity lower than 5 pixels). Some tradeoffs must be done between the maximum depth range, the view field volume and the accuracy of the depth, knowing that depth measurements have an error proportional to the square depth, and also inversely proportional to the baseline and to the horizontal focal length.

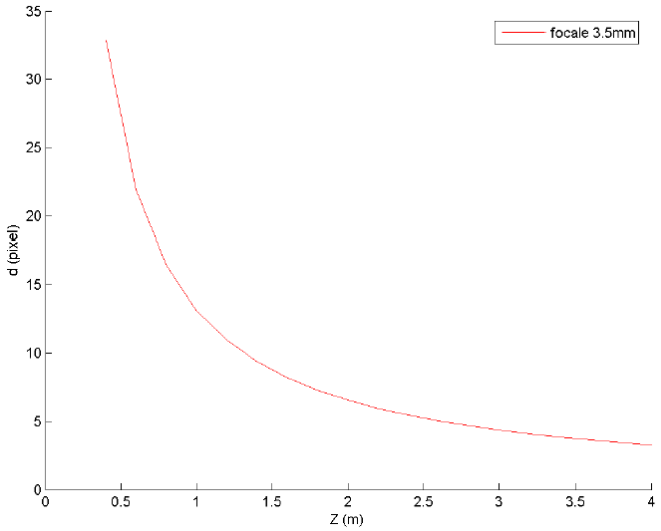


Figure 4.10: Disparity function of the depth, for a short baseline stereo rig.

A stereo rig with a shorter baseline will acquire very noisy measurements beyond 4-5m if it has a large field of view. Moreover, because it is a passive method (no random light projected on the scene), the disparity could be estimated only on textured regions.

If it is required to provide 3D measurements even on untextured scenes, active vision technologies such as RGB-D sensors (e.g. Kinect1 or a PrimeSense sensor) or 3D time-of-flight (TOF) cameras (e.g. Kinect2, Swiss Ranger or PMDTec sensor) need to be used. Nevertheless, for practical reasons related to the integration of the 3D sensor on the KEMAR head, passive stereovision is preferred to RGB-D technology.

Pre-segmentation: target detection

This step must extract Regions of Interest from the current image(s), i.e. regions in which the probability to detect a target is high. The detection could be confirmed (true positive) or discarded (false positive) using subsequent images in order to update the detection probability. In TWO!EARS, sensors are embedded on a mobile robot, so methods based on static background modelling are not pertinent.

A target must be segmented, i.e. the corresponding region must be recognised as a part or a view point of an object. The detection could be based on:

- The clustering of local cues, according to some criteria, e.g. consistent optical flow or apparent motions, with respect to the background. Many visual percepts could be used, e.g. interest points SIFT or SURF as described in Appendix B.
- Deterministic or stochastic reasoning using *a priori* information such as the target colour (e.g. chromatic segmentation based on the learnt skin colour in order to detect the hands or the head of a person) or the target shape (e.g. elliptic).
- Region classification based on an *a priori* learning step in order to configure the classifier (neural net, SVN, AdaBoost, decision trees) using histograms (colour, HOG) or simpler features computed from the Haar basis functions. Histograms of oriented gradients (HOG) are described in more detail in Appendix B.

Target detection from the clustering of local cues has been studied at LAAS-CNRS (Almanza-Ojeda *et al.*, 2011), using *a contrario* reasoning (Veit *et al.*, 2007) to group interest points according to their apparent motions. Gamez and Devy (2012) made this method more robust with the integration of a labelling process of the bounding box defined from the points cluster, using the standard OpenCV classification method based on a cascade of classifiers (AdaBoost) and global descriptors (Haar wavelets).

These global region descriptors are more adapted to multi-class detection, while local cues fit better with the detection of a given object. Nevertheless, considering local cues (i.e. interest points), firstly points are matched from their descriptors, typically from a nearest neighbour search in the descriptor space. Then different approaches could be executed:

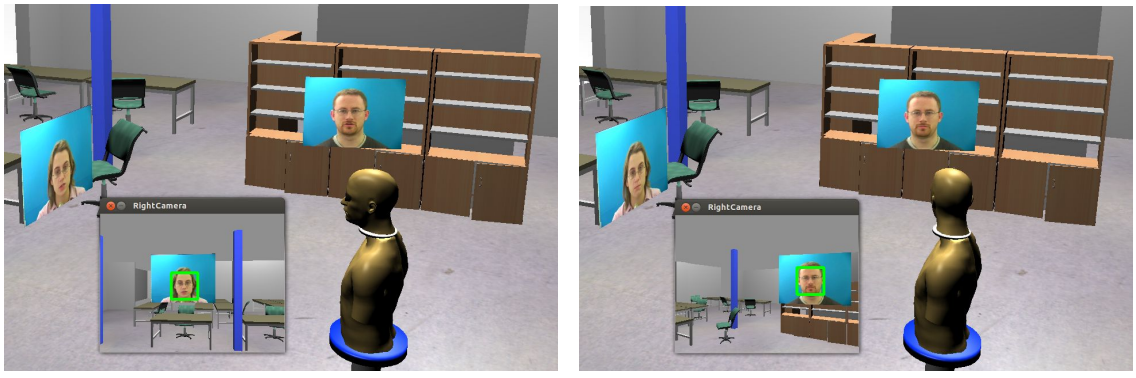
- Methods based on bags-of-words (BoW) allow detection of an object in an image region, using the histogram of the "codeword" occurrences in this region, without taking into account the positions of the matched points in the region. These codewords are defined during a codebook generation step during learning.
- Spatial constraints between points detected on an object can be learnt, and then exploited during the interpretation step by several methods (maximal cliques, ge-

ometric hashing, generalised Hough Transform) in order to find sets of consistent matches.

- Generative models allow a trade-off between these two approaches, i.e. spatial relationships between codewords are added in the object model and taken into account in the classifier.

The target localisation (position and attitude) can only be computed from matching between local features detected from images and learnt on models. So detection methods based on global region descriptors (i.e. histograms: HOG, colour, BoW) give only the position and the size of an image region that corresponds to an object (e.g., the face of a person, a human body, or a rigid object). Using stereo, a rough object position could be computed.

We set up a first proof-of-concept application to demonstrate capabilities on visual detection in the MORSE environment. An emulated KEMAR-like head/torso combination is embedded in a standard MORSE scenario (Figure 4.11). The artificial head is allowed to rotate freely and has two virtual cameras attached. Videos of human speakers are projected into the virtual environment using basic video texturing methods. Note that the videos are currently chosen from the GRID audio-visual corpus (Cooke *et al.*, 2006). One of the robot's cameras is connected to an external ROS node that was created using *GenoM3* (Mallet and Herrb, 2011). This external node is enabled to perform fast image processing based on the OpenCV library (WillowGarage, 2014): images from the virtual camera are streamed into the node and are then analysed with OpenCV's face detection engine. Figure 4.11 shows the results of face detection in MORSE: found faces are marked by green rectangles and the corresponding face regions could be propagated to higher system layers, e.g., to perform audio-visual speaker identification as described below.



(a) (b)

Figure 4.11: Face detection in the MORSE simulator

Note that there are still some issues to be solved with respect to synchronisation between the audio and the video data stream, see D4.1 for a more detailed discussion. Also, the above toy application is not yet fully integrated into Two!Ears; to eventually become consistent with the framework's architecture, the face detection mechanism will have to be encapsulated in a knowledge source according to ideas found in D4.1.

Target tracking

This function updates the target state from successive observations in an image sequence. It requires two main steps:

- The tracking itself identifies the target in image t given its state or where it was observed at time $t-1$.
- After obtaining this new measurement, the target state can be updated using an estimation framework or an optimisation one.

The two steps are interleaved, typically through the use of a joint probabilistic approach, both to predict the target observation at time t from its state at time $t-1$ and to update the target state from this observation. In the classical approach, the prediction step looks for the target only in an image region determined from the known target state in the operational space, or from the previous target observation in the image space. More efficient methods called "tracking-by-detection" exploit the detection function and a matching operation in order to associate detected targets at time t with tracks or tracklets built from the previous images.

These tracking and update steps become more complex when the camera itself is mobile, where it is required to estimate the sensor motion, using the same probabilistic approach. When the environment is not known, this problematic is designated by the SLAMMOT acronym, from SLAM (Simultaneous Localization and Mapping) and MOT (Mobile Object Tracking). LAAS-CNRS has studied several methods for SLAMMOT based on extended Kalman filtering (Sola, 2007, Gamez and Devy, 2012) or particle filtering (Zuriarrain *et al.*, 2013) which could be applied within Two!Ears.

Interpretation: target labelling

As shown in the previous sections, many detection methods in vision are model-based, i.e. based on classification in order to label an image region as an object from a given class. In such a case, target labelling is solved at the detection level, assigning a class label to the detected object. Hence, the detection function also performs generic object recognition.

A unified probabilistic framework was proposed by Gate *et al.* (2009) to cope with SLAM, detection, tracking and labelling, managing uncertainties both on the target states (variances on the estimated positions) and on the target labels (confidences). Approaches based on part-based models, first exploit matching between local features extracted from the current image with the ones memorised for a given object model, and then, analysis of the spatial relationships between the matched elements, using an exhaustive graph-based or relaxation method. In this case, the interpretation function must be done in parallel with the detection and tracking functions.

The more popular recognition methods are Robotearth (Di Marco *et al.*, 2012) and LINEMOD (Hinterstoisser *et al.*, 2010). Several methods (LINEMOD, TOD) have already been integrated in the ROS framework, by the Object Recognition Kitchen project initiated by Willow Garage (ORK, 2013). The LINEMOD package is currently being studied at LAAS-CNRS in order to be integrated later in the Two!Ears framework.

Considering these visual knowledge sources, an adaptive approach for the fusion of audio and visual modalities could be a loose coupling strategy, i.e. confidence levels about the target labels could provide confirmation of the interpretation result.

4.2.4 Outlook: work on audio-visual speaker identification

In the following reporting period, work on audiovisual integration will play an important part. One exemplary application is audiovisual speaker identification, which should dynamically integrate both information sources, audio and video stream, according to their respective reliability and information content.

For this purpose, we plan to use state-of-the art acoustic speaker identification and visual speaker identification, together with mechanisms for dynamic stream weighting, e.g. as described in (Abdelaziz and Kolossa, 2014). This method, or other related approaches, dynamically estimates the reliability and information content of both the audio and the video stream at each frame, and from this reliability, determines the optimum weight that the streams should be given when making the decision about the speaker identity.

Audiovisual speaker identification can thus become robust to variations in the environment, focusing only on the audio stream in very poor lighting conditions with good audio quality, and vice versa.

5 Formation of auditory objects

This section describes progress on the second layer of the blackboard, which concerns source models and predictors for assigning attributes to sound events. We present a system for sound localisation that is robust to reverberation and uses human-like head movements to resolve front-back confusions. Machine learning techniques have been applied to identify different source types (e.g., 'fire', 'baby crying'), using both discriminative classifiers and probabilistic models. Finally, an i-vector based approach has been developed for determining the identify of different speakers within the binaural setting of the Two!Ears project.

5.1 Sound localisation

Human sound localisation performance is very robust, even in the presence of multiple competing sounds and room reverberation (Hawley *et al.*, 1999). The two main cues that are used by the auditory system to determine the azimuth of a sound source are interaural time differences (ITDs) and interaural level differences (ILDs) (Blauert, 1997). However, these binaural cues are not sufficient to uniquely determine the location of a sound (Wightman and Kistler, 1999). In particular, a given ITD value actually corresponds to a number of possible locations that lie on the so-called *cone of confusion*. Hence, if listeners were only to use these binaural cues, then *front-back confusions* would frequently occur in which a source located in the front hemifield was mistaken for one located in the rear hemifield (or vice versa). However, human listeners rarely make front-back confusions because they often use information gleaned from head movements to resolve ambiguities (Wallach, 1940, Wightman and Kistler, 1999, McAnally and Martin, 2014).

The long-term aim of this study is to incorporate human-like binaural sound localisation in a mobile robot with an anthropomorphic dummy head. In particular, the use of head movements is investigated in a machine hearing system, which allows the prospect of human-like sound localisation performance in challenging acoustic conditions.

5.1.1 System

Machine hearing systems typically localise sounds by estimating the ITD and ILD in a number of frequency bands, and then mapping these values to an azimuth estimate. In order to increase the robustness of computational approaches in adverse acoustic conditions, a multi-conditional training (MCT) of binaural cues can be performed, in which the uncertainty of ITDs and ILDs in response to multiple sound sources and reverberation is modelled (May *et al.*, 2011, 2013, Woodruff and Wang, 2012).

The binaural signals were sampled at a rate of 16 kHz and subsequently analysed by a bank of 32 Gammatone filters with centre frequencies equally spaced on the equivalent rectangular bandwidth (ERB) scale between 80 and 5000 Hz. The envelope in each frequency channel was extracted by half-wave rectification. Afterwards, ITDs (based on cross-correlation analysis) and ILDs were estimated independently for each frequency channel using overlapping frames of 20 ms duration with a shift of 10 ms.

Sound source localisation was performed by a Gaussian mixture model (GMM) classifier that was trained to capture the azimuth- and frequency-dependent distribution of the binaural feature space (May *et al.*, 2011, 2013). Given a set of K sound source directions $\{\varphi_1, \ldots, \varphi_K\}$, that are modelled by frequency-dependent GMMs $\{\lambda_{f,\varphi_1}, \ldots, \lambda_{f,\varphi_K}\}$, a 3D spatial likelihood map can be computed for the kth sound source direction being active at time frame t and frequency channel f

$$\mathcal{L}(t, f, k) = p\left(\vec{x}_{t,f} | \lambda_{f,\varphi_k}\right). \tag{5.1}$$

The normalised posterior for each frame t was computed by integrating the spatial likelihood map

$$\mathcal{P}(k) = \frac{1}{T} \sum_{t}^{t+T-1} \frac{\prod_f \mathcal{L}(t, f, k)}{\sum_{\mathsf{k}} \prod_f \mathcal{L}(t, f, \mathsf{k})} \tag{5.2}$$

where T is the number of frames in each signal chunk. The most prominent peaks in the posterior distribution $\mathcal{P}$ were assumed to correspond to active source positions.

5.1.2 Multi-conditional training

The purpose of MCT is to simulate the uncertainties of binaural cues in response to complex acoustic scenes. This can be achieved by either simulating reverberant binaural room impulse responses (BRIRs) or by combining head-related impulse responses (HRIRs) with diffuse noise. In this study, binaural mixtures were created for the training stage by mixing a target source at a specified azimuth with diffuse noise, which consisted of 72 uncorrelated, white Gaussian noise sources that were placed across the full azimuth range

(360 °) in steps of 5 °.

The localisation model was trained with a set of 20 binaural mixtures for each of the 72 azimuth directions. For a given mixture, the target source was corrupted with diffuse noise at three different signal-to-noise ratios (SNRs) (20, 10 and 0 dB SNR), and the corresponding binaural feature space consisting of ITDs and ILDs was extracted. An energy-based voice activity detector (VAD) was used to monitor the activity of the target source. A frame was considered to be silent and excluded from training if the energy level of the target source dropped by more than 40 dB below the global maximum. The resulting binaural feature space was modelled by a GMM classifier with 16 Gaussian components and diagonal covariance matrices for each azimuth and each subband.

5.1.3 Head movements

Previous computational approaches have typically been limited to locating sound sources in the frontal hemifield. Hence, although MCT has been shown to provide robust localisation performance, the learned distribution of binaural cues for sound sources positioned in the front and rear hemifields will be quite similar. In order to reduce the number of front-back confusions, the localisation model is equipped with a hypothesis-driven feedback stage which can trigger a head movement in cases where the azimuth cannot be unambiguously estimated. The first half of the signal chunk (i.e., frames in the range $t = [1, T/2]$) is used to derive an initial posterior distribution of the sound source azimuth. If the number of local peaks in the posterior distribution above a pre-defined threshold θ is larger than the number of required source positions, the azimuth information is assumed to be ambiguous which triggers a head movement. The second half of the signal chunk is re-computed with the new head orientation, and a second posterior distribution is obtained.

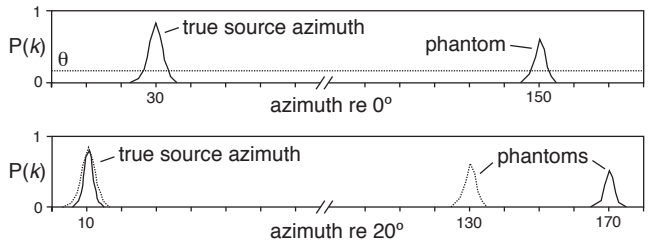


Figure 5.1: Head movement strategy. Top: Two candidate azimuths are identified above the threshold θ. Bottom: After head rotation by 20 °, only the azimuth candidate at 10 °agrees with the azimuth-shifted candidate from the first signal block (dotted line).

Assuming that sources are stationary over the duration of the signal chunk, the initial source azimuth distribution before the head movement can be used to predict the azimuth distribution after the head movement. This is done by circular shifting the azimuth indices

of the initial azimuth distribution by the amount of the rotation angle. If a peak in the initial posterior distribution corresponds to a true source position, then it should have moved towards the opposite direction of the head rotation and will appear in the second posterior distribution obtained for the second half of the signal chunk. On the other hand, if a peak is due to a *phantom source*, it will not occur at the expected position in the second posterior distribution. By exploiting this relationship, as illustrated in Figure 5.1, potential phantom source peaks are eliminated from both posterior distributions. Finally, the average of both posterior distributions is taken, producing a final posterior distribution for the signal chunk.

Head rotation strategies

Literature shows that there is a limit to how much listeners move their heads when localising sound sources and performance varies depending on head movement strategies (Perrett and Noble, 1997, Kim *et al.*, 2013). We investigated two strategies: (1) rotate head towards the location of the most likely (ML) source with a fixed rotation angle; (2) random rotation within limits. The ML source location is decided based on the initial azimuth posterior distribution before head movement.

Multi-step head rotation

Head rotation can either be completed with one step, or with multiple small steps. If an N-step strategy is used, then the signal is divided into $N + 1$ blocks in time and the first block is used to choose the overall head rotation angle ϕ°. At each step, a head rotation of a $1/N^{th}$ of ϕ° is used. This is illustrated in Figure 5.2. Such a rotate-stop-listen strategy can be more practical for a robotic platform, as head rotation may produce self-noise which makes the audio collected duration head rotation unusable.

5.1.4 Evaluation

Binaural simulation

We evaluated sound localisation in virtual listening environments by spatialising monaural sounds with head related impulse responses (HRIRs) for anechoic conditions or binaural room impulse responses (BRIRs) for reverberant conditions. Two listening configurations were employed. In Figure 5.3a, head rotations were simulated by using corresponding HRIRs or BRIRs for source azimuths relative to the new head orientation. Two sets of BRIRs were used to investigate the influence of mismatched binaural recording conditions:

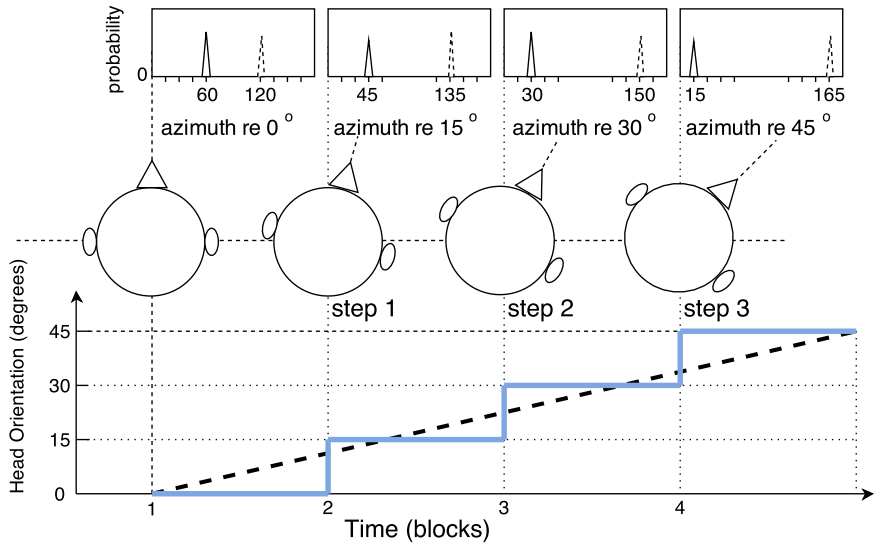
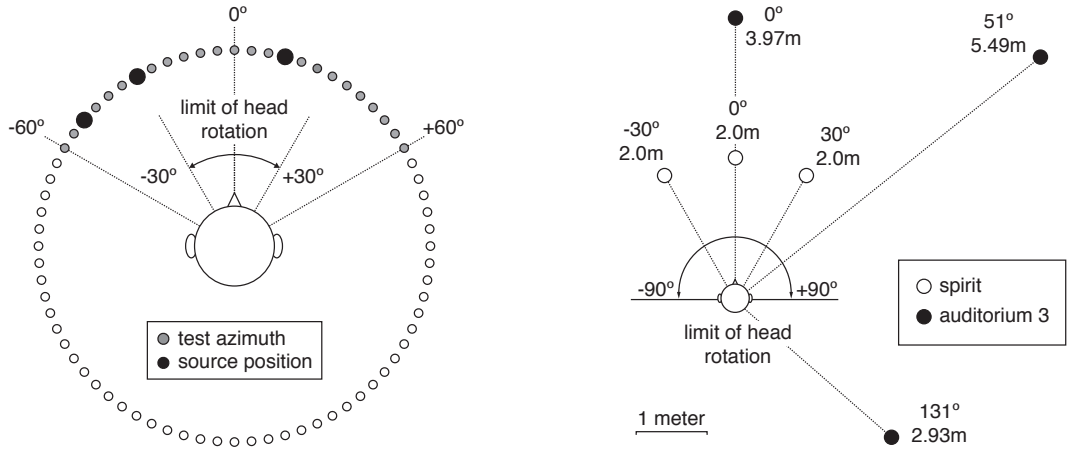


Figure 5.2: Illustration of head rotation with small steps. The overall rotation angle (45 °) is completed with 3 steps. The dotted thick line idealises human continues head rotation. Top: azimuth posterior distribution computed for each block regarding a different head orientation. The true source azimuth (solid peak) at 60 ° moves towards the opposite direction of head rotation while the phantom (dotted peak) moves with the head.

i) an anechoic HRIR catalog based on the Knowles Electronic Manikin for Acoustic Research (KEMAR) dummy head (Wierstorf *et al.*, 2011); ii) the Surrey database (Hummersone *et al.*, 2010). The anechoic KEMAR HRIRs were also used to train the localisation models. The Surrey database was captured using a head and torso simulator (HATS) from Cortex Instruments, and includes an anechoic condition as well as four room conditions with various amount of reverberation.

In Figure 5.3b, BRIRs measured for different head orientations ranging from -90 ° to 90 ° were used to simulate more realistic head rotations [1]. The BRIRs were measured for two rooms. The smaller *spirit* room has an estimated reverberation time $T_{60} \sim 0.5\,\mathrm{s}$ and the bigger *auditorium 3* has an estimated reverberation time of $T_{60} \sim 0.7\,\mathrm{s}$. Three source positions provided by each BRIR set were tested, as shown in Figure 5.3b.

1 The BRIRs are freely available at `https://gitlab.tubit.tu-berlin.de/twoears/data/tree/master/impulse_responses/qu_kemar_rooms`



(a) Virtual listener configuration 1

(b) Virtual listener configuration 2

Figure 5.3: Schematic diagrams of two virtual listener configurations. Left: BRIRs measured with a fixed head orientation. Filled circles indicate azimuths used for testing. Black circles indicate source azimuths in a typical three-talker mixture (in this example, at $-50\,^\circ$, $-30\,^\circ$ and $15\,^\circ$). All azimuths were used for training. During testing, head movements were limited to the range $[-30\,^\circ, 30\,^\circ]$. Right: BRIRs measured for different head orientations ranging from -90 ° to 90 °. Circles indicate azimuths and distances of sources used for testing.

Benefit of combining MCT and head movement

We used the virtual listener configuration 1 (Figure 5.3a) to investigate combined benefit of MCT and head movement. All the localisation models were tested using a set of 20 one-talker, two-talker, and three-talker acoustic mixtures from the TIMIT corpus (Garofolo *et al.*, 1993). The testing source azimuth was varied in $5\,^\circ$ steps within the range of $[-60\,^\circ, 60\,^\circ]$, as shown in Figure 5.3a. Source locations were limited to this range of azimuths because the Surrey BRIR database only includes azimuths in the frontal hemifield. However, the system was not provided with information that the azimuth of the source lay within this range, and was free to report the azimuth within the full range of $[-180\,^\circ, 180\,^\circ]$. Hence, front-back confusions could occur if the system incorrectly reported that a source originated from the rear hemifield.

Table 5.1 presents localisation accuracy of various systems. The MCT approach improves performance substantially over the clean training system in multi-talker scenarios and in the presence of room reverberation. Despite being trained with white Gaussian noise, the model generalised to reverberant conditions recorded. This confirms that MCT can account for the distortions of ITDs and ILDs caused by real reverberation.

The head movement strategy improved the performance for all localisation systems. This

Table 5.1: Gross accuracy in % for various sets of BRIRs when localising one, two and three competing speakers. HR – system with head rotation.

	KEMAR			Surrey Database								
	Anechoic			Anechoic			Room C			Room D		
	# competing speakers											
	1	2	3	1	2	3	1	2	3	1	2	3
Clean	91.3	52.2	28.4	65.9	33.9	19.4	26.7	13.0	8.0	13.0	7.6	5.7
Clean+HR	99.2	59.3	32.4	69.2	38.8	22.6	64.9	19.7	10.5	64.1	18.9	10.0
MCT	100	95.5	86.3	100	92.2	82.0	99.7	87.7	76.6	90.6	76.3	68.2
MCT+HR	100	96.4	87.7	100	94.8	84.9	99.8	92.5	82.1	97.5	86.3	74.3

benefit was particularly pronounced for the single-talker mixtures in the presence of strong reverberation (room C and D), where confusions are likely to occur due to the impact of reflections. Although the model based on clean ITDs and ILDs did not generalise well to the HATS artificial head, the head rotation strategy helped to improve performance in room C and D by more than 40 % for the single-talker scenario. Similarly, head movements were beneficial for the best MCT-based localisation model, for which performance increased from 90.6 % to 97.5 % for the most reverberant single-talker scenario.

Effect of head movement strategies

We used the virtual listener configuration 2 (Figure 5.3b) to investigate the effect of different head movement strategies. A set of 100 one-talker, two-talker, and three-talker mixtures was used. Each talker was simulated by randomly selecting a sentence from the TIMIT corpus, excluding the ones used for training. Binaural mixtures of multiple talkers were created by convolving each talker signal with the BRIRs separately before adding them together in each of the two binaural channels. Both rooms contain 3 source positions. For the one-talker and two-talker mixtures, the azimuth directions were randomly selected for a given mixture.

The localisation performance was evaluated by calculating the RMS errors in degrees for each sentence, averaged across all source positions for each room. The MCT-based localisation system described in Section 5.1.2 was selected as a baseline. The proposed localisation system employed the same statistical front-end but adopted various head rotation strategies as described in Section 5.1.3.

Figure 5.4 compares the results produced by the three systems. All systems exploiting head rotation improved the localisation accuracy over the 'No Rotation' baseline in both rooms. Rotating towards the ML source position produced better performance than random rotation. Localisation with access to longer signals (2-s over 0.5-s) did not have a strong effect for the baseline. However, both head rotation systems benefitted greatly from having

longer signals for localisation. This is consistent with findings in (Perrett and Noble, 1997) where longer signal duration has a large benefit on listener's localisation performance in head rotation conditions, but little effect in motionless conditions.

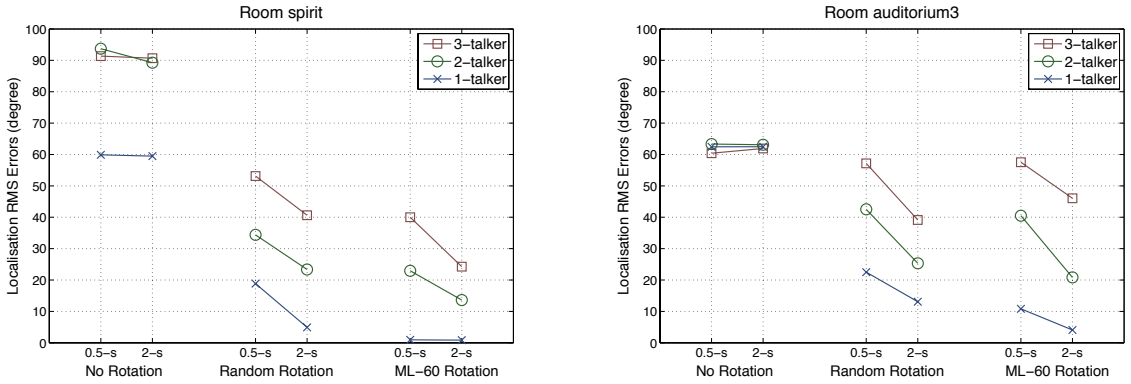


Figure 5.4: RMS errors in degrees of three localisation systems that exploit either no rotation, a random rotation or a rotation towards the ML source position by 60 °. Results are shown for two different signal durations (0.5-s and 2-s).

Table 5.2 shows the localisation performance of systems employing head rotation with multiple steps. This experiment investigated the trade-off between the number of steps of head rotation that is employed and the time that the system has to integrate binaural cues in between each head movement. The signal length was fixed at 2-s. In room *spirit*, only the 12-step head rotation system provides a benefit. This could be due to the fact that 'ML-60 °' was the best performing strategy for room *spirit* with a single head movement, and the localisation performance is already good. The 12-step head rotation system also produced the best performance in room *auditorium 3*.

Table 5.2: RMS errors (°) of systems that use a strategy in which the head is rotated towards the ML source position in multiple steps. The overall rotation angle for all conditions was fixed at 60 °. The length of all stimuli was 2-s.

Rotation strategy	Angle per step	Block (ms)	*spirit*			*auditorium 3*		
			1-spkr	2-spkr	3-spkr	1-spkr	2-spkr	3-spkr
No rotation	0 °	2000	59°	89°	91°	62°	63°	62°
1-step	60 °	1000	1°	14°	24°	4°	21°	46°
2-step	30 °	667	1°	25°	41°	4°	34°	39°
3-step	20 °	500	2°	21°	29°	4°	20°	36°
6-step	10 °	286	2°	17°	27°	4°	21°	31°
12-step	5 °	154	2°	6°	18°	4°	13°	31°

5.2 Source Type

Human listeners build auditory objects from perceived sound streams based on their internal models, incorporating gestalt principles like harmonicity, common onset, or common modulation (Bregman, 1990). In Section 5.1, one important attribute of auditory objects, source location, is discussed. Another important attribute of auditory objects is *source type*. Source types usually refer to the physical cause of sounds, such as 'baby crying' or 'woman speaking'. This section describes aspects and approaches of building models for determining the auditory object type in the TWO!EARS framework.

5.2.1 Source type classification

To allow source type classification in the TWO!EARS system, we train models for each source type. Training is currently carried out in an *offline* fashion (extension to online-trainable models is planned for the future). However, once trained, the models can be used in an *online* fashion in the `IdentityKS` within the blackboard system (see Section 3.3.5). Figure 5.5 shows a diagram of the source type classification system.

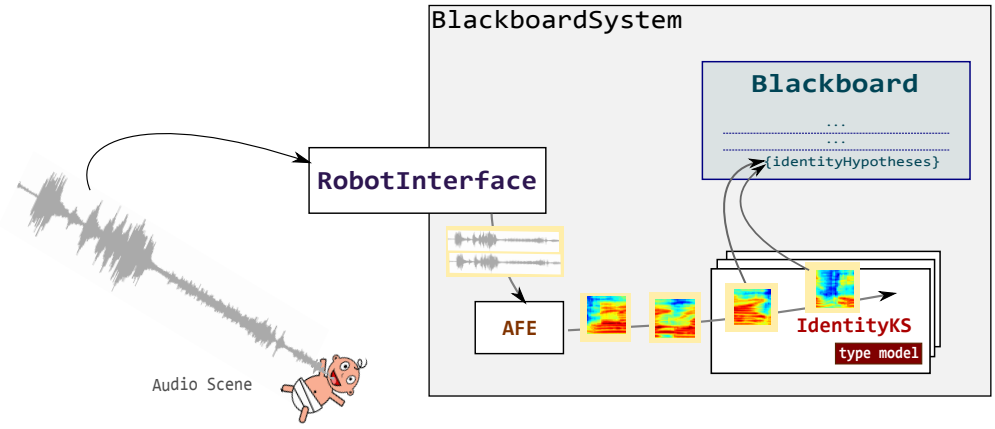


Figure 5.5: TWO!EARS source type classification system. This diagram shows that the continuous stream gets cut into blocks for type identification.

The online nature of the classification system allows for two different modes of classification:

Block-based classification In the block-based mode, audio data are packed in *blocks* of a certain length before being evaluated by the models, which should usually be trained using the same block size. This is the working mode for statistical models that do not explicitly model temporal characteristics of a sound type.

Continuous classification In this mode, continuous data are streamed to the model. However, because of the aforementioned need for information of a longer time span, these model need to then have some form of a *memory* to store context.

At the moment, we employ models of the first mode as they are easier to train. However, models including time and context through their architecture like hidden Markov models or LSTM (Hochreiter and Schmidhuber, 1997) may produce better results (Graves *et al.*, 2013) and are more plausible from a human perspective. Therefore we plan to investigate such models in the future.

5.2.2 Features

We select a set of features (extracted using the auditory front-end in the Two!Ears system, described in Deliverable 2.2) that may contain information about the auditory object type. The feature set is likely to be 'over-complete' and contain redundant information. However, it serves as the basis for later stages of feature selection:

- Frequency domain low-level statistics, such as spectral centroid, flatness
- Ratemap magnitudes (variable number of frequency channels, for instance 16 or 32). Ratemaps are biologically inspired spectrogram-like maps supposed to represent auditory nerve firing rate
- Amplitude modulation maps
- Onset strengths
- Gabor features

These features are calculated using frames with a length of 20 ms and a shift of 10 ms. We then apply the following operations to these features:

- Deltas of the features over time, corresponding to the first (discrete) derivative.
- For *blocks* of particular length (for instance, 500 ms), compute L-statistics[2] (L-mean, L-scale, L-skewness, L-kurtosis) over time.

Since Two!Ears is using binaural auditory processing, all these features are available for the left and right ear respectively. At the moment, we simply average both channels, since

2 L-statistics are given by L-moments, a sequence of statistics used to summarise the shape of a probability distribution (Hosking, 1990). L-statistics are shown to be more robust than conventional statistics, in particular with respect to the higher moments and when a small amount of data is available (David and Nagaraja, 2003, Ch. 9).

binaural information is not deemed important for the task of type classification. However, in the future we might change this in the process of integrating sound classification with localisation – it could, for example, be helpful to assign a higher weight to the ear closer to the source during classification, or simply use the 'better' ear.

In summary, a combination of low-level features, ratemap magnitudes, amplitude modulation maps, onset strengths and Gabor features, their deltas and block-statistics comprise our feature list.

Feature selection

We have defined a set of features that may contain redundant information. Thus, a subset of features is selected in order to improve the models (for example by reducing overfitting or increasing class separability), to shorten training time and to lower computational demands in the online usage of the models.

Feature selection methods are categorised into three main groups: wrapper methods, filter methods and embedded methods (Guyon and Elisseeff, 2003). Currently we have only considered embedded methods. Embedded methods perform feature selection as a part of the model learning. Specifically, we will work with the Lasso (least absolute shrinkage and selection operator) method, a regularised version of least squares with a L1-norm constraint. Further in Section 5.2.7, we study the application of mixture of factor analysers as an embedded approach towards feature selection.

5.2.3 Source type model training

To facilitate training of new models with different feature sets and different model techniques, we have implemented a highly flexible training pipeline tailored to the needs of offline training for source type classification. This pipeline among others has the following features:

Multi-conditional auditory scene simulation. Ear signals are produced using the WP1 auditory scene simulation tool (see Deliverable 6.1.1) from audio files. This can be done under various conditions (for instance with or without reverb, with or without interfering sources). An arbitrary number of such conditions can be specified and will be simulated to produce multi-conditional training data. The motivation to do so instead of training on 'clean' data is to include invariance to different conditions into the model by using data that enforces this while learning already, instead of artificially engineering the invariance after model training.

Auditory front-end (AFE). From the ear signals, the features are produced by the AFE in

exactly the same manner as in the blackboard system (refer Section 3.3.2). This is to ensure that training and application of the models is done under the same conditions.

Intermediate results. All products from intermediate stages, such as the ear signals or features produced by the AFE, are saved together with their configurations, since these stages can be very time-consuming. They are saved in such a way that they can be recombined whenever parts of a configuration have already been computed before.

Interface for feature creators. The whole pipeline is constructed in an object-oriented way, and with flexibility in mind. In particular, it is easily possible to create new feature sets that can be plugged into the pipeline by just implementing a class that inherits from a *feature creator* superclass.

Interface for model trainers. The same as with the feature creators, there are interfaces for *model trainers*. Any class inheriting from them can implement its own technique (like GMM or SVM), and can be plugged into the pipeline without modification. Wrapper trainer classes for conducting cross validation e.g. for hyper-parameter search are provided.

Plug and play. Models created by the training pipeline implement a model interface, and can be plugged directly into the `IdentityKS` to be used in the blackboard system (see Section 3.3.5).

5.2.4 Discriminative modelling using support vector machines

Support vector machines (SVM) are one of the most widely used supervised learning models for classification because of its solid performance and ability to efficiently model non-linear data through different kernels (Schölkopf and Smola, 2002). We use it as a well-proven baseline model to compare with more advanced techniques. We employ the excellent, very mature and broadly used toolbox, LIBSVM (Chang and Lin, 2011).

A support vector machine (SVM) is a binary classifier in nature, discriminating between two classes. Since we want to build models for each source type that decide whether or not this respective type is present in the current data block, we split the training data into two parts for each model: the pool of *positive* instances where the event is present, and the pool of *negative* instances where the respective event is not present. This would result in our pool of negative examples being much larger than the pool of positive examples. To be able to effectively train the model, it is necessary to set the importance of the positive examples to a higher value[3], which is possible through class-specific C-values with the

3 We set this to the ratio of the number of negative to positive examples.

SVM.

In the process of training an SVM model, hyper-parameter optimisation is usually done by a grid search over the parameter space, evaluated with cross validation. The choice of performance measure for this optimization is therefore of substantial importance, as it influences the model. We choose a variant of the *balanced accuracy* that prefers to have equally many false positives and negatives over having few mistakes of one category and many of the other:

$$BAC_2 = 1 - \sqrt{\left(\frac{(1 - sensitivity)^2 + (1 - specificity)^2}{2}\right)} \tag{5.3}$$

The trained model implements a decision boundary in the feature space: instances on either side then get labeled $+1$ or -1 by the model, corresponding to 'source type present in the block' or 'source type not present in the block'. However, for further processing and reasoning in higher stages of the Two!Ears system, it would be favourable to provide a score instead of a binary decision. Hence we train, on top of the native SVM model, a model that maps the distance of the instance to the decision boundary to a probability; this functionality is provided by LIBSVM. The final output of the model applied to a data instance is the probability that the source type is present in the respective data block.

Test on NIGENS[4] We trained SVM models using training data compiled from our general sound events database for the following source types: 'baby crying', 'female human speaking', 'fire burning', and 'piano playing'. Training and testing data were 'clean', without interfering sources or noise, in an anechoic environment. Apart from sound events of the four mentioned classes, examples from six other classes and a 'general' class were used. 631 respectively 205 files were used to create training and testing data (stratified, i.e. the same percentage from each class). Two different feature sets were used: *a)* 16-channel ratemaps plus deltas, mean and standard deviation over time for blocks of 500 ms, resulting in 64-dimensional feature vectors, and *b)* the first two L-moments applied to a combination of 16-channel ratemaps (without deltas), low-level spectral features, amplitude modulation maps and onset strength maps on blocks of 500 ms, resulting in 182-dimensional feature vectors. SVMs with a linear kernel were used. First a subset of 10,000 feature vectors was used for hyper-parameter optimisation, after which the full set of about 75,000 feature vectors was used to train the final model which was then applied to the test data to evaluate their performance using the above explained variant of balanced accuracy. The following table shows the results:

4 The NIGENS database includes general sound events recorded in isolation (see Deliverable 1.1).

model	feature set a	feature set b
'baby crying'	0.942	0.947
'female human speaking'	0.956	0.991
'fire burning'	0.824	0.877
'piano playing'	0.929	0.899

5.2.5 Probabilistic modeling using Gaussian mixture models

In this deliverable, we consider mixture of Gaussian distributions, commonly known as Gaussian mixture models (GMMs) (Bishop, 2006), for modelling underlying distribution of the training data.

Training models using GMM Let $\mathcal{X}$ denote our training data which is assumed to be a set of d-dimensional independent and identically distributed (i.i.d.) observations, that is $\mathcal{X} = \{\boldsymbol{x}_n \in \mathbb{R}^d,\ \forall\ 1 \leq n \leq N\}$. The underlying distribution of $\mathcal{X}$ is modelled using a mixture of multivariate Gaussian distributions

$$f(\mathcal{X}) = \prod_{n=1}^{N} \sum_{k=1}^{K} \tau_k\ \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_k, \Sigma_k),$$

where τ_k is the mixture weight, $\boldsymbol{\mu}_k$ is the mean vector, and Σ_k is the covariance matrix of the k-th mixture component. There are two main learning methods, namely maximum likelihood (ML) inference and Bayesian inference. In this deliverable, we focus on ML inference as described in (Bishop, 2006, Ch. 8).

The training is done using the binary 'one-vs.-all' strategy, as described in the section about SVM modelling, hence we train a 'positive' and a 'negative' model for each source type.

Classification of live streams using GMMs. From the stream of data, blocks with length of 500 ms and a shift of 10 ms are selected. The goal is to determine the identity of each block. For this purpose, we first extract the features for each block as described in Section 5.2.2. Next, this data instance is evaluated against all trained GMM models by computing and comparing the log-likelihood values.

Let $\mathcal{M} = \{\mathcal{M}_i \mid 1 \leq i \leq I\}$ denote the trained models for I sound event classes, where $\mathcal{M}_i$ is the trained model for the i-th class. Further let $\widehat{\mathcal{X}}$ denote the feature set observed at the current block of the live stream. We then evaluate log-likelihood values for $\widehat{\mathcal{X}}$ against all the trained models $\mathcal{M}$. The identity of the unknown feature set $\widehat{\mathcal{X}}$ is set to the identity of the model which produces the largest log-likelihood value among all the trained models.

Mathematically speaking,

$$i = \underset{i'}{\operatorname{argmax}} \ \log f(\widehat{\mathcal{X}} \mid \mathcal{M}_{i'}), \quad \forall\ 1 \leq i' \leq I.$$

5.2.6 Adapting models to improve discrimination

In this section we discuss how we can improve the discrimination performance of GMMs by increasing class separability and adapting the trained models to noise overlaying the auditory events. Specifically, we briefly study application of linear discriminant analysis and a recently proposed noise adaptive variant of it (Kolossa *et al.*, 2013).

GMM + LDA

A linear discriminant analysis (LDA) finds a linear projection of the feature-space which maximises the class separability. Let Σ^{b} denote the between-class scatter-matrix (covariance-matrix) and Σ^{w} denote the within-class scatter-matrix. Then $\boldsymbol{W}^{\mathrm{LDA}}$ is given by computing the p-largest generalised eigenvectors $\boldsymbol{W}^{\mathrm{LDA}} = \{\boldsymbol{w}_i^{\mathrm{LDA}} \mid 1 \leq i \leq p\}$ from solving

$$\Sigma^{\mathrm{b}} \boldsymbol{w}_i^{\mathrm{LDA}} = \lambda_i \Sigma^{\mathrm{w}} \boldsymbol{w}_i^{\mathrm{LDA}}, \tag{5.4}$$

where λ_i is the i-th eigenvalue associated with the i-th eigenvector $\boldsymbol{w}_i^{\mathrm{LDA}}$.

We use LDA in order to improve the separability between the positive and negative model for each class. Application of LDA is achieved by multiplying the feature vectors with the projection matrix $\boldsymbol{W}^{\mathrm{LDA}}$, as

$$\widehat{\boldsymbol{x}} = \boldsymbol{W}^{\mathrm{LDA}} \boldsymbol{x}, \tag{5.5}$$

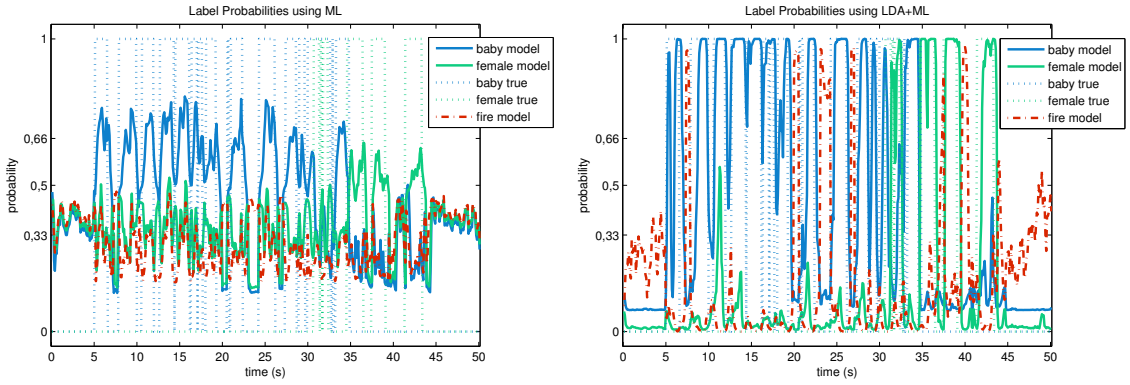
where $\boldsymbol{x}$ is the feature vector extracted from the data block, and $\widehat{\boldsymbol{x}}$ is the transformed feature vector. Subsequently, the already trained models have to also be modified to account for the transformation.

Experiment on NIGENS database. In the following, we show how LDA could serve as a post-processing block for improving the class-separability of an existing GMM classifier. We consider three different types of sound events, namely: 'BabyCrying', 'FemaleSpeech', and 'Fire'. We extract ratemap magnitudes with 32 frequency channels for each frame; the means along the time dimension result in the feature vector. We next train the classifiers for each type using only one component (a single Gaussian distribution) as described before. The resulting model parameters for each type are the covariance matrix and mean vector, i.e., Σ_c and μ_c where $c \in \{\mathrm{BabyCrying}, \mathrm{FemaleSpeech}, \mathrm{Fire}\}$.

In the testing phase, we process a scene 50 s in duration. The following methods are compared:

- ML, standard maximum likelihood decoder using the trained model parameters, Σ_c and $\boldsymbol{\mu}_c$, and full dimensional feature vectors $\boldsymbol{x}$.
- LDA+ML, computing the ML after mapping the feature vectors to the projection space using Eq. (5.5) and modifying the model parameters as $\widehat{\boldsymbol{\mu}}_c = \boldsymbol{W}^{\mathrm{LDA}}\boldsymbol{\mu}_c$, $\widehat{\Sigma}_c = \boldsymbol{W}^{\mathrm{LDA}}\Sigma_c\boldsymbol{W}^{\mathrm{LDA}}$.

Figure 5.6-(a) and Figure 5.6-(b) show the result of the experiment using ML and LDA+ML, respectively. It can be seen that there are considerably less class overlaps when using LDA as a post-processing step. Although the overall accuracy seems to be similar, the predictions are more confident with LDA. This shows that LDA can be used to improve the source type classification with GMMs.



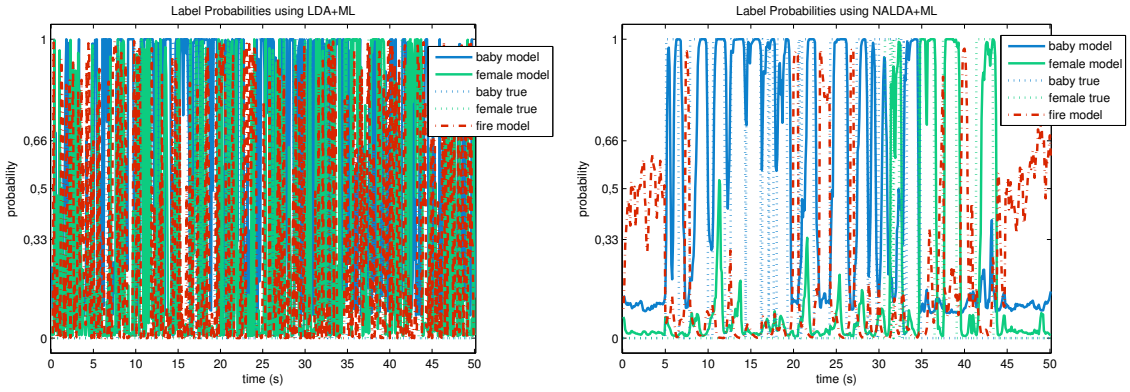
(a) ML (b) LDA+ML

Figure 5.6: Evaluation of the scene. The models are standard GMM with one component. Dotted lines represent ground truth and solid lines are predictions. Notably, the predictions with LDA+ML are more confident compared to the ones not using LDA.

GMM + NALDA

While LDA proves useful as a post-processing block, it does not help with the models' sensitivity to noise. Thus, we now discuss *noise adaptive LDA* (NALDA) (Kolossa *et al.*, 2013), that can handle observation uncertainty in a noisy environment under certain conditions.

Let $\mathcal{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ denote the training data for our models, assumed to be obtained



(a) LDA+ML (b) NALDA+ML

Figure 5.7: Evaluation of the scene. The models are standard GMMs with one component. White noise is added to the feature space during processing of the scene, with an SNR of about −18 dB. Dotted lines represent ground truth and solid lines are predictions. NALDA+ML leads to considerably less noisy predictions.

from data recorded in a noise-free environment. Now in application of these models, for example in a search-and-rescue scenario, observations will be noisy and thus the trained models are suboptimal. One possible solution is to adapt the models to account for the uncertainties in the observations.

With NALDA, noise is assumed as class-independent, additive with zero mean and with a time-varying covariance matrix (Kolossa *et al.*, 2013). Let $\boldsymbol{y}_c(\tau)$ denote the noisy observation at time frame τ so that

$$\boldsymbol{y}_c(\tau) = \boldsymbol{x}_c(\tau) + \boldsymbol{n}(\tau),$$

where we have introduced the subscript c to emphasize that the noise is assumed to be the same across all classes.

In (Kolossa *et al.*, 2013) it is shown that, given the covariance of the noise Σ^{n}, the within-class covariance of the noisy data is approximately given by $\Sigma^{\mathrm{wn}} \approx \Sigma^{\mathrm{w}} + \Sigma^{\mathrm{n}}$. Further, it is shown that using the adapted covariance Σ^{wn}, the noise optimal LDA projection matrix $\boldsymbol{W}^{\mathrm{NALDA}}$ can be computed in a similar way as in LDA using Eq. (5.4) where $\boldsymbol{w}_i^{\mathrm{LDA}}$ is replaced by $\boldsymbol{w}_i^{\mathrm{NALDA}}$.

Experiment on NIGENS database. The setting is the same as in the previous experiment, but superimposing the feature vectors with a known white noise (SNR of about -18 dB) in the testing phase. We have considered the following methods for evalua-

tion:

- LDA+ML, as described before.
- NALDA+ML: mapping the feature vectors to the projection space using $\widehat{\boldsymbol{y}} = \boldsymbol{W}^{\text{NALDA}}\boldsymbol{y}$, and modifying the model parameters as $\widehat{\boldsymbol{\mu}}_c = \boldsymbol{W}^{\text{NALDA}}\boldsymbol{\mu}_c$, $\widehat{\Sigma}_c = \boldsymbol{W}^{\text{NALDA}}\Sigma_c\boldsymbol{W}^{\text{NALDA}}$, then computing the maximum likelihood.

Figure 5.7-(a) and Figure 5.7-(b) show the result of this experiment using ML+LDA and NALDA+ML, respectively. It can be seen that the predictions using NALDA+ML are considerably less noisy. This experiment shows that given a good estimation of the noise statistics, it is possible to adapt the models to account for these uncertainties using NALDA. Although in this experiment the noise statistics were assumed known, the experiment proves the overall validity of this theoretical approach. Future work includes integrating a noise estimator in the Two!Ears framework.

5.2.7 Modelling and feature selection using mixture of factor analysers

As more features are extracted, it is important for the Two!Ears sound-event classification system to be able to handle high dimensionality. In this section, we look into a family of statistical methods known as mixture of factor analysers (MoFA) (Ghahramani and Hinton, 1997, Ghahramani and Beal, 2000). MoFA performs modeling and local dimensionality reduction in a single step.

Mixture of Factor Analysers (MoFA). A MoFA can in fact be considered as a GMM (see Section 5.2.5) with reduced parameterisations. Let $\mathcal{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ denote our training data. In MoFA, each d-dimensional real-valued data vector $\boldsymbol{x}$ is modelled using using p-dimensional vectors of factor $\boldsymbol{s}$, and a d-dimensional observation noise $\boldsymbol{u}$. Since in general $p < d$, MoFA performs the modelling in a reduced dimensionality space (Everitt, 1984). Ghahramani and Hinton (1997) show that the likelihood of $\mathcal{X}$ for a K-component MoFA can be expressed as

$$f(\mathcal{X}) = \prod_{n=1}^{N}\sum_{k=1}^{K}\tau_k\mathcal{N}(\boldsymbol{\mu}, \Lambda_k\Lambda_k^\top + \Psi_k),$$

where $\boldsymbol{\mu}$ denotes the mean vector, Λ_k is a $d \times p$ matrix known as the *factor loading matrix*, and Ψ_k denotes the covariance of the noise. We employ Bayesian inference as described by Ghahramani and Beal (2000) in order to learn the model parameters of the MoFA.

In the Two!Ears project, MoFA can play a useful role in particular when there is only a

small amount of training data available (for example, in future stages of TWO!EARS within the concept of active learning), and when modelling high-dimensional data without specific prior knowledge about the features. Since MoFA performs local feature selection and modelling in a single step, the resulting models can be used directly for the identification task.

In the following experiment, we compare the performance of a GMM classifier to MoFA. The experiment is designed to show the advantage of MoFA over GMM in handling small amounts of training data.

Experiment on the NIGENS Database. Consider our previous experiment setting with the selected sound classes from NIGENS. The performance of the following methods are compared:

- $\text{GMM}_{\text{fullset}}$: training models using GMM and all available training data;
- $\text{GMM}_{\text{subset}}$: training models using GMM and 10% of the available training data;
- $\text{MoFA}_{\text{subset}}$: training models using MoFA and 10% of the available training data.

A separate test set is evaluated against the trained models. The experiment is repeated 5 times and the average results, in terms of the balanced accuracy [5], are reported in Table 5.3. The classifiers are trained on the same data. We notice that $\text{MoFA}_{\text{subset}}$ is considerably more robust when only few training instances are available.

In this example, we have considered a GMM with a single component. Increasing the number of components in this case leads to overfitting and reduces the performance; this is because the features are quite simple. With increasing the complexity of the feature vectors, we might require larger number of components for the GMM.

Feature Selection using MoFA. So far we have used MoFA as a modelling tool with the property of local dimensionality reduction. In the following we discuss how we can use MoFA as an efficient tool for feature subset selection. This is important, since we are interested in knowing how much a particular feature contributes to the model's performance. If the influence is negligible, we can disregard it in future model training, and thus reduce the training time.

Each feature has its own weight for each mixture component of a MoFA, due to the local dimensionality reduction of MoFA. In the following we derive a criterion that can be used

5 The balanced accuracy is computed using: $BAC = \frac{sensitivity+specificity}{2}$.

Sound Class	$\text{MoFA}_{\text{subset}}$	$\text{GMM}_{\text{subset}}$	$\text{GMM}_{\text{fullset}}$
'FemaleSpeech'	90.2%	79.5%	93.2%
'BabyCrying'	90.1%	84.0%	92.8%
'Fire'	79.6%	72.2%	81.3%

Table 5.3: Performance comparison of MoFA and GMM with respect to the amount of training data in terms of the balanced accuracy.

to obtain a global dimensionality reduction using MoFA. The proposed method is validated with empirical evaluations.

Let $\widehat{\Lambda}_k = (\widehat{\boldsymbol{\lambda}}_{ik} \mid 1 \leq i \leq d)$, where $\widehat{\boldsymbol{\lambda}}_{ik} = (\widehat{\lambda}_{ik1}, \ldots, \widehat{\lambda}_{ikp})^\top$ denotes the estimated loading matrix of the k-th mixture component and $\widehat{\Psi}_k = (\widehat{\psi}_{ik} \mid 1 \leq i \leq d)$ is the estimated covariance matrix of the noise at the k-th MoFA component. Then we define:

$$\widehat{\Sigma}_k = (\widehat{\boldsymbol{\sigma}}_{ik} \mid 1 \leq i \leq d), \quad \widehat{\boldsymbol{\sigma}}_{ik} = \left(\widehat{\boldsymbol{\lambda}}_{ik}\widehat{\boldsymbol{\lambda}}_{ik}^\top + \widehat{\psi}_{ik} \mid 1 \leq i \leq d\right),$$

where $\widehat{\boldsymbol{\sigma}}_{ik}$ is a $p \times p$ covariance matrix at the i-th dimension and for the k-th mixture component, where $i \in \{1, \ldots, d\}$ and $k \in \{1, \ldots, K\}$. Note that the number of necessary mixture components, K, is determined automatically during learning.

Next, the contribution of each feature vector to the k-th mixture component is shown by a d-dimensional vector $\boldsymbol{w}_k = (w_{ik} \mid 1 \leq i \leq d)$, where w_{ik} is computed as:

$$w_{ik} = \frac{\widehat{\tau}_k \ \text{tr}(\widehat{\boldsymbol{\sigma}}_{ik})}{\sum_p \widehat{\tau}_k \ \text{tr}(\widehat{\boldsymbol{\sigma}}_{ik})}, \quad \forall \ 1 \leq i \leq d, \ 1 \leq k \leq K. \tag{5.6}$$

Here, $\widehat{\tau}_k$ is the estimated mixture weight and $\text{tr}(\cdot)$ denotes the mathematical trace operation. Note that $\sum_{i=1}^{d} w_{ik} = 1$ and $0 \leq w_{ik} \leq 1$ for all $k \in \{1, \ldots, K\}$.

Those features which have negligible weights can be disregarded by assigning a suitable threshold. Let $\mathcal{I}_k \subseteq \{1, \ldots, d\}$ be a set which includes indices of all feature vectors with significant contribution to the k-th mixture component as

$$\mathcal{I}_k = \underset{i}{\arg} \ w_{ik} > \frac{\text{t}}{\text{d}} \quad \forall \ 1 \leq i \leq d \tag{5.7}$$

where $\frac{t}{d}$ is a threshold (in the experiments $t = 0.5$, which means 50% of the average contribution). Then indices of all feature vectors with significant contribution across all mixture components can be included in a set $\mathcal{I}$ defined as

$$\mathcal{I} = \mathcal{I}_1 \cup \ldots \cup \mathcal{I}_k \cup \ldots \cup \mathcal{I}_K,$$

where $\mathcal{I} \subseteq \{1, \ldots, d\}$.

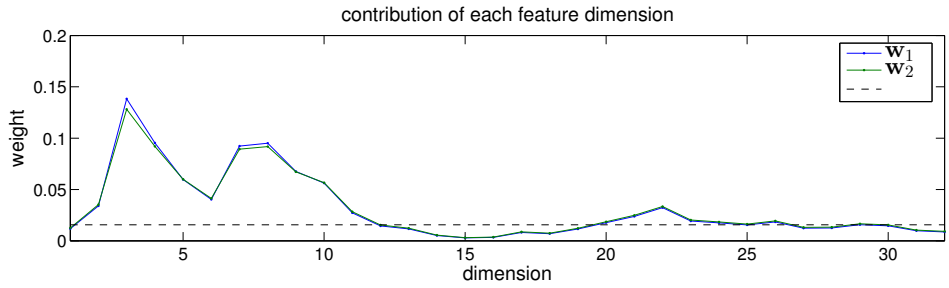
Experiment on the NIGENS database. Consider the previous experiment. The goal now is to evaluate the change in performance as the result of using the proposed global dimension reduction scheme. The following methods are compared:

- MoFA$_{\mathcal{D}}$: MoFA with full dimension feature vector $\mathcal{D} = \{1, \ldots, d\}$;
- MoFA$_{\mathcal{I}}$: MoFA after global dimension reduction using $\mathcal{I} \subseteq \{1, \ldots, d\}$ computed by Eq. (5.7).

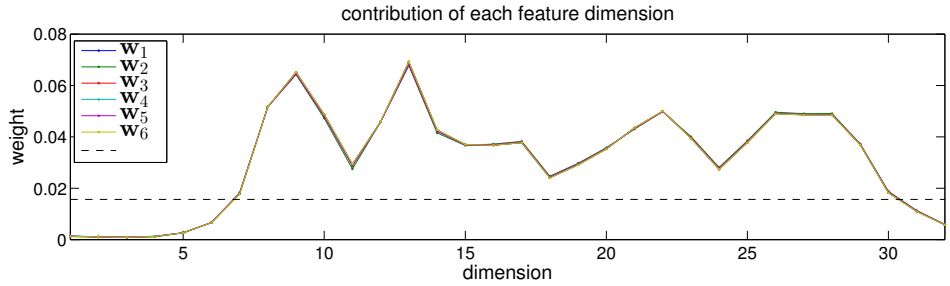
Table 5.4 shows results of this experiment. Although the classification performance is slightly reduced when using MoFA$_{\mathcal{I}}$ compared to that of MoFA$_{\mathcal{D}}$, the result is still of interest given a considerable speed-up: for our data, the training time using MoFA$_{\mathcal{I}}$ was on average three to four times faster compared to that of MoFA$_{\mathcal{D}}$. Figure 5.8 shows the contribution of each feature to the overall performance, i.e., $\boldsymbol{w}_k$, computed using (5.6), for the three sound classes.

Sound Class	MoFA$_{\mathcal{D}}$	MoFA$_{\mathcal{I}}$
'FemaleSpeech'	91.2%	89.1%
'BabyCrying'	90.3%	88.60%
'Fire'	80.4%	77.2%

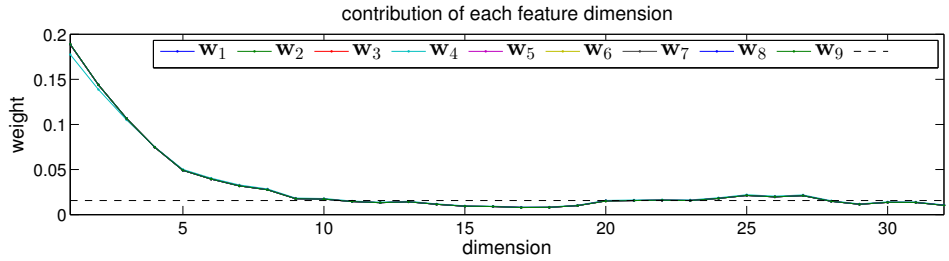
Table 5.4: Performance comparison (BAC) of MoFA with full dimension feature vector, MoFA$_{\mathcal{D}}$, and with reduced dimensionality, MoFA$_{\mathcal{I}}$, on the same training and test data. Only 10% of the available data has been used for training.



(a) "FemaleSpeech"



(b) "BabyCrying"



(c) "Fire"

Figure 5.8: Contribution of each feature to the prediction. Weights are between 0 and 1. Higher weights imply higher contribution. The dotted line is the chosen threshold, here 50% of the average contribution. The necessary number of mixture components are determined automatically during learning; each colour represents the behaviour of a certain mixture component in its respective class. Note that the weight plots for different mixture components are almost identical.

5.3 Speaker identity

The previous section has presented progress on classification of general sounds. In this section we investigate a more specific sound classification problem — speaker identification. Speaker identification, including gender identification, has a particular relevance within the TWO!EARS project and is itself a well-studied field. This section focuses on the state-of-the-art in speaker identification and presents initial results in the context of the binaural setting of the TWO!EARS project. The most suitable techniques will be identified and employed by the `SpeakerID` knowledge source within the blackboard framework.

5.3.1 Models

GMM-UBM

Many widely used speaker identification methods are based on GMMs, which model feature vectors from each speaker with a GMM as a weighted sum of Gaussian distributions. The basic approach is realised by the log-likelihood ratio test from signal detection theory. GMMs are used for both target and background models. The target model is trained using speech signals from the target speaker. The background model, often referred to as the universal background model (UBM), is trained using speech signals from as many speakers as possible. Given a test signal $\vec{x}$, the log-likelihood ratio between the target model and the UBM is computed as

$$\Lambda = \log \frac{p(\vec{x}|\lambda_s)}{p(\vec{x}|\lambda_{ubm})} \tag{5.8}$$

where $p(\vec{x}|\lambda)$ is the likelihood of $\vec{x}$ given model λ. The log-likelihood ratio is compared to a pre-defined threshold for acceptance of the target speaker.

The parameters of universal background model (UBM) are estimated by maximising the likelihood of a set of training feature vectors using expectation-maximisation (EM). The target model, however, is often trained by adapting from the background model, particularly when there is limited amount of training speech from the target speaker. We employ a popular adaptation method, relevance maximum *a posterioi* (MAP) adaptation, which is a linear interpolation of all mixture components of the UBM that increase likelihood of speech from the target speaker. It is found empirically better to only adapt the mean vectors from the UBM.

i-vectors

One of the problems of relevance maximum *a posteriori* (MAP) is that it adapts not only to speaker-specific information but also to channel and other undesired factors. This problem is addressed by the joint factor analysis (JFA) model proposed for the GMM framework (Kenny *et al.*, 2008). joint factor analysis (JFA) has become the basis for the state-of-the-art in speaker identification, and has the following two properties. First, a speech recording of a variable length is represented by a fixed-length supervector. Secondly, and more importantly, such a supervector representation allows explicit modelling of the undesired variability in the speech signal. The speaker supervector can be decomposed into speaker-dependent factors and channel-dependent factors. The values of the speaker factors are assumed to be the same for all recordings of the speaker but the channel factors are assumed to vary from one recording to another. The supervector is constructed by concatenating mean vectors from each GMM component. As a result, the JFA model has to estimate hyper-parameters with a huge dimensionality. Therefore principle component analysis (PCA) is typically applied with JFA.

Dehak *et al.* (2011) showed that the channel factors in the JFA model also contain some speaker information. They proposed the i-vector model based on JFA, which makes no distinction between speaker effects and channel effects in the GMM supervector space. Instead, a total variability space that contains both speaker and channel variabilities is defined. The hyperparameters of the i-vector model can be extracted by using the EM algorithm and have a much lower dimensionality than JFA. Channel compensation can be further applied in the i-vector space using probabilistic linear discriminant analysis (PDLA) (Prince and Elder, 2007).

Given a test speech signal, the test i-vector w is extracted and compared to speaker model i-vectors w_s. Let H_1 indicate the hypothesis that the two i-vectors are from the same speaker, and H_0 be the hypothesis that the two i-vectors are from different speakers. The verification score is computed as the log-likelihood ratio of the same versus different speaker models hypotheses:

$$\Lambda = \log \frac{p(w, w_s|H_1)}{p(w|H_0)p(w_s|H_0)} \tag{5.9}$$

Similar to the GMM-UBM approach, the log-likelihood ratio is compared to a pre-defined threshold to make a decision on the target speaker.

5.3.2 Features

The auditory front-end in the Two!Ears system produces ratemaps from a bank of 32 gammatone filters with centre frequencies equally spaced on the equivalent rectangular bandwidth (ERB) scale between 80 and 8000 Hz (Wang and Brown, 2006). Ratemaps are

spectral features that represent a map of auditory nerve firing rate (Brown and Cooke, 1994). In the binaural setting of TWO!EARS, ratemaps can be computed from both ears. The *average* ratemaps are constructed by taking the average ratemap value of each pair of time-frequency cells between the two ears. The *binaural* ratemaps are constructed by concatenating the left ear and right ear ratemaps.

Most the state-of-the-art speaker identification systems employ mel-frequency cepstral coefficient (MFCC) features. To make the ratemap features more orthogonal, we also apply a discrete cosine transform (DCT) to ratemaps from individual ears to produce *binaural* gammatone filter cepstral coefficients (GFCC).

5.3.3 Evaluation

We used speech materials from the GRID corpus (Cooke *et al.*, 2006) for initial evaluation. The corpus consists of 1000 simple command-like sentences spoken by each of 34 speakers. The monaural speech signals were spatialised by convolution with an anechoic HRIR measured with a KEMAR dummy head (Wierstorf *et al.*, 2011). 72 azimuth angles between $0\,^\circ$ and $359\,^\circ$ (with an angular resolution of $5\,^\circ$) were used. For each speaker, training was done using 6 different sentences from each azimuth angle, totalling 432 sentences per speaker. For testing, we used 288 sentences per speaker (4 sentences for each azimuth angle) that were not used for training.

The UBM was trained on training features from all the 34 speakers using 128-component GMMs. For the GMM-UBM model, each speaker model was adapted from the UBM using mean-only MAP relevance adaptation. For the i-vector model, the same UBM was used to train the total variability subspace and extract the sentence i-vectors. Final model i-vectors were obtained by taking the average of the extracted i-vectors. A Gaussian PLDA model was trained with the extracted i-vectors to score the testing trials.

The equal error rate (EER) performance of both models is shown in Table 5.5. Both models produced very similar results as all the sentences from the GRID corpus were recorded in a very similar setup and there was little channel variability. We plan next to evaluate the models in more adverse conditions such as in the presence of overlapping sources and reverberation.

Table 5.5: Equal error rate performance (%) of two speaker identification models using various type of features.

	Average Ratemap	Binaural Ratemap	Binaural GFCC
GMM-UBM	0.8	1	0.5
i-vector	0.2	1.3	0.8

6 Conclusions

In the first year of the Two!Ears project, the software architecture has been defined, spanning all modules from signal generation or acquisition, via pre-segmentation and feature extraction to the cognitive stage and the feedback loops. The architecture has been defined in a flexible manner, allowing the composition of a world model on the system's blackboard and the planning and feedback processes required for active exploratory listening. With the presented architecture, already, many of the envisaged tasks have been solved.

For example, feedback to head-position control allows improved source tracking, internal adaptation processes allow robust operation in high environmental noise levels, the pre-segmentation stage can distinguish sources of interest from sources of noise, and the implemented machine learning strategies allow the systems to quickly learn tasks such as the identification of speakers and auditory objects based on state-of-the-art methods.

In the next two years, this presented framework will form the basis for both fundamental and applied research on active, model-based listening. Statistical methods can be combined with logical reasoning, and an understanding of the environment can be attained by composing models of the sources and the environment with those of position and movement possibilities of the robotic system itself, to arrive at an active and exploratory behaviour directed towards an optimal understanding of auditory and audiovisual scenes. This, it is envisaged, will help to comprehend, describe and ultimately emulate the impressive performance that humans achieve in understanding their environment even from few, and unreliable stimuli, which is, as yet, unmatched in machine listening systems.

Appendices

A. Publications during this period

Published

- Käsbach, J, May, T., Oskarsdottir, G., Jeong, C.-H. and Chang, J. (2014) " The effect of interaural-time-difference fluctuations on apparent source width", *FORUM ACUSTICUM*, Krakow.
- May, T. and Dau, T. (2015) "Computational speech segregation based on an auditory-inspired modulation analysis", *Journal of the Acoustical Society of America*, in press.
- May, T. and Dau, T. (2014) "Requirements for the evaluation of computational speech segregation systems", *Journal of the Acoustical Society of America*, **136**(6), pp.EL398-EL404.
- May, T. and Gerkmann, T. (2014) "Generalization of supervised learning for binary mask estimation", *Proceedings of IEEE IWAENC*, Juan le pins, France.
- Schymura, C., Ma, N., Brown, G., Walther, T., and Kolossa, D. (2014) "Binaural sound source localisation using a Bayesian-network-based blackboard system and hypothesis-driven feedback", *FORUM ACUSTICUM*, Krakow.

Submitted and in preparation

- Blauert, J., Kolossa, D., and Danès, P. (2015), "Feedback loops in engineering models of binaural listening", submitted to JASA Express Letters.
- Ma, N., May, T., Wierstorf, H., and Brown, G. (2015), "A machine-hearing system exploiting head movements for binaural sound localisation in reverberant conditions", submitted to *ICASSP 2015*.
- May, T., Ma, N., and Brown, G. (2015), "Robust localisation of multiple speakers exploiting head movements and multi-conditional training of binaural cues", submitted

to *ICASSP 2015*.

B. Candidate visual features

Below, we define a list of visual features that could become of interest in the course of the Two!Ears project. The listed cues extend from basic features (like skin color) to complex descriptors (SIFT, HOG,...). The main purpose of the following enumeration is to give an overview of contemporary techniques in computer vision that relate to feature-based scene analysis. We emphasize the pros and cons of each feature, as learned from the literature or experimental evaluation. Attached literature hints provide a first idea of the 'architecture' of each proposed feature and give some examples of its likely application:

- Haar-like wavelets (used, e.g., in the 'Viola-Jones' face detector, cf. Viola and Jones (2001)) – advantages: very fast and good true positive rate with adequate training data; disadvantages: non-negligible false positive detection rate, cascade training might be expensive.

- Discrete cosine transforms – advantages: compact feature representation, fast calculation, nearly uncorrelated; disadvantages: depend on scale, rotation, illumination.

- Histograms of oriented gradients (HOG), used, e.g., for pedestrian detection, cf. Dalal and Triggs (2005) – advantages: once trained, detection of the 'target object' seems reliable. Potentially, training on real-world data sets has a certain chance of becoming useful in a simulated environment. Disadvantages: require quite a lot of training data, however, data sets for pedestrians seem available.

- Scale invariant feature transforms (SIFT) represent a state-of-the-art method for object detection in cluttered environments with partial occlusion, cf. Lowe (1999) – advantages: 'invariant to image scaling, translation, and rotation, and partially invariant to illumination changes and affine or 3D projection' (Lowe, 1999); disadvantages: non-rigid deformations could hamper visual analysis.

- Speeded up robust features (SURF) can basically be seen as a faster variant of the SIFT scheme, see Bay *et al.* (2008) – advantages: similar to SIFT, but faster; disadvantages: cf. SIFT.

- Shape Context: a descriptor that encodes the silhouette of an object, storing a 'signature' histogram of the silhouette, for details, please refer to Belongie *et al.* (2002) – advantages: somewhat invariant to small deformations, shows potential for generalization; disadvantages: non-negligible matching costs.

- Gabor features: a biologically motivated image descriptor that tries to mimic the 'oriented edge' analysis capabilities of the human visual cortex, cf. Daugman (1985) – advantages: biological motivation, well-established theory (Gabor features are widespread in face recognition); disadvantages: generally requires combination of Gabor filters with different orientations/scales to achieve acceptable analysis results, dense calculation for larger images might be computationally expensive. On the other hand, a dense calculation is rarely required.

- Skin color: this nearly self-explanatory feature helps to focus on regions of interest (ROI) in an image when it comes to face detection/recognition – advantages: fast calculation, good ROI 'pre-selector'; disadvantages: without cue fusion, skin color is a weak cue and can easily give numerous false positives.

- Silhouette: being also quite basic with respect to information content, silhouettes show potential for inter-subject generalization. For Two!Ears, this descriptor could become extremely helpful for person detection: silhouette features should behave in an invariant manner in simulated and real environments – advantages: putative invariance, needs testing. Fast calculation. Can possibly be trained with synthetic data and then directly be applied to real world scenarios (also requires testing).

C. Example of BEFT

A video showing a typical scenario rendered with the BEFT virtual 3D visualisation environment can be seen by following this link:

`http://twoears.aipa.tu-berlin.de/2014/11/beft-demonstration/`

Acronyms

ACG autocorrelogram

AFE auditory front-end

AMS amplitude modulation spectrogram

ASA auditory scene analysis

BRIR binaural room impulse response

BEFT Bochum Experimental Feedback Testbed

CF centre frequency

CASA computational auditory scene analysis

EM expectation-maximisation

ERB equivalent rectangular bandwidth

F0 fundamental frequency

GMM Gaussian mixture model

GMM-UBM Gaussian mixture models with a universal background model

HRIR head-related impulse response

IBM Ideal binary mask

JFA joint factor analysis

KF Kalman Filter

KS knowledge source

LC local criterion

LDA linear discriminant analysis

MAP maximum *a posterioi*

MCT multi-condition training

MFCC mel-frequency cepstral coefficient

ML maximum likelihood

MoFA mixture of factor analysers

NALDA noise-adaptive linear discriminant analysis

PDLA probabilistic linear discriminant analysis

RMS root mean square

ROS robot operating system

SNR signal-to-noise ratio

SVM support vector machine

T-F time-frequency

UBM universal background model

UKF Unscented Kalman Filter

VTE virtual test environment

WP1 work package one

WP2 work package two

WP3 work package three

WP4 work package four

WP5 work package five

XML extensible markup language

Bibliography

Abdelaziz, A. H. and Kolossa, D. (**2014**), "Dynamic Stream Weight Estimation in Coupled-HMM-based Audio-visual Speech Recognition Using Multilayer Perceptrons," in *Proc. Interspeech*. (Cited on page 53)

Adiloglu, K., Anniés, R., Wahlen, E., Purwins, H., and Obermayer, K. (**2012**), "A graphical representation and dissimilarity measure for basic everyday sound events," *Audio, Speech, and Language Processing, IEEE Transactions on* **20**(5), pp. 1542–1552. (Cited on page 39)

Almanza-Ojeda, D. L., Devy, M., and Herbulot, A. (**2011**), "Active Method for Mobile Object Detection from an Embedded Camera, Based on a Contrario Clustering," in *Informatics in Control, Automation and Robotics*, vol. 89, pp. 267–280. (Cited on page 50)

Assmann, P. and Summerfield, Q. (**1990**), "Modeling the perception of concurrent vowels: Vowels with different fundamental frequencies," *J. Acoust. Soc. Am.* **88**(2), pp. 680–697. (Cited on page 28)

Bacon, S. P. and Grantham, D. W. (**1989**), "Modulation masking: Effects of modulation frequency, depths, and phase," *Journal of the Acoustical Society of America* **85**(6), pp. 2575–2580. (Cited on page 31)

Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (**2008**), "Speeded-Up Robust Features (SURF)," *Comput. Vis. Image Underst.* **110**(3), pp. 346–359. (Cited on page 84)

Belongie, S., Malik, J., and Puzicha, J. (**2002**), "Shape matching and object recognition using shape contexts," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24**(4), pp. 509–522. (Cited on page 84)

Bishop, C. M. (**2006**), *Pattern Recognition and Machine Learning*, Springer. (Cited on page 68)

Blauert, J. (**1997**), *Spatial hearing - The psychophysics of human sound localization*, The MIT Press, Cambride, MA, USA. (Cited on page 55)

Blender Foundation (**2014**), "Blender - 3D open source animation suite," URL `http://www.blender.org/`. (Cited on page 46)

Bregman, A. S. (**1990**), *Auditory scene analysis: The perceptual organization of sound*, The MIT Press, Cambridge, MA, USA. (Cited on pages 7, 27, 37, and 63)

Brown, G. J. and Cooke, M. P. (**1994**), "Computational auditory scene analysis," *Computer Speech and Language* **8**(4), pp. 297–336. (Cited on page 80)

Brungart, D. S., Chang, P. S., Simpson, B. D., and Wang, D. L. (**2006**), "Isolating the energetic component of speech-on-speech masking with ideal time-frequency segregation," *Journal of the Acoustical Society of America* **120**(6), pp. 4007–4018. (Cited on page 31)

Bullet (**2014**), "The Bullet Physics Engine," URL `http://bulletphysics.org/wordpress/`. (Cited on page 46)

Chang, C.-C. and Lin, C.-J. (**2011**), "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology* **2**, pp. 27:1–27:27, software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`. (Cited on page 66)

Christensen, H., Ma, N., Wrigley, S., and Barker, J. (**2009**), "A speech fragment approach to localising multiple speakers in reverberant environments," in *Proc. IEEE ICASSP?09*, pp. 4593–4596. (Cited on page 30)

Cooke, M., Barker, J., Cunningham, S., and Shao, X. (**2006**), "An audio-visual corpus for speech perception and automatic speech recognition," *Journal of the Acoustical Society of America* **120**(5), pp. 2421–4. (Cited on pages 51 and 80)

Cooke, M., Brown, G. J., Crawford, M., and Green, P. (**1993**), "Computational auditory scene analysis: listening to several things at once," *Endeavour* **17**(4), pp. 186–190. (Cited on page 7)

Cooke, M., Green, P., Josifovski, L., and Vizinho, A. (**2001**), "Robust automatic speech recognition with missing and unreliable acoustic data," *Speech Communication* **34**, pp. 267–285. (Cited on page 31)

Dalal, N. and Triggs, B. (**2005**), "Histograms of oriented gradients for human detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 886–893. (Cited on page 84)

Dau, T., Kollmeier, B., and Kohlrausch, A. (**1997**a), "Modeling auditory processing of amplitude modulation. I. Detection and masking with narrow-band carriers," *Journal of the Acoustical Society of America* **102**(5), pp. 2892–2905. (Cited on page 31)

Dau, T., Kollmeier, B., and Kohlrausch, A. (**1997**b), "Modeling auditory processing of amplitude modulation. II. Spectral and temporal integration," *Journal of the Acoustical Society of America* **102**(5), pp. 2906–2919. (Cited on page 31)

Daugman, J. G. (**1985**), "Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters," *J. Opt. Soc. Am. A* **2**(7), pp. 1160–1169. (Cited on page 85)

David, H. A. and Nagaraja, H. N. (**2003**), *Order Statistics*, Wiley, 3 ed. (Cited on page 64)

Dehak, N., Kenny, P., Dehak, R., Dumouchel, P., and Ouellet, P. (**2011**), "Front-End Factor Analysis for Speaker Verification," *IEEE Transactions on Audio, Speech, and Language Processing* **19**(4), pp. 788–798. (Cited on page 79)

Di Marco, D., Koch, A., Zweigle, O., Haussermann, K., Schiessle, B., Levi, P., Galvez-Lopez, D., Riazuelo, L., Civera, J., Montiel, J. M. M., Tenorth, M., Perzylo, A., Waibel, M., and van de Molengraft, R. (**2012**), "Creating and using RoboEarth object models," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 3549–3550. (Cited on page 53)

Ellis, D. P. W. (**1996**), "Prediction-driven computational auditory scene analysis," Ph.D. thesis, Massachusetts Institute of Technology. (Cited on page 7)

Erman, L. D., Hayes-Roth, F., Lesser, V. R., and Reddy, D. R. (**1980**), "The Hearsay-II speech understanding system: integrating knowledge to resolve uncertainty," *Computing Surveys* **12**(2), pp. 213–253. (Cited on pages 7 and 19)

Everitt, B. S. (**1984**), *An Introduction to Latent Variable Models*, Chapman and Hall, London. (Cited on page 72)

Ewert, S. D. and Dau, T. (**2000**), "Characterizing frequency selectivity for envelope fluctuations," *Journal of the Acoustical Society of America* **108**(3), pp. 1181–1196. (Cited on pages 31 and 33)

Gamez, D. M. and Devy, M. (**2012**), "Active visual-based detection and tracking of moving objects from clustering and classification methods," in *Proc. Advanced Concepts for Intelligent Vision Systems (ACIVS 2012), Brno (Czetch Republic)*. (Cited on pages 50 and 52)

Garofolo, J. S., Lamel, L. F., Fisher, W. M., Fiscus, J. G., Pallett, D. S., and Dahlgren, N. L. (**1993**), "DARPA TIMIT Acoustic-phonetic continuous speech corpus CD-ROM," *National Inst. Standards and Technol. (NIST)* . (Cited on page 60)

Gate, G., Breheret, A., and Nashashibi, F. (**2009**), "Centralised Fusion for Fast People Detection in Dense Environments," in *IEEE Int. Conf. on Robotics Automation (ICRA), Kobe, Japan.* (Cited on page 53)

Georganti, E., May, T., van de Par, S., and Mourjopoulos, J. (**2013**), "Sound Source Distance Estimation in Rooms based on Statistical Properties of Binaural Signals," *Audio, Speech, and Language Processing, IEEE Transactions on* **21**(8), pp. 1727–1741.

(Cited on page 44)

Ghahramani, Z. and Beal, M. J. (**2000**), "Variational Inference for Bayesian Mixtures of Factor Analysers," in *In Advances in Neural Information Processing Systems 12*, MIT Press, pp. 449–455. (Cited on page 72)

Ghahramani, Z. and Hinton, G. E. (**1997**), "The EM Algorithm for Mixtures of Factor Analyzers," Tech. rep., Department of Computer Science, University of Toronto. (Cited on page 72)

Godsmark, D. and Brown, G. J. (**1999**), "A Blackboard Architecture for Computational Auditory Scene Analysis," *Speech Commun.* **27**(3-4), pp. 351–366, URL `http://dx.doi.org/10.1016/S0167-6393(98)00082-X`. (Cited on page 7)

Graves, A., Mohamed, A.-r., and Hinton, G. (**2013**), "Speech recognition with deep recurrent neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, IEEE, pp. 6645–6649. (Cited on page 64)

Guyon, I. and Elisseeff, A. (**2003**), "An Introduction to Variable and Feature Selection," *J. Mach. Learn. Res.* **3**, pp. 1157–1182. (Cited on page 65)

Han, K. and Wang, D. L. (**2012**), "A classification based approach to speech segregation," *Journal of the Acoustical Society of America* **132**(5), pp. 3475–3483. (Cited on pages 31 and 32)

Hawley, M. L., Litovsky, R. Y., and Colburn, H. S. (**1999**), "Speech intelligibility and localization in a multi-source environment," *Journal of the Acoustical Society of America* **105**(6), pp. 3436–3448. (Cited on page 55)

Healy, E. W., Yoho, S. E., Wang, Y., and Wang, D. (**2013**), "An algorithm to improve speech recognition in noise for hearing-impaired listeners," *Journal of the Acoustical Society of America* **134**(6), pp. 3029–3038. (Cited on pages 31, 32, and 34)

Hinterstoisser, S., Lepetit, V., Ilic, S., Fua, P., and Navab, N. (**2010**), "Dominant orientation templates for real-time detection of texture-less objects," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 2257–2264. (Cited on page 53)

Hochreiter, S. and Schmidhuber, J. (**1997**), "Long short-term memory," *Neural computation* **9**(8), pp. 1735–1780. (Cited on page 64)

Hosking, J. R. M. (**1990**), "L-moments: analysis and estimation of distributions using linear combinations of order statistics," *Journal of the Royal Statistical Society. Series B (Methodological)* **52**(1), pp. 105–124. (Cited on page 64)

Houtgast, T. (**1989**), "Frequency selectivity in amplitude-modulation detection," *Journal*

of the Acoustical Society of America **85**(4), pp. 1676–1680. (Cited on page 31)

Hu, G. and Wang, D. L. (**2007**), "Auditory segmentation based on onset and offset analysis," *IEEE Transactions on Audio, Speech, and Language Processing* **15**(2), pp. 396–405. (Cited on pages 38 and 39)

Hummersone, C., Mason, R., and Brookes, T. (**2010**), "Dynamic precedence effect modeling for source separation in reverberant environments," *IEEE Transactions on Audio, Speech, and Language Processing* **18**(7), pp. 1867–1871. (Cited on page 59)

Jørgensen, S. and Dau, T. (**2011**), "Predicting speech intelligibility based on the signal-to-noise envelope power ratio after modulation-frequency selective processing," *Journal of the Acoustical Society of America* **130**(3), pp. 1475–1487. (Cited on page 33)

Jørgensen, S., Ewert, S. D., and Dau, T. (**2013**), "A multi-resolution envelope-power based model for speech intelligibility," *Journal of the Acoustical Society of America* **134**(1), pp. 1–11. (Cited on page 33)

Julier, S. J. and Uhlmann, J. K. (**2004**), "Unscented Filtering and Nonlinear Estimation," in *Proceedings of the IEEE*, pp. 401–422. (Cited on page 40)

Kalman, R. E. (**1960**), "A New Approach to Linear Filtering And Prediction Problems," *ASME Journal of Basic Engineering* . (Cited on page 40)

Kenny, P., Ouellet, P., Dehak, N., Gupta, V., and Dumouchel, P. (**2008**), "A study of inter-speaker variability in speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing* **16**(5), pp. 980–988. (Cited on page 79)

Kim, C., Mason, R., and Brookes, T. (**2013**), "Head Movements Made by Listeners in Experimental and Real-Life Listening Activities," *J. Audio Eng. Soc.* **61**(6), pp. 425–438. (Cited on page 58)

Kim, G., Lu, Y., Hu, Y., and Loizou, P. C. (**2009**), "An algorithm that improves speech intelligibility in noise for normal-hearing listeners," *Journal of the Acoustical Society of America* **126**(3), pp. 1486–1494. (Cited on pages 31, 32, 34, and 35)

Kjems, U., Boldt, J. B., Pedersen, M. S., Lunner, T., and Wang, D. L. (**2009**), "Role of mask pattern in intelligibility of ideal binary-masked noisy speech," *Journal of the Acoustical Society of America* **126**(3), pp. 1415–1426. (Cited on page 31)

Klapuri, A. (**1999**), "Sound onset detection by applying psychoacoustic knowledge," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Phoenix, Arizona, USA, vol. 6, pp. 3089–3092. (Cited on page 37)

Kollmeier, B. and Koch, R. (**1994**), "Speech enhancement based on physiological and

psychoacoustical models of modulation perception and binaural interaction," *Journal of the Acoustical Society of America* **95**(3), pp. 1593–1602. (Cited on page 31)

Kolossa, D., Zeiler, S., Saeidi, R., and Fernandez Astudillo, R. (**2013**), "Noise-Adaptive LDA: A New Approach for Speech Recognition Under Observation Uncertainty," *IEEE Signal Processing Letters* **20**(11), pp. 1018–1021. (Cited on pages 69, 70, and 71)

Lesser, V. R., Nawab, S. H., and Klassner, F. I. (**1995**), "IPUS: An architecture for the integrated processing and understanding of signals," *Artificial Intelligence* **77**, pp. 129–171. (Cited on page 7)

Li, N. and Loizou, P. C. (**2008**), "Factors influencing intelligibility of ideal binary-masked speech: Implications for noise reduction," *Journal of the Acoustical Society of America* **123**(3), pp. 1673–1682. (Cited on page 31)

Lowe, D. (**1999**), "Object recognition from local scale-invariant features," in *The Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, pp. 1150–1157. (Cited on page 84)

Lu, Y.-C. and Cooke, M. (**2010**), "Binaural Estimation of Sound Source Distance via the Direct-to-reverberant Energy Ratio for Static and Moving Sources," *Trans. Audio, Speech and Lang. Proc.* **18**(7), pp. 1793–1805. (Cited on page 44)

Lu, Y.-C. and Cooke, M. (**2011**), "Motion Strategies for Binaural Localisation of Speech Sources in Azimuth and Distance by Artificial Listeners," *Speech Commun.* **53**(5), pp. 622–642. (Cited on page 44)

Ma, N., Barker, J., Christensen, H., and Green, P. (**2013**), "A hearing-inspired approach for distant-microphone speech recognition in the presence of multiple sound sources," *Comput. Speech Lang.* **27**(3), pp. 820–836. (Cited on page 31)

Ma, N., Green, P., Barker, J., and Coy, A. (**2007**), "Exploiting correlogram structure for robust speech recognition with multiple speech sources," *Speech Commun.* **49**(12), pp. 874–891. (Cited on page 29)

Mallet, A. and Herrb, M. (**2011**), "Recent developments of the GenoM robotic component generator," in *6th National Conference on Control Architectures of Robots*, INRIA Grenoble Rhône-Alpes, Grenoble, France, no. Rapport LAAS n° 11353, p. 4 p. (Cited on page 51)

May, T. and Dau, T. (**2013**), "Environment-aware ideal binary mask estimation using monaural cues," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pp. 1–4. (Cited on pages 31, 32, and 34)

May, T. and Dau, T. (**2014**), "Requirements for the evaluation of computational speech

segregation systems," *Journal of the Acoustical Society of America* **136**(6), pp. EL398–EL404. (Cited on pages 31 and 32)

May, T. and Gerkmann, T. (**2014**), "Generalization of supervised learning for binary mask estimation," in *Proc. IWAENC*, Juan le pins, France. (Cited on page 32)

May, T., van de Par, S., and Kohlrausch, A. (**2011**), "A probabilistic model for robust localization based on a binaural auditory front-end," *IEEE Transactions on Audio, Speech, and Language Processing* **19**(1), pp. 1–13. (Cited on page 56)

May, T., van de Par, S., and Kohlrausch, A. (**2012**a), "A binaural scene analyzer for joint localization and recognition of speakers in the presence of interfering noise sources and reverberation," *IEEE Transactions on Audio, Speech, and Language Processing* **20**(7), pp. 2016–2030. (Cited on page 31)

May, T., van de Par, S., and Kohlrausch, A. (**2012**b), "Noise-robust speaker recognition combining missing data techniques and universal background modeling," *IEEE Transactions on Audio, Speech, and Language Processing* **20**(1), pp. 108–121. (Cited on page 31)

May, T., van de Par, S., and Kohlrausch, A. (**2013**), "Binaural Localization and Detection of Speakers in Complex Acoustic Scenes," in *The technology of binaural listening*, edited by J. Blauert, Springer, Berlin–Heidelberg–New York NY, chap. 15, pp. 397–425. (Cited on page 56)

McAnally, K. I. and Martin, R. L. (**2014**), "Sound localization with head movements: Implications for 3D audio displays," *Frontiers in Neuroscience* **8**, pp. 1–6. (Cited on page 55)

Meddis, R. and Hewitt, M. (**1991**), "Virtual pitch and phase sensitivity of a computer model of the auditory periphery. I: Pitch identification," *J. Acoust. Soc. Am.* **89**(6), pp. 2866–2882. (Cited on page 28)

Meddis, R. and Hewitt, M. (**1992**), "Modeling the identification of concurrent vowels with different fundamental frequencies," *J. Acoust. Soc. Am.* **91**(1), pp. 233–245. (Cited on page 28)

MORSE (**2014**), "MORSE, the Modular OpenRobots Simulation Engine," URL `https://www.openrobots.org/wiki/morse/`. (Cited on pages 9 and 46)

Munkres, J. (**1957**), "Algorithms for the Assignment and Transportation Problems," *Journal of the Society of Industrial and Applied Mathematics* **5**(1), pp. 32–38. (Cited on page 39)

Nielsen, J. B. and Dau, T. (**2011**), "The Danish hearing in noise test," *Int. J. Audiol.* **50**(3), pp. 202–208. (Cited on page 35)

OGRE (**2014**), "OGRE - Open Source 3D Graphics Engine," URL `http://www.ogre3d.`

org/. (Cited on page 45)

ORK (**2013**), "The Object Recognition Kitchen project," URL http://wg-perception.github.io/object_recognition_core/. (Cited on page 53)

Perrett, S. and Noble, W. (**1997**), "The effect of head rotations on vertical plane sound localization," *J. Acoust. Soc. Am.* **102**(4), pp. 2325–2332. (Cited on pages 58 and 62)

Prince, S. J. and Elder, J. H. (**2007**), "Probabilistic linear discriminant analysis for inferences about identity," in *IEEE ICCV*, pp. 1–8. (Cited on page 79)

ROS (**2014**), "The Robot Operating System," URL http://www.ros.org/. (Cited on page 46)

Russell, S. J. and Norvig, P. (**2003**), *Artificial Intelligence: A Modern Approach*, Pearson Education, 2 ed. (Cited on page 20)

Scharstein, D. and Zeliski, R. (**2012**), "The MiddleBury Computer Vision Pages," URL http:/vision.middlebury.edu/stereo//. (Cited on page 49)

Schölkopf, B. and Smola, A. J. (**2002**), *Learning with kernels: support vector machines, regularization, optimization, and beyond*, MIT press. (Cited on page 66)

Shinn-Cunningham, B. G. (**2000**), "Distance cues for virtual auditory space," in *IEEE 2000 International Symposium on Multimedia Information Processing*. (Cited on page 44)

Slaney, M. and Lyon, R. (**1990**), "A perceptual pitch detector," in *Proc. IEEE ICASSP'90*, Albequerque, pp. 357–360. (Cited on page 28)

Smith, E. and Lewicki, M. S. (**2005**), "Efficient Coding of Time-Relative Structure Using Spikes," *Neural Computation* **17**(1), pp. 19–45. (Cited on page 39)

Sola, J. (**2007**), "Towards Visual Localization, Mapping and Moving Objects Tracking by a Mobile Robot: a Geometric and Probabilis- tic Approach," *PhD thesis, Institut National Polytechnique de Toulouse* . (Cited on pages 48 and 52)

Summerfield, Q., Lea, A., and Marshall, D. (**1990**), "Modelling auditory scene analysis: strategies for source segregation using autocorrelograms," in *Proc. Institute of Acoustics*, vol. 12, pp. 507–514. (Cited on page 29)

Tchorz, J. and Kollmeier, B. (**2003**), "SNR estimation based on amplitude modulation analysis with applications to noise suppression," *IEEE Transactions on Audio, Speech, and Language Processing* **11**(3), pp. 184–192. (Cited on pages 31 and 32)

Turgeon, M., Bregman, A. S., and Ahad, P. A. (**2002**), "Rhythmic masking release: Contribution of cues for perceptual organization to the cross-spectral fusion of concurrent

narrow-band noises," *Journal of the Acoustical Society of America* **111**(4), pp. 1819–1831. (Cited on page 38)

Veit, T., Cao, F., and Bouthemy, P. (**2007**), "Space-time A Contrario Clustering for Detecting Coherent Motions," in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 33 –39. (Cited on page 50)

Vermaak, J. and Blake, A. (**2001**), "Nonlinear filtering for speaker tracking in noisy and reverberant environments." in *ICASSP*, IEEE, pp. 3021–3024. (Cited on pages 41 and 42)

Viola, P. and Jones, M. (**2001**), "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. I–511–I–518 vol.1. (Cited on page 84)

Wallach, H. (**1940**), "The role of head movements and vestibular and visual cues in sound localization," *Journal of Experimental Psychology* **27**(4), pp. 339–368. (Cited on page 55)

Wang, D. L. (**2005**), "On ideal binary mask as the computational goal of auditory scene analysis," in *Speech Separation by Humans and Machines*, edited by P. Divenyi, Kluwer Academic, Dordrecht, The Netherlands, chap. 12, pp. 181–197. (Cited on page 31)

Wang, D. L. and Brown, G. J. (Eds.) (**2006**), *Computational Auditory Scene Analysis: Principles, Algorithms and Applications*, Wiley/IEEE Press. (Cited on pages 27 and 79)

Wang, D. L., Kjems, U., Pedersen, M. S., and Boldt, J. B. (**2009**), "Speech intelligibility in background noise with ideal binary time-frequency masking," *Journal of the Acoustical Society of America* **125**(4), pp. 2336–2347. (Cited on page 31)

Wang, Y., Han, K., and Wang, D. L. (**2013**), "Exploring monaural features for classification-based speech segregation," *IEEE Transactions on Audio, Speech, and Language Processing* **21**(2), pp. 270–279. (Cited on pages 31 and 32)

Wielemaker, J., Schrijvers, T., Triska, M., and Lager, T. (**2012**), "SWI-Prolog," *Theory and Practice of Logic Programming* **12**(1-2), pp. 67–96. (Cited on page 20)

Wierstorf, H., Geier, M., Raake, A., and Spors, S. (**2011**), "A free database of head-related impulse response measurements in the horizontal plane with multiple distances," in *Proc. 130th Conv. Audio Eng. Soc.* (Cited on pages 59 and 80)

Wightman, F. L. and Kistler, D. J. (**1999**), "Resolution of front–back ambiguity in spatial hearing by listener and source movement," *Journal of the Acoustical Society of America* **105**(5), pp. 2841–2853. (Cited on page 55)

WillowGarage (**2014**), "Open Source Computer Vision Library," URL `http://sourceforge.net/projects/opencvlibrary/`. (Cited on page 51)

Woodruff, J. and Wang, D. L. (**2012**), "Binaural localization of multiple sources in reverberant and noisy environments," *IEEE Transactions on Audio, Speech, and Language Processing* **20**(5), pp. 1503–1512. (Cited on page 56)

Zuriarrain, I., Mekonnen, A., F.Lerasle, and N.Arana (**2013**), "Tracking-by-detection of multiple persons by a resample-move particle filter," *Machine Vision and Applications, Vol.24, N°8, pp.1751-1765* . (Cited on page 52)