Biologically Inspired Binaural Sound Source Localization and Tracking for Mobile Robots

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der RWTH Aachen University zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Diplom-Informatiker

Laurent Calmes

aus Luxemburg

Berichter: Universitätsprofessor Gerhard Lakemeyer, Ph.D.
Universitätsprofessor Dr. rer. nat. Hermann Wagner

Tag der mündlichen Prüfung: 23.12.2009

Acknowledgments

I would like to thank my supervisors Gerhard Lakemeyer and Hermann Wagner for their support and patience and for giving me the opportunity to work on this fascinating project on the intersection of Biology, Computer Science and Robotics. I am also grateful for the freedom they gave me to develop my own ideas.

My warmest thanks go to Stefan Schiffer, with whom I had many passionate discussions and whose insights and contributions to MCMCDA were invaluable. His non-scientific input was equally as important as his scientific collaboration.

I would also like to thank Dominik Röttsches, Daniel Peger and Tobias Krämer, who made significant contributions to this project with their diploma theses.

I am also very grateful to all the friends and colleagues in the Knowledge-Based Systems Group and the Institute for Biology II who made life in academia easier and more enjoyable, through their scientific and non-scientific contributions, especially my office colleague, Frank Endler.

And finally, my warmest gratitude goes to Katrin, who kept me going by providing the right nudge at the right time.

Abstract

This thesis proposes, firstly, biologically inspired methods of binaural sound source localization for mobile robots. Secondly, we propose a method for modulating the robot's attention inspired from the barn owl and thirdly a tracking system which makes it possible for a robot to track objects emitting sounds.

Regarding sound source localization, the method that was best understood and evaluated is an algorithm based on the evaluation of interaural time differences (ITDs). There is a very simple reason for this state of affairs. Interaural time differences are influenced mainly by the inter-microphone distance, provided that there is no major obstruction (like an artificial head) between them. This would make the sound waves bend around the structure and thus increase the path length and consequentially ITD in a frequency-specific manner. As long as there is no obstruction between the microphones and the far-field assumption is satisfied, the interaural time difference directly relates to azimuth through a simple equation, where only the additional parameters of inter-microphone distance (constant) and speed of sound (can be regarded constant) are required. Under these conditions, it is straightforward to adapt ITD localization to different hardware platforms: it only requires mounting the microphones and providing the correct microphone baseline value to the software. The method we use for ITD based sound localization relies on detecting phase coincidence for individual frequencies in the frequency domain and subsequent frequency integration to eliminate phase ambiguities. Overall, the results of the system are excellent. Broadband signals could be localized with an accuracy of about $\pm 2^\circ$. The localization of pure tones was highly erratic, as was to be expected. The only unexpected behavior was the low accuracy in localizing 100 Hz — 1 kHz bandpass noise. By performing simulations in which the room acoustics could be controlled, we could show that this is caused by sound reflections from the environment. In larger rooms or, equivalently, rooms with a lower direct-to-reverberant ratio, localization precision of broadband signals also degrades significantly, which becomes evident in experiments on a real robot. All in all, care has to be taken as to the acoustic environment in which the ITD based sound source localization is to be deployed, in order to achieve best performance.

Interaural level differences based sound source localization by principle relies on the acoustical properties of the microphone mount assembly and supporting structures. This means that adapting ILD localization to a new platform is more difficult. It requires mounting the microphones and then calibrating the whole setup to record the resulting azimuth / elevation / frequency dependent ILD values, which can then be used by the ILD based sound source localization algorithm. This is a quite elaborate, time-consuming procedure which has to be repeated every time something changes in the way the microphones are mounted - or, indeed, if the microphones themselves are changed. Experiments with artificial owl ruffs illustrate this point: even small changes in the ruff can have a huge impact on the ILDs (and, to a lesser degree, on the ITDs). The method for ILD based sound source localization relies on a neuronal model of the barn owl's auditory intensity pathway. Specifically, the neuronal responses in the VLVp and the ICc ls as well as the connections between these areas are modeled. The results of the experiments with the algorithm are very encouraging. The first tests showed that the system was able to accurately localize broadband sound sources in the range of $-30^\circ \ldots +30^\circ$. The more elaborate artificial ruffs experiments confirmed these results. Furthermore, with the correct acoustic design of the artificial ruff, it is possible to use the ILDs for various purposes as for example localization in elevation and/or verification/correction of the ITD based azimuth estimates.

With the attentional module based on a neuronal saliency map it is possible to preactivate a robot's attention to a specific region of interest. With this method it was possible to successfully reproduce with a robotic pan-tilt unit attentional latency experiments that were performed with barn owls. But the system we propose can easily be generalized to modulate (in several instances) the attention of the robot at various levels, from the basic sensor level up to the planning level.

The Markov chain Monte Carlo based combined sound source and dynamic object tracking had a few problems accurately tracking our simulated entities. Although the general viability of the method could be shown, the algorithm still has several shortcomings. MCMCDA with a virtual sensor is able to correctly track sound sources and objects alone, but the combination of both modalities in one track proved to be difficult. As long as the individual entities are in clearly distinct positions, correct tracks are produced, but if they approach each other or - even worse - cross paths, tracking breaks down. This seems to be caused mainly by the lack of distance information in the sound source localization modality. As long as these shortcomings are not addressed, it makes little sense to test the method on a real robot. This is why the MCMCDA experiments in this thesis were limited to simulations.

Zusammenfassung

Diese Dissertation befasst sich mit, erstens, biologisch inspirierten Methoden der binauralen Schallquellenlokalisierung für mobile Roboter. Zweitens wird eine von der Schleiereule inspirierte Methode zur Modulierung der Aufmerksamkeit des Roboters vorgestellt und drittens ein System, das es dem Roboter ermöglicht, Schall erzeugende Objekte zu verfolgen.

Die Schalllokalisierungsmethode die am besten verstanden und untersucht wurde ist ein Algorithmus der auf der Auswertung von interauralen Zeitdifferenzen (interaural time differences; ITDs) beruht. Dafür gibt es einen sehr einfachen Grund. Interaurale Zeitdifferenzen werden hauptsächlich durch den Mikrophonabstand beeinflusst, vorausgesetzt, die Mikrophone stehen frei - es befindet sich also z.B. kein Kunstkopf zwischen ihnen. Durch eine solche Struktur würden die Schallwellen frequenzabhängig um den Kopf herum "gebogen" werden, was den Weg der Schallwelle von einem Mikro zum anderen verlängert und somit die ITD erhöht. Solange sich keine Struktur zwischen den Mikrophonen befindet und die Schallquelle sich im sog. Fernfeld befindet, wird die ITD vom Schallquellenazimut über eine einfache Gleichung bestimmt, die als einzige zusätzliche Parameter den Mikrophonabstand (konstant) und die Schallgeschwindigkeit (kann man als konstant annehmen) enthält. Unter diesen Bedingungen lässt sich die Schalllokalisierung sehr einfach an verschiedene Hardware-Plattformen anpassen: man braucht nur die beiden Mikrophone zu montieren und der Software den korrekten Abstand mitzuteilen. Die Methode, die wir für die ITD-basierte Schalllokalisierung benutzen detektiert Phasenkoinzidenz für individuelle Frequenzbänder im Frequenzbereich und eliminiert Phasenmehrdeutigkeiten durch anschließende Frequenzintegration. Die Ergebnisse, die mit dem System erzielt wurden sind ausgezeichnet. Breitbandige Signale konnten mit einer Präzision von $\pm 2^\circ$ lokalisiert werden. Die Lokalisierung von Sinustönen war erwartungsgemäß extrem unzuverlässig. Das einzig Unerwartete war die niedrige Lokalisierungspräzision für 100 Hz — 1 kHz Rauschen. Durch Simulationen der Raumakustik konnte ermittelt werden, dass dies auf von den Wänden herrührenden Echos zurückzuführen ist. In größeren Räumen (oder solchen mit einem schlechteren Verhältnis von Direktschall zu Echos) wird die Lokalisierung von Breitbandsignalen auch schlechter, was wir in Versuchen auf einem Roboter überprüfen konnten. Bei benutzen der ITD-basierten Schalllokalisierung muss also Rücksicht auf die Raumakustik genommen werden, um die bestmögliche Leistung zu erzielen.

Interaurale Pegeldifferenzen (interaural level differences; ILDs), werden prinzipbedingt stark durch den Mikrophonaufbau beeinflusst. Dadurch ist es schwieriger, ILD-basierte Schalllokalisierung auf neue Plattformen anzupassen. Dafür muss der gesamte Aufbau kalibriert werden, d.h. es müssen für jede mögliche Kombination von Azimut und Elevation die entsprechenden frequenzabhängigen ILD Werte gemessen werden, welche dann vom Schalllokalisierungsalgorithmus verwendet werden können. Dies ist ein langwieriger Prozess, der jedes Mal wiederholt werden muss, wenn sich am Mikrophonaufbau (oder an den Mikrophonen) was ändert. Versuche mit künstlichen Eulenschleiern illustrieren dies: kleine Änderungen am Schleier können große Änderungen der ILDs (und kleinere Änderungen der ITDs) hervorrufen. Die Methode, die wir für ILD-Lokalisierung benutzen, beruht auf einem neuronalen Modell des Intensitätspfades der Schleiereule. Es werden spezifisch die neuronalen Antworten des VLVp und des ICc ls und die Verbindungen zwischen diesen Arealen modelliert. Die Ergebnisse von Versuchen mit diesem Algorithmus sind ermutigend. Erste Experimente haben gezeigt dass das System breitbandige Signale mit hoher Präzision im Bereich von $-30^\circ \ldots +30^\circ$ lokalisieren kann. Die aufwendigeren Versuche mit den künstlichen Eulenschleiern konnten dies

bestätigen. Zudem ist es möglich – mit dem korrekten akustischen Entwurf des Schleiers – ILDs für diverse Anwendungen zu erzeugen. Z.B. kann man in Elevation lokalisieren und/oder ITD-basierte Azimut-Schätzungen verifizieren/korrigieren.

Das auf einer sog. “saliency map” basierte Aufmerksamkeitsmodul ist in der Lage, die Aufmerksamkeit eines Roboters auf einen bestimmten Bereich zu präaktivieren. Mit diesem System waren wir in der Lage (mit den Mikrophonen auf einer Schwenk-Neige-Einheit), einschlägige, mit Eulen durchgeführte Versuche zu durch Aufmerksamkeit bedingte Reaktionslatenz zu wiederholen. Das System das wir vorstellen kann sehr einfach generalisiert werden, um die Aufmerksamkeit des Roboters auf mehreren Ebenen (von der Sensor- bis hin zur Planungsebene) zu modulieren.

Die auf Markov chain Monte Carlo data association (MCMCDA) basierte Methode zur Verfolgung von Entitäten, bestehend aus der Kombination von Schallquellen und dynamischen Objekten, hatte einige Probleme, unsere simulierten Entitäten korrekt zu verfolgen. Zwar konnte gezeigt werden, dass die Methode im Wesentlichen funktioniert, dennoch hat sie noch einige Schwächen. MCMCDA mit einem virtuellen Sensor ist in der Lage, Schallquellen oder dynamische Objekte alleine zu verfolgen. Die Kombination beider Modalitäten erzeugt aber Probleme. So lange individuelle Entitäten klar voneinander getrennt waren, konnten sie korrekt verfolgt werden. Sobald sie sich allerdings annäherten (oder schlimmer, ihre Wege sich kreuzten), versagte die Verfolgung. Dies scheint durch die fehlende Entfernungsinformation der Schalllokalisierungsmodalität verursacht zu werden. Solange diese Unzulänglichkeiten nicht behoben werden, macht es wenig Sinn, die Methode auf einem realen Roboter zu testen. Deshalb wurden die Versuche mit MCMCDA in dieser Dissertation nur in Simulationen durchgeführt.

Contents

List of Figures

List of Tables

List of Abbreviations

A/D	analog-to-digital, page 5
ATC	adaptive tolerance control, page 76
CD	characteristic delay, page 16
CSP	crosspower spectrum phase (*another name for PHAT*), page 34
dB	decibel, page 13
DFT	discrete Fourier transform, page 7
EKF	extended Kalman filter, page 25
FFT	fast Fourier transform, page 11
GCC	generalized cross-correlation, page 34
gcd	greatest common divisor, page 28
HRTF	head-related transfer function, page 12
ICc core	core of the central nucleus of the inferior colliculus, page 20
ICc ls	lateral shell of the central nucleus of the inferior colliculus, page 20
ICX	external nucleus of the inferior colliculus, page 21
IID	interaural intensity difference (*equivalent to ILD*), page 13
ILD	interaural level difference, page 13
IPPF	independent partition particle filter, page 36
ITD	interaural time difference, page 13
JPDAF	joint probabilistic data association filter, page 36
MC-JPDAF	Monte Carlo joint probabilistic data association filter, page 36
MCL	Monte Carlo localization, page 25
MCMC	Markov chain Monte Carlo, page 27
MCMCDA	Markov chain Monte Carlo data association, page 80
MHT	multiple hypothesis tracking, page 36
NA	nucleus angularis, page 21

NL	nucleus laminaris, page 20
NM	nucleus magnocellularis, page 20
PHAT	phase transform (*another name for CSP*), page 34
PTU	pan-tilt unit, page 44
SLAM	simultaneous localization and mapping, page 22
SMC	sequential Monte Carlo, page 25
SNR	signal-to-noise ratio, page 54
SSPF	sequential sampling particle filter, page 36
UKF	unscented Kalman filter, page 25
VLVa	nucleus ventralis lemnisci lateralis, pars anterior, page 20
VLVp	nucleus ventralis lemnisci lateralis, pars posterior, page 21

Chapter 1

Introduction

Autonomous mobile robotics has come a long way since the first hesitant "steps" of the robot *Shakey* — the first to reason about its own actions — four decades ago (Nilsson, 1984). Where Shakey could only perform in a tightly controlled environment and had reaction times that precluded dynamic environments (due to the computing resources available in the late 60s, early 70s), its "descendants" fare much better today in more realistic settings. Indoor navigation in highly dynamic environments has been possible for some years now (e.g. the museum tour guide robot *Rhino* (Buhmann et al., 1995)). More recently, autonomous mobile robots in the shape of robotic, driverless all-terrain cars were able to cross hundreds of kilometers of desert environment without human intervention (DARPA Grand Challenge (Thrun et al., 2006)).

Nevertheless, there are still at least two key areas that have to be actively researched in order for autonomous mobile robots to be really usable as service robots, interacting with humans.

One is the ability of the robot to manipulate objects in the environment. This is not a problem of mechanical engineering, as actuators for that task — dexterous, powerful, small and light enough — are available. Rather, it is a matter of object recognition and controlling the actuator in a way that objects are handled carefully.

The other area where there is still room for a lot of improvement is the way in which the robot is controlled. The easiest way to do this — at least for the human controlling the robot (the *user*) — is via speech commands. Voice recognition and natural speech processing have reached a point where basic voice control of mobile robots is possible (Doostdar et al., 2008). One major drawback still persists: the microphone has to be close to the mouth of the person controlling the machine in order to provide the cleanest possible signal (with the highest *signal-to-noise ratio*) to the voice processing software. This problem can be solved relatively elegantly by using a wireless headset. But then what if the robot interacts with a great number of different people in the course of a day, as would be probable e.g. in a hospital environment? Providing every single person likely to interact with the agent with a wireless headset is highly impractical.

It would be most elegant if the microphone was on the robot and one could somehow find a way to deal with the noisy signals inevitable in that configuration. But how could this be achieved?

The localization of sound sources is a step in the direction of a solution to this problem. If a robot knows the location of a sound source, it can take proactive measures to enhance the signal from that source. It could for example approach the source, and thereby increase the signal-to-noise ratio. Another possibility is to use directional filtering.

In order to be able to localize sound sources, the robot will need more than one

microphone. The more microphones, the more precise the localization estimate will be. But using a high number of microphones has several drawbacks:

1. Spatial constraints on an autonomous mobile robot might make it difficult to find room for many microphones with suitable inter-microphone distances.

2. Specialized, multichannel audio hardware would be needed to digitize the microphone signals. This is expensive and would need additional space, either inside the onboard computer (add-in card) or externally (external audio hardware).

3. The processing of many audio signals takes up computing resources which could be allocated to other tasks.

The goal for sound source localization on mobile robots therefore should be to use a minimal number of microphones. We chose to take our inspiration from nature. Evolution has had millions of years to come up with solutions to various problems, amongst which there is sound source localization.

Most animals have two ears. It is not known if this number emerged through evolutionary pressure and if it is in fact an optimal[1] number for sound localization. It could also be caused by embryological constraints related to a bilaterally symmetric body plan (Schnupp and Carr, 2009). Nevertheless, at least two ears are needed for a useful precision. With one acoustic sensor, only a crude directional estimate is possible (Sections 2.3.1 & 2.3.2 will address the issue of sound localization with one ear vs. two ears).

Using only two microphones on mobile robots avoids the above-mentioned drawbacks associated with large microphone arrays:

1. Two microphones do not take up a lot of space and can be fit nearly anywhere on a robot.

2. The signals can be digitized using standard off-the-shelf hardware, because they have stereo, i.e. two-channel inputs. Furthermore, most modern computer motherboards feature integrated audio hardware, so no additional space is required and no additional costs are incurred.

3. The computer only needs to process two audio signals.

For the actual sound source localization methods, we turned to a specialist in the field, the barn owl (*Tyto alba*), whose auditory system is also one of the best understood and investigated.

Localization of a sound source might be enough to enhance the signal from that source, but in order to effectively process voice commands from users, a robot must be able to follow them over time, i.e. to track them. This means that the robot has to know that the person (the sound source) it detects now at a specific position x, is the same that was at a different position y a few seconds earlier.

As our localization system cannot differentiate between different types of sound sources or speakers (this would require some sort of sound or speaker recognition), we chose a different approach to keep track of sound sources.

We take advantage of other sensor modalities available on the robot, namely laser-based dynamic (i.e. moving) object recognition (Section 2.4.6). The idea is that a dynamic object emitting sound is with high probability human (and therefore interesting

[1] optimal in the sense of best localization accuracy with a minimum number of neurons required for processing

for the robot) and should be tracked. And because humans do not permanently emit sound (i.e. speak), the tracking system should also be able to track only objects. Conversely, as laser-based dynamic object recognition is by no means perfect, the system should also be able to track sound sources alone. The requirements for the tracking module should therefore be the following:

1. The tracking system should be able to track objects emitting sound.
2. It should still continue to track an entity if it temporarily stops emitting sound or an object is temporarily not detected.
3. In the limit cases of there being no object (or no sound), the system should degenerate to a sound-only (or object-only) tracker.

1.1 Goals and Contributions of this Thesis

One goal of this thesis is to provide a mobile robot with binaural hearing, which enables it to estimate the azimuth to a sound source. This will be achieved by combining interaural time and level difference approaches to sound source localization.

The other goal of the thesis is the combination of binaural hearing and laser-based dynamic object recognition into a system for tracking sound-emitting objects.

In order to achieve these goals, the following contributions are made in this thesis:

Interaural time differences based sound source localization The interaural time differences (ITD) based sound source localization module is a modification of the dual delay-line algorithm by Liu et al. (2000). The interaural time difference (and thus the azimuth) to a sound source is found by computing phase differences in individual frequency bands. Phase ambiguities are eliminated through across-frequency integration.

Interaural level differences based sound source localization The interaural level differences (ILD) based sound source localization is an implementation of a model by Spence and Pearson (1989). This itself is an attempt at a model of the ILD-processing pathway in the barn owl's auditory system.

Artificial owl ruff The function of the barn owl's ruff is to induce significant interaural level differences in elevation. Additionally, it extends the interaural time differences range beyond $\pm 90°$, which might help in solving front/back confusions. We developed two types of artificial owl ruffs. Common to both types was the idea that they should induce significant, easy to detect interaural level differences. One type was used to generate ILDs in the horizontal plane, in order to complement azimuthal information available through interaural time differences. The other type generated ILDs in the vertical plane, in order to be able to extract elevation from binaural cues.

Attentional module As a bridge between the biologically inspired sensor-level work on sound source localization and the statistics-based control-oriented work on the tracking module, we propose a biologically inspired system for modulating attention. It uses a neural network based saliency map and a visual attention signal to preactivate the saliency map on the side of the visual cue. In this way, once the acoustic stimulus is received, the reaction latency will be lower if it is on the same side as the visual cue. This system can be generalized to other than visual

attention signals and acoustic stimuli and could be applied to generally modulate the robot's attention at various levels.

Tracking module with a virtual sensor We propose a tracking method based on the Markov chain Monte Carlo data association (MCMCDA) algorithm (Oh et al., 2004). The sensor fusion (i.e. combination of sound source localization and laser rangefinder data) is performed by a frontend to MCMCDA called *virtual sensor*. This way, only one kind of data has to be processed by MCMCDA, which greatly simplifies the algorithm.

All these individual components were tested in experiments with encouraging results. The various methods for sound source localization as well as the artificial owl ruffs behaved as expected and could localize sound sources with good precision as long as the stimuli were broadband and the acoustic environment was not too echoic.

The attentional module could reproduce with high fidelity the behavior of barn owls in attentional experiments described by Johnen et al. (2001).

The tracking module was only tested in simulations. We chose to forego experiments on a real robot as the simulation results hinted at some problems that will have to be addressed before the method is ready to be tested on real hardware.

1.2 Outline

The rest of the thesis is organized as follows:

Chapter 2 reviews the theoretical background, ranging from digital signal processing over binaural hearing basics and its implementation in the barn owl to the application of statistical filtering methods to self-localization and object tracking on mobile robots.

Chapter 3 overviews the related work on binaural hearing on mobile robots as well as tracking and data association algorithms.

Chapter 4 presents the methods related to binaural hearing, which were developed for this project. Specifically, the interaural time differences localizer, the interaural level differences localizer, the artificial barn owl ruffs and the multimodal attention module.

Chapter 5 relates the first experiments with sound localization on a mobile robot in the first part. The second part describes the Markov chain Monte Carlo data association algorithm (MCMCDA) for tracking entities based on sound source localization and laser rangefinder data.

Chapter 6 concludes the thesis with a summary and an outlook for further research.

The results presented in Section 4.1 and Section 5.1 were previously published in Calmes et al. (2007a) and Calmes et al. (2007b), respectively.

Chapter 2

Background Material

In this chapter, we will present background material necessary to the understanding of the methods and algorithms described in later chapters. We will start with analog-to-digital conversion (Section 2.1) and explain spectral analysis using the discrete Fourier transform (Section 2.2), which form the basis of many signal processing applications.

We will then present the theoretical basis of binaural sound source localization and describe its implementation in the barn owl (*Tyto alba*) which, as a nocturnal predator, is a specialist in sound source localization (Section 2.3).

The final part of this chapter will be concerned with the description of algorithms used in mobile robotics pertaining to this thesis as well as the Markov chain Monte Carlo method (Section 2.5), underlying the sound source and dynamic object tracking system for mobile robots.

2.1 A/D Conversion

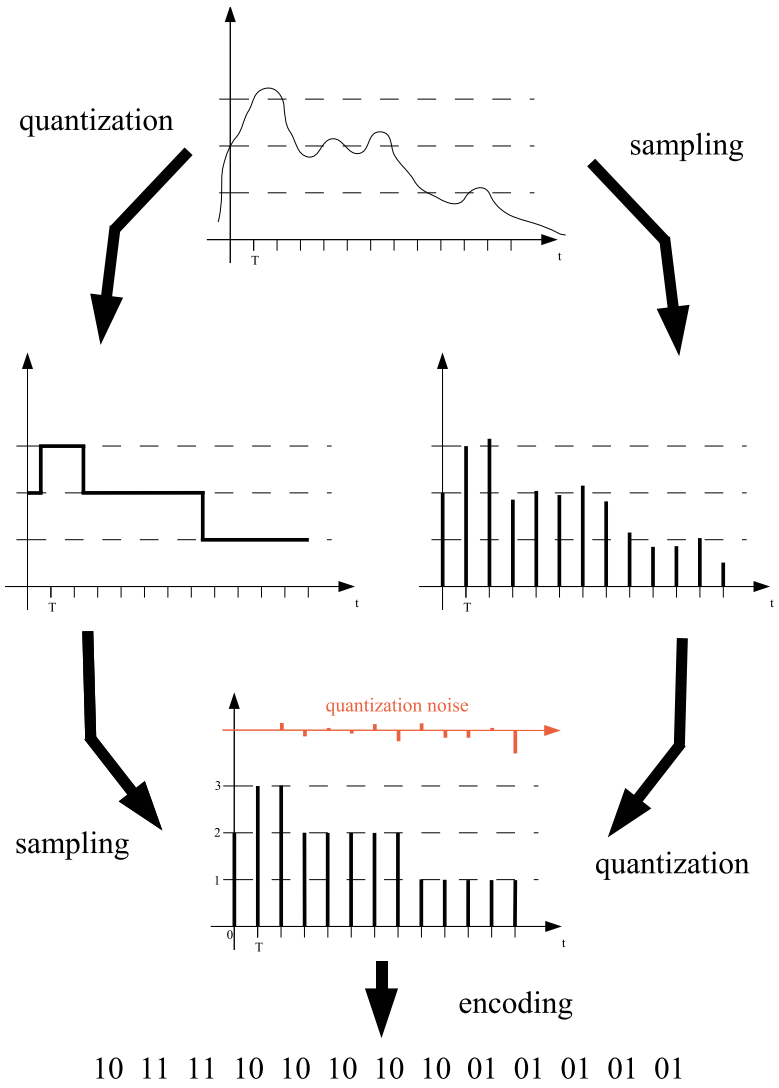
In order to perform sound source localization on a technical system, the signals from two microphones (in our case) have to be evaluated. These consist of an electrical current with a time-varying amplitude that reflects the movement of the microphone membrane. The microphone membrane in turn vibrates in response to local air pressure variations, which correspond to sound.

For a computer to be able to process these time- as well as amplitude-continuous electrical signals, each microphone signal first needs to be digitized, i.e. they need to be rendered time- and amplitude-discrete. This provides the computer with two streams of numbers, one for the left and one for the right microphones.

The digitization process — also called analog-to-digital conversion or A/D conversion — is essentially a two-step process. One step discretizes time, whereas the other step discretizes amplitude. Figure 2.1 illustrates a 2 bit A/D conversion. Note that the actual order of the two steps (sampling first / quantization first) has no influence on the result. In a technical implementation though, it might be advantageous to choose one sequence over the other.

2.1.1 Sampling

Sampling is the step by which the time axis is rendered discrete. This is done by holding the signal value constant at regular intervals. The length of such an interval is called sampling interval. The reciprocal is correspondingly called sampling frequency or sampling rate.



(after Lüke (1999))

Figure 2.1: Conversion of an analog time signal into a 2 bit time- and value-discrete digital signal. The sampling step renders the time axis discrete, whereas the quantization step results in discretized amplitudes. Two paths are shown, as the order in which these steps are performed is irrelevant.

If, according to the Nyquist-Shannon theorem (Shannon, 1949), the sampling frequency satisfies the following criterion:

$$f_s \geq 2f_l, \tag{2.1}$$

where f_l is the maximal frequency contained in the signal and f_s is the sampling frequency, the process can be reversed without loss of information. This means that the original signal can be recovered completely from the sampled values. The limit frequency f_l is called Nyquist frequency.

If Equation (2.1) is not satisfied, frequencies above f_l will get mirrored onto frequencies below f_l, which is called *aliasing*. If this happens, there is no way to remove the aliased frequencies from the sampled signal, which is why the f_l limit is usually enforced by analog filters before sampling.

Audio compact discs use sampling rates of 44.1 kHz. Modern high definition audio systems can work with sampling frequencies as high as 192 kHz, thus enabling restitution

of signals with frequency components going up to 96 kHz.

2.1.2 Quantization

During quantization, the amplitude axis is rendered discrete. This is done simply by rounding to the nearest value representable by the computer. In contrast to sampling, this rounding process introduces errors, meaning that it is impossible to fully recover the original signal from its quantized version. The difference between the real and rounded values can be considered as noise and is therefore called quantization noise (see Figure 2.1).

Although quantization noise cannot be completely eliminated, it can be minimized by increasing the bit resolution. This in effect increases the range of numbers available to quantization. Thereby the rounding error — and thus quantization noise — is reduced.

Whereas audio CDs use a 16 bit resolution ($2^{16} = 65\,536$ different amplitude values), high definition audio systems nowadays use a 24 bit resolution ($2^{24} = 16\,777\,216$ amplitude values), sometimes even 32 bit.

2.2 The Discrete Fourier Transform (DFT)

Up until now, the digitized signals are in the time domain. This means that the stream of numbers represent amplitude varying over time.

Another — equivalent — representation of signals is the frequency domain. In the frequency domain, the spectral (frequency) content of a signal becomes apparent. Whereas the time domain representation provides no information on frequency content, the frequency domain representation does not provide any information on the temporal structure of a signal.

Nevertheless, the two domains are equivalent in the sense that one representation can be transformed into another without loss of information.

Spectral analysis (i.e. representation of signals in the frequency domain) is a more "natural" and intuitive representation, as the human auditory system separates the ear input signals into their frequency components (with the help of the basilar membrane in the cochlea) before any further processing takes place.

The frequency domain representation is based on the concept that every signal can be regarded as a sum of sinusoidal oscillations of varying frequencies:

$$y(t) = A\sin(\omega t + \theta), \quad \omega = 2\pi f, \tag{2.2}$$

or equivalently:

$$y(t) = a\cos\omega t + b\sin\omega t \tag{2.3}$$

with $A = \sqrt{a^2 + b^2}$ and $\tan\theta = a/b$.

These sinusoidal oscillations can be uniquely characterized by only three parameters:

Amplitude A, specifying the maximal absolute deviation from 0.

Frequency f (in Hz), related to the angular frequency ω by $\omega = 2\pi f$ (in rad/s).

Phase angle θ (in rad), which specifies a shift of the oscillation on the time axis. The corresponding shift in seconds is frequency dependent: $\Delta t = \theta/\omega$.

The mathematical tool which performs the transformation from time domain to frequency domain is the Fourier transform:

$$F(f) = \int_{-\infty}^{+\infty} f(t)e^{-j2\pi ft}dt. \tag{2.4}$$

Transforming back from the frequency to the time domain can be done with the inverse Fourier transform:

$$f(t) = \int_{-\infty}^{+\infty} F(f)e^{j2\pi ft}df. \tag{2.5}$$

We should mention that, in Equations (2.4) & (2.5) and hereafter, j is used to denote the *imaginary unit* ($j^2 = -1$, by definition).

The relationship between Equations (2.4), (2.5) and the alternative definition of a sinusoidal oscillation (Equation (2.3)) can be intuitively grasped through Euler's formula:

$$Ae^{jx} = A(\cos x + j \sin x), \tag{2.6}$$

meaning that complex exponentials of the form Ae^{jx} are the complex versions of sinusoidal oscillations.

Note that the defining integrals of the continuous Fourier transform (Eq. (2.4)) and its inverse (Eq. (2.5)) are unbounded. This is of course intractable for practical applications. Furthermore, computers process data as discrete numbers, which disqualifies the continuous Fourier transform for practical signal processing.

In digital signal processing applications therefore, the discrete Fourier transform (DFT) is used:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-j2\pi \frac{k}{N} n}, \quad k = 0, \ldots, N-1, \tag{2.7}$$

with the corresponding inverse (IDFT):

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{j2\pi \frac{n}{N} k}, \quad n = 0, \ldots, N-1, \tag{2.8}$$

where x_n is the n-th sample of the time domain signal, X_k is the k-th frequency domain sample and N is the size of the transform, i.e. the time domain as well as the frequency domain signals contain N samples.

From now on, we will only write about the discrete Fourier transform, even if the word *discrete* is not explicitly stated.

The actual frequency f_k (in Hz) corresponding to a given Fourier index k can be obtained with the help of the sampling frequency f_s and the Fourier size N:

$$f_k = k\frac{f_s}{N}, \quad k = 0, \ldots, N/2. \tag{2.9}$$

and the bandwidth represented by each Fourier point is simply:

$$f_{bw} = f_s/N. \tag{2.10}$$

Note that Equation (2.9) only holds for $k = 0, \ldots, N/2$, as we know that, according to the sampling theorem, the maximal frequency present in the signal is $f_s/2$ (see Equation (2.1) in Section 2.1.1 and Shannon (1949)).

The output of the discrete Fourier transform is a sequence of length N of complex numbers $z_k = A_k e^{j\varphi_k}$ $(k = 0, \ldots, N-1)$. The index k (i.e. the position in the sequence) specifies frequency according to Equation (2.9). The absolute value of z_k specifies the amplitude of that frequency component (i.e. how much this frequency component contributes to the whole signal): $A_k = |z_k|$. The phase angle of the sinusoidal oscillation at frequency f_k is obtained through the argument of z_k: $\varphi_k = \arg(z_k)$. Thus, the three parameters that characterize a sinusoidal oscillation (amplitude A_k, phase φ_k and frequency f_k) are contained in the output of the DFT.

2.2.1 Properties of the DFT

The DFT has several remarkable properties. We will only discuss those relevant to this dissertation. In the description of these properties, we will use $\Longleftrightarrow$ to denote Fourier pairs, i.e. the left-hand side will represent the time domain and the right-hand side of $\Longleftrightarrow$ will denote the frequency-domain equivalent. Furthermore, time-domain signals will use lowercase, whereas frequency-domain signals will use uppercase letters. E.g.:

$$x_n \Longleftrightarrow X_k$$

is a Fourier pair.

Linearity

The discrete Fourier transformation is a linear operation:

$$ax_n + by_n \Longleftrightarrow aX_k + bY_k \tag{2.11}$$

Duality

The similarity of Equations (2.7) & (2.8) results in a symmetry of the Fourier transform called *duality*. This means that, e.g. the DFT of a sinusoid is a point in the frequency domain. Conversely, a sinusoid in the frequency domain corresponds to a point in the time domain.

Periodicity

A question that arises from Equation (2.9) is why it only holds for $N/2 + 1$ frequencies, when the DFT actually returns N complex values? That is so because the DFT is *periodic*. Its input is actually only one period of an unbounded periodic function. The complex array returned by the DFT is also an unbounded periodic function. Most DFT implementations, by convention, return the complex array with the upper half representing the lower half of the current period, in which case these values correspond to negative frequencies.

Symmetry for real-valued signals

One property of the DFT stipulates that in the case of real-valued input signals, the magnitudes of the result array are an even function whereas the phases are an odd function of frequency. Because of this redundancy, one can discard the negative frequencies, which results in some data reduction. Care must be taken though, when dealing with data that is to be transformed back to the time domain. In that case, the missing negative frequencies have to be added back, by copying and mirroring of the positive frequencies.

Shift theorem

Another important property — especially when dealing with computation of interaural time differences — is the *shift theorem*:

$$x_{n-m} \Longleftrightarrow X_k e^{-j2\pi mk/M}. \tag{2.12}$$

As X_k is a complex number of the form $X_k = A_k e^{j\varphi_k}$, multiplication with $e^{-j2\pi mk/M}$ adds a factor of $-2\pi mk/M$ to the phase component of X_k. Thus, the shift theorem stipulates that a shift of m samples in the time domain corresponds to a phase adjustment of $2\pi mk/M$ in the frequency domain.

A very useful consequence of the shift theorem is implicit sub-sample accuracy. If shifting were to be done in the time domain alone, the shift amount could only be a multiple of one sample (without resorting to explicit interpolation). Therefore, a very high sampling frequency (meaning a higher amount of data) would be useful in that context. In the frequency domain, on the other hand, by directly specifying phase, one can achieve an arbitrary shift amount, resulting in implicit interpolation in the time domain (this can be verified by transforming back into the time domain). Thus, the sampling frequency can be chosen to be not higher as actually required by the bandwidth of the input signal.

Zero padding theorem

When performing zero padding in the time domain (i.e. appending or prepending zeros to the input signal), the spectral content of the resulting zero padded signal is the same as before, but the length increases. After transformation to the frequency domain, more spectral values will be available, resulting in an increased spectral resolution. This means that zero padding in the time domain results in ideal interpolation in the frequency domain.

Uncertainty principle

The last property of the discrete Fourier transform we would like to address is the underlying *uncertainty principle*. In analogy to the famous Heisenberg uncertainty principle of quantum physics (position and velocity of a particle cannot be measured together, measuring one parameter renders the other uncertain), in Fourier analysis, time and frequency cannot be measured together with high precision. Increasing the precision of one parameter reduces that of the other parameter.

An increase of frequency resolution is achieved by increasing the number of frequency samples N. But as N is also the number of signal samples, the signal segment which has to be recorded in order to perform the N-point DFT increases in length. This means that the frequency information obtained spreads out over a longer time, thus the time resolution is reduced. Conversely, if the size of the input signal segment is reduced, one has more DFTs during a given time, which means a higher time resolution. But as the number of DFT points is also reduced, this results in a lower frequency resolution.

We will not go into further detail concerning the theory, properties and applications of the discrete Fourier transform, as that is beyond the scope of this dissertation. A good in-depth introduction can be found in Smith (2007).

2.2.2 The Fast Fourier Transform

As the direct implementation of Equations (2.7) & (2.8) has time complexity $O(N^2)$ (i.e. the running time is proportional to the square of the input size), this becomes intractable for larger N.

The most popular and well-known fast Fourier transform algorithm (FFT) is the one from Cooley and Tukey (Cooley and Tukey, 1965). It appears that the algorithm was already known to Carl Friedrich Gauss around 1805 (Heidemann et al., 1984)[1]. It was forgotten and rediscovered multiple times in the one and a half century following Gauss' work. Cooley and Tukey were the first to describe it with computational complexity and applications on electronic computers in mind.

Based on a divide and conquer strategy, the algorithm reduces the time complexity to $O(N \log_2 N)$ by recursively re-expressing a DFT of composite size $N = N_1 \cdot N_2$ in terms of DFTs of sizes N_1 and N_2.

In one of the most-used variants of the Cooley-Tukey algorithm (called radix-2), $N = 2^p$, $N_1 = 2$ and $N_2 = 2^{n-1} = N/2$. The input is halved at each step, hence the requirement for N to be a power of two ($N = 2^p$, p positive integer).

Nowadays, every algorithm able to perform the DFT with a time complexity of $O(N \log_2 N)$ is called a fast Fourier transform. Although most of them rely on some sort of factorization of N, FFT algorithms with $O(N \log_2 N)$ complexity exist even for prime N (Duhamel and Vetterli, 1990).

We will now leave the subject of digital signal processing. The concepts explained will be useful for the understanding of Chapter 4 (*Biologically inspired sound source localization*).

We will now turn to the fundamentals of sound source localization, which will also be very helpful for the understanding of Chapter 4.

2.3 Sound Source Localization

In sound source localization as it is considered here, two acoustic sensors (ears in biological systems; microphones in technical systems) are used to obtain directional information on sound sources. The directional information sought after consists of two parameters: azimuth (the horizontal angle to a sound source) and elevation (the vertical angle to a sound source). We do not try to estimate the distance to a sound source, as distance estimation based on acoustical cues alone has to rely heavily on contextual information on the sound source type, which is beyond the scope of this dissertation.

Figure 2.2 shows the coordinate system used for sound localization. The azimuth α has a negative sign on the left of the midline and a positive sign to the right. The elevation δ takes positive values for the up direction, and negative values for the down direction. The plane defined by the up/down and front/back axes is called *median plane*. The plane defined by the up/down and left/right axes is called the *frontal plane* and the plane defined by the front/back and left/right axes is the *horizontal plane*.

The cues available in the input signals, which encode direction, fall into two major classes: monaural and binaural cues, both of which will be described in the following sections.

[1] if the year is correct, this even predates Fourier's work on harmonic analysis from 1807!

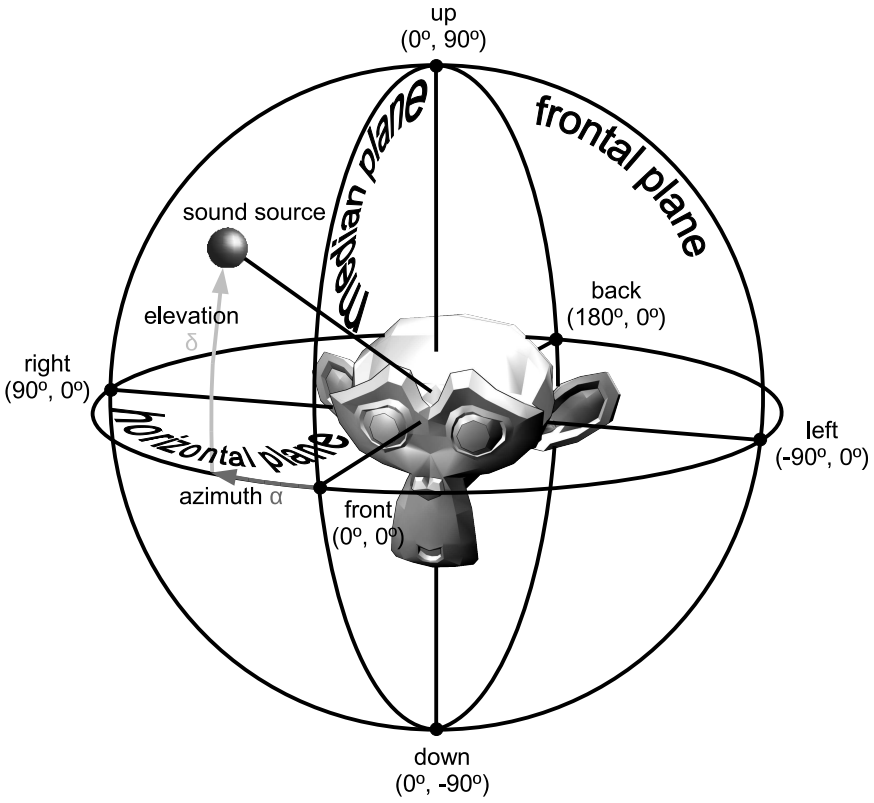


Figure 2.2: Coordinate system used for sound source localization.

2.3.1 Monaural Cues

Monaural cues can be extracted from the input signal of only one acoustic receiver. These so-called head-related transfer functions (HRTFs) are direction-dependent frequency filters which arise through the physical setup of the sensor.

This means that, depending on the angle of incidence relative to the sensors (in azimuth as well as in elevation) of the acoustic wave arriving from the sound source, characteristic frequency bands are amplified or attenuated. In humans and animals, the brain learns to interpret these specific frequency patterns as directional information.

In humans, the HRTFs are mainly caused by the pinna, the head and the shoulders (Blauert, 1997). In the barn owl, the HRTFs arise in a major part through the facial ruff feathers (von Campenhausen and Wagner, 2006).

These HRTFs are highly specific to the physical layout of the receiver assembly, which is why a localization system based on head-related transfer functions needs to be calibrated to the respective sensor setup.

Although sound source localization solely based on HRTFs (i.e. monaural sound localization) is possible, the accuracy is greatly reduced in humans (Blauert, 1997), as well as in animals like the barn owl (Knudsen and Konishi, 1979).

Nevertheless, the directional information available through HRTFs is important in helping to resolve ambiguities which might remain if binaural cues were exclusively used for localization.

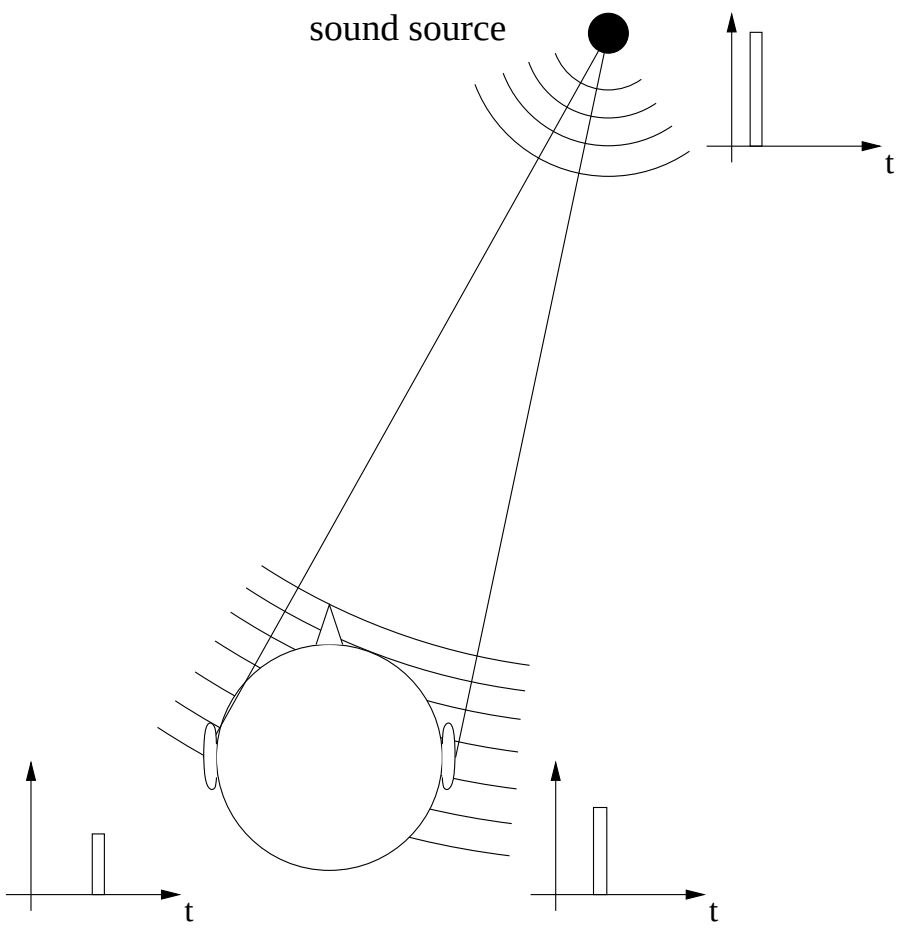


Figure 2.3: Interaural time and level differences. Interaural time differences arise through the different distances a sound wave has to travel in order to reach both ears. Interaural level differences are differences in loudness caused by the acoustic shadow of the head.

2.3.2 Binaural Cues

Binaural cues arise through the comparison of the signals from both receivers. There are two kinds of binaural cues: interaural level differences (ILDs), sometimes also called interaural intensity differences (IIDs) and interaural time differences (ITDs). Both are schematically depicted in Figure 2.3.

Interaural Level Differences

Interaural level differences are caused by the acoustic "shadow" of the head (or microphone assembly). These are strongly frequency dependent, as higher frequency sound waves have a shorter wavelength and thus are attenuated by structures which would be "transparent" (because they are too small) to longer wavelength signals. If, for example, a sound source is located to the right of the head, the sound wave will reach the right ear nearly unchanged, whereas it will reach the left ear with a frequency-dependent attenuation (see Figure 2.3). Although ILDs are called level *differences*, they are actually amplitude ratios. But as they are expressed in decibels[2] (dB) they are differences of

[2]The decibel (dB) is a logarithmic unit expressing the magnitude of a quantity relative to a reference. It is defined in terms of a power ratio, i.e. $L_{\mathrm{dB}} = 10 \log_{10} \left(\frac{P_1}{P_0} \right)$, P_0 being the reference level (Martin,

amplitude ratio *logarithms*:

$$\mathrm{ILD} = 20 \log_{10} \left(\frac{\mathrm{Amp_R}}{\mathrm{Amp_L}} \right), \tag{2.13}$$

where $\mathrm{Amp_L}$ and $\mathrm{Amp_R}$ are the left and right signal amplitudes, respectively.

Interaural Time Differences

Due to the finite speed of propagation of acoustic waves in air (assumed to be 340 m/s hereafter), the sound wave from a sound source will reach both receivers at slightly different moments in time (assuming the sound source is not located in the median plane). For example, with a sound source located on the right from the median plane, the sound waves will first reach the right ear and after a short delay, the signal from the sound source will reach the left ear (see Figure 2.3). These interaural time differences (ITDs) reach a maximum of around 660 µs at an azimuth of roughly ±90° in humans (Blauert, 1997). In the barn owl, the maximal ITD of approximately 270 µs is reached at a position of about ±110° (von Campenhausen and Wagner, 2006).

Interaural time differences are easier to exploit in a technical implementation than interaural level differences. Under certain assumptions, the ITD can be related to azimuth, speed of sound and microphone distance by

$$\Delta t = \frac{b}{c} \sin \alpha, \tag{2.14}$$

where Δt is the interaural time difference (in s), b the distance between the microphones (in m), c the speed of sound (in m/s) and α is the sound source azimuth. Equation (2.14) applies under the following conditions:

1. There is no obstruction between the microphones (like an artificial head) which could alter ITDs by requiring sound waves of higher frequencies to "bend" around the obstacle in order to reach the second microphone.

2. The microphones have an *omnidirectional* receiving characteristic, i.e. their sensitivity is not direction-dependent.

3. We assume that the acoustic wavefront arriving at the microphones is planar ("far-field assumption"). This means that the distance from the microphones to the sound source is large enough for the curvature of the wavefront to be negligible between the microphones. In reality, the wavefront will conform to the surface of a sphere if it is assumed that the sound emanates from a point source.

Under the preceding assumptions, Equation (2.14) holds and the maximal ITD with $\mathrm{ITD_{max}} = b/c$ is reached for $\sin \alpha = \pm 1$, which means that the source is positioned at the most lateral azimuth of $\alpha = \pm 90°$.

Equation 2.14 will be derived in Appendix A. Furthermore, the error to the correct (near-field) case will be calculated.

1929). If dealing with amplitude ratios (as in Eq. 2.13), the amplitudes have to be squared (power is proportional to amplitude squared), i.e. $L_{\mathrm{dB}} = 10 \log_{10} \left(\frac{A_1^2}{A_0^2} \right) = 20 \log_{10} \left(\frac{A_1}{A_0} \right)$, A_0 being the reference amplitude. The decibel corresponds to one tenth of a bel (B), hence the name.

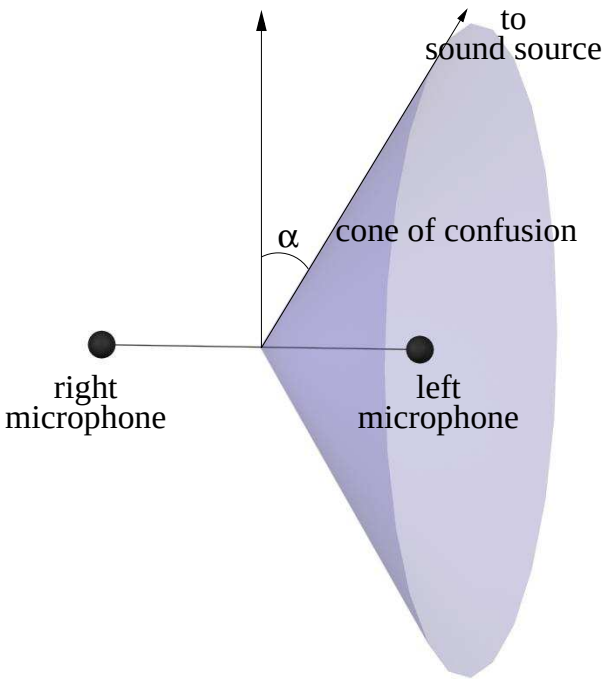


Figure 2.4: Cone of confusion.

Cone of Confusion

Because interaural time and level differences vary mainly with the azimuth of a sound source in humans, the elevation to the source does not have a major influence on these cues. This results in the so-called cone of confusion, depicted in Figure 2.4. The same combination of ITD and ILD will be generated by a sound source positioned at any location on the surface of a cone. This makes it impossible for humans to determine the elevation and the front-back position of a sound source using binaural cues alone. To estimate elevation and front-back position, the human auditory system has to take into account the head-related transfer functions.

The situation is altogether different in barn owls. Because the facial ruff extends the maximal ITD beyond $\pm 90°$ (particularly the reflecting feathers along the ruff's rim), the cone of confusion for time differences is asymmetric and not centered on the $\pm 90°$-axis between the ears (von Campenhausen and Wagner, 2006). Furthermore, as the ILDs change mainly with elevation, the owl can get a valid position estimate for a sound source (azimuth and elevation) using binaural cues alone. The evaluation of HRTFs can help the owl in resolving any remaining front-back confusions.

2.3.3 The Jeffress Model

One of the most successful models for explaining the processing of interaural time differences in the brain is the Jeffress model (Jeffress, 1948). It is characterized by two fundamental features (see also Figure 2.5):

Axonal delay lines induce internal delays, thus compensating for the external ITDs. Due to the finite signal transduction speed of neural fibers, varying axon lengths cause different delays in the transmitted action potentials.

Coincidence detector neurons fire maximally if excited simultaneously from the ipsi- and contralateral sides. The sum of the delays of the ipsi- and contralateral axonal

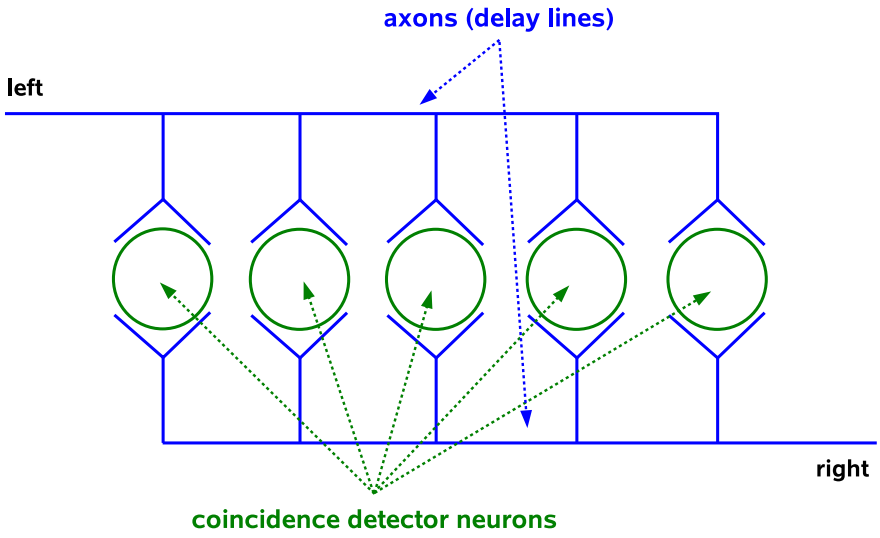


Figure 2.5: The Jeffress model. Coincidence detector neurons fire maximally if excited simultaneously from both sides. The axonal delay lines compensate for the external interaural time differences.

delay lines connected to the neuron constitute the coincidence detector's characteristic delay (CD) .

If a sound source is positioned e.g. to the left of the head, the spike train from the left ear will enter the structure first. It will travel further along the delay line, before the action potentials from the right ear arrive. Both spike trains will meet at a coincidence detector neuron at the right of the structure, which will then fire maximally.

Mathematically speaking, the most simple description of the Jeffress model is a cross-correlation between the input signals:

$$\psi_{l,r}(\tau, t) = \int_{-\infty}^{t} r(\vartheta) l(\vartheta - \tau) W(t - \vartheta) d\vartheta. \tag{2.15}$$

Where l and r are the input signals from the left and right ear, respectively. τ is the lag (delay) and $W(s)$ is a weighting function defined as follows:

$$W(s) = e^{-s/T_{int}}. \tag{2.16}$$

$W(s)$ implements a forgetting average, giving maximal weight to the most recent samples and increasingly less weight to samples in the past. How fast the weighting decays is controlled by the parameter T_{int}.

The lag τ, at which the correlation function $\psi_{l,r}(\tau, t)$ exhibits a maximum, corresponds to the interaural time difference present in the input signals.

After reviewing the basic theory of sound localization, we will now describe how this is realized in nature, using the barn owl as an example.

2.3.4 Inspiration: the Barn Owl (*Tyto alba*)

The barn owl (*Tyto alba*) is a nocturnal predator, feeding primarily on small vertebrates, particularly rodents (see Figure 2.6). It is able to hunt in complete darkness and find food solely by localizing the sound its prey makes (Payne, 1971). As such it is a specialist for

Stefan Johnen, 1998

Figure 2.6: The barn owl (*Tyto alba*).

passive sound source localization — in contrast to active sound localization (i.e. "sonar") as e.g. bats perform it. Consequently, the barn owl's auditory system has been studied for decades and is quite well understood. The in-depth knowledge as well as the high specialization of the barn owl's auditory system makes it an interesting model system for technical implementations.

The barn owl's specialization starts at the ruff — recognizable as the white, heart shaped "face" with the brown contour in Figure 2.6. Figure 2.7 shows the effect of the ruff on the binaural cues. With the ruff intact, the maximal ITD is around $\pm 110°$, whereas the ILD shows a strong gradient with elevation (Figure 2.7 **a**, **d**). With all feathers removed, maximal ITD moves to $\pm 90°$ and the strong elevational gradient of ILD all but disappeared (Figure 2.7 **c**, **f**). Note that the removal of the reflector feathers (Figure 2.7 **b**, **e**) has the greatest effect on the head-related transfer functions.

The processing of binaural cues happens in two parallel pathways in the barn owl's brain: one for time and one for level differences (Figure 2.8).

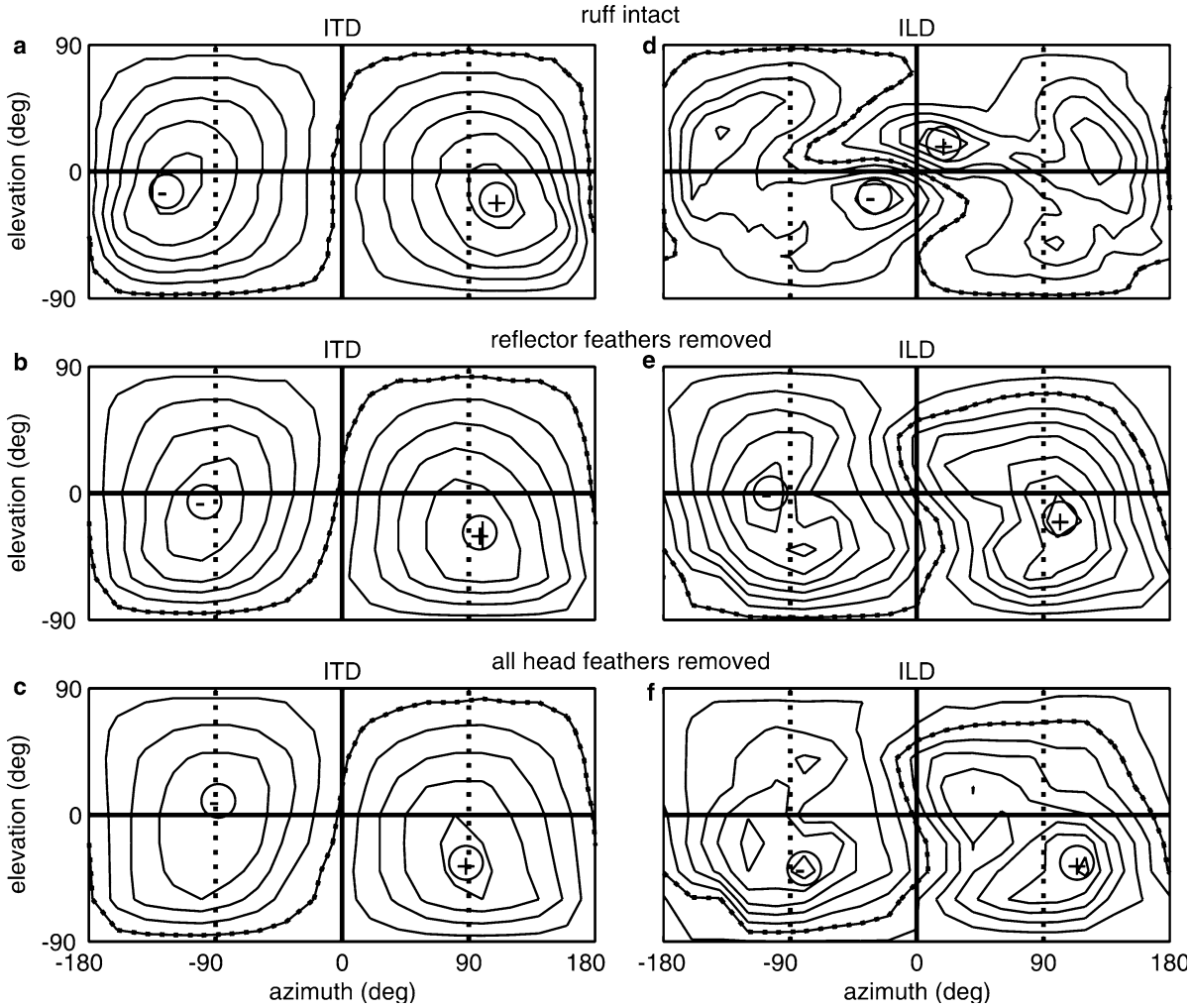


Figure 2.7: Influence of the facial ruff feathers on binaural cues. Contour lines show regions of same ITD (**a**-**c**) and same ILD (**d**-**f**). 0 ITD and 0 ILD are indicated by dotted contour lines. ITD contour lines are separated by 50 µs; ILD contour lines are separated by 2 dB. With all feathers intact, the ITD extrema (+, −, in circles) lie at about ±110°; ITDs strongly change with azimuth, but hardly with elevation (**a**). With an intact ruff, a strong ILD gradient can be seen in elevation (**d**). Removing the reflector feathers brings the ITD extrema to about ±90° (**b**) and removes the elevation gradient from the ILDs, which now vary mainly with azimuth like the ITDs (**e**). Removing the remaining head feathers does not induce any further significant change to the binaural cues (**c** and **f**). Data are from the living bird (von Campenhausen and Wagner, 2006).

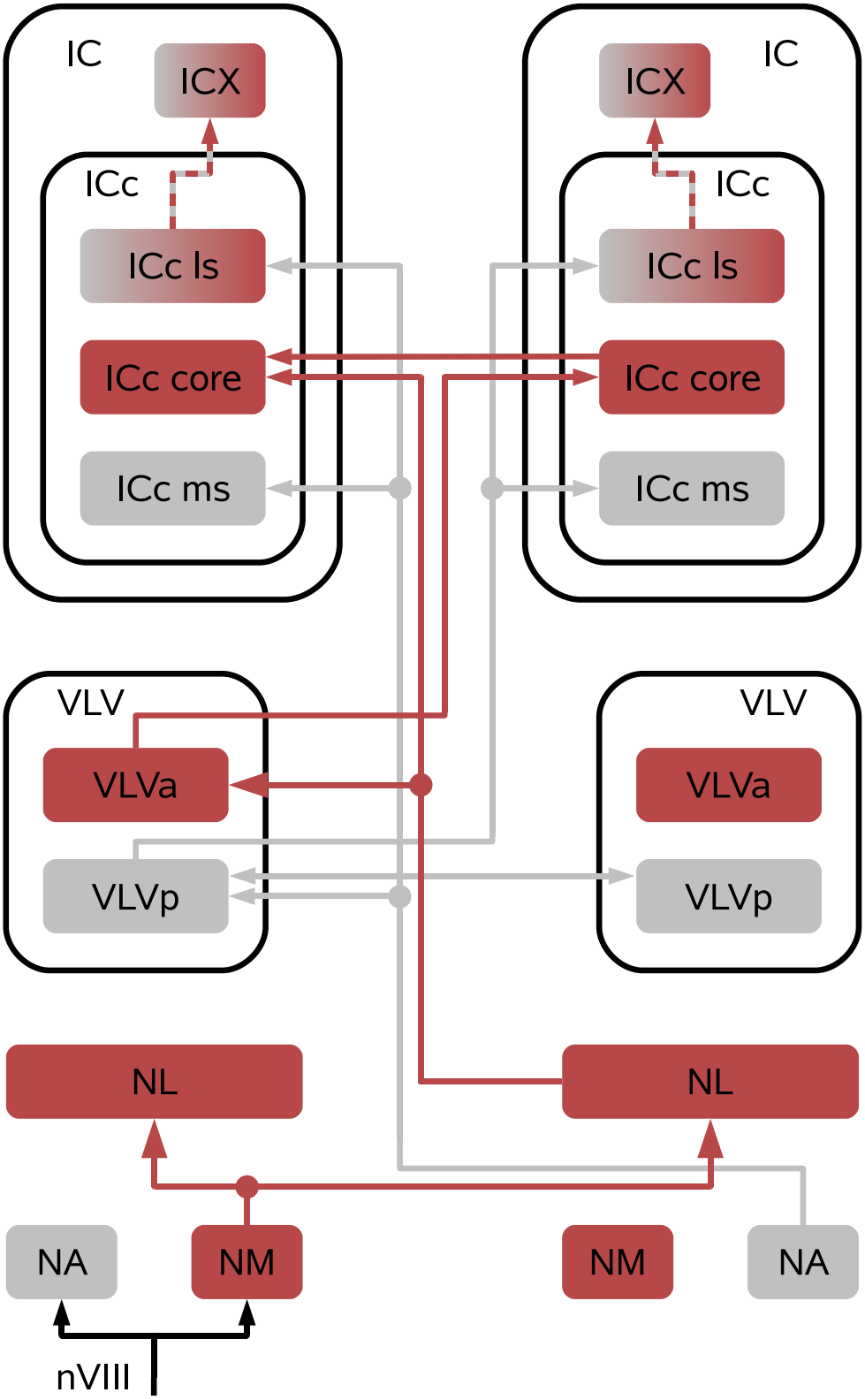
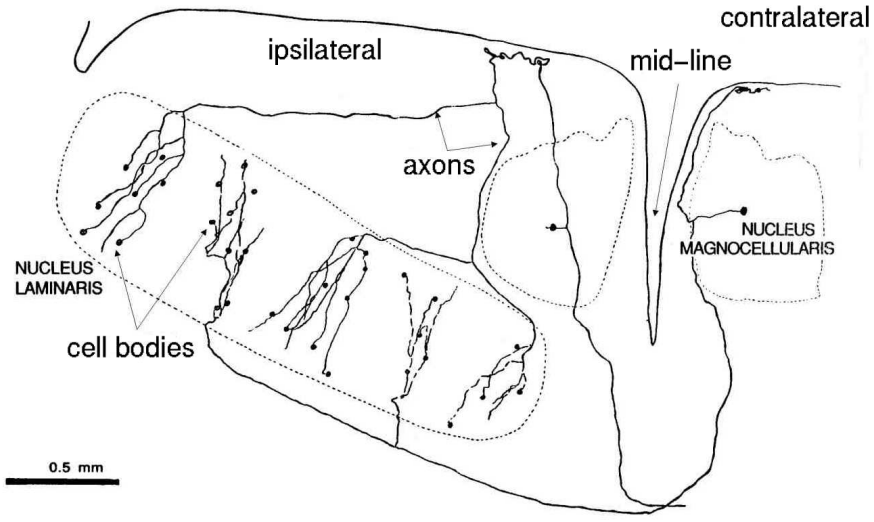


Figure 2.8: Barn owl time and intensity pathways. Nuclei and connections in red belong to the time pathway, whereas nuclei and connections in gray process level differences. Structures filled with a gradient and two-colored connections are involved in the processing of both time- and level differences (based on Konishi (2000)).



(Carr and Konishi, 1988)

Figure 2.9: Time pathway nuclei in the barn owl midbrain. The cell bodies in nucleus laminaris constitute the coincidence detector neurons, whereas the axons connecting the nucleus magnocellularis neurons to the nucleus laminaris cells act as delay lines. Note the similarity to the Jeffress model in Figure 2.5.

Processing of Time Information

Auditory nerve fibers of the barn owl achieve phase locking (preferential firing at a specific phase of the stimulus) for frequencies up to 10 kHz, which is remarkable since most other investigated species achieve phase locking only up until about $5 - 6$ kHz. Although inferior for frequencies above 1 kHz, strong phase locking is also present in the nucleus magnocellularis (NM), which is directly innervated by the auditory nerve (Köppl, 1997b). Furthermore, 8th cranial (i.e. auditory) nerve fibers as well as NM neurons exhibit narrow frequency tuning (Köppl, 1997a).

NM is the beginning of the time pathway. The second station, nucleus laminaris (NL) is the first to receive binaural input and thus is the first site for processing ITD. NL neurons are highly frequency specific. The axons projecting from NM to NL (delay lines) and the neurons in NL (coincidence detectors) constitute a Jeffress-like system (see Section 2.3.3). Figure 2.9 shows a detail of this area, crucial to ITD processing. Note the similarity of the structures to the Jeffress model (Figure 2.5). As the binaural input is phase locked, NL neurons actually detect phase difference, resulting in phase ambiguities. These manifest themselves as cyclic responses (i.e. multiple peaks) of the coincidence detector neurons to stimuli with varying ITD. The period of these cyclic responses corresponds to the neuron's best frequency (Carr and Konishi, 1990). The effect of these multiple peaks are phantom sources, which also show in the behavior of the owl if pure tones are used as stimuli (Saberi et al., 1998).

NL projects contralaterally to one of the lemniscal nuclei (VLVa — nucleus ventralis lemnisci lateralis, pars anterior) and to the core of the central nucleus of the inferior colliculus (ICc core).

ICc core neurons exhibit similar properties as NL neurons, i.e. ITD sensitivity and narrow frequency tuning, as well as phase ambiguities (Wagner et al., 2002). ICc core projects contralaterally to the lateral shell of the central nucleus of the inferior colliculus (ICc ls). Here the time and intensity pathways converge.

Neurons in ICc ls are more broadly tuned than in earlier processing stations, as they

integrate the responses of several neurons from ICc core with varying best frequency, but same best delay. This way, phase ambiguities are eliminated (Mazer, 1998).

ICc ls projects to the ipsilateral external nucleus of the inferior colliculus (ICX). In ICX, neurons are sensitive to specific regions of space (ITD/ILD). Furthermore, they are arranged in a map, meaning that elevation sensitivity (represented by ILD) of the neurons changes in the dorsoventral direction, whereas azimuth sensitivity (represented by ITD) changes in the rostrocaudal direction (Knudsen and Konishi, 1978a,b).

Processing of Intensity Information

The intensity pathway starts with nucleus angularis (NA), which receives input from the auditory nerve. Auditory nerve fibers encode sound intensity as firing rate, responding to an increase in sound pressure level with an increased firing rate (Köppl and Yates, 1999). Angularis neurons hardly exhibit any phase locking, but are sensitive to stimulus intensity (Sullivan and Konishi, 1984).

Nucleus angularis projects to the contralateral nucleus ventralis lemnisci lateralis, pars posterior (VLVp). In addition to this excitatory input, VLVp receives inhibitory input from the contralateral VLVp (Takahashi and Keller, 1992). Thus, VLVp, due to its direct excitatory and indirect inhibitory (via contralateral VLVp) input, is the first site of binaural interaction in the barn owl's intensity pathway. Consequently, VLVp neurons are sensitive to interaural level differences (Moiseff and Konishi, 1983). Responses of VLVp neurons vary from dorsal to ventral, with more dorsal neurons responding best to ILDs with higher intensities on the contralateral ear. Neurons with a more ventral position respond maximally to ILDs with higher intensities on the ipsilateral ear (Manley et al., 1988; Mogdans and Knudsen, 1994; Takahashi et al., 1995). The cause of this variation lies in a dorsoventral gradient of inhibition within VLVp. This inhibition is strongest for dorsal neurons and decreases when moving to more ventral regions of the nucleus.

The next stage in the intensity pathway is the lateral shell of the central nucleus of the inferior colliculus (ICc ls). ICc ls receives direct input from the contralateral NA as well as from the contralateral VLVp. ICc ls neurons are more broadly tuned to frequency than NA or VLVp neurons.

The last auditory-only nucleus, ICX, performs across-frequency integration and integrates ITDs and ILDs into a map of auditory space.

Before going into the description of the algorithms and methods proper to this dissertation, we have to give a brief account on basic methods used in mobile robotics, which are used in this work as basis. We therefore now have to perform a complete change of subject and go from the description of methods implemented by nature, to the description of methods implemented on autonomous mobile robots.

2.4 Mobile Robotics

In this section, we give an overview on techniques and methods used in mobile robotics which are relevant to this thesis.

2.4.1 Robot Navigation

The single most important feature a mobile robot needs to have is its ability to navigate in its environment. This involves two different, but related aspects: mapping and

self-localization. During mapping, the robot explores an unknown environment and memorizes it by building a map. In the case of self-localization, the robot has to find out its current position within the map of the environment. The combination of these two problems, i.e. building a map of an unknown environment while keeping track of its position, is called *simultaneous localization and mapping* or simply *SLAM*.

We will not go into detail on the problem of mapping or SLAM in general. For an overview, see Durrant-Whyte and Bailey's two-part tutorial on essential algorithms (Durrant-Whyte and Bailey, 2006a) and state of the art (Durrant-Whyte and Bailey, 2006b) on SLAM.

For the purpose of this thesis, we will only concern ourselves with self-localization, i.e. given a known map of the environment and sensor readings, how can the robot know at what position in the map it is currently located.

The simplest way to do this would be to give the robot its starting point within the map and let it update its position with odometry data, i.e. keeping track of distance traveled and direction changes. Unfortunately, in practice, this method is highly unreliable. The main problem are errors in odometry, which cannot be avoided (e.g. wheel slippage, which, among other things, depends heavily on the nature of the floor surface). These errors quickly add up to a point where navigation by odometry alone becomes futile. Additionally, it would be beneficial to let the robot work out by itself where it is instead of having to specify the starting point as this alone can already introduce significant errors.

Using a less error-prone sensor for the orientation in the environment would reduce, but not eliminate the problem. For example, a laser scanner does have a high precision in its measurements, but from time to time, false readings appear. Maybe because a surface did not reflect the beam (reading too long or missing) or an unexpected (i.e. not in the map) object appeared (reading too short).

Clearly, what is needed for self-localization is a method that is robust against measurement errors due to the sensors themselves as well as errors caused by the environment. Furthermore, the method should make no assumptions as to where the robot is initially located.

It turns out that statistical methods can solve the problem. By not trying to find an exact solution, but rather by modeling the position as a probability distribution over the set of all possible locations, self-localization becomes more robust against sensor and environmental noise.

In the following, we formalize the robot's position at a discrete time k as the vector $\mathbf{x}_k = (x, y, \theta)^T$ of Cartesian x, y-coordinates and the orientation θ. The coordinate system origin is the $(0, 0, 0)^T$ position of the occupancy grid map of the environment. An occupancy grid defines a probability $Occ(x, y)$ for each cell of the grid with $Occ(x, y) = 0$ if the cell position is certain to be empty and $Occ(x, y) = 1$ if the cell is sure to be occupied by an obstacle (Moravec and Elfes, 1985).

An observation at time k is a vector $\mathbf{z}_k$ of data from one or more sensors and $\mathbf{Z}_k = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k\}$ is the set of all observations from time 1 up to and including the current time k.

2.4.2 Bayesian Filtering

The framework in which this probabilistic self-localization is performed is that of *Bayesian filtering*.

We will first describe the process in a more generic fashion, using continuous time (indexed t), before discussing its use in mobile robot self-localization, which uses discrete

time (indexed k).

Bayesian filtering is a recursive process. For the algorithm to be computationally tractable at all, it has to make use of the *Markov assumption*. This means that the state at time t, $\mathbf{x}_t$, is only dependent on $\mathbf{x}_{t-1}$, i.e. $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \ldots, \mathbf{x}_0) = p(\mathbf{x}_t|\mathbf{x}_{t-1})$. Furthermore, it is assumed that observations are only dependent on the current state, i.e. $p(\mathbf{z}_t|\mathbf{x}_t, \ldots, \mathbf{x}_0) = p(\mathbf{z}_t|\mathbf{x}_t)$. The result of Bayesian filtering is a probability distribution $p(\mathbf{x}_t|\mathbf{Z}_t)$ which specifies (in our case) the probability of the robot at being in any position $\mathbf{x}_t$ in the map, if the data $\mathbf{Z}_t$ were observed up to time t. Bayesian filtering is performed in two steps:

Prediction step During the prediction step, the old belief, $p(\mathbf{x}_{t-1}|\mathbf{Z}_{t-1})$, is projected to the current time t. This is done with a model of the underlying system dynamics, the *motion model* or *system model* $p(\mathbf{x}_t|\mathbf{x}_{t-1})$. This model incorporates assumptions about the robot's movement like finite speed, i.e. it is more probable for $\mathbf{x}_t$ to be located relatively close to $\mathbf{x}_{t-1}$ and highly improbable that $\mathbf{x}_t$ is at the opposite side of the map from $\mathbf{x}_{t-1}$ (for a large map at least). Additionally, motor commands to the robot have to be taken into consideration by the motion model in order to produce accurate projections. The result of the prediction step is

$$p(\mathbf{x}_t|\mathbf{Z}_{t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{Z}_{t-1})d\mathbf{x}_{t-1}, \tag{2.17}$$

i.e., the probability of the robot being at position $\mathbf{x}_t$, given all sensor measurements $\mathbf{Z}_{t-1}$ up to and including time $t-1$.

Update step During the update step, the predicted positions are updated (or corrected) by the current observations, i.e. sensor data. This is done by multiplying the probability of each projected position $p(\mathbf{x}_t|\mathbf{Z}_{t-1})$ with the *perceptual model* or *sensor model* $p(\mathbf{z}_t|\mathbf{x}_t)$ (the probability of observing $\mathbf{z}_t$ from position $\mathbf{x}_t$).

$$p(\mathbf{x}_t|\mathbf{Z}_t) = \alpha p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{Z}_{t-1}) \tag{2.18}$$

Here, α is a normalizing factor, constant relative to $\mathbf{x}$. It is usually ignored in practice as its only purpose is to make the integral of $p(\mathbf{x}_k|\mathbf{Z}_k)$ equal to 1 (as required by the correct mathematical definition of a probability distribution). It is defined as follows:

$$\alpha = p(\mathbf{z}_t|\mathbf{Z}_{t-1}) = \int p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{Z}_{t-1})d\mathbf{x}_t. \tag{2.19}$$

Bayes filters are a relatively abstract concept. They basically represent a probabilistic framework for recursive state estimation. Actual implementations depend on the specification of the system model $p(\mathbf{x}_t|\mathbf{x}_{t-1})$, the perceptual model $p(\mathbf{z}_t|\mathbf{x}_t)$ and the representation of the belief $p(\mathbf{x}_t|\mathbf{Z}_t)$ (Fox et al., 2003).

Performing Bayesian filtering as described above on an occupancy grid, consisting of individual cells and in discrete time is called grid-based Markov localization (because of the Markov assumption) or simply *Markov localization* (Fox et al., 1999). As $p(\mathbf{x}_k|\mathbf{Z}_k)$ has to be computed for every $\mathbf{x}_k$, i.e. for every cell in the occupancy grid and for every possible robot orientation in each cell, Markov localization is not very efficient. As an example, for a medium-sized 30 m × 30 m occupancy grid with a 5 cm × 5 cm cell size and 2° in angular resolution, there are 7 200 000 different positions $\mathbf{x}_k$, which the robot can occupy.

We will now present two additional implementations of Bayesian filtering for mobile robot localization, namely Kalman filtering and Monte Carlo localization.

2.4.3 The Kalman Filter

One very efficient implementation of a Bayes filter is the Kalman filter (Kalman, 1960). It estimates the state of a linear dynamic system from a series of noisy measurements.

The underlying dynamic system model is described by the following two equations:

$$\mathbf{x}_k = \mathbf{F}_k\mathbf{x}_{k-1} + \mathbf{B_k}\mathbf{u}_k + \mathbf{w}_k \tag{2.20}$$

$$\mathbf{z}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k \tag{2.21}$$

Equation 2.20 is the filter's state transition (or system) model. $\mathbf{F}_k$ is the state transition matrix. $\mathbf{B}_k$ is the control-input model matrix, applied to the control vector $\mathbf{u}_k$. The vector $\mathbf{w}_k$ represents the process noise which is assumed to be drawn from a zero mean multivariate normal distribution with covariance matrix $\mathbf{Q}_k$, i.e. $\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{Q}_k)$.

Equation 2.21 is the perceptual model. $\mathbf{H}_k$ is the observation model matrix which provides the mapping from the state vector $\mathbf{x}_k$ to the observation vector $\mathbf{z}_k$. $\mathbf{v}_k$ is the observation noise, drawn from a zero mean multivariate normal distribution with covariance matrix $\mathbf{R}_k$, i.e. $\mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R}_k)$.

The actual update and prediction steps of the Kalman filter are described in Equations 2.22–2.28. The notation $\hat{\mathbf{x}}_{n|m}$ stands for the estimate of $\mathbf{x}$ at time n, given observations up to and including time m.

The state of the Kalman filter at time k is characterized by the two variables $\hat{\mathbf{x}}_{k|k}$ (the estimate of the state $\mathbf{x}$ at time k, given observations up to and including time k) and $\mathbf{P}_{k|k}$ (the covariance of the state estimate at time k, given observations up to and including time k).

Prediction step

state prediction:
$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k\hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}_{k-1}\mathbf{u}_{k-1} \tag{2.22}$$

estimate covariance prediction:
$$\mathbf{P}_{k|k-1} = \mathbf{F}_k\mathbf{P}_{k-1|k-1}\mathbf{F}_k^T + \mathbf{Q}_{k-1} \tag{2.23}$$

Update step

innovation:
$$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k|k-1} \tag{2.24}$$

innovation covariance:
$$\mathbf{S}_k = \mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^T + \mathbf{R}_k \tag{2.25}$$

optimal Kalman gain:
$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}_k^T\mathbf{S}_k^{-1} \tag{2.26}$$

updated state estimate:
$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\tilde{\mathbf{y}}_k \tag{2.27}$$

updated estimate covariance:
$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k|k-1} \tag{2.28}$$

Through innovation, the Kalman filter minimizes the error between the actual state and the estimated state. Ideally, if the model is accurate and the start values $\hat{\mathbf{x}}_{0|0}$ and $\mathbf{P}_{0|0}$ reflect the distribution of the initial state values (which corresponds to a normal distribution), the mean of the error of all estimates is 0 (Kalman, 1960), i.e. the Kalman filter is an ideal state estimator.

Although the Kalman filter is very efficient and fast, it has two major drawbacks, which make it ill-suited for mobile robot-self localization:

- The state transition (Eq. 2.20) and observation (Eq. 2.21) models are linear. This means that the systems described by these models are restricted to linear dynamic systems.

- As the system and observation noises are assumed to be Gaussian and the models linear, the distribution of the state estimates is also a Gaussian. This means that the distribution of state estimates is unimodal, i.e. the Kalman filter can only produce one hypothesis on the state (which might be wrong).

The first drawback (the linearity) can be overcome using modifications to the original Kalman filter, like e.g. the *extended* Kalman filter (EKF) or the *unscented* Kalman filter (UKF), at the cost of losing the optimality of the solution (Julier et al., 1995).

A solution to the second problem consists in taking into consideration multiple hypotheses simultaneously, using one Kalman filter for each hypothesis. The problem of this method is how to generate or remove individual hypotheses.

2.4.4 Monte Carlo Localization (MCL)

For the reasons mentioned above, we do not use Kalman filtering for the robot's self-localization module. Instead we use a variant of Bayes filtering for robot localization called Monte Carlo localization (MCL) (Dellaert et al., 1999).

It is an instance of the class of *sequential Monte Carlo* (SMC) algorithms. This type of algorithm tries to approximate a target distribution that is hard (or impossible) to derive analytically by sampling from a *proposal distribution* or *proposal density*.

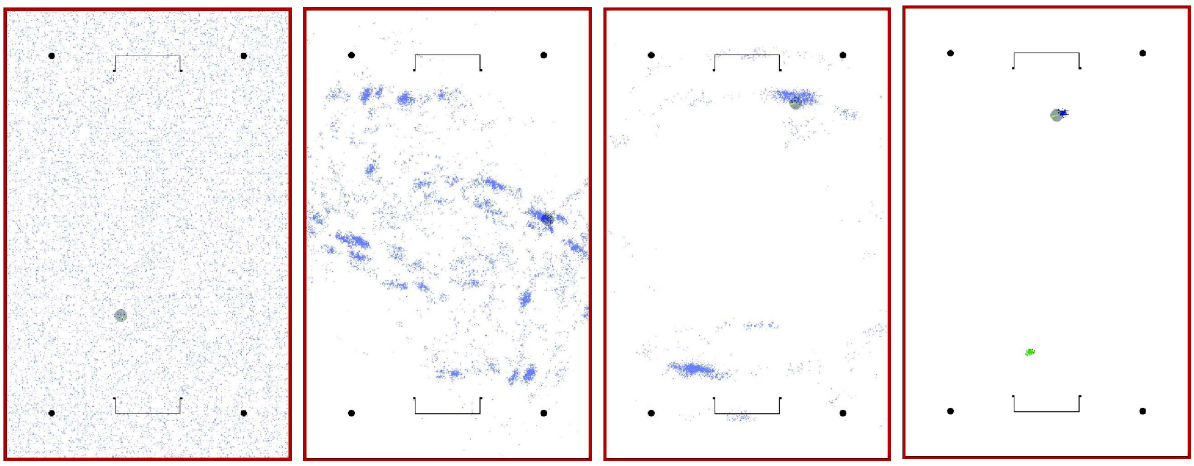
The method estimates the probability density $p(\mathbf{x}_k|Z_k)$ at timestep k by a set of N samples (or particles; this is why it is also called *particle filtering*) $S_k = \{s_k^i; i = 1, \ldots, N\}$, drawn from it. Each sample represents one hypothesis about the robot position. The method works recursively at each timestep k and consists of the prediction and update steps:

Prediction step Starting from the set of samples S_{k-1}, computed in the previous timestep, the robot's motion model $\mathbf{u}_{k-1}$ is applied to each particle s_{k-1}^i, giving a prediction of where it is located at time k, taking into consideration the assumptions on the robot's movement and motion commands contained in $\mathbf{u}_{k-1}$. This corresponds to sampling from the density $p(\mathbf{x}_k|s_{k-1}^i, \mathbf{u}_{k-1})$ (the proposal density). Thus, a new sample set $S'_k = \{s'^i_k : i = 1 \ldots N\}$ is built, approximating a random sample from the empirical predictive density $\hat{p}(\mathbf{x}_k|Z_{k-1})$.

Update step In the second step, the empirical predictive density is updated by sensor readings $\mathbf{z}_k$ from the current timestep. This is done by weighting each particle in S'_k by the weight $m_k^i = p(\mathbf{z}_k|s'^i_k)$. $p(\mathbf{z}_k|s'^i_k)$ is the sensor model, i.e. the probability of having sensor readings $\mathbf{z}_k$ at position s'^i_k. The final sample set S_k is obtained by randomly drawing N samples (resampling) from the weighted set $\{(s'^i_k, m_k^i)\}$. The resampling is done in such a way that samples with higher associated weight get selected with higher probability. The new S_k approximates a sample from the sought-after distribution $p(\mathbf{x}_k|\mathbf{Z}_k)$.

After the update, the algorithm starts over again in the next timestep, $k + 1$, with the prediction phase.

To summarize, at the start of a global localization process ($k = 0$), the robot does not know its position and therefore every hypothesis is equally likely. This means that the samples from the initial sample set $S_0 = \{s_0^i : i = 1 \ldots N\}$ are distributed uniformly among the unoccupied cells of the occupancy grid map. At time $k = 1$, the samples will be "moved" on the grid according to the motion commands received by the robot and its motion model. After taking sensor readings, each sample is assigned a weight



a **b** **c** **d**

Figure 2.10: Example of MCL on a RoboCup field. The actual position of the robot is depicted by the gray circle. **a** Initially, the robot position is unknown and the samples are randomly distributed. **b** After some movement of the robot, sample clusters have appeared. **c** Now most samples have clustered around two position hypotheses. **d** Due to the inherent symmetry of the field, two possible position estimates emerge as final hypotheses (Strack et al., 2005).

corresponding to the sensor model, i.e. samples nearer to positions from which the sensor readings would make sense get higher weights. Resampling produces with higher probability samples in the vicinity of the samples with higher weight. Thus, the samples start to *condense* around these positions (hence another name for the algorithm — *condensation algorithm*).

After driving around in the environment for some time and taking new sensor updates, the samples cluster around only a few positions. Thus, the belief about the robot's position converges to only a few most likely hypotheses (Figure 2.10).

Ideally, if there are not too many symmetries in the environment, one hypothesis remains, representing the robot's strongest belief on its position.

Localization accuracy

With the above approach it is possible to localize with high accuracy in almost any indoor environment. Depending on the cell size of the occupancy grid the average error usually is around 15 cm in position and 1° in orientation even in large environments. The method used for one particular robot is presented in detail in Strack et al. (2005).

2.4.5 Novelty Filter

Although the Markov assumption underlying Monte Carlo localization (see Section 2.4.2) imposes a static environment, the method is as robust against occasional changes (e.g. doors opening or closing, or people walking by) as Markov localization (Fox et al., 1999).

If the changes are too drastic with regard to the map (if it is too crowded, for example), the method breaks down.

By adding a novelty filter as described in Fox et al. (1998), MCL can also be used in such highly dynamic environments. The novelty filter discards laser readings which, related to the map and the currently believed position, are too short and can thus be classified to hit dynamic (i.e. not in the known map) obstacles.

If the robot believes it is at position $\mathbf{x}$ in the map of its environment, there exists a probability $p(d_j|\mathbf{x})$ of measuring distance d_j with the laser sensor ($d_1, \ldots, d_n$ being a discrete set of distance measurements). This probability distribution represents the expected measurement. The probability $p_n(d_i|\mathbf{x})$ of d_i being shorter than expected can be computed by

$$p_n(d_i|\mathbf{x}) = 1.0 - \sum_{j \leq i} p(d_j|\mathbf{x}) \tag{2.29}$$

As there sometimes can be multiple probable hypotheses about the robot's position, $p_n(d_i|\mathbf{x})$ is averaged over all these position beliefs:

$$p_n(d_i) = \sum_{\mathbf{x}} p_n(d_i|\mathbf{x}) Bel(\mathbf{x}), \tag{2.30}$$

with $Bel(\mathbf{x})$ being the robot's belief that it is at position $\mathbf{x}$.

Now the novelty filter will exclude every laser reading d from MCL that is too short with a high probability, i.e. $p_n(d) > \theta$, where θ is a pre-determined threshold.

2.4.6 Laser-based Object Recognition

The discarded laser readings need not be lost however, as they represent information on dynamic objects.

In fact, object recognition can be performed on the readings classified as being too short. The first step consists in clustering groups of these dynamic readings. This can be done because measurements belonging to one particular object cannot be farther away from each other than the diameter of the object's convex hull. The laser signature of the objects is then used to classify the clustered groups by size and form. This enables the robot to distinguish between different objects. This classification takes place each time new laser readings arrive. Thus, they can change both in number and position. To stabilize the robot's perception, the Hungarian method (Kuhn, 1955) is used to track objects from one cycle to the next. This method is based on a $n \times m$ cost matrix with each element representing the cost of assigning the i-th currently detected object to the j-th previously detected object. The Hungarian method solves assignment problems in polynomial time (Frank, 2004).

The object recognition was originally developed for robotic soccer. In the soccer setting the robots are able to distinguish between teammates and opponents, and even humans can be told apart. The classification heuristic is still rough at the moment. Nevertheless, the object recognition output is accurate enough to perform an association between sound sources and dynamic objects.

2.5 Markov Chain Monte Carlo

The basis of our method for tracking entities consisting of dynamic objects and associated sound sources (Section 5.2) is the *Markov chain Monte Carlo* (MCMC) algorithm.

Like sequential Monte Carlo (Section 2.4.4), MCMC is a method for sampling from arbitrary probability distributions.

2.5.1 Markov Chains

In order to explain the Markov chain Monte Carlo algorithm, we first have to explain what Markov chains are.

A Markov chain is a sequence of states, having the *Markov property*. This means that, given the present state, future states are only dependent on the present state, not on past states. At each timestep, the Markov chain may remain in the same state or change to another one according to a certain probability distribution.

In fact, we already implicitly encountered Markov chains in Section 2.4.2. Bayesian filtering methods try to estimate the hidden (because not directly observable) states of a Markov chain with the help of models of the system and measurements taken at each timestep.

More formally: a Markov chain is a sequence of random variables $X_1, X_2, X_3, \ldots$ with the Markov property:

$$P(X_{n+1} = x | X_n = x_n, \ldots, X_1 = x_1) = P(X_{n+1} = x | X_n = x_n), \tag{2.31}$$

n being the time index. This means that the next state depends only on the current state. The countable set Ω of possible values for the random variables X_i is called the *state space* of the Markov chain.

Markov chains can be represented by a directed graph, where the edge labels are the probabilities of going from one state to other states. Such a probability is called *transition probability*.

Markov Chain Properties

Transition probabilities The *single step transition probability* is defined as the probability of directly going from state i to state j, without intermediate states:

$$p_{ij} = P(X_1 = j | X_0 = i). \tag{2.32}$$

The probability of going from state i to state j in n steps (*n-step transition probability*) is defined as

$$p_{ij}^{(n)} = P(X_n = j | X_0 = i). \tag{2.33}$$

Reducibility A state j is said to be *accessible* from state i, if there exists a non-zero probability of reaching j from i:

$$P(X_n = j | X_0 = i) > 0, \qquad n \geq 0. \tag{2.34}$$

State i and j *communicate*, if i is accessible from j and j is accessible from i. One can define a *communicating class* C of states as a set of states, where each pair of states communicates with each other, but with no other state outside of C.

If it is possible to get from any i in the state space to any other state j in the state space (i.e. the state space is a communicating class), the Markov chain is said to be *irreducible*. The directed graph of an irreducible Markov chain is strongly connected. This means that a disconnected graph cannot represent an irreducible Markov chain.

Periodicity If returning to a state i is only possible after k steps ($k > 1$), the state is said to be *periodic* with period k, i.e.

$$k = \gcd\{n : P(X_n = i | X_0 = i)\}, \tag{2.35}$$

gcd being the greatest common divisor. If $k = 1$, the state is *aperiodic*.

RECURRENCE A state i is defined to be *transient*, if there is a non-zero probability of never returning to it:

$$P(T_i = \infty) > 0, \tag{2.36}$$

with the random variable $T_i = \inf\{n \geq 1 : X_n = i | X_0 = i\}$ being the *hitting time* of state i, i.e. the first return time to i. A state that is not transient is *recurrent* or *persistent*. If the expected return time $M_i = E[T_i]$ is finite, the state i is *positive recurrent*. If it is impossible to leave a state i, it is *absorbing*, i.e. $p_{ii} = 1$ and $p_{ij} = 0$ for $i \neq j$.

ERGODICITY A state i is *ergodic*, if it is aperiodic and positive recurrent. A Markov chain is ergodic, if all its states are ergodic. Note that a finite state irreducible Markov chain needs only to satisfy aperiodicity in order to be ergodic, as irreducibility and a finite state set imply positive recurrence.

STATIONARY DISTRIBUTION If the Markov chain can be described by a time-independent $N \times N$ matrix p_{ij} ($|\Omega| = N$), i.e. the stochastic process describing the chain is time-homogeneous (the transition probabilities do not change over time), the vector $\boldsymbol{\pi} = (\pi_1 \dots \pi_N)$ is called *stationary distribution* (or *invariant measure*) if its elements π_j sum to 1 and it satisfies the following property:

$$\pi_j = \sum_{i \in \Omega} \pi_i p_{ij} \tag{2.37}$$

An irreducible Markov chain has a stationary distribution if all its states are positive recurrent. In that case, $\boldsymbol{\pi}$ is unique. Furthermore, $\pi_j = 1/M_j$, i.e. π_j is related to the expected return time.

In case of an ergodic, finite state Markov chain,

$$\lim_{n \to \infty} p_{ij}^{(n)} = \frac{1}{M_j} \qquad \forall i, j \in \Omega. \tag{2.38}$$

Such a chain will — regardless of the distribution of the starting state — reach its equilibrium distribution after a certain time, the *mixing time*.

In summary, the stationary distribution $\boldsymbol{\pi}$ of an ergodic Markov chain represents the overall probability distribution of the states of the chain, i.e. π_j is the overall fraction of time spent in state j.

2.5.2 Markov Chain Monte Carlo Basics

Markov chain Monte Carlo is used to sample from a probability distribution from which it is difficult to sample directly. The algorithm achieves this by constructing an ergodic Markov chain having state space Ω and the desired distribution as its stationary distribution $\boldsymbol{\pi}$. As the Markov chain transitions correspond to random perturbations in Ω, these are easy to simulate. In order to sample from $\boldsymbol{\pi}$, the algorithm starts with a random state from Ω and simulates the Markov chain for a number N, of steps. Due to the ergodicity of the chain, the distribution of the final state will be close to $\boldsymbol{\pi}$, if N was chosen large enough.

Although sequential Monte Carlo (the basis of Monte Carlo localization; see Section 2.4.4) and Markov chain Monte Carlo are both used to generate samples from a probability distribution from which it is difficult to sample directly, they have different strengths and weaknesses.

Algorithm 2.1: Metropolis-Hastings algorithm to produce a sample from a distribution $\pi(x)$ that is difficult to sample from. The input is a sample number n_{smpls} and an initial state x^0. The output is a sample $\hat{x}$ from the probability distribution $\pi(x)$.

Input: n_{smpls}, x^0
Output: $\hat{x}$
$x^k \leftarrow x^0$
for $k = 1$ to n_{smpls} **do**
  sample x' from $Q(x'|x^k)$
  compute $\alpha = \min\left(\frac{\pi(x')}{\pi(x^k)}\frac{Q(x^k|x')}{Q(x'|x^k)}, 1\right)$
  sample u from $\mathrm{U}(0,1)$
  if $u < \alpha$ **then**
    $x^k \leftarrow x'$
  end if
end for
$\hat{x} \leftarrow x^k$

As a sequential, recursive algorithm, SMC can be very fast and efficient and thus is ideally suited for online applications. Additionally, as past data is not saved, its memory footprint is comparatively low. The drawback is, however, that SMC requires a proposal distribution from which it is easy to sample and which, at the same time, is a relatively good approximation of the target distribution. This requirement often is impossible to meet (especially with high dimensional probability distributions). MCMC on the other hand is iterative and, as a consequence, it can be much slower than SMC. Furthermore, it requires the storage of available data and thus has a much higher memory footprint than SMC. Another disadvantage of MCMC is that it is difficult to control the mixing time, i.e. the time it takes for the Markov chain to reach its stationary distribution. The advantage of MCMC is that it is quite easy to construct a Markov chain with the desired properties — in contrast to the stringent requirements imposed on the proposal distribution by SMC (Doucet et al., 2001; Andrieu et al., 2003).

2.5.3 Metropolis-Hastings Algorithm

Probably the most widely used Markov chain Monte Carlo algorithm is the Metropolis-Hastings algorithm (see Algorithm 2.1), developed by Metropolis et al. (1953) and generalized by Hastings (1970). The main features of the algorithm are the following (Hastings, 1970):

1. The computation relies on the target distribution $\pi(x)$ only through the ratio $\pi(x')/\pi(x)$. This means that the normalizing constant need not be known as it cancels out anyway. Furthermore, the only requirement on $\pi(x)$ is that it can be evaluated (up to the normalizing constant) at x.

2. A sample sequence is generated by simulating a Markov chain. This means that, in contrast to SMC (Section 2.4.4), the samples are correlated.

The algorithm generates a sequence of states $x^0, \ldots, x^k, x^{k+1}, \ldots$ of a Markov chain having $\pi(x)$ as its stationary distribution.

In order to find a successor state x^{k+1} to the state x^k, a state x' is sampled from the *proposal density* $Q(x'|x^k)$. The proposal density has to be defined on the same domain as the target distribution $\pi(x)$. Furthermore, it should be easy to sample from $Q(x'|x^k)$. Other than these two requirements, there are hardly any restrictions on $Q(x'|x^k)$, although for best performance, the proposal density should approximate the target distribution (for a more in-depth discussion of proposal density selection, see Chib and Greenberg (1995)).

The new state x^{k+1} is found by performing the following steps:

1. Compute α (the *probability of move*):

$$\alpha = \min\left(\frac{\pi(x')}{\pi(x^k)}\frac{Q(x^k|x')}{Q(x'|x^k)}, 1\right). \tag{2.39}$$

2. Draw u from the uniform distribution $\mathrm{U}(0,1)$.

3. Accept x' ($x^{k+1} = x'$) under the condition that

$$u < \alpha, \tag{2.40}$$

   otherwise reject x' ($x^{k+1} = x^k$).

Or, formulated differently, these three steps are equivalent to the following two statements:

$$P(x^{k+1} = x') = \alpha, \tag{2.41}$$
$$P(x^{k+1} = x^k) = 1 - \alpha, \tag{2.42}$$

i.e. x' is accepted with probability α as the new state and rejected with probability $1 - \alpha$. This means that, even for high values of α ($0.5 \leq \alpha < 1$), there still is a (small) probability of the chain remaining in the previous state x^k. Conversely, for low values of α ($0 < \alpha \leq 0.5$), there remains a (small) chance of x' being accepted as the new state.

It can be shown that after a certain time, the so-called *burn-in period* (which corresponds to the mixing time of the Markov chain), the samples generated by Metropolis-Hastings are distributed according to the stationary distribution $\pi(x)$ of the simulated Markov chain (Hastings, 1970).

If a symmetric proposal density is used, i.e. $Q(x^k|x') = Q(x'|x^k)$, the ratio underlying the computation of α (Eq. 2.39) simplifies to $\pi(x')/\pi(x^k)$. This is the original Metropolis algorithm Metropolis et al. (1953). In this case, if x' is more probable than x^k (or as probable), i.e. $\pi(x') \geq \pi(x^k)$, x' is always accepted. Otherwise, x' is only accepted with probability α. This means that the algorithm will always go "uphill", but it will only go "downhill" with probability α.

This behavior is reminiscent of *simulated annealing*, an algorithm for finding the global minimum of a function in a large search space. Simulated annealing works in a similar fashion to the Metropolis algorithm by starting at a random point of the search space and probabilistically generating new points, trying to minimize the given function. Simulated annealing has the general tendency of going "downhill", but with a non-zero probability of going "uphill", which allows it to get unstuck from local minima encountered on the way to the global minimum. In fact, simulated annealing is the adaptation of the Metropolis algorithm to optimization (Kirkpatrick et al., 1983).

Chapter 3

Related Work

In this chapter we will review the related work relevant to this thesis. It is subdivided into two major parts. Section 3.1 will present work pertaining to the sound localization part of the dissertation, whereas Section 3.2 will consider tracking and data association algorithms.

3.1 Binaural Sound Source Localization on Mobile Robots

The method proposed by Huang et al. (1999) uses three microphones, arranged in a horizontal plane on the tips of an equilateral triangle. They are mounted on the robot in such a way that one tip of the triangle points in the moving direction. An echo prediction model is used to filter out reverberations from the microphone signals. The time delay Δt_{ij} between microphones i and j is determined as the time difference between zero-crossings of the echo-free signals at each microphone in the pair. To compute the azimuth to the sound source, the microphone pair yielding the minimum $|\Delta t_{ij}|$ is selected. From this the value of the angle is computed, whereas the microphone pair with the maximal time difference determines the sign. The accuracy of the algorithm lies in the range of 2°–3° using a broadband signal (hand-clapping). Interestingly, the accuracy with a narrowband signal (1 kHz sinusoid) was higher (0°–1°).

Lv and Zhang (2008) use four microphones arranged in a square on a robot head. Using geometry and trigonometry, the position of a sound source — azimuth, elevation and distance — can be expressed in terms of time delays between microphone pairs. To obtain these time differences, correlation in the time domain is used. For the experiments, a speech sound source was positioned at a distance of 4 m, with an azimuth of 60° and an elevation of 60°. In a quiet environment, using a microphone array with a side length of 20 cm, the error was 0.8° in azimuth, 3° in elevation and 20 cm in distance. With a side length of 8 cm, the errors were 5°, 7.5° and 80 cm in azimuth, elevation and distance, respectively. In a noisy environment, the error increased to 10°, 10.5° and 70 cm (azi, ele, dist) for the 20 cm array side length case (the 8 cm case was not tested in this situation). Unfortunately, no description was given of the kind of noise and the signal-to-noise ratios used.

Another correlation-based system (in the frequency domain) is proposed by Li and Levinson (2002). They use two microphones and compute interaural time differences through linear phase unwrapping of the cross power spectrum. It was tested in realistic conditions (4 m × 4 m room) on a mobile robot and yielded an accuracy of ±10°. This was enough for the robot to locate and approach a person carrying a food can, recognize

the can with an onboard camera and grab it.

The methods described in Nishiura et al. (2002); Martinson and Schultz (2006) are based on the *generalized cross-correlation* (GCC), specifically the variant known as *phase transform* (PHAT) or *crosspower spectrum phase* (CSP) (Knapp and Carter, 1976; Omologo and Svaizer, 1994). In this approach, time differences can be derived from CSP coefficients computed for each microphone pair. This is a fairly standard and relatively robust method in the field of sound source localization in technical systems, but requires multi-microphone arrays.

Nishiura et al. (2002) use a circular (60 cm diameter) 16-microphone array on a mobile robot. In order to avoid the shortcomings of the CSP method for multiple, correlated sound sources, they use a special microphone pair arrangement. This allows minimization of phantom sources caused by correlation artefacts through addition of the CSP coefficients of the individual microphone pairs (Nishiura et al., 2000). In combination with speech / non-speech identification, their mobile robot was capable of correctly approaching a voice source in the presence of a noise distracter and reverberation ($RT_{60} = 0.85$ s)[1]. Unfortunately, they give no quantitative evaluation of the accuracy of the algorithm.

Martinson and Schultz (2006) use the standard PHAT technique for computing time difference between microphone pairs. The originality of their approach is the *auditory evidence grid*. Similar to the occupancy grid map we use for robot navigation (cf. Section 2.4.1), each cell of the auditory evidence grid specifies the probability (initially 0.5) of the corresponding room position being occupied by a sound source. Probabilities in the auditory evidence grid are modified by a Bayesian update scheme (see Section 2.4.2), taking into account robot position, sound source localization data and statistical models of the sensor and of the robot movement. A single PHAT measurement of the microphone array would yield a multitude of possible sound source positions in the grid (considering errors and e.g. front / back confusions). But with the help of Bayesian update and modifications of the robot pose, these possibilities can be significantly narrowed down. In experiments, a 4-microphone square array was mounted on a mobile robot. Various sound source configurations were tested and a distinction was made whether the robot stopped its movement for sampling audio data or not. General performance was quite good. If only one or two sources were present, they were correctly mapped, whether the robot was pausing movement during data collection or not. During trials with three or more sources, however, at least one source was always missed.

In Murray et al. (2009), the sound source localization is based on computation of interaural time delay by correlation in the time domain. The azimuth is computed using Eq. A.4. A bandpass filter 1 kHz–4 kHz is used to use only frequency bands relevant to human speech. A signal energy measure is used to additionally filter out spurious noise bursts that could distract the system. Finally, normalizing the signals to the baseline noise level of the environment provides a form of automatic gain control. In order to track sound sources, a predictor-corrector approach using a recurrent neural network is taken. This uses current and previous sound source positions in order to predict the trajectory. With this system, the robot can keep a sound source in the $-45° \ldots +45°$ range, where the sound source localizer is the most sensitive. The system was tested on a mobile robot, using two microphones at a distance of 30 cm from each other. The accuracy lies within $\pm 1.5°$ at the center ($0°$) and $\pm 7.5°$ at the more lateral positions of $\pm 90°$.

The method presented in Valin et al. (2007) is based on the concept of a beamformer,

[1] for an explanation of RT_{60} see the end of Appendix B

which is an array of sensors electronically configured to act as one highly directional sensor (van Veen and Buckley, 1988). Eight microphones were used in two different array configurations. One (C1) was an open cube with a side length of 16 cm and the microphones mounted at the vertices. In the other configuration (C2), the microphones were placed in small holes at different locations inside the body of the robot (a vaguely humanoid torso). Sound source localization was performed by steering the beamformer[2] through different spatial positions, located on a spherical search grid and looking for the position maximizing the array's output energy. This corresponded to the sound source location. Optionally, the search grid resolution could be refined in order to increase precision. The accuracy is very good, with the error of configuration C1 being 1.10° in azimuth and 0.89° in elevation. Configuration C2 had errors of 1.44° in azimuth and 1.41° in elevation. A particle filter (cf. Section 2.4.4) was used for sound source tracking. With this, it was possible to reliably track multiple moving sound sources.

Andersson et al. (2004) present a binaural sound source localization system based on the evaluation of interaural phase differences (IPD) and interaural level differences (ILD). Both cues are computed in the frequency domain by taking the phase difference and amplitude ratios of the Fourier points of the left and right channels at each frequency. In order to generate ILDs, a spherical "head" with a diameter of 20 cm is used. Theoretical IPD and ILD values are computed by solving 3D acoustic wave propagation equations for this sphere (Handzel and Krishnaprasad, 2002). In order to perform localization, measured IPD/ILD values are matched to theoretical IPD/ILD values by minimizing an adequate norm distance value. The system achieved an error of $\pm 2°$ over the whole azimuth range. Because of the inherent spherical symmetry of the "head", sound source localization was restricted to the horizontal plane, i.e. azimuth. The system was also tested on a mobile robot. It was able to successfully approach a sound source and avoid obstacles along the way.

The method described in Berglund et al. (2008) is based on the concept of a parameterless self-organizing map (PLSOM) algorithm, a neural network algorithm used for mapping from one (usually high-dimensional) to another (usually low-dimensional) space (Berglund and Sitte, 2006). The map uses a feature vector consisting of ITD, IPD, IID, RIID[3] and volume, extracted from the signals of two microphones. The PLSOM performs the mapping of the feature vector to azimuth / elevation / distance. A reinforcement learning system is used to turn the head of the robot (a Sony Aibo ERS-210 dog-like robot) to the sound source, i.e. the robot learned online (without training) what motor commands to use in order to place a sound source in front of the head. The training set for PLSOM was white noise. Different test sets were used, consisting of white noise, speech and an 800 Hz sine wave with two harmonics. Source distance estimation was best at $\pm 60 - 70°$, and extremely bad at 0°, which comes as no surprise at it is essentially based on ILD, which should be 0 dB at 0°, independent of distance. The overall localization accuracy is in the range of $\pm 5°$ in both azimuth and elevation.

[2] "steering the beamformer" means changing the parameters of the beamformer in software in order to change the directional characteristics. The physical microphone array is not moved.

[3] The RIID is the relative interaural intensity difference, defined as IID/v, with v being the averaged volume of the left and right channel. It is used for distance detection.

3.2 Target Tracking and Data Association

In this section, we review the literature pertinent to target[4] tracking and data association (robustly assigning sensor measurements to individual targets being tracked). This will include papers on the general subject as well as papers on sound source tracking with robots.

Two of the most widespread methods in this context are Reid's *multiple hypothesis tracking* (MHT) (Reid, 1979) – including its variants and alternative implementations (overview in Blackman (2004)) – and the *joint probabilistic data association filter* (JPDAF) (Bar-Shalom and Fortmann, 1987). For an introductory review of statistical data association and tracking algorithms (including MHT and JPDAF), see e.g. Cox (1993).

In each iteration, MHT works on a set of hypotheses. Each hypothesis consists of a set of plausible tracks. When a new observation is made, a data point is assigned to a track, if it falls within a certain distance of the track. As this can be done in several ways, new hypotheses are likely to emerge. Hypotheses are evaluated by a probabilistic measure, based on Kalman filtering (Section 2.4.3) each track of the hypothesis. In this way, unlikely hypotheses can be eliminated. MHT allows the creation of new tracks (if an observation could not be assigned to an existing track) and the deletion of tracks (if that is more probable than keeping the track). As more data becomes available with time, the number of possible association vectors (i.e. observation to target mappings) increases, which is why complexity of the original MHT algorithm increases exponentially with time. Various methods have been developed to circumvent this restriction, like e.g. clustering, hypothesis and track pruning, and track merging (Blackman, 2004), but the complexity issue of MHT still persists to some degree.

Underlying JPDAF is the assumption that the number of targets is constant and known. This is a rather strong assumption, but acceptable in some situations. JPDAF combines *all* measurements with *all* tracks, i.e. it constructs all possible association vectors. With a Kalman-filter based evaluation of the probability of each assignment, only the most probable association is retained in the end. Producing every possible association might seem quite laborious, but as the number of targets is assumed to be constant, the complexity of this task remains essentially constant for each iteration. Because of this same assumption, JPDAF cannot initiate new tracks or terminate existing tracks.

Schulz et al. (2001) use JPDAF on a mobile robot to track people with measurements obtained through two laser scanners. They replaced the usual Kalman-based posterior estimation of track probabilities by a particle filter based approach, which allows them to take into account non-linear, non-Gaussian movement models. In experiments, the robot Rhino (Buhmann et al., 1995) was able to reliably track a person even in the presence of occlusion and when it was moving at speeds of up to 40 cm/s.

In principle, it would be possible to perform tracking directly with particle filters. But this is difficult, especially for multiple targets, as the target-data associations are unknown.

Vermaak et al. (2005) also attack the target tracking problem with Monte Carlo methods. They propose three algorithms. The first, called *Monte Carlo joint probabilistic data association filter* (MC-JPDAF) is a generalization of the method by Schulz et al. (2001) (mentioned above) to multiple observers and arbitrary proposal distributions. The other two algorithms rely on using particle filters to directly track targets. One is *sequential sampling particle filter* (SSPF) and the other is *independent partition*

[4] "target" here is used in a general sense, i.e. meaning an object of interest

particle filter (IPPF). They are different in the way that data associations are handled. SSPF uses a proposal distribution to propose an association vector and then each track can be treated independently. IPPF assumes that observation associations to individual targets are independent. This essentially means that a single observation can be assigned to multiple tracks. They can show that this assumption greatly reduces the complexity of the problem. The authors performed simulations of challenging tracking problems with three and four slow moving targets. They compared their three algorithms to the standard particle filter. All three outperformed the standard particle filter in terms of tracking precision and computational costs. With increasing difficulty of the association problem, the performance of IPPF degraded due to its independence assumption. MC-JPDAF needed the least number of samples to produce accurate tracks. It should be noted that all three algorithms assume a known, fixed number of targets.

Liang et al. (2008) track a single moving sound source using the data from a sound source localization system based on the crosspower spectrum phase method (cf. CSP in Section 3.1). Using four omnidirectional microphones, the sound localization was capable of providing all three coordinates of a sound source (azimuth, elevation and distance). As there is only one target, the data association problem does not arise. Correctness of the track is estimated through a multiple model Kalman filter (MMKF) approach. With this method, input data are processed by multiple Kalman filters in parallel (each with a different motion model) and the most likely output is determined through a probability model of the motion models. Three speaker motion models were used in this case, representing a stationary speaker, linear (constant speed) and accelerated (constant acceleration) motion. In this sense, the method proposed by Liang et al. (2008) performs data smoothing rather than tracking, as it assumes a single target which has to be audible at all times. It could be shown that MMKF produces smaller errors in smoothing, than a single Kalman filter approach would.

An altogether different method from the ones described until now is presented by Fritsch et al. (2003). In this case, data from camera-based face detection and laser-based leg detection were used as input data to a tracking system. Sensor fusion and target tracking was achieved through a procedure called multi-modal anchoring. Here, features extracted from the individual sensor data are linked by an internal symbolic representation. A so-called *anchor* is used as a link between the *Symbol system* (representing abstract concepts like e.g. *face*) and the *Perceptual system* (representing sensor-level data like e.g. position or size) (Coradeschi and Saffiotti, 2001). Fritsch et al. (2003) extend this to multiple modalities through composition relations and composite objects as well as composition, motion and fusion models. The system was later extended by binaural sound source localization (Fritsch et al., 2004).

Chapter 4

Biologically Inspired Sound Source Localization

In this chapter, we present the sound source localization-related algorithms developed for this dissertation. Section 4.1.1 presents the interaural time differences-based localization method. This was previously published in Calmes et al. (2007a). Section 4.2 describes the algorithm for evaluating interaural level differences. In Section 4.3, an artificial barn owl ruff is presented as well as first attempts at combining ITD and ILD-based sound source localization. Section 4.4, finally, describes a spatial attention module, capable of modulating auditory attention using a cue signal.

4.1 Sound Source Localization Using Interaural Time Differences

4.1.1 Mathematical Model

The method for sound source localization by interaural time differences is derived from the dual delay-line algorithm published by Liu et al. (2000). Figure 4.1 shows the so-called dual delay-line structure, which is in essence an implementation of the Jeffress model (Jeffress, 1948). The axonal delays are represented by the triangular delay elements whereas the coincidence detector neurons are depicted by circular elements. Note that Figure 4.1 shows only one of many frequency bands.

The basic unit of computation of the model is a timeframe from a digitized stereo audio signal encompassing N samples per channel. The first step is to transform the current timeframe with index n (possibly zero-padded to the FFT size $M \geq N$) to the frequency domain using a short-time Fourier transform

$$x_{Ln}(k) \iff X_{Ln}(m), \tag{4.1a}$$

$$x_{Rn}(k) \iff X_{Rn}(m), \tag{4.1b}$$

$$k = 0, \ldots, M-1;\ m = 0, \ldots, M/2-1.$$

Next, delaying in the frequency domain has to be performed. The complex Fourier points for each channel and frequency are delayed by

$$\tau_i = \frac{\mathrm{ITD}_{\max}}{2} \sin\left(\frac{i}{I-1}\pi - \frac{\pi}{2}\right), \tag{4.2}$$

$$i = 0, \ldots, I-1,$$

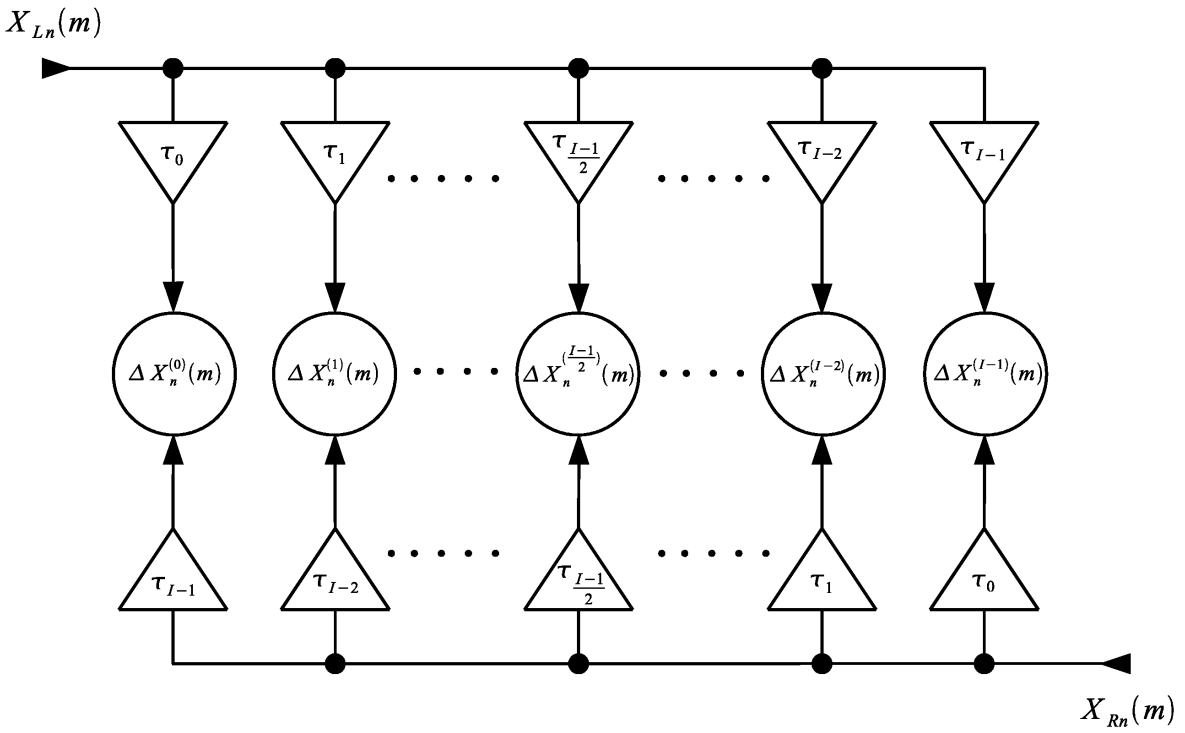


Figure 4.1: Dual delay-line structure. $X_{Ln}(m)$ and $X_{Rn}(m)$ represent the spectral values of the m-th frequency band of the left and right signals for timeframe n after Fourier transformation. τ_i represent the axonal delay elements and $\Delta X_n^{(i)}(m)$ represent the coincidence detector neurons.

where $\mathrm{ITD}_{\max} = b/c$ is the highest possible interaural time difference given the microphone distance b (20.5 cm are used here) and the speed of sound c (340 m/s). This yields an $\mathrm{ITD}_{\max}$ of 602 µs. $\mathrm{ITD}_{\max}$ corresponds to 90° in azimuth. Thus, by Eq. (4.2) the azimuthal space is partitioned into I sectors of equal size. Azimuths ranging from $\alpha = -90°$ to $+90°$ are considered. Negative values indicate a sound source positioned in the left hemisphere, while positive azimuths point to a sound source in the right hemisphere.

The actual delaying is performed by adding a phase shift corresponding to the delay τ_i to the original phase of the input signals in each frequency band

$$X_{Ln}^{(i)}(m) = X_{Ln}(m)e^{-j2\pi f_m \tau_i}, \tag{4.3a}$$

$$X_{Rn}^{(i)}(m) = X_{Rn}(m)e^{j2\pi f_m \tau_i}, \tag{4.3b}$$

$$i = 0, \ldots, I-1;\ m = 0, \ldots, M/2-1,$$

where M is the FFT size, τ_i specifies the delay in seconds, $f_m = mf_s/M$ is the center frequency of the m-th frequency band and n is the number of the current timeframe.

As this operation is performed in the frequency domain, subsample accuracy is achieved without any additional effort, because interpolation between samples is done implicitly. In the time domain, interpolation would have to be done explicitly in order to shift the signals by an amount smaller than one sample (see Section 2.2). Equation (4.2) may seem unintuitive as it allows for negative delays. However, this has no practical implications due to the periodic nature of the discrete Fourier transform and thus can be safely ignored, as long as the delayed signals are not meant to be transformed back into the time domain.

The delays are symmetric around the 0-valued delay $\tau_{(I-1)/2}$. For the left channel, the negative internal delays are situated to the left of the midline in the dual delay-line structure, while positive internal delay values are situated to the right. For the right channel, the reverse is true. τ_0 has the value $-\mathrm{ITD}_{\max}/2$, while τ_{I-1} has the value $\mathrm{ITD}_{\max}/2$. Thus, coincidence detection for external negative delays (sound sources positioned to the left) happens at the right side in the delay-line structure, while coincidence detection for positive external delays (sound sources positioned to the right) happens at the left side in the dual delay-line structure. As can be deduced from Figure 4.1, the delay value for the right channel corresponding to the point i in the dual delay-line would be τ_{I-i-1}, whereas in Eq. (4.3b), $-\tau_i$ is used. It can easily be shown by substituting $I-i-1$ in Eq. (4.2), that $\tau_{I-i-1} = -\tau_i$.

The external time delay at the point i in the dual delay-line structure, which is compensated by the internal time delay thus corresponds to

$$\mathrm{ITD}_i = -[\tau_i - (-\tau_i)] = -2\tau_i. \tag{4.4}$$

As the azimuth space has been partitioned into I sectors by Eq. (4.2), there exists a linear relationship between the azimuth α_i and the position i of a coincidence detector element

$$\alpha_i = 90 - \frac{i}{I-1}180. \tag{4.5}$$

In the ideal case, the time domain signals from both channels are identical except for a time difference. This results in identical amplitude spectra in the frequency domain, whereas the time difference leads to a difference in the phase spectra for both channels. Equations (4.3a) & (4.3b) induce a phase change in the left and right channels, respectively for every point i in the dual delay-line. At a given point (the coincidence location), the left and right phase spectra will be identical (in the ideal case) or at least minimally different (for signals recorded by the microphones). To detect that point, first a coincidence map is built with

$$\begin{aligned} \Delta X_n^{(i)}(m) &= |X_{Ln}^{(i)}(m) - X_{Rn}^{(i)}(m)|, \\ i = 0, \ldots, I-1;\ m &= 0, \ldots, M/2-1. \end{aligned} \tag{4.6}$$

This coincidence map shows the distribution of the response amplitude as a function of both the position in the dual delay-line structure (corresponding to azimuth) and frequency. Since we create a three-dimensional structure, we refer to the map as a three-dimensional coincidence map. As an example, Figure 4.2 shows the ideal case derived from computer-generated inputs. As the azimuth α_i depends linearly on the index i in the delay-line structure, the i has been replaced by the corresponding azimuth in Figure 4.2 for clarity. The map has a frequency independent minimum at or close to $-24.5°$ of azimuth, which corresponds to the time shifts in the input signals. There are more response minima (caused by phase ambiguities), especially in the high-frequency region, but these minima change their location with frequency. Minima occurring in an azimuth-independent manner over the whole frequency range for a given i specify coincidence location. This is where the method deviates from that described in (Liu et al., 2000), where the indices of the minima for each frequency band are computed from the map:

$$i_n(m) = \arg\min_i [\Delta X_n^{(i)}(m)] \tag{4.7}$$

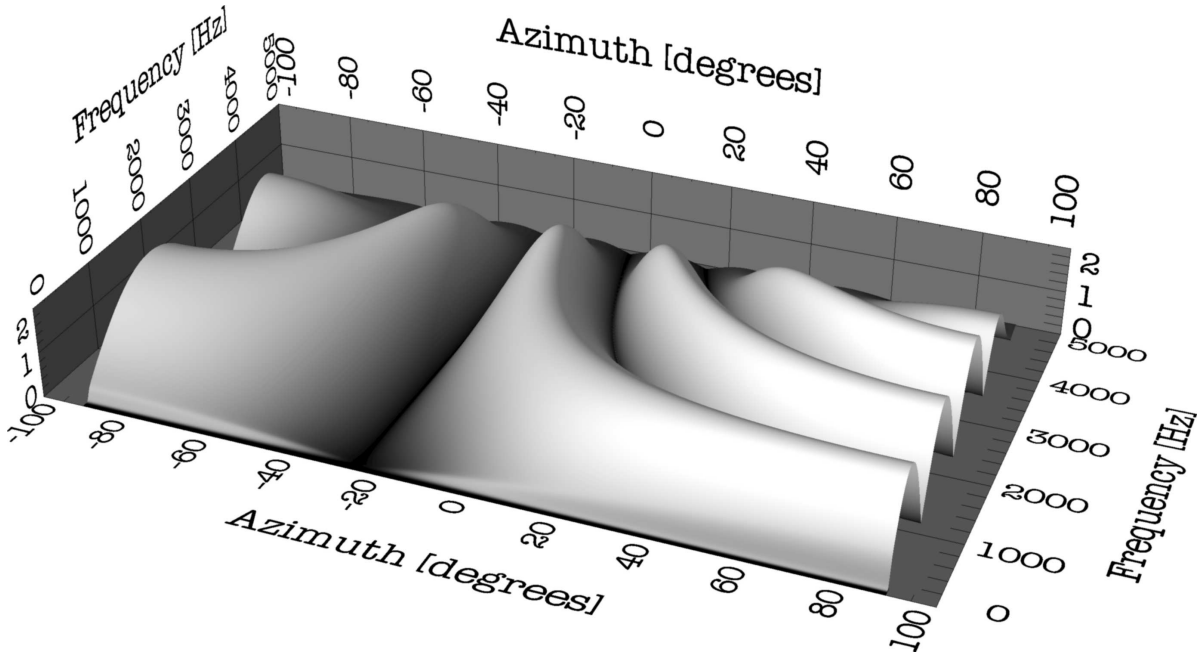


Figure 4.2: Example of a 3D coincidence map. It is computed by generating two unit samples (discrete-time Dirac Delta function) in software and using them as input to the system. The left channel leads the right channel by an inter-channel time difference of four samples corresponding to an azimuth of $-24.5°$ (sampling frequency is set to 16 kHz, microphone distance is set to 20.5 cm). The algorithm returned a value of $-24.5°$ which corresponds to the frequency-independent minimum at $-24.5°$ azimuth in the graph. Z-axis values denote dissimilarity between left and right signals. The lower the value, the higher the similarity at that azimuth. The map was computed using the data from the first timeframe of the input signals.

The coincidence map is integrated over time by performing a running average with time constant β on the coincidence maps computed for all timeframes:

$$P_n(i,m) = \sum_{k=0}^{n} \beta^{n-k} \Delta X_k^{(i)}(m), \tag{4.8}$$
$$i = 0, \ldots, I-1;\ m = 0, \ldots, M/2-1.$$

This again is in contrast to the algorithm used in (Liu et al., 2000) where integration over time is done by accumulating the coincidence locations of the minima for each frequency band (here, δ refers to the Kronecker delta function):

$$P_n(i,m) = \sum_{k=0}^{n} \beta^{n-k} \delta(i - i_k(m)). \tag{4.9}$$

In our algorithm, integration over frequency is performed by summing up the coincidence map at the current timeframe index n over all frequency bands

$$H_n(i) = \sum_{m=0}^{M/2-1} P_n(i,m), \tag{4.10}$$
$$i = 0, \ldots, I-1.$$

(Liu et al., 2000) describe two methods for frequency integration. The first (called "direct" method) is the same as Eq. (4.10) above. The second method (called "stencil"

filter) is more complex. While the "direct" method only sums up coincidence locations over frequency corresponding to a position i in the delay line, the "stencil" filter also takes into account coincidence locations corresponding to phase ambiguities for the index i. This is possible, because the pattern of high-frequency phase ambiguities is unique for each index i. To make this method computationally tractable, a broadband coincidence pattern has to be precomputed, providing the theoretical positions of coincidence locations. As the delay values τ_i vary in a non-uniform manner, the coincidence pattern varies with the index i, thus requiring storage space for I different patterns. To circumvent this disadvantage, Liu et al. (2000) chose to use uniform delays, thus requiring only one precomputed theoretical broadband coincidence pattern. A sliding window, centered at the position i in the dual delay-line provides the coincidence positions needed for frequency integration. The tradeoff of this method is that, with uniform delays, the angular resolution across azimuth positions is not constant. Positions close to the midline will have higher angular resolution than more lateral positions, thus requiring a higher number I of coincidence detectors to achieve a resolution equivalent to that obtainable by the "direct" method. We chose not to implement the "stencil" filter because we wanted to keep a constant angular resolution and because the results obtained by using Eq. (4.10) were sufficient for our purposes.

The final localization function is obtained by normalizing the function $H_n(i)$ at the current timeframe index n to the range $0 \ldots 1$ and by transforming the minima into maxima by subtracting from 1

$$\mathrm{Loc}_n(i) = 1 - \frac{H_n(i) - \min(H_n(i))}{\max(H_n(i)) - \min(H_n(i))}, \qquad i = 0, \ldots, I-1. \tag{4.11}$$

To determine the points of coincidence location, the indices i_n^{MAX} of the local maxima of $\mathrm{Loc}_n(i)$ have to be found satisfying the following properties:

$$\mathrm{Loc}_n(i_n^{\mathrm{MAX}}) \geq 0.5, \tag{4.12a}$$
$$\mathrm{Loc}_n(i_n^{\mathrm{MAX}}) > \mathrm{Loc}_n(i_n^{\mathrm{MAX}} - 1) \tag{4.12b}$$
$$\mathrm{Loc}_n(i_n^{\mathrm{MAX}}) > \mathrm{Loc}_n(i_n^{\mathrm{MAX}} + 1) \tag{4.12c}$$

The threshold Eq. (4.12a) is necessary to suppress unwanted side peaks in the localization function which can result from high-frequency phase ambiguities. A value of 0.5 proved to be quite effective in suppressing side peaks not attributable to real sound sources. From i_n^{MAX}, the azimuth to the corresponding sound source can be computed with the help of Eq. (4.5).

The final output of the localizer is an array of pairs of azimuth with corresponding peak height

$$\left(\alpha_{i_n^{\max}},\ \mathrm{height}_{i_n^{\mathrm{MAX}}}\right) = \left(90 - \frac{i_n^{\mathrm{MAX}}}{I-1}180,\ \mathrm{Loc}_n(i_n^{\mathrm{MAX}})\right). \tag{4.13}$$

An implementation using Eqs. (4.7), (4.9) & (4.10) was initially tried, but this resulted in strong outliers at $\pm 90°$ with noisy or non-broadband signals. Finally the method of integrating the complete three-dimensional coincidence map (instead of coincidence locations) over frequency (Eqs. (4.6), (4.8) & (4.10)) was chosen and this solved the problem.

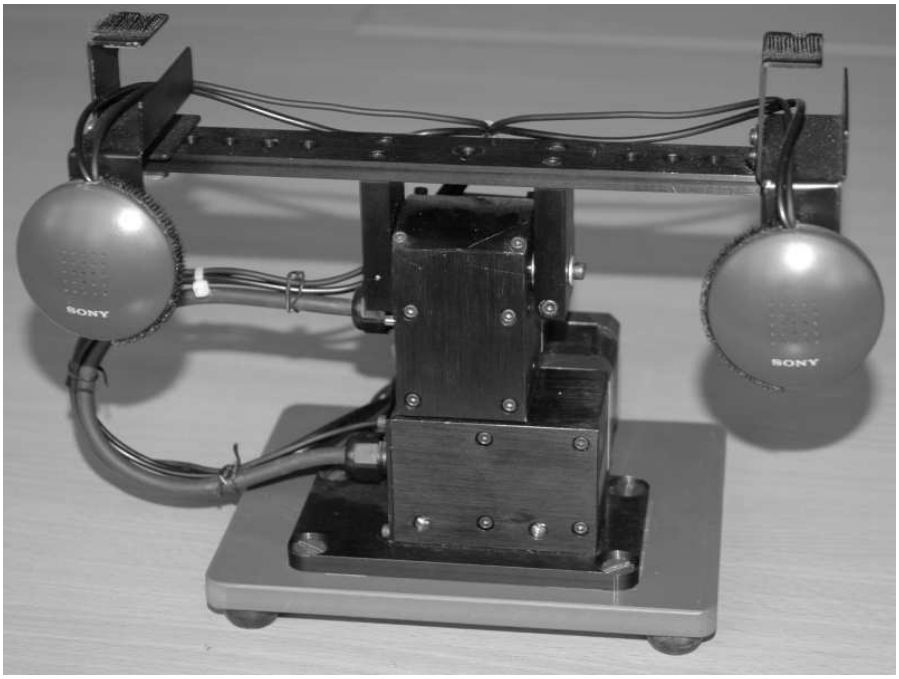


Figure 4.3: Pan-tilt unit with microphones.

4.1.2 Results

Hardware setup

Throughout the experiments — with the exception of the pan-tilt unit (PTU) — standard, readily available, off-the-shelf components were used. The microphones were two Sony ECM-F8 omnidirectional electret condenser microphones (frequency range: $\approx$ 50 Hz – 12 kHz), connected to two preamplifiers built around an LF351N op-amp (frequency range: $\approx$ 50 Hz – 20 kHz; obtained as kit from an electronics supplier). The preamplifiers were connected to the line-in input of the on-board sound chip of a standard PC.

The microphones were mounted on a Directed Perception PTU-D46-17 pan-tilt unit with a separation of 20.5 cm (Figure 4.3), controlled by the same computer which was running the localization algorithm. The angular resolution of the PTU (0.0514°) is one order of magnitude higher than the angular resolution of the sound source localizer (0.5°), ensuring that the microphone assembly is able to pan towards the positions indicated by the algorithm.

All the experiments were conducted in a normal office environment (4.95 m$\times$3.48 m), with background noise from computers and ventilation. The sound source (a Sony SRS-57 loudspeaker; frequency range: $\approx$ 100 Hz – 20 kHz) was placed at a distance of approximately 1 m from the microphone assembly. The loudspeaker output volume was set in such a way that the signals recorded over the microphones (at the 0° azimuth setting) had a maximal amplitude of about -2 dB with respect to the maximal input amplitude of the A/D converters. This ensured a high signal to noise ratio while avoiding clipping in the input signals.

Software configuration

The algorithm was implemented in C++ on a Linux OS. All signal processing was done in software. Whenever possible, the algorithm parameters mentioned in the Liu

et al. (2000) article have been used. Due to hardware and real-time considerations, some parameters had to be changed. The sampling frequency was 16 kHz. Signals were quantized at 16 bits. The FFT size was 2 048 points, yielding a frequency resolution of 7.8125 Hz per frequency band. The system was set up with 361 delay elements per delay line. With this configuration, a linear angular resolution of 0.5° is achieved. A value of 340 m/s was used for the speed of sound. The time-integration constant β from Eq. (4.8) was set to 0.8.

As early versions of the software processed a timeframe in about 60 ms (on an AMD Athlon PC clocked at 1.3 GHz), the timeframe size was set to 62 ms (992 samples at 16 kHz), with no overlap. In this way, real-time operation was achieved. Even though the latest, optimized version of the software completes the computation in less than 2 ms on a newer computer (Intel Core2 Duo, 2.2 GHz), the timeframe size was not reduced, in order to keep the data from later experiments consistent with earlier measurements.

After A/D conversion, the timeframes were filtered with a 12th order Butterworth band-pass filter (passband approx. 100 Hz - 4 000 Hz) and weighted with a Hann window. The 992 samples were then zero-padded to the FFT size of 2 048 points.

In the case of click signals, a simple signal detector was used to prevent the algorithm from producing azimuth estimations corresponding to background noise. Before the experiment, 2 s of background noise were recorded. As the click was very short, it could be that the variance of a whole timeframe containing a click was still quite low. Therefore, for every timeframe, the mean of the subframe (32 samples) variances was computed. The threshold was set to 1.7 (value determined empirically) times the mean of the individual timeframe values. During the experiment, a timeframe was accepted as containing a signal if the mean of the subframe variances was above the threshold computed from background noise. In that case, localization computations were performed, otherwise the timeframe was dropped. The sole purpose of this signal detection system was to ignore timeframes that did not contain samples belonging to the click stimulus. We did not intend to develop a signal detector suitable for practical applications.

Time was measured by reading out the processor cycle counter at appropriate locations in the program. By subtracting two readouts enclosing a part of the code which is to be timed and dividing by the clock frequency, accurate timing information was obtained (within the limits of a non real-time multitasking operating system).

To obtain different sound source positions for the experiments, it was not the source loudspeaker that was moved with respect to the microphone assembly, but rather the microphone assembly was rotated with respect to the loudspeaker. This was done for practical reasons, as the spatial requirements for moving the loudspeaker in an arc from $-70°$ to $+70°$ around the microphones exceed the space available in most office environments. In the following, for simplicity, it will still be referred to a variation of the azimuth of the sound source, but it should be kept in mind that it were actually the microphones that were panned while the sound source position remained fixed.

Experiments

Four different types of tests of the algorithm were performed. Tests using the computer generated signals showed the general correctness of the implementation of the algorithm. Tests using signals transmitted via loudspeaker and recorded in open loop conditions, demonstrated the robustness of the algorithm in real situations. Additionally, the performance of the algorithm was assessed in closed loop conditions on a robotic pan-tilt unit. Finally, to find an explanation for the poor performance with low-frequency, narrowband signals, room simulations were conducted.

Table 4.1: Signal types used in the experiments

Signal type	frequency range
noise	broadband
click	broadband
bandpass noise	1 kHz – 4 kHz
bandpass noise	500 Hz – 4 kHz
bandpass noise	100 Hz – 1 kHz
sine	1.5 kHz
sine	1 kHz
sine	500 Hz

To assess the influence of spectral content of a signal on the localization system, stimulus signals with increasing bandwidth were chosen, from pure tones to broadband noise and clicks (Table 4.1). Because of the bandpass filter employed, broadband here actually refers to a frequency range of 100 Hz–4 kHz.

It should be noted that multiple azimuths can be returned by the system (especially for sine stimuli), although only one stimulus source is present. In this case, the position with maximal height was selected from the azimuths detected in Eq. (4.13). For sine stimuli, peaks not attributable to the real source position corresponded to phase ambiguities and could in some cases produce the maximal peak height. This resulted in the system localizing phase ambiguities instead of the real source position. For non-sine stimuli, spurious peaks above the threshold imposed by Eq. (4.12a) were sometimes detected. They were either caused by transient environmental noise (e.g. door closing), or they corresponded to phantom sources caused by the room acoustics. In either case they never produced the azimuth with maximal peak height, so that the azimuth estimation from the sound source localizer corresponded to the real source.

Tests using computer-generated signals directly The coincidence map in Figure 4.2 was created with two time-shifted unit samples (the discrete-time version of the Dirac Delta function) generated in software. The frequency-independent minimum at $-24.5°$ represents the simulated position of the sound source. The output of the algorithm indeed yielded a sound-source position of $-24.5°$. In Figure 4.4 a similar coincidence map was created, but with broad-band noise as stimulus. Again, one minimum, occurring at $-24.5°$ was clearly frequency independent, while all other minima changed their position with frequency. Although the minimum was less well defined than in Figure 4.2, the algorithm had no problem finding the position of the sound source.

Tests with many different ITDs and signal types were conducted with computer-generated signals. The algorithm always found the right peak corresponding to the original ITD with an error smaller than $1°$. Moreover, the localization estimate remained stable during the whole experiment. Even for sinusoidal stimuli, the correct ITD could be extracted. However, as expected for this type of input, for frequencies above ≈ 830 Hz, virtual peaks corresponding to phase ambiguities were also detected.

Tests using microphone signals: open loop experiments In the open loop[1] tests, the computer generated sound was transmitted via a loudspeaker and recorded by a pair of microphones. Stimulus presentation was continuous (with the exception of

[1]Open loop means no feedback, i.e. the sensor input (sound source azimuth) is not fed back to the actuator (PTU).

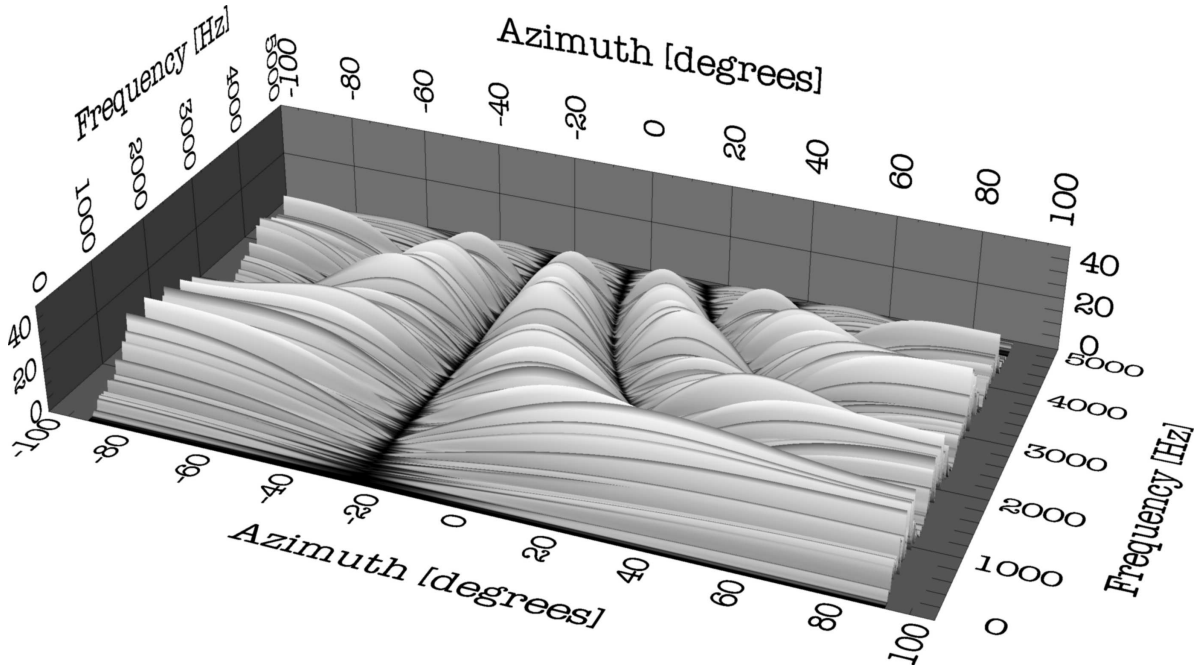


Figure 4.4: Example of a 3D coincidence map of the first timeframe of a broadband noise signal with a time difference of 4 samples corresponding to an azimuth of −24.5°. The algorithm computes the correct azimuth of −24.5°. Sampling rate is 16 kHz, inter-microphone distance 20.5 cm. Z-axis represents dissimilarity as in Figure 4.2

the click stimulus). The sound-source azimuth remained fixed during a localization run. The output of the algorithm was not fed back to PTU. Recorded data included source position, number of detected azimuths and the pairs of azimuth with corresponding peak heights from Eq. (4.13).

As an illustration for a coincidence map of real signals, Figure 4.5 shows an example generated by the third timeframe of a broadband noise stimulus recorded through the microphones. The source was positioned at an azimuth of −20° with respect to the microphone assembly. Whereas in simulations (cf. Figure 4.4), there is a clear, frequency-independent minimum at the source azimuth, the frequency-independent minimum in Figure 4.5 is much more diluted.

Figure 4.6 shows the azimuths as a function of time, for three different signal types and a source position of −60°. The algorithm was able to precisely localize the source at −60° for a broad-band stimulus. When the bandwidth is limited to the 100 Hz – 1 kHz range, a systematic localization error of some 20° occurred throughout the run (the outlier after the 3 s mark is caused by a transient noise in the environment as e.g. a door slam). In a similar way, the algorithm gave a stable estimate when the stimulus was a 500 Hz sinusoid. However, it can be seen in Figure 4.6, that at about +65°, the localization error is much larger. High-frequency phase ambiguities arise at wavelengths smaller than twice the microphone distance. With a microphone distance of 20.5 cm, this would be the case for frequencies above 830 Hz (speed of sound 340 m/s). Thus, the mislocalization of the 500 Hz sinusoid (wavelength 68 cm) cannot be caused by the system locking onto a high-frequency phase ambiguity.

Figure 4.7 shows the averages over five runs for random noise, 100 Hz–1 kHz bandpass noise, 1.5 kHz sine (along with the first phase ambiguities depicted as open circles) and 500 Hz sine stimulus types (80 timeframes per run and azimuth for each signal type). Table 4.2 shows the minimum, maximum, and the mean of detected azimuths alongside

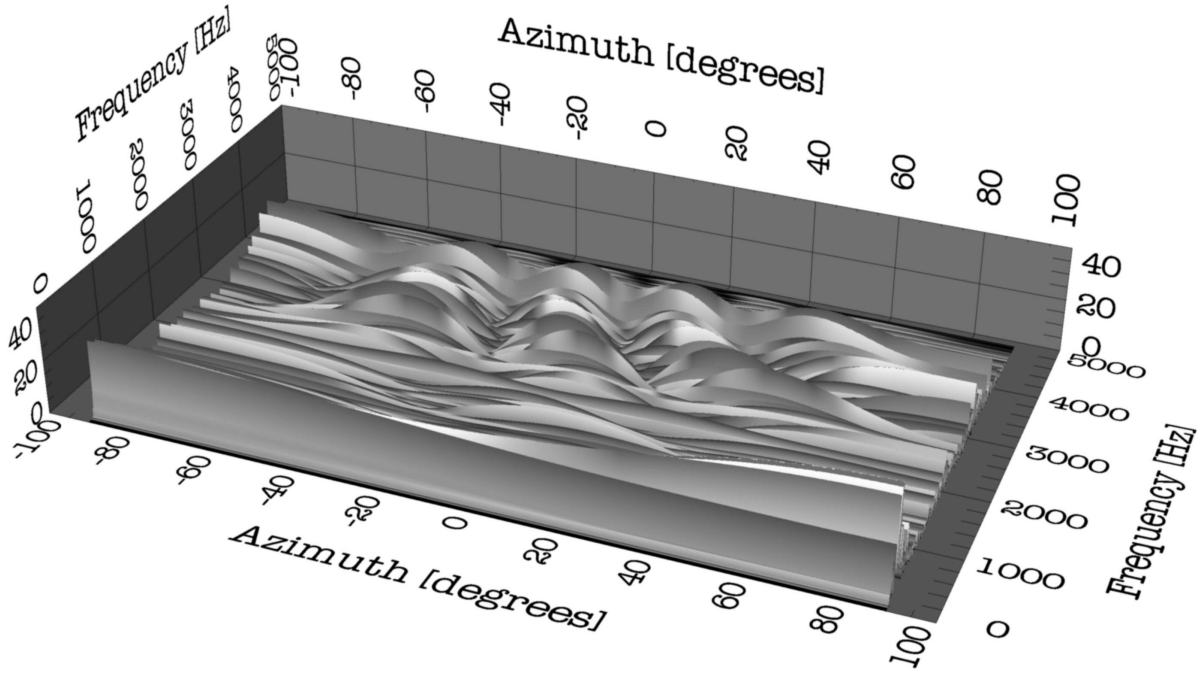


Figure 4.5: Example of a 3D coincidence map of a real signal (broadband noise, third timeframe), recorded with microphones. Source position was −20°. The value returned by the algorithm is −20°. Sampling rate is 16 kHz, inter-microphone distance 20.5 cm

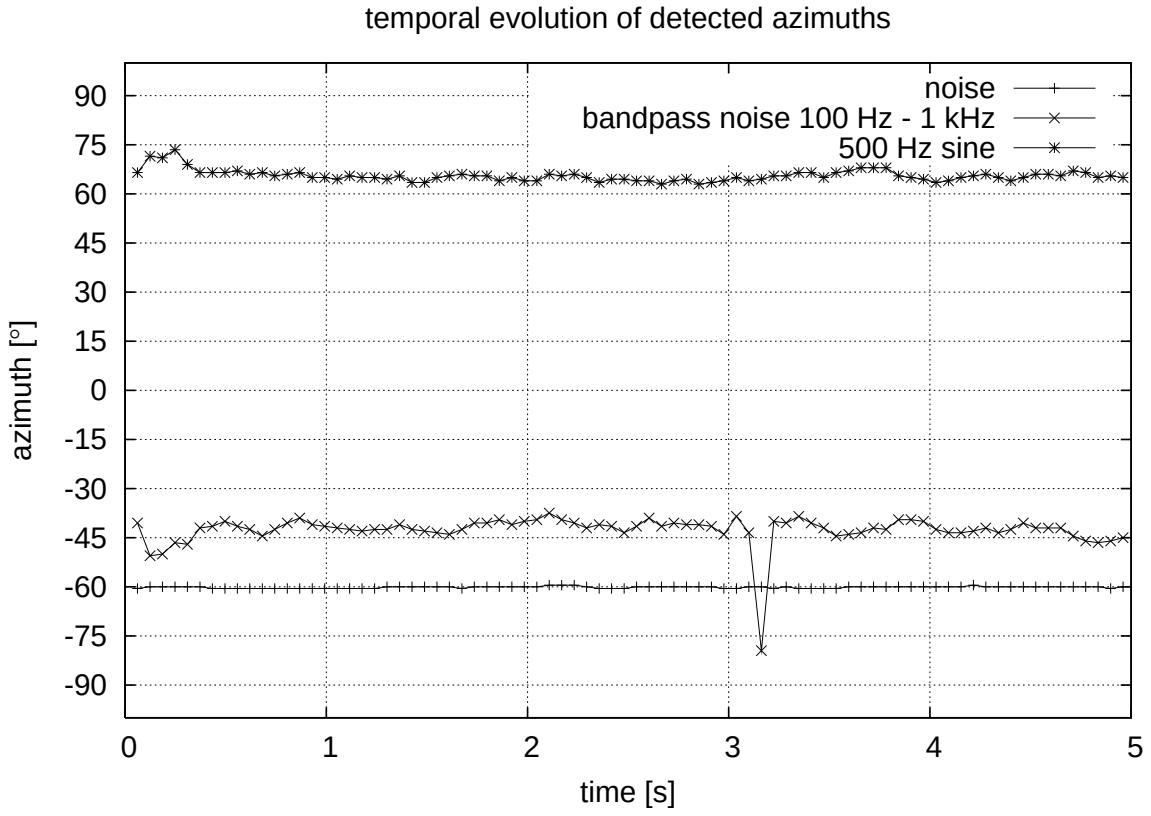


Figure 4.6: Temporal sequence of estimated azimuths for three different signal types measured at a source position of −60°. One 80-timeframe (≈ 5 s) run per signal.

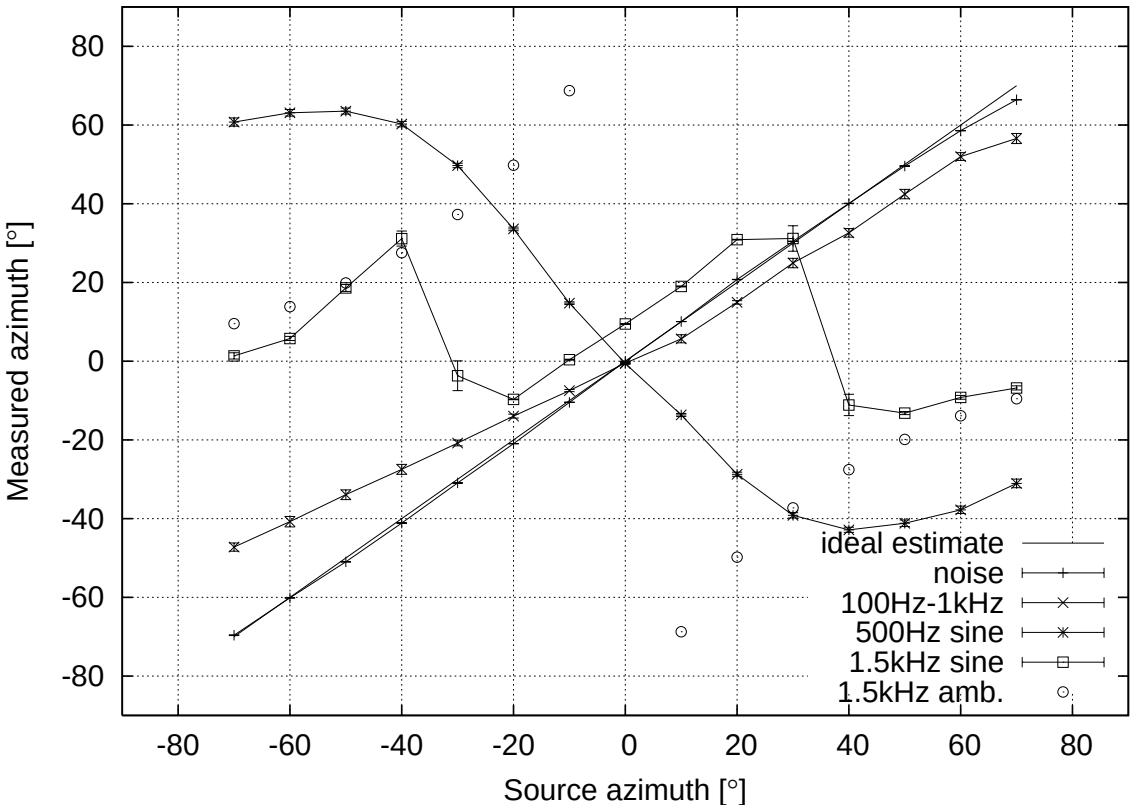


Figure 4.7: Averages of measured azimuths for four different signal types (5 runs per signal, 80 timeframes per run and azimuth). Error bars indicate 99% confidence interval for the given source position. Note the decrease in accuracy with decreasing bandwidth. For every new localization run, the source is positioned at a different azimuth ($-70°$ to $+70°$ in $10°$ steps). Open circles indicate the expected locations of phase ambiguities for the 1.5 kHz sine.

the standard deviations for all the signal types used in the experiments at the $-70°$, $0°$ and $+70°$ source positions (5 runs, 80 timeframes per run and azimuth for each signal type). In these representations, the impressions from the examples shown in Figure 4.6 confirm themselves: the algorithm performed almost perfectly for broadband noise, and very well also for clicks, although with clicks some variation may be seen (Table 4.2). High-frequency noise (1 kHz – 4 kHz) could be localized as well as broadband noise, but problems arose with low-frequency, bandpassed noise as manifested by increased standard deviations.

The localization of sinusoids having a frequency of 1.5 kHz shows a periodic curve (Figure 4.7). For azimuths close to zero the localization of sinusoids is quite acceptable, but for larger (smaller) stimulus positions a jump occurs. This is a consequence of the algorithm detecting the real peak for small azimuths and virtual peaks (offset by 1 or more periods) for larger azimuths. A similar observation was made with 1 kHz tones (Table 4.2 shows only data for $\pm 70°$ azimuth). This explains the large errors seen in Figure 4.7 and Table 4.2 for these frequencies and certain azimuths.

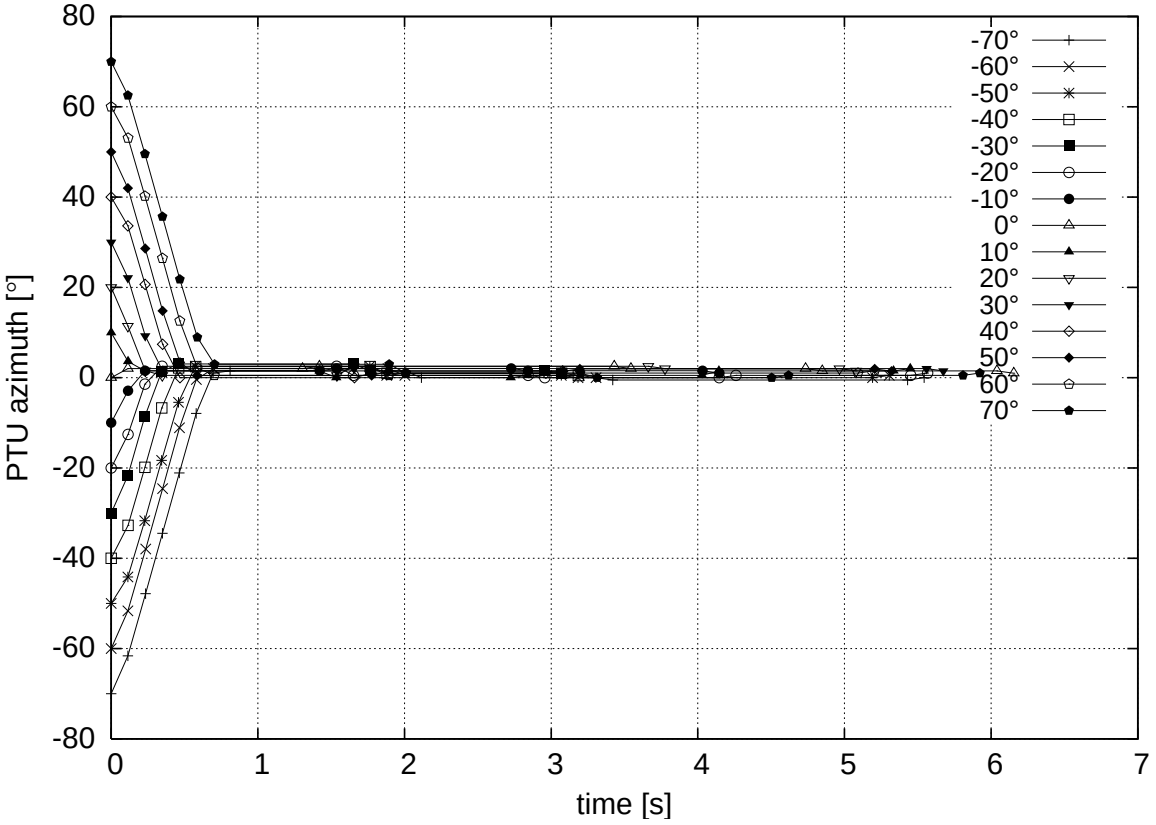


Figure 4.8: PTU tracking noise (single run). Inset indicates initial position of sound source.

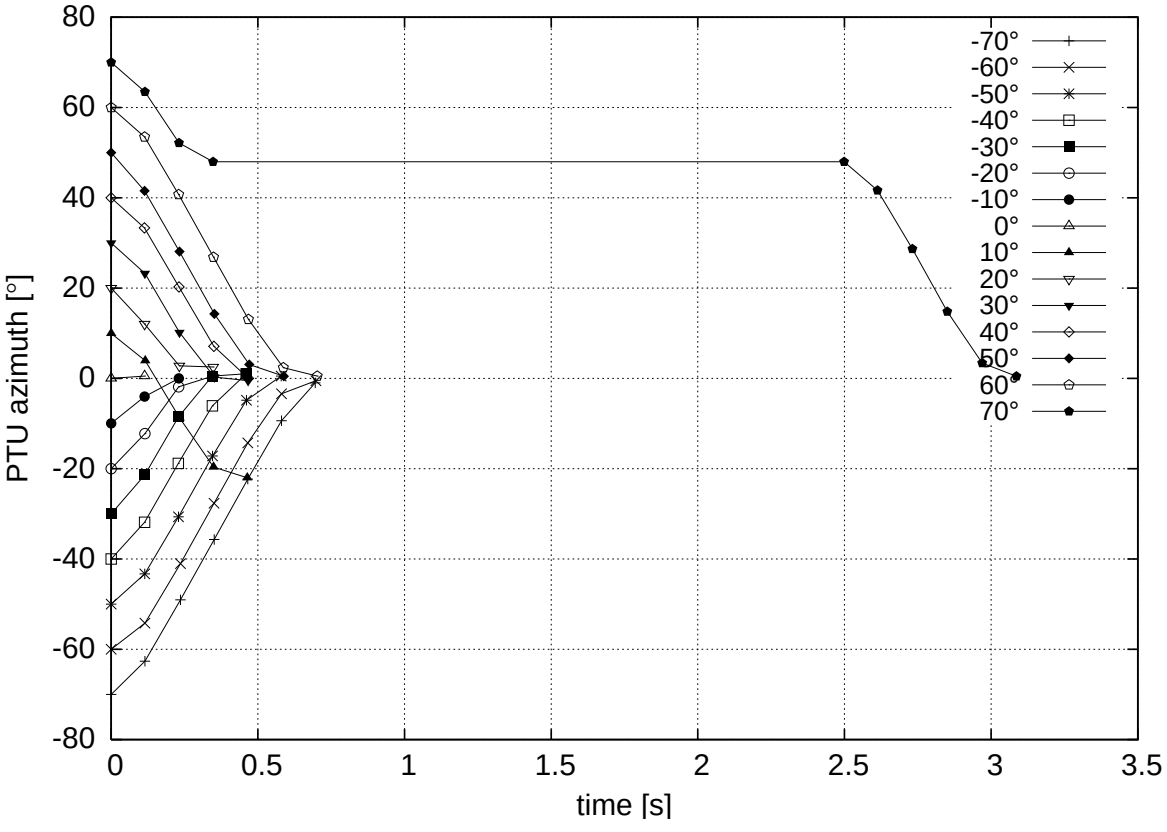


Figure 4.9: PTU tracking a click (single run). Click duration is about 180 µs.

Table 4.2: Open loop experiments results (5 runs and 80 timeframes per run and azimuth).

Sig. type	source	meas. az.			
		min	max	mean	σ
Noise	-70.00	-71.50	-66.00	-69.61	0.57
	0	-0.50	0.00	-0.04	0.14
	70.00	65.50	68.00	66.42	0.37
Click	-70.00	-70.50	-69.50	-70.00	0.45
	0	-0.50	0.00	-0.40	0.20
	70.00	-5.00	68.00	53.20	29.10
Noise 1 kHz–4 kHz	-70.00	-70.50	51.50	-68.25	6.92
	0	-30.50	0.50	-0.17	2.13
	70.00	66.00	69.00	67.19	0.43
Noise 500 Hz–4 kHz	-70.00	-71.50	33.50	-69.24	5.65
	0	-0.50	0.50	-0.01	0.07
	70.00	64.50	69.50	66.51	0.49
Noise 100 Hz–1 kHz	-70.00	-90.00	0.00	-47.16	8.30
	0	-4.50	3.00	-0.50	1.00
	70.00	0.00	89.00	56.55	10.34
Sinusoid 1500 Hz	-70.00	-90.00	4.50	1.52	5.12
	0	8.00	11.00	9.48	0.97
	70.00	-8.50	-4.00	-6.99	1.56
Sinusoid 1000 Hz	-70.00	-47.50	75.00	-30.34	38.16
	0	8.50	17.00	11.75	2.24
	70.00	-60.00	52.00	46.27	22.17
Sinusoid 500 Hz	-70.00	0.00	70.50	61.09	4.87
	0	-5.00	8.00	-0.60	1.73
	70.00	-87.00	0.00	-31.29	6.76

Tests using microphone signals: closed loop experiments During the closed loop[2] experiments, the algorithm produced an estimate of the sound-source position (Eq. (4.13)) and transmitted this to the PTU that had to rotate towards that position within a time limit of about 5 s (in the following, "run" refers to this time period). As long as the PTU was moving, sound source localization was suspended in order to avoid confusion from motor noise, but resumed after the panning movement ceased. This was done by ignoring timeframes during PTU movement, so that the algorithm would only "see" timeframes during which no movement took place. From the viewpoint of the sound source localization system, a single closed loop experiment is actually a sequence of open loop experiments. Nevertheless, the whole system consisting of sensor (sound localizer) and actuator (PTU) can be considered as a closed loop controller due to the sensory feedback to the PTU, which is why we refer to these experiments as "closed loop" in the following.

Stimulus presentation for the non-click signals was again continuous. Thus, the azimuth of the sound source changed during a run. In addition to the data described in the open loop experiments section, PTU positions with corresponding timestamps were recorded. Time 0 was set to the moment the first pan command was issued to

[2]Closed loop means feedback, i.e. the sensor data (sound source azimuth) were used to control the actuator (PTU).

Table 4.3: Closed loop experiments results (5 runs)

Signal type	end positions			
	min	max	mean	σ
Noise	-0.51	2.01	0.90	0.57
Click	-65.52	16.51	-0.77	8.22
Noise 1 kHz–4 kHz	-2.01	2.52	0.25	0.68
Noise 100 Hz–1 kHz	-93.09	26.02	2.19	14.01

the unit. PTU positions were sampled from this moment on until the estimated source position was reached, by continuously requesting the current position from the PTU controller. As can be deduced from Figure 4.8, Figure 4.9 and Figure 4.10, the PTU could provide its current position approximately every 0.1 s. In order to let the motor noise reverberations die out, localization was only resumed approximately 0.8 s (value determined empirically) after movement stopped. Due to the time-measurement method employed (see Section 4.1.2), the software only checked how much time had passed after the PTU stopped. This explains the intervals longer than 0.8 s between PTU movements and the run times longer than 5 s in the figures. In the case of clicks, the presentation of the stimulus happened some time after the experiment started. As the signal detector ignored every timeframe before the click, the moment 0 of the experiment could be well into the 5 s measurement interval, explaining the shorter run times. The run leads to a fixation, if the source moves towards 0 azimuth and stays there. The localization precision was estimated by averaging the end positions of several runs. The PTU was able to move the sound-source toward 0 azimuth independently of the starting position when the stimulus was broad-band noise. This situation is shown in Figure 4.8. The data points indicate PTU position and not the azimuth estimates from the localization algorithm. The standard deviation at the end of the run is only slightly larger than the spatial resolution of the algorithm. In general, the localization of the click signals was excellent (Figure 4.9). However, in some cases larger errors occurred and were not corrected throughout a run, leading to a wrong fixation. This becomes also manifest in the relatively large standard deviation for click signals as shown in Table 4.3, which depicts the minimum, maximum, mean and standard deviations of the end positions for the tested stimulus types over 5 runs. These outliers are caused by problems in the signal detector. Usually, only one click was presented for a given start position. However, if the signal detector decided to present a spurious transient (by e.g. a door slamming shut) to the algorithm, a second click was presented (cf. starting position of 70° in Figure 4.9). The panning movement starting at about 2.5 s was caused by the second click in an attempt to bring the PTU towards 0°. With low-frequency noise (100 Hz – 1 kHz), an increased standard deviation is seen (Figure 4.10 and Table 4.3). Interestingly, signals with a starting position on the left were mislocalized to the right and vice versa.

Room simulation tests To test the algorithm further in different sound field conditions and to learn more about the strong deviations for low-frequency bandpass stimuli (100 Hz–1 kHz, Figure 4.7), room simulations were performed. The simulated, empty room consisted of six surfaces (floor, ceiling, walls) and had the same dimensions as the room in which the real experiments took place. Receiver position and configuration, as well as sound source position were also the same. In addition to the sound source distance of 1 m, a source distance of 3.5 m was simulated to assess the impact of direct-to-reverberant ratio on the localization estimates. As in the real room, the virtual

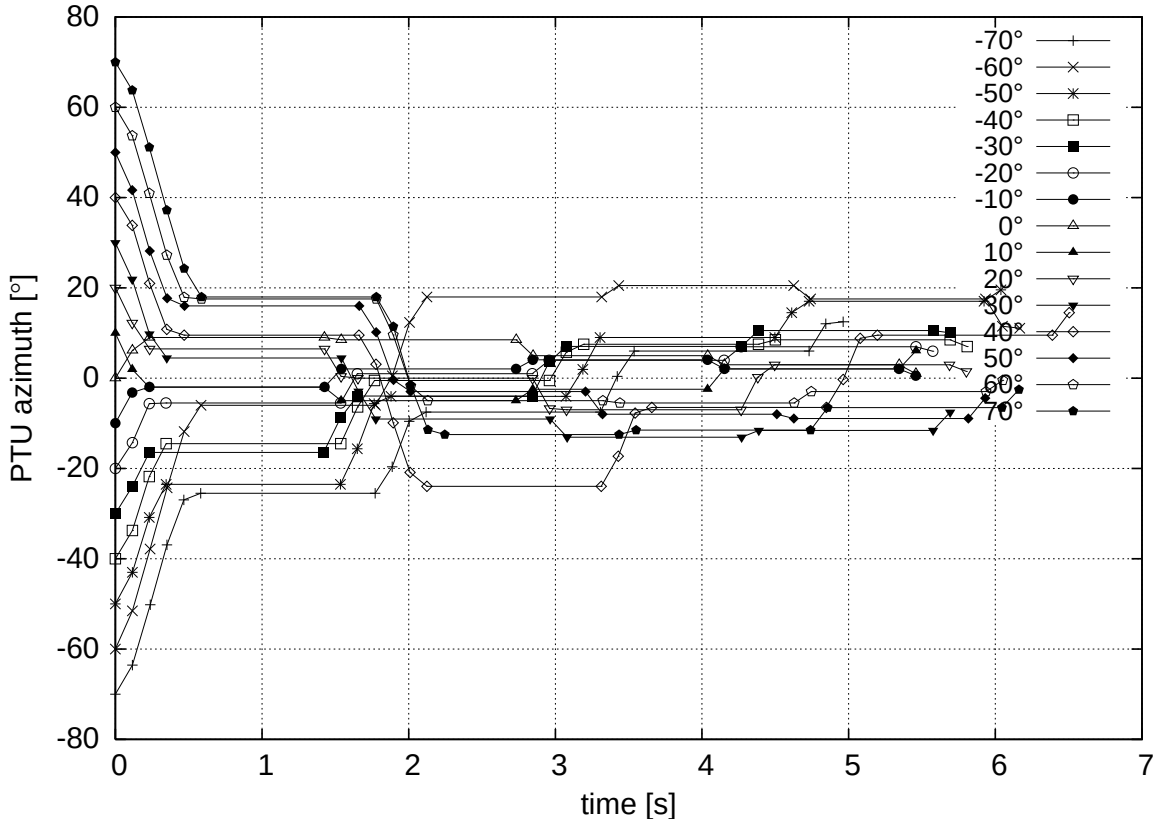


Figure 4.10: PTU tracking bandpass noise (100 Hz–1 kHz; single run). Other conditions as in Figure 4.8.

microphone assembly was rotated whereas the source remained at the same position to generate the 15 different sound source positions ($-70° \ldots +70°$ in 10° steps). Three sets of absorption coefficients were used for all six surfaces of the room:

1. anechoic (total absorption)

2. 50% (50% absorption)

3. unpainted concrete (absorption coefficients corresponding to surfaces made of unpainted concrete)

With the help of the freely available MATLAB program Roomsim, 90 impulse responses (2 source distances, 3 sets of absorption coefficients, 15 source azimuths) were generated. After convolution with the two audio files corresponding to the stimuli used (broadband random noise and 100 Hz – 1 kHz bandpass noise), these yielded 180 audio files which served as input to the algorithm. The actual parameter values used for generating the room impulse responses can be found in the Appendix.

To assess the impact of noise on localization precision, uncorrelated random noise was mixed into the left and right channels by additive superposition at 11 different signal to noise ratios (+30 dB, +20 dB, +10 dB, +6 dB, +3 dB, 0 dB, −3 dB, −6 dB, −10 dB, −20 dB, −30 dB). Although this method of noisification does not represent an accurate simulation of noise in a room, it is useful for measuring the sensitivity of the algorithm to the quality of the input signals. Effectively, as the input signals due to the stimulus are gradually drowned in noise with decreasing signal to noise ratio, the correlation also

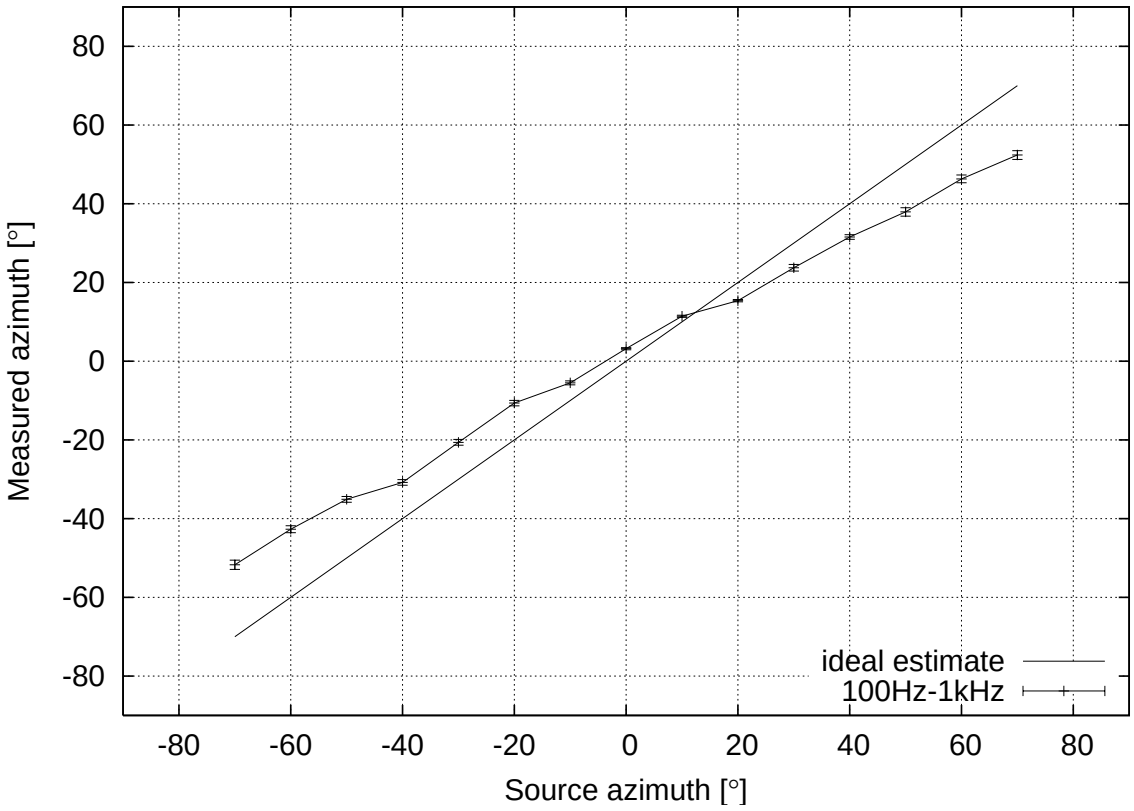


Figure 4.11: Averages of measured azimuths for a bandpass noise (100 Hz–1 kHz) in the room simulation (81 timeframes per azimuth). Source distance was 1 m. Absorption coefficients for all surfaces were set to "unpainted concrete". Signal to noise ratio was +30 dB. Error bars indicate 99% confidence interval.

will decrease. At a given signal to noise ratio, it will no longer be possible to produce a reliable localization estimate.

Except for the timing information, the same data as in the "real world" tests (open loop) was collected. As an example, Figure 4.11 shows the result of a simulation of 100 Hz–1 kHz bandpass noise in the room with the absorption coefficients set to "unpainted concrete" and a signal to noise ratio of +30 dB. Note the similarity with the results in Figure 4.7 for the same type of stimulus in the real room.

Table 4.4 shows the results for the simulations with random noise at the different signal to noise ratios and for source distances of 1 m and 3.5 m. The values were obtained by first computing the difference of the simulated source position to the mean (over 81 timeframes) of the localization estimates for that source position. The mean of the absolute values of the individual errors for each source position yielded the final error value shown in the table.

For the source distance of 1 m, the localization error starts to significantly increase at a signal to noise ratio of −20 dB with absorption coefficients set to "anechoic" and "50%". In the case of the "unpainted concrete" absorption coefficients, a major degradation in localization performance can be observed beginning at a signal to noise ratio of −10 dB.

For the simulations with a source distance of 3.5 m, performance in the "anechoic" case again worsens at an SNR of −20 dB, whereas a slight increase in localization error can already be observed at −10 dB for the "50%" absorption coefficients setting. In

Table 4.4: Room simulation, broadband noise stimulus with varying signal to noise ratio. Values shown are localization errors obtained by computing the means over all source positions of the absolute values of the differences between "real" source position and the mean (over 81 timeframes) of the localization estimates at that source position.

	source dist. 1 m			source dist. 3.5 m		
SNR	anech.	50%	unp. conc.	anech.	50%	unp. conc.
+30 dB	2°	2°	2.1°	2.3°	2.3°	6.5°
+20 dB	2°	2°	2.1°	2.3°	2.3°	6.9°
+10 dB	2°	2°	2°	2.3°	2.3°	10°
+6 dB	2°	2°	2.1°	2.3°	2.3°	10°
+3 dB	2°	2°	2.1°	2.3°	2.3°	10°
0 dB	2.1°	2°	2.1°	2.3°	2.2°	15°
−3 dB	1.9°	2.1°	2°	2.4°	2.2°	16°
−6 dB	2°	2.1°	2.8°	2.2°	2.7°	26°
−10 dB	2.3°	2.7°	18°	2.9°	8.7°	31°
−20 dB	33°	38°	36°	32°	38°	42°
−30 dB	36°	41°	34°	34°	40°	30°

Table 4.5: Room simulation, bandpass noise (100 Hz–1 kHz) stimulus with varying signal to noise ratio. Error values obtained the same way as in Table 4.4.

	source dist. 1 m			source dist. 3.5 m		
SNR	anech.	50%	unp. conc.	anech.	50%	unp. conc.
+30 dB	2°	2.7°	10°	2.3°	6.7°	26°
+20 dB	2.1°	2.8°	10°	2.4°	6.6°	25°
+10 dB	1.9°	2.7°	10°	3.1°	6.7°	25°
+6 dB	2.8°	2.6°	9.9°	3.4°	6.7°	24°
+3 dB	2.6°	2.5°	9°	3.5°	6.8°	25°
0 dB	4.8°	3.7°	9.5°	4.7°	6.7°	28°
−3 dB	4.1°	5.4°	12°	5.6°	11°	28°
−6 dB	7.5°	11°	16°	12°	16°	26°
−10 dB	18°	24°	29°	20°	26°	36°
−20 dB	29°	35°	36°	39°	40°	38°
−30 dB	34°	40°	40°	41°	39°	39°

contrast to the 1 m sound source distance, the localization error is already quite important at high SNR in the "unpainted concrete" case and degrades further beginning at −6 dB.

Table 4.5 shows the simulation results for the 100 Hz–1 kHz bandpass noise. The errors are generally higher when compared to the broadband noise stimulus shown in Table 4.4.

In the "anechoic" case, a major increase in error can already be observed at −10 dB for the source distance of 1 m and at −6 dB for the source distance of 3.5 m. In the "50%" case, this already happens between −3 dB and −6 dB for both distances. The worst case is the one with the "unpainted concrete" absorption coefficients. Although a major increase in error happens at a lower SNR (at around −10 dB for both distances), this is due to the fact that the initial error at +30 dB is already about three times as high when compared to the "anechoic" and "50%" cases (at both distances).

Table 4.6: Room simulation, (Liu et al., 2000) algorithm ("direct" frequency integration method). Broadband noise stimulus with varying signal to noise ratio. Error values obtained the same way as in Table 4.4.

	source dist. 1 m			source dist. 3.5 m		
SNR	anech.	50%	unp. conc.	anech.	50%	unp. conc.
+30 dB	2°	2.3°	10°	2.4°	8.6°	31°
+20 dB	2°	2.4°	12°	2.4°	10°	33°
+10 dB	2.1°	2.4°	16°	2.5°	11°	34°
+6 dB	2.1°	3.8°	22°	2.3°	16°	35°
+3 dB	3.1°	8.2°	28°	3.9°	23°	36°
0 dB	8.7°	16°	29°	10°	27°	37°
−3 dB	24°	27°	32°	23°	33°	38°
−6 dB	29°	35°	36°	32°	36°	37°
−10 dB	34°	36°	37°	34°	36°	37°
−20 dB	37°	37°	38°	37°	37°	37°
−30 dB	37°	37°	35°	35°	38°	38°

Table 4.7: Room simulation using (Liu et al., 2000) algorithm ("direct" frequency integration method). Bandpass noise (100 Hz–1 kHz) stimulus with varying signal to noise ratio. Error values obtained the same way as in Table 4.4.

	source dist. 1 m			source dist. 3.5 m		
SNR	anech.	50%	unp. conc.	anech.	50%	unp. conc.
+30 dB	2.4°	17°	30°	2.8°	26°	35°
+20 dB	7.3°	22°	29°	8.2°	27°	35°
+10 dB	22°	26°	32°	22°	30°	35°
+6 dB	26°	28°	33°	29°	30°	36°
+3 dB	29°	31°	34°	30°	33°	36°
0 dB	32°	32°	34°	34°	31°	39°
−3 dB	33°	34°	35°	34°	36°	38°
−6 dB	36°	35°	35°	36°	37°	37°
−10 dB	35°	35°	37°	38°	36°	37°
−20 dB	38°	38°	37°	38°	37°	40°
−30 dB	38°	38°	39°	37°	37°	38°

As a comparison, we computed error values for the data from the "real world" tests (open loop) in the same way as in Table 4.4 and Table 4.5. The error for the broadband noise stimulus was 0.79°. The result for the 100 Hz–1 kHz bandpass noise was 9.34°. Note the similarity of the error of the real-world 100 Hz–1 kHz bandpass noise measurements, carried out at high SNRs, to the simulation values for high SNR and a sound source distance of 1 m in the "unpainted concrete" case (Table 4.5).

Table 4.6 and Table 4.7 show the results of the room simulations using our implementation of the Liu et al. (2000) algorithm with the "direct" frequency integration method (see Section 4.1.1). Although the issue of the outliers at ±90° mentioned in Section 4.1.1 could not be solved, a workaround was found by restricting the localization function $\mathrm{Loc_n}(i)$ (Eq. (4.11)) to the index range $i = 1 \ldots I-2$. This in effect reduces the available azimuths to the range $-89.5° \ldots +89.5°$ (with $I = 361$), but has the advantage of discarding the unwanted outliers.

Results for the broadband noise stimulus (Table 4.6) in the "anechoic" (1 m and 3.5 m source distance) and "50%" absorption cases (1 m source distance) are similar to those shown in Table 4.4, with the difference that major decreases in localization accuracy already appear at higher SNR. The "50%" case for a source distance of 3.5 m as well as the "unpainted concrete" case (both source distances) exhibit much higher angular errors than those shown in Table 4.4.

The angular errors for the bandpass noise stimulus (100 Hz–1 kHz) shown in Table 4.7 are significantly higher than those shown in Table 4.5, except for an SNR of +30 dB in the case of the "anechoic" absorption coefficients (both source distances).

4.1.3 Discussion

Method and control tests

While Liu et al. (2000) was a good starting point, it was noted that this algorithm, by taking into account only the minima of the coincidence map, does not use all of the information available. We minimized information loss by performing the frequency integration over the whole three-dimensional coincidence map. The modified algorithm produced excellent results without any indications of failures with computer-generated signals. The ambiguities observed for pure tones with a frequency higher than about 830 Hz were expected, given the structure of the algorithm. We chose not to implement the "stencil" filter method of frequency integration, because then we would have lost the constant angular resolution over the whole azimuth range. Furthermore, we did not want to incur the additional computational overhead associated with the method.

It is difficult to compare the performance of this algorithm with the performance of the original method, because Liu et al. (2000) restricted their experiments to simulations and anechoic chamber tests, and mainly conducted multi-source measurements. However, the one-speaker tests conducted in an anechoic chamber by Liu et al. (2000) seem to have produced a similar localization accuracy as our open-loop tests in a laboratory environment. Our own tests with the Liu et al. (2000) algorithm initially produced outliers at $\pm 90°$ (using the "direct" method), which overshadowed the correct source azimuth (if it was present at all) in all cases except high SNR broadband stimuli. By restricting the range of estimated azimuths to $-89.5° \ldots +89.5°$ (thus ignoring the outliers), a workaround was found which could produce usable data. With high signal-to-noise and direct-to-reverberant ratios, the precision is quite good and seems to reflect the data from the original publication. Nevertheless, the Liu et al. (2000) algorithm with the "direct" method of frequency integration showed higher sensitivity to SNR and reverberation. We suppose that this is related to the minimum operation performed on the coincidence values prior to frequency integration (Eq. (4.7)). For every frequency band, one minimum is returned, indicating the location of coincidence. This assigns equal weights to all frequency bands. This is not a problem with high SNR broadband signals. But at low signal-to-noise ratios or with narrowband stimuli, giving equal weight to frequency bands containing little or no energy pertaining to the signal seems to seriously corrupt the localization estimate.

One advantage of this type of algorithm, is that they can achieve sub-sample accuracy for interaural delays without requiring explicit interpolation between samples. This is a consequence of carrying out all computations in the frequency domain. Moreover, a high number of frequency bins may be implemented without an increase in the data size. Algorithms working in the time domain are using filter banks for frequency separation (e.g. Roman and Wang (2003)). These generate a high number of additional signals for

the left and right channels, which is computationally intensive. The algorithm presented here allows for efficient frequency filtering and, thus, restricting the computation of the coincidence map to frequency ranges relevant to the intended practical application.

Open-loop and closed-loop tests

In the open-loop tests in real-world conditions, performance was excellent for broad-band signals, but decreased with narrowing bandwidth of the stimuli. Specifically, problems in the low-frequency range were observed. As initial simulations (cf. Section 4.1.2) showed that the software was able to accurately determine the correct azimuth for all signal types, these high localization errors are not due to the algorithm but to reverberation as evidenced by subsequent room simulations (cf. Section 4.1.2). Adding an echo-avoidance system (Huang et al., 1999) might improve the situation. These authors used three omni-directional microphones on a mobile robotic platform. The localization was restricted to a single frequency band centered at 1 kHz and with a bandwidth of 600 Hz in order to avoid phase ambiguities. ITDs were computed by the zero crossings of the waveforms from microphone pairs and from these, the direction to the sound source could be computed. The localization accuracy was tested with a 1 kHz sinusoid and a hand-clapping noise. The error for the sinusoid stimulus was within $\pm 1^\circ$ whereas for the hand-clapping noise, the accuracy was within $\pm 7^\circ$. Although this system is able to perform sound localization in three dimensions as well as resolving front-back confusions, this is only possible through the use of three microphones. The restriction to a single frequency band in order to avoid phase ambiguities in the ITD computation seems to be too much of a constraint for practical applications. Additionally, extracting ITDs from several frequency bands with this method would entail a considerable additional computational overhead. In this respect, the algorithm described here is much more robust and suitable for future extensions.

In Nakadai et al. (2000, 2002) a frequency-domain algorithm for the sound localization subsystem of the humanoid torso SIG was used. The method performs ITD extraction by directly computing the phase difference between the left and right channels from FFT frequency peaks. Additionally, interaural level differences were included in the azimuth estimation. The error of the sound localization system was within $\pm 5^\circ$ from 0° to 30° and deteriorated for more lateral positions (Nakadai et al., 2002). Compared to this system, the method proposed here performs better for broadband noise.

The closed-loop experiments were performed to test the algorithm in an environment closer to its later application on a mobile robotic platform. The results confirmed those obtained during the open loop tests. An excellent localization was achieved with broadband signals. High-frequency signals were localized better than low-frequency signals. This demonstrated that the algorithm may be applicable to dynamic, real-world situations.

Although comparisons are difficult, from the above we have the impression that our system, despite its simplicity, doesn't perform much worse in azimuth estimation than microphone arrays with more than two microphones.

Omologo and Svaizer (1994) used 4 equispaced microphones with a separation of 15 cm. Three different localization algorithms were tested, with the so-called crosspower-spectrum phase algorithm providing the best results. For the experiments, 97 stimuli were used with frequency content ranging from narrowband to wideband at various azimuths and distances ranging from 1 m to 3.6 m. Half of the stimuli had a noise component with an average SNR of 15 dB. Localization accuracy was 66 % with a tolerance $< 2^\circ$, 88 % with a tolerance $< 5^\circ$ and 96 % with a tolerance $< 10^\circ$.

Brandstein and Silverman (1997) used a bilinear array of 10 microphones with an inter-microphone separation of 25 cm. Their system used a frequency-domain time-difference of arrival estimator designed for speech signals combined with a speech source detector. For experiments with single, non-moving sources, 18 different source positions were tested. Speech stimuli were used. The angular error was approximately 2.5° over a range of 3 m.

Valin et al. (2003) used 8 microphones arranged on the summits of a rectangular prism of dimensions 50 cm × 40 cm × 36 cm. The acoustic environment was noisy with moderate reverberation. The localization system used the crosspower-spectrum phase algorithm enhanced with a spectral weighting scheme. The angular error was approximately 3° over a range of 3 m. The stimuli used for the experiments consisted of snapping fingers, tapping foot and speaking.

Room simulations

The simple "shoebox" room model helped with understanding the acoustic environment in which the real experiments took place. Three conclusions can be drawn from these simulations. Firstly, the algorithm is relatively robust against noise, as important changes in localization error can only be observed beginning at signal to noise ratios between −3 dB in the worst case and −20 dB in the best case. Secondly, the most important parameter degrading localization performance is direct-to-reverberant ratio. This becomes particularly apparent with the sound source at a distance of 3.5 m from the microphones and highly reflective surfaces ("unpainted concrete"), where even the azimuth estimation of a broadband stimulus produces rather large errors. Thirdly, the room simulations suggested that the systematic deviations observed with the low-frequency narrowband noise stimulus were due to room reverberations. This is in accordance with findings that binaural cues vary depending on the acoustic environment and noise conditions (Nix and Hohmann, 2006).

After the ITD-based sound source localization algorithm, we will now present an ILD-based method.

4.2 Sound Source Localization Using Interaural Level Differences

The sound source localization method using interaural level differences is inspired by the model of Spence and Pearson (Spence and Pearson, 1989) and was implemented by Daniel Peger for his diploma thesis (Peger, 2005).

4.2.1 Spence & Pearson Model

This is based on a computational model of the sigmoid response curves of VLVp neurons (the first site of binaural interaction in the owl's intensity pathway with neurons narrowly tuned to frequency; see Section 2.3.4) and the peaked responses found in ICc ls neurons (sensitive to ILD and broadly tuned to frequency; see Section 2.3.4).

VLVp model

The voltage of a VLVp unit is governed by the following differential equation:

$$C \cdot \dot{v} = -g_l \cdot (v - v_l) - g_e \cdot (v - v_e) - g_i \cdot (v - v_i). \tag{4.14}$$

With $\dot{v}$ being the derivative of voltage with respect to time and C the cell capacitance. The leakage, excitatory and inhibitory conductances, driven by the voltages v_j, are represented by g_j (with $j \in \{l, e, i\}$). The variations of the conductances g_j obey a damped oscillation:

$$\ddot{g}_j + \gamma \dot{g}_j + \omega^2 \cdot g_j = 0 \tag{4.15}$$

$$\Leftrightarrow \quad \ddot{g}_j = -\gamma \cdot \dot{g}_j - \omega^2 \cdot g_j, \text{ with } j \in \{l, e, i\}, \tag{4.16}$$

where γ and ω are chosen in a manner that the oscillator is at least critically damped, so that g_j remains positive.

The output of a unit is a sigmoidal function of the voltage v. Thus it is a real number, representing the neuron's activity in a similar manner as the spike rate per second of a neuron.

To simulate the dorsoventral gradient in inhibition present in the VLVp (see Section 2.3.4), a gradient in the number of inhibitory cells is assumed, the ventral end containing more inhibitory cells than the dorsal end. This is achieved in the model by a linear increase of the maximum firing rate of units from dorsal to ventral.

The central idea of the model is the *criss-cross* pattern of projections between VLVp nuclei (see Figure 4.12). Dorsal units from the VLVp on one side project to ventral units of the VLVp on the other side of the brain and vice versa.

The VLVp input, provided by NA is modeled as a constant spike rate, proportional to ipsilateral sound pressure level.

ICc ls model

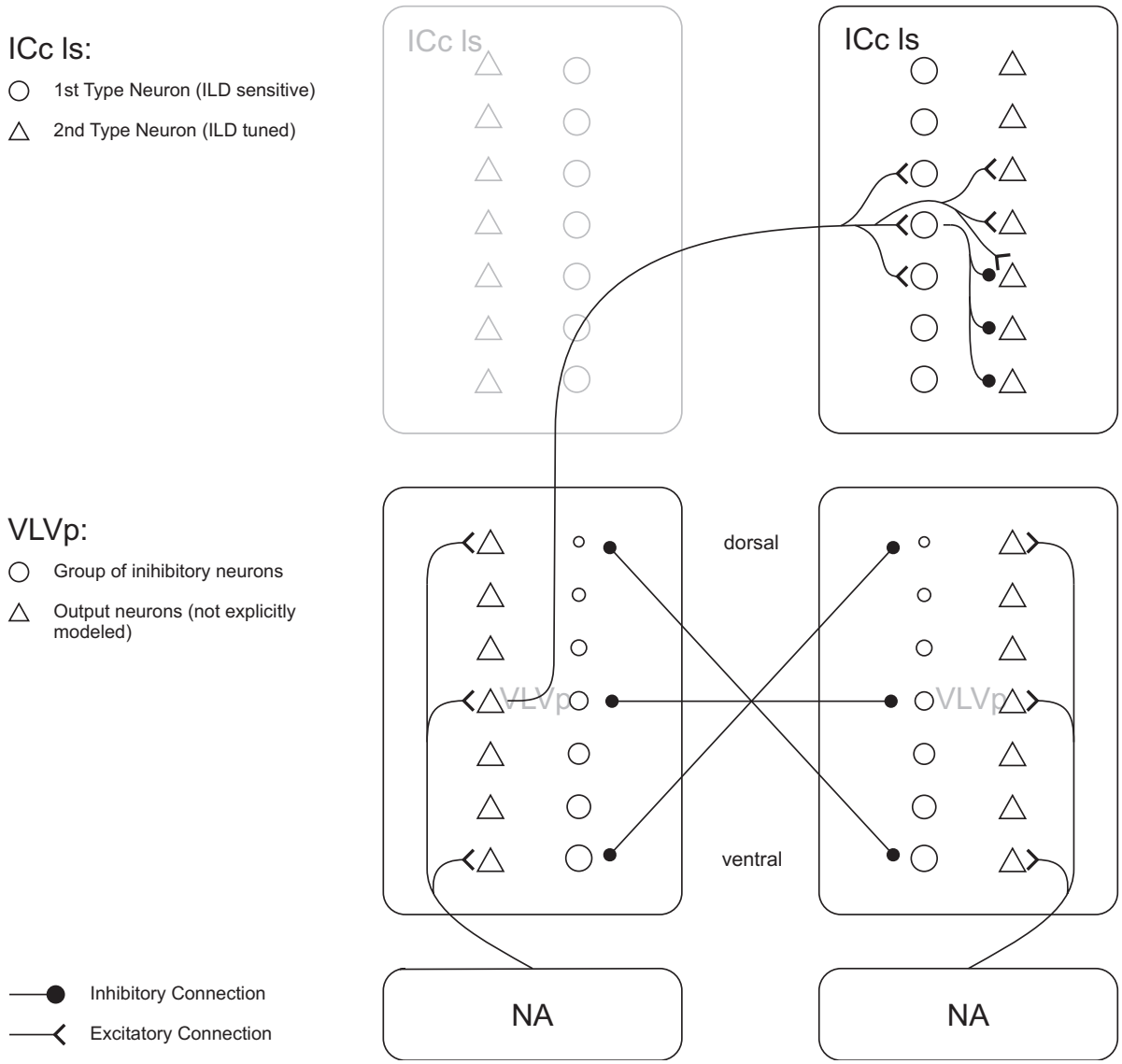
The ICc ls model is simpler. Taking the derivative of the input from the VLVp — which is sigmoid in shape — results in the desired peaked response.

In order to achieve this derivation on a neuronal basis, Spence and Pearson assumed two distinct populations of neurons, receiving excitatory input from the VLVp. The first (triangles in Figure 4.12) receives only input from VLVp and is thus sensitive to ILD. The second population of neurons (circles in Figure 4.12) receives input from the VLVp as well as an inhibitory connection from the first type of neurons. The first type of neurons inhibit only second-type neurons which are sensitive to lower ILD values (i.e. located more ventrally).

4.2.2 Implemented Model

The model that was implemented relatively closely follows that proposed in Spence and Pearson (1989). The artificial neural network consists of six layers — one for each VLVp and two layers for each ICc ls modeling the two assumed neuron populations. The neuron model (Equation 4.14) has been simplified. Real neurons have a time-dependent response, but we were only interested in a constant output, given a constant input (i.e., the neuron's state does not change between network updates). Under this assumption ($\dot{v} = 0$), Equation 4.14 becomes:

$$v = \frac{g_e \cdot v_e + g_i \cdot v_i + g_l \cdot v_l}{g_e + g_i + g_l} \tag{4.17}$$



(Peger, 2005)

Figure 4.12: Model of the VLVp and the ICc ls. Circle size in VLVp indicates number of inhibitory cells that project to contralateral VLVp (after Spence and Pearson (1989)).

A unit's activity, a, is defined by a sigmoid function of voltage v, weighted to increase the slope of the sigmoid curve:

$$a = \frac{1}{1 + e^{-w_{sp}(s) \cdot (v - v_t)}}, \tag{4.18}$$

$$w_{sp}(s) = ln(s), \tag{4.19}$$

where v_t determines the unit's threshold and $w_{sp}(s)$ is the aforementioned weighting factor.

Contrary to the proposition by Spence and Pearson (1989), it was chosen to make the randomization of the activity optional, in order to obtain repeatable results.

The pattern of the connections between the units is given in Figure 4.12, but the description of the connection weights in Spence and Pearson (1989) is a rather conceptual one, leaving room for interpretations.

In the following, the left superscript refers to the nature of the connection (+ for excitatory; − for inhibitory), whereas the right superscript specifies the receiving layer.

The first subscript specifies the projecting, whereas the second subscript denotes the receiving unit.

We chose to model the connection weights for the four different types of connections as detailed below:

Excitatory from NA to contralateral VLVp, serving as input to the network:

$$ {}^{+}w_{j,k}^{\mathrm{VLVp}} = \frac{1}{\max_{\mathrm{input}}} \tag{4.20} $$

${}^{+}w_{j,k}^{\mathrm{VLVp}}$ is the excitatory input to the k-th unit of VLVp from the j-th unit of the projecting layer. This means that the input to the network is normalized to $\max_{\mathrm{input}}$.

Inhibitory from contralateral to ipsilateral VLVp:

$$ {}^{-}w_{j,k}^{\mathrm{VLVp}} = \begin{cases} \frac{j}{|\mathrm{VLVp}|}, & \text{if } j+k = |\mathrm{VLVp}| \\ 0, & \text{otherwise.} \end{cases} \tag{4.21} $$

This models the inhibitory gradient in the VLVp as a linear increase in connection weight from dorsal to ventral ($|\mathrm{VLVp}|$ denotes the number of elements in the modeled VLVp).

Excitatory from contralateral VLVp to both ICc ls populations:

$$ {}^{+}w_{j,k}^{\mathrm{ICc}} = \begin{cases} \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{\frac{-(k-j)^2}{2\sigma^2}}, & \text{if } j-\sigma \leq k \leq j+\sigma \\ 0, & \text{otherwise} \end{cases} \tag{4.22} $$

The connection weight is a Gaussian with standard deviation σ centered at the index of the projecting unit. Additionally, only units within a range of $\pm\sigma$ of the index of the projecting unit are taken into account, all other connections being set to 0.

Inhibitory from 1st type ICc ls to 2nd type ICc ls neurons:

$$ {}^{-}w_{j,k}^{\mathrm{ICc}} = \begin{cases} \frac{k-j}{\sigma}, & \text{if } j < k \leq j+\sigma \\ 0, & \text{otherwise} \end{cases} \tag{4.23} $$

Here too, the connections to only a restricted range of units within σ of the projecting neuron's index are considered. The weight increases linearly with increasing distance from the projecting neuron.

4.2.3 Results

All the experiments took place in a sound attenuating chamber (IAC, 2 m × 2 m × 2.6 m). The pan-tilt-unit with the microphones was placed roughly in the center of the chamber. The microphones and the PTU were the same as in Section 4.1.2 (see Figure 4.3). The loudspeaker (Sony SRS-57) was at a distance of approximately 50 cm[3] from the microphones. The stimulus was broadband noise, generated online during the experiments in order to avoid frozen noise. The signal sampling frequency was 44.1 kHz. FFT size was

[3]Due to the spatial constraints of the chamber as the microphones and loudspeaker mounting, a bigger separation was not possible.

4 096 samples resulting in a frequency resolution of approximately 11 Hz. A timeframe also consisted of 4 096 samples, yielding a timeframe size of 93 ms. Preliminary tests had shown that the microphone assembly, as used in Section 4.1.2, did not produce usable ILDs. Therefore, paper flaps were mounted on the edges of the microphones, facing each other in order to increase level differences.

In a first step, the microphone assembly had to be calibrated, i.e. the actual ILDs had to be recorded and stored in tables along with the azimuth values for the software to evaluate.

For the actual experiments, the PTU was panned from $-90°$ to $+90°$ in $2°$ steps. At each azimuth, white noise was presented and 20 timeframes were recorded.

Figure 4.13 shows the result of these experiments with different algorithm setups. Figure 4.13 **a** (`ILDEstimateSub`) shows the performance of directly computing the ILDs from the input signals (see Equation 2.3.4). Figure 4.13 **b** & Figure 4.13 **c** show the results for our implementation of the Spence and Pearson model with 20 neurons (`ILDEstimateSP20`, Fig. 4.13 **b**) and 40 neurons (`ILDEstimateSP40`, Fig. 4.13 **c**).

As becomes evident from these diagrams, the simple `ILDEstimateSub` algorithm performs equally well as `ILDEstimateSP20` and `ILDEstimateSP40`. Increasing the number of neurons in the Spence and Pearson model does not significantly improve performance. What can also be seen is that from $-30°$ to $30°$ (approximately), the ILD localization is quite precise.

4.2.4 Discussion

The performed experiments show that ILD-based sound source localization is possible, even with a rather basic microphone assembly (paper flaps added to the microphones). Although only in a restricted azimuth range (about $-30°\ldots30°$).

It could also be shown that the implementation of the Spence & Pearson model (`ILDEstimateSP`) did not yield better results than the direct computation of ILDs (`ILDEstimateSub`), which is faster. This was half-expected as both methods relied on the same HRTF data. Furthermore, the tests took place in an anechoic chamber.

More realistic (i.e. real-world) tests would have to be performed in order to find any advantages in using the Spence & Pearson model (apart from the fact that it models the barn owl's auditory system).

Additionally, more elaborate microphone assemblies should be tested in order to compensate for the restricted azimuth range. A first attempt in that direction was done by implementing an artificial owl ruff. This project is described in the next section.

4.3 Artificial Owl Ruff & Combined ITD/ILD Localization

In order to further explore the potential of ILD-based localization, we decided to develop a structure mimicking the properties of the barn owl's facial ruff. The basic characteristics of the barn owl's ruff (from the point of view of binaural cues) are the extension of the interaural time difference range beyond $\pm90°$ and a strong vertical gradient in interaural level differences (von Campenhausen and Wagner, 2006). This resulted in the diploma thesis of Tobias Krämer (Krämer, 2008).

Different variants of artificial ruffs (see Figure 4.14) were tested for their performance in azimuth localization using ITDs only, azimuth localization using ILDs only, localiza-

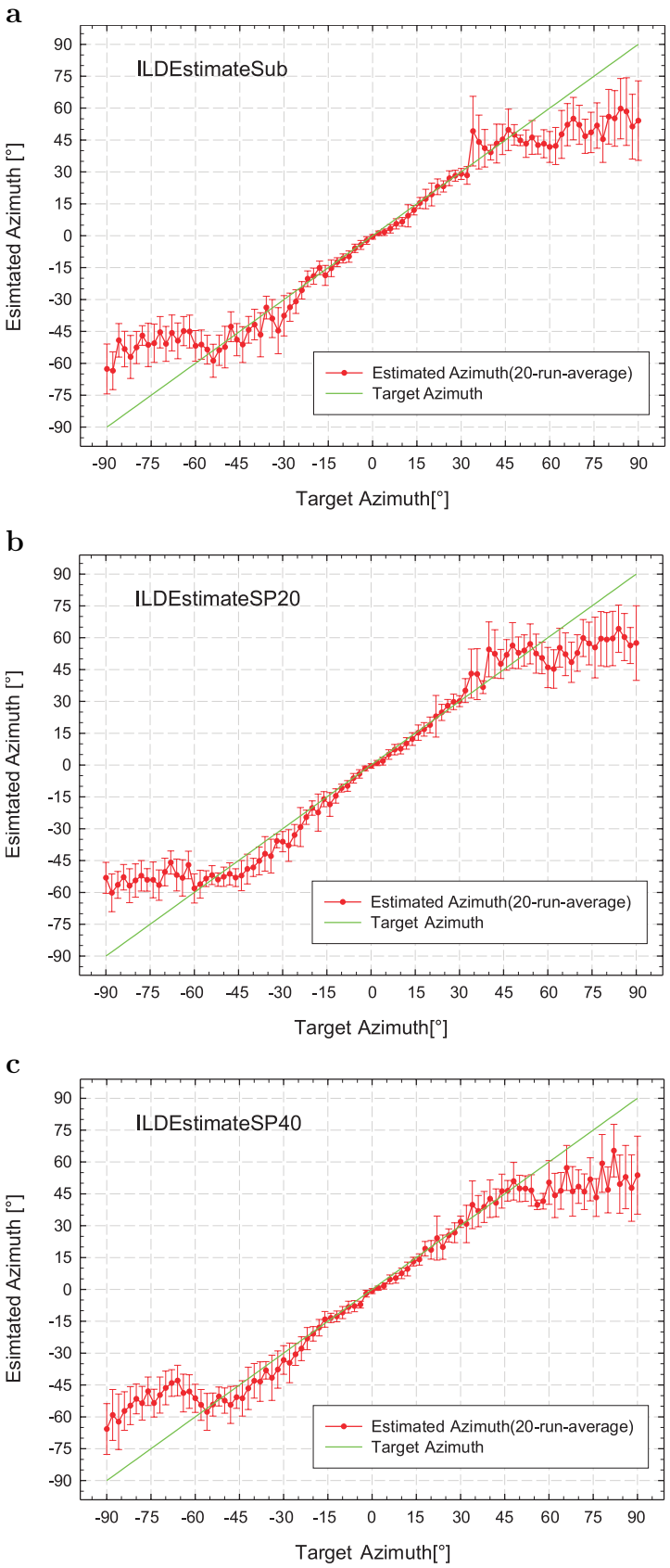


Figure 4.13: Performance of the ILD localization system. **a** Simple subtraction of dB-value for left channel from dB-value of right channel. **b** Spence & Pearson model with 20 neurons. **c** Spence & Pearson model with 40 neurons. (Peger, 2005)

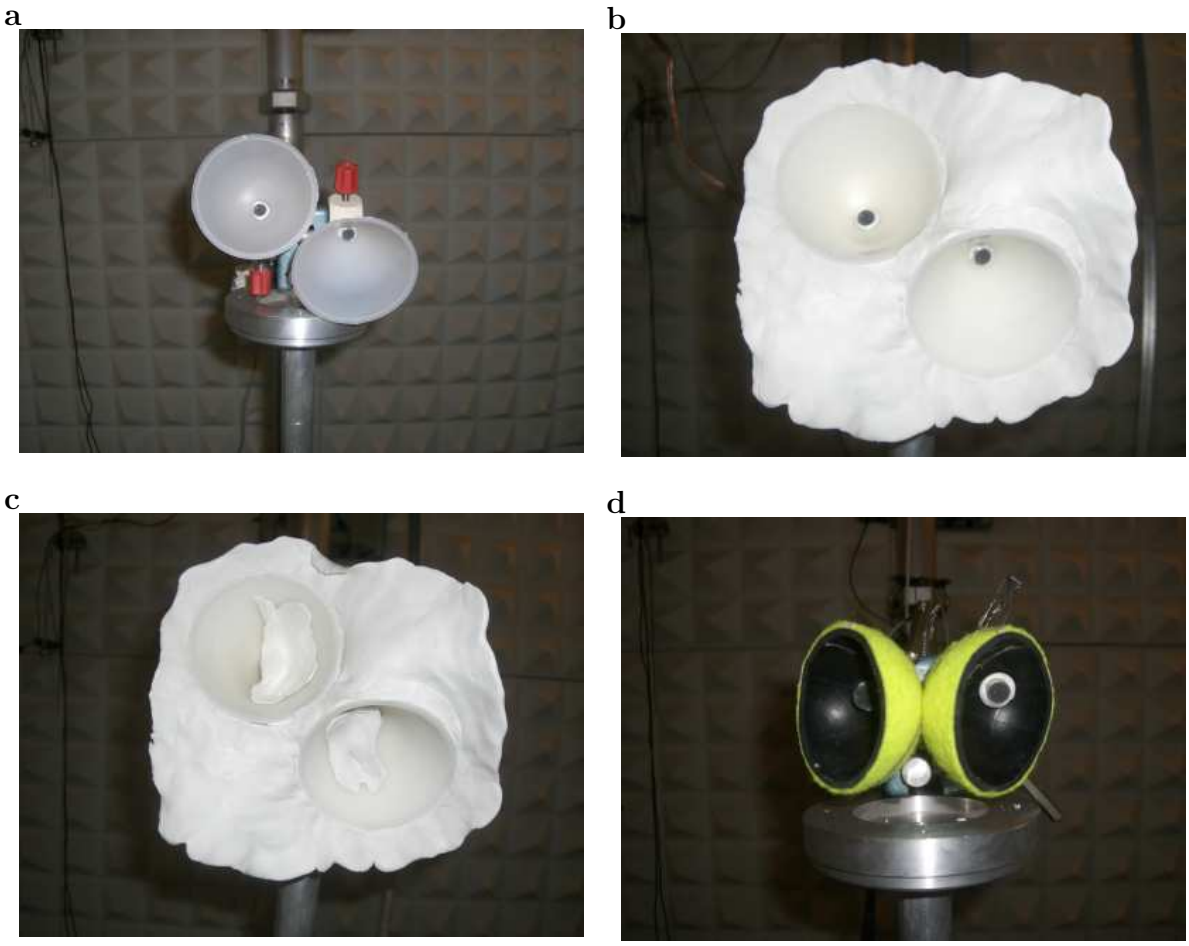


Figure 4.14: Artificial ruff variants. **a** Funnels only. The microphones capsules can be seen as black dots in the centers of the funnels. **b** Funnels with polymer clay "ruff". **c** Funnels with polymer clay "ruff" and flaps. **d** Tennis ball halves. (Krämer, 2008)

tion in azimuth and elevation using ILDs alone and azimuth/elevation localization using both cues.

For these tests, the algorithm for ITD localization described in Section 4.1.1 and the ILD localization method described in Section 4.2 were used.

In order to combine ITD and ILD cues into a single azimuth/elevation estimate, we used a method similar to that described in (Viste and Evangelista, 2004). In that approach, unambiguous, but imprecise azimuth estimates based on ILD were used to select the correct and more precise ITD-based estimate from the phase-ambiguous ITD computation. We had a similar problem for azimuth/elevation localization using both cues. The ILD localization system provided multiple position hypotheses, among which the correct one was mostly to be found, but not necessarily as the best estimate. As the ITD model of the dual delay-line algorithm was not adjusted for the altered ruff ITDs, azimuth estimates based on ITD were rather inexact, but unambiguous. The system we implemented then selected the ILD-based position estimate nearest to the ITD-based one in order to eliminate the shortcomings of both methods.

4.3.1 Results

In order to evaluate different kinds of artificial ruffs, HRTF measurements of the various designs were performed. Except for the microphones, the setup (including analysis

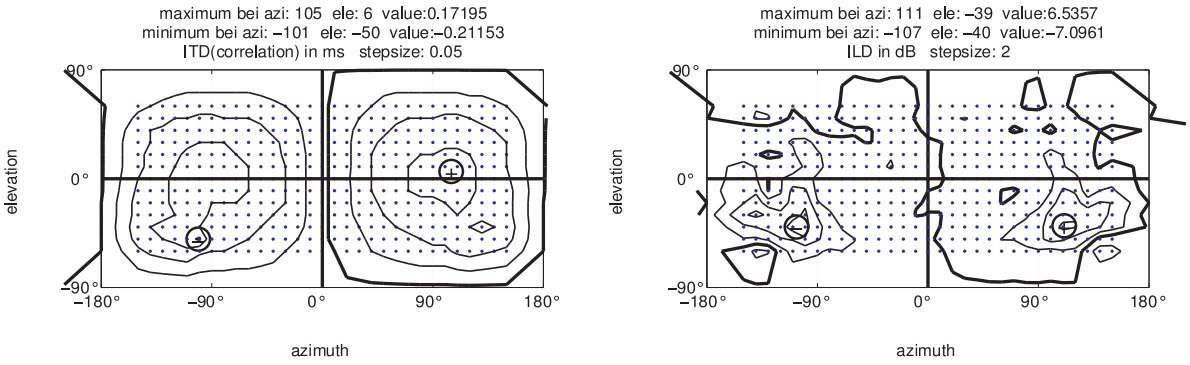


Figure 4.15: ITD/ILD contour lines of microphones only assembly (used as control for artificial ruffs). Extrema of ITD and ILD (+, −) shown in circles. Distance between ITD contour lines is 50 µs. Distance between ILD contour lines is 2 dB. Note that, as expected, ITDs have their maxima at approximately ±90° and do not vary with elevation. ILDs show hardly any variation. (Krämer, 2008)

software) used for these experiments was the same as that used by von Campenhausen and Wagner (2006). The microphones were omnidirectional electret capacitor cartridges (Monacor MCE-101) with a rated frequency range of 50 Hz–10 kHz.

Figure 4.15 shows the results of the control measurements with the microphones alone (no ruff). The HRTFs look as expected, with the extrema around ±90° and not very strong interaural level differences.

Figure 4.16 shows the results of the HRTF measurements for the different artificial ruffs presented in Figure 4.14.

Surprisingly, the variant with the funnels alone (Fig. 4.16 **a**) yields the best results. ILD values are much higher than in the case of the control measurements and there is a strong ILD gradient in the vertical direction. ITDs exhibit an oblique gradient, meaning that they vary with azimuth as well as with elevation. Although this is rather detrimental to the dual delay-line algorithm as described in Section 4.1.1, the method can easily be adapted to the situation.

Adding a polymer clay "ruff" (Fig 4.16 **b**) and flaps (Fig 4.16 **c**) does not have much of an effect on ITDs and ILDs. On the contrary, it even tends to make matters worse. This is clearly visible in the ILD contour plot in Fig 4.16 **c**.

The tennis ball halves (Fig 4.16 **d**) do have a positive influence on the binaural cues. The variation of ITDs remains confined to the azimuth and there are strong ILDs with this setup. Unfortunately, the ILDs exhibit no vertical gradient, making the tennis ball halves variant unsuited for localization in elevation using ILDs.

After measuring the characteristics of the various artificial ruffs, we went on to measure sound source localization performance using these setups. This was done by using the recorded audio files from the HRTF measurements as input to the different localization modules. In the following, we will only show the results for the funnels-only artificial ruff, which yielded the best results.

Figure 4.17 **a** shows the performance in azimuth estimation with the funnels-only artificial ruff and using ITDs only. The red crosses indicate the measurements made with the ruff. For comparison, The green dashed line shows the averages of measurements made with the microphones only. The dotted blue line shows actual sound source position. The azimuth deviations are caused by the use of a simple model of ITDs in the dual delay-line algorithm (Eq. 4.2), which is only suitable for setups where the micro-

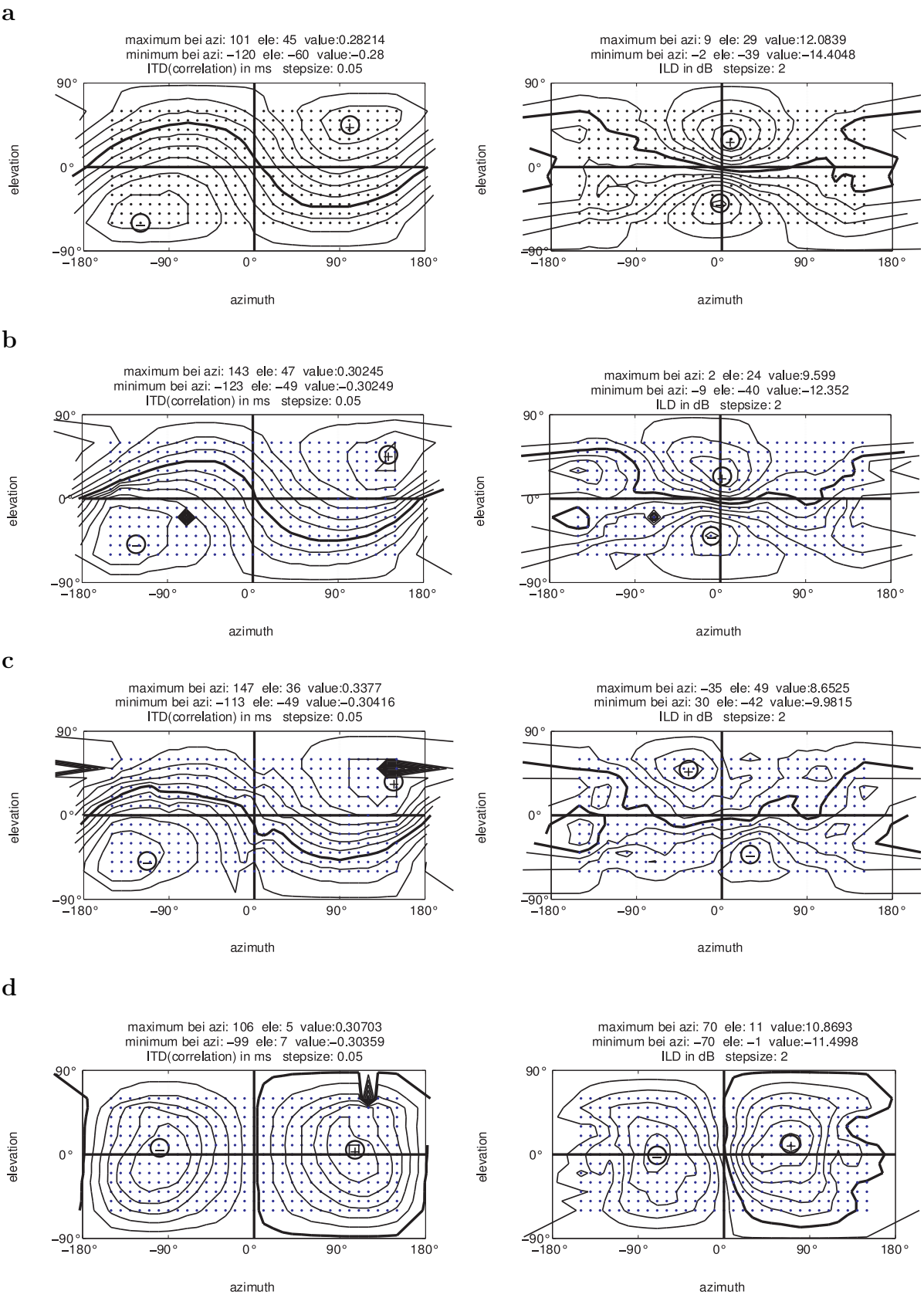


Figure 4.16: ITD/ILD contour lines of artificial ruff variants. Left panels show ITD, right panels show ILD. Extrema (+, −) in circles. ITD contour lines are 50 µs apart, ILD contour lines distance is 2 dB. **a** Funnels only (Fig. 4.14 **a**). **b** Funnels with polymer clay "ruff" (Fig. 4.14 **b**). **c** Funnels with polymer clay "ruff"and flaps (Fig. 4.14 **c**). **d** Tennis ball halves (Fig. 4.14 **d**). (Krämer, 2008)

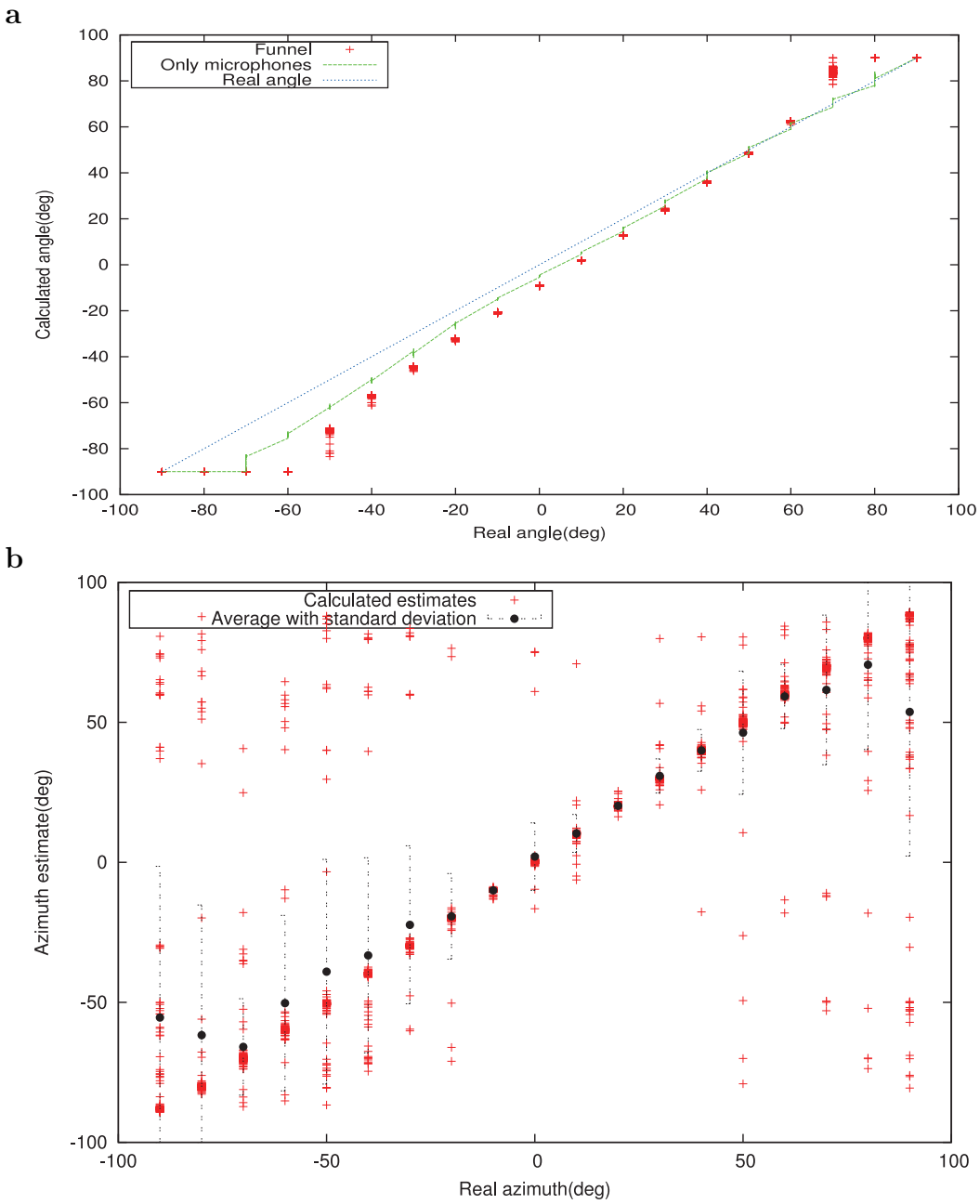


Figure 4.17: Azimuth localization using artificial owl ruff (funnels-only variant). Localization using **a** interaural time differences and **b** interaural level differences. (Krämer, 2008)

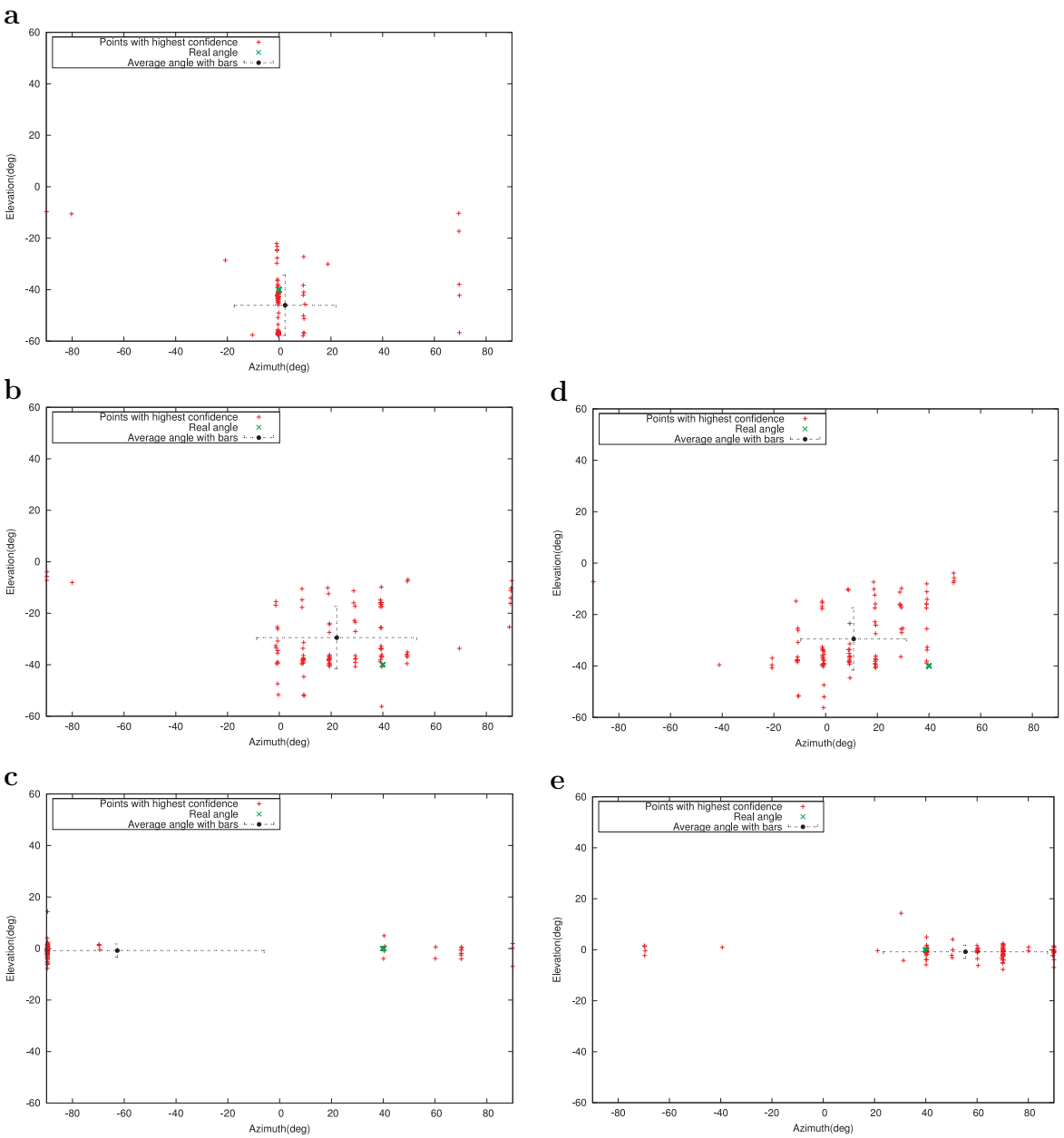


Figure 4.18: Localization in azimuth and elevation using the artificial owl ruff (funnels-only variant). **a-c** Localization using ILDs only. **d-e** Localization using the combination of ITDs and ILDs. Only a few exemplary positions are shown: **a**: (0°, −40°); **b**, **d**: (40°, −40°); **c**, **e**: (40°, 0°). (Krämer, 2008)

phones are mounted without any obstructions between them. Directly using the ITDs for the funnel measurements would improve the situation.

Figure 4.17 **b** depicts the performance in azimuth estimation using ILDs and the funnels only artificial ruff. Depicted are all measured angles (red crosses) and the averages with standard deviations (dots with error bars). The deviations are caused by the funnels themselves, as they do not generate a usable ILD gradient in the azimuth direction (see Fig. 4.16 **a**, right panel). Nevertheless, localization seems to be quite accurate in a range from approx. −20° to 30°, which is similar to the performance measured by Peger (see Figure 4.13).

Finally, we did some experiments to evaluate the performance of sound source localization with artificial ruffs in azimuth as well as in elevation.

Figure 4.18 shows the results for only a few positions in space. The figures in the left column (Fig. 4.18 **a-c**) were generated by exclusively evaluating ILDs. As expected with this setup, localization in elevation is quite accurate. The azimuth, similar to Figure 4.17 **b**, could not be determined as accurately.

The right column (Fig. 4.18 **d-e**) shows the results of evaluating ITDs and ILDs in combination. In Fig. 4.18 **e**, the combination with ITDs brings a clear improvement over the case of ILDs only (Fig. 4.18 **c**). On the other hand, in Fig. 4.18 **d**, the additional taking into account of ITDs worsens the previous position estimate (Fig. 4.18 **b**). The cause of these ambivalent results are to be found in the current implementation of the dual delay-line algorithm. It supposes that ITDs for a specific azimuth are the same, regardless of elevation. As with the funnels-only ruff, ITD varies also with elevation (cf. Figure 4.16 **a**), this assumption is clearly violated, leading to imprecise sound source localization.

4.3.2 Discussion

The work on the artificial owl ruffs represents a natural extension to not only the work done on ILD localization (Section 4.2), but also to the ITD localization system (Section 4.1).

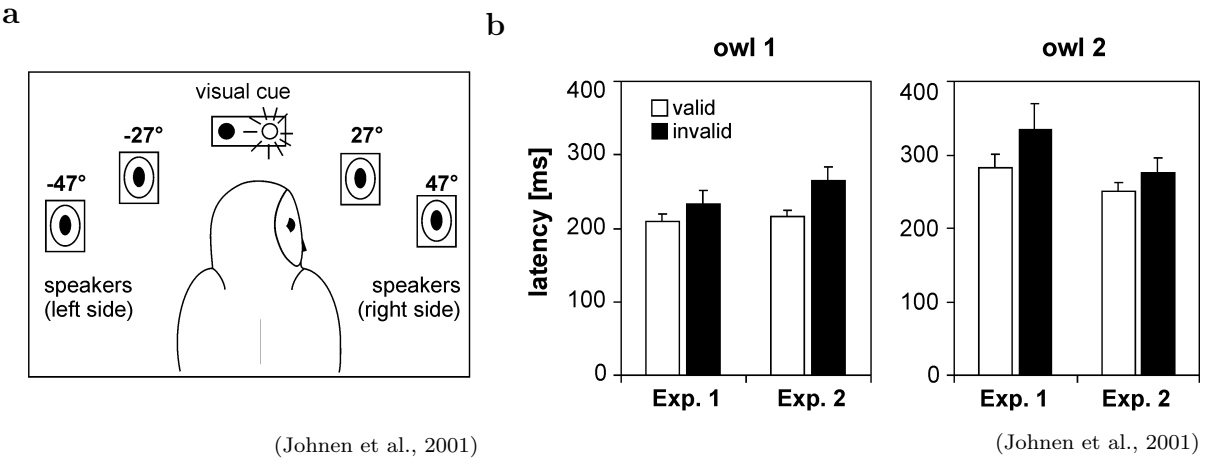
It could be shown that — through microphone assembly ("ruff") design — the binaural cues can be modified to fit specific purposes (e.g. azimuth localization using ITDs and ILDs or azimuth / elevation localization using ILDs alone or in combination with ITDs). It could also be shown that the combination of the binaural cues can be (unsurprisingly) beneficial to sound source localization precision. Nevertheless, even if the results are quite positive, some work still remains to be done. Chiefly, the artificial ruff has to be designed very carefully in order for the binaural cues to behave as expected. Also, either the ITDs should not be modified by the ruff — which seems quite difficult to achieve — or the dual delay-line algorithm has to be modified in such a way that it can work with arbitrary (i.e. measured as opposed to computed) ITD to azimuth mappings.

4.4 Multimodal Spatial Attention Module

As a sound localizing mobile robot will have to be provided with a mechanism for modulating its attention in the presence of multiple sound sources, it was only natural to again turn towards the barn owl to look for inspiration.

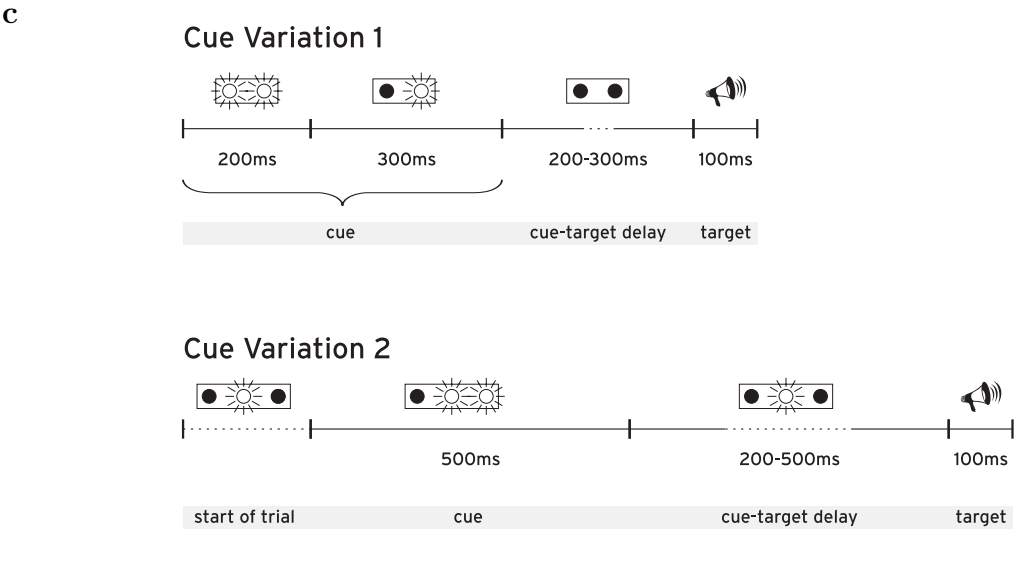
The result was the diploma thesis of Dominik Röttsches (Röttsches, 2004), which was concerned with multimodal spatial attention, specifically audio-visual spatial attention. The basis for this project was the work done by Johnen et al. (2001), which showed that attention modulates the reaction latency of barn owls in a visual-auditory cueing paradigm.

A visual cue in front of the owl pointed in 80% of the trials to the side of an upcoming auditory stimulus (valid configuration). In 20% of the cases, the visual cue pointed in the opposite direction (invalid configuration). The owl had to respond to the auditory stimulus with a head saccade (i.e. a rapid head movement) towards the sound source (27° or 47° to the side, 0° elevation). The owl's response latencies are significantly shorter in valid trials when compared to invalid trials. Depending on owl and target size, an average reduction of response latencies by 10%-20% occurred in valid trials (see Figure 4.19 **b**). This clearly shows that, if the owl's attention was attracted to one



(Johnen et al., 2001)

(Johnen et al., 2001)



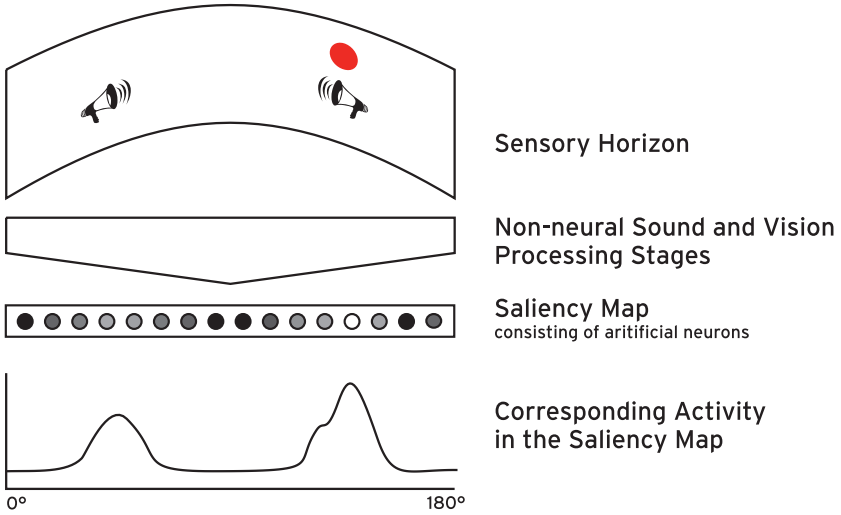
(Röttsches, 2004)

Figure 4.19: Visual-auditory cueing paradigm. **a** Experimental setup. **b** Owl response latencies (means and 95% confidence intervals). Open bars: valid configuration; closed bars: invalid configuration. **c** Cue variations. Variation 1 and 2 correspond to Exp.1 and 2 in **b**, respectively.

hemisphere, it was more difficult for the animal to turn towards a target sound if it was presented in the other hemisphere.

Building on these findings, we built a model based around the concept of a *saliency map* (Itti and Koch, 2000).

The main features of the model are depicted in Figure 4.20. The saliency map receives auditory (from the dual delay-line algorithm; cf. Section 4.1.1) as well as visual input (simple color or light detection from a camera). The auditory input covers a range of 180° (±90°), whereas the visual input, due to the limited field of view of the camera, only covers a range of 30° (±15°). The visual input provides a preactivation of the neural network on the side of the visual cue and a preattenuation on the contralateral side. The auditory input is added to that preactivation. This results in the activation of the saliency map being highest on the side of the visual cue. If a given threshold of neural activity is reached, a signal is sent to the pan-tilt-unit (PTU) on which the camera and microphones are mounted and the unit turns towards the sound source. This way, the PTU will turn with a lower latency (and higher probability) towards the sound source if the visual cue was on the same side. With that setup, we could quite



(Röttsches, 2004)

Figure 4.20: Multimodal spatial attention module: model overview.

accurately reproduce the experiments described in Johnen et al. (2001).

Figure 4.21 **a** shows the connections between the neurons of the auditory and visual layers and the saliency map (hidden layer) units. In order to implement attention, some sort of memory for past events has to be available. This is done via a memory map, consisting of two neurons (for the left and right side). The recurrent connections between the memory map and the saliency map are shown in Figure 4.21 **b**.

The following criteria have to be satisfied if the network is to function as expected:

1. With no sensory input, the network must return and stay in an equilibrium state.

2. A stimulus from the auditory or visual inputs should lead to

   - a peak at the corresponding position in the saliency map.
   - a rising preactivation of the corresponding saliency map half until a saturated state is reached.
   - a preattenuation of the opposite half of the saliency map until a minimum preattenuation is reached.

3. Higher preactivation means more time steps until equilibrium state is reached again.

4. High preactivation on one side and strong preattenuation on the other leads to longer time for the network to reach a new state with the stimulus on the non-preferred side.

5. The recurrent connections must not lead to a positive feedback loop from which the network cannot recover to the equilibrium state.

As the network behavior is largely determined by the forward and recurrent weights between saliency and memory map, $w_{forward}$ and $w_{recurrent}$ have to be carefully chosen in order to fulfill the above criteria.

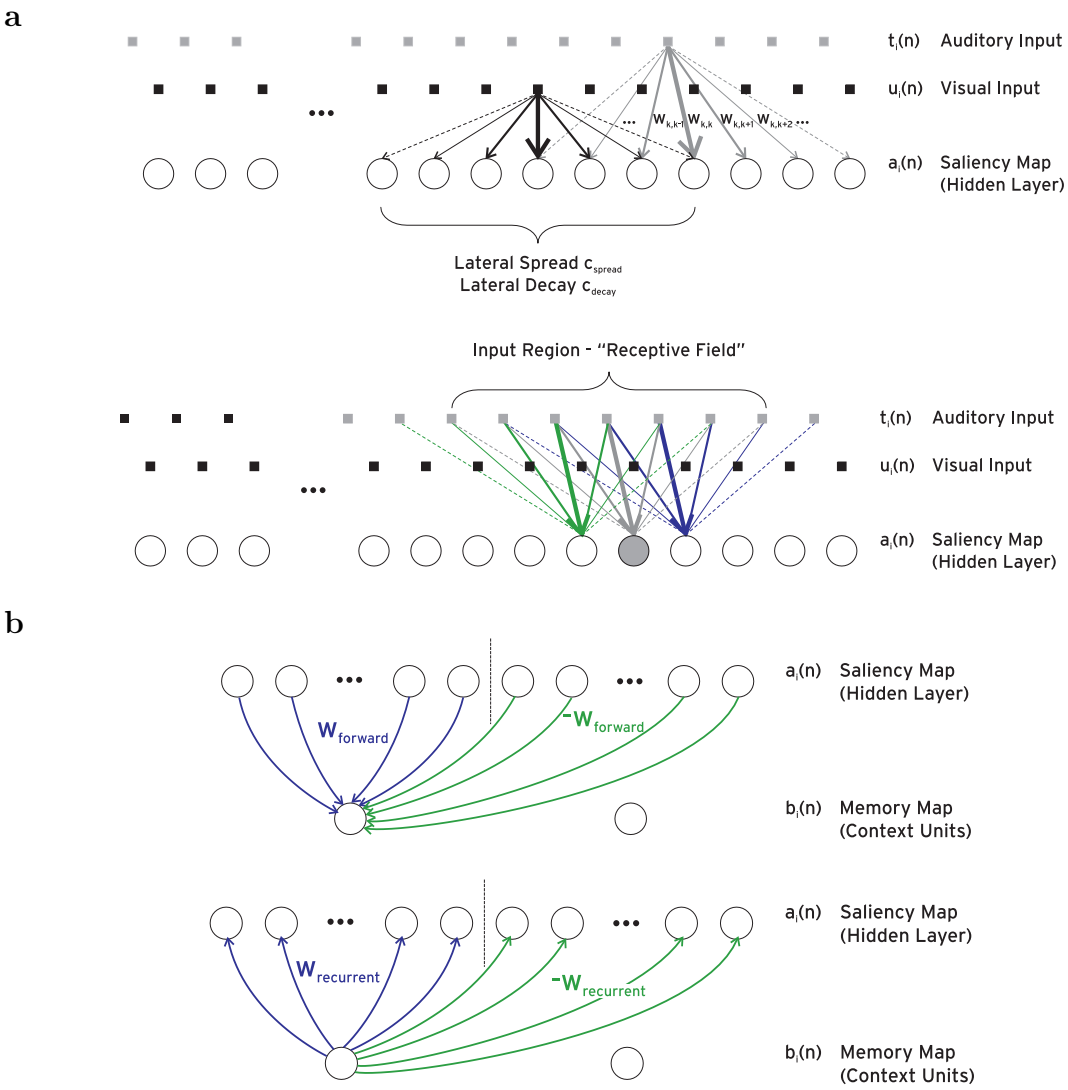


Figure 4.21: Multimodal spatial attention module: neural network. **a** Top panel: Connections from inputs to saliency map (only two units shown, other units similar). Bottom panel: Structure of weighted connections to saliency map units. **b** Recurrent connections between saliency map and the memory map. Top panel: forward connections. Bottom panel: recurrent connections. (Röttsches, 2004)

4.4.1 Results

We performed similar experiments to those described in Johnen et al. (2001). For these experiments, the pan-tilt unit (Directed Perception PTU-D46-17) with the microphones and the camera was set up in an anechoic chamber (IAC, base 2 m × 2 m, height 2.6 m; see Figure 4.22).

The microphones were two Sony ECM-F8 electret capacitor omnidirectional microphones. The camera was a Philips ToUcam PRO II webcam mounted in the center between the microphones. As visual cues, two super bright LEDs were used instead of the original LED display of Johnen et al. (2001), as this proved to be too dark for the camera. The loudspeakers were positioned at an azimuth of −25° and 25° with respect to the microphone assembly.

Although saliency map dynamics were also evaluated, no comparable data can be found in Johnen et al. (2001). Therefore we will only show the results of the "behavioral" experiments, performed using the Cue Variation 1 (Figure 4.19 **c**) with cue-target

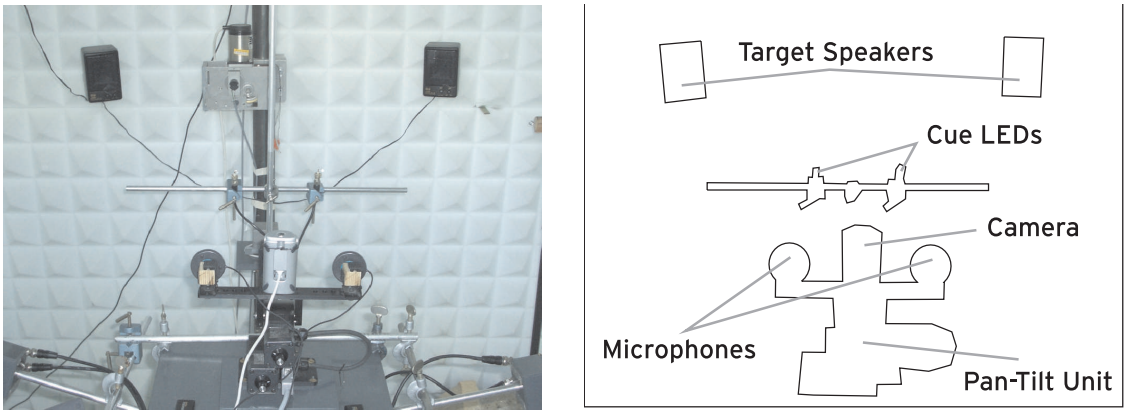


Figure 4.22: Multimodal attention module, experimental setup. (Röttsches, 2004)

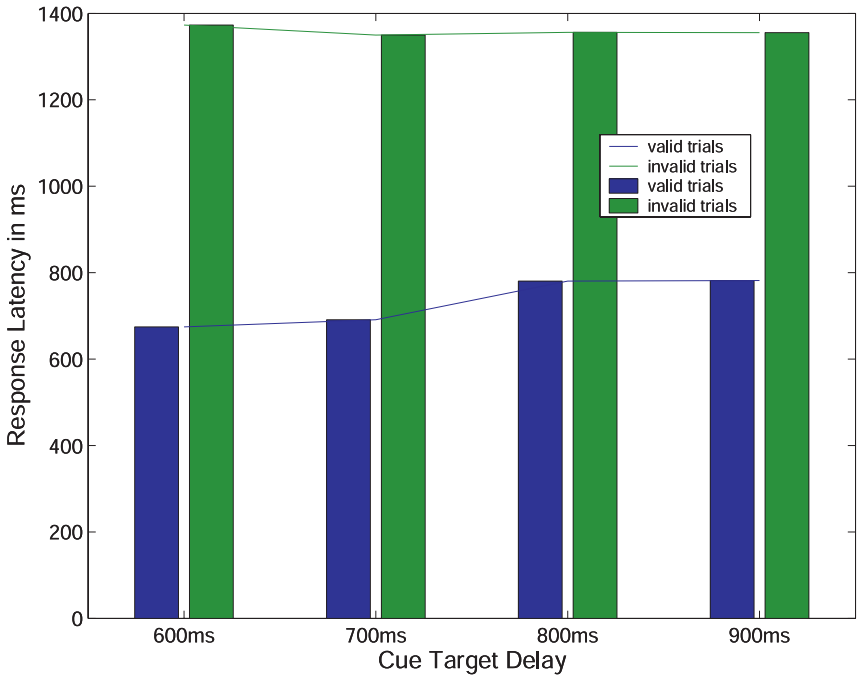


Figure 4.23: Multimodal attention module, response latencies. (Röttsches, 2004)

delays of 600 ms, 700 ms, 800 ms and 900 ms. The results of these tests are shown in Figure 4.23. For every cue-target delay, there is a significant increase in response latency for invalid trials (green bars). The increase in response latency for the valid trials (blue bars) at cue-target intervals of 800 ms and 900 ms can be attributed to the tendency of the saliency map to go back from preactivation to equilibrium level.

4.4.2 Discussion

Although quite simple, this saliency map model is able to accurately simulate the behavior of the barn owl under similar conditions. The results depicted in Figure 4.19 **b** and Figure 4.23 are strikingly similar (except for timescales), showing that the model — like the barn owl — shows a significant increase in response latency if cued on the wrong side.

Further results (not shown; see Röttsches (2004)) reveal more similarities: like the barn owl the model is able to discriminate between two competing sounds. Meaning

that the cued side is preferred if two sounds are played simultaneously on both sides.

Although not purely sound source localization related, this project built upon the work done on the ITD localization system and provides a link between the sensory-based work (bottom up) in this chapter and the more control oriented (top down) work in the next chapter. In this context, the model can be used to modulate the robot's attention by using an attentional cue. This cue can be directly derived from sensory perception, like for example, keeping the attention of the robot in the direction of a sound. The cue could also be derived from more abstract information, synthesized over time from various sensor modalities, like e.g. in robotic soccer, where the attention of the robot should be kept on the ball.

Chapter 5

Sound Source Localization on a Mobile Robot

This chapter is concerned with sound source localization on a mobile robot. As the interaural time differences-based localization method from the previous chapter is the method that is best understood, most evaluated and easiest to adapt to different microphone mountings, it is the only sound source localization algorithm used throughout this chapter.

Section 5.1 describes the first attempts with sound localization on a real robot. In these tests, the sound source localizer data is combined with the laser-based object recognition (described in Section 2.4.5) in order to raise the robot's attention to dynamic objects emitting sound. This work was previously published in Calmes et al. (2007b).

Section 5.2 describes a much more sophisticated method of combining sound source localization with laser-based object recognition, which is capable of tracking sound sources, dynamic objects and combinations of both over time.

5.1 Preliminary Tests: Sound Source Localization & Laser-based Object Recognition on a Robot

In a first approach to binaural hearing on a mobile robot, the ITD localization algorithm (Section 4.1.1) was combined with the laser-based object recognition (Section 2.4.5) in a very simple manner (Calmes et al., 2007b).

The basic idea was for the robot to only turn towards a sound source if there was a dynamic (i.e. laser-based) object associated with it. As the longterm goal is for the robot to respond to spoken commands from humans, the rationale behind the paradigm is that humans issuing commands should appear as sound sources with associated dynamic objects. Thus, sound sources combined with dynamic objects should be of special interest to the robot.

In these first experiments, a dynamic object was combined with a sound source if it was within a certain angular distance (or tolerance) of the sound source. This might appear overly simplistic, but these tests were mainly performed to gain some basic data and experience with sound source localization on mobile robots and to learn what kind of problems are to be expected.

Half of the experiments were performed with a fixed angular tolerance. Although the sound source localizer was set up to have linear 0.5° angular precision, in practice the accuracy decreases with more lateral source azimuths. This is why the other half of the experiments were conducted with a variable angular tolerance. With this adaptive

Table 5.1: List of positions of the loudspeakers and whether or not there was a dynamic object associated.

Speaker #	x	y	object
1	−1.00 m	1.00 m	yes
2	−0.15 m	1.75 m	no
3	−1.50 m	1.25 m	no
4	−1.75 m	−0.15 m	yes
5	2.25 m	−0.75 m	no
6	2.00 m	0.75 m	yes

Table 5.2: Performance evaluation for ATC and fixed angular resolution (%symmetric indicates the percentage of correct trials caused by front/back confusions due to the sound localizer)

	# of trials	%correct	%symmetric (of correct trials)
ATC	200	63.00	2.38
No ATC	200	57.50	8.70

tolerance control (ATC), angular tolerance was varied linearly between 5° (for a source azimuth of 0°) and 30° (for a source azimuth of ±90°), computed by

$$\mathrm{tol}_{\mathrm{ATC}} = 25^\circ \cdot \mid \frac{\mathrm{azimuth}_{\mathrm{src}}}{90^\circ} \mid +5^\circ. \tag{5.1}$$

5.1.1 Evaluation Setup

The experiments took place in the seminar room of the Department of Computer Science 5 at RWTH Aachen University. The room has a size of approximately 5 m × 10 m. The robot was placed at the center of this room at coordinates $(0, 0)$. Six sound sources (loudspeakers) were placed around the robot, three of which had a (dynamic) object associated to them. The coordinates of the sound sources are shown in Table 5.1. Loudspeakers 1, 4 and 6 were placed on cardboard boxes so that the robot's laser scanner could detect an object corresponding to these sources. Loudspeakers 2, 3 and 5 were mounted in such a way that no object could be detected. The evaluation setup is shown in Figure 5.1. Within this setup 4 evaluation experiments were conducted. An experiment consisted of 100 trials, where each trial consisted of randomly selecting a loudspeaker for stimulus playback. In order to achieve the best possible sound source localization performance (cf. Section 4.1.2), a broadband signal (random noise) was chosen as stimulus. The task of the robot was to turn towards an object if the source was associated with this object. We conducted two experiments (200 trials) with a fixed angular tolerance of 23° and two experiments (200 trials) with a varying tolerance value as described by Equation (5.1) above. In the following, we will refer to these experiments as non-ATC trials (fixed tolerance) and ATC trials (adaptive tolerance), respectively.

5.1.2 Results

Table 5.2 shows the results of the experiments. There were three cases in which a trial was considered as being correct:

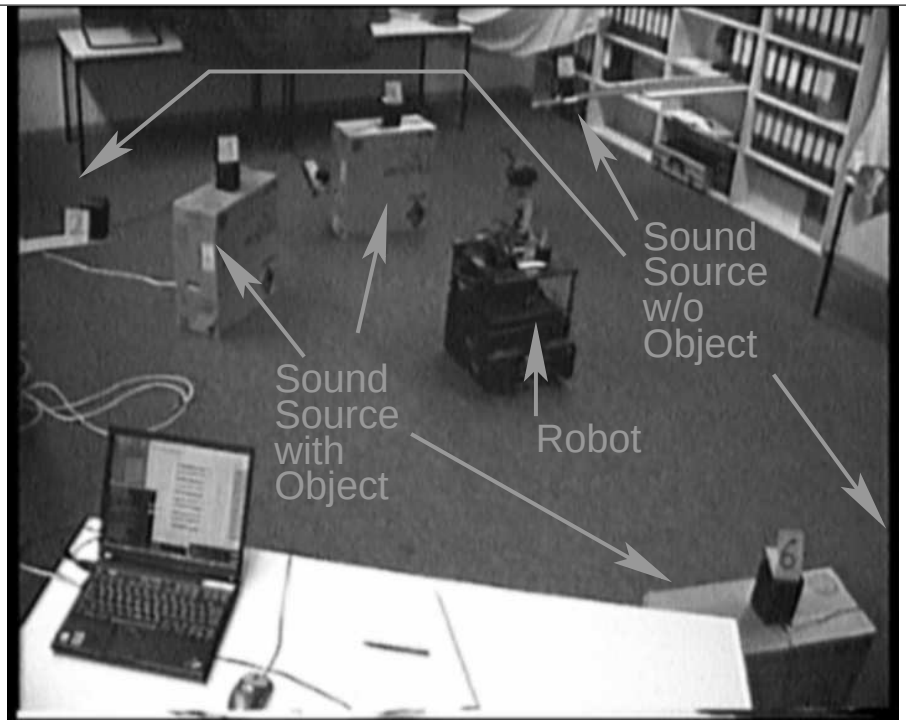


Figure 5.1: Preliminary tests setup

1. No object was associated with the source emitting a sound and the robot did not select any target.

2. There was an object associated with the source emitting the sound cue and the robot selected that object (with the given angular tolerance) as its target.

3. Either there was an object associated with the source or there was no object associated but one on the opposing side of the robot. Then the robot selected an object symmetric to the source (front/back confusions) as target.

All relevant state data from the sensors were logged as well as the generated stimuli and the motion commands issued to the robot. As can be seen in Table 5.2, the overall accuracy of the system is not very high, although the system managed to produce a correct response to the given stimulus in more than 50 % of the trials. A slight improvement could be achieved with the adaptive tolerance control algorithm (Eq. (5.1)) in comparison to the fixed tolerance value. In the following sections, the system's performance will be analyzed in more detail.

Sound source localization performance

In order to assess how good the sound source localization subsystem performed within the evaluation, real source positions (with respect to the microphone assembly) were plotted against the azimuths returned by the localization system for all trials (non-ATC trials and ATC trials pooled together). These data are shown in Figure 5.2. From this, it becomes evident that the sound source localization system did not perform very well, especially when one compares Figure 5.2 to the results depicted in Figure 4.7. Because of the differing conditions (larger room, larger distance to sound sources), we did not expect as high a precision as for the broadband noise in Figure 4.7. Still, we were

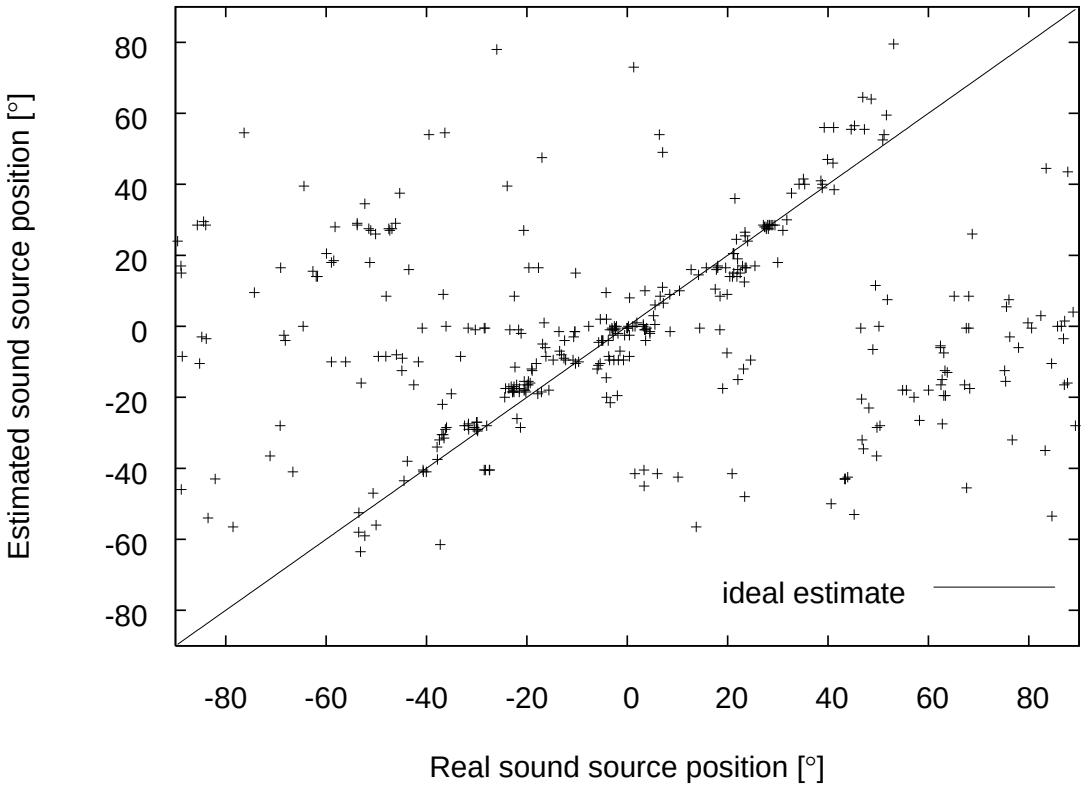


Figure 5.2: Real vs. estimated sound source azimuth (with respect to the microphones) of all trials, as returned by the sound source localization system (non-ATC and ATC trials pooled together).

surprised by the low performance. We will address this issue again in the discussion at the end of this section.

For almost all absolute sound source azimuths above 55° the detection error was greater then 25°. We already mentioned that the sound localizer's accuracy decreases with increasing laterality of the source azimuth. However, this cannot be the only reason for the rather weak performance of the sound localizer in this evaluation setup, as there are also significantly high errors in the detection for source azimuths less than 45°.

It will now be shown that the additional information about dynamic obstacles can, at least partly, make up for the sound localizer's performance.

Object association performance

To assess the correctness of the object association method, for all correct trials, the azimuths of the selected objects relative to the microphones were plotted against the associated sound localization estimates (non-ATC and ATC trials pooled together). This is shown in Figure 5.3. In this case, estimated sound source positions correspond well with the target objects. Deviations from the correct azimuths are consistently within the limits of the respective angular tolerance applied for each trial.

As one can see the robot was able to identify and make up for the low reliability of sound source localization estimates with a fairly simple sound source / laser-based object association scheme. By only allowing object associations within a certain angular tolerance, output from the sound source localizer with a large error could be eliminated successfully. For one, this is a cheap way to determine whether the sound source lo-

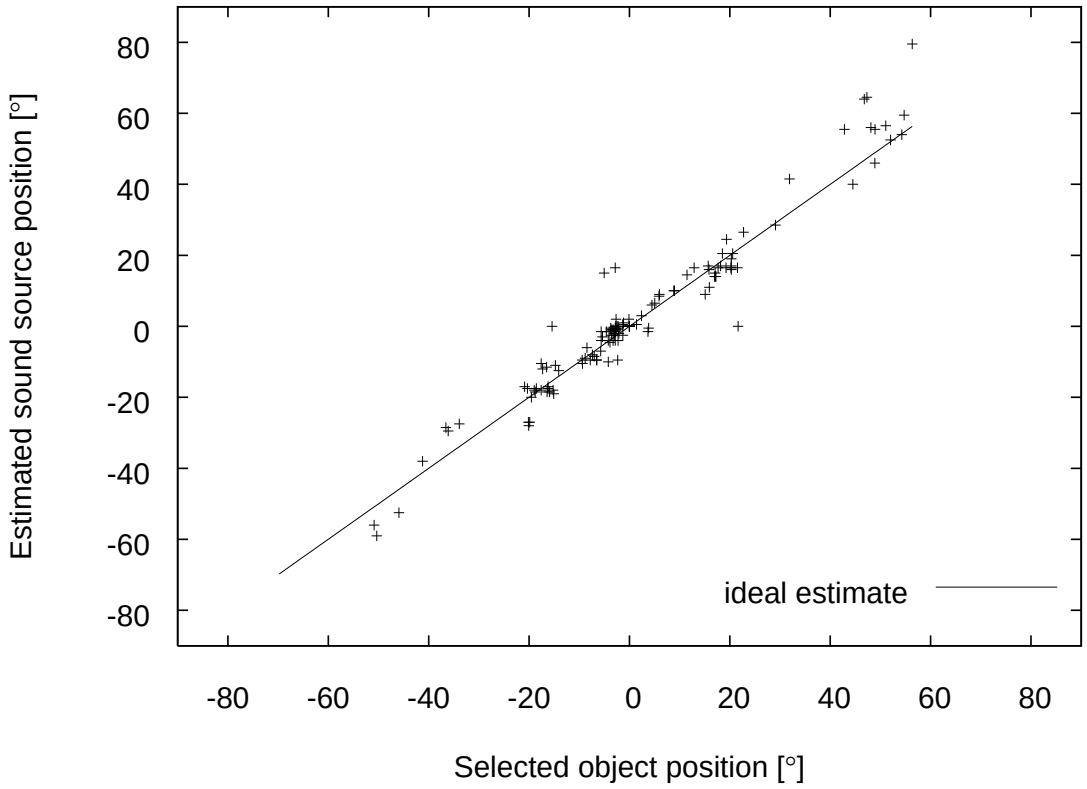


Figure 5.3: Real azimuths of correctly selected targets vs. associated estimated sound source azimuths as provided by the sound source localization (relative to the microphones, non-ATC and ATC trials pooled together).

calization works correctly. For another, in some cases symmetric confusion could be resolved by combining the sound sources with dynamic objects. However, there have also been erroneous associations with alleged symmetric objects (%symmetric column in Table 5.2).

5.1.3 Discussion

These very simple experiments show that, in order to use sound source localization effectively in realistic environments for mobile robotics applications, the acoustic information has to be combined with data from other sensor modalities. In this sense, the unreliable behavior of the sound source localization algorithm in this case might well have been a blessing in disguise. With a sound source localization system in good working order, the experiments would not have yielded such interesting results. As it is, only the combination of object recognition and sound source localization makes it possible for the robot to detect and eliminate errors in estimated sound source positions.

The question remained why the sound source localization system did not perform well in this experimental setup. The initial evaluation of the dual delay-line algorithm (Section 4.1.2) showed that, although the ITD-based algorithm can be very accurate, it is sensitive to reverberations. The room in which the experiments took place is larger than any in which the system had been tested before and relatively empty. This leads to perceivable reverberations which could account for (some of) the error.

Furthermore, previous experiments had all been conducted with no obstruction be-

tween the microphones. On the robot, the two microphones were mounted on opposite sides of a plastic tube with a diameter of approximately 13 cm. This might have altered ITDs in a frequency-dependent way, as from a critical frequency upwards, the sound wave would have to bend around the tube to reach the second microphone.

Subsequent measurements of the head-related transfer functions of the robot's microphone mount did not show any significant changes in the ITDs which could account for the bad localization performance.

Room simulation experiments similar to those performed in Section 4.1.2 finally revealed the truth. As reverberations proved to be a problem with bandlimited signals in smaller rooms, they (unsurprisingly) also affected broadband signals in larger rooms, leading to a corruption of the ITDs that threw off the sound source localizer on the robot.

Building on these preliminary tests, the next step was to devise a more advanced method of combining the two sensor modalities that are sound source localization and laser-based object recognition. This algorithm is described in the next section.

5.2 Tracking Objects and Sound Sources Using Markov Chain Monte Carlo Data Association and a Virtual Sensor

In this section, we present the method used for tracking laser-based dynamic objects (as described in Section 2.4.5) and sound sources. In its current state, it is able to keep track of three kinds of entities, based on the data from two different sensor modalities:

1. Sound sources alone.

2. Laser-based dynamic objects.

3. Combined entities, consisting of *one* sound source and *one* laser-based dynamic object.

The method is based on the *Markov chain Monte Carlo data association* (MCMCDA) algorithm (Oh et al., 2004), adapted to dealing with multiple sensor modalities that provide heterogenous data.

5.2.1 Problem Description

The problem consists of keeping track, during an observation interval of length $T \in \mathbb{N}$, of an unknown number N of entities, or targets.

These targets move through the observation space for specific time intervals $[t_i^n, t_f^n]$ with $n = 0, \ldots, N-1$ and $1 \leq t_i^n < t_f^n \leq T$. As the robot's internal representation of the environment is mainly two-dimensional (through the occupancy grid; see Section 2.4.4), it is assumed that the entities to be tracked move in a plane (i.e. their position can be represented by two-dimensional coordinates). The appearance of targets is governed by a Poisson distribution[1] with the parameter $\lambda_b A$. Here, A is the area (in m^2) of the

[1]The Poisson distribution expresses the probability of a number of independent occurrences of an event in a fixed interval (which can represent time, distance, area, ...) with an expected number of occurrences λ . The probability of exactly $k \in \mathbb{N}_0$ occurrences in a given interval is $f(k;\lambda) = \frac{\lambda^k e^{-\lambda}}{k!}$.

observation space (basically the size of the floor plan of the room containing the robot[2]) and λ_b the birth rate of targets per unit time, per unit area.

At each observation time t ($1 \leq t \leq T$), a number n_t of observations $y_1^t, y_2^t, \ldots, y_{n_t}^t$ are collected from the (virtual) sensor (the virtual sensor will be described in Section 5.2.2).

It is assumed that, at each observation timestep, some of the n_t data provided by the sensor will be false alarms. These follow a Poisson distribution with parameter $\lambda_f A$, where λ_f is the false alarm rate per unit time and unit area.

Targets can be detected by the system with a detection probability p_d. At any given time, a target will stay within the observation space with the probability $(1 - p_z)$ and disappear with probability p_z.

One of the design goals of the tracking system was that it should not show a preference to a specific sensor modality. This means that, with only sound source azimuths available, it should degenerate to a sound source azimuth tracker. Conversely, if only object positions are available, the system should degenerate to an object tracker.

5.2.2 Sensor Fusion Using a Virtual Sensor

In order to keep the modifications to the original algorithm as minimal as possible, an approach was chosen that presented a homogeneous type of data to the tracking system, regardless of the sensor that provided the data (laser-based object recognition or sound source localization).

To this effect, a 'virtual sensor' was developed, taking as input the heterogeneous data from the different sensors and providing a single type of data — called *observation* — as output.

As the sound source localizer can only provide azimuth data on sound sources relative to the robot's position, it was natural to choose robot-centric polar coordinates (instead of map-centric Cartesian coordinates) to represent object positions.

A single observation from the virtual sensor is a tuple containing the polar coordinates r and θ, a timestamp t, the type of observation *type* and the so-called combination probability $CombProb$:

$$observation = (r, \theta, t, type, CombProb). \tag{5.2}$$

As it was not possible to make the MCMCDA algorithm completely oblivious to the kind of data it receives as input, the information on the data source is passed via the *type*-component of an observation. This component can have three possible values:

S The observation is of the **S**ound-only type, meaning it contains the azimuth to a sound source provided by the sound source localization system. In this case, the r-component is always set to 0, which is not a problem as it will receive special treatment later on (see end of subsection *Posterior computation* of Section 5.2.3).

O The observation is of the **O**bject-only type, meaning it contains coordinates of an object detected by the laser-based object recognition.

SO The observation is of the **S**ound/**O**bject combination type, meaning the virtual sensor associated a sound source to a dynamic object.

As it is, the virtual sensor will relay any detected sound-source azimuth as an **S**-type observation and any detected laser-based object as an **O**-type observation to MCMCDA.

[2]This can easily be deduced from the robot's navigation map.

In order to associate sound sources to objects, adaptive tolerance control (ATC, see Section 5.1) is used. If there are several detected sound sources that can be combined with a given object (according to ATC), the association will only be performed with the source that has the minimal angular distance to the dynamic object.

This is where the aforementioned combination probability comes into play. Initially, for **S** and **O**-type observations, it is set to a default value. It is only altered if the **S** and **O**-type observations are used to generate a combined observation (**SO**-type). The combination probability is computed using the following equation:

$$cp = 1 - \frac{\text{abs}(\theta_S - \theta_O)}{\pi}, \tag{5.3}$$

where θ_S is the azimuth of the **S**-type observation and θ_O is the azimuth component of the **O**-type observation (both values in rad). The value of cp is 1 for $\theta_S - \theta_O = 0$, meaning that if the sound source and the object coincide perfectly in azimuth, the combination probability is maximal. Equation 5.3 equals 0 for $\theta_S - \theta_O = \pi$, i.e. the combination probability is minimal if the sound source and the object azimuth are separated by the maximal possible angular distance (namely π).

The *CombProb* component of the **SO** observation is set to the value cp. The *CombProb* components of the **S** and **O**-type observations used to assemble the **SO**-type observation are set to the value $1 - cp$.

The **SO**-type observations as well as the original **S** and **O** observations are passed to the MCMCDA algorithm. The *CombProb* component ensures that, in the later step of the computation of the a-posteriori probability of each track, **SO**-type observations are preferred over the other two types of observations.

Passing all three observations (**SO** and its constituting **S** and **O**) introduces a problem, however. We assume that, at a given spatial position, we can only detect one kind of observation. Passing all three observations to MCMCDA introduces false alarms, because there can be more than one observation at a given position, which MCMCDA should not include into tracks. In order to alleviate the influence of these *virtual* false alarms, the virtual sensor additionally provides MCMCDA with a *false alarm offset* at each timestep.

Furthermore, as the coordinates stored in an observation are relative to the robot's position at the time the observation was made, the virtual sensor also provides MCMCDA with the robot's position at that timestep:

$$RobPos_t = (x_t, y_t, ori_t), \tag{5.4}$$

where x_t and y_t are the robot's coordinates relative to the map origin and ori_t is the robot's orientation in relation to the map's 0 angle. With the help of the robot's position, it is easy to transform an observation's relative (robot-centric) coordinates into absolute (map-centric) coordinates.

5.2.3 Markov Chain Monte Carlo Data Association (MCMCDA) Algorithm

We will now describe the Markov chain Monte Carlo data association algorithm (Oh et al., 2004), whose structure is shown in Algorithm 5.1.

The algorithm can be decomposed into three parts, which we will describe independently:

1. Sample generation.

Algorithm 5.1: Markov chain Monte Carlo data association (MCMCDA). The input is a set of observations Y, an number of samples n_{smpls} and an initial trackset ω_0. The output is a trackset $\hat{\omega}$ of estimated observations-to-tracks associations.

Input: $Y, n_{\text{smpls}}, \omega_0$
Output: $\hat{\omega}$
$\omega \leftarrow \omega_0$
for $n = 1$ to n_{smpls} **do**
  sample m from $\chi(\,\cdot\,)$
  propose ω', based on ω and m
  sample α from $\mathrm{U}(0, 1)$
  if $\alpha < A(\omega, \omega')$ **then**
    $\omega \leftarrow \omega'$
  end if
end for
$\hat{\omega} \leftarrow \omega$

2. Metropolis-Hastings algorithm.

3. Posterior computation.

Sample generation

A sample of MCMCDA algorithm consists of a whole trackset ω. This represents one hypothesis on the correct assignment of observations to tracks.

A trackset contains individual tracks τ, i.e. $\omega = \{\tau_0, \ldots, \tau_K\}$. If $Y_t = \{y_t^1, \ldots, y_t^{n_t}\}$ is the set of observations available at timestep t and $Y = \bigcup_{t \in \{1,\ldots,T\}} Y_t$ is the set of all observations, then a track is defined as follows:

1. A track has n elements: $\tau = \{y_{t_i} | 1 \leq i \leq n\}$.

2. At each timestep t_i represented in the track, there can only be one observation y_{t_i} assigned to track τ.

3. $|\tau| \geq 2$. A track contains at least two observations, as only one observation could result from a false alarm.

4. τ_0 is the set of false alarms under the current hypothesis ω, i.e. the set of all observations that could not be assigned to any regular track in the current trackset.

Additionally, tracks exhibit the following properties:

1. $\bigcup_{k=0}^{K} \tau_k = Y$.

2. $\tau_k \cap \tau_l = \emptyset$ for $k \neq l$.

A trackset ω is generated and modified using one of eight operations: *birth*, *death*, *split*, *merge*, *extend*, *reduce*, *update*, *switch*.

One of these 'moves' is randomly selected for execution by the algorithm during one sample iteration (see Algorithm 5.1). Except for two cases, all of these moves can always be selected:

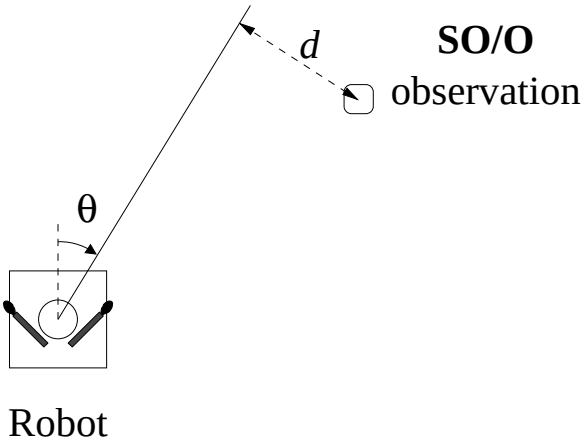


Figure 5.4: Distance d from an **SO**/**O**-type observation to an **S**-type observation (represented by the azimuth θ).

1. If there is not yet any track in the trackset ω, only the birth move is possible.
2. If there is only one track, all moves except for *switch* and *merge* are possible (those moves require at least two tracks).

Before explaining the individual track moves in detail, we first have to explain the concept of a candidate observation. Some of the moves assign new observations from the set of unassigned observations to a track τ. Starting from an observation y_t already present in τ, the move tries to find an adequate successor observation.

A candidate observation is an adequate successor to y_t, if:

1. It is only $d < d_{\max}$ timesteps from y_t away, with $d_{\max}$ being the maximal accepted interval during which it is allowed to not have any observations for a given track.
2. Its distance from y_t is less than $v_{\max} \cdot d_{\max}$, with $v_{\max}$ being the maximal directional speed allowed for any object.

More formally, the set of all possible successor observations to observation y_t at time $t + d$ is

$$succ_d(y_t) = \{y^j_{t+d} \in Y_{t+d} | \mathrm{dist}(y_t, y^j_{t+d}) < d_{\max} \cdot v_{\max}\}. \tag{5.5}$$

Distance computation In order to compute the distance $\mathrm{dist}(y_t, y_{t+d})$ between two observations in Equation 5.5, three cases have to be considered, depending on the respective types of the observations:

1. *SO/O ↔ SO/O:* If both observations are of the **SO** or **O** type, their coordinates are transformed to absolute coordinates with the help of the corresponding robot positions. As the r and θ components are available for both positions, the distance is simply the Euclidean distance between the observations.
2. *SO/O ↔ S:* If one of the candidates is of the **SO** or **O** types and the other observation is of the **S** type, the distance computation is more complex. In this case, the distance is the distance between the point represented by the **SO**/**O** observation (in absolute coordinates) and the line going through the robot's position at the time of the **S** observation. The azimuth of the **S** observation is the angle of the line relative to the robot (see Figure 5.4).

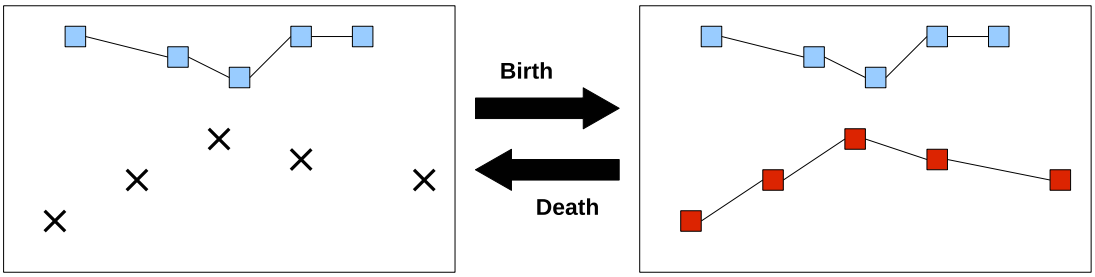


Figure 5.5: Birth and Death moves. Squares connected by lines show observations assigned to tracks. Crosses show false alarms (after Oh (2008)).

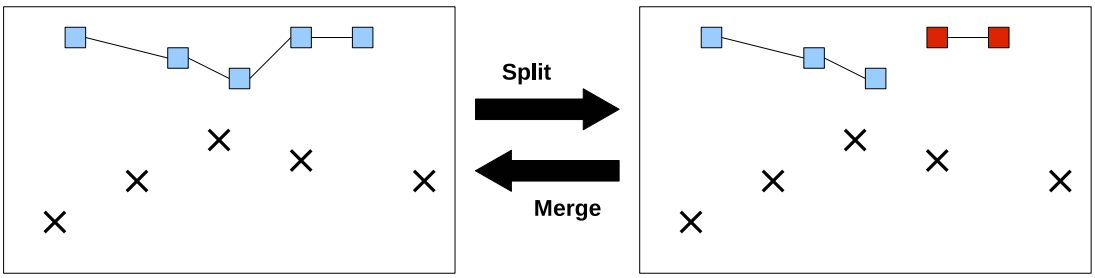


Figure 5.6: Split and Merge moves. Squares connected by lines show observations assigned to tracks. Crosses show false alarms (after Oh (2008)).

3. $S \leftrightarrow S$: If both observations are of the **S**-type, the distance between the observations is simply the absolute value of the angular distance between the azimuths (corrected by the respective robot orientations) of the observations.

We will now describe in detail the eight track moves, which are used for creating and modifying tracks.

BIRTH MOVE (Figure 5.5). During a birth move, an observation y_t^j is selected at random from τ_0, at a random time t ($1 \leq t \leq T$). This is (potentially) the first observation from the new track. Then, a distance factor d is chosen from an appropriate distribution[3] and the set $succ_d(y_t^j)$ is generated. If $succ_d(y_t^j) = \emptyset$, the birth move fails, as a valid track requires at least two observations. Otherwise, the two observations are removed from τ_0 and added to a new track.

Starting at the second observation in the new track, the birth move tries to find a new successor in the way described for the first element and if successful, adds it to the track. This is repeated until no more successors are found. The new track is added to ω and the Birth move is accepted.

DEATH MOVE (Figure 5.5). For a death move, a track $\tau_i \in \omega$ ($i > 1$) is randomly selected and deleted. The observations that were assigned to τ_i are reassigned to τ_0. The death move is the only move that is always accepted, as it is only called if $|\omega| > 1$ (ω always contains τ_0, so $|\omega|$ is at least 1).

[3]We generated d by sampling a value x from a normal distribution $\mathcal{N}(0, \sigma^2)$ with $\sigma^2 = \frac{\min(d_{\max}, T-t)-1}{3.9}$ and setting $d = 1 + |x|$.

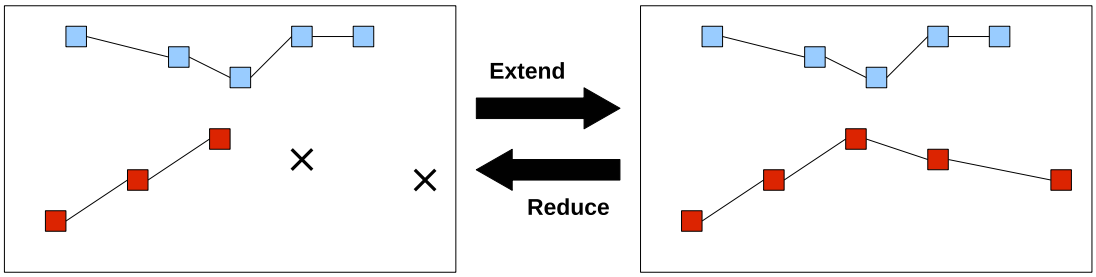


Figure 5.7: Extend and Reduce moves. Squares connected by lines show observations assigned to tracks. Crosses show false alarms (after Oh (2008)).

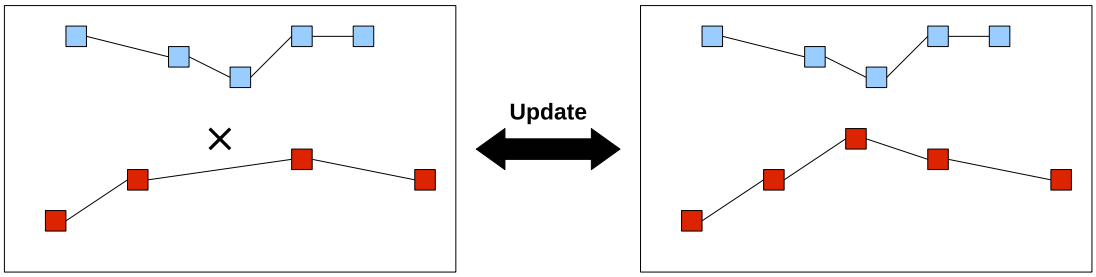


Figure 5.8: Update move. Squares connected by lines show observations assigned to tracks. Crosses show false alarms (after Oh (2008)).

Split move (Figure 5.6). In a split move, a track $\tau_i \in \omega$, $(1 \leq i \leq |\omega|)$ is selected, with $|\tau_i| \geq 4$. If no such track exists, the move is rejected. Otherwise, let $K = |\tau_i|$. An observation $y_{t_k} \in \tau_i$ $(2 \leq k \leq K-2)$ is selected. τ_i is split at y_{t_k} and a new track, τ_j is created with $\tau_i = \{y_{t_1}, \ldots, y_{t_k}\}$ and $\tau_j = \{y_{t_{k+1}}, \ldots, y_{t_K}\}$. For a split move to be considered, ω has to contain at least one track besides τ_0.

Merge move (Figure 5.6). For a merge move, two tracks, $\tau_i \in \omega$ and $\tau_j \in \omega$, are selected. Let $K = |\tau_i|$. For the move to accept, $y^j_{t_1} \in succ_d(y^i_{t_K})$ has to be true, i.e., the first observation from τ_j has to be in the successor candidate set of the last observation from τ_i. In this case, the observations from τ_j are appended to τ_i and τ_j is subsequently removed from ω.

Extend move (Figure 5.7). In an extend move, a track $\tau_i \in \omega$ is selected. Let $K = |\tau_i|$. The extend move tries to find successors to $y_{t_K} \in \tau_i$ as done during the birth move.

Reduce move (Figure 5.7). The reduce move selects a track $\tau_i \in \omega$, with $|\tau_i| > 2$. Let $K = |\tau_i|$. An observation $y_{t_k} \in \tau_i$ (with $2 \leq k \leq K-1$) is randomly chosen and the track is reduced to k elements, i.e. $\tau_i = \{y_{t_1}, \ldots, y_{t_k}\}$. The discarded observations $(y_{t_{k+1}}, \ldots, y_{t_K})$ are inserted into τ_0.

Update move (Figure 5.8). An update move chooses $\tau_i \in \omega$ and selects an observation $y_{t_k} \in \tau_i$. The update move tries to find new successor observations to y_{t_k} in τ_0, the same way as is done in a birth move. Possibly discarded observations are reinserted into τ_0.

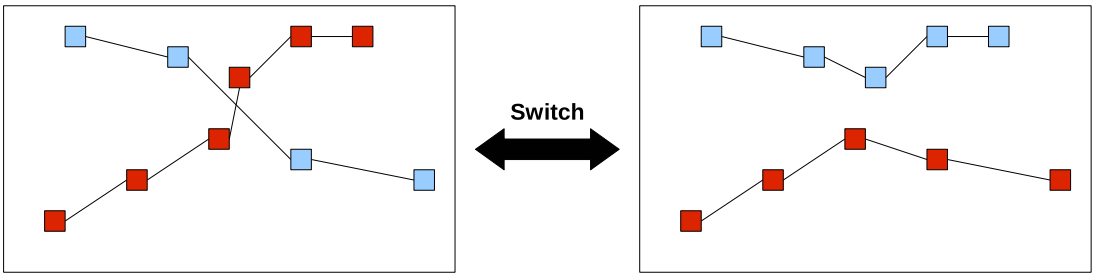


Figure 5.9: Switch move. Squares connected by lines show observations assigned to tracks. Crosses show false alarms (after Oh (2008)).

SWITCH MOVE (Figure 5.9). Two tracks $\tau_i = \{y^i_{t_1}, \ldots, y^i_{t_k}, y^i_{t_{k+1}}, \ldots, y^i_{t_{K_i}}\}$ and $\tau_j = \{y^j_{t_1}, \ldots, y^j_{t_l}, y^j_{t_{l+1}}, \ldots, y^j_{t_{K_j}}\}$ are selected in a switch move ($\tau_i, \tau_j \in \omega$). With $K_i = |\tau_i|$ and $K_j = |\tau_j|$. The switch move tries to find two observations $y^i_{t_k} \in \tau_i$ and $y^j_{t_l} \in \tau_j$, such that $y^i_{t_{k+1}} \in succ_d(y^j_{t_l})$ and $y^j_{t_{l+1}} \in succ_d(y^i_{t_k})$. If these observations are found, the move is accepted, and the corresponding track segments are swapped, i.e. $\tau_i = \{y^i_{t_1}, \ldots, y^i_{t_k}, y^j_{t_{l+1}}, \ldots, y^j_{t_{K_j}}\}$ and $\tau_j = \{y^j_{t_1}, \ldots, y^j_{t_l}, y^i_{t_{k+1}}, \ldots, y^i_{t_{K_i}}\}$. Taking Figure 5.9 as an example, the blue track being τ_i and the red track τ_j. In that case, $y^i_{t_k}$ would be the second element of the blue track (counting from the left). Similarly, $y^j_{t_l}$ would be the third element of the red track (counting from the left).

Metropolis-Hastings algorithm

The basic structure of MCMCDA is that of the classic Metropolis-Hastings algorithm for Markov chain Monte Carlo (see Section 2.5). In this sense, it tries to sample from a probability distribution corresponding to the equilibrium distribution of a Markov chain. In MCMCDA, the individual states ω of the Markov chain correspond to tracksets, i.e. a set of individual target tracks τ during a given observation interval.

The value $A(\omega, \omega')$ from Algorithm 5.1 corresponds to the value α as described by Eq. 2.39 in Section 2.5.3:

$$A(\omega, \omega') = \min\left(\frac{P(\omega'|Y)Q(\omega|\omega')}{P(\omega|Y)Q(\omega'|\omega)}, 1\right), \tag{5.6}$$

where $P(\omega|Y)$ ($P(\omega'|Y)$, respectively) is the posterior probability of the trackset ω (ω', respectively), given the set of all measurements Y up to and including the current time. $Q(\,\cdot\,)$ is the proposal density with $Q(y|x)$ being the probability of going from trackset x to trackset y. This is basically the uniform distribution, because the move which generates a new trackset is selected uniformly from the set of possible moves. However, the size of the set of possible moves depends on the size of the trackset x (the number of tracks in x):

$|x| = 0:\ Q(y|x) = 1$, because the birth move is the only possible move.

$|x| = 1:\ Q(y|x) = 1/6$, as every move, except for *switch* and *merge* is possible.

$|x| \geq 2:\ Q(y|x) = 1/8$, because every move is possible.

Posterior computation

The posterior of a trackset ω, given the measurements Y, is computed as follows:

$$P(\omega|Y) = \frac{1}{Z}\prod_{t=1}^{T} p_{\mathrm{z}}^{z_t}(1-p_{\mathrm{z}})^{c_t} p_{\mathrm{d}}^{d_t}(1-p_{\mathrm{d}})^{u_t}\lambda_{\mathrm{b}}^{a_t}\lambda_{\mathrm{f}}^{f_t} \times \prod_{\tau\in\omega\setminus\{\tau_0\}} \mathcal{N}(\tau(t_{i+1})|\bar{x}_{t_{i+1}}(\tau), B_{t_{i+1}}(\tau)). \quad (5.7)$$

The first part of the posterior expression (the first Π in Eq. 5.7) implements the target model, estimating how probable it is to have the number of tracks present in ω, what the probability of the tracks starting and ending at specific times is and the probability of the number of false alarms and missing observations.

Here, z_t is the number of tracks terminated at time t and a_t is the number of new tracks. This means that $p_z^{z_t}$ is the probability of having z_t terminated tracks at time t and $\lambda_{\mathrm{b}}^{a_t}$ is the probability of a_t new tracks appearing at time t, given the birth rate λ_b.

Let e_t be the number of targets from time $t-1$. Then $c_t = e_t - z_t$ is the number of tracks from the previous timestep, still present at the current timestep and hence $(1-p_{\mathrm{z}})^{c_t}$ is the probability of c_t tracks remaining from the previous timestep.

d_t is the number of detections at time t and $u_t = c_t + a_t - d_t$ is the number of undetected tracks at t. Consequently, $p_{\mathrm{d}}^{d_t}$ is the probability of detecting d_t targets and $(1-p_{\mathrm{d}})^{u_t}$ is the probability of u_t undetected targets.

f_t is the number of false alarms ($f_t = n_t - d_t$), hence $\lambda_{\mathrm{f}}^{f_t}$ is the probability of having f_t false alarms at time t.

The second part of the posterior (the second Π in Eq. 5.7) consists of the a posteriori probabilities of the individual tracks in the trackset. It is assumed that the target movement follows a linear function with Gaussian noise. The probability of an individual track, then, is the goodness of fit of the track elements at each timestep t, compared to the estimated positions of the track at time t. These estimates are generated by a Kalman filter (Section 2.4.3). Here, $\mathcal{N}$ is a multivariate normal distribution whose mean vector $\bar{x}(\tau)$ and covariance matrix $B(\tau)$ are the estimated state and the covariance of the innovation of the Kalman filter, applied to τ.

In order for the Kalman filter to be applicable, a system model and a measurement model of the targets have to be specified (Section 2.4.3).

If a target n is observed k times at the discrete points in time $t_0, t_1, \ldots, t_{k-1}$, the system model is as follows:

$$x_{t_{i+1}}^n = \begin{pmatrix} 1 & 0 & \Delta_t & 0 \\ 0 & 1 & 0 & \Delta_t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot x_{t_i}^n + \begin{pmatrix} \frac{\Delta_t^2}{2} & 0 \\ 0 & \frac{\Delta_t^2}{2} \\ \Delta_t & 0 \\ 0 & \Delta_t \end{pmatrix} \cdot w_{t_i}, \quad (5.8)$$

with $\Delta_t = t_{i+1} - t_i$ and $i = 0, \ldots, k-1$. The state $x_t = \begin{pmatrix} r_t & \theta_t & \dot{r}_t & \dot{\theta}_t \end{pmatrix}^T$ contains the polar coordinates as well as their first derivatives, i.e. the translational and angular velocities. Considering this, it can easily be seen that equation 5.8 implements the classic equations of motion with the noise component (w_{t_i}) providing the acceleration values. The following measurement model is used, specifying how an observation is related to a state:

$$y_{t_i} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \cdot x_{t_i} + v_{t_i}, \quad i = 0, \ldots, n., \quad (5.9)$$

with $y_t = \begin{pmatrix} r_t & \theta_t \end{pmatrix}^T$ being an observation vector. In Equations 5.8 and 5.9, w_{t_i} and v_{t_i} are Gaussian noises, representing the system and measurement noise with covariance matrices Q and R, respectively.

As was mentioned in Section 5.2.2, the coordinates stored in an observation are polar, robot-centric coordinates. With the help of the robot position $RobPos_t$ (also provided by the virtual sensor, see Section 5.2.2), the observations are transformed to absolute coordinates before applying the Kalman filter.

This simplifies Kalman filtering, but introduces another problem. As some of the observations were obtained by the sound source localization system alone (**S**-type), their polar coordinates lack a meaningful r component (the virtual sensor simply sets $r = 0$). As there is no easy solution, a simple heuristic is applied:

1. If the track consists only of **S**-type elements up to the observation's timestep, set the observation's r component to the minimal realistic distance, which we defined to be 0.5 m. We assumed that no human would get closer to the robot. Furthermore we wanted to include a wider range of angular velocities. With a higher value of r, only sources with low angular speeds would be considered realistic.

2. If the track already contains **O** or **SO** components up to the current time, set the observation's r component to the r component of the Kalman-predicted state in relative coordinates.

5.2.4 Results

We will now present the results of the simulation experiments performed with the modified MCMCDA algorithm. Unfortunately, no real world tests could be done, for reasons which will be detailed further in the discussion section below.

Five different simulation setups were used. They will be presented and described side by side with the corresponding results. The details of the MCMCDA and simulation experiment parameters can be found in the tables in Appendix C.

For reasons of simplicity, the results were plotted in figures with the x axis being time and the y axis being azimuth in absolute coordinates (i.e. not robot-centric). Although this discards distance information (where available), we estimate that the simple nature of the simulation experiments (i.e. target movements) ensures that the provided information is still enough to give an overview on the algorithm's performance. With distance information included, three dimensional plots become necessary, which are not so intuitive and difficult to interpret.

For each experiment, two simulation data sets were generated. In one data set, the positions of sound sources and objects directly corresponded to their 'real' simulated positions. In the other data set, noise was added to each position datum. The value added to the 'real' position was sampled from a normal distribution with mean $\mu = 0$ and standard deviation $\sigma = 0.5$ for sound source localization azimuths, $\sigma = 0.005$ for x/y coordinates of objects and robot and $\sigma = 0.0025$ for robot orientation.

Experiment 1 was designed to prove the general validity of the algorithm. There are two objects and two sound sources present, with one object and one sound source together at 0° of azimuth, relative to the robot (Figure 5.10, top and Table C.2). None of the entities moved during the experiment, i.e. it was completely static. Ideally, the algorithm should generate three separate tracks, one for the sound source, one for the combined sound source/object, and one for the object.

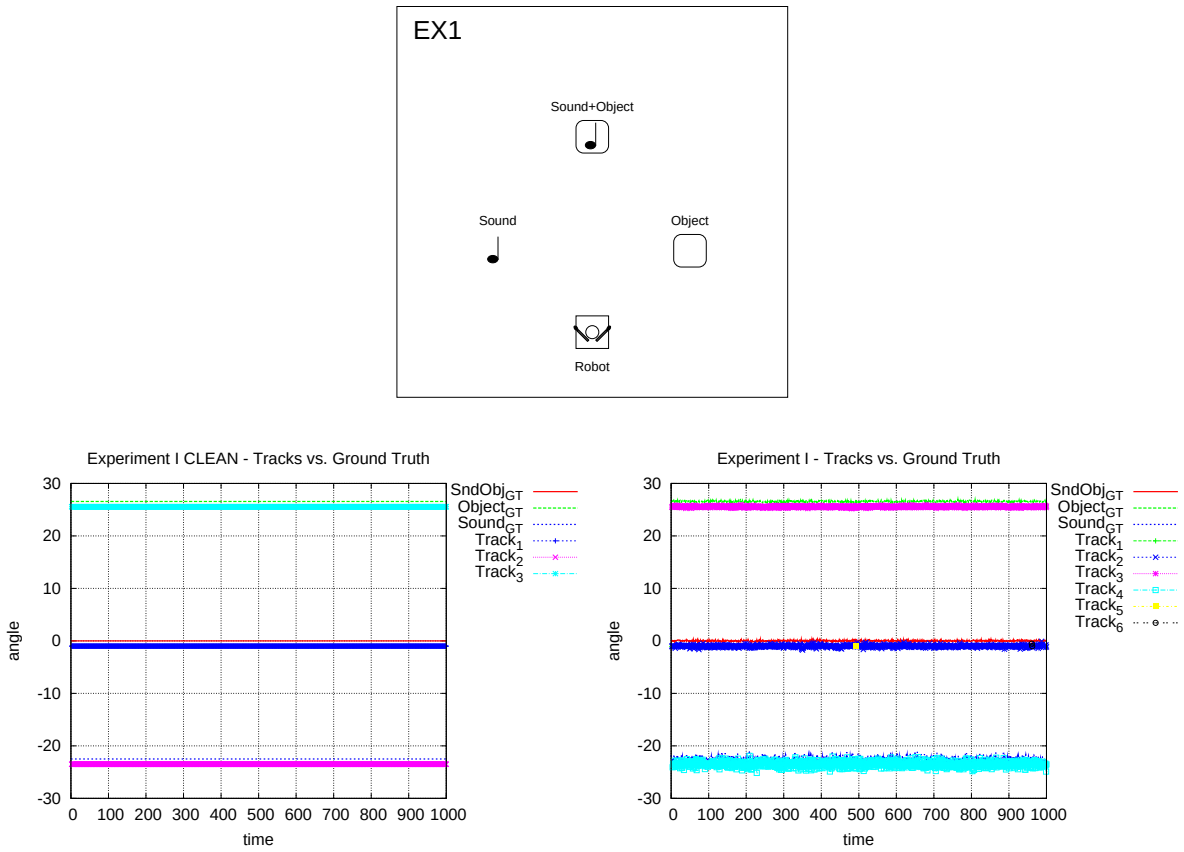


Figure 5.10: Simulation Experiment 1. **Top:** Setup. **Bottom left:** Results on clean dataset. **Bottom right:** Results on noisy dataset.

The outcome of the experiment for the clean data set is shown in Figure 5.10, bottom, left, whereas the result of the experiment on the noisy data set is in Figure 5.10, bottom, right.

The known positions of the sound sources and objects (the ground truth) are labeled $\mathsf{SndObj_{GT}}$, $\mathsf{Object_{GT}}$ and $\mathsf{Sound_{GT}}$ in the legend. In the figures, the actual track data (as computed by MCMCDA) are shown slightly offset in relation to ground truth. This is not a bias induced by the tracker, but this has been done deliberately, for clarity. Otherwise the ground truth data would be covered by the tracks and thus become invisible.

On the clean data set (Figure 5.10, bottom left), MCMCDA generates three tracks ($\mathsf{Track_1}, \ldots, \mathsf{Track_3}$), corresponding exactly to the simulated entities.

In the case of the noisy data (Figure 5.10, bottom right), the results are (almost) as encouraging. Here, six tracks ($\mathsf{Track_1}, \ldots, \mathsf{Track_6}$) were created. $\mathsf{Track_2}$ corresponds exactly to the **SO**-entity at 0° azimuth ($\mathsf{SndObj_{GT}}$). $\mathsf{Track_3}$ is the object at about 25° ($\mathsf{Object_{GT}}$) and $\mathsf{Track_4}$ represents the sound source at −22° ($\mathsf{Sound_{GT}}$). $\mathsf{Track_1}$, $\mathsf{Track_5}$ and $\mathsf{Track_6}$ are very short, spurious tracks, hardly visible in the figure. Interestingly, all three correspond to the **SO**-entity (sound/object combination) at 0°. This point will be further addressed in the discussion.

Experiment 2 (Figure 5.11, top and Table C.3) was designed to simulate the real-life case of two people (two objects) engaged in a dialogue (sound source at objects' positions alternately switching on and off, respectively alternating between the two objects).

The correct behavior of the tracking system would be to assign an individual track to each object, and not be fooled by the sound source shifting place. The outcome of this experiment is shown in Figure 5.11, bottom left (clean data) and bottom right (noisy

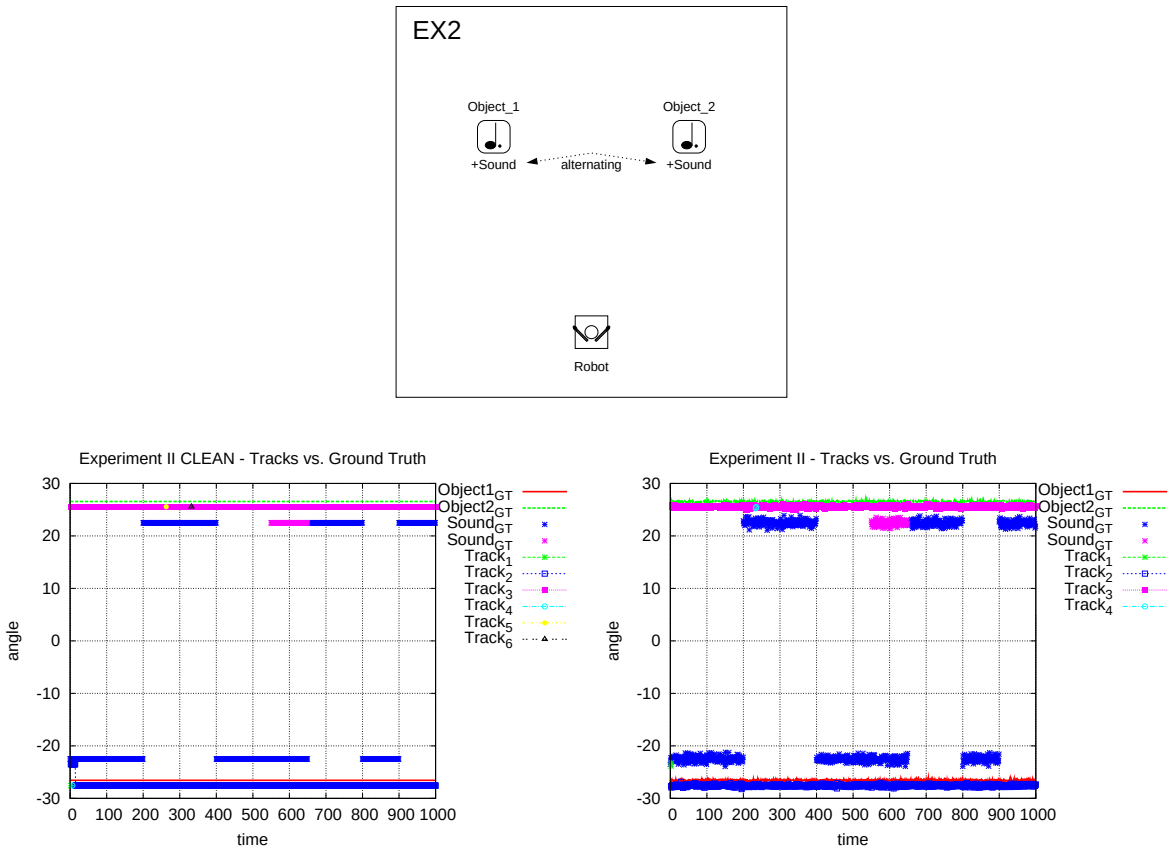


Figure 5.11: Simulation Experiment 2. **Top:** Setup. **Bottom left:** Results on clean dataset. **Bottom right:** Results on noisy dataset.

data).

Again, $\mathsf{Object1}_{\mathsf{GT}}$, $\mathsf{Object2}_{\mathsf{GT}}$ and $\mathsf{Sound}_{\mathsf{GT}}$ denote ground truth about the two objects and the sound source. Note that the sound source azimuth is offset by a few degrees from the actual position, to make the figure more readable. Note also, that a second $\mathsf{Sound}_{\mathsf{GT}}$ is introduced at about time 550 and lasting until about time 650. This is caused by the data analysis software. As an entity cannot be at two positions at the same time, a second source is introduced, overlapping the first in time. For our dialogue scenario, it would have been more correct to show one source at around $+25°$, switching on and off at regular intervals, and a second source around $-25°$, doing the same. But, from the point of view of the tracking algorithm, it makes no difference. And actually, the sound sources cannot be differentiated, which makes one way of plotting it as good as another.

$\mathsf{Track}_1, \ldots, \mathsf{Track}_6$ in Figure 5.11, bottom left denote the tracks returned by MCMCDA for the clean dataset. Overall, the outcome is quite good, with Track_2 corresponding to $\mathsf{Object1}_{\mathsf{GT}}/\mathsf{Sound}_{\mathsf{GT}}$ and Track_3 corresponding to $\mathsf{Object2}_{\mathsf{GT}}/\mathsf{Sound}_{\mathsf{GT}}$. The two objects are correctly tracked throughout the experiment, even though the sound source sometimes switches sides. Nevertheless, a few extraneous tracks are created. Track_1 and Track_4 appear very briefly at the beginning of the experiment at the position of $\mathsf{Object1}_{\mathsf{GT}}$. Track_5 and Track_6 appear very briefly at the position of $\mathsf{Object2}_{\mathsf{GT}}$, in the period where the sound source is first turned on.

$\mathsf{Track}_1, \ldots, \mathsf{Track}_4$ in Figure 5.11, bottom right show the algorithm's response to the noisy simulation. Track_2 and Track_3 prove that the algorithm is able to correctly track the two entities. But here again, two spurious tracks are present. Whereas Track_1 can be attributed to the burn-in phase of the Metropolis-Hastings algorithm (it appears only

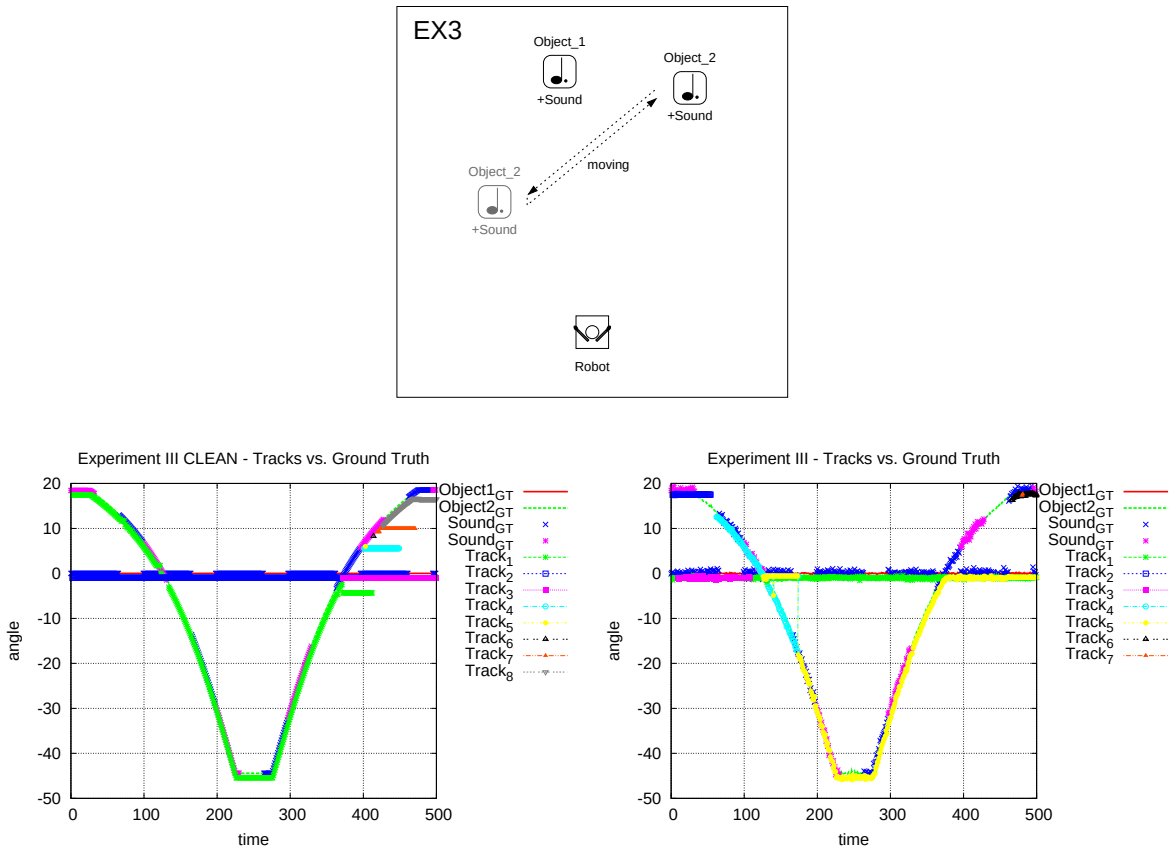


Figure 5.12: Simulation Experiment 3. **Top:** Setup. **Bottom left:** Results on clean dataset. **Bottom right:** Results on noisy dataset.

very briefly at around time 0), Track_4 has to be caused by another effect.

Notice that here, exactly as in Experiment 1, the errors only happen in the presence of both an object and a sound source.

In Experiment 3 (Figure 5.12, top and Table C.4) a more challenging setup was tried. It was meant to test the tracker's behavior with moving entities. Here, one static object is right in front of the robot, while a second object moves from the robot's right to its left and back. To complicate matters further, sound sources are associated with both objects, regularly turning on and off.

This is by far the most difficult situation for the algorithm to handle, as there are two tracks crossing each other.

The outcome is shown in Figure 5.12, bottom left (clean) and bottom right(noisy). $\mathsf{Object1}_{\mathsf{GT}}$, $\mathsf{Object2}_{\mathsf{GT}}$ and $\mathsf{Sound}_{\mathsf{GT}}$ show ground truth about objects and sound sources.

The sheer number of tracks produced — eight in the clean case, seven in the noisy case (where the correct number would be two) — is an indication for the difficulties the tracking algorithm had with this dataset.

The best performance is shown on the clean dataset (Figure 5.12, bottom left). The first crossing of the two tracks (Track_1 and Track_2) is handled gracefully and the two tracks are continued correctly afterwards. It is at the second crossing where things go bad. Both tracks just stop at the crossing and Track_2 suddenly continues as Track_3. A few short lived tracks are created ($\mathsf{Track}_4, \ldots, \mathsf{Track}_7$), before the moving entity is correctly reacquired again, but not as Track_1, but as a different track, namely Track_8.

On the noisy dataset (Figure 5.12, bottom right), the situation gets worse. At the beginning, two tracks (Track_1 and Track_3) are created for the static entity (one for the

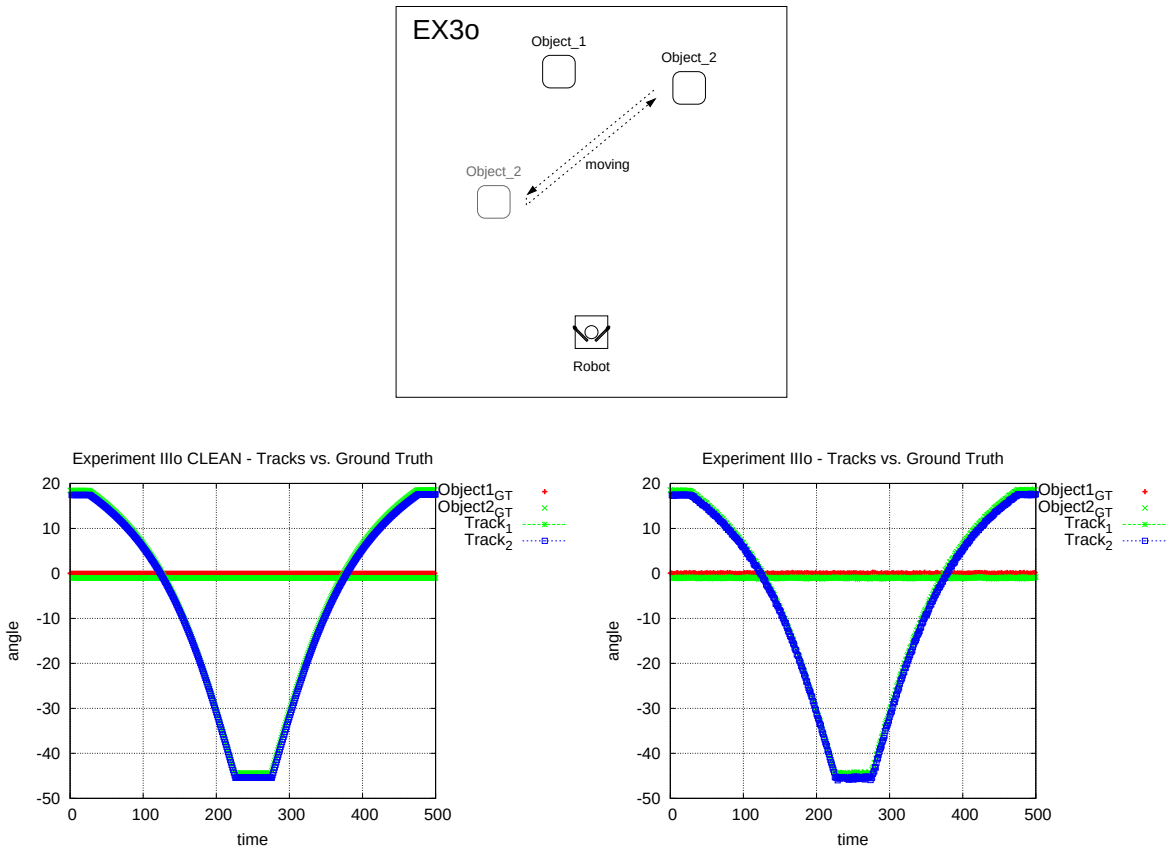


Figure 5.13: Simulation Experiment 3**o** (objects only). **Top:** Setup. **Bottom left:** Results on clean dataset. **Bottom right:** Results on noisy dataset.

object and one for the sound source). The moving entity starts out as Track_2, before being detected as Track_4, under which label it passes the crossing event. In the meantime, Track_3 disappeared and a unique track remains for the static entity (Track_1). At and directly after the crossing, a second track (Track_5) briefly gets assigned to the non-moving entity, before the moving entity is labeled Track_5. This situation (Track_1 for static, Track_5 for moving entity) remains until the second crossing. Now, both tracks get assigned to the static entity, whereas the moving entity is not tracked anymore. It is only reacquired at the very end as Track_6.

In order to evaluate the influence of the combinations of sound sources and objects, two further variants of Experiment 3 were designed.

Experiment 3**o** (Figure 5.13, top and Table C.5) is the same as the initial experiment 3, only with the sound sources removed. As can be seen, MCMCDA had no difficulty at all tracking the objects. Both in the clean case (Figure 5.13, bottom left) as well as in the noisy case (Figure 5.13, bottom right) no errors whatsoever were produced during tracking and both objects were assigned the same tracks throughout the experiment. Namely Track_1 for Object_1 and Track_2 for Object_2.

The second variant of Experiment 3, Experiment 3**s** (Figure 5.14, top and Table C.4) is the same as the original, only with the objects removed and both sound sources switched on for the whole duration of the experiment. In this case, tracking proved to be much more difficult. The problem always were the crossing events. At these points, the algorithm could not really decide on which sound belonged to which track, which is why it alternated between the two for a time after the crossings. But finally, MCMCDA always settled on one hypothesis.

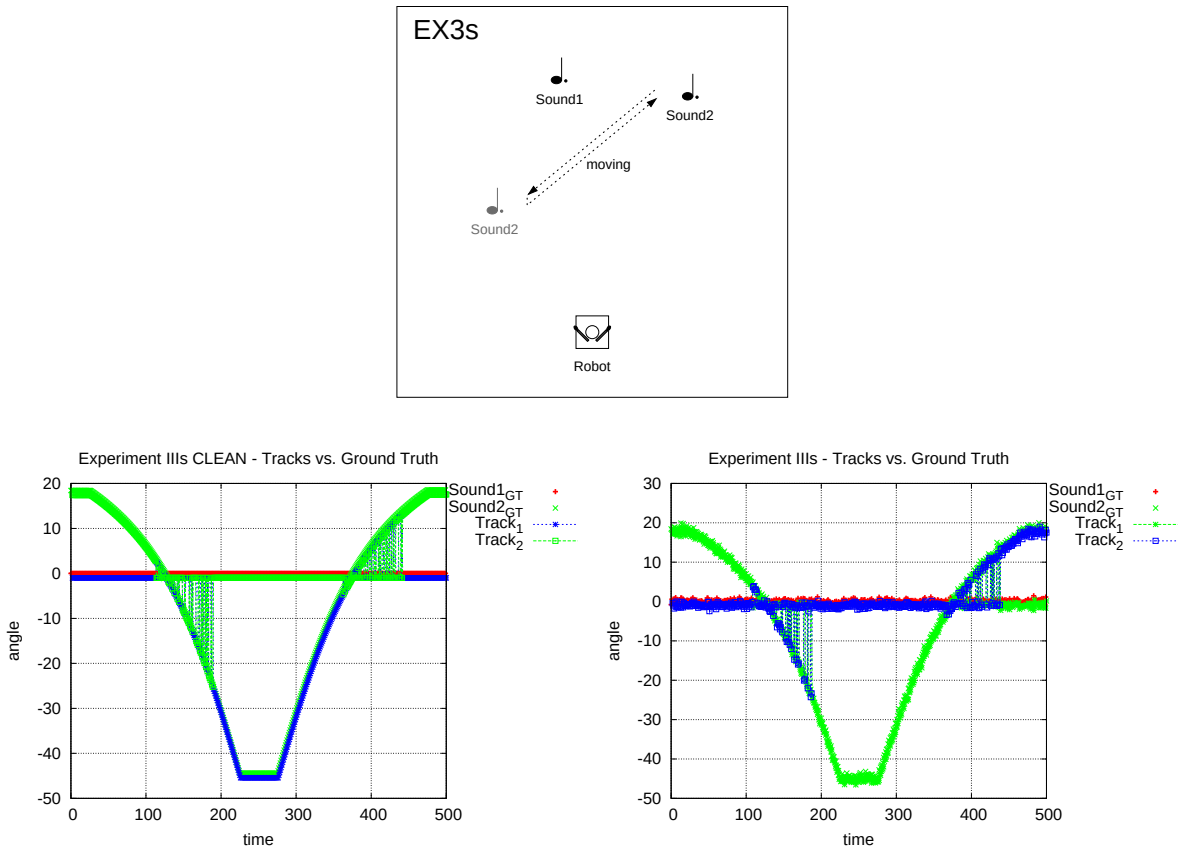


Figure 5.14: Simulation Experiment 3s (sounds only). **Top:** Setup. Note that, although the sound sources are depicted at given (x,y) coordinates, in the simulation they are only represented by the azimuth relative to the robot. **Bottom left:** Results on clean dataset. **Bottom right:** Results on noisy dataset.

For the clean case (Figure 5.14, bottom left) one can say that, apart from the periods of confusion due to the crossing events, the algorithm settled for the following sound source-to-track assignments. Track_1 followed $\mathsf{Sound1}_{\mathsf{GT}}$ until the first crossing, after which it followed $\mathsf{Sound2}_{\mathsf{GT}}$. After the second crossing, Track_1 followed $\mathsf{Sound1}_{\mathsf{GT}}$ again. Track_2 was initially assigned $\mathsf{Sound2}_{\mathsf{GT}}$, but after the first crossing it followed $\mathsf{Sound1}_{\mathsf{GT}}$. After the second crossing, Track_2 was back to $\mathsf{Sound2}_{\mathsf{GT}}$ again.

In the noisy case (Figure 5.14, bottom right), the situation was exactly reversed. Track_1 initially followed $\mathsf{Sound1}_{\mathsf{GT}}$. After the first crossing, it was assigned to $\mathsf{Sound2}_{\mathsf{GT}}$ and finally back to $\mathsf{Sound1}_{\mathsf{GT}}$ after the second crossing. Track_2 followed $\mathsf{Sound2}_{\mathsf{GT}}$, then $\mathsf{Sound1}_{\mathsf{GT}}$ and $\mathsf{Sound2}_{\mathsf{GT}}$ again, with the switch to a different sound source happening after each crossing event.

It should be noted that the exact opposite assignments in the clean and noisy cases are not caused by the noise in the simulation data, but rather by the nondeterministic nature of MCMCDA. It could as well have happened that both cases resulted in the same sound source to track mappings.

Experiment 4 (Figure 5.15, top and Table C.7), was meant to assess how the system would react to a sound source and an object — initially co-located — separating and reassociating after a while.

Overall, the system response is very good (Figure 5.15, bottom left & right). Initially (and correctly) only one track is present (Track_1). When the sound source and the object start to diverge, another track — Track_2 for the clean data (Figure 5.15, bottom left)

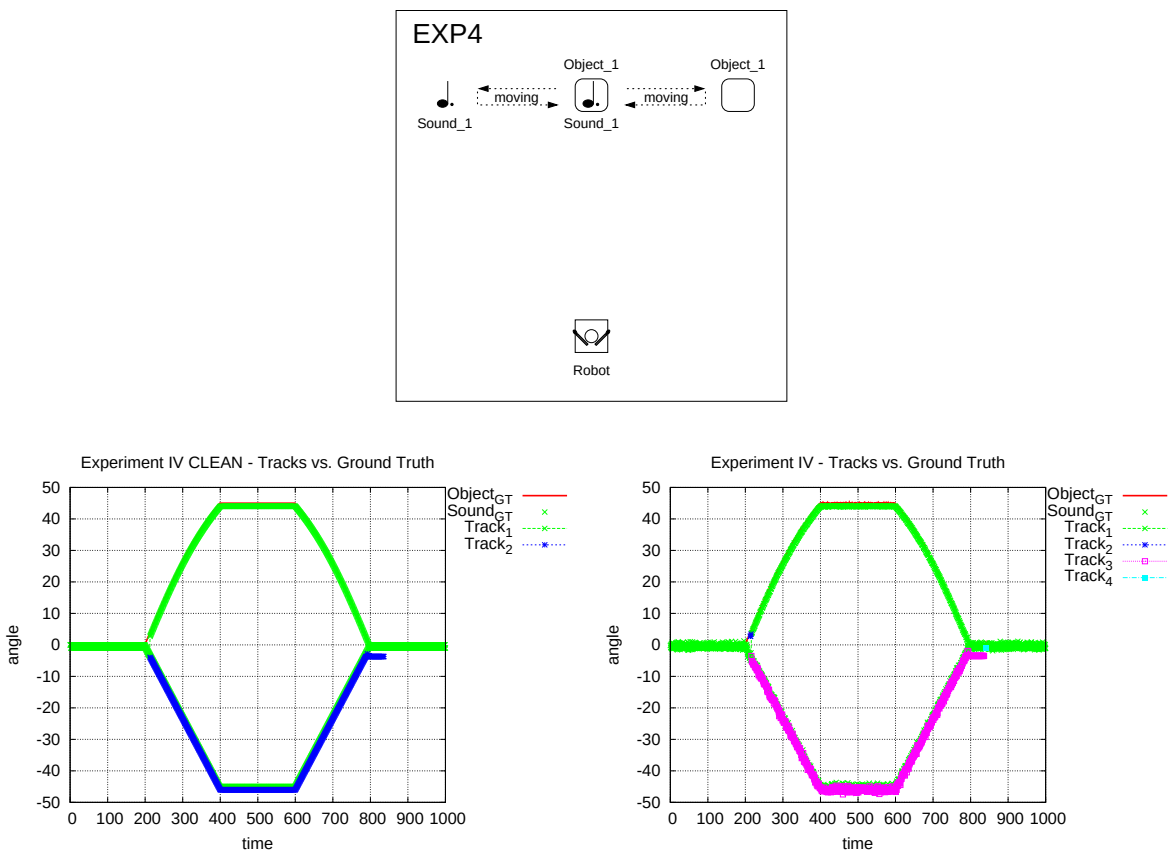


Figure 5.15: Simulation Experiment 4. **Top:** Setup. **Bottom left:** Results on clean dataset. **Bottom right:** Results on noisy dataset.

and Track_3 for the noisy data (Figure 5.15, bottom right) — is created, which is tracking the sound source. Meanwhile, Track_1 remains with the object. When the sound source and the object reunite, Track_3 disappears and Track_1 alone remains.

This is the response to be expected from this dataset. As long as the sound source and the object are co-located, the probability of the sensor data emanating from one and the same entity is much higher as it being two separate entities. Hence the single track. As soon as the sound source and the object start to diverge, however, the assumption has to be that, after all, they are two separate entities and an additional track is created.

Once the sound source and the object have converged again, the one-entity hypothesis prevails once more as soon as the algorithm has 'forgotten' that the object and the sound source were once dissociated. This happens about fifty timesteps after the junction, because the algorithm only keeps the sensor data of the last fifty timesteps. Older data are simply discarded as new data come in.

Although the overall performance of the algorithm in this experiment is quite good, one can see that here also — on the noisy dataset — short spurious tracks were created at critical moments (Track_2 and Track_4 at the dis- and conjunctions).

Note also, that, exactly as in the previous experiments, these events again involved a track consisting of an association of a sound source with an object.

The final simulation experiment — experiment 5 (Figure 5.16, top and Table C.8) — was meant to study the behavior of MCMCDA with only one sensor modality available (sound source azimuth, in this case).

Two sound sources move from opposite sides towards each other, cross in front of the robot and move to the other side. There, the path reverses and after crossing once

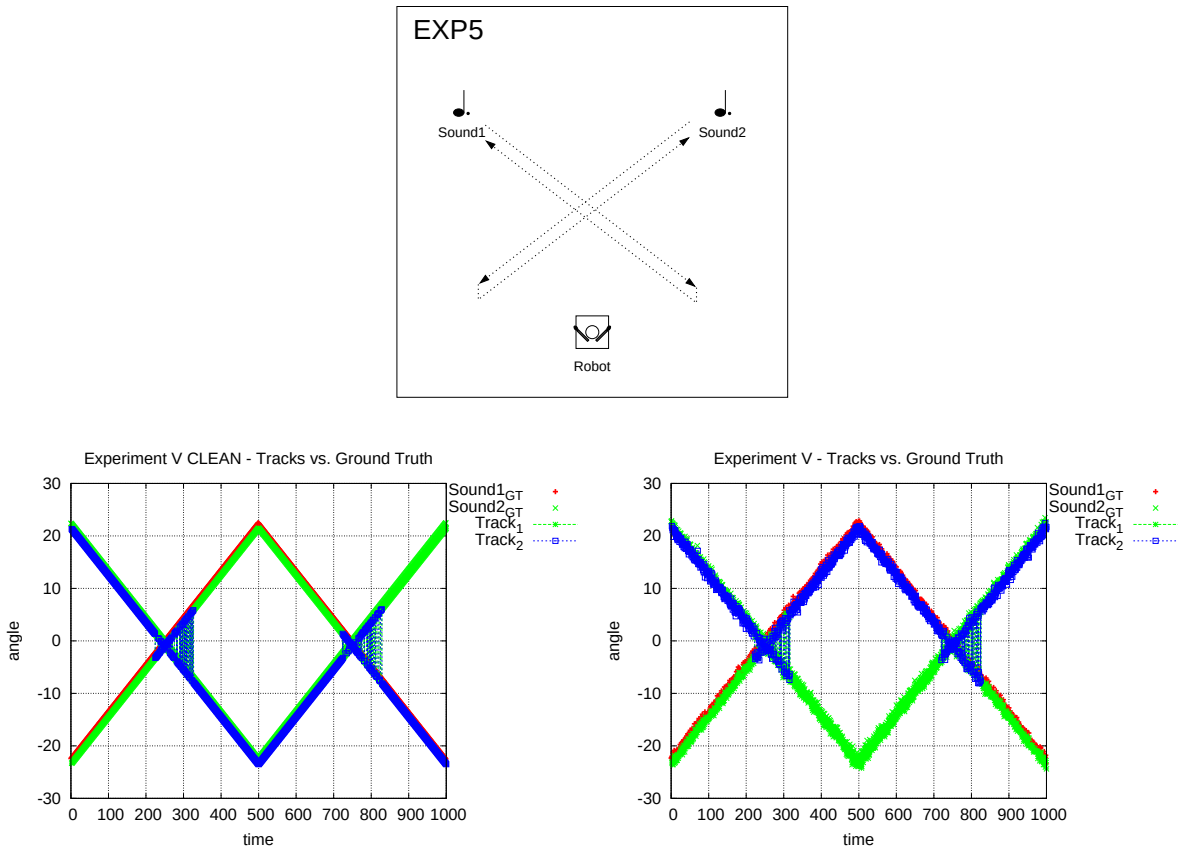


Figure 5.16: Simulation Experiment 5. **Top:** Setup. **Bottom left:** Results on clean dataset. **Bottom right:** Results on noisy dataset.

again in the middle, they go back to their initial positions.

For the clean dataset (Figure 5.16, bottom left), the first crossing is tracked correctly. For the second crossing, however, the algorithm decided that the sources would not continue their linear course, but reverse directions, i.e. meet at 0° and then go back.

This behavior can be observed for both crossings on the noisy dataset (Figure 5.16, bottom right).

Now, this in itself does not constitute an error in tracking. As the only information available to MCMCDA at any given timestep were azimuth values, to the algorithm it was more or less equally probable that both sound sources would continue their course or reverse direction. In order for MCMCDA to correctly detect the linear continuation of the sound sources movement after the crossings, some additional, distinguishing, feature of each source would have had to be available (like, e.g. spectral content).

All in all, the outcome of Experiment 5 is quite excellent, although the tracker was confused for a time after the crossings for both the clean and noisy datasets.

Note that, unlike the previous experiments, no spurious tracks appear. Also unlike previous experiments, only one sensor modality is tracked.

Here, as in Experiment 3s (Figure 5.14, bottom), the differences in assignment between the clean and noisy datasets stem from the non-deterministic nature of MCMCDA, rather than from the added noise in the simulation data.

5.2.5 Discussion

The results of the simulation experiments show that — in principle — MCMCDA with a virtual sensor as sensor-fusion frontend is functional.

Nevertheless, the results are far from satisfying. Compared to the sensor data from the real robot, the simulated objects and sound sources should have been really easy to track. There are a few major differences to real sensor data, that greatly simplify the task:

- The simulated positions contained no false alarms, i.e. position data delivered to the algorithm always corresponded to a (simulated) object or sound source. In contrast, on the real robot, false alarms are possible and probable.
- Occlusion was not simulated. With a real laser scanner, an object passing behind another (like in Figure 5.12) — from the point of view of the sensor — simply disappears for the time it stays behind the object nearer to the robot. Additionally, the real robot has four blind spots, corresponding to the four poles, on which the upper platform is mounted. These were not simulated either.
- The lateral degradation of the accuracy of the sound source localizer was not simulated. The error of the sound source azimuth was either 0° (clean datasets) or corresponded to the added Gaussian noise mentioned in Section 5.2.4. In either case, the error was independent of source direction and much less than the error to be expected in reality.
- Sound source localization front/back confusions were not considered at all, as in all experiments, the sound sources were in the frontal hemisphere of the simulated robot. On the real robot, of course, there is no guarantee that this will always be so.

As already mentioned, these four points should really make tracking easier for MCMCDA. And yet, the algorithm had great difficulty in providing correct tracks for the simulation datasets, which is the reason why no experiments with the real robot were performed. As it is, the algorithm is not ready for use on a real robot.

The only real difficulty that was included in the simulations was the added random noise. But, apart from Experiment 3 (Figure 5.12), the noise had no significant influence on the outcome.

So, what is the real problem? As already hinted to in the description of the experiments, MCMCDA made errors each time a track was to be expected which consisted of objects as well as sound sources.

Sound sources alone are tracked reliably, as well as objects alone. This was one of the design goals of the tracking system and as such seems to work as expected.

The issue with tracks containing a combination of sound sources and objects can have several causes.

The one cause that makes the problem so difficult in the first place is that, in the case of a sound source, the only datum available is the direction with respect to the robot. A sound source can thus — in theory — be anywhere on a straight line extending from the robot to infinity in the measured direction. In the simulation experiments, the dynamic object nearest to this imaginary line was always the one to which the sound source was to be associated. In practice, this might not be the case. Of two or more objects detected by the laser sensor in the vicinity of a sound source detected by the sound source localizer, it might very well be the case that the one actually emitting the sound is not the one nearest to the sound source. It might also be possible that none of these objects emits a sound — think of a radio on a table behind a dynamic object. While this sound source should actually be ignored by the robot, the tracking system will still associate it with a dynamic object, producing a false **SO**-type observation.

Another cause which adds to the complexity of the problem (and is related to the lack of distance information) is the modeling of sound sources behavior. As the tracking algorithm has to have an idea of how sound sources move — in order to eliminate unlikely hypotheses — it has to be provided with a model of the movement of sound sources.

In the case of dynamic objects, this statistical model basically consists of the classic Newtonian equations of motion, with acceleration and velocity variance parameters chosen to be consistent with human walking speed.

Unfortunately for sound sources, this model cannot be specified with constant variances, as the angular speed of the sound source is dependent on the distance to the robot. For example, a sound source moving with a constant translational speed of $v = 3$ m/s along an arc at a constant distance of $r = 1$ m from the robot, will have an angular speed of $\omega = v/r = 3$ rad/s. At a distance $r = 3$ m from the robot, the same source with the same translational speed would be detected with an angular speed of $\omega = 1$ rad/s by the tracking system.

As can be seen, simple movement models based on assumptions on the angular velocity of sound sources are doomed to fail. It is only in combination with a laser-based object, providing distance information, that assumptions about sound source movement can be made. But in order to reliably associate sound sources with objects in the first place, some knowledge of sound source distance would be highly beneficial.

Nevertheless, it could be argued that the tracking system should be able to infer the right object / sound source association after a few iterations, by the movement pattern of objects and sound sources.

In order for that to happen, **SO**-type observations have to be created for *every* possible — given the constraints (ATC, Section 5.1) — sound source / object combination. But this would entail a slew of new problems.

Right now, it is ensured that each object and each sound source only appear once in a **SO**-type observation (by assigning the sound source with minimal angular distance to an object). Furthermore, the probability of assigning the **SO**-generating **S** or **O**-type observation to another track is minimized by the *CombProb*-property of an observation (Section 5.2.2). Spawning all possible combinations (like mentioned above) would bring the risk of including a single sound source or object in multiple tracks, which should be avoided as we assume that every detected sound source or object can only appear once in a single track.

The obvious solution to the problem would be to acquire distance information about the sound source. Unfortunately, this is nearly impossible with only two microphones. Even biological systems with two ears (like humans) are not good at determining distance to sound sources, based on acoustical information alone. Distance computation relies heavily on contextual information. As an example, having knowledge about the nature of the sound source (i.e. what frequency spectrum to expect), the frequency specific attenuation (low frequencies travel farther in air than high frequencies) can provide a distance cue.

On a robot, the simplest solution to acoustically determine the distance to a sound source is to use active audition, i.e. include the movement of the robot in the sound source localization process. Through triangulation (either explicit or implicit depending on the method used), the distance to a particular source can be determined.

Another possibility is to increase the number of microphones. In combination with a suitable array geometry, 3D localization becomes possible, i.e. computing all three spatial coordinates of a source: azimuth, elevation and distance. Furthermore, this would solve the problem of front/back ambiguities (there would be none). But this requires special hardware, having as many precisely synchronized audio inputs as there

are microphones. Additionally, computing times increase, as there are more signals which have to be processed. Also, physical constraints on the robot are much higher for an array consisting of 4, 6, 8 or more microphones, than for just two microphones.

Another solution, easily feasible with MCMCDA, would be to stop treating sound sources and objects equivalently. The tracking algorithm would be used to exclusively track dynamic objects. The tracks could then be annotated with sound source information, i.e. if and when a sound source was available in the direction of the specific track. This information could also be weighted with the angular distance of the sound source from the track. This would provide a measure of how long (if at all) a sound source was available in the direction of a given object and how reliable this information was. In that way, the sound source could be used to modulate the attention of the robot and direct it to a dynamic object emitting sounds.

The perfect solution, of course, would be to make the original idea work. Meaning a tracking system that can track combined sound source / dynamic object entities and which in the limit cases of only sound sources / only dynamic objects would degenerate to a sound source-only / dynamic object-only tracker. This would also have the advantage — as a more generic solution — that adding additional sensor modalities could (probably) be implemented more easily.

Unfortunately, in hindsight, it seems that the virtual sensor approach was too naïve. The basic idea of this approach was to delegate the sensor fusion part of the system to a preprocessing frontend (the virtual sensor), leaving the core MCMCDA algorithm largely unaware of the individual sensor modalities. Adding sensor modalities, then, would mainly consist in extending the virtual sensor. We still think that this idea is feasible, as long as the individual sensors provide similar data, i.e. which can be transformed into one another. In our case, the sensors would have to deliver coordinates that can be represented as a point on the navigational occupancy map. Which is the case for the laser-based object recognition, but not for the sound source localizer.

In order for MCMCDA to work with sound source azimuths and dynamic object coordinates, the concept of the virtual sensor would have to be abandoned in favor of a solution in which sensor fusion is fully integrated into MCMCDA's statistical models. Furthermore, triangulation of sound sources through robot movement would have to be modeled in order to obtain distance information.

Chapter 6

Conclusion

6.1 Summary

In this thesis we proposed, firstly, biologically inspired methods of binaural sound source localization for mobile robots. Secondly, we proposed a method for modulating the robot's attention inspired from the barn owl and thirdly a tracking system which makes it possible for a robot to track objects emitting sounds.

Regarding sound source localization, the method that was best understood and evaluated is the algorithm based on the evaluation of interaural time differences (Section 4.1). There is a very simple reason for this state of affairs.

Interaural time differences are influenced mainly by the inter-microphone distance, provided that there is no major obstruction (like an artificial head) between them. This would make the sound waves bend around the structure and thus increase the path length and consequentially ITD in a frequency-specific manner. As long as there is no obstruction between the microphones and the far-field assumption is satisfied, the interaural time difference directly relates to azimuth through a simple equation (Equation 2.14), where only the additional parameters of inter-microphone distance (constant) and speed of sound (can be regarded constant) are required (cf. Section 2.3.2 and Appendix A). Under these conditions, it is straightforward to adapt ITD localization to different hardware platforms: it only requires mounting the microphones and providing the correct microphone baseline value to the software.

The method we use for ITD based sound localization relies on detecting phase coincidence for individual frequencies in the frequency domain and subsequent frequency integration to eliminate phase ambiguities. Overall, the results of the system are excellent. Broadband signals could be localized with an accuracy of about $\pm 2^\circ$. The localization of pure tones was highly erratic, as was to be expected. The only unexpected behavior was the low accuracy in localizing 100 Hz–1 kHz bandpass noise. By performing simulations in which the room acoustics could be controlled, we could show that this is caused by sound reflections from the environment (Section 4.1.2). In larger rooms or, equivalently, rooms with a lower direct-to-reverberant ratio, localization precision of broadband signals also degrades significantly, which becomes evident in our experiments on a real robot (Section 5.1.3). All in all, care has to be taken as to the acoustic environment in which the ITD based sound source localization is to be deployed, in order to achieve best performance.

Interaural level differences based sound source localization by principle relies on the acoustical properties of the microphone mount assembly and supporting structures. This means that adapting ILD localization to a new platform is more difficult. It requires

mounting the microphones and then calibrating the whole setup to record the resulting azimuth / elevation / frequency dependent ILD values, which can then be used by the ILD based sound source localization algorithm. This is a quite elaborate, time-consuming procedure which has to be repeated every time something changes in the way the microphones are mounted — or, indeed, if the microphones themselves are changed. The experiments with the artificial owl ruffs (Section 4.3) illustrate this point: even small changes in the ruff can have a huge impact on the ILDs (and, to a lesser degree, on the ITDs).

The method for ILD based sound source localization relies on a neuronal model of the barn owl's auditory intensity pathway. Specifically, the neuronal responses in the VLVp and the ICc ls as well as the connections between these areas are modeled. The results of the experiments with the algorithm are very encouraging. The first tests (cf. Section 4.2.3) showed that the system was able to accurately localize broadband sound sources in the range of $-30°\ldots+30°$. The more elaborate artificial ruffs experiments (cf. Section 4.3.1) confirmed these results. Furthermore, with the correct acoustic design of the artificial ruff, it is possible to use the ILDs for various purposes as for example localization in elevation and/or verification/correction of the ITD based azimuth estimates.

With the attentional module based on a neuronal saliency map it is possible to preactivate a robot's attention to a specific region of interest. With this method it was possible to successfully reproduce with a robotic pan-tilt unit attentional latency experiments that were performed with barn owls (Johnen et al., 2001). But the system we propose can easily be generalized to modulate (in several instances) the attention of the robot at various levels, from the basic sensor level (as we did, cf. Section 4.4) up to the planning level.

The Markov chain Monte Carlo based combined sound source and dynamic object tracking had a few problems accurately tracking our simulated entities. Although the general viability of the method could be shown, the algorithm still has several shortcomings. MCMCDA with a virtual sensor is able to correctly track sound sources and objects alone, but the combination of both modalities in one track proved to be difficult. As long as the individual entities are in clearly distinct positions, correct tracks are produced, but if they approach each other or — even worse — cross paths, tracking breaks down (Section 5.2.4). This seems to be caused mainly by the lack of distance information in the sound source localization modality. As long as these shortcomings are not addressed, it makes little sense to test the method on a real robot. This is why the MCMCDA experiments in this thesis were limited to simulations.

6.2 Future Work

The methods presented in this thesis can only be considered as a basis, a foundation for binaural hearing with tracking on mobile robots. Although the results are quite good, some work still remains to be done.

First of all, ITD localization has to be rendered more robust against acoustic reflections if it is to be really useful in practice. This could be done by incorporating for example a model of the *precedence effect*, or *law of the first wavefront* (Litovsky et al., 1999). In biological systems, this effect inhibits further binaural processing of sound waves for a specific time after the first wavefront arrived. In this way, reflections are excluded from sound source localization. What might also help would be to fully combine ITD and ILD based localization. In this way, shortcomings in one method could be compensated by the other method. Furthermore, as it is unlikely that a structure

for generating ILDs does not modify ITDs, the currently used method of precomputing delay values (Eq. 4.2) should be abandoned (or at least be rendered optional). In its stead, a table of ITD values measured from the actual microphone assembly should be used.

More research has also to be performed on ILD localization, especially on the ILD generating structure — the artificial ruff. This has to be carefully designed in order to achieve the desired binaural cues. One could design it in such a way that ITDs and ILDs vary only with azimuth, which could be used to achieve the compensating effect mentioned above. Alternatively, if elevation information is required, the design could be so that ITD varies only with azimuth and ILD only with elevation. Besides the acoustic design, the mechanical properties of the whole assembly should not be neglected. Ideally, it should be a compact, self contained structure incorporating the microphones and artificial ruff. This would make it possible to easily adapt it to various hardware platforms, e.g. a mobile robot or a pan-tilt unit in an anechoic chamber. To this effect, it should be a sturdy mechanical design, so that the binaural cues are not modified through handling.

The attentional module should be integrated into the robot software. For example, visual face recognition could be used as the attention modulation signal. In that way, the robot could already "expect" sound coming from that direction and react accordingly faster, when this happens. Alternatively, the attentional module with face recognition could be combined with tracking. The attentional module would then preselect (associate with a higher weight) tracks on the side of a recognized face.

This would of course require a correctly working tracking module. The simple sound source / dynamic object association currently performed by the MCMCDA frontend is clearly inadequate. It has to be investigated if the idea in itself of delegating sensor fusion to a virtual sensor frontend is at fault. Perhaps using more elaborate techniques and providing the tracking algorithm with more information can solve the issue. If this is not the case, the sensor fusion would have to be performed within MCMCDA, i.e. statistical models would have to be developed for sound sources, dynamic objects and their interactions. This would greatly reduce the generality and extensibility of the approach. It all boils down to the question if the lack of distance information to sound sources simply constitutes insufficient information for combined tracking. In this case, one would have to either try to obtain this distance information — the easiest way to do this would be to either move the robot or use microphone arrays with more than two microphones arranged in an adequate geometry. Another possibility would be to abandon combined tracking altogether and use sound source azimuth only as meta-information to annotate individual tracks of laser-based dynamic objects.

Appendix A

The Far-field Assumption

In this Appendix we derive the equation relating ITD to azimuth under the far-field assumption (Equation 2.14). Furthermore, we will investigate how this can impact localization accuracy.

If one assumes omnidirectional microphones (as were used in this thesis), with no obstructing structures between them, then the propagation of the acoustic waves is governed by simple geometric laws and does not require any knowledge of acoustics. Specifically, propagation is frequency-independent, i.e. there are no frequency-specific attenuation, refraction, reflection or diffraction effects[1].

The time delay is the difference in path length needed for a signal to travel from a sound source to the left and right microphones. If d_1 and d_2 are the distances from the source to the left and right microphones respectively (in m), the time delay Δt (in s) is given by the following equation:

$$\Delta t = \frac{d_1 - d_2}{c} = \frac{\Delta d}{c}, \tag{A.1}$$

where c is the speed of sound (in m/s) and Δd is the path length difference (in m).

A.1 The Far-field Assumption

Under the far-field assumption, the acoustic wavefront reaching the microphones is planar and not spherical, as it would be in the near-field case. This means that the waves travel in parallel (see Figure A.1 **a**).

The derivation of Equation 2.14 then only requires basic trigonometry, as the microphone baseline b, the path length difference Δd and the wavefront form a right triangle (Figure A.1 **a**). It follows that

$$\Delta d = \sin(\frac{\pi}{2} - \gamma)b = b \sin \alpha \tag{A.2}$$

Substituting Equation A.2 into Equation A.1 gives:

$$\Delta t = \frac{b \sin \alpha}{c} = \frac{b}{c} \sin \alpha, \tag{A.3}$$

which is Equation 2.14

[1] In this, we base our considerations exclusively on the microphones / sound source system, disregarding room acoustics. As we have seen in Sections 4.1.2 & 5.1.3, room acoustics can play a significant role in real-life tests.

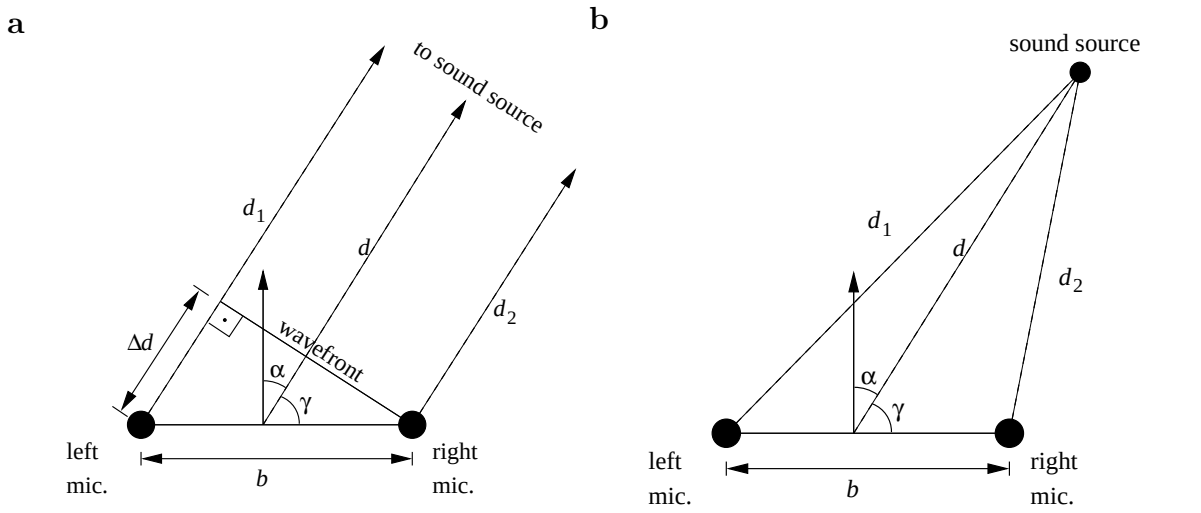


Figure A.1: Far- and near-field situations. **a** Far-field situation. As we assume a plane wave, the incident acoustic waves are parallel, i.e. the angle of incidence is the same for the two microphones and is equal to γ. **b** Near-field situation. Note the differing angles of incidence of the sound waves at the microphones.

A.2 Error to the Near-field case

In order to compute the error between the actual angle and the far-field approximation, Equation A.3 has to be rearranged in order to obtain α:

$$\alpha = \arcsin \frac{\Delta t c}{b} = \arcsin \frac{\Delta d}{b} = \arcsin \frac{d_1 - d_2}{b} \tag{A.4}$$

Using the law of cosines, d_1 and d_2 for the near-field case (Figure A.1 **b**) can be computed with the help of the following equations:

$$d_1 = \sqrt{d^2 + \frac{b^2}{4} + db\cos\gamma} \tag{A.5}$$

$$d_2 = \sqrt{d^2 + \frac{b^2}{4} - db\cos\gamma} \tag{A.6}$$

From Equations A.5 & A.6 and with $\alpha = \frac{\pi}{2} - \gamma$ it follows that

$$\begin{aligned} \frac{d_1 - d_2}{b} &= \frac{\sqrt{d^2 + \frac{b^2}{4} + db\cos\gamma} - \sqrt{d^2 + \frac{b^2}{4} - db\cos\gamma}}{b} \\ &= \sqrt{\left(\frac{d}{b}\right)^2 + \frac{1}{4} + \frac{d}{b}\sin\alpha} - \sqrt{\left(\frac{d}{b}\right)^2 + \frac{1}{4} - \frac{d}{b}\sin\alpha} \end{aligned}$$

i.e., the ratio $(d_1 - d_2)/b$ depends only on the angle α and the ratio d/b between the distance to the sound source d and the microphone baseline b.

Thus, the error between the far-field approximation and the actual angle also only depends on α and d/b:

$$\mathrm{E}(\alpha, \frac{d}{b}) = \left|\alpha - \arcsin \frac{d_1 - d_2}{b}\right| \tag{A.7}$$

Figure A.2 shows a plot of Equation A.7 for values of α over the whole azimuth range and for values of d/b going up to ten. The maximal error values are found at

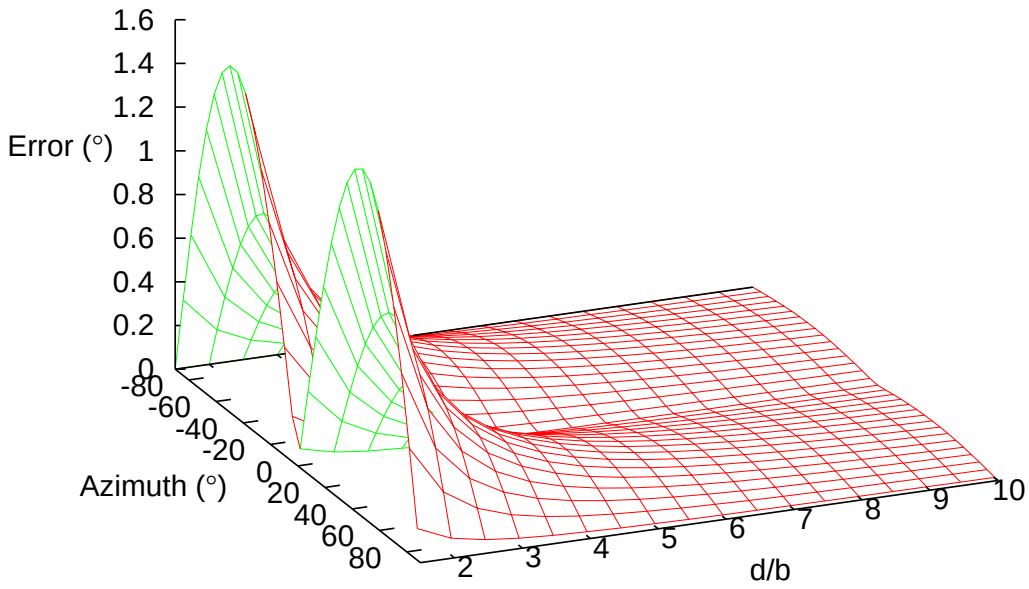


Figure A.2: Far-field approximation error. Shown is the error over the whole azimuth range of $-90°\ldots+90°$ and d/b ratios up to 10.

$\pm 45°$, regardless of the value of d/b. Note that with $d/b > 2.7$, the error of the far-field approximation drops below 0.5°, which is the maximal theoretical resolution with which we tested our ITD based localization system.

Taking $d/b = 2.7$ as a basis, we can compute where far-field started for the experiments we conducted. During the evaluation of the sound source localization methods (see Chapter 4), the microphone distance was 20.5 cm. Thus far-field started at a distance of 55.35 cm in this case. On the real robot (Section 5.1), the microphone distance was 13 cm, resulting in a far-field situation starting at 35.1 cm.

Appendix B

Roomsim Setup

Table B.1 shows the parameters used for the MATLAB package Roomsim. This program was used during the room simulation experiments which characterized the performance of the dual delay-line algorithm for different room sizes and acoustic properties (cf. Section 4.1.2). Only the parameters which were changed from the default Roomsim values are shown in Table B.1. Source parameters are specified in relative to the receiver position.

Table B.2 shows the surface absorption coefficients used for the room simulation experiments in Section 4.1.2. For simplicity, all six room surfaces (floor, ceiling, four walls) received the same absorption coefficients. The values are those provided by the Roomsim package.

Table B.3 shows the RT_{60} reverberation times, estimated according to the Eyring

Table B.1: Roomsim parameters.

Sampling frequency	16 kHz
Room depth (Lx)	4.95 m
Room width (Ly)	3.48 m
Room height (Lz)	4 m
Receiver x position	0.8 m
Receiver y position	1.5 m
Receiver z position	1.0 m
Receiver type	two sensors
Receiver sensor separation	0.205 m
Receiver sensor directivity	omnidirectional
Receiver azimuth offset	$-70°\ldots+70°$ ($10°$ steps)
Source radial distance	1 m or 3.5 m
Source azimuth	$0°$
Source elevation	$0°$

Table B.2: Surface absorption coefficients.

	Standard measurement frequencies					
	125 Hz	250 Hz	500 Hz	1 kHz	2 kHz	4 kHz
anechoic	1	1	1	1	1	1
50%	0.75	0.75	0.75	0.75	0.75	0.75
unpainted concrete	0.4	0.4	0.3	0.3	0.4	0.3

Table B.3: Estimated reverberation times (RT_{60}) for the different simulation setups.

	Standard measurement frequencies					
	125 Hz	250 Hz	500 Hz	1 kHz	2 kHz	4 kHz
anechoic	0 s	0 s	0 s	0 s	0 s	0 s
50%	0.08 s	0.08 s	0.08 s	0.08 s	0.08 s	0.08 s
unp. concrete	0.216 s	0.216 s	0.309 s	0.309 s	0.216 s	0.309 s

formula (Eyring, 1933):

$$\mathrm{RT}_{60} = \frac{0.16V}{-S\ln(1-\alpha)}, \tag{B.1}$$

where S is the total surface area (in m^2), V is the room volume (in m^3) and α is the absorption coefficient (taken from Table B.2).

RT_{60} specifies the time it takes for the sound pressure level in a room to drop to 1000th the level of the sound source, after the source was switched off. This corresponds to a decrease in sound pressure level of 60

More intuitively, RT_{60} specifies the time it takes for reverberation to decay by 60 dB below the level of the direct sound. As such, RT_{60} is a measure of the reverberation qualities of an acoustic environment. It is especially important in architecture, as rooms with a low reverberation time sound "dead" (the first time in an anechoic chamber can be a creepy experience). Spaces with a high RT_{60} sound "live" (think of a church). On the other hand, speech intelligibility decreases with reverberation time, especially for hearing impaired listeners.

Conference rooms (speech only) should have a low reverberation time in the range of 0.5 s–1 s for optimal speech intelligibility. Concert halls, on the other hand can have RT_{60} values of 1.5 s–2.5 s, depending on hall size and type of music (Everest, 2000).

Appendix C

MCMCDA Setup

C.1 General Algorithm setup

Table C.1 shows the parameter setup of the MCMCDA algorithm during the simulation experiments. The individual parameters are described in detail in Section 5.2.

C.2 Simulation Experiments Parameters

Tables C.2–C.8 show the detailed setups of the simulation experiments. The origin of the robot and object coordinates is the map origin. The origin of the sound source coordinates (i.e. azimuth) is the robot position and orientation.

The "timesteps" column indicates at what individual timesteps an entity was at what position. If there are missing timesteps for a given entity, that entity was not present at those timesteps.

All movement was linear, i.e. a "moving" entry in the "coordinates" column indicates that the entity's position changed in a linear fashion starting from the previous entry and ending at the next entry for that entity.

Table C.1: Parameters used for the MCMCDA algorithm during the simulation experiments.

Name	Description	Value
A	observation area (7 m × 6 m)	42 m^2
n_{smpls}	# of Monte Carlo smpls (per timestep)	200
T_s	sampling interval (s/timestep)	0.1 s
T	observation time	5 s (50 timesteps)
d_{max}	max. sensor data skip interval	3 s
$\lambda_b A$	birth rate times obs. area	10^{-6} /s
$\lambda_f A$	false alarm rate times obs. area	10^{-8} /s
P_z	track termination probability	0.1
P_d	detection probability	0.9
MaxAngSpeed	max. angular speed (of sound sources)	60°/s
MaxSpeed	max. speed (of objects)	1.4 m/s

Table C.2: Simulation experiment 1.

Entity	coordinates	timesteps	remarks
Robot	(0 m, 0 m, 0 rad)	$0-999$	
Object 1	(2.5 m, 0 m)	$0-999$	
Object 2	(2 m, 1 m)	$0-999$	
Sound 1	$0°$	$0-999$	corresponds to Object 1
Sound 2	$-22.5°$	$0-999$	

Table C.3: Simulation experiment 2.

Entity	coordinates	timesteps	remarks
Robot	(0 m, 0 m, 0 rad)	$0-999$	
Object 1	(2 m, -1 m)	$0-999$	
Object 2	(2 m, 1 m)	$0-999$	
Sound 1	$-22.5°$	$0-199 \vert 400-649 \vert 800-899$	corresponds to Object 1
Sound 2	$22.5°$	$200-399 \vert 550-799 \vert 900-999$	corresponds to Object 2

Table C.4: Simulation experiment 3.

Entity	coordinates	timesteps	remarks
Robot	(0 m, 0 m, 0 rad)	$0-499$	
Object 1	(3 m, 0 m)	$0-999$	
Object 2	(3 m, 1 m)	$0-25$	start point
”	moving	$26-224$	
”	(1 m, -1 m)	$225-275$	reversal point
”	moving	$276-474$	
”	(3 m, 1 m)	$475-499$	end point
Sound 1	$0°$	$0-65 \vert 99-164 \vert 198-263$	corresponds
		$297-362 \vert 396-461 \vert 495-499$	to Object 1
Sound 2	computed from	$0-32 \vert 66-131 \vert 165-230$	corresponds
	Obj. 2 coords.	$264-329 \vert 363-428 \vert 462-499$	Object 2

Table C.5: Simulation experiment 3**o**.

Entity	coordinates	timesteps	remarks
Robot	(0 m, 0 m, 0 rad)	$0-499$	
Object 1	(3 m, 0 m)	$0-999$	
Object 2	(3 m, 1 m)	$0-25$	start point
”	moving	$26-224$	
”	(1 m, -1 m)	$225-275$	reversal point
”	moving	$276-474$	
”	(3 m, 1 m	$475-499$	end point

Table C.6: Simulation experiment 3**s**.

Entity	coordinates	timesteps	remarks
Robot	(0 m, 0 m, 0 rad)	$0-499$	
Sound 1	$0°$	$0-499$	
Sound 2	$18.5°$	$0-26$	start point
”	moving	$27-224$	
”	$-44.5°$	$225-275$	reversal point
”	moving	$276-474$	
”	$18.5°$	$475-499$	end point

Table C.7: Simulation experiment 4.

Entity	coordinates	timesteps	remarks
Robot	(0 m, 0 m, 0 rad)	$0-999$	
Object	(2 m, 0 m)	$0-199$	start point
”	moving	$200-398$	
”	(2 m, 2 m)	$399-599$	reversal point
”	moving	$600-798$	
”	(2 m, 0 m)	$799-999$	end point
Sound	$0°$	$0-199$	start point
”	moving	$200-398$	
”	$-45°$	$399-599$	reversal point
”	moving	$600-798$	
”	$0°$	$799-999$	end point

Table C.8: Simulation experiment 5.

Entity	coordinates	timesteps	remarks
Robot	(0 m, 0 m, 0 rad)	$0-999$	
Sound 1	$-22.5°$	0	start point
”	moving	$1-498$	
”	$22.5°$	499	reversal point
”	moving	$500-998$	
”	$22.5°$	999	end point
Sound 2	$22.5°$	0	start point
”	moving	$1-498$	
”	$-22.5°$	499	reversal point
”	moving	$500-998$	
”	$22.5°$	999	end point

Index

Bibliography

Andersson, S. B., Handzel, A. A., Shah, V., and Krishnaprasad, P. S. (2004). Robot phonotaxis with dynamic sound-source localization. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, volume 5, pages 4833–4838.

Andrieu, C., de Freitas, N., Doucet, A., and Jordan, M. I. (2003). An introduction to MCMC for machine learning. *Machine Learning*, 50(1–2):5–43.

Bar-Shalom, Y. and Fortmann, T. E. (1987). *Tracking and data association*, volume 179 of *Mathematics in Science and Engineering*. Academic Press Professional, Inc., San Diego, CA, USA.

Berglund, E. and Sitte, J. (2006). The parameterless self-organizing map algorithm. *IEEE Transactions on Neural Networks*, 17(2):305–316.

Berglund, E., Sitte, J., and Wyeth, G. (2008). Active audition using the parameter-less self-organising map. *Autonomous Robots*, 24(4):401–417.

Blackman, S. S. (2004). Multiple hypothesis tracking for multiple target tracking. *IEEE Aerospace and Electronic Systems Magazine*, 19(1, Part 2):5–18.

Blauert, J. (1997). *Spatial Hearing: The Psychophysics of Human Sound Localization*. MIT Press.

Brandstein, M. S. and Silverman, H. F. (1997). A practical methodology for speech source localization with microphone arrays. *Speech Communication*, 11:91–126.

Buhmann, J., Burgard, W., Cremers, A., Fox, D., Hofmann, T., Schneider, F., Strikos, J., and Thrun, S. (1995). The mobile robot RHINO. *AI Magazine*, 16(2):31–38.

Calmes, L., Lakemeyer, G., and Wagner, H. (2007a). Azimuthal sound localization using coincidence of timing across frequency on a robotic platform. *Journal of the Acoustical Society of America*, 121(4):2034–2048.

Calmes, L., Wagner, H., Schiffer, S., and Lakemeyer, G. (2007b). Combining sound localization and laser based object recognition. In Tapus, A., Michalowski, M., and Sabanovic, S., editors, *Papers from the AAAI Spring Symposium*, pages 1–6, Stanford CA. AAAI Press.

Carr, C. E. and Konishi, M. (1988). Axonal delay lines for time measurement in the owls brain stem. *Proceedings of the National Academy of Sciences of the United States of America*, 85:8311–8315.

Carr, C. E. and Konishi, M. (1990). A circuit for detection of interaural time differences in the brainstem of the barn owl. *Journal of Neuroscience*, 10:3227–3246.

Chib, S. and Greenberg, E. (1995). Understanding the Metropolis-Hastings algorithm. *The American Statistician*, 49(4):327–335.

Cooley, J. W. and Tukey, J. W. (1965). An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19(90):297–301.

Coradeschi, S. and Saffiotti, A. (2001). Perceptual anchoring of symbols for action. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-2001)*.

Cox, I. J. (1993). A review of statistical data association techniques for motion correspondence. *International Journal of Computer Vision*, 10(1):53–66.

Dellaert, F., Fox, D., Burgard, W., and Thrun, S. (1999). Monte Carlo localization for mobile robots. In *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*.

Doostdar, M., Schiffer, S., and Lakemeyer, G. (2008). A robust speech recognition system for service-robotics applications. In *Proceedings of the RoboCup Symposium 2008*.

Doucet, A., de Freitas, N., and Gordon, N., editors (2001). *Sequential Monte Carlo methods in Practice*. Springer.

Duhamel, P. and Vetterli, M. (1990). Fast Fourier transforms: A tutorial review and a state of the art. *Signal Processing*, 4(19):259–299.

Durrant-Whyte, H. and Bailey, T. (2006a). Simultaneous localisation and mapping (SLAM): Part I. *Robotics & Automation Magazine*, 13(2):99–108.

Durrant-Whyte, H. and Bailey, T. (2006b). Simultaneous localisation and mapping (SLAM): Part II. *Robotics & Automation Magazine*, 13(3):108–117.

Everest, F. A. (2000). *The Master Handbook of Acoustics*. McGraw-Hill.

Eyring, C. F. (1933). Methods of calculating the average coefficient of sound absorption. *Journal of the Acoustical Society of America*, 4(3):178–192.

Fox, D., Burgard, W., and Thrun, S. (1999). Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11:391–427.

Fox, D., Burgard, W., Thrun, S., and Cremers, A. B. (1998). Position estimation for mobile robots in dynamic environments. In *AAAI '98/IAAI '98: Proceedings of the 15th National/10th Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, pages 983–988, Menlo Park, CA, USA. American Association for Artificial Intelligence.

Fox, D., Hightower, J., Liao, L., Schulz, D., and Borriello, G. (2003). Bayesian filters for location estimation. *IEEE Pervasive Computing*, 2(3):24–33.

Frank, A. (2004). On Kuhn's Hungarian Method – a tribute from Hungary. Technical report, Egerváry Research Group on Combinatorial Optimization, Pázmány P. sétány 1/C, H-1117, Budapest, Hungary.

Fritsch, J., Kleinehagenbrock, M., Lang, S., Fink, G. A., and Sagerer, G. (2004). Audiovisual person tracking with a mobile robot. In *Proceedings of the International Conference on Intelligent Autonomous Systems*, pages 898–906.

Fritsch, J., Kleinehagenbrock, M., Lang, S., Plötz, T., Fink, G. A., and Sagerer, G. (2003). Multi-modal anchoring for human–robot interaction. *Robotics and Autonomous Systems*, 43(2–3):133–147.

Handzel, A. A. and Krishnaprasad, P. S. (2002). Biomimetic sound-source localization. *IEEE Sensors Journal*, 2(6):607–616.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109.

Heidemann, M. T., Johnson, D. H., and Burrus, C. S. (1984). Gauss and the history of the fast Fourier transform. *IEEE ASSP Magazine*, 1(4):14–21.

Huang, J., Supaongprapa, T., Terakura, I., Wang, F., Ohnishi, N., and Sugie, N. (1999). A model-based sound localization system and its application to robot navigation. *Robotics and Autonomous Systems*, 27:199–209.

Itti, L. and Koch, C. (2000). A saliency-based search mechanism for overt and covert shifts of visual attention. *Vision Research*, 40(10–12):1489–1506.

Jeffress, L. A. (1948). A place theory of sound localization. *Journal of Comparative & Physiological Psychology.*, 41(1):35–39.

Johnen, A., Wagner, H., and Gaese, B. H. (2001). Spatial attention modulates sound localization in barn owls. *Journal of Neurophysiology*, 85(2):1009–1012.

Julier, S. J., Uhlmann, J. K., and Durrant-Whyte, H. F. (1995). A new approach for filtering nonlinear systems. In *Proceedings of the American Control Conference*, pages 1628–1632.

Kalman, R. E. (1960). A new approach to linear filtering and predicition problems. *Journal of Basic Engineering*, 82(1):35–45.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Knapp, C. H. and Carter, G. C. (1976). The generalized correlation method for estimation of time delay. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 24(4):320–327.

Knudsen, E. I. and Konishi, M. (1978a). A neural map of auditory space in the owl. *Science*, 200(4343):795–797.

Knudsen, E. I. and Konishi, M. (1978b). Space and frequency are represented separately in auditory midbrain of the owl. *Journal of Neurophysiology*, 41(4):870–884.

Knudsen, E. I. and Konishi, M. (1979). Mechanisms of sound localization in the barn owl (tyto alba). *Journal of Comparative Physiology A*, 133(1):13–21.

Konishi, M. (2000). Study of sound localization by owls and its relevance to humans. *Comparative Biochemistry and Physiology Part A*, 126:459–469.

Köppl, C. (1997a). Frequency tuning and spontaneous activity in the auditory nerve and cochlear nucleus magnocellularis of the barn owl *Tyto alba*. *Journal of Neurophysiology*, 77(1):364–377.

Köppl, C. (1997b). Phase locking to high frequencies in the auditory nerve and cochlear nucleus magnocellularis of the barn owl, *Tyto alba*. *Journal of Neuroscience*, 17(9):3312–3321.

Köppl, C. and Yates, G. (1999). Coding of sound pressure level in the barn owl's auditory nerve. *Journal of Neuroscience*, 19(21):9674–9686.

Krämer, T. (2008). Attempts to build an artificial ruff mimicking the barn owl (Tyto alba). Diploma thesis, RWTH-Aachen University.

Kuhn, H. (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97.

Li, D. and Levinson, S. E. (2002). A linear phase unwrapping method for binaural sound source localiation on a robot. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*.

Liang, Z., Ma, X., and Dai, X. (2008). Robust tracking of moving sound source using multiple model Kalman filter. *Applied Acoustics*, 69(12):1350–1355.

Litovsky, R. Y., Colburn, H. S., Yost, W. A., and Guzman, S. J. (1999). The precedence effect. *Journal of the Acoustical Society of America*, 106(4):1633–1654.

Liu, C., Wheeler, B. C., O'Brien, Jr., W. D., Bilger, R. C., Lansing, C. R., and Feng, A. S. (2000). Localization of multiple sound sources with two microphones. *Journal of the Acoustical Society of America*, 108(4):1888–1905.

Lv, X. and Zhang, M. (2008). Sound source localization based on robot hearing and vision. In *Proceedings of the International Conference on Computer Science and Information Technology 2008*.

Lüke, H. D. (1999). *Signalübertragung*. 7. Auflage, Springer Verlag.

Manley, G. A., Köppl, C., and Konishi, M. (1988). A neural map of interaural intensity differences in the brain stem of the barn owl. *Journal of Neuroscience*, 8(8):2665–2676.

Martin, W. H. (1929). Decibel – the name for the transmission unit. *Bell System Technical Journal*, VIII(1):1–2.

Martinson, E. and Schultz, A. (2006). Auditory evidence grids. In *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Mazer, J. A. (1998). How the owl resolves auditory coding ambiguity. *Proceedings of the National Academy of Sciences of the United States of America*, 95(18):10932–10937.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092.

Mogdans, J. and Knudsen, E. I. (1994). Representation of interaural level difference in the VLVp, the first site of binaural comparison in the barn owl's auditory system. *Hearing Research*, 74(1–2):148–164.

Moiseff, A. and Konishi, M. (1983). Binaural characteristics of units in the owl's brainstem auditory pathway: precursors of restricted spatial receptive fields. *Journal of Neuroscience*, 3(12):2553–2562.

Moravec, H. and Elfes, A. (1985). High resolution maps from wide angular sensors. In *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, pages 116–121.

Murray, J. C., Erwin, H. R., and Wermter, S. (2009). Robotic sound-source localisation architecture using cross-correlation and recurrent neural networks. *Neural Networks (2009 Special Issue)*, 22(2):173–189.

Nakadai, K., Lourens, T., Okuno, H. G., and Kitano, H. (2000). Active audition for humanoid. In *Proceedings of 17th National Conference on Artificial Intelligence*, pages 832–839.

Nakadai, K., Okuno, H. G., and Kitano, H. (2002). Real-time sound source localization and separation for robot audition. In *Proceedings of the 7th International Conference on Spoken Language Processing*, pages 193–196, Denver, USA.

Nilsson, N. J. (1984). Shakey the robot. Technical Report 323, AI Center, SRI International.

Nishiura, T., Nakamura, M., Lee, A., Saruwatari, H., and Shikano, K. (2002). Talker tracking display on autonomous mobile robot with a moving microphone array. In *Proceedings of the 2002 International Conference on Auditory Display*, Kyoto, Japan.

Nishiura, T., Yamada, T., Nakamura, S., and Shikano, K. (2000). Localizaton of multiple sound sources based on a CSP analysis with a microphone array. In *Proceedings of the 2000 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages II1053–II1056.

Nix, J. and Hohmann, V. (2006). Sound source localization in real sound fields based on empirical statistics of interaural parameters. *Journal of the Acoustical Society of America*, 119(1):463–479.

Oh, S. (2008). Bayesian formulation of data association and Markov chain Monte Carlo data association. In *Proceedings of the Robotics: Science and Systems Conference (RSS) Workshop Inside Data association*, ETH Zürich, Switzerland.

Oh, S., Russell, S., and Sastry, S. (2004). Markov chain monte carlo data association for general multiple target tracking problems. In *Proceedings of the 43rd IEEE Conference on Decision and Control*.

Omologo, M. and Svaizer, P. (1994). Acoustic event localization using a crosspower-spectrum phase based technique. In *Proceedings of the 1994 IEEE International Conference on Acoustics, Speech, and Signal Processing*.

Payne, R. S. (1971). Acoustic location of prey by barn owls (*Tyto Alba*). *Journal of Experimental Biology*, 54:535–573.

Peger, D. (2005). Biologically inspired sound localization using interaural level differences. Diploma thesis, RWTH-Aachen University.

Reid, D. B. (1979). An algorithm for tracking multiple targets. *IEEE Transactions on Automation and Control*, 24(6):843–854.

Roman, N. and Wang, D. (2003). Binaural tracking of multiple moving sources. In *Proceedings of the 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pages 149–152.

Röttsches, D. (2004). A computational model for multimodal spatial attention. Diploma thesis, RWTH-Aachen University.

Saberi, K., Farahbod, H., and Konishi, M. (1998). How do owls localize interaurally phase-ambiguous signals? *Proceedings of the National Academy of Sciences of the United States of America*, 95(11):6465–6468.

Schnupp, J. W. H. and Carr, C. E. (2009). On hearing with more than one ear: lessons from evolution. *Nature Neuroscience*, 12(6):692–697.

Schulz, D., Burgard, W., Fox, D., and Cremers, A. B. (2001). Tracking multiple moving targets with a mobile robot using particle filters and statistical data association. In *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*.

Shannon, C. E. (1949). Communication in the presence of noise. *Proceedings of the Institute of Radio Engineers*, 37(1):10–21.

Smith, J. O. (2007). *Mathematics of the Discrete Fourier Transform (DFT) with Audio Applications, Second Edition*. W3K Publishing, http://www.w3k.org/books/.

Spence, C. and Pearson, J. C. (1989). The computation of sound source elevation in the barn owl. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 2, [NIPS Conference, Denver, Colorado, USA, November 27-30, 1989]*, pages 10–17. Morgan Kaufmann.

Strack, A., Ferrein, A., and Lakemeyer, G. (2005). Laser-based localization with sparse landmarks. In *Proceedings of RoboCup 2005 Symposium*.

Sullivan, W. E. and Konishi, M. (1984). Segregation of stimulus phase and intensity coding in the cochlear nucleus of the barn owl. *Journal of Neuroscience*, 4(7):1787–1799.

Takahashi, T. T., Barberini, C. L., and Keller, C. H. (1995). An anatomical substrate for the inhibitory gradient in the VLVp of the owl. *Journal of Comparative Neurology*, 358(2):294–304.

Takahashi, T. T. and Keller, C. H. (1992). Comissural connections mediate inhibition for the computation of interaural level difference in the barn owl. *Journal of Comparative Physiology A*, 172(2):161–169.

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L.-E., Koelen, C., Markey, C., Rummel, C., van Niekerk, J., Jensen, E., Alessandrini, P., Bradski, G., Davies, B., Ettinger, S., Kaehler, A., Nefian, A., and Mahoney, P. (2006). Stanley: The robot that won the DARPA Grand Challenge. *Journal of Field Robotics*, 23(9):661–692.

Valin, J.-M., Michaud, F., and Rouat, J. (2007). Robust localization and tracking of simultaneous moving sound sources using beamforming and particle filtering. *Robotics and Autonomous Systems*, 55(3):216–228.

Valin, J.-M., Michaud, F., Rouat, J., and Létourneau, D. (2003). Robust sound source localization using a microphone array on a mobile robot. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*.

van Veen, B. D. and Buckley, K. M. (1988). Beamforming: A versatile approach to spatial filtering. *IEEE ASSP Magazine*, 5(2):4–24.

Vermaak, J., Godsill, S. J., and Pérez, P. (2005). Monte Carlo filtering for multi-target tracking and data association. *IEEE Transactions on Aerospace and Electronic Systems*, 41(1):309–332.

Viste, H. and Evangelista, G. (2004). Binaural source localization. In *Proceedings of the 7th International Conference on Digital Audio Effects (DAFx'04)*, pages 145–150, Naples, Italy.

von Campenhausen, M. and Wagner, H. (2006). Influence of the facial ruff on the sound-receiving characteristics of the barn owl's ears. *Journal of Comparative Physiology A*, 192(10):1073–1082.

Wagner, H., Mazer, J. A., and von Campenhausen, Mark (2002). Response properties of neurons in the core of the central nucleus of the inferior colliculus of the barn owl. *European Journal of Neuroscience*, 15(8):1343–1352.

Curriculum vitae

Schulische und wissenschaftiche Ausbildung

26.07.1975	geboren in Luxemburg-Stadt
1981-1987	Grundschule Schüttringen
1987-1994	Lycée de Garçons, Luxembourg (Gymnasium)
1994	Diplôme de fin d'études secondaires (Abitur)
1995-2003	Studium der Informatik an der RWTH Aachen
April 2003	Diplom in Informatik
2003-2005	Wissenschaftlicher Angestellter am Institut für Biologie II (Zoologie/Tierphysiologie) der RWTH Aachen
2005-2008	Wissenschaftlicher Angestellter am Lehr- und Forschungsgebiet Informatik V (Wissensbasierte Systeme) der RWTH Aachen
2003-2009	Promotion am Institut für Biologie II und am Lehr- und Forschungsgebiet Informatik V

Konferenzen und Kurse

2003	Jahrestagung der Deutschen Zoologischen Gesellschaft (DZG), Berlin
2004	Interdisciplinary College IK2004, Möhnesee
2005	Adaptive Motion in Animals and Machines (AMAM 2005), Ilmenau
2007	AAAI Spring Symposium on Multidisciplinary Collaboration for Socially Assistive Robotics, Stanford University