

ULTRASONIC MICROPHONE ARRAYS

for

BAT BIOACOUSTICS

using

APPLIED OPEN SOURCE SOLUTIONS

by

Tórrur Andreassen

<thor@biology.sdu.dk>

Supervisor: Annemarie Surlykke *<ams@biology.sdu.dk>*
Co-supervisor: John Hallam *<john@mmmi.sdu.dk>*

Institute of Biology
University of Southern Denmark
May 31, 2013

Abstract

Today, state-of-the-art bioacoustic research requires high-sample-rate, multi-channel, and often long-term recording systems. Commercial systems are very costly and sometimes do not satisfy the experimental parameters. This project proposes and demonstrates an ultrasonic recording system design that is arbitrarily scalable. The system is modular and based on retail components and open source software/hardware. Each module has four microphones and modules can be combined to extend the coverage area, obtain higher spatial recording resolution and/or add recording redundancy. The system is designed to have no inherent scalability limits. We have named it MCLURS.

We have deployed MCLURS in six different test settings:

- A setup that tests the system's ability to make medium-term recordings (1-2 minutes) with many microphones.
- MCLURS did a 4-channel long-term ecological study of the local bat population at campus of University of Southern Denmark, which established the presence of two bat species that were otherwise not known to occupy the area.
- The system was integrated in a mobile robot as an echolocating guidance sensor.
- One setup demonstrates full-spectrum transducer calibration.
- MCLURS was applied to find echo profiles of various objects.
- The system has also been used to emit calibrated ultrasonic signals.

We have developed semi-automatic methods that can extract and, to some extent, classify recorded bat calls in huge datasets.

In most respects this system's hardware specification surpasses all competitors on the market at a quarter of the price. Tests demonstrate that large deployments are feasible and accurate ultrasonic measurements can be obtained. The system has proven its worth as a long-term bat survey platform.

Resumé

Optageudstyr til bioakustik i dag kræver høj samplerate, mange kanaler og ofte langtidsoptagelser. Kommercielle systemer er ofte meget dyre og opfylder ikke altid eksperimentets parametre. Dette projekt foreslår og demonstrerer et design til et optagesystem til ultralyd som er vilkårligt skalerbart. Systemet er modulært og baseret på let tilgængelige komponenter og open source software/hardware. Hvert modul har fire mikrofoner, moduler kan kombineres for at udvide dækningsområde, optage med højere rumlig opløsning, eller øge optageredundans. Systemet er designet til ikke at have nogen indbyggede skaleringsgrænser. Vi har navngivet systemet MCLURS.

Systemet har været anvendt i seks forskellige testopsætninger:

- En opstilling som tester systemets evne til at lave mellemlange optagelser (1-2 minutter) med mange mikrofoner.
- MCLURS har lavet en 4 kanaler langtids økologisk undersøgelse af den lokale flagermus bestand ved Syddansk Universitet, den viste at der forekom to flagermusarter som ikke er set i området før.
- Systemet blev integreret i en mobilrobot som en ekkoloktions-sensor.
- En opstilling demonstrerede fullspekter transducer kalibrering.
- MCLURS er blevet brugt til at opdage objekters ekkoprofil.
- Systemet er også blevet anvendt til at udsende kalibrerede ultralydssignaler.

Vi har udviklet semi-automatiske metoder som kan udvælge, og til en vis grad, klassificere flagermuslyde i meget store datasæt.

I de fleste tilfælde så overgår dette systems' hardware specifikationer alle dets konkurrenter på markedet til en kvart af prisen. Forsøg har demonstreret at store opsætninger er mulige og at korrekte ultralydsmålinger kan opnås. Systemet har bevist sit værd som langtids flagermusovervågnings platform.

to Jóhanna and our wonderful kids

I, Tórrur Andreassen, confirm that the work presented in this thesis is my own and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. I have had help from my supervisors with the wording of some sections. Information derived from other sources has been indicated in the thesis.

Tórrur Andreassen
Odense, May 31, 2013

Contents

Contents	i
Preface	v
Project progress	v
Acronyms	vii
Acknowledgments	ix
1 Introduction	1
1.1 Problems solved by this system	2
1.2 Problems solvable by this system	4
1.3 Achievements	5
1.4 Publications	6
2 Background and design	7
2.1 Criteria for an ultrasonic recording system	7
2.2 Analysis	10
2.3 Related work	12
2.4 The issues with existing solutions	15
2.5 System design	16
2.6 Summary of design choices	18
3 Prototype implementation	21
3.1 First prototype	21
3.2 Second prototype	25
3.3 Summary	26
4 Experiments and evaluation	29
4.1 Manually triggered	29
4.2 Robust long-term	31
4.3 Echolocating robot	36

4.4	Calibration setup	40
4.5	Discovering the echo profile of an object	45
4.6	Calibrated emission	46
4.7	Logging other data	53
4.8	Summary	54
5	Long-term technological	57
5.1	Experimental method	57
5.2	Recording the raw data stream	59
5.3	Extraction and classification	61
5.4	Summary	71
6	Ecological study	73
6.1	Introduction	73
6.2	Study site	74
6.3	General notes and observations	74
6.4	Event marking	76
6.5	Event classification	79
6.6	Long-term survey	84
6.7	Summary	90
7	Discussion	93
7.1	The MCLURS recording system	93
7.2	Flight tracking	97
7.3	Localized weather sensing	98
7.4	Open source	98
7.5	Advanced use of the system	105
7.6	Comments about available ultrasonic speakers	106
8	Concluding remarks	109
8.1	Possible avenues to pursue	109
8.2	Review of claimed achievements	111
A	Long-term study – source code	113

A.1	DAQ-2010 AD/DA conversion	113
A.2	Signal detection	119
A.3	Calculate quality measures	123
B	Reliability tests	129
B.1	Variable gain setting	129
B.2	Knowles microphone calibration	129
C	Recipes for interesting challenges	141
C.1	Dealing with streamed data	141
C.2	Parallel calculations	148
D	Calibrated emission source code	149
D.1	Align and average	149
D.2	Apply transfer function	152
E	Robot control	153
E.1	Control programs	153
E.2	Future improvements	161
	Bibliography	163
	List of Figures	174
	List of Tables	176
	List of Source Code Listings	177
	List of Algorithms	178
	Article – “Ultrasonic recording limits without intrinsic limits”	179
	Article – “Semi-automatic long-term ecological surveying: A case study with bats”	193

Preface

Project progress

At the close of this project it is time to take stock of what was achieved and what remains. There are often mentions of “still ongoing”, which is another way of saying that we did not have time to complete these sub-projects. This means that there are several interesting strains of thinking that are orphaned. Hopefully these will be carried on by me or other projects addressing similar issues. Where allowed, the data and software written are published so as to facilitate sub-project progression if there is interest.

Citing on-line material

During this project many on-line sources have been used. Because credit should be placed where credit is due, those sites/persons which provided the information have been cited to the extent deemed reasonable. Citing on-line material comes with its own set of issues as discussed in Fisher (URL provided, accessed: July 24 2008); Page (URL provided, accessed: July 24 2008). These discussions took place around 1996. Later in 1997 the ISO 690 standard¹ was revised to include electronic media. Given that it would cost 90 Swiss francs² to acquire this standard, I have chosen to mix and match with the MISC and ONLINE entry types in BibTeX/BibLaTeX (Patashnik, Oren et al. URL provided, accessed: July 24 2008; Various, URL provided, accessed: Sep 2012[a]) and the instructions given by the UK TeX enthusiasts (URL provided, accessed: July 24 2008).

As the on-line medium is inherently changeable, it becomes important to note the time when the material was retrieved. The material can then at a later time be accessed through archive functionality on the source in

¹Titled: “Documentation – Bibliographic references – Content, form and structure”

²http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=25921.

question, or, if that does not exist, through the version archived by the Internet Archive project³, which has been found to be slow but reliable.

Using Wikipedia as reference

There has been much controversy about using the Wikipedia encyclopedia⁴ as a reference source, e.g. Denning et al., 2005. Given prior experience with the lexicon on subjects I am knowledgeable about, the arguments given in the previous section and the fact that Wikipedia keeps a history of all changes to a given page, I believe that it can sometimes stand as the only reference. This assumes that the content is checked and the date of access is noted. Of course when dealing with more critical matters other sources should be referenced and online lexicon references should only be provided as a supplement.

³<http://www.archive.org>

⁴<http://www.wikipedia.org>

Acronyms

Acronym	Description
AD	<i>Analog to digital</i>
ADC	<i>Analog to digital converter</i>
DMI	<i>Danmarks Meteorologisk Institut</i>
GB	$1000^3 = 1,000,000,000$
GiB (IEC, 2006)	$1024^3 = 1,073,741,824$
HRTF	<i>Head Related Transfer Function</i>
IO	<i>Input and Output</i>
MCLURS	<i>Multi ChanneL Ultrasonic Recording System</i>
MB	$1000^2 = 1,000,000$
MiB (IEC, 2006)	$1024^2 = 1,048,576$
NP	<i>Non Polynomial</i>
SPL	<i>Sound Pressure Level</i>
TOAD	<i>Time Of Arrival Difference</i>

Acknowledgments

I want to thank my project supervisors, Prof. Annemarie Surlykke and Prof. John Hallam, for their helpful and inspiring advice, not least for their immeasurable patience. As electronics is not my strong suit I greatly appreciate the help I received from David Brandt and Charlotte Ryberg in this regard. This project has dealt a great deal with signal processing and I have been happy to receive invaluable help from my colleague Ali Shekarchi.

Part of this project was funded by the EU 7th Framework Program (FP7), through the ChiRoPing project (ICT-2007-1 STREP project 215370). I also wish to thank the Smithsonian Tropical Research Institute for allowing me to use their facilities on Barro Colorado Island, and also the biologists who helped me with the recordings. Thank you to DMI (Danmarks Meteorologisk Institut) for providing us with various climatic measurements for one of our survey studies.

I have great appreciation for the OSS community which has brought about many good and useful things, many of which have been used in this project. Thank you to all the proofreaders for their valuable advice, and special thanks to my family for their support and patience.

Introduction

Seek simplicity, and distrust it.

— Alfred North Whitehead

This project grew out of the need to do near arbitrary resolution ultrasonic recordings when ensonifying bat-shaped objects, typically 3D prints of actual bats. These experiments were meant to explore the correlations of bat face morphology and the bats biosonar. Below is an excerpt from the original problem description.

Bats use echolocation or “bio-sonar” to orient and find prey in the dark. They emit short high frequency sound pulses and listen for the echoes reflected from surroundings to orient in space. Bats from different families emit sounds either through the mouth or through the nose. Bats emitting sound through the nose have elaborate skin folds or “nose-leaves” around the nostrils. The shape and size of the sound emitting structures (mouth or nose) is believed to strongly affect the directionality or shape of the emitted sound beam. The shape of the sound beam will subsequently strongly affect the reflected “echo-picture” of the scene. A broad beam will ensonify objects within a wide angle, whereas a narrow beam will restrict the bat to a narrow angle of view, but perhaps reach further ahead than the broader beam. Thus, the received filtering is referred to as “head-related-transfer-function” (HRTF) and is well described for human listening, but not at all for bats in spite of the obvious importance which is reflected in the very diverse and sometimes bizarre morphology of bat faces.

The goal of the project is therefore to record and use echolocation sounds from real bat species to determine the effect of morphology on directionality of the emitted beam and HRTF of returning echoes. Based on these data we want to develop algo-

rhythms to effectively simulate the sound in 3D. 3D simulations of sound fields will be used for implementing bio-mimetic robots using sound (echolocation) to orient in space and to discriminate between objects.

However, it soon became apparent that state-of-the-art commercially available ultrasonic recording systems were either too limited in their function, or too expensive to scale up to the number of channels we were considering. So the focus turned towards developing a system capable of arbitrary scale. The design of this system together with a review of other related systems are described in Chapter 2. We have named the system Multi-ChanneL Ultrasonic Recording System, MCLURS for short.

We chose a very open-ended design, which resulted in a system that would be applicable to many tasks other than the one we originally wanted to solve, and so when the first prototype of this system was completed (see Chapter 3 for implementation details), it was found, as expected, to be applicable to a range of other areas. In the interest of testing the system thoroughly and solving interesting problems, it was applied to these first, thus leaving the bat-morphology experiment yet to be completed. We do not consider this a failure of the system, but rather its success, because there has been much demand for it.

One of the funding partners of this project, ChiRoPing, had as a goal to mimic bat bio-sonar in bio-mimetic robots both with regards to navigation and identifying objects. The efforts of this project have contributed toward these goals and we are now closer to a system capable of solving these tasks.

1.1 Problems solved by this system

First use of the system was to deploy it for cage recording of bats on Barro Colorado Island in the Panama Canal. Here the system recorded 12 synchronous channels triggered with other equipment. The reason for deploying it here was that it is capable of arbitrary size pre-trigger buffering, which

is important for recording the whole scanning sequence of *Micronycteris microtis* which can last for up to 2 minutes.

Two separate developments then started. First, we applied the system to really long-term recordings, in the order of months, and second, we developed the synchronized emission reception software needed to build an echolocator. A paragraph is dedicated to each of these developments here, but see their respective sections for more details, Section 4.2 and Section 4.3 respectively.

The object of the long-term recordings is to survey areas and monitor bat activity for long periods, possibly while the environment is changed in some way; construction of new buildings, demolishing of buildings, removal of forest etc. This would also provide the opportunity to compare bat activity throughout the season as well as year to year. Storing climatic values at the same time provides insight into what effect the climate may have on the bat activity in the particular area. As the system features a standard computer, attaching weather monitoring equipment is straightforward, and has indeed been verified to work adequately. See Section 4.2 for an overview of long-term recordings done and see chapters 5 and 6 for the methods and results of one of surveys which recorded for three months on campus grounds at the University of Southern Denmark.

We wanted to build a robot that could navigate, find and identify objects placed in its environment using only ultrasound (Barshan and Kuc, 1992); given the analog IO hardware on this system, and the synchronized nature of the input and output, it is amenable to this task. New software and hardware were developed for this application: software to achieve the synchronized analog IO and hardware to attach a 3D print of a bat head with embedded microphones onto an ultrasonic speaker (Section 4.3).

After the synchronized emission/reception had been implemented, many other experiments became possible. Some of these are described in the following. For example during our trials it became necessary to calibrate our equipment, so we could relate what we measured to actual air pressure. To do full spectrum calibration with commercial equipment can be difficult or impossible unless it is actually designed for the task. As we had no such spe-

cialist equipment available, we decided to use MCLURS. The synchronized emission/reception is easily adapted to this task (Section 4.4).

To identify objects using only ultrasound (Manthey, Kroemer, and Magori, 1999). We wanted to find the echo-profiles of various objects and, as we already had developed the synchronized emission/reception, this was a matter of attaching the echolocation sensor to fixed elevation angles relative to the ensonified object and rotating the object. This was all implemented and automated in one of the MCLURS modules (Section 4.5).

A more recent experiment led us to want to include calibrated emissions capabilities. We found the speaker’s transfer function by emitting sweeps to a calibrated microphone. Calibrated emissions were achieved by pre-convolving the signal to be emitted with the inverse transfer function Section 4.6.

1.2 Problems solvable by this system

Attempts have been made at other applications, but time constraints have limited us to the ones described in the previous section.

The original problem we set out to solve when this project started would be achievable now as the system has matured. Ensonifying objects and embedding microphones in an object and its environment, would allow us to examine the specific effect of a bat’s face morphology on ultrasound. The task of actually emitting through a nostril is still a challenge as transducers capable of the same SPL as a bat have yet to be invented. The only current alternatives are scaled experiments or simulations.

Some work has been put into recording sounds from a radio microphone attached to a bat’s head and connected to MCLURS through a USB RF receiver (Hiryu et al., 2005). There is great potential in combining the wireless microphone with other recording channels but due to time constraints this has not been done yet.

Automatic flight tracking (Madsen and Wahlberg, 2007; Spiesberger, 2004) can be integrated in several different ways. In principle it should be

sufficient to have 4 microphones placed in a non-planar arrangement, from which full 3D source position can be calculated.

As described by Guarato and Hallam (2009), the call of a flying bat can be reconstructed with reasonable accuracy if the bat's position can be determined, its emission directivity pattern is known (or can be approximated) and it emits its broadband call towards the center of a suitable array of calibrated microphones. As MCLURS is an arbitrarily scalable array, it would be possible to place a sufficient number of calibrated microphones in the environment to allow call reconstruction.

1.3 Achievements

An open-ended design for an arbitrarily-scalable recording system has been devised, implemented and proven in a variety of applications:

- Two prototypes have been implemented, using commodity computing components, standard interfaces and a small amount of custom electronics.
- The system has been scaled up to 16 simultaneous synchronized microphone channels.
- Long-term deployments have been tested with good results. The longest was 6 months.
- Synchronized emission and reception has been developed and deployed towards: an echolocation sensor, equipment calibration, and echo-profile discovery.
- The huge data-volume generated by long-term studies presents difficult automation and computational challenges. We demonstrate how to design and use a Support Vector Machine model to automatically extract bat calls from 236 GiB of data with good accuracy.
- The system was deployed for a three month period in 2011 doing an ecological study of the local bat population. The survey yielded

biologically interesting results using the above technique. We demonstrate that surveys using this technique work and provide reliable, less biased data on population sizes.

- We have successfully recovered a speaker’s transfer function and used it to emit calibrated signals.
- The sheer number of experiments that the system has been applied to demonstrates the versatility and relatively easy reconfigurability of the design.
- The use of open source software and open source hardware in the development of this project has been evaluated and deemed a very powerful combination.

1.4 Publications

Journal article: (Andreassen, Surlykke, Hallam, and Brandt, In press) “Ultrasonic recording system without intrinsic limits”.

Journal article: (Andreassen, Surlykke, and Hallam, Manuscript submitted) “Semi-automatic long-term ecological surveying: A case study with bats”.

Background and design

All models are wrong, but some are useful.

— George Box

An important goal of this work is that the system should be applicable to a wide range of experiments. The design chosen is therefore open-ended and the first prototype implementation is based on a versatile analog-to-digital and digital-to-analog platform. The design criteria we set up to ensure the system is useful for the anticipated applications and goals that are listed below.

We also want the system to be applicable to long-term surveys because acoustical surveyors, especially ultrasonic surveyors, are limited today by the available equipment. This problem concerns both the area that can be surveyed and the duration over which the survey can be performed. To ameliorate these shortcomings we wanted to design a system that is adaptable to this task. It should function as an inexpensive scalable array recording system. The system should be capable of running for months while automatically recording and saving data on ultrasonic signals in its vicinity.

2.1 Criteria for an ultrasonic recording system

The criteria are divided into three categories: *critical* criteria are considered essential — that is, if one of these is not fulfilled the system would not be useful; *important* criteria are features that the system might work without, but much emphasis has been put on achieving those goals; the *nice* criteria were implemented if convenient (Table 2.1).

critical	1. Record ultrasound. 2. Easily adaptable to new experiments. 3. Record for long periods. 4. No inherent scalability limits. 5. Pre-trigger recording. 6. Multi-channel recordings.
important	7. Synchronized emission and recording. 8. Real-time access to signal data from computer programs. 9. Use generic mass-market components leading to easy upgrade and low cost. 10. Use well defined protocols and standards to facilitate interoperability.
nice	11. Platform agnostic. 12. Generate real-time event log of data.

Table 2.1: System requirements for an ultrasonic recording system. The criteria are ordered according to importance and placed into three priority groups, i.e. **critical**, **important**, and **nice**..

Critical Criteria

The system should be: capable of recording ultrasound for long periods and easily adaptable to new experiments. It should have no inherent scalability limits, which implies that all aspects of the system should be extensible i.e. more microphones, more storage, and more computers can be added with a linear increase in spatial coverage and performance.

Pre-trigger recording, i.e. recording of events happening n seconds prior to triggering requires that n seconds of recording are continuously stored in a round-robin or ring buffer. The data in the buffer is permanently stored upon a trigger event. This is essential for recording unpredictable events such as animal sounds.

Multi-channel recordings are essential for many of the applications to which we wish to apply the system, for example to achieve high recording resolution. Also, in order to recover flight paths, a minimum of four synchronized channels is needed. This follows from the time delay of arrival equations (Madsen and Wahlberg, 2007; Spiesberger, 2004) from which one recovers the 3D position of the emitting source.

Important Criteria

In order to produce an echolocation sensor it is much more convenient if the emission and recording are synchronized. Real-time access to the data during long-term recordings enables the system to record only bits that satisfy pre-programmed criteria, e.g. sections identified by a matched filter bank of relevant bat calls. The system should use generic mass-market components leading to easy upgrade and low cost. It should communicate with well-defined publicly available formats and standards to facilitate interoperability, both in direct inter-system communication and indirectly through stored data.

Nice Criteria

The design of the system should preferably be platform agnostic, so it can be implemented on any available hardware. It would be convenient if the system could generate a real-time event log of recorded data: interesting events should be time-stamped, analyzed, and recorded in a streamable log, which would facilitate real-time display and analysis of array recordings.

2.2 Analysis

Scalability implies modularity, as that is the only way we can continuously add more channels to the system. Thus the scalability issue boils down to being able to synchronize all modules, either when they start, or as a post-processing step. Of course each module will carry as many channels as it can support.

The system should be convenient to use, so automatic configuration determination (Dubgaard, 2012), e.g. finding the layout of the microphones automatically, is important especially for large scale deployments. Wireless communication would reduce logistical requirements, but would also reduce the available bandwidth compared with a wired network.

As we want this system to proliferate and be used as much as possible, the right price and availability are essential. This means using mass-produced components which do not always deliver reliable service, but this issue can be mitigated by built-in redundancy. Another price reducing feature is to reuse other peoples work in order to reduce the need for reinvention; this is the same idea that the philosopher Bernard of Chartres expressed *nani gigantum humeris insidentes* (Dwarfs standing on the shoulders of a giant) in the twelfth century (McGarry, 1955, p. 167), that is, even if you are a dwarf if you are raised aloft you may see even further than the giant. This has, to a large extent, brought us out to the open source world where openness and tweakability are central hallmarks, thus maximizing possibilities of reuse and reapplication. Such focus limits our range of choices when it comes to choosing hardware but has been deemed more important than adhering to arbitrary limits set by different vendors.

Mechanical challenges

Building a microphone array that is useful in as many places as possible implies that it should be relocatable and deployable without too much work. This entails many interesting mechanical challenges, including weight reduction, volume reduction (at least in packaged form), and fast deployment schemes.

There is also the challenge of long-term deployment which requires the microphones to survive outside for extended periods. Even with piezoelectric elements, this is a major challenge, and indeed some microphones have actually failed during long-term trials.

Acoustical challenges

Ensuring the system mechanically often becomes a tradeoff between acoustics and mechanics. Distortions often result from reflections off the bits protecting the system and there are limitations on microphone/speaker placement. We have for example chosen to use piezoelectric microphones with the system, which are small and very sturdy, but this also means lower sensitivity. The size however is a benefit when it comes to directionality: the microphones are near omnidirectional and thus improve some of the distortion issues.

Network topologies

A modular system inherently needs communication channels to enable the modules to interact with each other and their controller. There are many widely available networking technologies and many of them come pre-installed on new systems, e.g. LAN (IEEE 802.3), Wi-Fi (IEEE 802.11). There are other technologies but they are either too slow for substantial throughput, e.g. Bluetooth (IEEE 802.15.1), ZigBee (IEEE 802.15.4), or have other problems, for instance USB 2.0 and 3.0 cables have too short reach. A host of other technologies were not considered because they are not generally available in retail hardware.

There exist various meta-topologies for example mesh-networking, which uses any available network links to build larger networks. Here nodes also work as network relays, such that a transitive relationship exists. For example if A can reach B and B can reach C then A will also be able to reach C. This involves a lot of route planning and may ultimately make the network too slow for real-time acoustic data transfer. It would however be well suited for low bandwidth communication.

Distributed sensing and computing

Given the modularity of the system, there is an inherent possibility of substantial computing power in the ultrasonic recording system. This can be used as a strength for the system, and would allow us to execute distributed algorithms, of course within the networking bandwidth constraints. Given that the data is generated in a distributed manner, it makes sense to pre-process the data on the module. We could identify interesting channels which are then propagated back to a central location, or if the algorithm allows it, processed directly on the node, and the results sent back to the controller.

2.3 Related work

Recording ultrasound requires hardware which is capable of sampling data at sufficiently high rates (Luke, 1999; Shannon, 1948) but this generates large data volumes. Thus, the system should be able to handle large data volumes with specific emphasis on the task at hand (e.g. to identify and survey bats in an area).

When recording ultrasound in the field there are limited options with regards to equipment. The system described here is compared to several commercial solutions and one university-developed solution (VoxNet) with respect to the design criteria listed in Section 2.1. The existing commercial solutions can be divided into two categories: multi-channel triggered arrays, and single- or dual-channel long-term recording systems. The comparison of these systems, VoxNet, and the one proposed in this dissertation is presented in Table 2.2.

The commercial solutions perform fairly well with regards to some of our criteria, but not all. Either they are too simple (like bat detectors) or, if they can deal with multiple channels, they are inflexible and become prohibitively expensive when scaling them up. The **VoxNet** system is both versatile and arbitrarily scalable, but in its current form it does not record ultrasound without major re-implementation and partial re-design. We include **VoxNet**

here nonetheless as it has some important design choices that are also used in MCLURS, and thus is a good comparison of the systems' design.

Hardware

Within the **triggered** array category, a popular system among biologists is the Avisoft AD converter, e.g. the USG1216H (Avisoft, URL provided, accessed: Dec 2011). Similar systems from other companies, such as National Instruments, Tucker/Davis etc., are commonly in use. Such systems are used for obtaining many synchronized acoustic samples from many places simultaneously for short periods of time. The AD converter is fed from an analog processing chain comprising analog microphones, filters, pre-amplifiers, and amplifiers — such as the Avisoft analog chain used by Gillam, Ulanovsky, and McCracken (2007) and Dzal et al. (2011) or a chain based on G.R.A.S./B&K equipment as used by Sorainen and Rytkönen (2002), Kirchner and Röschar (1999), and Surlykke and Kalko (2008). The exact choice of system and optional hardware depends on the requirements and resources available.

There are now a number of different **long-term** recording systems, which are primarily used for surveys assessing bat activity at an area of interest, e.g. when planning to build large structures. Systems evaluated include the SM2BAT+ from Wildlife Acoustics (URL provided, accessed: Apr 2012), HD-P2 from TASCAM (URL provided, accessed: Dec 2011), the Batcorder from ecoObs (URL provided, accessed: Dec 2011) and detectors from Petterson Elektronik AB (URL provided, accessed: Dec 2011). These are single- or dual-channel systems which are often acoustically triggered and store their recordings on-board. They are usually rugged and can be deployed outdoors to capture data continuously for months. But it is difficult to have them record synchronously.

The academic solution referred to is the **VoxNet** embedded sensing platform (Allen et al., 2008) and its predecessor the **ENSBox** (Girod, Lukac, et al., 2006), used in bioacoustics by, for example, Ali et al. (2007) and Collier (2011). **VoxNet** and **ENSBox** are based on networked nodes where

each node records data and, if appropriate, propagates it back to a network master. Thus these two use a different design from any of the systems described above. For networking they rely on a mesh network topology, a type of networking where each node not only handles its own network communication but also serves as a network relay for other nodes (Wu and Stojmenovic, 2004). This means that as long as each node can “see” enough other nodes, it can communicate with *all* other nodes. Each node has on-board storage, a processing unit, and a wireless link to nearby nodes. The system scales very well, is flexible and cheap (easily adaptable to new problems). Long-term recording to local storage is possible, and recording can be triggered by analyzing the incoming data stream. The system includes its own stream programming language called Wavescript (Newton et al., 2008). Depending on the iteration of the system, it is based on a Slauson Single Board Computer or the smaller Mica Mote and a fair amount of custom-made electronics and protocols. **VoxNet**'s design is modular and distributed, like the MCLURS design presented here. However, it cannot record ultrasonic signals in its present form and therefore does not satisfy our critical criteria; its extensive dependence on custom components means that to adapt it for ultrasonic recording would be a major redesign.

Software

Recording acoustical data in real-time means handling continuous streams of data. Usually this is done either by storing a set amount of data, or by processing the stream and only storing desired parts of it. Most of the systems mentioned above have developed their own proprietary solutions to handle the data streams, normally excluding any outside access. That is, if you have a concrete task involving a criterion you wish to apply to the real-time data (such as online classification, saving only some bits of the stream, triggering only when a matched filter hits, etc.) it is usually impossible without involving the vendor of your product. Only the vendor's accompanying software handles real-time streams, and the user is restricted to the built-in software solution.

	Record ultrasound (1)	Flexibility (2)	Long-term (3)	Scalability (4)	Pre-trigger (5)	Multi-channel (6)	Synch. emit/receive (7)	Real-time access (8)	Generic (9)	Interoperable (10)	Platform agnostic (11)	Event log (12)
triggered	✓	(✓)	✓	✓	✓	✓						
long-term	✓		✓		✓							
VoxNet		✓	✓	✓		✓		✓	✓	✓	✓	
MCLURS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 2.2: Comparison of the different recording systems, with the criteria from Table 2.1 on page 8 as keys in the table column (numbers refer to the corresponding point in that table). The compared systems are shortened thus: **triggered** refers to commercial triggered arrays, **long-term** to mono- or dual-channel long-term commercial recording systems, **VoxNet** is the university developed system, and **MCLURS** is the proposed system. The (✓) means it is possible but infeasible.

This real-time restriction does not hold for the **Voxnet** system, where external real-time access was designed in from the beginning (Girod, Lukac, et al., 2006; Girod, Jamieson, et al., 2006; Newton et al., 2008).

2.4 A summary of the problems with existing solutions

As may be noted from the previous section, the commercial systems are useful and applicable to many experiments, but there are also many things they are not easily applied to. In particular, when it comes to scaling up the recording area or the recording resolution of interest, the **triggered** array’s attention span is too short, i.e. they cannot record with many channels for arbitrary time, unless each system runs with one channel which would make it prohibitively expensive. On the other hand **long-term** recording systems

are encumbered by not being synchronized. This makes them very difficult to apply to experiments where multiple channels are needed to work in unison.

The design of **VoxNet** carries similar benefits to MCLURS but lacks generic hardware as it depends heavily on custom designed hardware. If it were to be applied to ultrasonic recording it would need a faster AD converter and either faster networking or larger local storage or both.

2.5 System design

As guiding design principles, we have taken inspiration from Kelly Johnson (Skunk Works) and Douglas McIlroy (of Unix fame). Johnson said, perhaps as a variation on Occam’s razor (Baker, 2011, §1), *always use the simplest solution possible* (Rich, 2004) and McIlroy recommends (slightly modified) *creating components that do one thing and do it well* (McIlroy, Pinson, and Tague, 1978).

The component design, which is a core feature of the Unix operating systems, has proven its versatility as a problem solving paradigm. The basic idea is to divide problems into subproblems that are easier to solve. See “Doing the actual recording” in Section 5.2 on page 59 for an example application.

It is helpful to attain a higher level of abstraction in design, making it possible to ignore underlying complexity; this necessitates that the possibilities and limitations of the component interfaces are well understood. Another desirable trait is that the design should be platform agnostic, so it can be implemented on any platform.

In order to attain all of the criteria, some radical design decisions needed to be taken. To be able to operate long-term the system needs to be robust. It needs to be self-sufficient with regards to power but almost all test versions had power from the grid. The microphones need to be protected as prolonged exposure might damage them.

Goals

As mentioned in Section 2.1 on page 7, we want to create a low-cost ultrasonic recording system that is capable of long-term surveys, is massively distributable and uses, so far as possible, off-the-shelf components and traditional communication protocols.

Based on the arguments mentioned above and those put forth by the open source community as a whole, we have chosen to base this project as much as possible on solutions which are licensed with Open Source Initiative (Tiemann, URL provided, accessed: July 2011) compatible licenses, i.e. those based on the Open Source Definition (Coar, URL provided, accessed: July 2011). This work is licensed in a similar manner so the non-trivial solutions found here that are relevant to others have been documented to make them easier to reapply in a similar setting.

In addition, the system needs to be modular, where each module is self-sufficient in regards to analog-to-digital conversion, storage, memory, and computing power. At the same time we wish to use retail components and open source solutions.

A diagram of the design can be seen in Figure 2.1. The system consists of multiple ultrasonic microphone array recordings systems, i.e. module 1 through n . For all modules, the transducers (microphones) convert the pressure change impinging on them into a voltage. The microphone signal is amplified and filtered. The signal is then digitized in an analog-to-digital converter, ADC. Finally the signal is passed on to a computer, which further processes, stores, and/or forwards it.

As illustrated in Figure 2.1, all modules can be considered as *one* array recording system provided they all have a common time reference. In this way we can obtain an arbitrarily scalable transducer recording system. As the figure also shows, this system can be controlled through standard networking technologies, such as LAN (IEEE 802.3), Wi-Fi (IEEE 802.11), GSM (ETSI), etc., by a computer, smartphone or other device supporting the control protocol. Recorded data can be forwarded to a Network Attached Storage box, i.e. a network-accessible hard disk. When recording

continuously, the network bandwidth will limit how many channels can operate simultaneously, so to achieve arbitrary scalability local storage must be used.

2.6 Summary of design choices

In conclusion, the important characteristics of the design are as follows:

- Modular design.
- Customized signal adaptation electronics.
- Solid state microphones.
- Open standards.
- Open source solutions.
- Open source supported hardware.

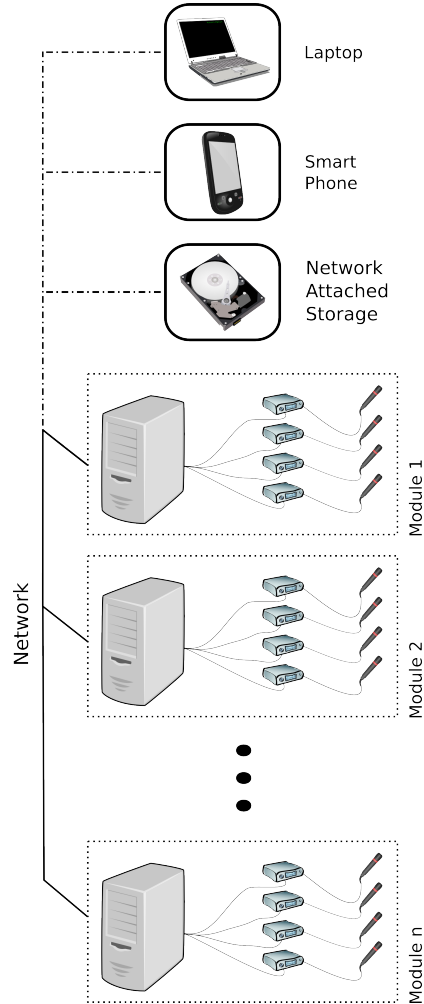


Figure 2.1: Block diagram of a generic modular ultrasonic microphone array. Note that even if the modules have a limited number of microphones, no restrictions are put on the total number of modules. The analog voltage from microphones is filtered and amplified and A/D converted into digital information which is then handled by a computer. The array computer controls triggering, recording, and optionally emission of ultrasound. **Legend:** Computing power and data storage/handling, Filter, amplification, and AD conversion, Ultrasonic microphone.

Prototype implementation

*The computing scientist's main
challenge is not to get confused by the
complexities of his own making.*

— E. W. Dijkstra

This implementation has striven to use existing solutions where possible, and only implement customized solutions when absolutely necessary.

3.1 First prototype

The first prototype implementation is based around the Adlink DAQ-2010 A/D converter because of its availability and applicability to the task. As found in Section 2.5 on page 16, it was necessary to implement some customized electronics to adapt the microphone signals to our A/D converter and provide correct power to the microphones' pre-amplifiers. The systems computer was chosen based on price and that it would support the DAQ-2010.

The system is designed and implemented around a commercial analog-to-digital and digital-to-analog converter which is capable of reading 4 channels at 2 MHz sample rate and emitting 2 channels at 1 MHz sample rate. This analog IO hardware is controlled by a standard computer which is joined to other computers, also with analog IO hardware embedded. Note that this makes the system arbitrarily scalable both for receiving ultrasound and for emitting ultrasound. The only requirement is that all systems have the same time of reference, and as each analog IO hardware is also equipped with trigger inputs this was straightforward to implement.

Analog filter and amplification

Designing custom hardware is not a trivial task, and we have sought to limit the exercise. When we had to, it was designed as generically as possible and the design is published in Brandt (URL provided, accessed: Dec 2011).

This entails a simple open design utilizing cheap off-the-shelf and easily replaceable components.

Attaching ultrasonic sensors to a standard computer requires some specialized amplification and power supply; some customized hardware has therefore been developed. This is not needed for some high-end microphones, which are sold with filters, pre-amplifiers and amplifiers. Although for these devices signal conditioning might still be needed to increase the dynamic range of the conversion or to convert voltage levels to those supported by the analog-to-digital converter.

We have emphasized making the system as open and accessible as possible. Open source software is today a widespread and well recognized paradigm. Open source hardware, on the other hand, is not so widespread although it is not a new phenomena (Ackerman, 2008).

Our custom-designed hardware includes bandpass filters and analog amplification for each channel. The analog bandpass filter for the system is a 2nd order Bessel biquad filter with 6 dB high-pass at 15 kHz and 6 dB low-pass at 150 kHz. The analog amplification on each input channel is programmable, which allows each channel to be attenuated or amplified individually from -20 dB to +60 dB. As the pre-amplifier of the chosen microphones for this system requires 1.6 V, this is also provided. The hardware supports voltages in the range $\pm 10V$, which means that a wide range of different transducers are directly supported. The source for the filter and amplifier has been published at (Brandt, URL provided, accessed: Dec 2011).

The filter and variable gain were tested by Ali Shekarchi. The results from `box6` (amplifier box number 6) are shown in Figures B.1 to B.3 on pages 130–132. The abscissa is given in the internal gain value range, where 200 is unity and 450 is circa 60 dB. Attenuation was not tested. The plots show that within the 200-450 range the setting and amplification have a hyperbolic relationship. They also show that the amplification is frequency invariant and amplitude invariant, although note that saturation is reached (circa 10 V) when the input signal is strong.

Microphones

The microphone mainly used in our prototype is the FG23329-PO7 from Knowles Acoustics (Knowles Acoustics, URL provided, accessed: Aug 2012). Originally designed for cell phone use, this microphone has been tested to be usable for ultrasonic reception and is sensitive up to approximately 140 kHz. It is also cheap, circa 40 USD, and small, 2.54 mm in diameter with a 1 mm acoustic aperture, making it a near ideal choice given the criteria for the system.

As illustrated in Figure 3.1 on the next page, 4 microphones are connected to the filter/amplifier box, using the connectors which have been used for all prototypes/versions of the analog part of the system. To make the system rugged we used IP68-certified connectors (Hirose Electric type LF07WBJ-6P and 6S) so that the joints are dust proof and submersible in water to below 1 m . The microphone cables are commonly available shielded cable (Carol Brand type C0743A.21.10).

The DAQ-2010 analog to digital converter

For A/D conversion we used a DAQ-2010 data acquisition board from Adlink Technology Inc. (Adlink Technology inc. URL provided, accessed: May 2012), chosen for its hardware specification and relatively open driver software. It provides 4 channels with an analog-to-digital conversion rate of 2 MSample/s/channel at 14 bit/sample. In addition, it has 8 digital input channels which are synchronized to the analog input. It also provides two 12 bit digital-to-analog channels with a maximum sample rate of 1 MHz/s/channel. Finally, it has 24 bit programmable general purpose digital input/output. It comes with proprietary binary Linux drivers.

Mini-ITX system computer

The smallest computer that provides the required PCI port for the A/D board is a Single Board Computer with the Mini-ITX form factor (mother-board size of 17x17 cm) defined by VIA Technologies (Via, 2001). Figure 3.1

shows the first prototype. From the node computer the customized hardware is connected to the A/D board through an off-the-shelf SCSI cable (Wikipedia, URL provided, accessed: Apr 2012).

We have used multiple Mini-ITX motherboards built by Intel and AMD, respectively based around the Atom and Geode processors. This has not presented any major issues, as the operating system hides the implementation details from us, and because the architectures are closely related.

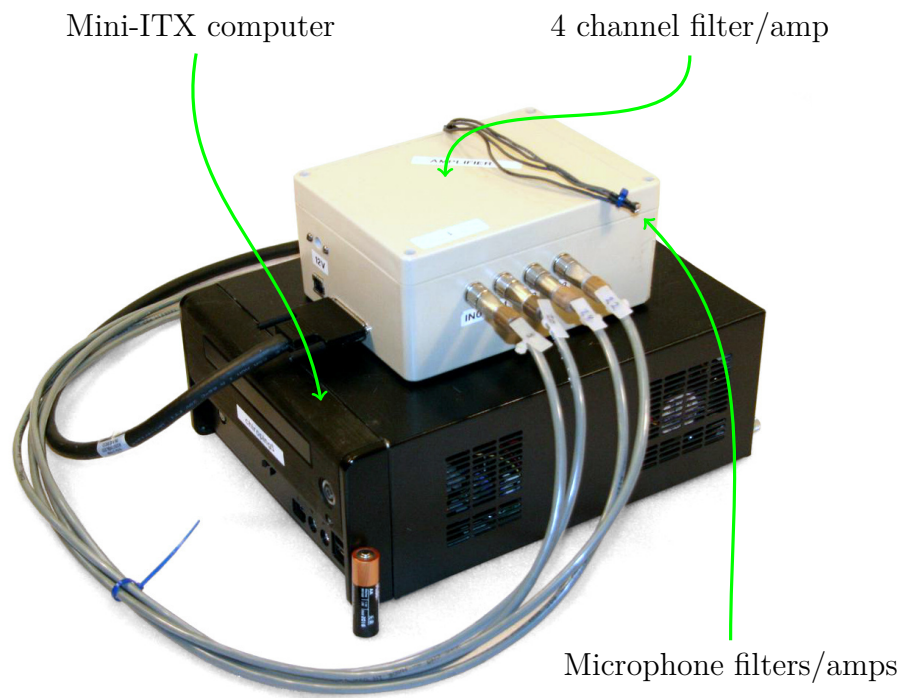


Figure 3.1: First array module prototype. Scale is shown by the AA battery in the foreground. The computer and ADC are in the black box, whilst the custom filter and amplifier hardware is in the light box. Four Knowles microphones are bundled and placed on top of the filter/amplifier box.

3.2 Second prototype

The first prototype is rather bulky, so from the beginning there was an ambition to implement a similar system with smaller hardware. One year into the project, components satisfying our needs turned up in the marketplace, and some effort was put towards using them within the same design. Incidentally this is a good demonstration of the platform independence offered by the MCLURS design; it may be worth noting the process by which a portable and low-power prototype has been developed.

The prototype is based on the open source/hardware A/D converter *USB-DUXfast* (Incite Technology Ltd. 2012) and the DreamPlug computer (Globalscale Technologies Inc. URL provided, accessed: Jul 2012[a]; Globalscale Technologies Inc. URL provided, accessed: Jul 2012[b]), but any ARM based SBC (single board computer) would do. The connection between the computer and the A/D converter is USB based so the need for a PCI slot is removed. This combination of A/D converter and system computer offers a compelling framework in which to combine multiple units. One implementation of the second prototype can be seen in Figure 3.2.

Everything that was custom-built for the first implementation can be reused with small modifications. The only substantial change is to the interconnection between the A/D converter and the filter/amplifier box. The DAQ-2010 uses a mini SCSI connector (Small Computer System Interface ANSI (1994); Wikipedia (URL provided, accessed: Apr 2012)) type VHDCI female whilst the new A/D converter (Incite Technology Ltd. 2012) uses a female 44 pin HD connector see Norcomp Inc. (URL provided, accessed: May 2013) for an example.

The customized filter and amplification electronics used with this system are the same ones used with the first prototype; the same is true for the microphones used.

Note that this is still work in progress. We have successfully tested the A/D converter with a plug-computer in lab settings. We have not yet tested multi-system deployments. A new filter/amplifier box with the correct interconnection still needs to be made, although the current implementation

can still be used with some makeshift wiring.

The USB-DUXfast analog-to-digital converter

The A/D converter for the smaller system is connected to the computer through a USB 2.0 link. It provides 16 channels A/D channels which are sampled at max 3 MHz total at 12 bit single ended resolution. Note that the channels share this sample rate, which implies that we can at most have 187.5 kHz sample rate per channel, but if each signal is connected to 4 input channels, e.g. channel 1, 5, 9 and, 13, then the effective sample rate would be 750 kHz on that signal, with the total number of channels reduced to 4. The *USB-DUXfast* is a low-power unit running at around 1 Watt.

Due to limitations in the hardware, the *USB-DUXfast* can only sample 1, 2, 3, or 16 channels. This meant that to achieve 4 channel recording, each signal was connected to multiple channels and “reshuffled” in software.

Dreamplug system computer

We have initially tested the *USB-DUXfast* with a Dreamplug computer. Some highlights from this computer: 1.2 GHz Marvell Sheeva CPU, 2 x USB 2.0 ports, 2 x Gigabit Ethernet ports, 1 x eSATA 2.0 port, and built-in Wi-Fi ethernet. This is a very low-power unit as it is based on the ARM architecture, using only around 5 W. The low power consumption would permit longer operational deployments if running on batteries.

3.3 Summary

There are currently two development lines. The first prototype, which is a multipurpose system that can be applied to many different experiments, and the second prototype, which is well suited for most of the recording tasks except those involving emission. The strengths of the second prototype are its portability, size, cost, and low power consumption.

The first prototype has been successfully deployed under many different conditions (see Chapter 4) and has proven the validity and durability of

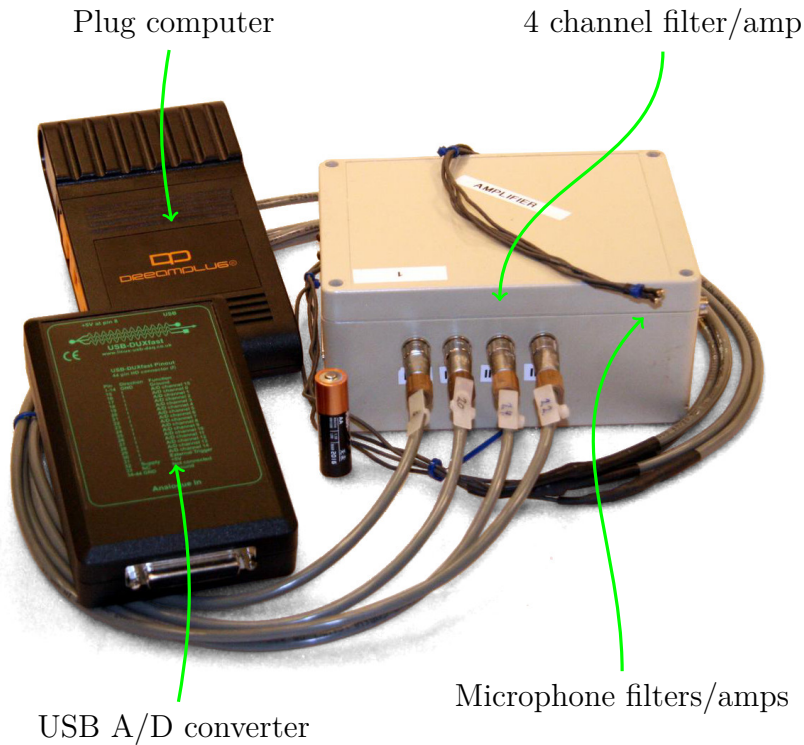


Figure 3.2: Portable low-power prototype. Scale is shown by the AA battery in the foreground. The computer is in the black and yellow box at the back, while the A/D converter is in the black and green box in the front. The interconnection between the A/D converter and filter/amplifier box still needs to be modified.

the design. The choice of the A/D converter, however, has caused many issues as there have been several instances where the systems become inoperable because of driver crash. As the driver is the sole responsibility of the vendor, perhaps a more open solution should be considered for a future multi-purpose system.

We are satisfied with the current state of the second prototype, but it has its limits. For instance, the sample depth is only 12 bit. It is inconvenient to sample more than 3 channels. It does not support synchronized emission. Given time and resources it would be useful to construct a new A/D-board, which would eliminate these limitations.

Experiments and evaluation

*Science is the belief in the ignorance
of experts.*

— Richard Feynman

The first prototype has been deployed in seven different setups which demonstrate its capabilities: one with a high number of channels; long-term recordings, operation under harsh conditions; and several synchronized playback/recording. The prototype has been deployed with several different transducers in the different setups and we took specific measures to deal with the large amount of data generated. In one setup we also attached a weather station to the prototype.

Many of the experiments described in this chapter rely on calibrated microphones from B&K and G.R.A.S. We compared them to each other which has convinced us of their reliability as a “measuring stick” for transfer functions. The transfer function for these calibrated microphones is flat in the frequency range we are interested in. Given in Figure 4.1 as a reference is the transfer function for a G.R.A.S. model type 40BF (G.R.A.S. URL provided, accessed: May 2013). It was very similar to a $\frac{1}{4}$ inch B&K microphone’s transfer function, only different in its sensitivity.

4.1 Manual triggered deployment — medium-term, many microphones

Multi-channel arrays enable us to increase the recording area and/or increase the spatial recording resolution. This helps us learn more about, e.g. what a bat is doing, as we can see it in greater detail. Thus this is an important test for the system.

The system was deployed to record *Noctilio leporinus* and *Micronycteris microtis*, flying in two outdoor flight rooms at the Smithsonian Tropical Research Institute’s field station on Barro Colorado Island, Panama in March

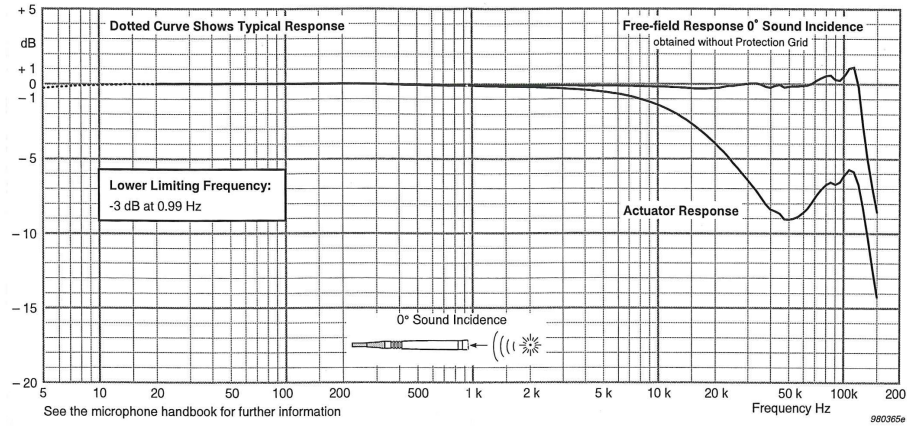


Figure 4.1: Transfer function for a G.R.A.S. $\frac{1}{4}$ inch microphone model 40BF. We used it to find the transfer function of other equipment.

2010 as part of fieldwork for the *ChiRoPing* project (ChiRoPing, URL provided, accessed: May 2013). The microphone arrangements were “T”-shaped and “H”-shaped respectively, and in both cases placed to face the bat as it approached the array. Three modules were used, so a total of 12 channels were available. Figure 4.2 on page 32 shows the spectrograms of a 1 s long sample taken out of a 1 minute *M. microtis* recording. 10 microphones were used; the eleventh channel recorded a synchronization signal for off-line synchronization with high speed video recordings. The twelfth channel was not connected and has been omitted from the figure.

N. leporinus and *M. microtis* have very different hunting behaviors. *N. leporinus* is a trawling bat (Schnitzler, Kalko, et al., 1994), while *M. microtis* scans leaves for prey (Geipel, Jung, and Kalko, 2013). Thus recording these two bat species poses different demands on the system. *N. leporinus* searches for prey on or over smooth water surfaces and once detected, swoops down and catches the prey. Capturing a whole hunting sequence requires a recording length of about 2–3 seconds with a 2 second pre-trigger time to be sure to capture the sequence from the very beginning. *M. microtis*, on the other hand, can scan leaves for up to 2 minutes before finally taking the prey. This posed no problem for this system as it was a mat-

Nº	<i>Micronycteris microtis</i>			<i>Noctilio leporinus</i>		
#1	41 files	à	630 MiB	41 files	à	40 MiB
#2	25 files	à	573 MiB	20 files	à	92 MiB
#3				9 files	à	50 MiB
#4				10 files	à	69 MiB

Table 4.1: Summary of *Micronycteris microtis* and *Noctilio leporinus* recordings, Nº refers to the number of the recording session.

ter of increasing the pre-trigger buffer length to a couple of minutes. The disparity in recording strategy is also reflected in the number and length of the recordings, see Table 4.1.

A cursory look at the data has convinced us that something is recorded, however, analysis of the captured sequences remains to be done. Most likely this will require automatic methods to be applied. Section 5.3 describes a method that could potentially be used to extract and classify *M. microtis* sequences from these recordings.

4.2 Robust long-term recording, 4 microphones

In the past decade worries about decline of biodiversity have fostered growing interest in technology for environmental monitoring, using surveys of animals as diverse as insects, birds and bats, which can automate species recognition based on acoustic communication (Blumstein et al., 2011). For bats and birds, special concerns have been linked to the growing number of large scale wind turbine parks (Kunz, Arnett, Erickson, et al., 2007).

Today, state-of-the-art bioacoustic research requires high-sample-rate, multi-channel, and often long-term recording systems. One application of long-term recordings has been when building large structures, because there are increasing concerns about the impact of such structures on local bat populations. The concerns involve, for example, habitats, foraging grounds and migratory routes. Many methods of assessing the impact

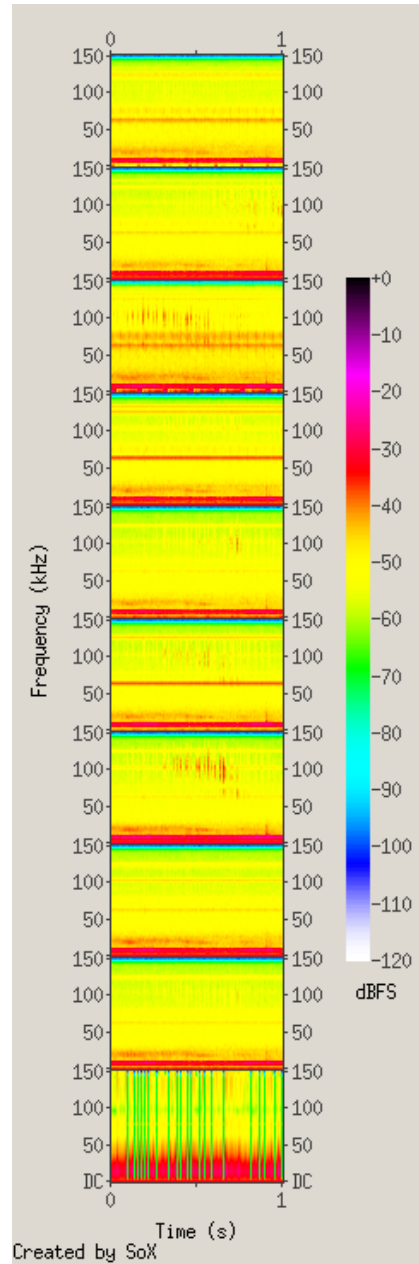


Figure 4.2: Multi-module recording done in Panama in March of 2010 (session 9 recording 21). Note that only microphone 3 and 7 have clear reception. These are calls of *Micronycteris microtis* scanning a leaf in search of prey. The recording consists of 11 synchronous channels. The 11th channel recorded a synchronization signal.

are used: carcass counting (Fiedler, 2004; Barclay, Baerwald, and Gruver, 2007), radar tracking (Bruderer and Popa-Lisseanu, 2005; Kunz, Arnett, Cooper, et al., 2007), infrared thermography tracking (Hristov, Betke, and Kunz, 2008) and passive acoustical measurements, usually with one or more bat-detectors (Ahlén et al., 2007; Baerwald and Barclay, 2009). Each of these methods has benefits and drawbacks.

When trying to survey a large area, none of these methods covers all of the following important aspects: 1) estimating how many bats have passed by, 2) estimating what species have passed by, 3) estimating their flight paths, and 4) supporting automatic unsupervised long-term survey. Carcass counting is done too late and does not reveal the flight behavior, death rate and reaction to the structure. Radar tracking and stereo infrared thermography can give suitable data, at least for large species, but availability and price make them infeasible; multiple bat-detectors could, with some effort, be used to automatically count passing bats and estimate their species — but unless they are synchronized, flight-path extraction is difficult.

In principle, an acoustical array would provide all the information needed as bats echolocate when commuting or hunting. Bats hunting for insect prey out in the open emit high-intensity calls (Jensen and Miller, 1999; Surlykke and Kalko, 2008). The MCLURS system exploits this fact. This gives the system an inherent bias towards loud bats; but windmills present the largest danger to bats out in the open hunting or migrating (E. F. Baerwald, 2011; Barclay, Baerwald, and Gruver, 2007) and since those bats are among the loudest this bias might not be a large issue.

This mode of surveying has been tested with four separate outdoor deployments. There are a couple of important aspects of this type of monitoring: electronics need to be protected, the microphones need to be protected but still be acoustically unencumbered, supply power should be connected, and enough storage space should be available.

Typical experimental method

The typical method we have used is to mount MCLURS inside a weather-proofed box (Fibox type EK-W) rated at IP65, which means it is rain and dust proof. Only microphone cables and power enter/leave through weatherproofed holes in the box, that is, if the system is not battery operated. The Knowles microphones used are weather resilient; pointing them downwards with a water guide gives them some water protection. One setup even featured a shielded microphone. Both methods of protection can be seen in Figure 4.3 on the next page. Which one works best still needs to be quantified. This type of weather hardening has proven itself to be sufficient. The system has been used for recording right before the wet season in Panama with well above 90% relative humidity; it has also recorded during the Danish winter with sometimes freezing temperatures and winds as high as 20 ms^{-1} .

Long-term recordings generate large amounts of data, e.g. for a node with 4 channels sampling at 500 kHz:

$$4 \text{ channels} \times 2 \frac{\text{bytes}}{\text{channel}} \times 500\text{K} \frac{\text{samples}}{\text{seconds}} \times 24 \text{ hours} \approx 322 \text{ GB}$$

Analyzing such an amount of data manually would be very time-consuming, so automated methods are needed. We have used a simple recording paradigm to reduce the amount of stored data: the system was set to a wake-up/shutdown cycle where it started recording at 8 PM and stopped at 6 AM which reduces the recording size to $10/24$ of the total. When recording, only signals which exceed a triggering threshold are stored. The threshold was set at 5 times the noise floor of the topmost microphone. To trigger a recording, the threshold had to be exceeded for at least half a millisecond; at that moment the system stores the pre-trigger buffer and continues recording until 2 seconds of “silence” occur (i.e. below threshold). This has a major impact on the data volume produced. More advanced triggers could be developed, for example using acoustic profiles for a matched filter bank. Techniques have been developed for extracting and classifying huge amounts of data, please refer to Section 5.3 on page 61 for the details.

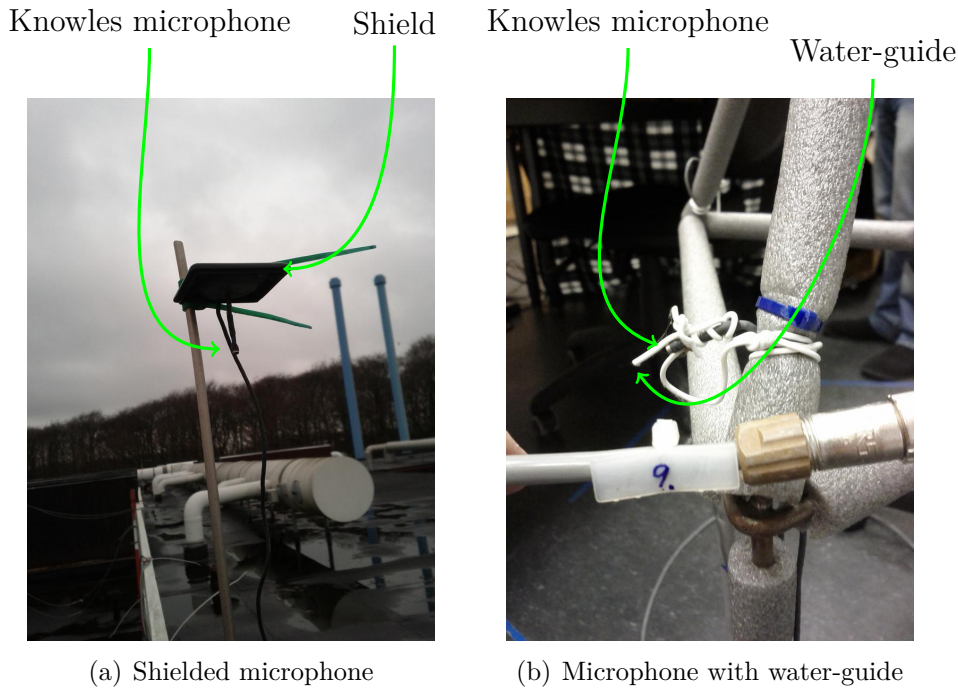


Figure 4.3: Microphone protection strategies. The microphone in (a) is mounted with strips on the same stick as a plastic cap which provides cover. The microphone in (b) is not as well protected, but this technique ensures that no water droplets linger on the microphone.

Deployments

A four channel system was deployed on the roof of the university from summer 2010 to summer 2011. There were only minor disruptions, and only one failing microphone, even though the system remained outdoors in Denmark throughout the winter. This proved to be a very low activity area so we have almost no interesting data. In 2012 we deployed two systems, one in the tree line south of the university, and one closer to the city center but still in a green area. Unfortunately, the first system was too far inside the tree line and became covered by foliage. It was too dangerous to move as it was placed 10 meters off the ground. The second system had a technical failure, the nature of which is still to be determined. Finally we deployed a system in the late summer of 2011 which had more success. The system was

deployed for more than three months and recorded for approximately two of those months. Operational failures were due to accidental disconnection from the power grid by university ground staff. The details of this survey are copious and have thus had a whole chapter dedicated to them (Chapter 6 on page 73).

Summary

We are encouraged by the success in the three month survey, which has brought some new biological facts to light. We are sure that this type of survey is an economical and biologically significant way of assessing bat presence. Although it is biased towards loud bats this can be ameliorated to some extent, see discussion in Section 6.6 on page 84.

The placement of the array is very important and changes in its environment need to be anticipated. The system's function needs to be assessed frequently during the first days of deployment, so access is important at that time. Some sort of keep-alive check should be employed, perhaps by sending back recorded data during daytime.

4.3 Echolocating robot

This experiment was done to guide a robot with the help of ultrasound. At the same time the sensor should be able to discriminate and identify ensonified targets. In order to achieve this a combined emission and reception sensor was built, two instances of these can be seen in Figure 4.4 on the next page.

Setup

To add flexibility to the test, the echolocation sensor was mounted onto a mobile robot with 6 degrees of freedom. This allows us to position the sensor at various positions, heights and angles relative to the floor, and thus makes it easier to determine optimal conditions for the given task. An annotated picture of the robot can be seen in Figure 4.5 on page 38.

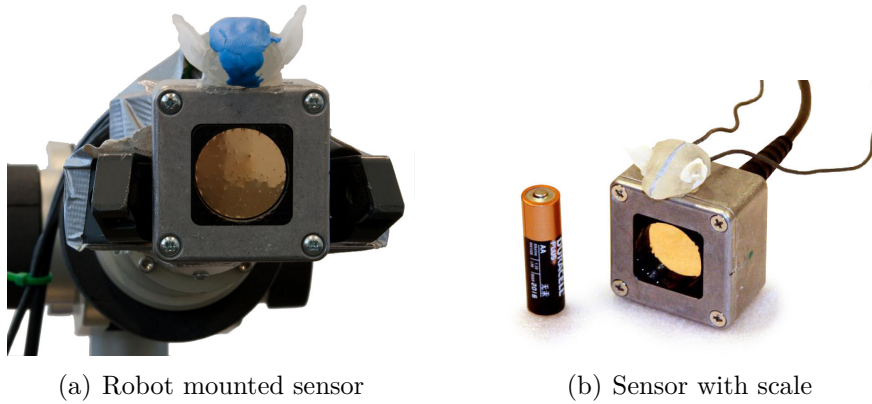


Figure 4.4: Two implementations of the echolocation sensor. The sensor in (a) has blue sticky tack attached to the top to reduce reflections from the joint of the two head-halves. (b) was better assembled and needed much less substance to hide the crack. Note that (b) also includes an AA battery to show the scale of the sensor.

Due to availability and requirements a Neuronics Katana 450 robot arm (Neuronics AG, URL provided, accessed: Sep 2012) was mounted on top of a Pioneer 3-AT mobile robot (Adept MobileRobots LLC, URL provided, accessed: Sep 2012). The arm has 3 degrees of freedom and the mobile robot has 3 degrees of freedom, yielding a very agile robot capable of very precise spatial control of the echolocation sensor. The robot was informally named *Natasha* and has a rather complicated control surface as two separate robots were involved.

Here is an outline of the essentials of the control interface. The mobile robot and the arm robot had different user interfaces, so some glue programming was needed to make them work seamlessly together. A brief description and core elements are described here, see appendix E on page 153 for a more detailed description. Each robot had a different control interface, the mobile robot through the Player/Stage API (Various, URL provided, accessed: Sep 2012[d]) and the arm robot through a proprietary control library. Because of the proprietary nature of the arm robot interfaces and the discontinuation of its maintenance, the control required two separate programs to be written: one main program to interpret and convert com-

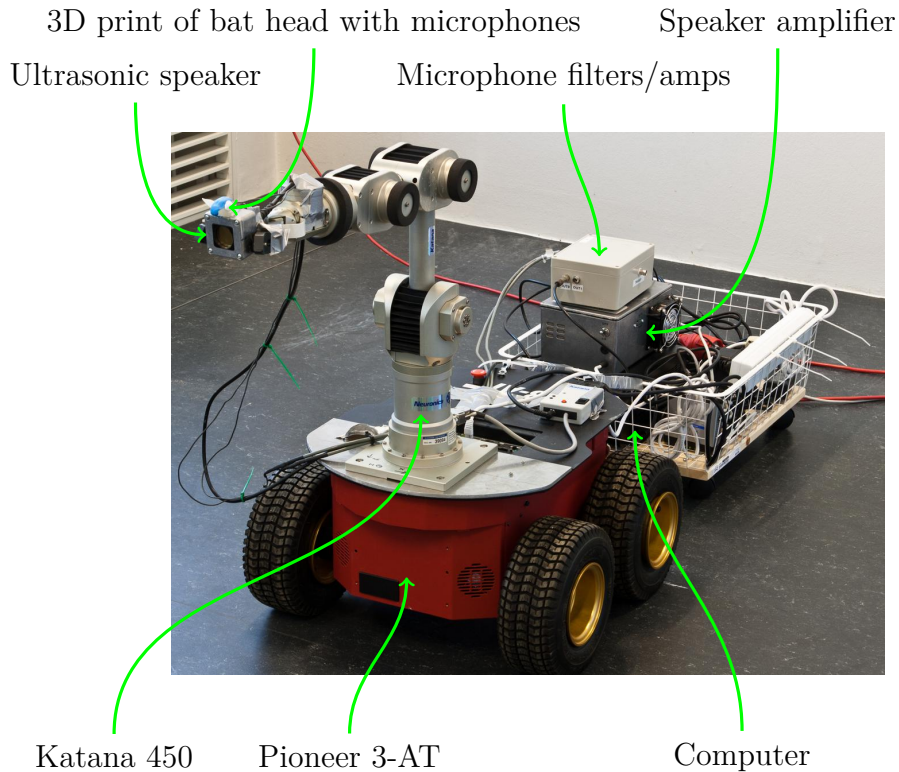


Figure 4.5: Echolocating robot, a demonstration of synchronized playback and recording. The robot consists of two robots joined together — a Neuronics Katana 450 robot arm with 4 degrees of freedom while the mobile robot is a Pioneer 3-AT robot from Adept Technology Inc.

mands received from the signal processing computer, and one that would be compatible with the old proprietary library.

This robot has been applied to two different experiments, object locating and object identification. It has performed satisfactorily, acoustically speaking, and successfully located object. Identification however has proved a more difficult task so far.

Note that this is still an ongoing project so only the setup and preliminary results are presented here. As the robot was assembled and customized for this experiment, its control and echolocation programming are described briefly.

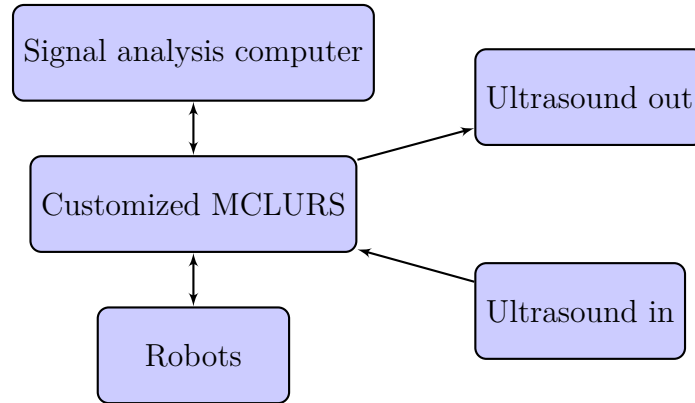


Figure 4.6: Diagram of echolocating robot control. Ultrasound emissions and control commands are sent from the signal analysis computer to the MCLURS computer, which returns two channels of synchronized ultrasound recordings.

Experimental method

Figure 4.6 shows a block diagram of the setup. The signal analysis system is the controller and receives feedback through the ultrasonic emission/reception loop. Based on what is received here the signal analysis system decides the appropriate direction to move and sends its command to MCLURS.

The object was placed on a fairly reflective floor and the robot placed a couple of meters away. Typical objects used can be seen in Figure 4.7 on the following page. The robot then proceeded to emit 2 ms ultrasonic down-sweeps (90 kHz to 20 kHz) towards the target and toward $\pm 15^\circ$ azimuth angles. The echoes from these emissions were then compared and target position was estimated. The robot proceeded at the estimated direction-to-target vector for a predefined distance and the emission was repeated.

Preliminary results

A typical left-ear/right-ear recording can be seen in figs. 4.8 and 4.9. Notice that there are three distinct peaks in each recording, most visible on the spectrogram. These peaks come from, in order: the primary emission, the

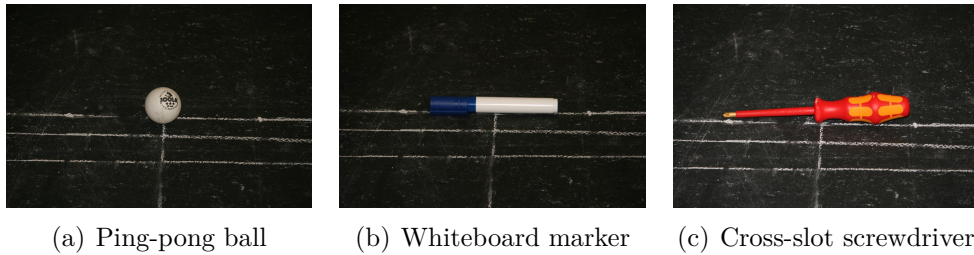


Figure 4.7: Objects to be located by the echolocating robot.

reflection of the floor directly below the sensor, and the echo from the object.

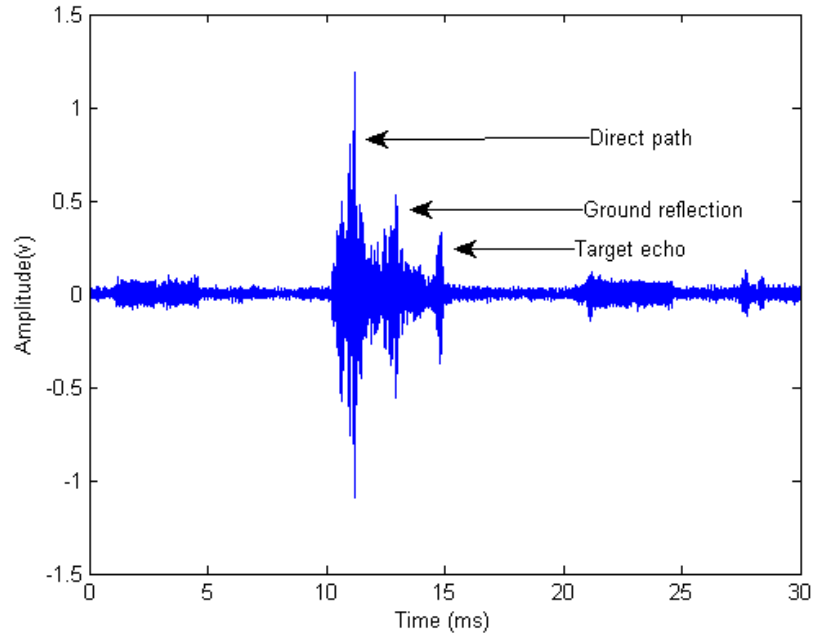
The recordings have a fair amount of noise in them, we believe that this originates from the robots, but further investigation is warranted.

Summary

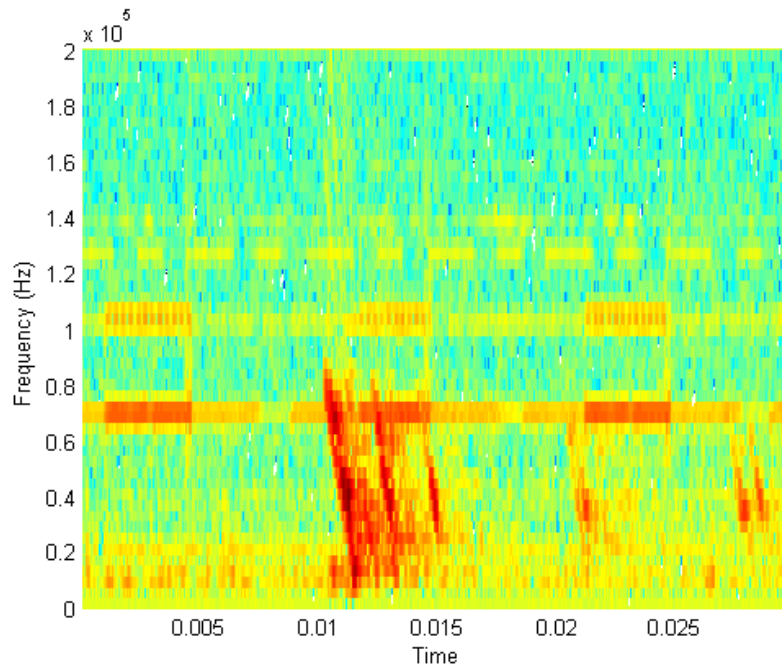
As mentioned this is an ongoing project and much more work needs to be done before any conclusions can be drawn. There are some indications that the Katana arm robot will be supported by the Player/Stage framework, thus simplifying the control surface a great deal. This should be taken advantage of.

4.4 Calibration setup — Calibrating microphones and transducers

In order to find the transfer function of the various transducers and signal chains we have used MCLURS to build a calibration setup. The synchronization of input and output using the DAQ-2010 lends itself readily to linked emission and recording, which is not only useful for echolocation, as described in Section 4.3, but also makes averaging of data from many emission-recording cycles trivial. If there still are alignment problems then Algorithm 4.1 on page 49 can be used.

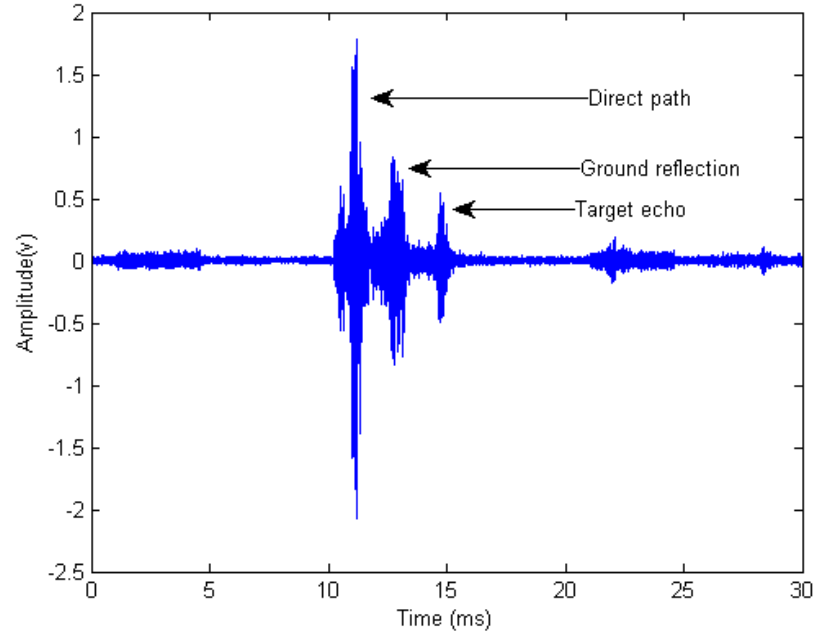


(a) A-plot – left ear

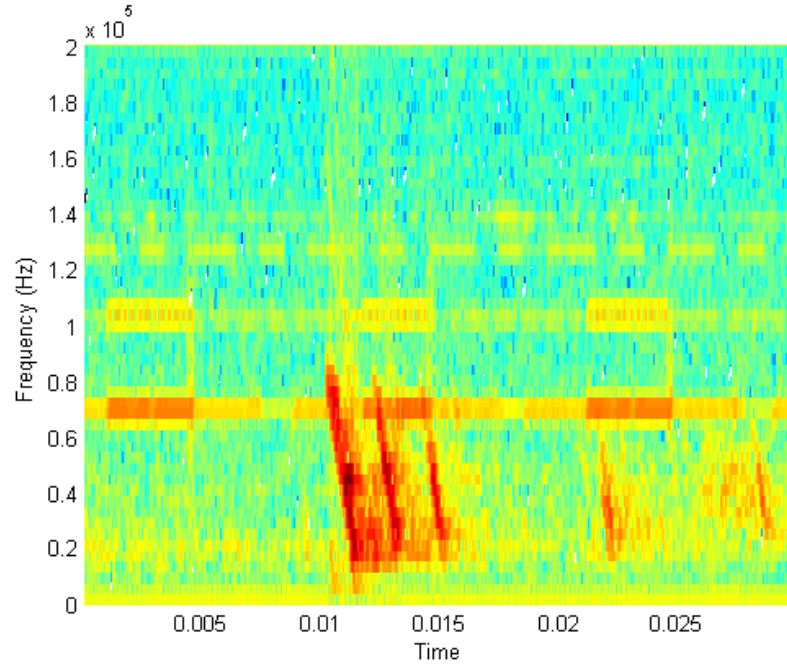


(b) Spectrogram – left ear

Figure 4.8: Typical recording when locating objects – left ear.



(a) A-plot – right ear



(b) Spectrogram – right ear

Figure 4.9: Typical recording when locating objects – right ear.

Experimental method

The frequency response of the emitter is recorded with a calibrated microphone in a reasonably anechoic place. By replacing the calibrated microphone with the microphone that needs to be calibrated, we acquire data to compute the transfer function of the microphone needing calibration.

Types of equipment that can be calibrated

The recording system will work with input from any transducer delivering voltages of ± 10 V peak, e.g. ultrasonic microphones from Knowles Acoustics, Avisoft, B&K, and G.R.A.S. The analog output channels have a maximum voltage range of ± 10 V and have been used to drive a number of different speakers. The output impedance was chosen to be $50\ \Omega$ to be compatible with the input for our custom-designed power amplifier for the Senscomp's Series 7000 (22 mm) Polaroid transducer (SensComp, URL provided, accessed: Aug 2012), but this can be changed if need be.

The setup can be seen in Figure 4.10, notice the extra soundproofing material placed in the environment of the setup, especially in the path of the main emission. This is so to minimize overlap of primary transmission and reflected transmissions.

Results

Ali Shekarchi has used the calibration setup to calibrate most of our Knowles microphones with the method described. Some of them were calibrated with extension cables attached to test their effect on the whole system. Figures B.4 to B.10 on pages 133–139 show the results of these tests. The calibrations show that all of the Knowles microphones have reduced sensitivity at higher frequencies. The “ripples” in these plots we think are related to the makeshift quality of the setup, which was quickly assembled in our lab. So better tests are needed to verify the transfer functions.

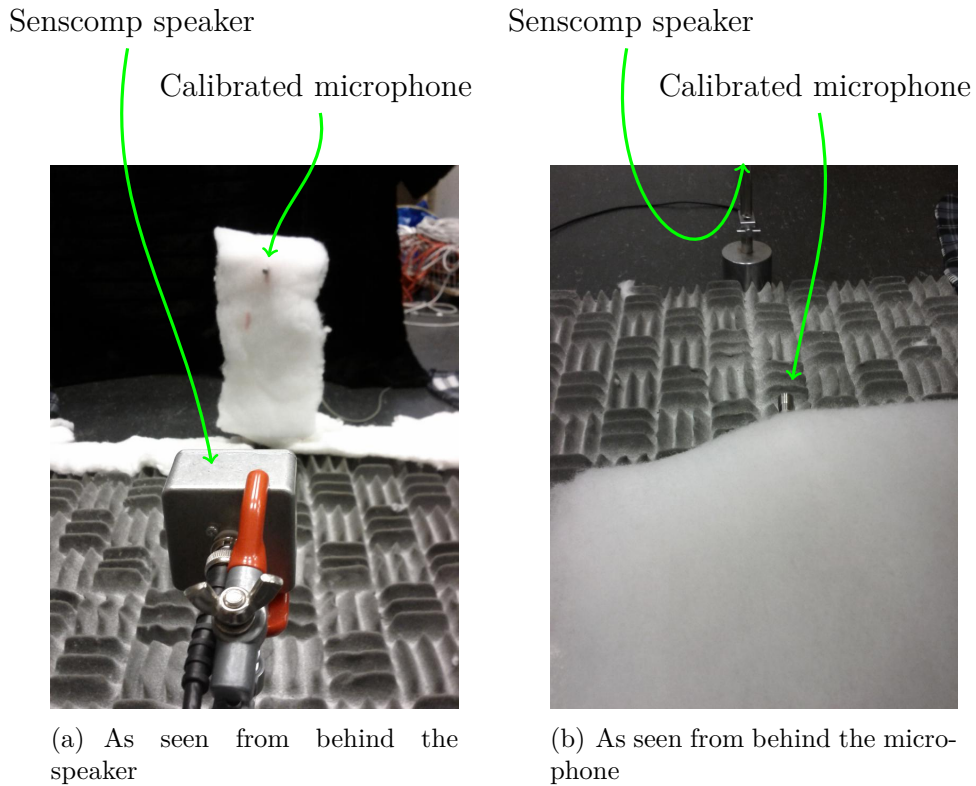


Figure 4.10: Calibration setup. We emit several signals to a calibrated microphone. An uncalibrated microphone then replaces the calibrated one and is subjected to the same type of sound field. Comparing what the calibrated microphone records and the uncalibrated microphone records allows us to find the transfer function of the uncalibrated microphone.

Summary

This is a very useful setup and would help us compensate for calibration any recordings we previously made, assuming that the Knowles microphone's transfer function is fairly stable over time and environment. This is an assumption that still needs to be verified.

It is clear that this setup is lacking in the anechoic regard. The setup needs to be much better built up, so as to reduce reflections. We do think that this setup allows us to get a qualitative estimate of the transfer function of the measured equipment and that is sufficient in many regards.

4.5 Discovering the echo profile of an object

In order to be able to identify objects based on their echo profile, a setup was built that would give us an idea what these “look” like by ensonifying known objects from equidistant azimuth and elevation angles. To automate this process a turntable was acquired. This allowed us to programmatically control the experiment.

This is still an ongoing project so only the technical details of the setup and some preliminary results are presented here.

Experimental method

We used a modified Brüel & Kjær type 9640 (Kjær, URL provided, accessed: Sep 2012) (Figure 4.11). The modification consisted of a new motor controller, namely a Samotronic SE2 (Samotronic, URL provided, accessed: Sep 2012). The communication from the ultrasonic recording system computer to the motor-controller was handled by a generic digital and analog IO board from Velleman (type K8055), both can be seen in Figure 4.12.

The microphones and speaker were the same as the ones used in Echolocating robot (Section 4.3) and in the same configuration, i.e. microphones embedded inside a plastic print of a 1.5 scale *Myotis daubentonii* glued on top of a speaker (Figure 4.4 on page 37), but this time mounted on a laboratory pole instead of a robot arm.

The sensor was attached to a vertical bar from which arbitrary elevation and emission angle could be controlled. As this was done manually, the estimated measurement error in elevation is $\pm 1/2$ centimeter and angle of emission $\pm 2^\circ$. The speaker’s output cone is -6 dB at $\pm 7.5^\circ$, so this should be sufficient for representative results. An object is placed on the turntable, for each angle a number of emissions/receptions were done.

Summary

As mentioned earlier, this is still work in progress and no definitive results have yet been produced. However, the system has proven itself easily usable

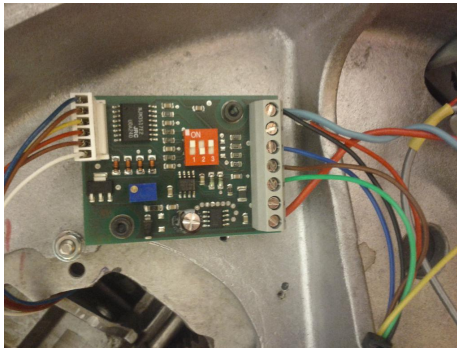


(a) Turntable with customized object holder

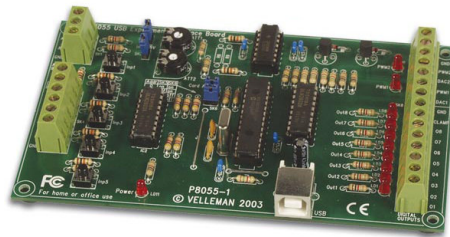


(b) Turntable before it was customized

Figure 4.11: The B&K turntable used for finding echo-profiles of objects.



(a) Samotronic motor controller SE2



(b) Velleman K8055 analog/digital IO-board

Figure 4.12: Motor-controller and digital IO used with turntable.

in this application.

4.6 Calibrated emission

It is often desirable to avoid the filtering effect that the emission has on an output signal, for example when trying to affect bats by playing back bat sounds. In order to emit a predictable signal, the transfer function of the whole chain needs to be known. Chain here is everything that affects the signal levels from a digital number inside the computer until it is a pressure

wave in the air — that means converters, amplifier, and transducer. How each device affects the level is often frequency dependent.

Theory

Signal theory tells us that in order to recover the effects of the chain we must convolve the input signal with the reciprocal of the chains' transfer function, or in the frequency domain we can divide the recorded signal by the emitted one. To find the transfer function we record signals containing frequencies of interest on a calibrated microphone.

Here is a more formal discussion of the same concepts. In the following lower-case letters signify signals in the time domain and upper case letters signify signals in the frequency domain. Let x be the input signal and h be the transfer function of the intervening equipment, then x will be convolved with h upon emission thus yielding $x*h$ at the output, i.e. the pressure wave. The same can be done in the frequency domain but using multiplication instead of convolution (Smith, 2003, Chp. 8), see figs. 4.13(a) and 4.13(b). Now we know x and we know $x*h$, or equivalently X and XH . This makes it trivial to calculate h or H by deconvolution or division respectively. Thus to emit a signal x where the relation between each frequency is preserved, divide X by H thus in effect emitting X/H resulting in X on the output. Figure 4.13(c) shows a block diagram that illustrates how the parameters are related.

Experimental method

The used equipment was an Avisoft Scanspeak speaker and a G.R.A.S. 40BF calibrated microphone. The speaker was connected to a MCLURS system whilst the microphone was connected to an Avisoft amplifier and a laptop. These were then triggered together. In spite of this there was still some jitter between the recordings. The speaker and microphone were placed 1 meter apart facing each other. We then emitted multiple same level sweeps, where the sweep covers the frequencies of interest (downsweep

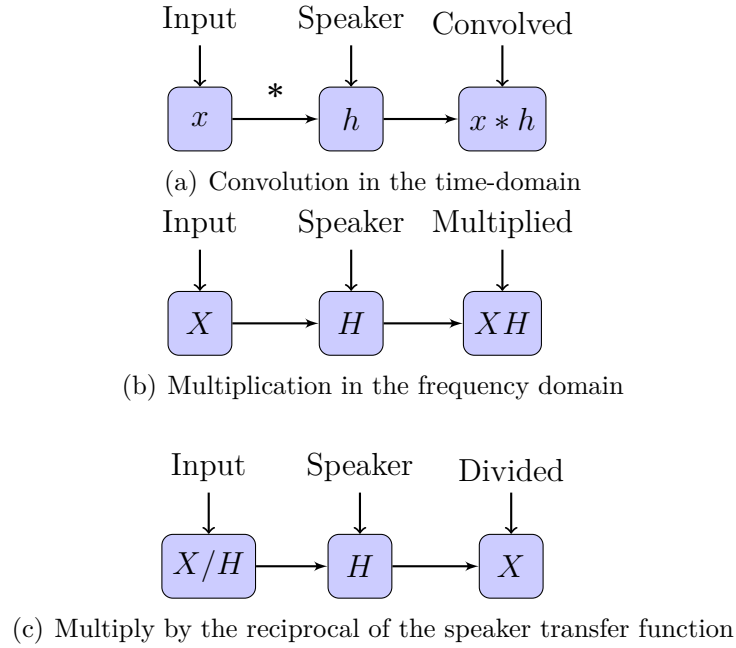


Figure 4.13: Emitted signal convolved with the transfer function of the speaker and the adapted signal.

from 110 kHz to 10 kHz). Aligning and averaging these recordings produces a fairly accurate image of the chain's effect on the signal.

In practical experiments it is always useful to repeat and average to reduce the effects of noise and jitter, this was especially needed here as the synchronization between emitter and receiver was not perfect (usually less than $20 \mu s$). The reason for the jitter is that the emission program was not implemented correctly. Each emission was repeated thirty times and averaged. The recovery process involved aligning before averaging. The process is outlined in Algorithm 4.1. See Source Listing D.1 on page 150 for implementation details of this procedure.

When the speaker's transfer function has been recovered, calibrated emission can be achieved by deconvolving the input signal with the transfer function. In this case the algorithm is straightforward, see Algorithm 4.2. See Source Listing D.4 on page 152 for implementation details of this pro-

Require: Thirty or more near identical recordings

```

1:  $sum \leftarrow 0$ 
2:  $S \leftarrow \text{LISTOF SIGNALFILES}(pattern)$ 
3:  $r \leftarrow S[1]$ 
4:
5: for  $s \leftarrow S[2]$  to  $S[\text{end}]$  do
6:    $offset \leftarrow \text{CROSSCORRELATE}(r, s)$ 
7:    $s \leftarrow \text{SHIFT}(s, offset)$ 
8:    $sum \leftarrow sum + s$ 
9:
10: end for
11:
12:  $avg \leftarrow sum / \text{LENGTH}(S)$ 

```

Algorithm 4.1: Align and average algorithm. See Source Listing D.1 on page 150 for implementation details of this procedure.

```

1: function APPLY_TF( $s, tf$ )
2:    $S \leftarrow \text{FFT}(s)$ 
3:    $TF \leftarrow \text{FFT}(tf, \text{LENGTH}(S))$ 
4:    $ST \leftarrow S/TF$ 
5:   return RE(IFFT( $ST$ ))
6: end function

```

Algorithm 4.2: Apply transfer function to a signal. See Source Listing D.4 on page 152 for implementation details of this procedure..

cedure.

Results

Equal energy logarithmic sweeps (Figure 4.14) were emitted (N=30) at 1 meter distance into a decapped G.R.A.S. 40BF 1/4 inch calibrated microphone, aligning and averaging resulted in Figure 4.15.

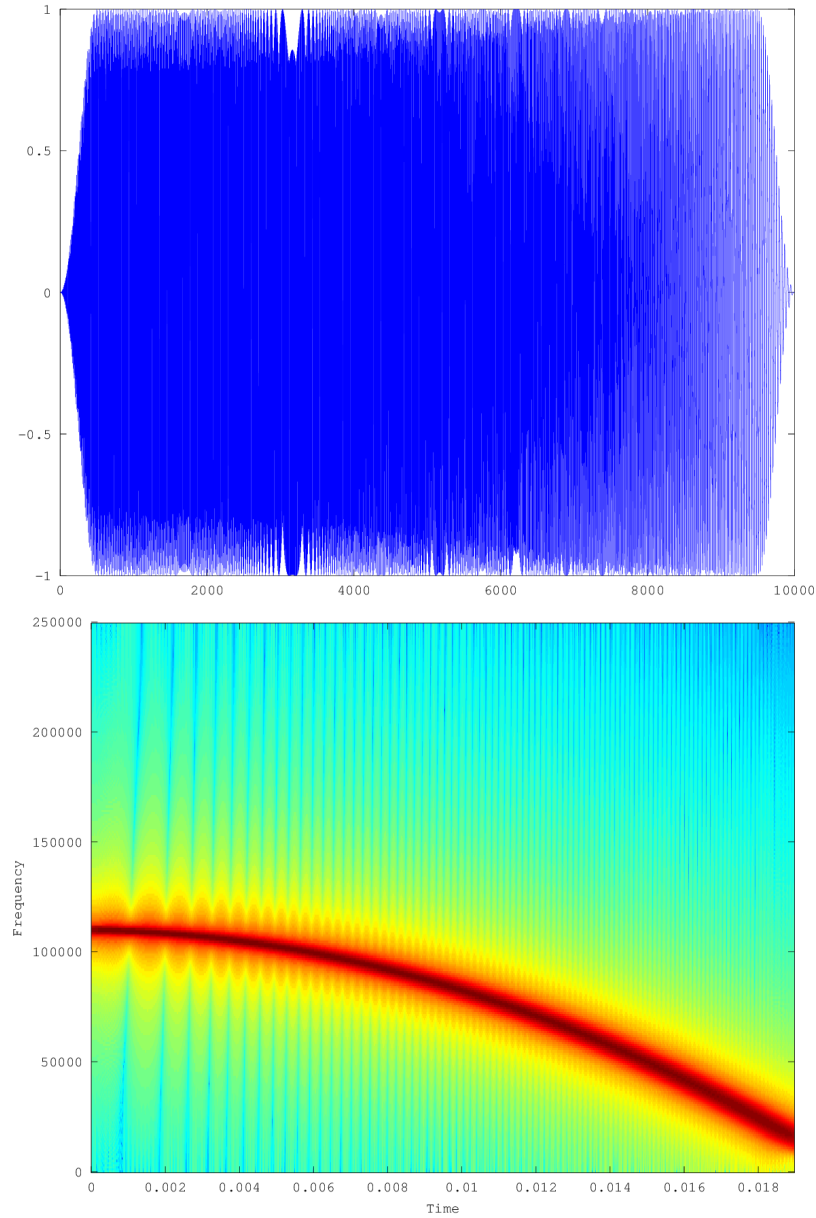


Figure 4.14: Before emission – x . The signal to be emitted, it is a logarithmic down-sweep from 110-10 kHz. Spectrograms were generated with 512 sample Hanning windows and 511 sample overlap. Notice that the signal fades in and out to reduce onset and offset distortions.

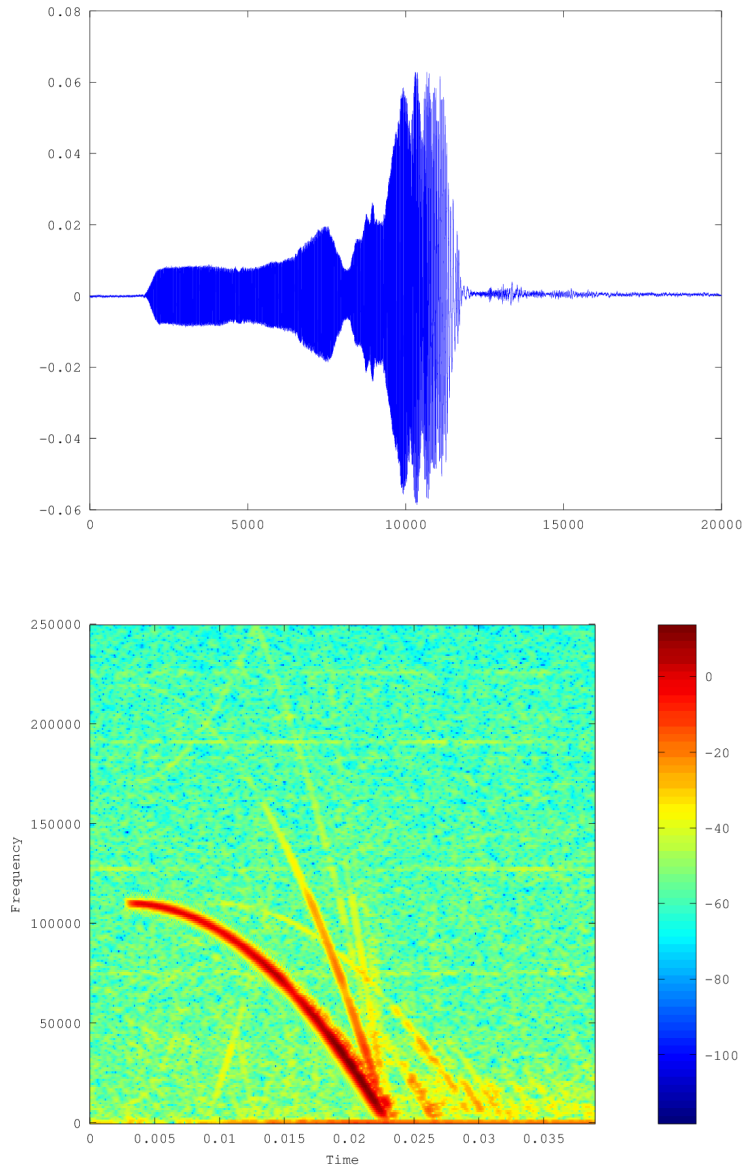


Figure 4.15: After emission – x convolved with h ($x * h$). Spectrograms were generated with 512 sample Hanning windows and 511 sample overlap. The filtering effects of the analog chain are clear both in the amplitude plot and the spectrogram.

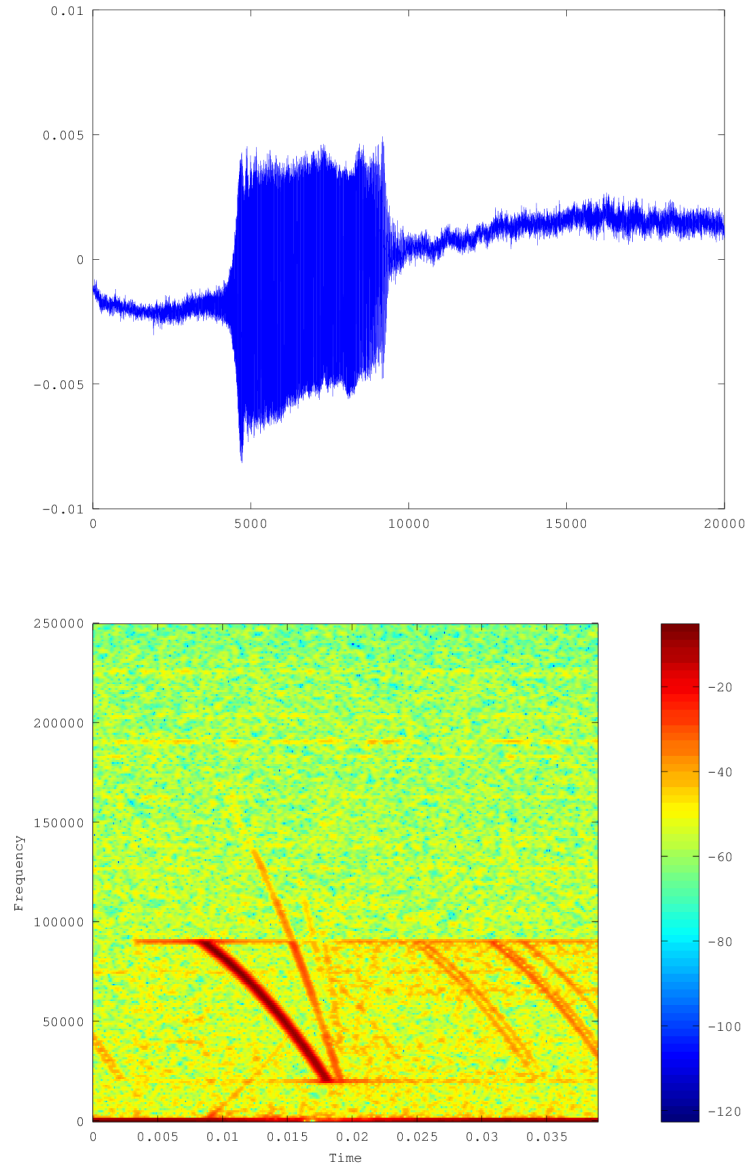


Figure 4.16: Emission of x deconvolved with h limited to 90 to 20 kHz because the speaker was too weak at higher frequencies. Spectrograms were generated with 512 sample Hanning windows and 511 sample overlap. Notice that the emission contains equal energy at frequencies, thus a calibrated emission has succeeded.

To verify that the correct transfer function was actually recovered, we emitted x deconvolved by h (again $N=30$). Note that we reduced the sweep to go from 90 kHz to 20 kHz because otherwise the emitted signal would be too weak. The results can be seen in Figure 4.16. If the low frequency noise is ignored the emitted signal is very similar to the original signal, i.e. we have equal energy in all emitted frequencies. This shows that we have succeeded in emitting x in a calibrated manner. The signal is much attenuated because the highest emission amplitude is limited by where the speaker is most inefficient within the sweep range. The measured transfer function for this speaker can be seen in Figure 7.5(b) on page 108.

Summary

Although this has been done many times before by many other researchers, we were pleased to confirm that our method was correct. The system is now ready for calibrated emission experiments. Given a second chance we would use the built-in emit/receive feature of the MCLURS system, as this would make this task easier.

4.7 Logging other data

Each module includes a standard retail Single Board Computer, which will accept input from most available retail hardware. As a demonstration of this, we sampled data from a weather station, the wireless Fine Offset WH1090 (Fine Offset Electronics, URL provided, accessed: Apr 2012), using the open source software **wwsr** (Pendec et al., URL provided, accessed: Apr 2012). The recorded data includes time-stamped weather measurements such as wind speed, wind direction, precipitation, air pressure, temperature, and relative humidity, and is time-synchronized with the acoustic recordings. Recording local weather conditions together with acoustical data enables us to correlate bat activity with climate. In addition it provides the basic parameters, temperature, pressure and relative humidity, needed to calculate the local speed of sound. The system also allows easy integration

of other types of data, for example light intensity, cloud coverage, wind turbine blade speed, etc.

As an illustration of what kind of data is forthcoming from such a system please see Figure 4.17. The figure shows a four day sequence in June 2012 of measurements done every minute. The placement of the weather-station was not ideal so wind-speed, wind-direction, and rain-fall measurements were highly influenced by a nearby structure. These have therefore been omitted from the plots.

4.8 Summary

In most of the experiments where we decided to apply MCLURS, it performed very well. This is an attestation of the applicability of the design and its implementation. Many of the experiments have not been completed yet, so only preliminary results are available. Hopefully these will be followed through either by me or others.

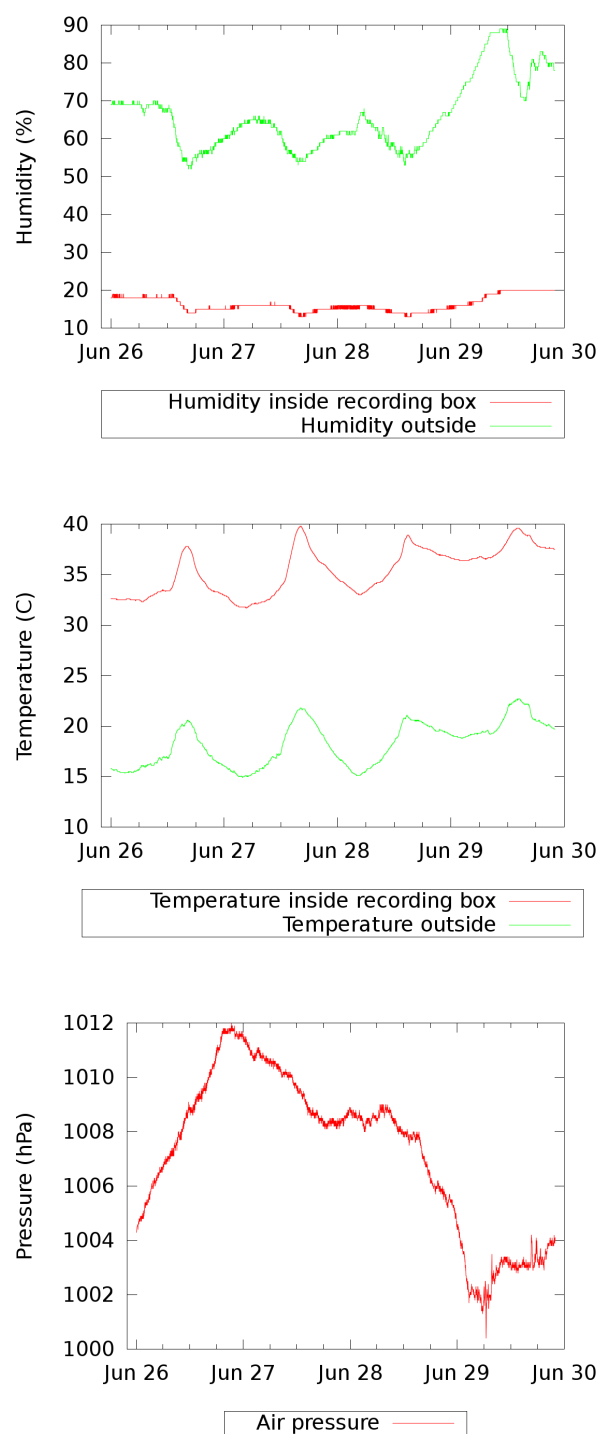


Figure 4.17: Automated weather measurements from a Fine Offsets WH1090 attached to MCLURS..

Long-term deployment – Technological issues

*The earth we abuse and the living
things we kill will, in the end, take
their revenge; for in exploiting their
presence we are diminishing our
future.*

— Marya Mannes

This chapter describes methods used during a three month long-term survey of bat activity on the campus grounds at the University of Southern Denmark. The methods describe how to extract and classify bat calls from huge data volumes. First, the recording method is covered, then parameter calculations are described. This is followed by how to use them to make a classifier, which in turn is able to recognize bat calls circa 70% of the time.

5.1 Experimental method

Four miniature microphones, type FG23329–PO7 (Knowles Acoustics, Itasca, Illinois, USA), were mounted on a tetrahedral structure with 30 cm sides, which was positioned on the roof of a local garage building. The tetrahedron can be seen in Figure 5.1(a). The omnidirectional microphones were directed downwards and protected by a water-guide (Figure 5.1(b)). For the technical details of the recording system please refer to Chapter 3. The recording system components fit comfortably in a sturdy IP65-compliant plastic box (Fibox type EK-W). Power to the system was supplied directly from the grid to avoid changing batteries. The system was configured to record four synchronized channels with a 500 kHz sampling rate per channel.

We configured the system to wake up and start recording at 8:00 PM. This was set through the Real Time Clock on the computer motherboard. The system automatically switched itself off at 6:00 AM through the op-

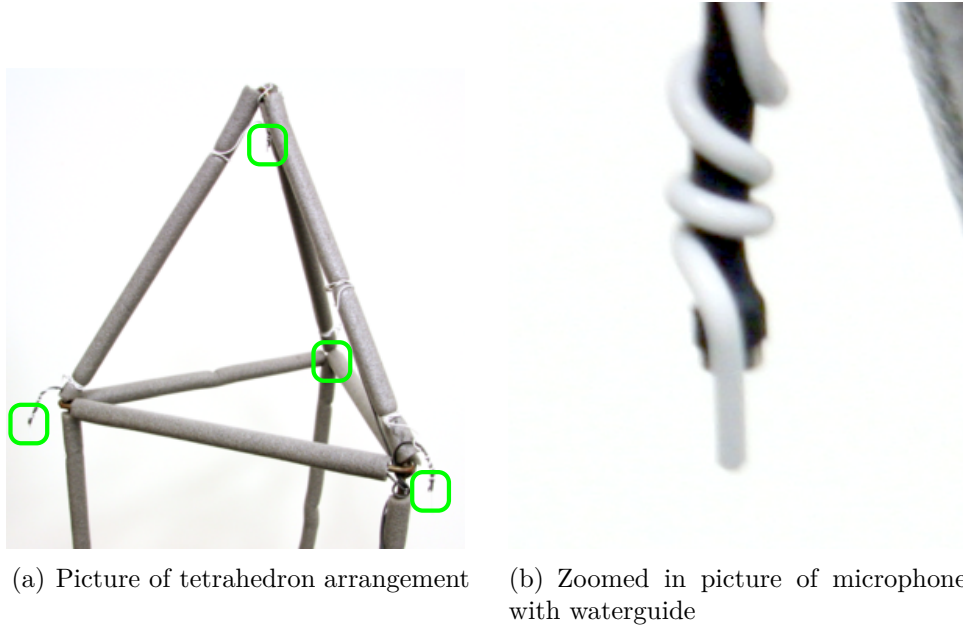


Figure 5.1: Left: The microphone assembly: a tetrahedron mounted with four Knowles microphones with water-guides. The microphones are marked by green rounded rectangles. **Right:** A close-up of the topmost microphone, showing its water-guide (the wire that projects below the bottom of the microphone).

erating system scheduler. Upon return from power-failure the previous wake-up/shutdown cycle is recovered.

The system was configured with a repeating trigger and a two-second pre-trigger buffer. The trigger was activated when the average energy of the signal exceeded a threshold for more than half a millisecond. The trigger is deactivated again when the average energy has been below threshold for more than two seconds. The recording system saves the pre-trigger buffer and up to and including the final two seconds of sub-threshold samples to permanent storage each time the trigger is activated. Each such recording generates one data file on the recording system computer.

5.2 Recording the raw data stream

In order to only record energetic parts of the incoming stream, signal detection is needed. Using the Unix pipeline paradigm (McIlroy, Pinson, and Tague, 1978) we funnel the data from the A/D process into a signal detection routine which in turn extracts only high energy parts of what the microphones recorded. The following explains pipelines in some detail to give an idea of their versatility. For more examples see appendix C.1 on page 141. This is followed by how the actual recording was performed for this survey.

Pipelines and replaceable components

Using a Unix-inspired operating system on the modules' local computers makes available a large set of advanced high-level tools and paradigms. The operating system pipes (McIlroy, Pinson, and Tague, 1978) abstract data handover between programs and readily handle infinite data streams in a sensible way. Using such a paradigm it becomes possible to separate acquisition, analysis, multiplexing, de-multiplexing, and forwarding of the streams. This allows us to use (or if they do not exist, write) targeted components that solve each problem in turn, dividing the complete problem into subproblems that are easier to solve. See Figure 5.2 for an illustration of the concepts.

Doing the actual recording

For this survey the data was collected from the A/D board by a program called `adac` (Source Listing A.1 on page 113). `adac` deals with the full datastream and sends it on to another program, `signal_detect` (Source Listing A.2 on page 119) which watches for high energy buffers. Each buffer is 1 ms long. When the average energy in a buffer exceeds the noise floor by circa 12 dB, `signal_detect` starts recording to a new file. The file will contain the previous 2 seconds before the high energy buffer was

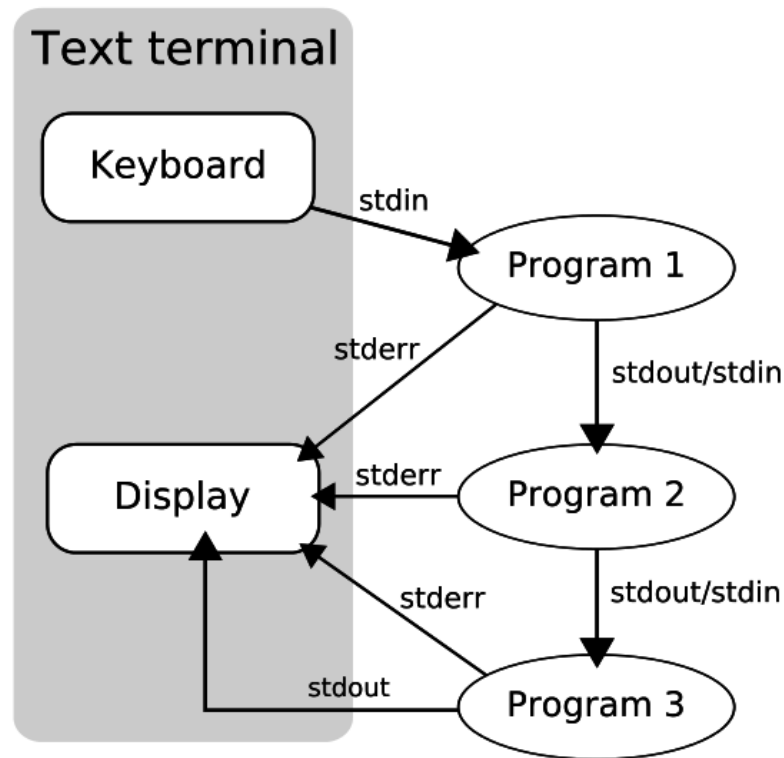


Figure 5.2: The relation between programs linked with Unix pipes.

encountered and everything up and including 2 seconds after the last high-energy buffer has occurred.

The recording was started at 20:00 by running the two programs connected by a pipe (Source Listing 5.1. Note that the “|” symbol is used to connect the two programs:

```
adac | signal_detect
```

Source Code 5.1: Start A/D conversion and signal-detection. Note that the “|” symbol is used to connect the two programs.

5.3 Extraction and classification

Upon the completion of this survey, the system had recorded 236 GiB of data. The volume of data is the key technical problem with our approach: such an amount of data requires a systematic and automatic approach to analyze. Our objective was to discover and classify as many bat calls as possible in the dataset, with a high detection success, and a low false positive rate.

Event marking

The goal of automatic processing is to identify interesting segments of the recorded signals, which we shall call *events*, within which bat calls are likely to be found. Note that in general not every bat call will result in an event, since not every call will pass the event-energy threshold we set. The idea is to identify likely candidates first, then search for quiet calls later once good examples have been identified. The likely candidates can then be used to extract the weaker signals, for example with cross-correlation.

Event detection and marking

In the first step of event detection, the data was corrected for DC shift, then squared and convolved with a normalized Gaussian moving average filter of length $100 \mu s$ ($\mathbf{G}$) in order to smooth the power series and construct a measure of the local intensity of the signal at each time. This will not remove interesting events as those are all much longer than the filter length.

The above can be expressed mathematically like this

$$\mathbf{e}(\mathbf{t}) = \text{diag}[(\mathbf{x} - x_{avg}\mathbf{1})(\mathbf{x} - x_{avg}\mathbf{1})] * \mathbf{G} \quad (5.1)$$

where $x_{avg}\mathbf{1}$ is the vectorized average for the channel and $*$ represents convolution.

Events were detected based on energy thresholding. Those samples of the energy signal that exceed a chosen threshold mark the presence of candidate events. Events are then defined as starting at a mark preceded by a

certain interval (500 μs is used here) without marks. The end of an event is defined by a mark followed by a certain interval without marks. In this way, a cluster of high energy values, each generating a mark, are grouped into a single event. Furthermore, each event has a front and back “porch” (region) where the signal energy is below the event threshold.

The final form of an event is an interval of the recorded data that begins 500 μs before the start mark and ends 500 μs after the end mark. Further analysis of these events will classify them as comprising noise alone, or one or more bat calls. Events shorter than 1100 μs (including the front and back porches) were ignored.

An empirically estimated energy threshold was determined by selecting those portions of the recordings where the signal power was in the top $\Theta\%$. This was found by sorting the samples in a recording into ascending order and setting the sample at the $1 - \Theta\%$ percentile in the order as the threshold. This is closely related to finding the cumulative distribution function. Figure 5.3 shows a typical sorted cumulative density of a recording. Formally the threshold was chosen to be:

$$\text{th} = \text{cumdensity}^{-1}(1 - \Theta) \quad (5.2)$$

We want the inverse because we are mapping a specific density Θ to an energy sample value. In our analyses we used $\Theta = 0.1$, i.e. 10%.

Event classification

Given the event extraction method, the problem becomes one of classification. We define a set of quantitative features that capture relevant aspects of the properties of bat call spectrograms and use the vectors of their values as inputs to a suitable classification algorithm. There are many ways to go about this; preliminary research suggested that the well-established and well-understood Support Vector Machine (SVM) (Boser, Guyon, and Vapnik, 1992) would be suitable for separating events into bat and non-bat classes. We used the LIBSVM (Chang and Lin, 2011) implementation of the method, which is a library that supports training and applying sup-

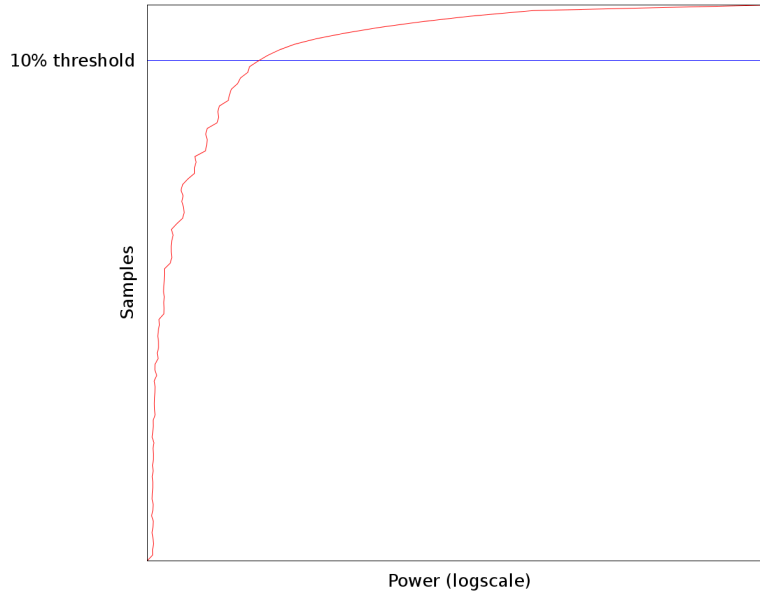


Figure 5.3: A typical sorted cumulative density plot. The x-axis is sample value; the y-axis runs from zero to one and shows for given abscissa the proportion of samples in the set exceeded by that value..

port vector machine models to a dataset. For the two-class case that we are considering, the SVM model can be envisioned as a hypersurface that separates a parameter space into two hypervolumes, one volume for each class.

The goal of the classifier is to identify, with as high an accuracy as possible, events that contain bat calls; in case of errors, we prefer false negatives over false positives. The best classifier obtained from training and cross-validation is used to classify all available events into candidate bat-calls, meriting attention from a biologist, and non-bat events (which may in fact contain bat calls, of course). From that point onward, in the current system design, data analysis requires human judgment.

Event quality measures

The parameter space in our case is constituted by the features we choose to describe events: the event quality measures. To classify the events we

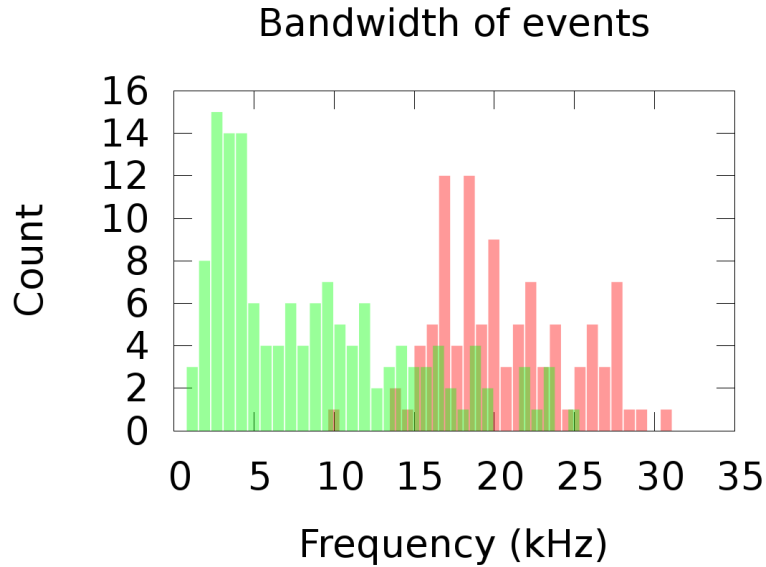


Figure 5.4: This shows how bandwidth is divided amongst bat (red) and non-bat (green) events. The less overlap the better separation the quality measure carries. The red and green bars are offset by half a bar to improve readability.

used a combination of temporal and spectral analyses. Bat calls are characterized by their relatively short duration and stereotypical spectrographic structure (Schnitzler and Kalko, 2001). Consequently, features such as duration, energy, bandwidth and so forth might be expected to provide good discriminants. After some experimentation, we found the measures described below to be good classifier features. Three of them are plotted in Figures 5.4 to 5.6 and give a feel for how well they separate the data. The quality measures were combined in a composite bat-call classifier described later. First, we formally define the quality measures used. See Source Listing A.3 on page 124 for the implementation used to calculate the quality measures.

Duration The duration of event j was calculated in milliseconds as

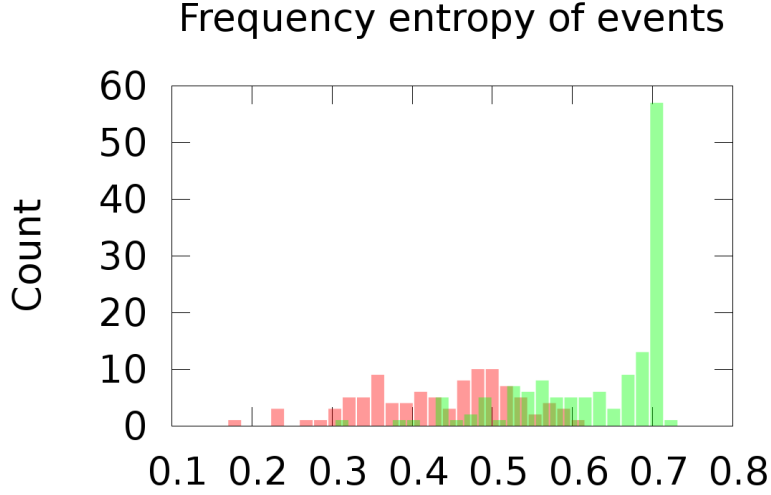


Figure 5.5: This shows how frequency entropy is divided amongst bat (red) and non-bat (green) events. The less overlap the better separation the quality measure carries. The red and green bars are offset by half a bar to improve readability.

$$d_j = \frac{\text{end}(ev_j) - \text{start}(ev_j)}{sr/1000} \quad (5.3)$$

where sr is the recording sample rate; $\text{start}(\cdot)$ and $\text{end}(\cdot)$ return the indices of the first and last samples belonging to the event supplied as argument.

Average power The average power of event j was calculated as the sum of squares of the samples comprising the event divided by the duration of the event:

$$W_j = \frac{1}{d_j} \sum_{i=\text{start}(ev_j)}^{i=\text{end}(ev_j)} ev_j(i)^2 \quad (5.4)$$

where $ev_j(i)$ is the i^{th} sample of event j .

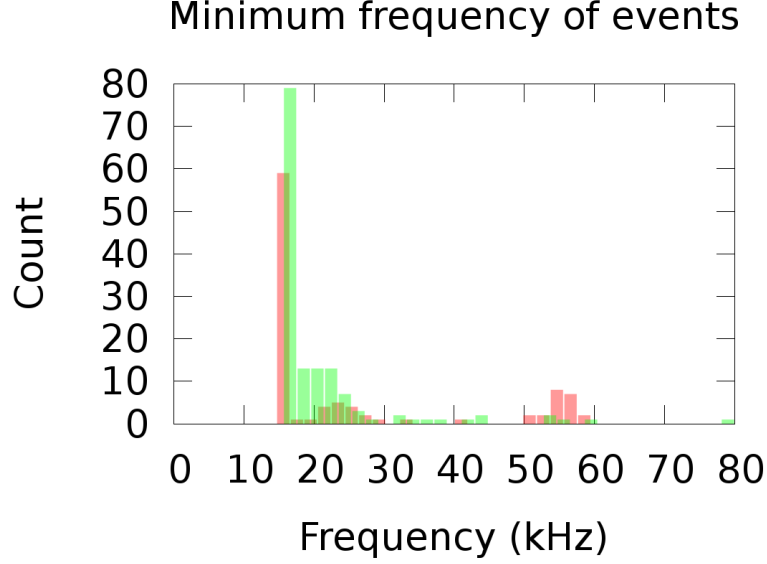


Figure 5.6: This shows how minimum frequency is divided amongst bat (red) and non-bat (green) events. The less overlap the better separation the quality measure carries. The red and green bars are offset by half a bar to improve readability.

Minimum and maximum frequency The most energetic frequency content was determined in a similar manner to the marks in the time domain which delimit events. The FFT of the event is constructed so that each frequency bin is always 50 Hz, which makes events more spectrally comparable. The resulting frequency bins are thresholded to mark the top 10%. Adjacent marked bins are merged into the same cluster if they were less than 300 Hz apart. The minimum $f_j^{\min}$ and maximum $f_j^{\max}$ were determined by the start and end marks of the largest spectral cluster.

Bandwidth Bandwidth follows directly from the minimum and maximum detected frequencies:

$$bw_j = f_j^{\max} - f_j^{\min} \quad (5.5)$$

Spectral entropy In order to detect the bat calls even under noisy conditions, we devised a feature that highlights the much more orderly time-frequency structure of bat calls relative to noise. This enabled us to detect bat calls even when masked by moderate noise in the same bandwidth. Entropy is a generic statistical measure (Shannon, 1948) that is high for unordered data and low for ordered data. As noise tends to have a broad unordered spectrum, entropy is a potentially good measure for separating bats from noise.

The spectral entropy was calculated by first determining the probability distribution of spectral power. This was done by creating 5 kHz frequency bins, calculating the power in each bin and dividing by the total power of the event:

$$P_j(m) = \frac{1}{d_j W_j} |EV_j(m)|^2 \quad (5.6)$$

where m indicates the frequency bin. So $|EV_j(i)|^2$ is the power of event j in the i^{th} frequency bin. The entropy of the event is then calculated as:

$$E_j^{\text{freq}} = - \sum_{m=1}^M P_j(m) \log P_j(m) \quad (5.7)$$

Training the model

Classification works by building a model based on vectors of the quality measures of manually classified events to determine an appropriate separatrix function (Boser, Guyon, and Vapnik, 1992). The events are treated as labeled points in a high-dimensional space, and the classifier training algorithm determines a surface that separates the two clouds of labeled points, assuming this to be possible. The SVM algorithm in particular also maximizes the distance from the separatrix to the members of the two classes (Boser, Guyon, and Vapnik, 1992), producing thereby the “best” separating surface in the sense of allowing maximal clearance between the surface and the class members. The distance to the separatrix is a measure of how “bat-like” an event is.

Building the classifier Building the classifier requires labeled data, i.e. samples of input grouped into each of the classes to be learned. This data was prepared manually. Thus: the events were randomized, the quality measures were calculated for all the events, these and accompanying spectrogram, spectrum and amplitude plots were visually inspected, and events which were clearly identifiable as bat calls that exceeded 20 dB signal-to-noise ratio were marked as bat events. During this process many clearly non-bat events were encountered and marked as such, to provide the training set entries for the non-bat class. In total the training set consisted of 99 bat call events and 141 non-bat events, 240 in total. The feature value vectors and their labels were provided to the LIBSVM training algorithm using `svm-train` which constructed the SVM classifier.

The actual model used by the SVM is based on a Radial Basis Function (RBF) kernel. The RBF is a real-valued function whose value depends only on the distance between a feature vector and a specified center for the function. It is the most common kernel function used with SVMs (Chang and Lin, 2011). The RBF layer in LIBSVM has two parameters, *gamma* and *cost*. The optimal value for these is found by running the *svm-grid* program and visually adjusting *gamma* and *cost* to optimize correct separation into correct classes.

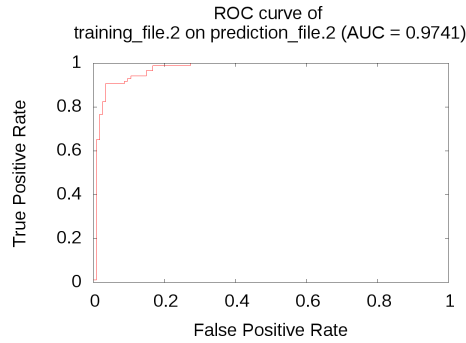
When assessing the accuracy of the SVM classifier, N -fold cross-validation is used. The manually classified training events are divided into N blocks. N SVM classifiers are constructed by training using $N - 1$ blocks of the data, while the final block is used for testing the performance of the model constructed on the other $N - 1$. Each block is therefore used as testing set, while the rest is used as the basis for the model, giving N independent estimates of the classifier's accuracy and an indication of the variability of accuracy as a function of training data.

Parameter scaling As the individual feature value ranges are quite disparate when compared to each other, feature rescaling is often recommended to improve the classification model's accuracy. Three classifiers were built using three scaling approaches: no scaling, relative linear scaling, and cus-

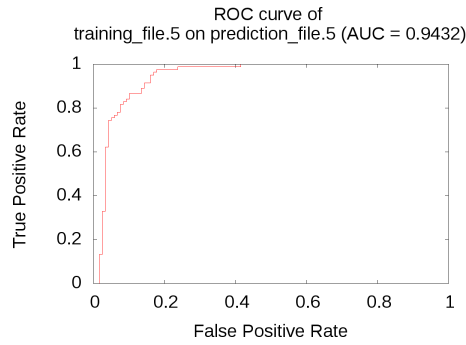
tomized absolute scaling. In the first case, the raw feature values were used unaltered. Relative linear scaling finds the global maximum and minimum values of each feature in the entire dataset, and uses that to scale the feature component in the SVM input parameter vector to the range -1 to 1 . This scaling depends on the data set available for training (hence “relative”) since it will differ for different data sets. A model trained on one set will thus not be applicable to another dataset if the minimum and/or maximum differ from the original.

The third scaling type rescales each feature based on our *a priori* knowledge of the expected range of values for that feature. It is independent of the actual feature values in the training data set, hence “absolute”. In this approach, there are three cases. First, if the feature has a fixed minimum and maximum (for instance, spectral entropy lies between 1 and $\log M$) then it is scaled linearly to ± 1 . If the feature has a fixed lower bound but a soft upper bound — that is, the *a priori* minimum value is never crossed while the maximum might be exceeded (though that is unlikely) — the feature value is rescaled using a logistic squashing function (sigmoid-like) to map the minimum to -1 and the soft maximum to 0.9 . Outliers are compressed by this into the $(0.9, 1]$ range. If the feature’s maximum and minimum are both soft, it is rescaled using the same squashing function to map the minimum and maximum to ± 0.9 respectively.

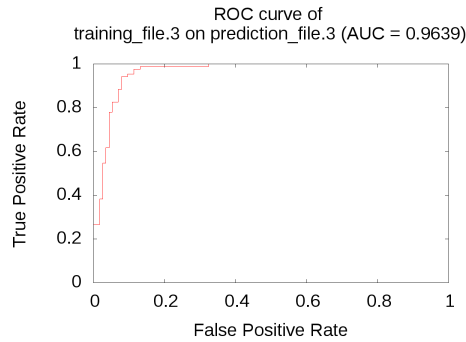
The performance of the three scaling approaches can be compared using the receiver operating curve plots for the classifiers built from those feature vectors. The greater the area under the curve, the better is the performance of the classifier. The ROC curves for unscaled, relative linear scaling, and customized absolute scaling are shown in Figure 5.7. These ROC curves were calculated from the labeled training set, and suggest that the raw feature values are such that feature rescaling provides no advantage for this problem.



(a) Unscaled training set



(b) Relative linear scaled training set



(c) Custom absolute scaled training set

Figure 5.7: ROC curves comparing the accuracy of SVM models based on unscaled, relative linear scaled and custom absolute scaled SVM training set. The suffixed numbers in the title of the plots indicates which of the 6 models was used. We always chose the one with highest accuracy. (a) Unscaled training set. (b) Relative linear scaled training set. (c) Custom absolute scaled training set (see text).

5.4 Summary

The methods described herein have successfully reduced 236 GiB of raw data to 8,413,249 events of which the automatic method identified 115,456 events as bats. Analysis of these events and their distribution with respect to time and date is the subject of Chapter 6.

Ecological study of local bat population

*The creation of a thousand forests is
in one acorn.*

— Ralph Waldo Emerson

6.1 Introduction

Bats are protected and some species are endangered; loss of foraging grounds and roosting places pose a serious threat to them (Cryan et al., 2012). Major new building projects not only disturb local bats during construction, but may also add to the decline in suitable habitat. The University of Southern Denmark, where we are situated, is undergoing major re-construction with plans for tripling the built-up area and introducing substantial changes to the surroundings. We are interested to know what effect such changes may have on our local bat populations.

In the above and many other cases bat surveillance is an important field. This survey only considers acoustical measurements, and as such, includes the biases that this entails, see introduction in Section 4.2 for a more in-depth discussion of the issues with using only one surveillance paradigm.

The MCLURS system allows us to record bat activity over months, and to correlate bat behavior over time to several parameters including change of season, day/night, sunset/sunrise, lunar luminance, and weather data. One can also study the distribution of social versus echolocating calls, and in principle identify sequences which predict particular behaviors (such as foraging) and see how these are distributed over time.

The key technical problem raised by our approach is what to do with the potentially huge volume of data collected. See Chapter 5 for explanation of the solutions we devised to handle this problem.

6.2 Study site

Our goal was both to assess bat activity and to test the novel prototype recording array, so convenience of access was important. Therefore we placed the system at a site on the Odense campus of the University of Southern Denmark ($55^{\circ}21'59.63''\text{N}$, $10^{\circ}25'54.10''\text{E}$) with low anthropogenic activity, high bat activity, power grid access, and convenient physical access. The campus is located outside the city of Odense in Denmark in an open area with patches of forest, large open fields and small streams. The array was placed on the roof of a garage building, about 5 meters off the ground, close to a small, old forest where we had often observed bats of various species flying along the tree-line and inside the forest.

6.3 General notes and observations

The system was first set up for automatic data capture from 20:00 to 6:00 on June 23rd 2011. The plot in Figure 6.1 shows the distribution of recorded data volume over time. The main panel shows data volume plotted with respect to day and time-of-night counted using 2 minute time bins; the top panel shows the total data volume for that day in bytes. The plot in Figure 6.2 shows a histogram of the sizes of data capture files; notice that most files are 4 seconds or equivalently 4 MiB in length (pre-trigger buffer plus post-trigger silence), but there is a long tail of much bigger files shown in more detail in the inset plot. Overall, the system captured 236 GiB of raw data in a total of 11059 files of varying sizes.

Unfortunately the system was accidentally disconnected from the grid on June 27th 2011. This happened again on July 6th 2011 and this time the system remained off-line for almost a month. This is reflected in Figure 6.1 by the many empty days during July. These disconnections always happened during working hours, i.e. between 8:00 and 17:00, so the system was never interrupted during a recording run. The result is that we have sporadic recording days in June and July. When the system was restarted on August 1st 2011 it remained operational until the survey period was

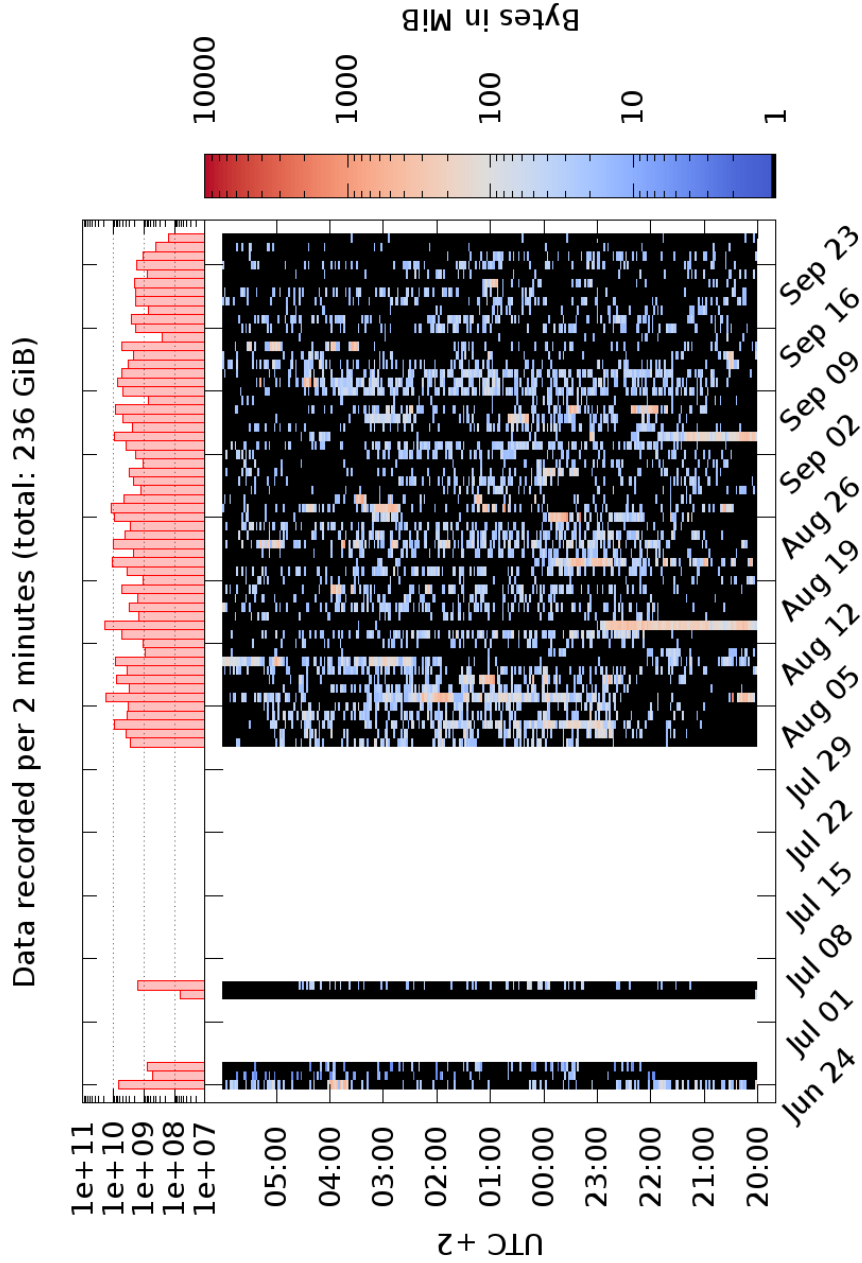


Figure 6.1: This plot shows the temporal distribution of recorded data. The color indicates number of MiB per 2 minutes. The histogram along the top shows daily totals in number of bytes.

over on September 27th 2011. As we are interested in patterns over time, we have chosen to focus on the last period only, i.e. August 1st through September 27th, in the results and analysis below.

Although we recorded four channels in this survey, this survey only considers data from channel one as that is sufficient for our analyses here. There are improvement possibilities and other analyses available with multi-channel recordings, which are discussed in more detail in sections 6.6 and 7.2.

6.4 Event marking

After applying the threshold and event marking methods described in Section 5.3 the data can be viewed as an event distribution with a total of 11,777,675 events. Notice that this number is inflated by the fact that a number of test signals were also included in this pre-analysis. These automatically-identified regions of the recordings are where we hope to find bat calls. Figure 6.3 shows how the data separates into number of events per second (top panel) and the percentage of the whole recording that an event covers (bottom panel). The top panel shows that most of our recordings contain fewer than 500 events, while the bottom panel shows that in most files the events cover less than 15 % of the samples.

A plot of the event data analogous to the temporal distribution of recordings is shown in Figure 6.4. The color of each block represents the number of events occurring during each 2 minute bin. Note that this overview map plots all events, bat and non-bat alike: at this stage, we have not classified the events. When narrowing down to only look at events found during August through September the number of events reduces to a total of 8,413,249. There are clear high incidence areas both in the recordings plot (Figure 6.1) and the event plot (Figure 6.4) — for instance, early evening on August 14th — and there are also areas which have few events, such as the later night and early morning of that same session. The lines plotted on the figure are the times for nautical sunset and sunrise (gold colored) and end and beginning of nautical twilight (goldenrod colored) for the given day. The distribution of events seems to peak within the inner sunset-sunrise

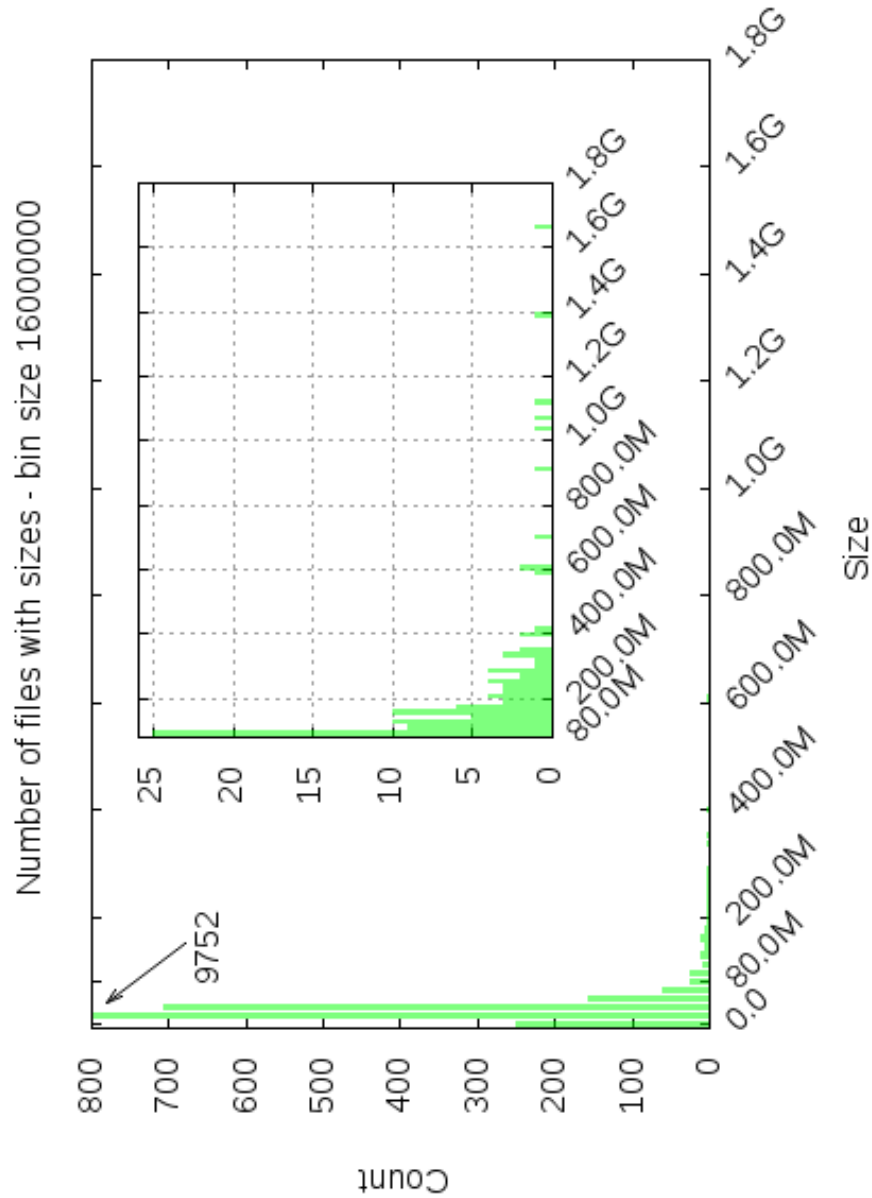
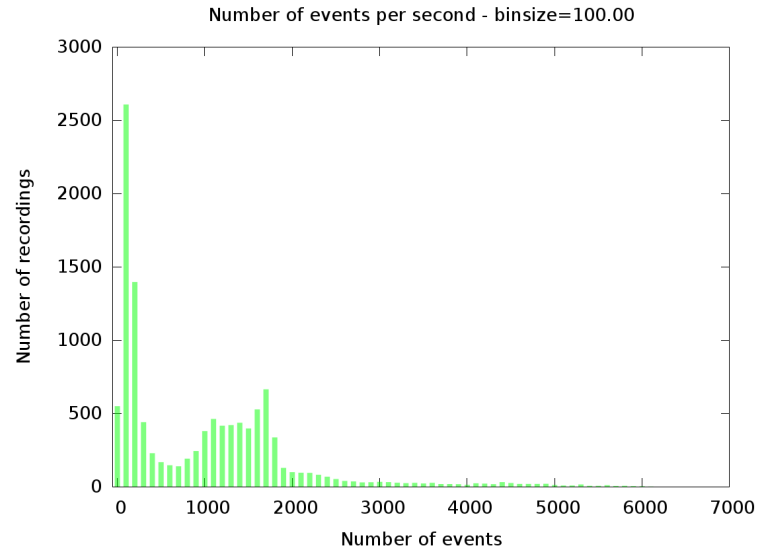
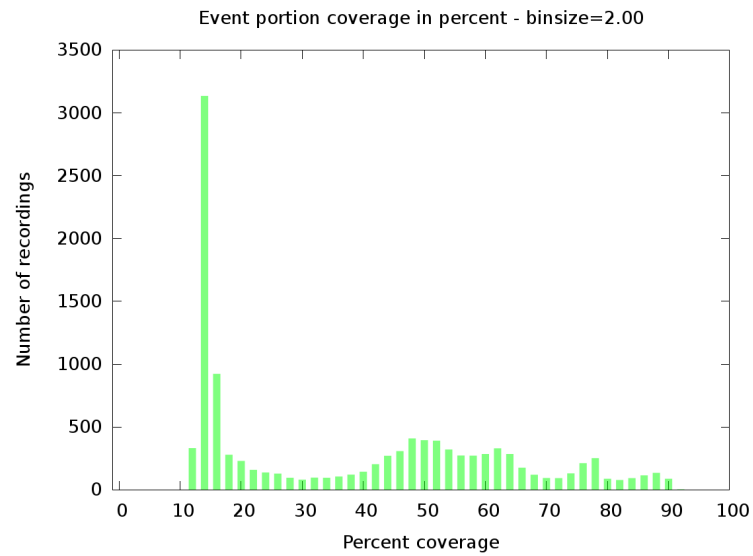


Figure 6.2: This figure shows a histogram of recording file sizes. It shows that most of the recordings were the size of two post-trigger buffers, 4 seconds or roughly 4 MiB. There is a long “tail” on the histogram of a few long recordings. This part is zoomed in by the inset plot, with the size range limited to 80 MiB to 1.8 GiB.



(a) Number of events per file per second



(b) Percent of samples covered by events

Figure 6.3: Number of events per recording per second and recording event ratio. (a) Shows a histogram of how many events were detected per recording per second. (b) shows a histogram of how many samples events cover of each recording file.

lines, apart from bright areas such as during August 10th morning, 14th evening and September 4th evening.

Applying the SVM classifier (Section 5.3) to the detected events results in the plot in Figure 6.5, a total of 115,456 events. The remaining events are all candidate bat events, their number is considerably smaller, and many of the bright areas of Figure 6.4 have disappeared. Inspection of the events in the previously-bright areas reveals that the majority of them are noise recordings, and correlation with weather data for the region suggests that the noise is caused by local rain. The SVM classifier rejects a large proportion of the rain-noise events, though given their density and the accuracy of the classifier it is unreasonable to expect it to eliminate all of them. The non-locally measured precipitation during this survey can be seen in Figure 6.6 (circa 2.5 km away) and Figure 6.7 (circa 5 km away). It is quite clear that there is strong correlation between high incidence periods of events and rain.

There are several interesting features to notice in these distribution plots. Much of the noise has been removed, but regions of relatively high activity remain. Sunset and sunrise appear to have a substantial impact on activity since the candidate bat events tend to occur between the sunset and sunrise boundaries. Throughout the recording period there appear to be both high activity days and low activity days. The start and end of the recording time window could be expanded toward the end of the period (from around September 1st) to better capture the longer nights in late summer and fall.

6.5 Event classification

The events that are plotted in Figure 6.5 are those that the SVM classifier classed as candidate bat calls: as yet we do not know what proportion of them really are bat calls. To address this question, we inspected a number of events by looking at their spectrograms, signal-to-noise ratio compared to the rest of the recording, and duration. Because of time restrictions we chose to look at 7 different days and only every n^{th} event in each day so as

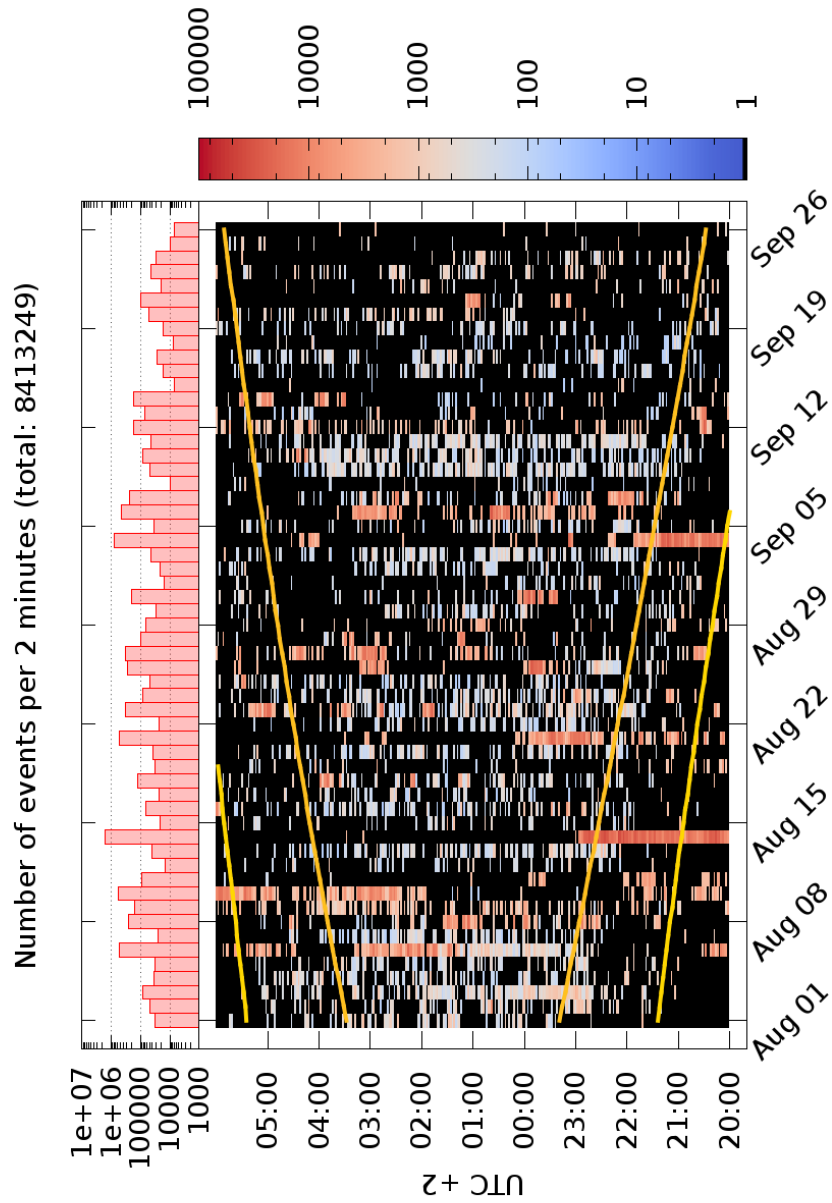


Figure 6.4: This plot shows the temporal distribution of automatically detected events. The color indicates the number of events per 2 minutes. The histogram along the top axis shows daily totals. The gold colored lines indicate sunset/sunrise and the goldenrod colored lines indicate beginning and end of nautical twilight.

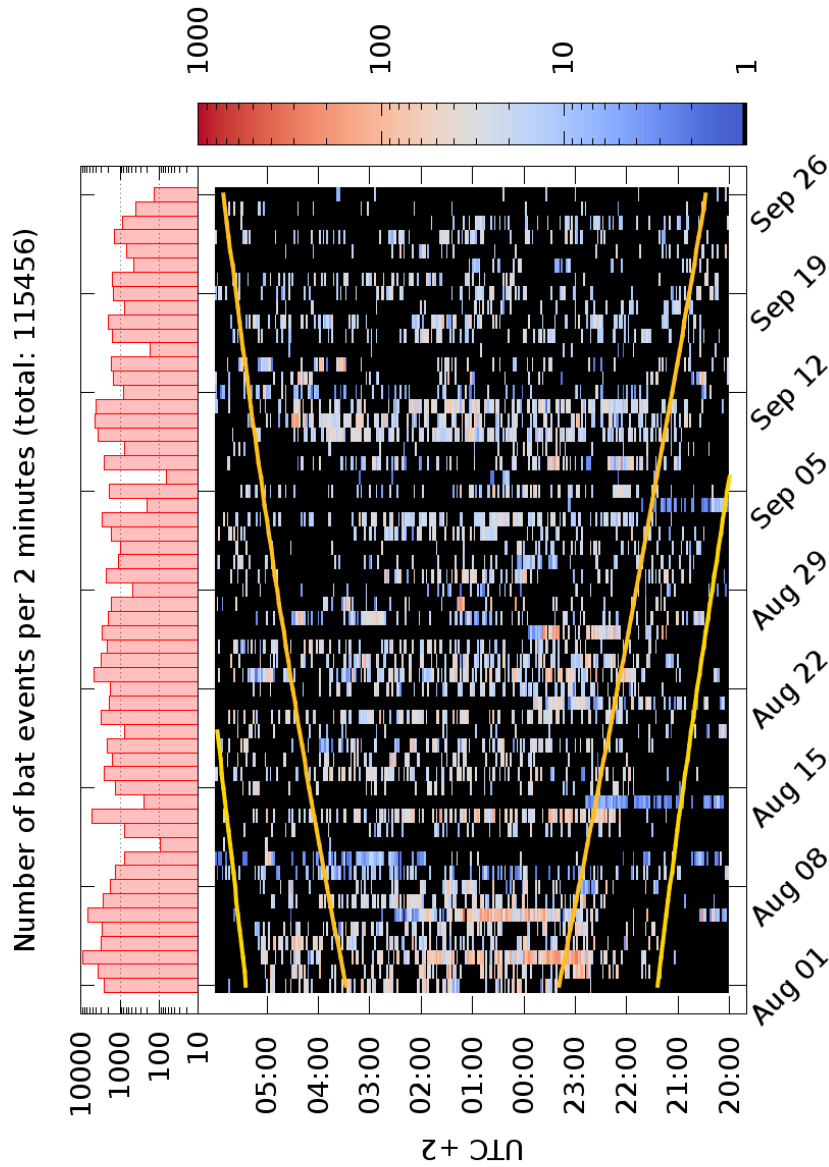


Figure 6.5: This plot shows the temporal distribution of events that are automatically classified as bats. The color indicates the number of events per 2 minutes. The histogram along the top axis shows daily totals. The gold colored lines indicate sunset/sunrise and the goldenrod colored lines indicate beginning and end of nautical twilight. Note that the color scales in this plot and the event plot (Figure 6.4) are different by two orders of magnitude.

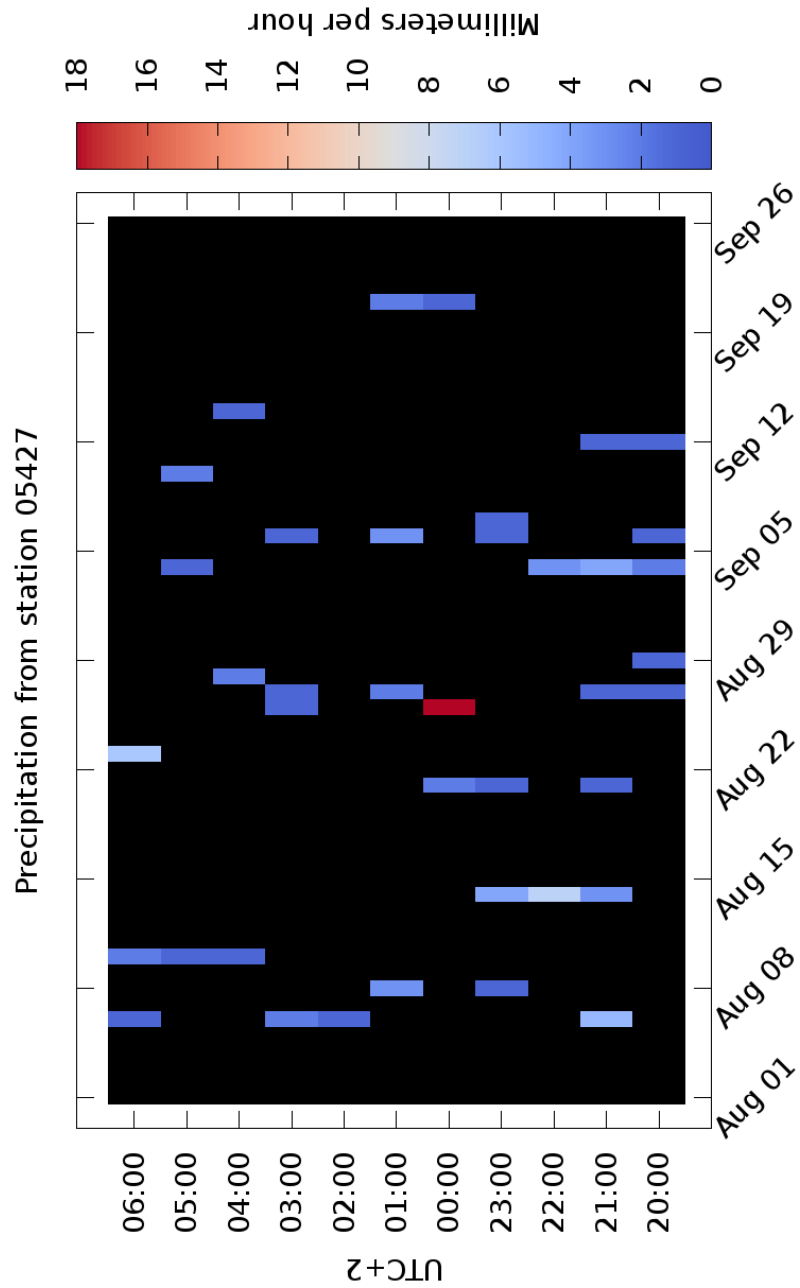


Figure 6.6: Precipitation measured 2.5 km away from the survey site.

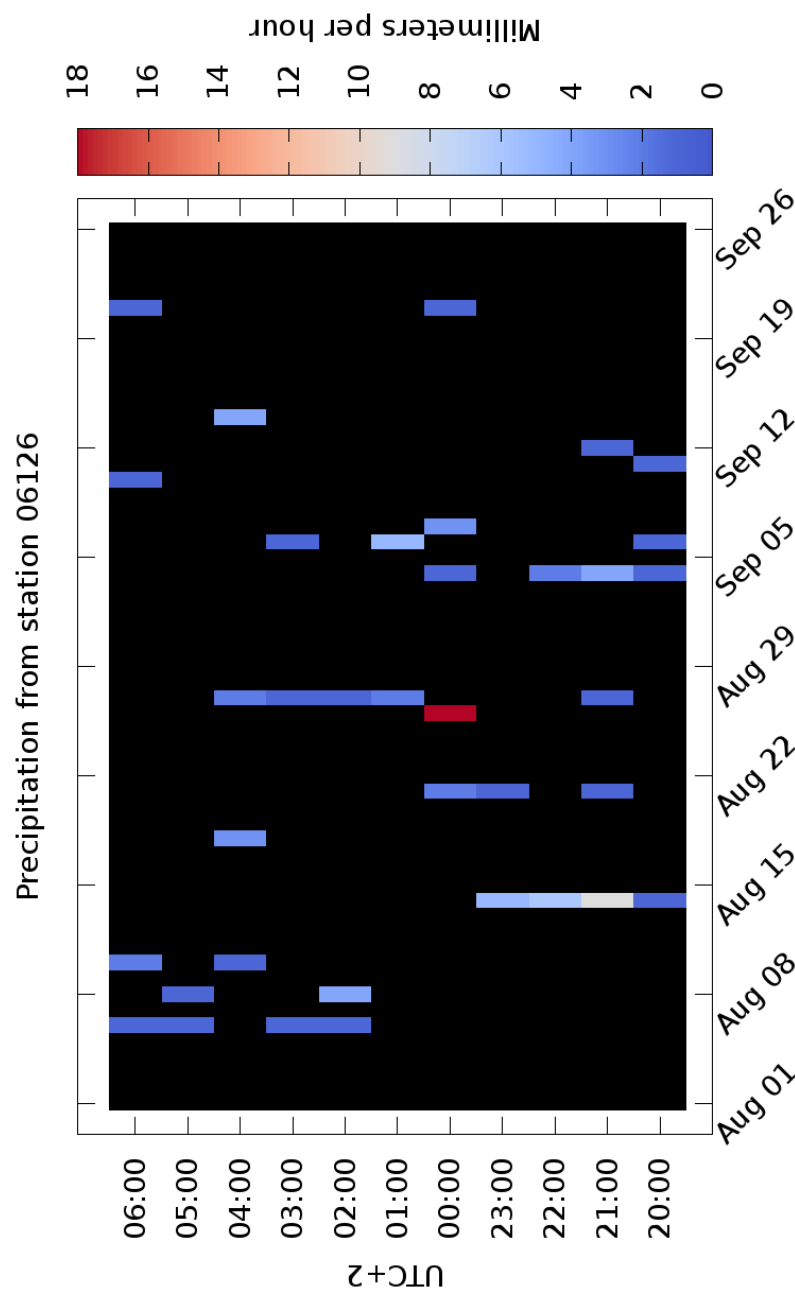


Figure 6.7: Precipitation measured 5 km away from the survey site.

	Aug 3	Aug 11	Aug 24	Aug 26	Sep 4	Sep 10	Sep 14
<i>Eptesicus serotinus</i>	145	0	34	58	0	20	0
<i>Nyctalus noctula</i>	12	0	31	6	0	3	1
<i>Pipistrellus pygmaeus</i>	16	10	38	13	0	47	31
<i>Pipistrellus nathusii</i>	0	0	2	0	0	5	1
<i>Myotis daubentonii</i>	0	0	3	0	0	0	0
<i>Myotis dasycneme</i>	0	0	2	0	0	1	0
Non-bat event	16	0	36	104	31	50	62

Table 6.1: Manually identified bat events in a sample of events labeled as candidate bat events by the custom scaled SVM classifier..

to keep the event count to be inspected below 200 per day. Note that this was done on a random subset of all events, so some days had few events. In total we reviewed just fewer than 800 events. Table 6.1 shows the results from this tally where we have identified a number of different bat species with fairly high degree of confidence. Figure 6.8 shows some of the typical events counted in the table.

Clearly, there are still many non-bat events present in the reduced collection but, when looking chronologically through the events, it is obvious that there are clusters of non-bat events, which in all likelihood are related to rain. See for example around midnight on August 26th. We are convinced that these can fairly easily be spotted and removed by an improved automatic analysis.

Another interesting feature of this tally is that we found evidence of two species that have not been registered (Baagøe, 2001) in the survey area before, namely *Pipistrellus nathusii* and *Myotis dasycneme*.

6.6 Long-term survey

The survey described, and the methodology used to effect it, suggest a number of interesting discussion points, both biological and technological.

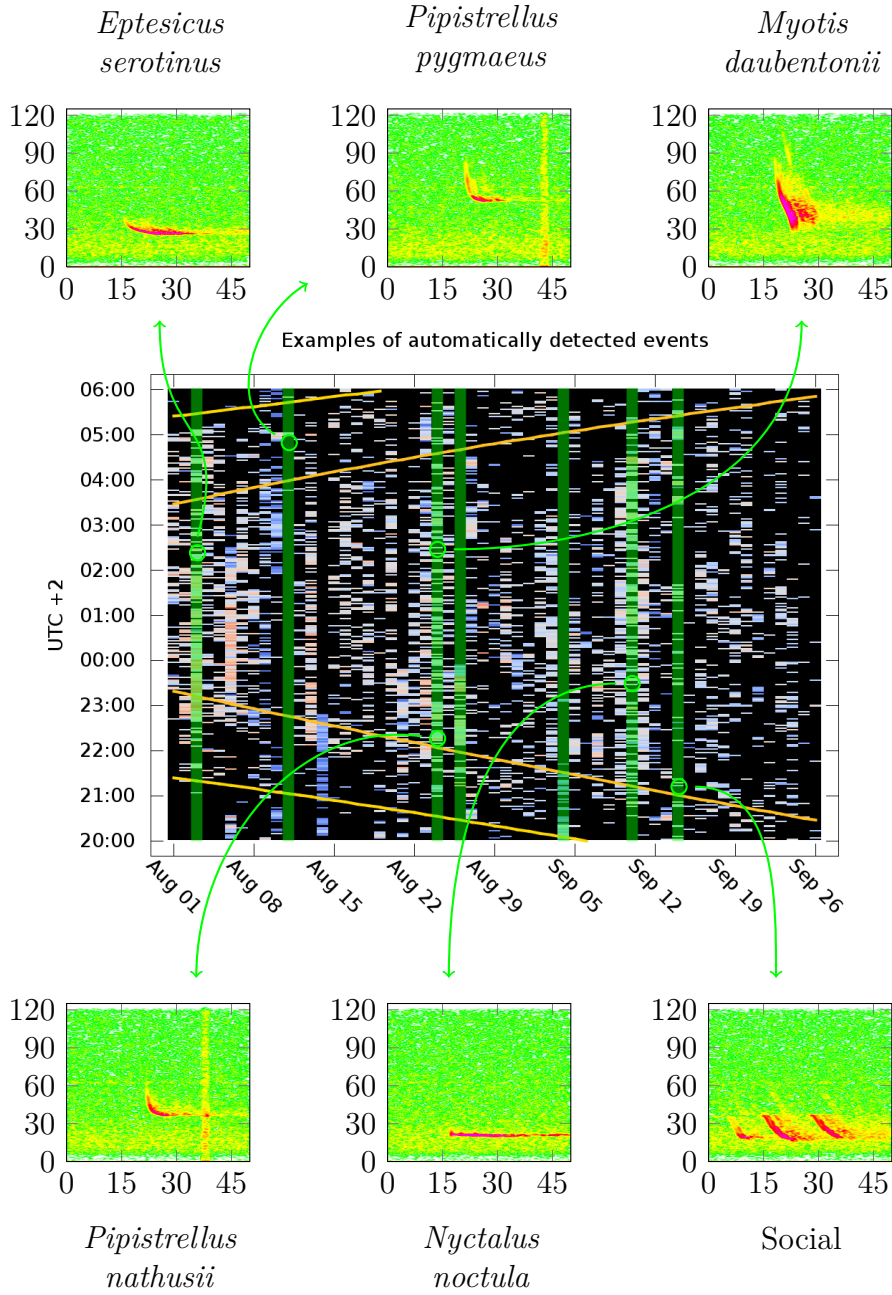


Figure 6.8: Example of typical events classified during the tally. The center figure is a reduced version of Figure 6.5 with the days from Table 6.1 highlighted. The smaller plots show spectrograms of manually classified bat calls with circles and a pointer showing when they occurred.

We discuss some of them below.

Technical viewpoint

Firstly, from a technical viewpoint, the data processing has been designed to favor simplicity whenever that provides adequate functionality. This reflects the emphasis of the current work on proving the concept of this approach rather than squeezing the last drop of power from it. For example, the single-threshold approach to event marking tends to label isolated regions of a recording file, and the mark clustering then aggregates nearby marks to make the events. A dual-threshold approach — a high threshold for the initial mark, and a lower threshold to determine the start and end of the candidate event — might be preferred because this would include low intensity calls from the beginning of a sequence of calls from a passing bat.

The support vector machine was chosen as a robust and versatile classifier, but there may be techniques better suited to the specific class discrimination problem posed by our data. In particular, adjusting the classifier parameters depending on the presence or absence of rain would be sensible, given the large number of rain events highlighted by the initial marking process: the prior distribution of the two classes is markedly different on rainy and dry days.

The question of scaling the feature vector for the classifier is also open. For the training data set, unscaled data clearly results in a very good classifier (Figure 5.7); absolute rescaling causes a small loss of performance but incorporates more biological knowledge about the reasonable range of values for the chosen features (Figure 5.7(c)). It remains to be seen whether it is better to use the absolute scaling trained classifier when running on the bulk of events, as we did here, or to keep to the unscaled data trained classifier that performs better on the training set.

One might also envisage refinements of the chosen features, or further features that would increase the discriminating power of the classification process.

Finally, given the initial biological survey carried out here, the available

pool of labeled training data can be expanded using all the candidate bat events examined by the biologists, since their true class is known after inspection.

The analysis presented here used data from a single channel in the recorded data and also treated events as isolated occurrences. Given the four channels recorded, one can in principle recover the spatial position of the sound source relative to the recording array using time-of-arrival-difference techniques (Chan and Ho (1994) and Madsen and Wahlberg (2007, §2)). This would allow the system automatically to recover spatial data for the bats detected. Correlating event marking between the four channels might improve event detection. Temporal patterns in the single channel event stream could also be extracted: bats do not generally emit isolated calls, but structure their call patterns temporally to reflect the activities in which they are engaged. One might therefore recover data not only about which bat species were present but also what they were doing, e.g. insects caught using buzz sequence counting. These analyses entail technical difficulties relating to signal quality and signal matching, but there are many approaches in the literature worth trying (O’Farrell, Miller, and Gannon, 1999; Gorresen et al., 2008; Skalak, Sherwin, and Brigham, 2012). Note that automatic recovery of buzz sequences with the automatic method still needs work. The current method has issues with the high repetition rate and the short duration of buzz phase emissions, especially the termination buzz which typically has 0.3 ms duration and -20 dB SPL.

Finally, it would be advantageous to augment the survey recording equipment with a small weather station able to monitor at least local temperature, humidity, illumination and rainfall. The former two data are needed to compute an accurate value for the speed of sound — for calculating spatial data — while all data are also relevant to bat behavior. This would certainly improve measurements of localized phenomena. Adding a sensor suite that measures some of these values together with the acoustic recordings has already been tested, see Section 4.7 on page 53.

Biological viewpoint

From a biological viewpoint, there are a number of interesting ecological results from the survey. First, the variety of species sampled at the recording site includes frequent appearances by individuals of at least two species believed not to be present in the area, *Pipistrellus nathusii* and *Myotis dasycneme* (Baagøe, 2001). Thus this technique proves very promising to survey species that are rare, and thus rarely sampled in short random surveys or transects. One advantage of a longitudinal survey such as this is that one potentially observes everything that occurs, rather than relying on careful judgment and some luck to be surveying at the right time, which will prove useful for estimating the true occurrence of less common species and also determine areas of importance for passing events such as, for example, migration.

Another, surprising, observation from the subset of survey data inspected so far is that social calls were frequent compared to the observed rates in a large number of echolocation studies. There are only few systematic studies of social calls in the field in temperate areas (Lundberg and Gerell, 1986; Barlow and Jones, 1997a; Barlow and Jones, 1997b; Russo and Jones, 1999; Russ and Racey, 2007; Georgiakakis and Russo, 2012). Recordings of social calls emitted by bats may provide data for assessing cryptic diversity and investigating reproductive strategies and foraging ecology. Further, social calls reveal areas of importance for reproduction and roosting. Occupancy may help evaluate reaction to anthropogenic impact, not only through exploitation of land resources, but also through climate change and other changes of habitat.

One may consider the extent to which the survey is a reliable indicator of bat population and activity. A full answer to this question awaits a more complete analysis of the collected data, but initially we can say that the observed bat activity certainly underestimates what is really present. The automatic processing focuses on minimizing false positives, but may generate many false negatives to be caught by more sophisticated automatic processing not yet implemented. Thus, while inspections suggest that around

30% of candidate bat-call events proposed by the current signal-processing chain will turn out to be misclassified noise, there are certainly bat calls present in the large bulk of data not highlighted for human examination.

It is apparent in Figure 6.5 that the bat activity varies across the sunrise and sunset periods in a systematic way, and one could construct plots of the density of bat calls (in total or per species) against the time relative to sunrise and sunset to investigate those phases of the bat's night. This also shows that the survey length should be adjusted to the length of the day instead of a fixed number of hours.

Of course, no survey methodology is unbiased (Flaquer, Torre, and Arizabalaga, 2007). In our case, the principal bias favors loud bats since they will record calls that more easily pass our energy thresholds. This is partially an artifact of the simple signal processing chain we have presented. Having extracted all the loud bats, one could continue and search for quieter ones in the remaining data — especially if given models of the types of call made by species of interest. Such a search could also be automated, but would be much more time-consuming than the process adopted in this survey.

A rather more subtle (absence of) bias is suggested by the observation that bat calls are detected in our survey very close to the end of periods of rain. Given that weather can be a very local phenomenon and that the nearest weather recording station is 2 km from our survey site, we can only speculate; but it appears that bats may either fly immediately after rain or indeed during rain. Careful analysis of the data from rainy periods may reveal bat calls partially masked by the rain noise. It could perhaps be that the generally held belief that bats do not fly in rainy weather reflects a preference of biologists (for not recording in the wet) rather than of bats. The serious point is that the methodology we demonstrate avoids this kind of potential observer bias, simply by observing all the time. The long-term automatic survey not only permit sampling of data that are un-biased by weather conditions, but also less biased by difficulty of access to a potential bat habitat, since it would require access only once or a few times.

As a final comment, the principal difficulty of the survey methodology

we advocate here is the sheer volume of data generated. A full season of recording bats, from say April to October, would generate nearly 1 TiB of recorded data. Storing such a data volume is trivial nowadays, but analyzing it by hand is infeasible. The automated processing developed in Chapter 5, with the improvements discussed above (and others), can reduce this data volume to something manageable for a busy biologist. For specific questions such as species density it may be possible to automate the answers completely. Our survey demonstrates the growing possibility of carrying out ecologically interesting comprehensive long-baseline studies of acoustically active animals without dedicating the whole of one's working life to the project.

6.7 Summary

The goals of this survey were two: to investigate and document local bat population behavior in the vicinity of the University of Southern Denmark; and to test and demonstrate that an automatic array recording system could be deployed long-term and the resulting voluminous data could be analyzed effectively.

Our inexpensive scalable array recording system was used for the survey, and was at the survey site for just over three months. Despite some unexpected power interruptions, the system functioned largely as intended and we were able to capture two continuous months' worth of recorded data totaling nearly 240 GiB. Automatic processing of the data successfully identified around 115,000 candidate bat "events" so-labeled by a support vector machine classification algorithm, of which we expect some 70% to be recordings of actual bat calls. This represents a small percentage of the total volume of data which requires human judgment for further analysis. A number of plausible improvements to the automatic processing may eliminate more of the data from initial consideration; conversely, improvements may also retrieve less conspicuous bat calls from the data rejected by the current analytic methods. We conclude that automatic long-term recording is both feasible and practical, given our experience with this system.

From a biological perspective, the survey also reveals some interesting results. Examination of a sample of events from specific days shows a variety of bat species observed at the survey site, including a number of rare or infrequently observed species. A surprisingly large proportion of the total calls inspected turned out to be social calls rather than echolocation calls. There are suggestions of correlations between bat activity and local weather: busy activity on warm nights especially following cloudless moonlit nights, and bat activity surprisingly close to the end of rain showers. These tentative observations require further detailed analysis to confirm them, and the data collected supports such analyses.

Discussion

Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it.

— Brian W. Kernighan
and P. J. Plauger

The discussion here mainly focuses on the technical aspects of this project, although many of them are biologically relevant. For the discussion related to the biological aspects of the long-term study, please refer to Section 6.6.

7.1 The MCLURS recording system

The ultrasonic recording system we propose is designed to achieve an unprecedented flexibility: as the interface is based on widely used open web standards, few limits are put on what type of equipment can communicate with the system. For example, any system which has a web browser installed will be able to control and get feedback from the recording system — which implies, for example, that the system can be controlled by smartphones allowing flexible, wireless, universal access to the data. This has been tested by successfully running a terminal program on an Android smartphone. Simple preview, analysis, and control interfacing can be achieved by installing GNU Octave on the same type of smartphone; alternatively, more sophisticated analysis could be carried out on the module's local computer and the smartphone be used only for display and control.

As shown by the tests we performed, the versatility of the system was sufficient to enable application to all of the experiments we intended. But an extra advantage of the open design is that it allows others to modify, remove or add components to the system and apply it straightforwardly to many other tasks which we have not yet anticipated. Any user interface could potentially be used to control the system. The stream format is

publicly available, so anyone can implement and integrate a user interface that would work with MCLURS. Below we outline some of the perspectives in such an open, modular, scalable, and cheap system.

MCLURS design has been tested in various settings, and the results show that it is also applicable for doing large-scale recording, synchronized emission/reception and calibrated emission. Another hallmark of the current design is the ease with which the configuration of prototype 1 was changed from, for example, multi-channel recording of short events like the 1–2 second echolocation sequence of a trawling *Noctilio leporinus* over to 1–2 minutes to record the scanning sequence of *Micronycteris microtis* over to long-term recordings of bat activity for a whole season and finally to function as an input/output system deployed for two different purposes, robot echolocation and microphone calibration. The proposed system design has proven that it would be a viable framework for doing large scale recordings.

As covered in Section 3.2, this design is very open ended and thus has allowed the development of a markedly different second prototype implementation, but without changing the design or most of the software components.

Component tagging

When scaling the system to arbitrary size, keeping track of individual channels or modules of the system becomes a book-keeping challenge. However, with an open framework everything is potentially programmable, and the system can be extended to deal automatically with this task. Electronically tagging microphones, filters and computers is straightforward and would allow system software to keep track of which components were in use in what combinations and at what time. The amplifier/filter electronics and microphone plugs have already been prepared to receive such an addition.

A second benefit of the possibility of electronically-tagged system components accrues if the tags can store acoustic calibration data. Each microphone could carry its own calibration information, as could each analog channel in each module. Plugging a microphone into a channel could then

cause the automatic application of compensating (digital) signal processing so that a calibrated recording transfer function was realized independent of which microphone and channel were actually used. By this means, relatively inexpensive microphones might achieve the same recording tasks as expensive, stable, calibrated microphones.

A further question implied by this possibility is the stability of the microphone with aging and environmental conditions. The functionality of the Knowles microphones after being subjected to the Danish outdoor environment has been verified, but the stability of their transfer function over time and with use still needs to be determined. In particular, it is not known whether (or how much) the transfer function varies with temperature, humidity, and wind. Tagging microphones with their calibration data would go some way toward answering this question too.

Distributed design

The distributed design also includes the possibility of distributing computation and storage and this has already been done to some extent. Often the module computers were made to generate time-plots and/or spectrograms with `SoX` and `Gnuplot`, see appendix C for details. This would then be passed on the controlling computer. This allows the user to decide what data is worth fetching.

Automatic geometry determination

One useful possibility is the automatic determination of microphone array geometry, which is essential for computation of flight paths and is a major contributor to the effort of deploying a large-scale array. Using a small, loud impulsive sound source — such as an electrical spark — the geometry of an already-installed array can be calculated using time-of-flight computations to determine the positions of the (perhaps many) microphones while simultaneously calculating the positions of the sparks. Such software could be integrated into the array module local computers, making the array almost self-calibrating. See Dubgaard (2012) for a more in-depth discussion.

Specification (per ch.)	Avisoft		Proto. 1		Proto. 2	
Sample rate	750	KHz	2	MHz	750	KHz
Bits per sample	12	bits	14	bits	12	bits
Variable gain	0-40	dB	-20-60	dB	-20-60	dB
Volume	240	cm ³	2200	cm ³	140	cm ³
Weight	260	g	1000	g	170	g
Power consumption	5.4	W	10.4	W	2.8	W
Price per 4 channels	\$2200		\$950		\$700	

Table 7.1: Three array systems are compared based on their per channel parameters: a system based around an Avisoft AD converter which needs a computer running Microsoft Windows, the prototype based around the Adlink AD converter and the prototype based around the *USB-DUXfast* AD converter. Power consumption calculation of the Avisoft array includes a laptop using 20 W. The price compares the cost of complete 4 channel arrays of the specific type.

Prototype comparisons

This second prototype has been optimized for portability, low-power usage, and many channels, thus making it a good candidate for a field multi-channel recorder. There are, however, some drawbacks with this approach compared to the first prototype: the maximum sample rate is lower, the bit depth is lower (12 bit compared to 14 bit) and it does not include a digital-to-analog converter.

Table 7.1 compares the prototypes and a system based on the Avisoft AD converter. Recently a new ARM based SBC (single board computer) has emerged that retails at around \$25 called the Raspberry PI (Pi, URL provided, accessed: May 2013). According to the specifications of this SBC it would be well suited as a drop-in replacement for the Dreamplug computer. This would bring the price of the second prototype in Table 7.1 down to circa \$500 for a 4 channel system.

7.2 Flight tracking

Flight tracking requires that at least 4 microphones record the same emission. The time of arrival delays (TOAD) can then be used to recover the sources' position at the time of emission (Madsen and Wahlberg, 2007, §2).

Configuration

Tracking inherently requires that the geometry of the microphone positions is known. The solution to this is either to have the microphones placed in a rigid structure with a known geometry, manually measure out the microphone positions, or automatically calculate it (Dubgaard, 2012). Theoretically the optimal rigid structure would be a tetrahedron, because its spherical symmetry offers position accuracy independent of direction, which may be important for monitoring in environments where it is unclear from which directions animals may approach. There is some anecdotal evidence (personal communication) that tetrahedral configurations with short edges can only reliably determine azimuth and elevation angles towards the source but not distance. This can be improved by using multiple tetrahedral configurations, and calculating the intersection of the vectors from each tetrahedron.

Biological application

If individual bat trajectories are calculable several interesting new features become possible. We discuss three of them here: volumetric normalization, behavior assessment, and individual counting.

Louder bats will always be detectable in a larger volume than quieter bats and this will inevitably bias population estimates in favor of the louder species. However, given spatial data for each call, one could straightforwardly normalize for volume and derive bat population density measurements in units of individuals per cubic meter.

Acoustical recordings can recover behavior based on the type of emitted sequences, if this data is augmented by movement data, it becomes possible

to know more details of what and with whom the bat was interacting.

One potential problem with surveying bats through acoustics alone is that the recorder might be recording the same individual for a long time. With spatial data it is much easier to attribute sequences to the same individual. Another issue is that multiple bats might be counted as one, separating emissions from multiple concurrent sources becomes possible when their positions are known.

7.3 Localized weather sensing

Correlation with weather and ephemeris information suggests that further interesting observations may be possible. For instance, the data plotted in in Figure 6.5 (re-plotted here as Figure 7.1 for convenience) shows nights with little bat activity followed by nights with much activity (e.g. September 8–11). Correlation with weather data (temperature and wind, principally), lunar phase and cloud cover (measured some 15 km from the survey site) suggests that the quiet nights in question were cloudless and therefore cool with bright moonlight, while the following nights were cloudy and warmer. See figs. 7.2 and 7.3. This type of analysis would be facilitated by a local weather sensing addition like the one discussed in Section 4.7 on page 53, although it is missing a luminance sensor. If local luminance is measured inference from cloud cover and lunar cycles are no longer needed. Another benefit of local climate recording is that we are in control of the measurement frequency and, we believe, would have fewer incidences of missing measurements, for example Figure 7.3 is missing data from August 22nd through September 1st. We received these measurements from DMI (Danish Meteorological Institute).

7.4 Open source

Given the two new additions discussed in the previous sections, one might expect them difficult to add to a MCLURS module, however, this is not the

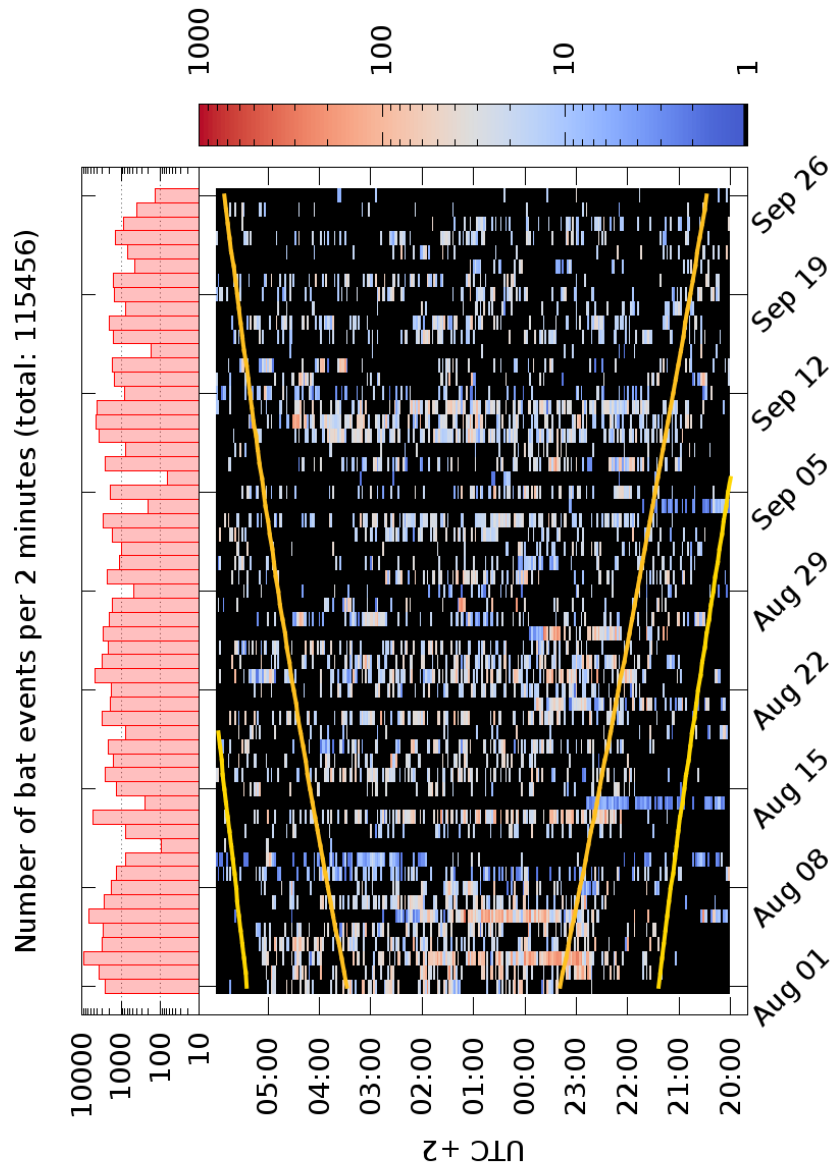


Figure 7.1: Re-plot of Figure 6.5 for convenient comparison with the cloud cover/lunar cycle plot in Figure 7.2.

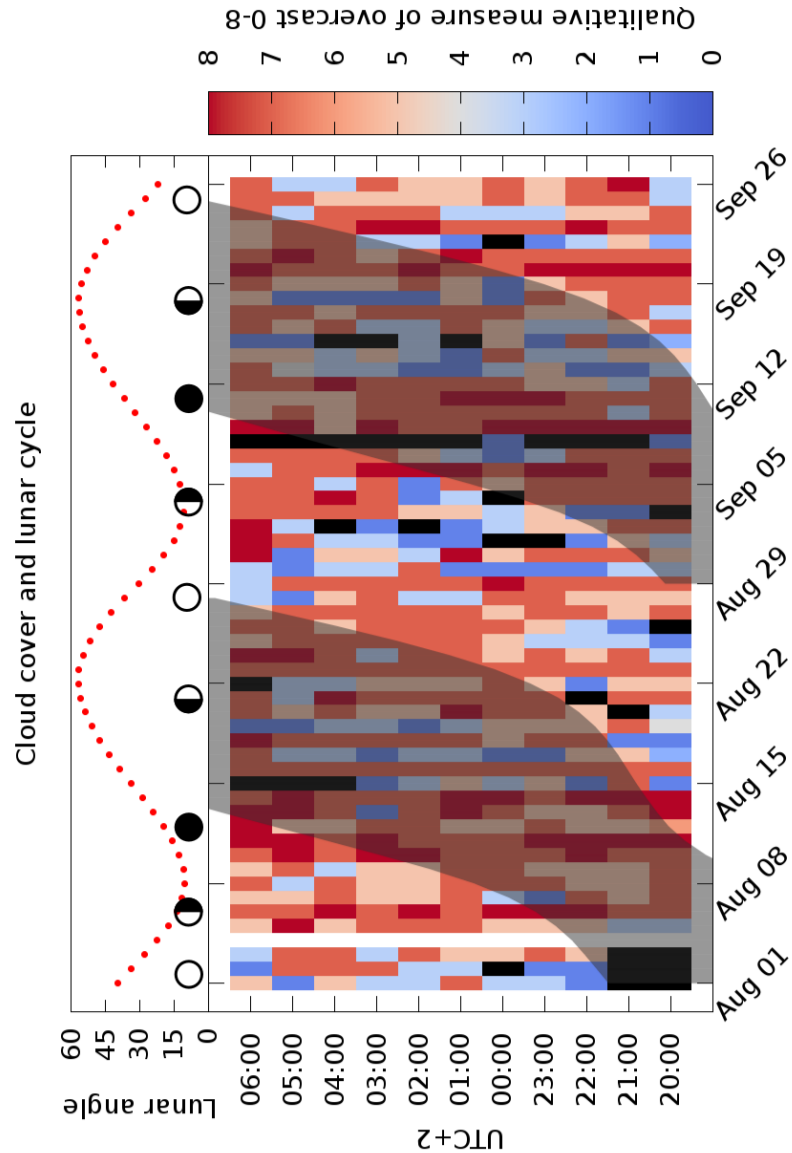


Figure 7.2: This plot shows how the cloud cover and lunar cycle change relative to each other. The color gradient shows the hourly overcast value, with 0 or black being clear skies and 8 or dark red being completely overcast. The overlaid shade shows when the moon rose and set. The phases of the moon are shown above the overcast plot. Finally, the top figure shows at what elevation angle the moon was at when it reached its apex.

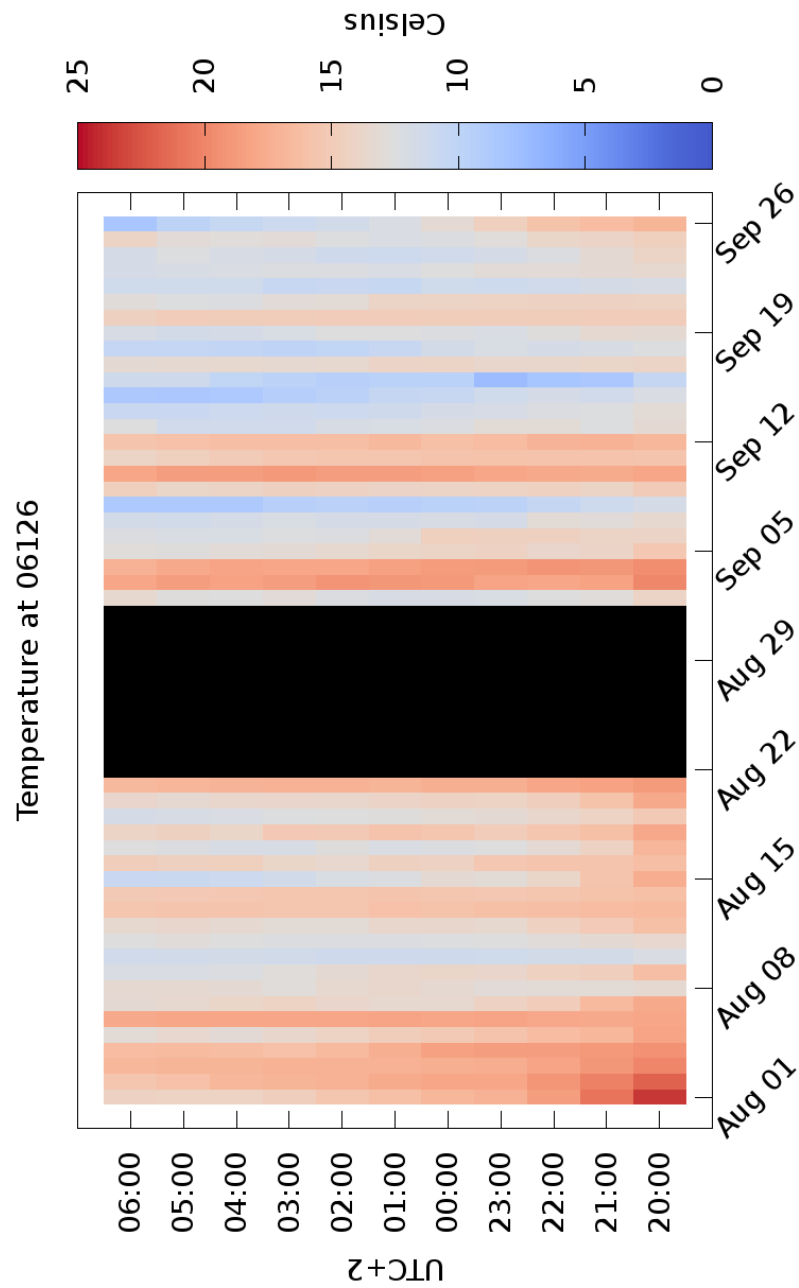


Figure 7.3: Temperature measurements during ecological study.

case. Because of the open source nature and the generic design it is arguably a fairly trivial task to do automatic flight tracking and weather monitoring directly in the MCLURS module.

This section presents the philosophical perspective of the method of design we have chosen and tries to provide some historical context. The benefits of this approach over other ones is documented with some examples.

Some history

In order for innovation to flourish, a free flow of ideas is needed. This is not the case in today's commercial research and development culture, and to some extent not the case for academic research and development either. On the sideline, we have the open source way of working. This is not a new phenomenon, however, it has been around as long as people have been inventing. Many examples of this sort of knowledge sharing have been documented all the way back to ancient times. More recently the Royal Society in London was established in 1665, and it took it upon itself to gather and disseminate research results. Engineers of the 1800s were known to collaborate and share ideas freely, although later they were financially pressured into the patent ecosystem in which they have been perpetually competing ever since. Enter the radio amateurs, who have carried the tradition onwards since the early 1900s, all designs and interesting tidbits have been shared and shared alike. No one hoards their knowledge, and the community was richer for it. The computer scientists inherited this culture and this has been carried forward into today's open source communities.

One of the main benefits of leaving everything open and public is that there are no restrictions on what the next person or organization can do with the designs. In a very real sense this also brings in competition: if you didn't create something other people want to use, it will perish and die a silent death in obscurity, whilst the next project over might be a success. Success is a fickle thing and has been a very real issue for many projects over the last decades, commercial and otherwise. What makes people choose one product over the next is not technological superiority. Betamax was technologically

much better than VHS, for those who remember that controversy. Clearly many technologies have done very well in the marketplace without being the best there.

Success through proliferation

Adding the freedoms to modify, share, and distribute to any work often has the effect of other people feeling they own part of it and making them more inclined to contribute back to that work. This is a very powerful concept, as there are potentially millions of contributors, depending on the proliferation of the work.

Allowing anyone to tinker with your design opens up the possibility of contributions back to the original work, and indeed this is how one of the most successful operating systems of today came about. This collaborative form of working, sometimes referred to as the bazaar as opposed to a cathedral (Raymond, 1999), has proven a very successful development model, although it requires some special ingredients.

Proprietary vs. open source solutions

I believe that this question is very important for all academic work. This opinion is based in part on the arguments put forward in Joyner and Stein (2007) which concerns mathematical software, but they ring true in many other areas as well.

There are other concerns that also arise from proprietary solutions, for example: integrating solutions becomes quite difficult when you do not understand the inner workings of all the implicated parts, distributing your solution is hard without breaking several laws, even internal distribution within your own organization might be unlawful, and it becomes nearly impossible to share your solution with others.

These issues were borne out during this project when a great amount of data became unavailable because it was saved in a proprietary format and the software needed to decode it was “missing”. This is a nonsense situation when operating in the academic world. Surely all formats used

should be open and readily accessible. The only reason for secrecy is to protect the vendor's product line, often with the side effect of failing to respect (or even consider) the user's rights.

A/D board driver issues

Adlink provided one of the most open AD platforms available at the time when this project started, but it was still proprietary and without source code. During most of this project we have depended on the proprietary drivers from Adlink to handle our A/D and D/A converter communication. This has meant that we were limited to running operating system kernels against which Adlink had specifically built their driver, effectively limiting us to optional modules that we could compile outside of this specific kernel. It is rather inconvenient to be forced in this manner, effectively putting restrictions on other components running on the same system. If we had had a choice of a lower-end AD converter, but with better open source support we would definitely have gone that way, and indeed this is what has happened with the second prototype. The assumption that one size fits all is detrimental to academic development if not to all types of R&D. Time and again working with your user community has proven to strengthen both the provider and the receivers of the technology.

When we are doing frequent repetitions of record or emit/record runs, on average every 50th run will freeze the system completely. We have tracked this issue down to the DAQ-2010 driver. Unfortunately, because of the proprietary nature of the driver, we have been unable to pursue this any further. One workaround has been implemented (`adac3`) which engages the A/D board upon system start and keeps it running perpetually. This process only stores anything when receiving commands to do so from a separate program. This works as long as the process that freezes the system is not the first one. The real solution to this issue is either to have Adlink solve it or find and use A/D and D/A equipment that comes with better drivers.

```
% adac -r 500e3 |  
2 tee raw_data |  
   signal_detect |  
4 sox -r 500e3 -c4 -t s16 - -t wavpcm remix 1 3 |  
   ssh -e none OTHER_COMPUTER 'further_analysis > outfile'
```

Source Code 7.1: Advanced pipe example. `adac` AD converts data and forwards it to a signal detection program, which decides what to send on to `SoX`. The stream is then converted to Wav format and sent on to `OTHER_COMPUTER` by `ssh`.

7.5 Advanced use of the system

In Source Listing 7.1 is a short working code example that illustrates the flexibility of MCLURS. The vertical bar (`|`) indicates a pipe, which links the output of one program to the input of another, running independently in parallel. The example uses linked programs to detect signals, convert recorded data into Microsoft Wave PCM (Microsoft, 1991) format, and to forward channel 1 and 3 to a different computer for further analysis. This is an actual working program chain; long command lines are split at the vertical bars for improved readability. Note that this is still valid syntax as pipe characters double as line continuation marks. The breakdown of the chain is described in more detail in the following:

1. `adac` (Source Listing A.1 on page 113), a custom program, takes one argument, namely the sample rate. It acquires samples from the AD converter at the specified rate, 500 thousand samples per second per channel in this case, and emits them in its output stream.
2. `tee` (GNU coreutils developers, 2012), a standard GNU coreutils program, is an input replicator. It saves one copy of the data stream to a file named `raw_data`, whilst sending a second copy forward to the next linked program.
3. `signal_analysis` (not implemented yet), a custom program, conducts **local** real-time analysis; this is where matched filters and other

data selection criteria could be applied, and where threshold-triggering is implemented when that is used. Data judged interesting by this program is passed along the chain; uninteresting data is dropped (but a copy remains in the `raw_data` file from the previous step).

4. `sox` (SoX developers, 2012), mentioned above, is a powerful audio processing program with more than 20+ years of usage and active development. Here it converts channels 1 and 3 of the (interesting) raw data to Wave PCM format. Note that `sox` needs to be told the sample rate (`-r 500e3`), number of channels in the input stream (`-c4`) and sample format (`-t s16`) i.e. little-endian signed 16 bit. This information is encoded in the Wave PCM file format header.
5. `ssh` (OpenSSH developers, URL provided, accessed: Feb 2012), a standard communication program, connects to the remote computer `OTHER_HOST` and forwards the Wave PCM-encoded data stream to a custom *remote* analysis program `further_analysis` (not implemented yet).

The complete command chain is built up piecewise, offering flexibility and choice to the user, while allowing rather complex analysis to be accomplished with a chain of individually-simple steps.

7.6 Comments about available ultrasonic speakers

Transducers capable of ultrasonic emission are readily available, but most of the ones we have tested all have issues with their packaging. One concern is, that because they are built into a metal box this results in backscatter from the backplane of the enclosure. We have tested this both with the Avisoft Scanspeak dynamic speaker (Figure 7.4(a) Avisoft (URL provided, accessed: Aug 2012)) and the electrostatic SensComp Series 7000 polaroid transducer (Figure 7.4(b) SensComp (URL provided, accessed: Aug 2012)) with similar results. The emitted signal needs to be very short before it

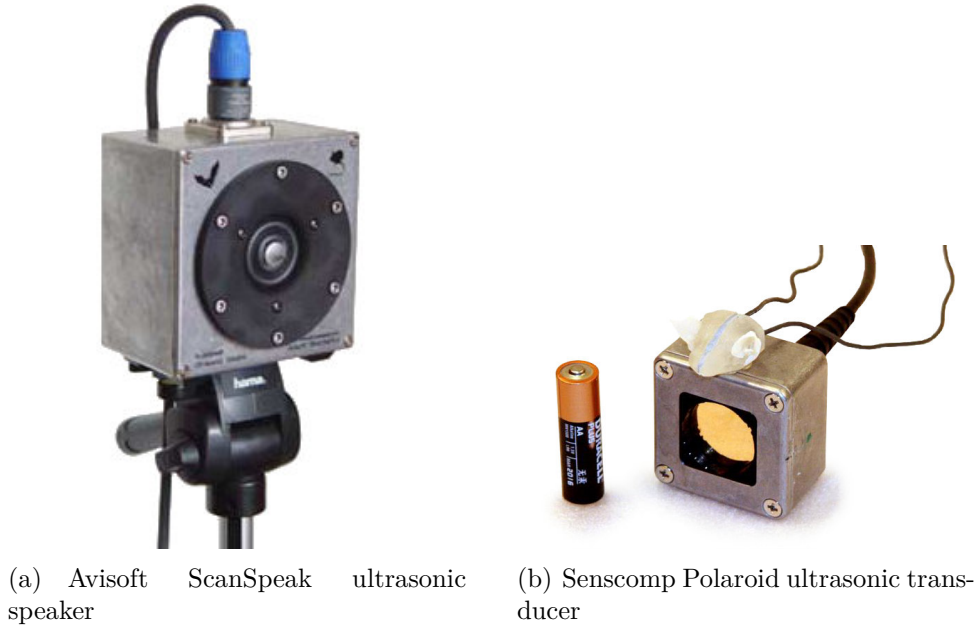
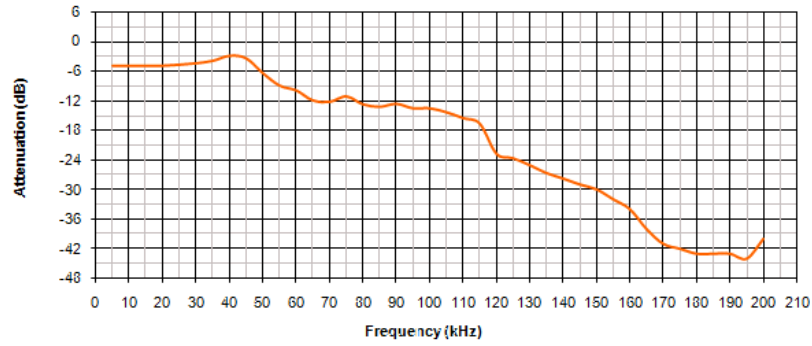


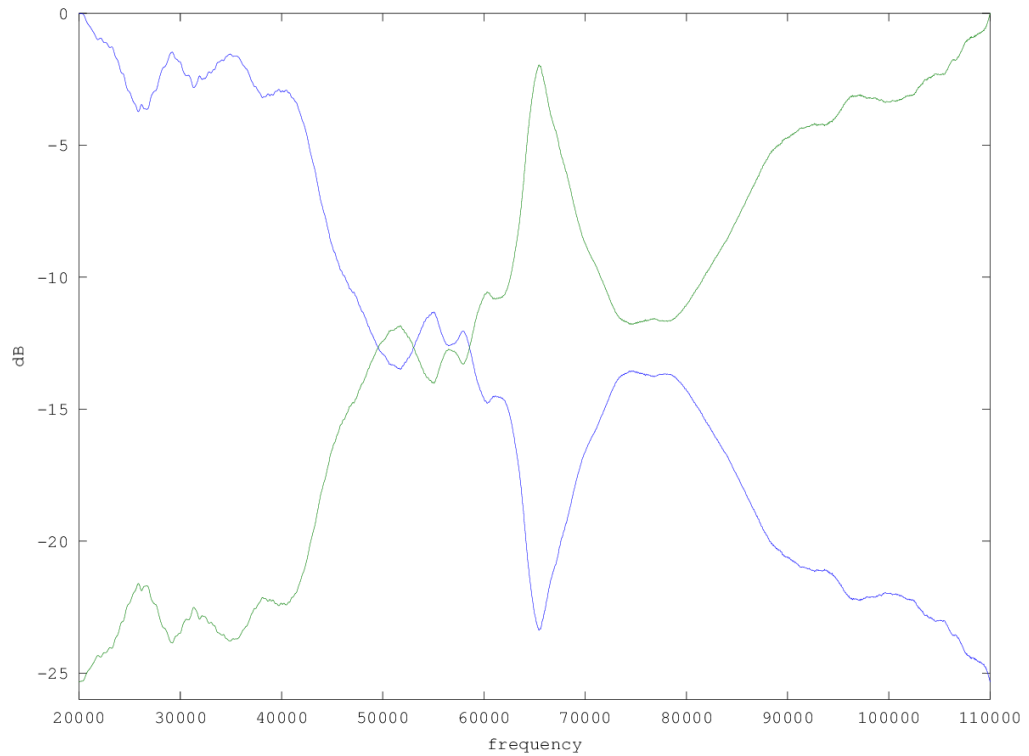
Figure 7.4: Picture of an Avisoft ScanSpeak ultrasonic speaker and a Senscomp Polaroid ultrasonic transducer built into a customized box.

becomes visible. This backscatter depends on the size of the transducers' enclosure. The preferred solution would be to have no backplane at all, and indeed some of the other custom built electrostatic transducers tested have been built in this way. They do not suffer from this effect.

Sometimes, equipment turns out not to comply with the published characteristics, this was our experience with the ScanSpeak ultrasonic speaker. Figure 7.5(a) shows the published transfer function for this model, while Figure 7.5(b) (blue line) shows the measured transfer function. If this is a general discrepancy or confined to this unit still needs to be ascertained. In any case a fact that is important to be aware of before performing experiments that rely on the speakers transfer function.



(a) Published transfer function



(b) Blue line is the measured transfer function, the green graph is its reciprocal

Figure 7.5: Published and measured transfer functions for Avisoft ScanSpeak ultrasonic speaker.

Concluding remarks

The cheapest, fastest, and most reliable components are those that aren't there.

— C. Gordon Bell

We have demonstrated a modular ultrasonic recording system design that is useful, inexpensive and versatile, and fulfills all of our criteria (Table 2.1 on page 8) for use in tasks involving recording or playback of ultrasonic signals for animals or animats.

The implemented system has been tested in a variety of application settings, and is able to record and emit ultrasound at sufficient rates and process the resulting data volumes. The use of open design allows for an easy upgrade path, distributed deployment, long-term recording and emission, low-cost implementations based on retail components and open source software/hardware.

This system allows exploration of new areas of the ultrasonic soundscape. Multi-channel long-term surveys become possible, making it feasible to obtain more accurate assessments of presence and activity of bats or other sound-emitting animals. The flexibility of the system may also lead to applications not yet anticipated.

When building tools for academic exploration, open source software and hardware provide transparency and ease of design sharing, thus increasing the possible proliferation of successful ideas.

8.1 Possible avenues to pursue

Improve microphone protection

A different approach used by other scientists has been to cover the microphone completely from above, and put an acoustic reflector below; directing the microphone towards the parabolic reflector (bobsdata.com, URL provided, accessed: July 28 2008) should give useful results. This approach

would be interesting to explore further and compare to the other methods. The downside is that the sound has to travel farther and the potential for extraneous reflections and filtering effects increases.

Microphone variability

The function of the Knowles microphones has been verified, after they have been subjected to the outdoor environment, however the stability of the transfer function of the microphones still needs to be determined. In particular, it is not known whether (or how much) it varies with temperature, humidity, and wind. The stability over time and usage also needs to be ascertained.

Speaker design

Two of the speaker types we have used are embedded inside metal boxes. This inevitably leads to reflections from the backplate being mixed in with the primary emission. We assert that a better design is to set the membrane inside a frame and avoid any sort of chamber.

Design an open AD/DA converter

The careful reader will have noted that we have had many issues with the current AD/DA solution and serious thought has been put into developing our own open solution in this area.

Species identification

Species identification should be possible based on sound profile, flight profile, daily rhythm and other relevant attributes. This would be interesting to explore and assess its usability. Many have tried already with varying success (Macías, Mora, and García, 2006; O’Farrell, Miller, and Gannon, 1999; Skalak, Sherwin, and Brigham, 2012), but it may be more feasible with a restricted number of well-known bats as input.

Deployment in inaccessible areas

Given the second prototype's cheap price and the potential for automatic geometry determination (Dubgaard, 2012), the leap to a widespread battery operated aerial deployment into inaccessible areas is easy to imagine.

Real-time overview

There is great potential in this design to allow each module to do pre-processing, for example they could determine if what they are recording is interesting enough to send on to a controller. Another option would be to characterize the incoming stream and send reports on to the controller.

8.2 Review of claimed achievements

Here we argue that the points made in Section 1.3 on page 5 were indeed achieved during this project:

- The first prototype has been extensively tested and proven its functionality to a large extent. The second prototype has been tested to some extent, but time restrictions have limited our efforts.
- The system can handle unlimited (or only limited by the computer's storage) duration pre- and post-trigger recording, as demonstrated by Section 4.1. At the same time we have demonstrated that modules can work seamlessly together.
- Robust long-term recording has been demonstrated by several separate long-term outdoor deployments, these are described in Section 4.2 and chapters 5 and 6.
- The system has been applied and proven to be a capable echolocation sensor (Section 4.3), a full-spectrum equipment calibrator (Section 4.4 and appendix B.2, and object echo-profile discoverer (Section 4.5).

- We have successfully developed automated tools that are able to handle the huge amounts of data collected during long-term surveys. The current analysis is capable of extracting bat calls with a success rate of 70% (Chapter 5).
- The system was successfully deployed in a long-term survey which, amongst other ecological results, established that two bat species not known to occupy this area were present on several occasions. This survey also revealed that long-term, all-weather monitoring provides data that may show activity at unexpected places, times or weather conditions (Chapter 6).
- Trying to sound like a bat may be difficult when the emission system distorts the signals. The system has been used to recover an ultrasonic speaker's transfer function and to emit calibrated signals (Section 4.6).
- As can be seen in the tests and evaluation chapter this system has many applications and indeed the system has been successfully applied in many situations and thus has demonstrated its versatility. The ease by which it was adapted to each experiment speaks to the strength of the design.
- We believe that we have demonstrated beyond any doubt that the open source and open design approaches chosen for this project have strengthened its validity and its possible survival in the future.

Long-term study – source code

That which can be asserted without evidence, can be dismissed without evidence.

Christopher Hitchens

A.1 DAQ-2010 analog-to-digital and digital-to-analog control program

An implementation that uses the library provided by Adlink. This program is a multi-purpose implementation that allows synchronized or delayed emission and reception. It can also be configured to only record or only emit. It is also possible to configure the recording and emission sample rate. In addition, duration is also configurable. The full source code can be fetched from Andreassen (2012). Note that `adac` requires that the library and operating system module from Adlink are installed.

Source Code A.1: `adac` – one way to continuously sample the DAQ-2010.

```

1  #include "adac.h"
2
3  void * ai_thread(void *temp_pt);
4  void * ao_thread(void *temp_pt);
5
6  BOOLEAN ai_ready = FALSE;
7  BOOLEAN ao_ready = FALSE;
8
9  prog_options_t opts;
10
11 int main(int argc, char **argv)
12 {
13     I16 ret;
14     void **Tret = NULL;
15     pthread_t T_ai, T_ao;
16
17     setup_signals();
18 
```

```

20     parse_arguments(argc, argv, &opts);

22     /* Register */
23     if(0 > (daq_card = D2K_Register_Card(DAQ_2010, 0)))
24         adac_error("Registering card error, D2K error number: %d", daq_card, FALSE);

26     /* read signal */
27     if(opts.emit)
28     {
29         if( 0 == strcmp(opts.infile, DEFAULT_INFILE, strlen(DEFAULT_INFILE))
30             || 0 == strcmp(opts.infile, "-", 1))
31             opts.insig = stdin_to_memory();
32         else
33             opts.insig = file_to_memory(opts.infile);

34         clear_op_ao = FALSE;

36         if(0 != (ret = pthread_create( &T_ao, NULL, ao_thread, NULL )))
37             adac_error("Unable to create AO thread: %d", ret, TRUE);
38     }

40     if(opts.fetch)
41     {
42         clear_op_ai = FALSE;

44         if( 0 != (ret = pthread_create( &T_ai, NULL, &ai_thread, NULL )))
45             adac_error("Unable to create AI thread: %d", ret, TRUE);
46     }

48     if(opts.emit || opts.fetch)
49     {
50         while(!clear_op_ao && !clear_op_ai)
51             usleep(500e3);
52         sleep(1);
53         if(opts.emit) pthread_join(T_ao, Tret);
54         if(opts.fetch) pthread_join(T_ai, Tret);
55     }

56     /* Release */
57     if((ret = D2K_Release_Card(daq_card)) < 0)
58         adac_error("Release error, D2K error number: %d", ret, FALSE);

60     return(0);
61 }

62 void *ai_thread(void *temp_pt)
63 {
64     /* noop */ temp_pt = NULL;

```

```

70     BOOLEAN halfReady, fStop;
    U32 ai_active_ptr, transfer_count;
    U32 i;
72     long long unsigned buf_switch = 0;
    U16 overrun_flag;
74     I16 *ai_active_buf, ret;
    U16 chans[4] = {0, 1, 2, 3};
76     I16 *ai_buf1, *ai_buf2;
    U16 buf_id;
78     size_t nwrite;
    FILE *ofp;
80     long long total_samples = 0;
    off_t in_buffer = 1<<16;
82
    ai_buf1 = (I16 *) malloc(2 * in_buffer);
84     ai_buf2 = (I16 *) malloc(2 * in_buffer);

86     if( 0 == strcmp(opts.outfile, DEFAULT_OUTFILE, strlen(DEFAULT_OUTFILE))
        || 0 == strcmp(opts.outfile, "-", 1))
88         ofp = stdout;
    else
90     {
        ofp = fopen(opts.outfile, "w+");
92         if(ofp == NULL)
            adac_error("Unable to open output file: %d", errno, TRUE);
94     }

96     if(NoError != (ret = D2K_AI_AsyncDblBufferMode (daq_card, TRUE))
        || NoError != (ret = D2K_AI_ContBufferSetup (
98         daq_card, ai_buf1, in_buffer, &buf_id))
        || NoError != (ret = D2K_AI_ContBufferSetup (
100         daq_card, ai_buf2, in_buffer, &buf_id)))
        adac_error("Error when enabling double-buffered mode, D2K error number: %d",
102         ret, TRUE);

104     ai_ready = TRUE;
    while(opts.emit && !ao_ready)
106         usleep(10);

108     if (NoError != D2K_AI_ContReadMultiChannels (
        daq_card, 4, chans, buf_id, in_buffer/4,
110         (U32) (0.5 + SAMPLE_RATE_MAGIC / opts.sample_rate_ai),
        IGNORED_WHEN_DB, ASYNCH_OP))
112         adac_error("Error when start reading multi-channel, D2K error number: %d",
            ret, TRUE);
114
    if(NoError != (ret = D2K_AI_AsyncDblBufferHandled(daq_card)))
116         adac_error("Error when marking double-buffer handled, D2K error number: %d",
            ret, TRUE);
118

```

```

do {
120   do {
        usleep(10);
122     D2K_AI_AsyncDblBufferHalfReady(daq_card, &halfReady, &fStop);
    } while (!halfReady && !clear_op_ai);
124
    D2K_AI_GetActiveBuffer(daq_card, &ai_active_ptr, &transfer_count);
126
    /* pointer magic to make access to the I16 array easier */
128    ai_active_buf = (I16 *) ai_active_ptr;

130    if (NoError != (ret = D2K_AI_AsyncDblBufferOverrun(
        aq_card, SET, &overrun_flag)))
132        adac_error("Error when resetting buffer overrun, D2K error number: %d",
            ret, TRUE);
134
    if (!clear_op_ai) {
136        for(i=0; i<transfer_count; i++)
            ai_active_buf[i] >>= 2;
138
        if(total_samples + transfer_count > 4 * opts.length)
140            transfer_count = 4 * opts.length - total_samples;

142        nwrite = fwrite(ai_active_buf, 2, transfer_count, ofp);
        if(nwrite != transfer_count)
144            adac_error("Short fwrite in ai_thread: %d", errno, TRUE);

146        total_samples += nwrite;

148        if(total_samples >= 4 * opts.length)
            clear_op_ai = TRUE;
150
        if(NoError != (ret = D2K_AI_AsyncDblBufferHandled(daq_card)))
152            adac_error("Error when marking double-buffer handled, D2K error number: %d",
                ret, TRUE);
154

        /* check overrun status - ignore result from first call as it seems spurious */
156        if (NoError != (ret = D2K_AI_AsyncDblBufferOverrun(
            daq_card, GET, &overrun_flag)))
158            adac_error("Error when checking buffer overrun, D2K error number: %d",
                ret, TRUE);
160

        if(overrun_flag && !clear_op_ai) {
162            fprintf(stderr, "Warning: Overrun happened before double-buffer switch.\n");
            if (NoError != (ret = D2K_AI_AsyncDblBufferOverrun(
164                daq_card, SET, &overrun_flag)))
                adac_error("Error when resetting buffer overrun, D2K error number: %d",
166                    ret, TRUE);
        }
168        buf_switch++;

```

```

    }
170 } while(!clear_op_ai);

172 return NULL;
173 }
174
175 void * ao_thread(void *temp_pt) {
176     /* noop */ temp_pt = NULL;

177     BOOLEAN halfReady;
178     short j = 1;
179     U32 cnt;
180     U16 Id;
181     I16 *buf, *ao_buf1, *ao_buf2;
182     off_t before, after, filei = 0;
183     int ret;
184     int i;
185     BOOLEAN ao_done = FALSE;
186     off_t out_buffer = 1<<16;

187     /* make output buffer size 1/100 of sample rate, experimentally determined */
188     // while((out_buffer <= 1) < (off_t) opts.sample_rate_ao / 10)
189     // ;

190     ao_buf1 = (I16 *) malloc(2 * out_buffer);
191     ao_buf2 = (I16 *) malloc(2 * out_buffer);

192     if(ao_buf1 == NULL || ao_buf2 == NULL)
193         adac_error("Unable to allocate memory for AO buffers: %d", errno, TRUE);

194     if(NoError != (ret = D2K_AO_CH_Config(daq_card, opts.out_channel,
195         DAQ2K_DA_BiPolar, DAQ2K_DA_Int_REF, 10.0))
196         || NoError != (ret = D2K_AO_Config (
197             daq_card, 0, DAQ2K_DA_TRGMOD_POST, 0, 0, 0,1)))
198         adac_error("Unable to configure DAQ card: %d", ret, TRUE);

199     if(NoError != (ret = D2K_AO_AsyncDblBufferMode (daq_card, 1))
200         || NoError != (ret = D2K_AO_ContBufferSetup (
201             daq_card, ao_buf1, out_buffer, &Id))
202         || NoError != (ret = D2K_AO_ContBufferSetup (
203             daq_card, ao_buf2, out_buffer, &Id)))
204         adac_error("Error when enabling double-buffered AO mode, D2K error number: %d",
205             ret, TRUE);

206     memset(ao_buf1, 0, 2 * out_buffer);
207     memset(ao_buf2, 0, 2 * out_buffer);

208     /* handle signals shorter than one and two ao buffers */
209     if (opts.insig_samples <= out_buffer) {
210         memcpy(ao_buf1, opts.insig, 2 * opts.insig_samples);
211     }

```

```

    }
220 else if (opts.insig_samples <= 2 * out_buffer) {
    memcpy(ao_buf1, opts.insig, 2 * out_buffer);
222 memcpy(ao_buf2, opts.insig + out_buffer, opts.insig_samples - out_buffer);
    }
224 else {
    memcpy(ao_buf1, opts.insig, 2 * out_buffer);
226 memcpy(ao_buf2, opts.insig + out_buffer, 2 * out_buffer);
    }

228 adjust_buf(ao_buf1, out_buffer);
230 adjust_buf(ao_buf2, out_buffer);

232 filei += 2 * out_buffer;

234 ao_ready = TRUE;
    usleep((long) (1e6 * opts.delay / opts.sample_rate_ao));
236 while(opts.fetch && !ai_ready)
    usleep(10);

238 if(NoError != (ret = D2K_AO_ContWriteChannel(
240     daq_card, opts.out_channel, Id, out_buffer, 1,
    (U32) (0.5 + SAMPLE_RATE_MAGIC / opts.sample_rate_ao),
242     IGNORED_WHEN_DB, ASYNCH_OP)))
    adac_error("Error when starting to write samples to DAQ card: %d", ret, TRUE);

244 do {
246     do {
        D2K_AO_AsyncDblBufferHalfReady(daq_card, &halfReady);
248     } while (!halfReady);
    if(ao_done)
250     {
        clear_op_ao = TRUE;
252     }
    if( !clear_op_ao )
254     {
        buf = (j%2 != 0) ? ao_buf1 : ao_buf2;

256         if(filei + out_buffer <= opts.insig_samples) { /* if end not reached */
258             memcpy(buf, opts.insig + filei, 2 * out_buffer);
        }
        else if(filei > opts.insig_samples) /* end passed */
260         {
            memset(buf, 0, 2 * out_buffer);
            ao_done = TRUE;
262         }
        else { /* if end reached */
264             before = opts.insig_samples - filei;
            after = out_buffer - before;
266             memcpy(buf, opts.insig + filei, 2 * before);
268

```

```

270     memset(buf + before, 0 , 2 * after);
271     ao_done = TRUE;
272     }
273     adjust_buf(buf, out_buffer);
274     filei += out_buffer;
275     j++;
276     } while( !clear_op_ao );

277
278     memset(ao_buf1, 0, 2 * out_buffer);
279     memset(ao_buf2, 0, 2 * out_buffer);
280
281     /* apparently three we need to wait for three more bufferswitches */
282     for(i=0; i<3; i++) {
283         do {
284             D2K_AO_AsyncDblBufferHalfReady(daq_card, &halfReady);
285         } while (!halfReady);
286     }

287
288     D2K_AO_AsyncClear(daq_card, &cnt, 0);

289
290     return NULL;
291 }

```

A.2 Signal detection

An example of how to do signal detection can be seen in Source Listing A.2. The source code for this program can be found at Andreassen (2012).

Source Code A.2: `signal_detect` – signal detection for long-term recordings.

```

2  /*
3   * Author: Thor Andreassen <ta@toggle.be>
4   *
5   * Description: Read in a stream and search for average energy levels
6   * that exceed a threshold [1] for a 'chunk_samples' time [2]. Signals
7   * are then stored in a file named with the time of occurrence and keeps
8   * storing samples until PRE_POST_EVENT_CHUNKS number of samples of
9   * silence have been seen.
10  *
11  * [1] th [0:2**14] = SENSITIVITY * (10 + average(
12  * pre_event_buffer, pre_event_buffer_samples));
13  * [2] len [us] = SAMPLE_SIZE * CHANNELS * us * fs [bytes]

```

```

14  */
15
16  #include <stdio.h>
17
18  #include <stdlib.h>
19
20  #include <string.h>
21
22  #include <math.h>
23
24  #include <assert.h>
25
26  typedef enum { false, true } bool;
27
28  #define NEWN(type, n) (type *) calloc(n, sizeof(type))
29
30  #define NEW(type) NEWN(type, 1)
31
32
33  #define CHANNELS 4
34
35  #define LONGEST_LONG_LONG 20
36
37  #define FILE_PREFIX "sr_500000_ch_4_offset-"
38
39  #define FILE_SUFFIX ".s16"
40
41  #define SENSITIVITY 5
42
43  #define SAMPLE_MAX pow(2,14-1) /* 14 bit signed samples */
44
45  #define SAMPLE_RATE 500000LL
46
47  #define SAMPLE_SIZE 2 /* in bytes */
48
49  #define BYTES_PER_SECOND (CHANNELS * SAMPLE_RATE * SAMPLE_SIZE) /* 2 MB @ 500 KHz */
50
51  #define CHUNK_SIZE 10000L
52
53  #define PRE_POST_EVENT_SECONDS 2
54
55  #define PRE_POST_EVENT_CHUNKS (PRE_POST_EVENT_SECONDS * \
56                                (BYTES_PER_SECOND / CHUNK_SIZE)) /* 3 * 200 */
57
58  #define ABOVE_LIMIT_SEC 500e-6
59
60
61  double average(signed short *X, size_t nsamps);
62
63  double fmax (double a, double b);
64
65  void save_pre_event_buffer(void);
66
67  void save_chunk(void);
68
69  void close_current_recording(void);
70
71
72  unsigned long long stream_offset = 0;
73
74  unsigned long long tmp_offset = 0;
75
76  int file_name_length = 0;
77
78  static char *storage_path;
79
80  static char *sbuf;
81
82  signed short *pre_event_buffer;
83
84  size_t pre_event_buffer_offset = 0;
85
86  size_t chunk_samples = (size_t) CHUNK_SIZE / 2; /* in samples */
87
88  size_t pre_event_buffer_samples = (size_t) (
89      PRE_POST_EVENT_CHUNKS * CHUNK_SIZE / 2); /* in samples */
90
91  FILE *current_file;
92
93  long gnt = 0;
94
95
96  int main(int argc, char **argv)
97  {
98
99      size_t nsamps, n_silence_chunks = 0;

```

```

double threshold, avg;
64  bool signal_detected = false;

66  if(argc < 2)
    return 0;

68

fprintf(stderr, "%d\n", pre_event_buffer_samples);
70  pre_event_buffer = NEWN(signed short, pre_event_buffer_samples);
assert(pre_event_buffer);

72

storage_path = argv[1];

74

file_name_length =
76  LONGEST_LONG_LONG
  + strlen(argv[1])
78  + strlen(FILE_PREFIX)
  + strlen(FILE_SUFFIX)
80  + 2;
sbuf = NEWN(char, file_name_length);
82  assert(sbuf);

84  fprintf(stderr, "Filling pre-event buffer ...\n");

86  /* determine the value of silence */
if(pre_event_buffer_samples >
88     (nsamps = fread(pre_event_buffer, sizeof(signed short),
                      pre_event_buffer_samples, stdin)))
90     return 1;

92  stream_offset += nsamps;

94  threshold = SENSITIVITY * (10 + average(
                      pre_event_buffer, pre_event_buffer_samples));

96

fprintf(stderr, "Threshold : %lf\n", threshold);
98  fprintf(stderr, "Pre-event buffer ready, "
            "commencing signal detection.\n");

100

while(1)
102 {
    nsamps = fread(pre_event_buffer + pre_event_buffer_offset,
104                  sizeof(signed short), chunk_samples, stdin);
    if(nsamps < chunk_samples) return 0;

106

    avg = average(pre_event_buffer + pre_event_buffer_offset, chunk_samples);

108

    if(avg < threshold && !signal_detected)
110    {
        tmp_offset = stream_offset;
112    }

```

```

114      /* differece should be at least ABOVE_LIMIT_SEC long */
115      if(stream_offset - tmp_offset > (long) (
116          CHANNELS * SAMPLE_SIZE * ABOVE_LIMIT_SEC * SAMPLE_RATE))
117      {
118          save_pre_event_buffer();
119          signal_detected = true;
120      }

121
122      if(signal_detected)
123      {
124          save_chunk();
125      }

126
127      if(avg > threshold)
128          n_silence_chunks = 0;
129      else
130          n_silence_chunks++;

131
132      if(n_silence_chunks == PRE_POST_EVENT_CHUNKS && signal_detected)
133      {
134          close_current_recording();
135          signal_detected = false;
136      }

137
138      stream_offset += chunk_samples;
139      pre_event_buffer_offset += chunk_samples;

140
141      if(pre_event_buffer_offset == pre_event_buffer_samples)
142      {
143          pre_event_buffer_offset = 0;
144      }
145    }

146    return 0;
147  }

148
149
150  double average(signed short *X, size_t nsamps)
151  {
152      size_t i;
153      double avg;

154
155      avg = X[0];

156
157      for (i=CHANNELS; i<nsamps; i+=CHANNELS)
158          avg += fabs(X[i+0]);

159
160      return avg / nsamps / CHANNELS;
161  }
162

```

```

double fmax(double a, double b)
164 {
    return (a >= b) ? a : b;
166 }

void save_pre_event_buffer(void)
168 {
170     fprintf(stderr, "open: %ld\n", gnt++);
    snprintf(sbuf, file_name_length, "%s/%s%0*lld%s", storage_path, FILE_PREFIX,
172             LONGEST_LONG_LONG, stream_offset - pre_event_buffer_samples, FILE_SUFFIX);
    current_file = fopen64(sbuf, "w");
174     assert(current_file);

176     if(pre_event_buffer_offset + chunk_samples == pre_event_buffer_samples)
    {
178         fwrite(pre_event_buffer, sizeof(signed short),
            pre_event_buffer_samples - chunk_samples, current_file);
180     }
    else
182     {
        fwrite(pre_event_buffer + pre_event_buffer_offset + chunk_samples, sizeof(signed ↵
            short),
184         pre_event_buffer_samples - pre_event_buffer_offset - chunk_samples, current_file↵
            );
        fwrite(pre_event_buffer, sizeof(signed short), pre_event_buffer_offset, ↵
            current_file);
186     }
    }
188

void save_chunk(void)
190 {
    /*fprintf(stderr, "save: %ld\n", gnt++);*/
192     fwrite(pre_event_buffer + pre_event_buffer_offset,
        sizeof(signed short), chunk_samples, current_file);
194 }

void close_current_recording(void)
196 {
198     /*fprintf(stderr, "close: %ld\n", gnt++);*/
    fclose(current_file);
200 }

```

A.3 Calculate quality measures

This is how we implemented the quality measures calculations in GNU Octave:

Source Code A.3: `process.m` implements calculations of quality measures.

```

2      % Author: Thor Andreassen <ta@toggle.be>

4      tic();

6      clear all

8      fs = 500e3;
9      thres_percent = .10; % what percentage level should the signal be within
10     bw_thres_percent = .10;
11     delta_events = 500 * fs/1000000; % min time between events, in us
12     event_hist_sz = 15; % bin size in ms
13     around_event = 1 * fs/1000; % area around event to analys (in ms)
14     fftwinsz = 128;
15     fft_overlap = .80;
16     overlap = round(fft_overlap * fftwinsz);
17     filter_length = 100e-6; % in seconds
18     evnt_f_bins = 20e3; % 50 Hz per bin at 500kHz
19     hz_per_bin = fs / evnt_f_bins;
20     f_delta_bins = 300 / hz_per_bin; % in Hz

21     ev_len_limit.min = 0.100 * fs/1000; % in milliseconds
22     ev_len_limit.max = 40 * fs/1000; % in milliseconds

23     f_to_ignore = 17e3;
24     bw_per_fbin = fs/fftwinsz;
25     bins_to_ignore = ceil(f_to_ignore / bw_per_fbin);

26     req_bw_bin_hist = 5e3;
27     % find closest match to requested bin bw
28     bin_alternatives = [2^nextpow2(req_bw_bin_hist/2 / bw_per_fbin), ...
29                         2^nextpow2(req_bw_bin_hist / bw_per_fbin)];
30     [NA, fi] = min(bw_per_fbin * bin_alternatives - req_bw_bin_hist);
31     act_bw_bin_hist = bin_alternatives(fi);

32     gfilt = gaussian(round(fs * filter_length - 1), .1);
33     gfilt /= sum(gfilt); % normalise filter
34     hlfilt = floor(length(gfilt)/2); % used when padding signal to be filtered

35

36     % On which file
37     basedir = '/home/ta/mnt/halifax/ams-garage/recordings';

38     fname = "2011-06-23/sr_500000_ch_4_offset_000000000000147774000-00.s16";

39     % Support command line specified filename
40     if(length(argv()) >= 1)

```

```

    infile = argv(){1};
48 else
    infile = [basedir, "/", fname];
50 endif

52 [dirname, basename] = fileparts(infile);

54 st = stat(infile);

56 % base is +- 2**15 (32784), but samples are only 14 bits ie. +- 8192
% +- 8192 are max/min which is +- 10V, so samples should be divided by 819.2
58 conv_to_volt = (2**15 / 4) / 10;
conv_to_norm = (2**15 / 4);

60

62 [rid, err] = fopen(infile);

64 if(rid == -1)
    err
    return
66 endif

68 printf("%s - Starting processing of %s \n", strftime("%F %T", localtime(time())), ←
    infile);

70 % Read channel 1 samples into memory:
y = fread(rid, [4 Inf], 'int16')(1,:);
72 fclose(rid);

74 y /= conv_to_norm;

76 dcshift = sum(y) / length(y);
y -= dcshift;

78

ysq = y.^2;
80 ysq = filter(gfilt, 1, [ysq; zeros(hlfilt, 1)](hlfilt+1:end, :));
sysq = sort(ysq);

82

thres = sysq(end - floor(thres_percent * length(sysq)),:);

84

% Where does y-squared exceed thres
86 sighigh = find(ysq > thres);

88 event_offsets = find(diff(sighigh) > delta_events);

90 % Create events vector which is a two columns vector, first columns contains
% starting offsets into y and second column contains the end offset
92 evnts = sighigh([[1; event_offsets+1], [event_offsets; length(sighigh)]]);

94 % Trim of events shorter than shortest_evt
evnts = evnts([evnts(:,2) - evnts(:,1)] >= ev_len_limit.min, :);

```

```

96 % Grow events to include "delta_events" samples before and after an event and
97 % ensure limits are not exceeded
98 evnts = [evnts(:,1) - delta_events, evnts(:,2) + delta_events];
100 if(evnts(1,1) < 1) evnts(1,1) = 1; endif
101 if(evnts(end,2) > length(y)) evnts(end,2) = length(y); endif
102
103 % Bugfix for rotated matrices
104 if(all(size(evnts) == [2,1])) evnts = evnts'; endif
105
106 % Samples not covered by events
107 background = [[1;evnts(:,2)], [evnts(:,1); length(y)]];
108
109 evnt_rngs = arrayfun(@(x, y) x:y, evnts(:,1), evnts(:,2), 'UniformOutput', false);
110 bg_rngs = arrayfun(@(x, y) x:y, background(:,1), background(:,2),
111                    'UniformOutput', false);
112
113 len_msecs = length(y) / (fs/1000);
114 len_msecs_evnts = sum((evnts(:,2) - evnts(:,1)) / 2) / (fs/1000);
115 len_msecs_bg = len_msecs - len_msecs_evnts;
116
117 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
118 %%
119 %% Output file data
120 %%
121 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
122
123 data_fid = fopen(strrep(infile, ".s16", ".data2"), "w");
124
125 fprintf(data_fid, "%-30s : %s\n", "File name"
126                , infile);
127 fprintf(data_fid, "%-30s : %.2f\n", "File size (MiB)"
128                , st.size / 1024**2);
129 fprintf(data_fid, "%-30s : %s\n", "Recorded at"
130                , strftime("%F %T", localtime(st.mtime)));
131 fprintf(data_fid, "%-30s : %d\n", "Seconds since the epoch"
132                , st.mtime);
133 fprintf(data_fid, "%-30s : %.2f\n", "Length (s)"
134                , length(y) / fs);
135 fprintf(data_fid, "%-30s : %.2f\n", "Number of events"
136                , length(evnts));
137 fprintf(data_fid, "%-30s : %.2f\n", "Portion of total (%)"
138                , 100 * length([evnt_rngs{:}]) / length(y));
139 fprintf(data_fid, "%-30s : %g\n", "Power rate (1/ms)"
140                , sum(ysq) / len_msecs);
141 fprintf(data_fid, "%-30s : %g\n", "Event power rate (1/ms)"
142                , sum(ysq([evnt_rngs{:}], 1)) / len_msecs_evnts);
143 fprintf(data_fid, "%-30s : %g\n", "Background power rate (1/ms)"
144                , sum(ysq([bg_rngs{:}], 1)) / len_msecs_bg);

```

```

146 fclose(data_fid);

148 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
150 %% Calculate quality measures for each event
152 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

154 events_fid = fopen(strrep(infile, ".s16", ".events2"), "w");

156 for event = 1:length(evnts)
    qm(event).event_nr = event;
158     qm(event).offset_samp = evnts(event,1);
    qm(event).dur_ms = (evnts(event, 2) - evnts(event, 1)) / (fs/1000);
160     qm(event).dur_samp = evnts(event, 2) - evnts(event, 1);
    qm(event).energy_ms = sum(ysq(evnts(event,1):evnts(event,2))) / qm(event).dur_ms;
162 
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
164     % Bandwidth calculations
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

166     % First calculate the spectrum with a constant number of bins, note also that
168     % we only use the magnitude and remove the mirrored part
    Y_e = abs(fft( y(evnts(event,1):evnts(event,2),1), evtnt_f_bins ))(1:end/2);

170     % Ignoring low frequencies in the following, everything below f_to_ignore is set to 0
172     % i.e. fs / 2 divided by length(Y), we then divide f_to_ignore with the result or:
    Y_ignore = floor(2 * length(Y_e) * f_to_ignore / fs);
174     Y_e(1:Y_ignore) = 0;

176     % Bandwidth is determined by the widest cluster of above threshold bins. This is done
178     % in a similar manner to the temporal calculations in the signal detection part above.
    bw_threshold = sort(Y_e)(round(length(Y_e) * (1-bw_thres_percent)));
    f_ev_high = find(Y_e > bw_threshold);
180     f_ev_offsets = find(diff(f_ev_high) > f_delta_bins);
    f_clusters = f_ev_high([1; f_ev_offsets+1],
182                             [f_ev_offsets; length(f_ev_high)]));

184     % Bugfix for rotated matrices
    if(all(size(f_clusters) == [2,1])) f_clusters = f_clusters'; endif

186     [NOT_USED, largest_cluster_i] = max(f_clusters(:,2) - f_clusters(:,1));

188     qm(event).f_min = f_clusters(largest_cluster_i, 1) * hz_per_bin;
190     qm(event).f_max = f_clusters(largest_cluster_i, 2) * hz_per_bin;
    qm(event).bw = qm(event).f_max - qm(event).f_min;

192 
    %%%%%%%%%%%%%%%%%%%%%%%%%%%
194     % Entropy
    %%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```
196 N = floor(2 * req_bw_bin_hist * length(Y_e) / fs);  
    P_f = sum(reshape(Y_e, N, [])) / sum(Y_e);  
198 P_f = P_f(P_f ~= 0);  
  
200 % normalized entropy, divide by log10(N)  
    qm(event).E_f = -sum(P_f .* log10(P_f)) / log10(N);  
202  
    qm(event).std_f = std(Y_e);  
204  
    fdisp(events_fid, qm(event));  
206 endfor  
  
208 fclose(events_fid);  
  
210 save(strrep(infile, '.s16', '.qm_array_2.oct'), 'qm')  
  
212 printf("Elapsed time is %.2f second(s)\n", toc());
```

B

Reliability tests

*Mathematics has no symbols for
confused ideas.*

George Stigler

B.1 Variable gain setting

Ali Shekarchi tested to see if the variable gain is frequency or amplitude independent. The results of these tests can be seen in Figures B.1 to B.3.

B.2 Knowles microphone calibration

Ali Shekarchi has used the calibration setup (Section 4.4 on page 40) to calibrate most of our Knowles microphones with the method described. Some of them were calibrated with extension cables attached to test their effect on the whole system.

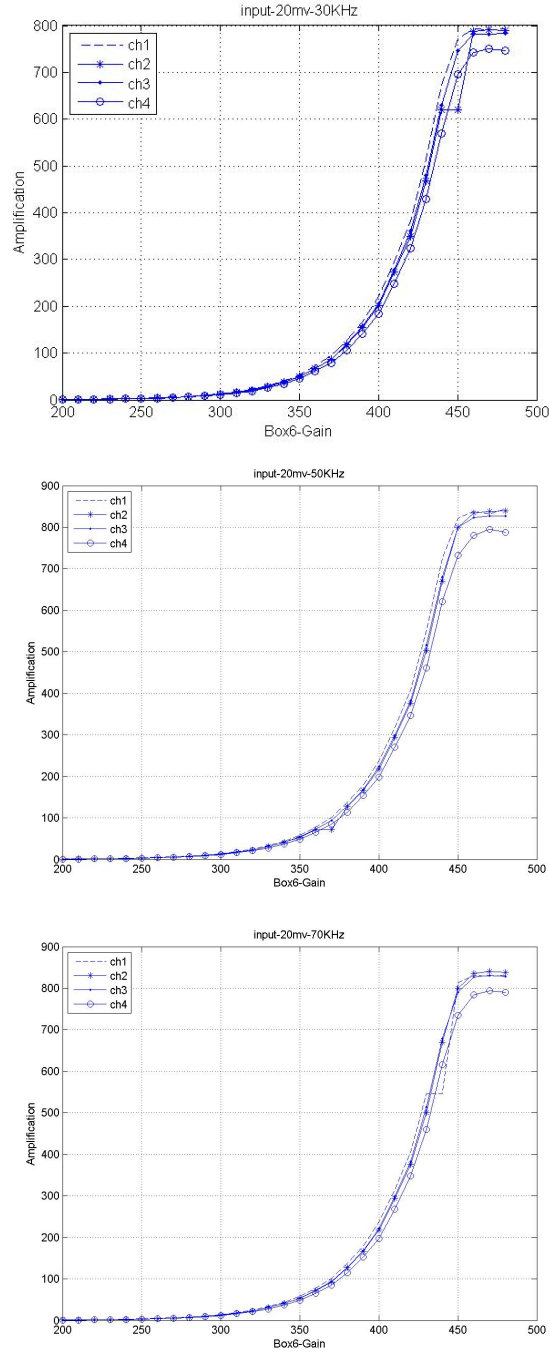


Figure B.1: Variable amplification test of box6 – 20 mV at 30, 50, 70 kHz.

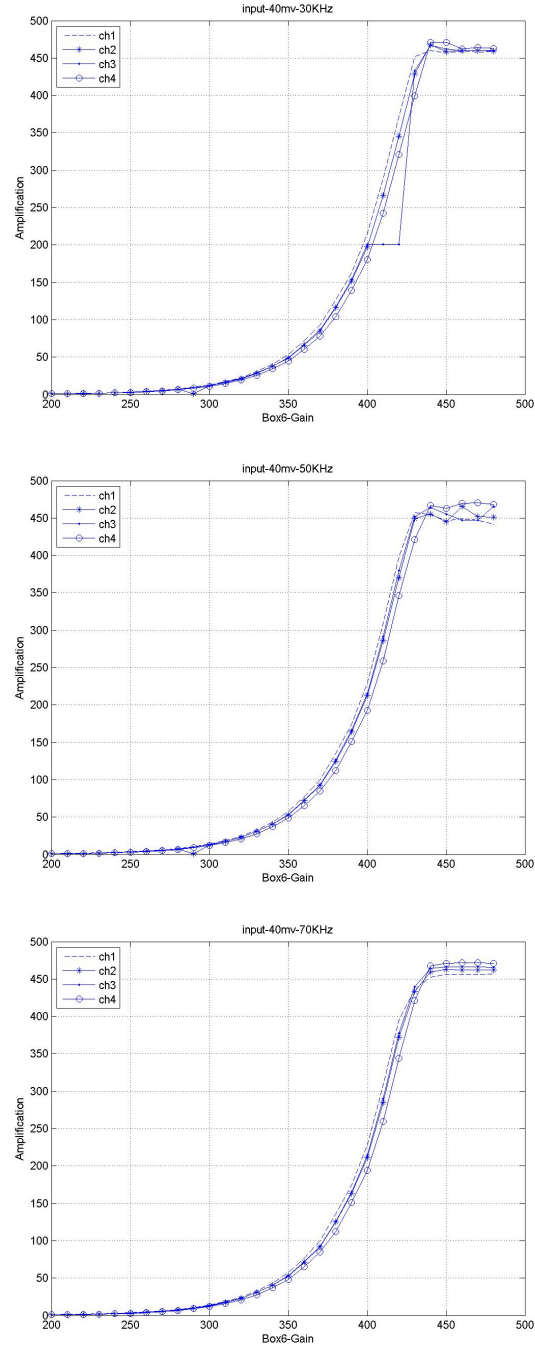


Figure B.2: Variable amplification test of box6 – 40 mV at 30, 50, 70 kHz.

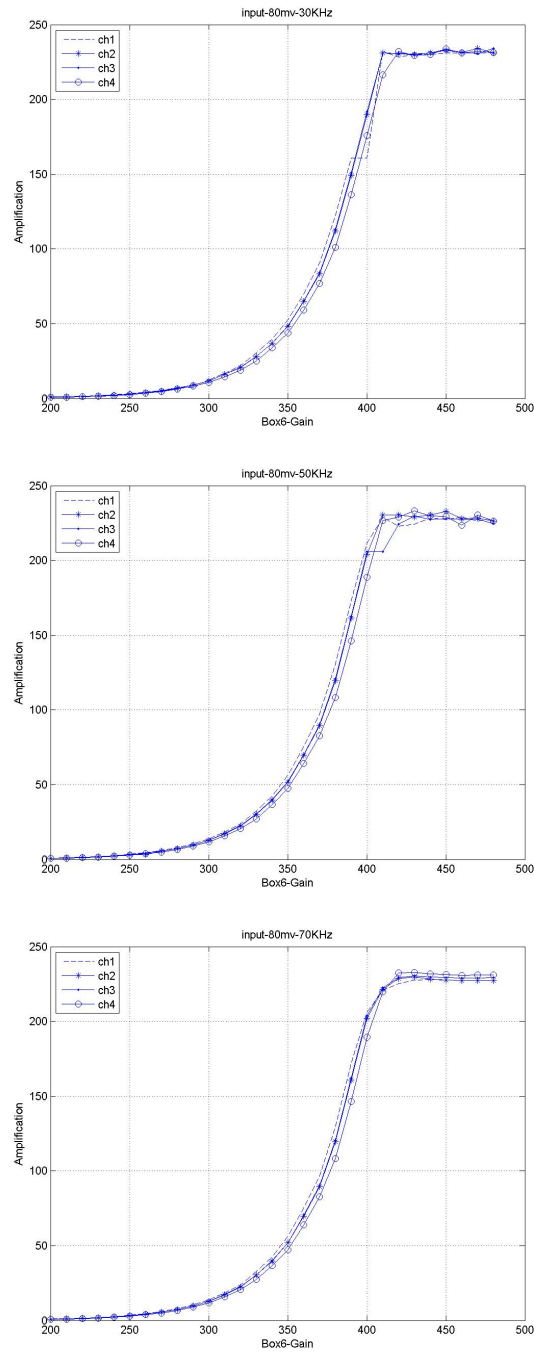


Figure B.3: Variable amplification test of box6 – 80 mV at 30 kHz, 50 kHz, and 70 kHz.

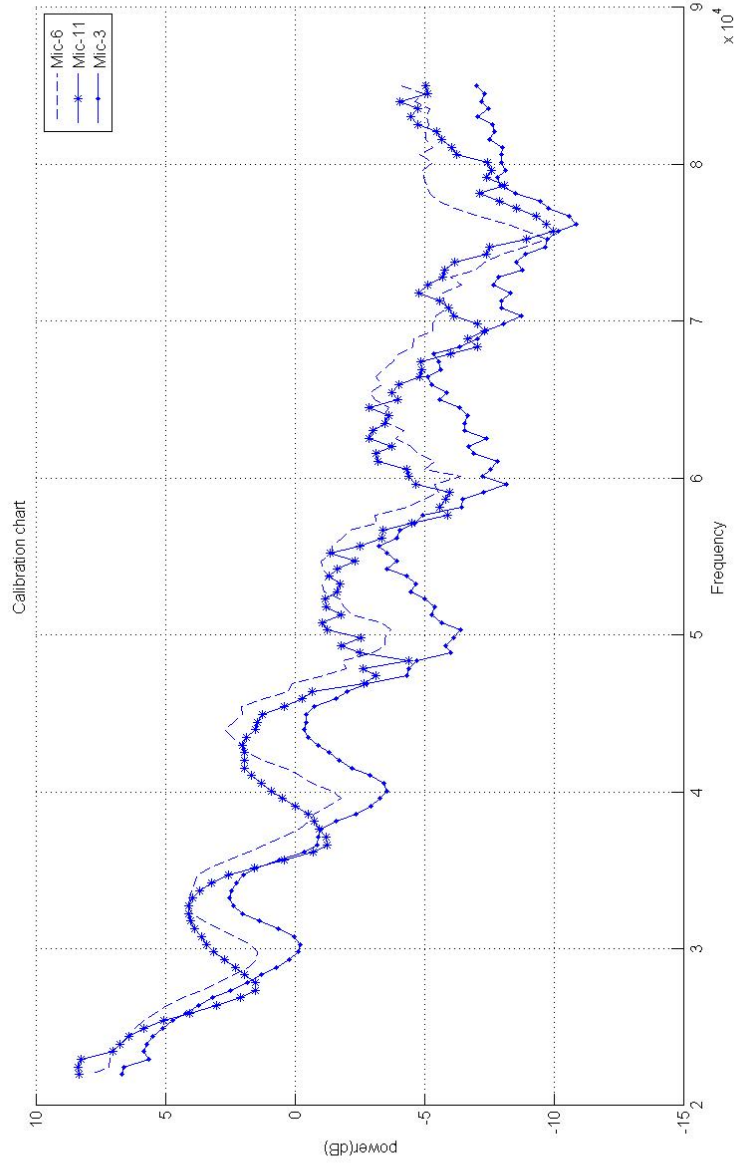


Figure B.4: Transfer functions of microphones 3, 6, and 11.

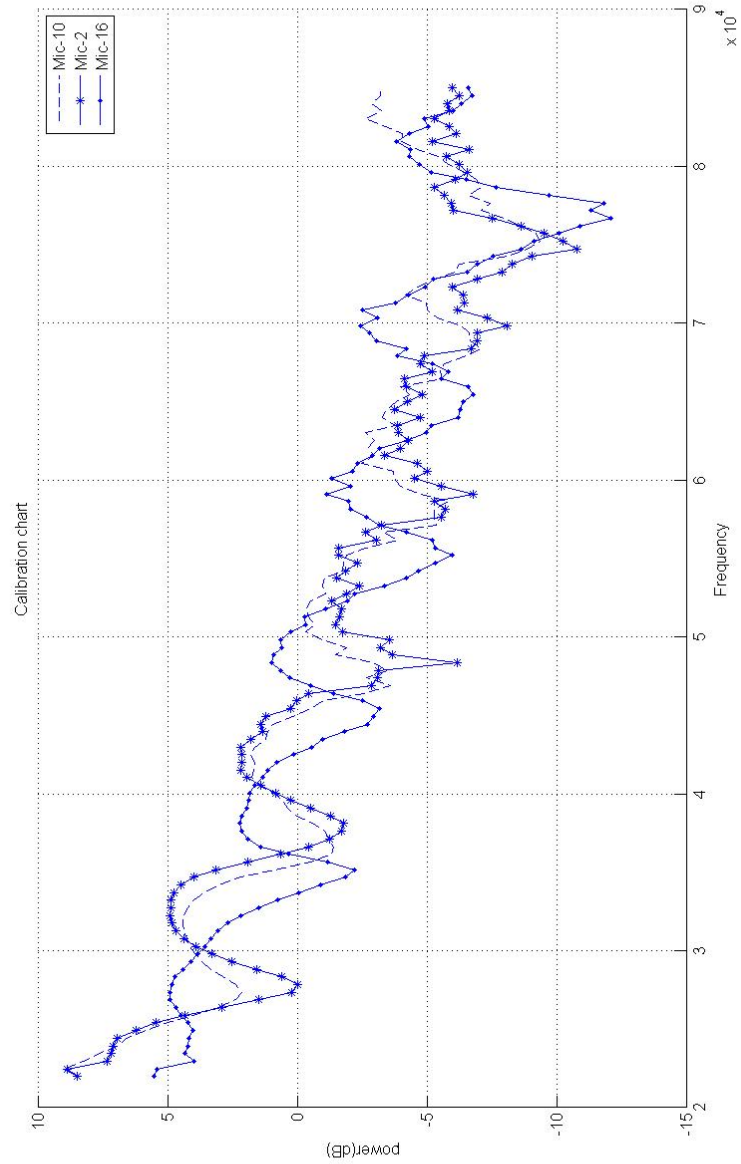


Figure B.5: Transfer functions of microphones 2, 10, and 16.

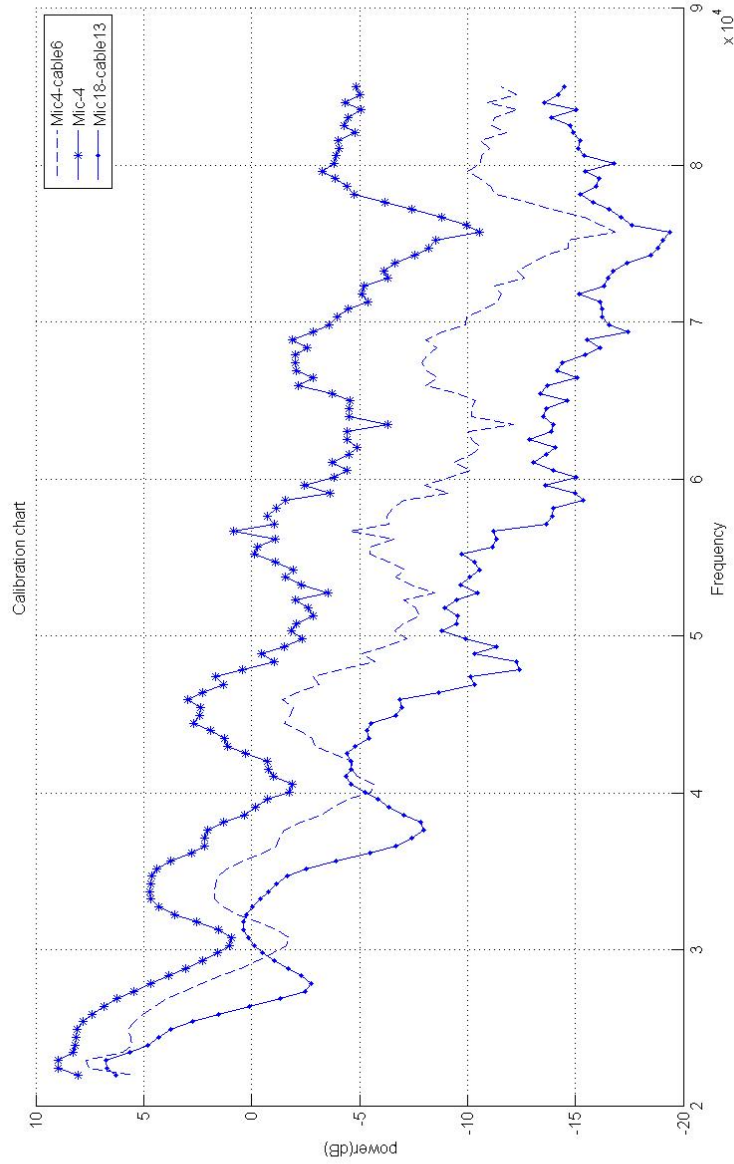


Figure B.6: Transfer functions of microphone 4, microphone 4 with extension cable 6, and microphone 18 with extension cable 13.

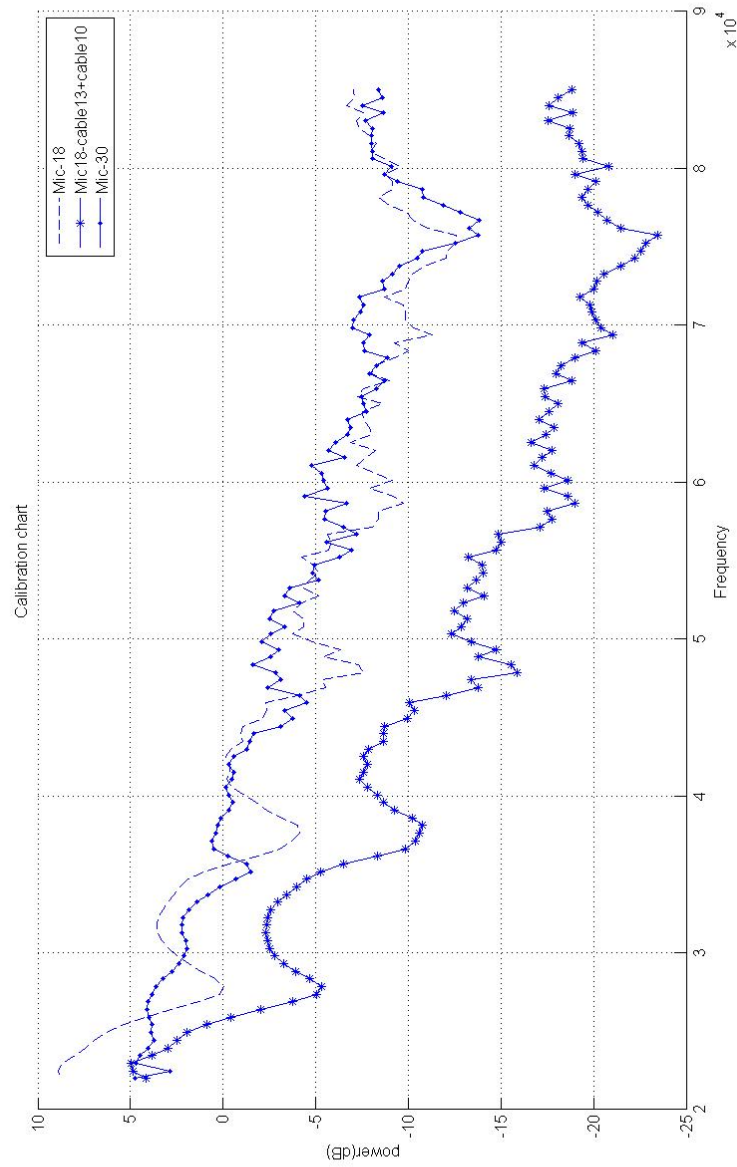


Figure B.7: Transfer functions of microphones 18 and 30 and microphone 18 with extension cable 10 and 13.

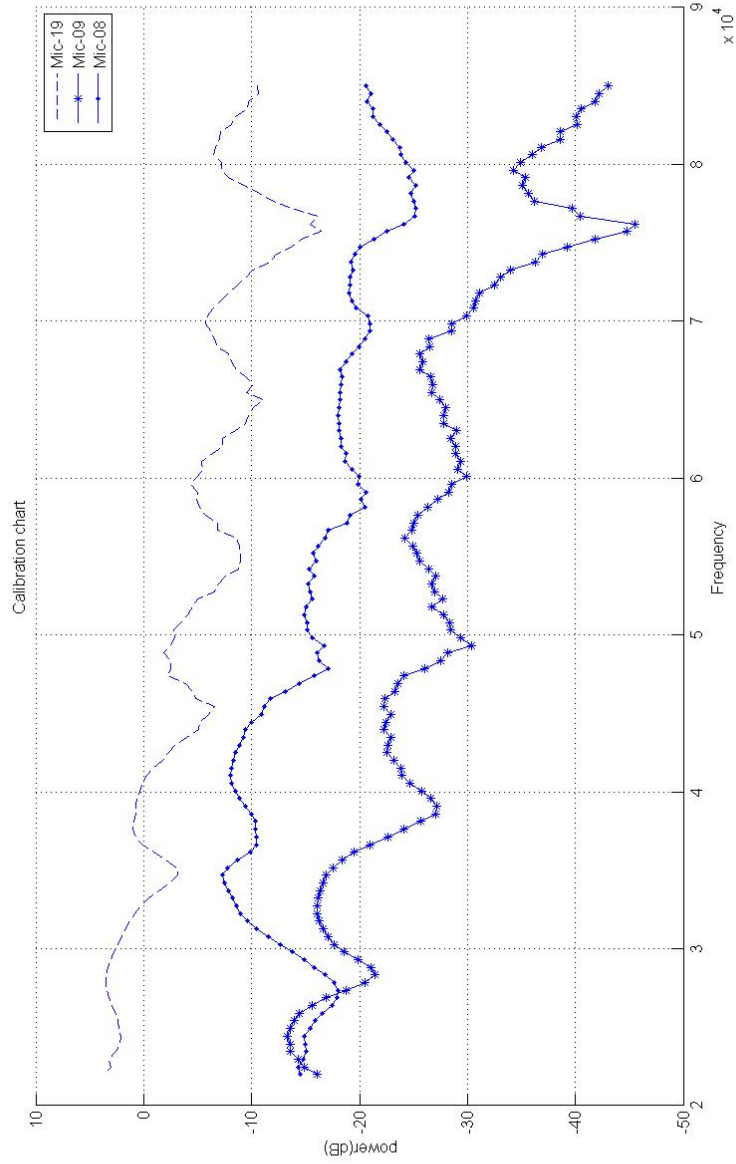


Figure B.8: Transfer functions of microphones 8, 9, and 19.

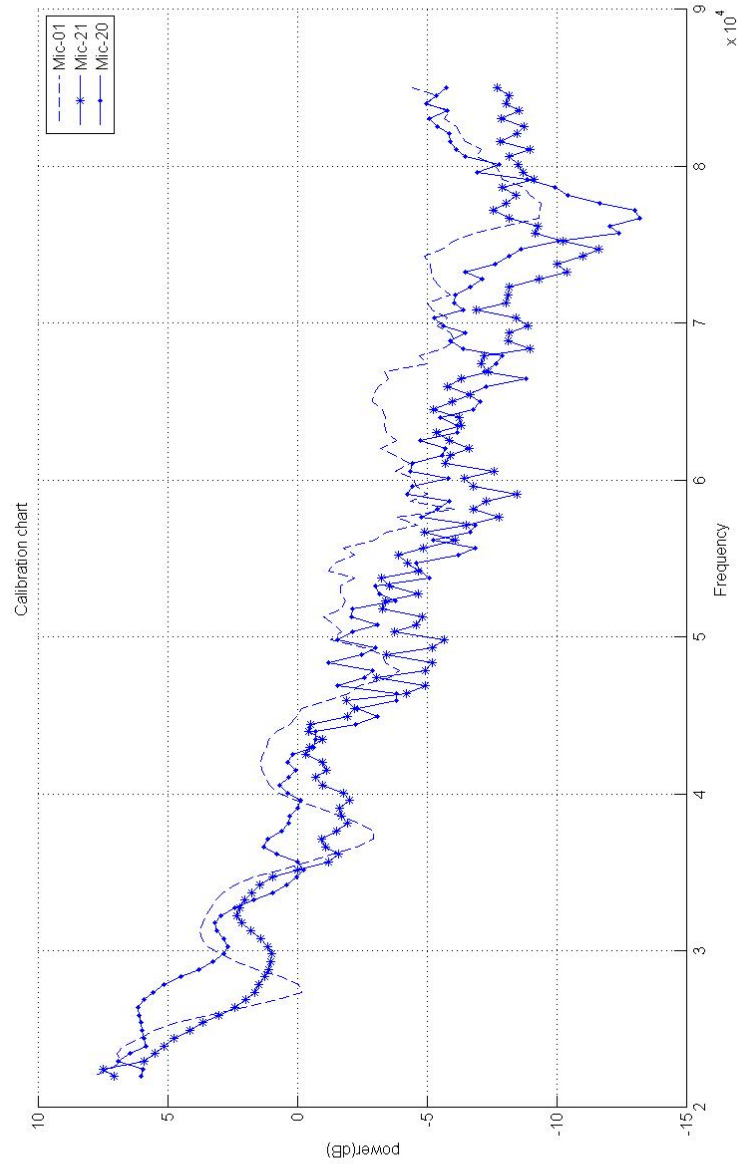


Figure B.9: Transfer functions of microphones 1, 20, and 21.

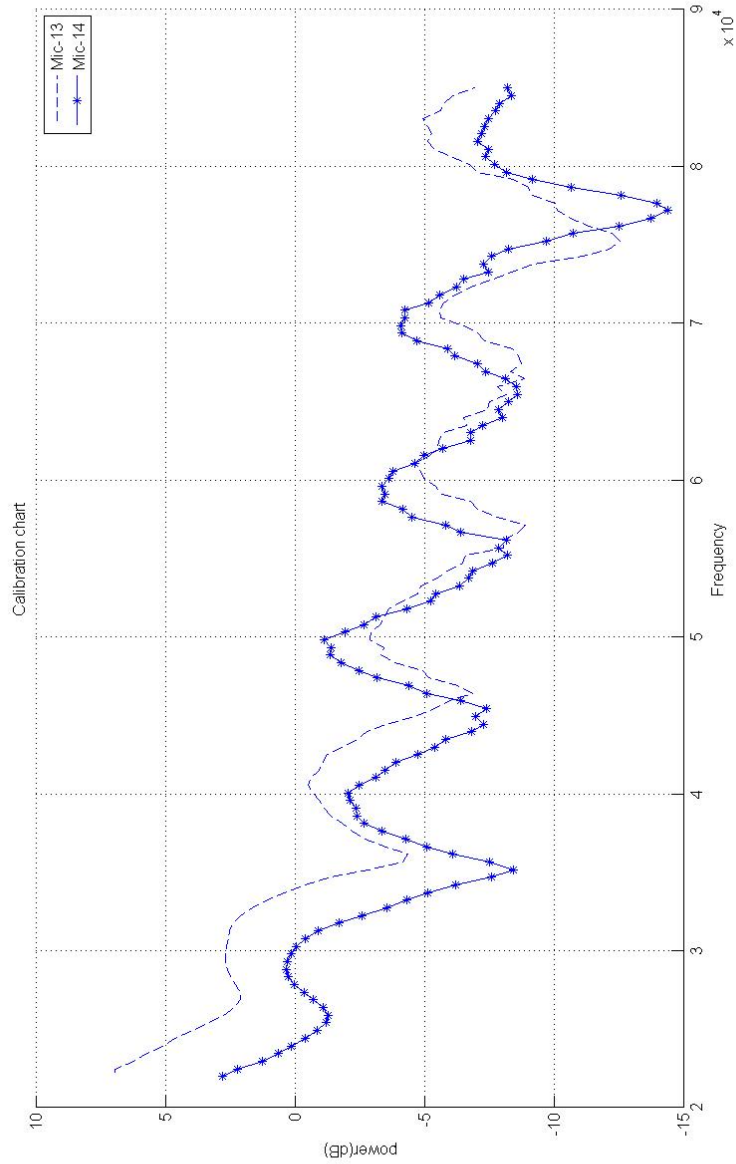


Figure B.10: Transfer functions of microphones 13 and 14.

Recipes for interesting challenges

*Walking on water and developing
software from a specification are easy
if both are frozen.*

Edward V. Berard

C.1 Dealing with streamed data

Visualizing a stream

When dealing with streams, discovering what they contain is not a trivial matter. Thus, algorithms are needed that can extract and present the data to the user. Depending on the bandwidth of the stream, which depends on the sample-rate, bit depth and number of channels, these algorithms need to compress the data in some way. Two specific open source solutions for compressing and displaying data streams have been explored during this project, namely **GNU Radio** and **SoX** effects.

Traditionally visualization of data streams is done with spectrograms which is a three-dimensional plot often with time on the abscissa, frequency on the ordinate and signal energy as a color-map in the x - y plane.

GNU Radio

GNU Radio was developed to provide signal processing blocks for software radios, originally those sold by *Ettus Research*. The challenges faced by software radios are similar and even more demanding than those of ultrasonic signal processing, and thus many solutions found there are directly applicable to our field. Figure C.1 shows an example of what a **GNU Radio** application looks like. This screenshot shows a “waterfall” view, which is a realtime updating spectrogram.

This software package is built as loosely related building blocks which fit together, where each block doesn’t need to know the contents of other blocks, there are limiter blocks, frequency and time domain plotting blocks,

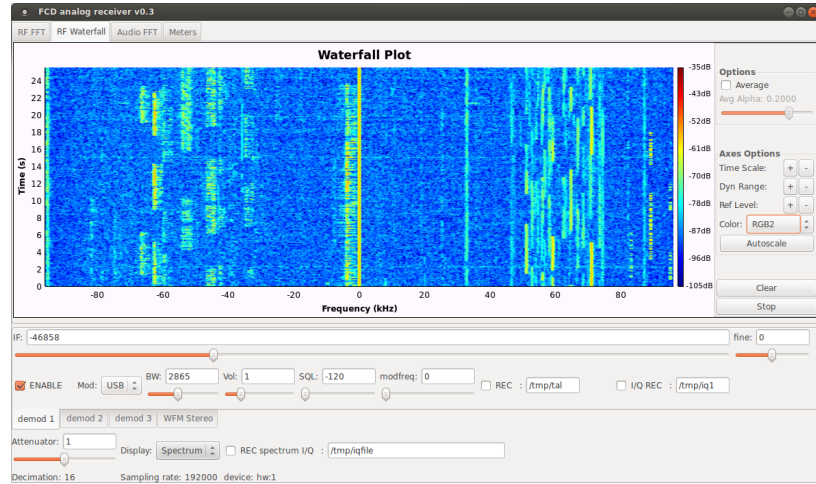


Figure C.1: GNU Radio spectrogram waterfall screenshot.

spectrogram plotting blocks, multiplexer-, demultiplexer blocks, and many other signal processing/filter blocks, which can be connected in sequence or parallel or a mix. GNU Radio's blocks are written in Python. Each block is setup to receive chunks of data, process, it and pass on to the next block, much like the pipe structure discussed in Section 5.2 on page 59. Using such high-level programming language allows for speedy development and ease of maintenance. Although this approach has not been fully integrated, we anticipate that it will be part of the finished user interface. An example implementation of a waterfall producing block program can be seen in Source Listing C.1.

Source Code C.1: GNU Radio block program that implements a spectrogram waterfall view.

```
#!/usr/bin/env python
2 #####
# Gnuradio Python Flow Graph
4 # Title: Top Block
# Generated: Wed May 9 17:59:58 2012
6 #####
8 from gnuradio import eng_notation
```

```

from gnuradio import gr
10 from gnuradio import window
from gnuradio.eng_option import eng_option
12 from gnuradio.gr import firdes
from gnuradio.wxgui import waterfallsink2
14 from grc_gnuradio import blks2 as grc_blks2
from grc_gnuradio import wxgui as grc_wxgui
16 from optparse import OptionParser
import wx

18

class top_block(grc_wxgui.top_block_gui):
20

    def __init__(self):
22        grc_wxgui.top_block_gui.__init__(self, title="Top Block")
        _icon_path = "/usr/share/icons/hicolor/32x32/apps/gnuradio-grc.png"
24        self.SetIcon(wx.Icon(_icon_path, wx.BITMAP_TYPE_ANY))

26        #####
        # Variables
28        #####
        self.samp_rate = samp_rate = 250000

30

        #####
        # Blocks
32        #####
34        self.wxgui_waterfallsink2_0 = waterfallsink2.waterfall_sink_f(
            self.GetWin(),
            baseband_freq=0,
            dynamic_range=100,
38            ref_level=0,
            ref_scale=2.0,
            sample_rate=samp_rate,
40            fft_size=1024,
            fft_rate=20,
            average=False,
42            avg_alpha=None,
            title="Waterfall Plot",
            win=window.flattop,
46        )

48        self.Add(self.wxgui_waterfallsink2_0.win)
        self.gr_throttle_0 = gr.throttle(gr.sizeof_float*1, samp_rate)
50        self.blks2_tcp_source_0 = grc_blks2.tcp_source(
            itemsize=gr.sizeof_float*1,
            addr="127.0.0.1",
            port=1234,
52            server=True,
54        )

56

        #####
58        # Connections

```

```

#####
60 self.connect((self.blks2_tcp_source_0, 0), (self.gr_throttle_0, 0))
   self.connect((self.gr_throttle_0, 0), (self.wxgui_waterfallsink2_0, 0))
62
   def get_samp_rate(self):
64       return self.samp_rate

   def set_samp_rate(self, samp_rate):
66       self.samp_rate = samp_rate
68       self.wxgui_waterfallsink2_0.set_sample_rate(self.samp_rate)

70 if __name__ == '__main__':
   parser = OptionParser(option_class=eng_option, usage="%prog: [options]")
72   (options, args) = parser.parse_args()
   tb = top_block()
74   tb.Run(True)

```

SoX

SoX (SoX developers, 2012) is a general purpose stream processor which can be adapted to handle and compress infinite streams. It has similar qualities as GNU Radio but is written in a lower level language and thus tends to be more efficient but harder to write. Implementing a visualization application at this level would allow us to handle more channels simultaneously, and enable us to use lower end hardware on the nodes.

Stream handler primitives

A pipeline is built up of primitives, among others there are: multiplexers, demultiplexers, forwarders, amplifiers, filters, and channel selectors. See al (URL provided, accessed: 2013) for more examples.

Multiplexer

Here is one way to multiplex channels with SoX. For example if two files, `in1.wav` and `in2.wav`

```

2 Input File : 'in1.wav'
   Channels : 4
   Sample Rate : 500000

```

```
4 Precision : 16-bit
   Duration : 00:00:14.00 = 31 samples ~ 2.325 CDDA sectors
6 File Size : 300 MiB
   Bit Rate : 27.4k
8 Sample Encoding: 16-bit Signed Integer PCM

10 Input File : 'in2.wav'
   Channels : 4
12 Sample Rate : 500000
   Precision : 16-bit
14 Duration : 00:00:14.00 = 31 samples ~ 2.325 CDDA sectors
   File Size : 300 MiB
16 Bit Rate : 27.4k
   Sample Encoding: 16-bit Signed Integer PCM
```

with four channels each are to be multiplexed, something like the listing in Source Listing C.2 would work.

```
sox -M in1.wav in2.wav -t wavpcm out.wav
```

Source Code C.2: Multiplex files *in1.wav* and *in2.wav* and save them to *out.wav* with SoX.

Here *out.wav* will contain all eight channels from *in1.wav* and *in2.wav*. The explicit mention of the Wave PCM file variant here (*-t wavpcm*) is needed to make the resulting wav file compatible with GNU Octave and Matlab as these programs only support the Microsoft Wave PCM format (Microsoft, 1991).

If the input files actually were stream the command would still be much the same. In Source Listing C.3 we read from named pipes which are a way of connecting multiple pipes through the file system. This is a virtual filesystem handled transparently by the operating system kernel and thus only operates in main memory. To SoX the streams can actual be treated as non-seekable files.

```
sox -M -t wavpcm /tmp/fifo1 -t wavpcm /dev/fifo2 out.wav
```

Source Code C.3: Multiplex streams from two named pipes *fifo1* and *fifo2*.

Demultiplexing

Demultiplexing can be done with the **SoX** and **tee** program from **coreutils** (GNU coreutils developers, 2012), an open source software package from the **GNU** organization. See Source Listing C.4 for an example of how to split a two channel file and pass each channel on to two separate instances of a second program.

```
cmd1 | tee >(sox -t wavpcm - - remix 1 | cmd2) \
2      >(sox -t wavpcm - - remix 2 | cmd2)
```

Source Code C.4: Demultiplex the stream from *cmd1* with two **SoX** subprocesses started by **tee**. Each channel is then processed by separate instances of *cmd2*.

Forwarders

It is possible to send streams on to a different computer, perhaps running *some_program* on the stream:

```
adac |
2  signal_detect |
    ssh other_computer some_program
```

Source Code C.5: Example of how to forward a stream.

Amplifier

Amplifying a stream is straight forward with **SoX**:

```
adac |  
2  sox -c4 -r500e3 -t s16 - out.wav gain +6
```

Source Code C.6: Example of how to amplify a 4 channel stream.

Filter

Here is how to apply a bandpass filter with cut-offs at 10 kHz and 110 kHz to an input stream with SoX:

```
adac |  
2  sox -c4 -r500e3 -t s16 - out.wav sinc 10k-110k
```

Source Code C.7: Example of how to bandpass filter a 4 channel stream.

Channel selector

If you want to only save the first channel of an input stream with 4 channels, you can do it like this with SoX:

```
adac |  
2  sox -c4 -r500e3 -t s16 - out.wav remix 1
```

Source Code C.8: Example of how to select a subset of channels from a stream.

Conditional stream processing

It is also possible to make conditional branches in stream processing. The following example will only execute `cmd3` if the `DEFINED` variable was set:

```
adac |  
2 cmd2 |  
  if [[ -n "$DEFINED" ]]; then cmd3; else cat; fi |  
4 cmd4 |  
  cmd5 |  
6 cmd6 |  
  (...) |  
8 cmdN > result.txt
```

Source Code C.9: Example of conditional pipelines.

C.2 Parallel calculations

Many of the calculations done during this project, have been parallelized by GNU `parallel` (Tange, 2011). This program allows us to move data and commands to multiple hosts, do the calculations, and automatically fetch the results.

Calibrated emission source code

First, solve the problem. Then, write the code.

John Johnson

D.1 Source code for aligning and averaging recordings

When trying to average identical recordings in practice, it is almost always necessary to align them first, as the results otherwise might be misleading. The strategy here is to find the maximum of the cross correlation between the first signal and any subsequent recordings, adjust the offset between the signals, and add them together. This has proven to work fairly well with very similar signals. This idea has been implemented in `matlab/octave` source code in Source Listing D.1 on the next page.

When $x * h$ has been found by aligning and averaging, we are ready to, first, recover the output chains transfer function H , and second, to create a deconvolved sweep by dividing X by H , see Source Listing D.2. `transfer_function.wav` can now be used to adapt future signals to be emitted in a calibrated manner. We can test the procedure by emitting `deconvolved_sweep.wav` which should come out with equal energy at all frequencies. When we tested it, this was indeed the results (Section 4.6 on page 46).

As it often is when dealing with practical recordings, these files originally contained five channels and five seconds of data, seeing as the actual emissions we used were 20 ms long, this a bit of a waste. We know that the signal is at circa 2.505 seconds into the recording so in order to save storage and make computation more efficient it makes sense to extract the relevant part of the recordings. This is fairly straight forward with `SoX` and `zsh` scripting, as demonstrated in Source Listing D.3 on page 151.

```

% Move to directory with the signal files
2 oldpwd = pwd();
signal_dir = '../data/speaker_calibration/even_sweep_emissions';
4 cd(signal_dir);

6 % Generate list of signal files
signal_files = dir('T0000*');

8 % Load first file as reference signal
10 [r, fs] = wavread(signal_files(1).name);

12 signal_sum = r;

14 % Iterate over the rest of the signals
for i=2:length(signal_files)
16     [sig, fs] = wavread(signal_files(i).name);

18     % Find offset between reference signal and sig
    [~, off] = max(xcorr(r', sig'));
20     off = length(r) - off;

22     % Pad signal accordingly
    if(off < 0)
24         sig = [zeros(1, -off), sig(1:end+off)'];
    elseif (off > 0)
26         sig = [sig(off:end)', zeros(1, off-1)];
    end

28     % Accumulate signal sum
30     signal_sum = signal_sum + sig';
end

32 % Average
34 xh = signal_sum/length(signal_files);

36 % Normalize x convolved with h
xh = signal_avg / max(signal_avg);

38 cd(oldpwd)

```

Source Code D.1: Sum and average.

```

2  % Load original file that was emitted
  x = wavread('logsweep.wav');

4  % Transform x into frequency domain and determine H (emission chain transfer function)
  X = fft(x);
6  H = fft(xh) ./ X;

8  % Calculate h, normalize and save it
  h = real(ifft(H));
10 h = h / max(h);
   wavwrite(h, "transfer_function.wav");

12 % Vector indices to choose only frequencies of interest
14 starti = 20e3 / 50;
   endi = 85e3 / 50;
16 rng = starti:endi;
   irng = (length(x) - endi) : (length(x) - starti);

18 % Zero out unwanted frequencies
20 X = [zeros(1, starti - 1), X( rng)', zeros(1, length(X)/2 - endi) ...
      zeros(1, length(X)/2 - endi), X(irng)', zeros(1, starti - 1)]];

22 % Deconvolve x with h
24 X_deconv_H = X ./ H;

26 % Transform X/H to time domain and normalise
   x_deconv_h = real(ifft(X_deconv_H));
28 x_deconv_h = x_deconv_h / max(x_deconv_h);

30 % Save the deconvolved sweep - which can be used to adjust future emission files
   wavwrite(x_deconv_h, fs, 'deconvolved_sweep.wav');

```

Source Code D.2: Source code for finding x deconvolved with h .

```

2  for sig in T0000*; do
     sox $sig -t wavpcm $sig:r_short.wav trim 2.5 .040
  done

```

Source Code D.3: Trimming many files with SoX.

```
function[yt] = apply_tf(y, tf)
2   Y = fft(y);
   TF = fft(tf, length(Y));
4
   YT = Y ./ TF;
6
   yt = real(ifft(YT));
```

Source Code D.4: Apply a transfer function to a signal.

D.2 Apply transfer function

When the transfer function has been recovered, it is straight forward to reverse its effects. The basic function we have used can be seen in Source Listing D.4 which is based on Algorithm 4.2 on page 49.

Robot control

*Premature optimization is the root of
all evil.*

Donald E. Knuth

This section will only deal with some the more interesting parts of the development of the control software, for the full source code see Andreassen (URL provided, accessed: Jun 2011).

E.1 Control programs

The robots were ultimately controlled by a signal analysis computer, usually a lab laptop running `Matlab`. The `Matlab` program sends its instructions through `ssh` to the node computer, which then mediates the motions to the mobile robot and/or the robot arm. The robot arm is connected directly to the node computer through a serial port while the mobile robot has its own on-board computer which takes care of the motions, this results in a need for multiple programs interacting with each other.

The main program is `natasha` (see Sources E.2 and E.3), it takes motion commands both for the mobile robot and the robot arm. Controlling the mobile robot is pretty straight forward as it is supported by the Player/Stage framework (Various, URL provided, accessed: Sep 2012[d]), as documented by `pioneer.cpp` in Source Listing E.4 on page 158. The arm, however, was more of a challenge as the company that developed the arm has ceased to exist, and so the library control software (Katana Native Interface or KNI) is no longer maintained. This results in the need for old compilers and backporting various tools. See Proprietary vs. open source solutions Section 7.4 on page 103 for a discussion on these issues. The dependence on old software needed a customized makefile to build source code with the driver. This customized file is listed in Source Listing E.1. Notice the compiler version and also the need to statically link archive files with the compiled program.

```

#
2 # Author: Thor Andreassen <ta@toggle.be>
#
4
6 TARGET = natasha
INSTALL_DIR = /usr/local/bin

8 CXX = g++-3.3
CXX_FILES = $(TARGET).cpp utils.cpp
10 OBJS = $(CXX_FILES:.cpp=.o)
KNI_AR = $(wildcard ../../neuronics/*KNI*.a)
12 CC = g++-3.3
CXXFLAGS = -Wall -g -I../../include/neuronics
14 LDFLAGS = -lm -lrt -lpthread

16 .PHONY: all install run clean

18 all: $(TARGET)

20 $(TARGET): $(OBJS)
    $(CXX) $(LDFLAGS) $(OBJS) -o $@ $(KNI_AR)
22
install:
24     sudo ln -sf $$PWD/$(TARGET) /usr/local/bin

26 run:
    $(TARGET) -x 0.2 -t 15
28
clean:
30     $(RM) $(OBJS)

```

Source Code E.1: Makefile to compile source code against Katana Native Interface library.

Source Code E.2: Main robot control program including arm robot interaction – header file.

```

/*
2  * Author: Thor Andreassen <ta@toggle.be>
  * Created: 2011-02-16
4  */

6 #ifndef __KATANA_H__
  #define __KATANA_H__

8

10 #include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
12 #include <unistd.h>
#include <libgen.h>

```

```

14  #include <iostream>
16  #include <cstdio>
18  #include <memory>
20  #include <vector>
22  #include <fstream>
24  #include <pthread.h>

26  /* KNI specific headers */
28  #include <kniBase.h>
30  #include <KNI/kmlMotBase.h>
32  #include <common/Timer.h>

34  #define KATANA_SERIAL_PORT 0
36  #define KATANA_CONFIG_FILE "/etc/katana6M.cfg"

38  #define KATANA_NOCHANGE 0
40  #define KATANA_LOW 1
42  #define KATANA_HIGH 2

44  typedef struct {
46      char *programe;
48      char *position_file;
50      std::vector<int> pos;
52      int speed;
54      bool calibrate;
56      double turn;
58      double len;
60      int azimuth;
62      int elevation;
64      int mode;
66   } options_t;

68  std::ostream& operator<<(std::ostream& out, const std::vector<int>& enc);
70  options_t *parse_options(options_t *opts, int argc, char **argv);
72  void usage(options_t *opts);
74  std::auto_ptr<CLMBase>& init_robot(std::auto_ptr<CLMBase>&);

76  #endif /* ifndef __KATANA_H__ */

```

Source Code E.3: Main robot control program including arm robot interaction – source file.

```

/*
2  * Author: Thor Andreassen <ta@toggle.be>
4  */

```

```

#include "natasha.h"
6 #include <sstream>

8 #define P3_CMD "ssh -i ~/.ssh/id_chirp_rsa chirp@192.168.100.99 screen -S robot -X ↵
    stuff ";

10 using namespace std;

12 auto_ptr<CLMBase> katana;

14 int main(int argc, char *argv[]) {
    options_t opts;
16 parse_options(&opts, argc, argv);
    string base_cmd = P3_CMD;
18 string is_moving = "ssh -i ~/.ssh/id_chirp_rsa chirp@192.168.100.99 test -f /tmp/↵
    vehicle_is_moving";

20 try {
    init_robot(katana);
22 if(opts.calibrate)
        katana->calibrate(true);
24 katana->setRobotVelocityLimit(opts.speed);
    katana->setRobotAccelerationLimit(1);
26 int limit = 100;
    for(short motorNumber = 0; motorNumber < katana->getNumberOfMotors() - 1; ++↵
        motorNumber) {
28         katana->setCrashLimit(motorNumber, limit);
    }
30 katana->enableCrashLimits();
    katana->unBlock();
32
    if(opts.pos.size() >= 6)
34         katana->moveRobotToEnc(opts.pos, true);

36 if(opts.azimuth != INT_MAX)
    {
38         vector<int> enc = katana->getRobotEncoders(true);
        enc.at(0) = opts.azimuth;
40         katana->moveRobotToEnc(enc, true);
    }
42
    if(opts.elevation != INT_MAX)
44     {
        vector<int> enc = katana->getRobotEncoders(true);
46         enc.at(3) = opts.elevation;
        katana->moveRobotToEnc(enc, true);
48     }
    if(opts.mode == KATANA_LOW)
50     {
        vector<int> enc = katana->getRobotEncoders(true);

```

```
52     enc.at(1) = 2000;
53     enc.at(2) = 2500;
54     katana->moveRobotToEnc(enc, true);
55 }
56 else if(opts.mode == KATANA_HIGH)
57 {
58     vector<int> enc = katana->getRobotEncoders(true);
59     enc.at(1) = -22000;
60     enc.at(2) = -21000;
61     katana->moveRobotToEnc(enc, true);
62 }
63 }
64 catch (Exception &e) {
65     cout << "\nERROR: Unable to initialize Katana robot arm: " << e.message() << endl;
66 }
67
68 if(opts.turn || opts.len)
69 {
70     try {
71         if(opts.turn != 0)
72         {
73             stringstream ss; string n; ss << opts.turn; ss >> n;
74             string cmd = base_cmd + "a\015" + n + "\015";
75             system(cmd.c_str());
76         }
77
78         while(system(is_moving.c_str()) == 0)
79             ;
80
81         if(opts.len != 0)
82         {
83             stringstream ss; string n; ss << opts.len; ss >> n;
84             string cmd = base_cmd + "x\015" + n + "\015";
85             system(cmd.c_str());
86         }
87
88         while(system(is_moving.c_str()) == 0)
89             ;
90     }
91     catch (Exception &e) {
92         cout << "\nERROR: Unable to initialize Pioneer-3 AT mobile robot: " << e.message()
93             << endl;
94     }
95 }
96 return 0;
97 }
```

Source Code E.4: Mobile robot control.

```
2  /*
   * Author: Thor Andreassen <ta@toggle.be>
   */
4
   #include <libplayerc++/playerc++.h>
6  #include <libplayerc++/playerclient.h>
8
   #include <stdio.h>
   #include <stdlib.h>
10  #include <iostream>
   #include <string>
12  #include <sstream>
   #include <time.h>
14
   #define ts2dbl(timeValStruct) \
16   ((double) timeValStruct.tv_sec + timeValStruct.tv_nsec * 1E-9)
18
   using namespace PlayerCc;
   using namespace std;
20
   PlayerClient *robot;
22   Position2dProxy *pp;
   bool were_done = false;
24
   double targetx = 0;
26   double targeta = 0;
   double thresx = 0.03; // cm
28   double thresa = 0.10; // rad
   int cnt = 1;
30   bool at_pos = true;
   string what;
32   bool turning = false;
34
   void time_stamp(void);
   double time_elapsed(void);
36   struct timespec ts;
   double time_stamp_dbl = 0;
38
   void time_stamp(void) {
40     clock_gettime(CLOCK_REALTIME, &ts);
     time_stamp_dbl = ts2dbl(ts);
42   }
44
   double time_elapsed(void) {
     clock_gettime(CLOCK_REALTIME, &ts);
46     return(ts2dbl(ts) - time_stamp_dbl);
   }
```

```
48
50 void new_position(void) {
51     double cx = pp->GetXPos();
52
53     if(!at_pos)
54     {
55         if(what == "x")
56         {
57             if (cx < targetx - thresx)
58                 pp -> SetSpeed(limit(abs(cx - targetx)/1.5, 0.0, 0.4), 0.0);
59             else if(cx > targetx + thresx)
60                 pp -> SetSpeed(-limit(abs(cx - targetx)/1.5, 0.0, 0.4), 0.0);
61             else
62             {
63                 pp -> SetSpeed(0.0, 0.0);
64                 at_pos = true;
65             }
66         }
67         else if(what == "a")
68         {
69             if(!turning)
70             {
71                 turning = true;
72                 time_stamp();
73                 if(targeta < 0)
74                     pp -> SetSpeed(0, M_PI / 18.0);
75                 else
76                     pp -> SetSpeed(0, -M_PI / 18.0);
77             }
78             else
79             {
80                 if(time_elapsed() > fabs(targeta) / (M_PI / 18.0))
81                 {
82                     pp -> SetSpeed(0.0, 0.0);
83                     at_pos = true;
84                     turning = false;
85                 }
86             }
87         }
88     }
89 }
90
91 int main(int argc, char** argv)
92 {
93     double num;
94     try
95     {
96         robot = new PlayerClient();
97         pp = new Position2dProxy(robot);
```

```

98     pp -> ConnectReadSignal(&new_position);
100     pp -> SetMotorEnable(true);
102     robot->StartThread();
104     string where;

106     do {
108         if(at_pos)
110         {
112             system("unlink /tmp/vehicle_is_moving");
114             cin >> what;
116             if(what == "q")
118                 break;
120             cin >> where;
122             cout << "Got " << what << " = " << where << endl;
124             stringstream ss(where);
126             if(ss >> num)
128             {
130                 if(what == "x" && fabs(num) <= 2.0)
132                     targetx += num;
134                 if(what == "a" && fabs(num) <= M_PI)
136                     targeta = num;
138                 at_pos = false;
140             }
142         }
144         else
146         {
148             cout << "Moving to " << what << " = " << where << endl;
150             system("touch /tmp/vehicle_is_moving");
152             sleep(1);
154         }
156     } while(what != "q" && where != "q");

158     robot->StopThread();

160     pp -> SetMotorEnable(false);
162     cout << pp->GetXPos() << endl;
164     cout << pp->GetYPos() << endl;
166 }
168 catch (PlayerCc::PlayerError e)
170 {
172     cerr << e << endl;
174 }
176 return 0;
178 }

```

Sending commands to the on-board computer on the mobile Pioneer 3-AT robot, was accomplished through the `ssh` communication protocol

(OpenSSH developers, URL provided, accessed: Feb 2012) passing on the command to a **screen** session (Various, URL provided, accessed: Sep 2012[b]) running on the on-board computer. **screen** is a terminal multiplexer which provides a pseudo terminal for programs that need an attached terminal to run, it also allows other programs to communicate with embedded applications.

The dependence on **ssh** and **screen** can be seen from the head of *natasha.cpp* (Source Listing E.3), the line that defines the *P3_CMD* macro. It uses the **screen stuff** instruction which will pass any input on to the embedded program running within that **screen** session.

E.2 Future improvements

- Remove dependence on **ssh** and **screen** by communicate directly with the Player/Stage daemon.
- If a terminal multiplexer will still be needed, use **tmux** (Various, URL provided, accessed: Sep 2012[c]) instead of **screen** as it is actively maintained and anecdotally more stable than **screen**.
- Recent developments indicate that the Katana robot will be supported by the Player/Stage API. This would simplify the control surface a great deal, which could now take place through Player/Stage. If this development makes it to fruition the implementation in this Appendix will be redundant.

Bibliography

- Ackerman, J.R. (2008). “Toward Open Source Hardware”. In: *U. Dayton L. Rev.* 34, p. 183.
- Adept MobileRobots LLC (URL provided, accessed: Sep 2012). *Pioneer 3-AT Datasheet*. URL: <http://www.mobilerobots.com/Libraries/Downloads/Pioneer3AT-P3AT-RevA.sflb.ashx>.
- Adlink Technology inc. (URL provided, accessed: May 2012). *DAQ-2010 data aquisition board*. URL: http://www.adlinktech.com/PD/web/PD_detail.php?pid=12.
- Ahlén, Ingemar et al. (2007). *Bats and offshore wind turbines studied in southern Scandinavia*. Tech. rep. Swedish Environmental Protection Agency.
- al, Chris Bagwell et (URL provided, accessed: 2013). *SoX manual page*. URL: <http://sox.sourceforge.net/sox.html>.
- Ali, A.M. et al. (2007). “Acoustic source localization using the acoustic ENSBox”. In: *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*. IEEE, pp. 555–556.
- Allen, M. et al. (2008). “VoxNet: An interactive, rapidly-deployable acoustic monitoring platform”. In: *Proceedings of the 7th international conference on Information processing in sensor networks*. IEEE Computer Society, pp. 371–382.
- Andreassen, Tórrur (2012). *Software used with the system: ADC driver, GNU/Linux setup files, signal detection program*. date last viewed 10/9/2012. URL: http://array.chioping.org/long_term_recording_array-software.tar.gz.
- Andreassen, Tórrur (URL provided, accessed: Jun 2011). *Control software for combined Katana 450 and Pioneer 3-AT robots*. URL: http://array.chioping.org/robot_control.tgz.
- Andreassen, Tórrur, Annemarie Surlykke, and John Hallam (Manuscript submitted). “Automatic long-term recording and automatic bat call ex-

- traction with Support Vector Machine models”. In: *Ecological Informatics*.
- Andreassen, Tórrur, Annemarie Surlykke, John Hallam, and David Brandt (In press). “Ultrasonic recording system without intrinsic limits”. In: *The Journal of the Acoustical Society of America*.
- ANSI (1994). *ANSI 131-1994 (R1999): Information Systems - Small Computer Systems Interface-2 (SCSI-2) (formerly ANSI X3.131-1994 (R1999))*. an overview of the subject can be found at <http://en.wikipedia.org/wiki/SCSI>. American National Standards Institute. URL: [http://webstore.ansi.org/RecordDetail.aspx?sku=ANSI+INCITS+131-1994+\(R1999\)](http://webstore.ansi.org/RecordDetail.aspx?sku=ANSI+INCITS+131-1994+(R1999)).
- Avisoft (URL provided, accessed: Aug 2012). *Ultrasonic Dynamic Speaker – ScanSpeak*. URL: <http://avisoft.com/usg/scanspeak.htm>.
- Avisoft (URL provided, accessed: Dec 2011). *AD converter UltraSoundGate 1216H*. URL: <http://avisoft.com/usg/usg1216h.htm>.
- Baagøe, Hans J (2001). “Danish Bats (Mammalia: Chiroptera): Atlas and analysis of distribution, occurrence, and abundance”. In: *Steenstrupia* 26, 1, pp. 1–117.
- Baerwald, Erin F and Robert MR Barclay (2009). “Geographic variation in activity and fatality of migratory bats at wind energy facilities”. In: *Journal of Mammalogy* 90.6, pp. 1341–1349.
- Baker, Alan (2011). “Simplicity”. In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Summer 2011.
- Barclay, Robert M. R., E. F. Baerwald, and J. C. Gruver (2007). “Variation in bat and bird fatalities at wind energy facilities: assessing the effects of rotor size and tower height”. In: *Canadian Journal of Zoology* 85.3, pp. 381–387.
- Barlow, Kate E. and Gareth Jones (1997a). “Differences in songflight calls and social calls between two phonic types of the vespertilionid bat *Pipistrellus pipistrellus*”. In: *Journal of Zoology* 241.2, pp. 315–324.
- Barlow, Kate E and Gareth Jones (1997b). “Function of pipistrelle social calls: field data and a playback experiment”. In: *Animal behaviour* 53.5, pp. 991–999.

- Barshan, B. and R. Kuc (1992). “A bat-like sonar system for obstacle localization”. In: *Systems, Man and Cybernetics, IEEE Transactions on* 22.4, pp. 636–646.
- Blumstein, Daniel T et al. (2011). “Acoustic monitoring in terrestrial environments using microphone arrays: applications, technological considerations and prospectus”. In: *Journal of Applied Ecology* 48.3, pp. 758–767.
- bobsdata.com (URL provided, accessed: July 28 2008). *Parabolic microphone*. URL: http://www.bobsdata.com/parabolic_microphone/index.html.
- Boser, Bernhard E., Isabelle M. Guyon, and Vladimir N. Vapnik (1992). “A training algorithm for optimal margin classifiers”. In: *Proceedings of the fifth annual workshop on Computational learning theory*. ACM, pp. 144–152.
- Brandt, David (URL provided, accessed: Dec 2011). *Amplifier/filter schematics, gerber files, and system software*. URL: http://array.chiroping.org/long_term_recording_array-hardware.tar.gz.
- Bruderer, B. and A.G. Popa-Lisseanu (2005). “Radar data on wing-beat frequencies and flight speeds of two bat species”. In: *Acta Chiropterologica* 7.1, pp. 73–82.
- Chang, Chih-Chung and Chih-Jen Lin (2011). “LIBSVM: a library for support vector machines”. In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 2.3, p. 27.
- Chan, YT and KC Ho (1994). “A simple and efficient estimator for hyperbolic location”. In: *Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on]* 42.8, pp. 1905–1915.
- ChiRoPing (URL provided, accessed: May 2013). *ChiRoPing: Developing versatile and robust perception using sonarsystems that integrate active sensing, morphology and behavior*. URL: <http://www.chiroping.org>.
- Coar, Ken et al. (URL provided, accessed: July 2011). *The Open Source Definition*. URL: <http://www.opensource.org/docs/osd>.

- Collier, T.C. (2011). “Wireless sensor network-based acoustic localization for studying animal communication in terrestrial environments”. PhD thesis. University of California, Los Angeles.
- Cryan, Paul M et al. (2012). “Evidence of Late-Summer Mating Readiness and Early Sexual Maturation in Migratory Tree-Roosting Bats Found Dead at Wind Turbines”. In: *PloS one* 7.10, e47586.
- Denning, Peter et al. (2005). “Wikipedia risks”. In: *Communications of the ACM* 48.12, pp. 152–152.
- Dubgaard, Anders (2012). “Automated Calibration of Acoustic Array Geometry”. MA thesis. University of Southern Denmark.
- Dzal, Y. et al. (2011). “Going, going, gone: the impact of white-nose syndrome on the summer activity of the little brown bat (*Myotis lucifugus*)”. In: *Biology Letters* 7.3, pp. 392–394.
- ecoObs (URL provided, accessed: Dec 2011). *ecoObs batcorder products*. URL: <http://www.ecoobs.com/cnt-batcorder.html>.
- E. F. Baerwald, R.M.R. Barclay (2011). “Patterns of activity and fatality of migratory bats at a wind energy facility in Alberta, Canada”. In: *The Journal of Wildlife Management* 75.5, pp. 1103–1114.
- Fiedler, J.K. (2004). “Assessment of bat mortality and activity at Buffalo Mountain Windfarm, eastern Tennessee”. PhD thesis. University of Tennessee, Knoxville.
- Fine Offset Electronics (URL provided, accessed: Apr 2012). *WH1090 - Professional weather center with PC interface*. URL: http://www.foshk.com/Weather_Professional/WH1090.htm.
- Fisher, Robert (URL provided, accessed: July 24 2008). In “*CVonline: Citation of Contents*”. URL: <http://homepages.inf.ed.ac.uk/rbf/CVonline/SUPPORT/citations.htm>.
- Flaquer, Carles, Ignacio Torre, and Antoni Arrizabalaga (2007). “Comparison of sampling methods for inventory of bat communities”. In: *Journal of Mammalogy* 88.2, pp. 526–533.
- Geipel, Inga, Kirsten Jung, and Elisabeth Kalko (2013). “Perception of silent and motionless prey on vegetation by echolocation in the glean-

- ing bat *Micronycteris microtis*”. In: *Proceedings of the Royal Society B: Biological Sciences* 280.1754.
- Georgiakakis, Panagiotis and Danilo Russo (2012). “The Distinctive Structure of Social Calls by Hanák’s Dwarf Bat *Pipistrellus hanaki*”. In: *Acta Chiropterologica* 14.1, pp. 167–174.
- Gillam, E.H., N. Ulanovsky, and G.F. McCracken (2007). “Rapid jamming avoidance in biosonar”. In: *Proceedings of the Royal Society B: Biological Sciences* 274.1610, p. 651.
- Girod, L., K. Jamieson, et al. (2006). “WaveScope: a signal-oriented data stream management system”. In: *Proceedings of the 4th international conference on Embedded networked sensor systems*. ACM, pp. 421–422.
- Girod, L., M. Lukac, et al. (2006). “The design and implementation of a self-calibrating distributed acoustic sensing platform”. In: *Proceedings of the 4th international conference on Embedded networked sensor systems*. ACM, pp. 71–84.
- Globalscale Technologies Inc. (URL provided, accessed: Jul 2012[a]). *Dreamplug – hardware specification*. URL: <http://www.globalscaletechnologies.com/t-dreamplugdetails.aspx>.
- Globalscale Technologies Inc. (URL provided, accessed: Jul 2012[b]). *Dreamplug – User guide*. URL: http://dreamplug.googlecode.com/files/DreamPlug%20User%20Guide_US_v1.1.pdf.
- GNU coreutils developers (Feb. 2012). *The POSIX compliant tee command from the GNU project*. URL: http://www.gnu.org/software/coreutils/manual/html_node/tee-invocation.html#tee-invocation.
- Gorresen, P Marcos et al. (2008). “Assessing bat detectability and occupancy with multiple automated echolocation detectors”. In: *Journal of Mammalogy* 89.1, pp. 11–17.
- G.R.A.S. (URL provided, accessed: May 2013). *40BF 1/4” Ext. Polarized Free-field Microphone*. URL: <http://www.gras.dk/g-r-a-s-40bf-free-field-microphone.html>.

- Guarato, F. and J. Hallam (2009). “Determining sound source orientation from source directivity and multi-microphone recordings”. In: *Bioinspired Applications in Artificial and Natural Computation*, pp. 429–438.
- Hiryu, S. et al. (2005). “Doppler-shift compensation in the Taiwanese leaf-nosed bat (*Hipposideros terasensis*) recorded with a telemetry microphone system during flight”. In: *The Journal of the Acoustical Society of America* 118, p. 3927.
- Hristov, N.I., M. Betke, and T.H. Kunz (2008). “Applications of thermal infrared imaging for research in aeroecology”. In: *Integrative and Comparative Biology* 48.1, pp. 50–59.
- IEC (June 2006). *IEC 60027: standard on Letter symbols to be used in electrical technology*. Tech. rep. International Organization for Standardization, Geneva, Switzerland.
- Incite Technology Ltd. (2012). *Product description of the USB-DUXfast AD converter*. date last viewed 10/9/2012. URL: http://www.linux-usb-daq.co.uk/prod2_duxfast.
- Jensen, M.E. and L.A. Miller (1999). “Echolocation signals of the bat *Eptesicus serotinus* recorded using a vertical microphone array: effect of flight altitude on searching signals”. In: *Behavioral Ecology and Sociobiology* 47.1, pp. 60–69.
- Joyner, David and William Stein (Nov. 2007). “Open source mathematical software”. In: *Notices of the American Mathematical Society*, p. 1279.
- Kirchner, WH and J. Röschard (1999). “Hissing in bumblebees: an interspecific defence signal”. In: *Insectes sociaux* 46.3, pp. 239–243.
- Kjær, Brüel & (URL provided, accessed: Sep 2012). *B&K turntable type 9640*. URL: <http://www.bksv.com/products/telecomaudiosolutions/electroacousticsaccessories/turntablessystemtype9640.aspx>.
- Knowles Acoustics (URL provided, accessed: Aug 2012). *Microphone type FG23329-P07 datasheet at Farnell*. URL: <http://www.farnell.com/datasheets/49984.pdf>.
- Kunz, T.H., E.B. Arnett, B.M. Cooper, et al. (2007). “Assessing Impacts of Wind-Energy Development on Nocturnally Active Birds and Bats:

- A Guidance Document". In: *The Journal of Wildlife Management* 71.8, pp. 2449–2486.
- Kunz, Thomas H, Edward B Arnett, Wallace P Erickson, et al. (2007). "Ecological impacts of wind energy development on bats: questions, research needs, and hypotheses". In: *Frontiers in Ecology and the Environment* 5.6, pp. 315–324.
- Luke, H.D. (1999). "The origins of the sampling theorem". In: *Communications Magazine, IEEE* 37.4, pp. 106–108.
- Lundberg, Karin and Rune Gerell (1986). "Territorial advertisement and mate attraction in the bat *Pipistrellus pipistrellus*". In: *Ethology* 71.2, pp. 115–124.
- Macías, Silvio, Emanuel C Mora, and Adianez García (2006). "Acoustic identification of mormoopid bats: a survey during the evening exodus". In: *Journal of mammalogy* 87.2, pp. 324–330.
- Madsen, P.T. and M. Wahlberg (2007). "Recording and quantification of ultrasonic echolocation clicks from free-ranging toothed whales". In: *Deep Sea Research Part I: Oceanographic Research Papers* 54.8, pp. 1421–1444.
- Manthey, W., N. Kroemer, and V. Magori (1999). "Ultrasonic transducers and transducer arrays for applications in air". In: *Measurement Science and Technology* 3.3, p. 249.
- McGarry, Daniel D (1955). *The Metalogicon of John Salisbury*. Univ of California Press.
- McIlroy, MD, EN Pinson, and BA Tague (1978). "Unix time-sharing system forward". In: *The Bell system technical journal* 57.6 part 2, p. 1902.
- Microsoft (1991). *Microsoft Windows multimedia programmer's reference*. Microsoft Press. ISBN: 1556153899. URL: <http://download.microsoft.com/download/9/8/6/9863C72A-A3AA-4DDB-B1BA-CA8D17EFD2D4/RIFFNEW.pdf>.
- Neuronics AG (URL provided, accessed: Sep 2012). *Katana 450 Benutzerhandbuch*. URL: http://www.bachofen.ch/fr/robotics/Vision/Software/Techdaten/Downloads/Katana_450_Benutzerhandbuch.

- pdf/moduleAktion?formular=pdfDownload&Download=Katana_450_Benutzerhandbuch.pdf.
- Newton, R. et al. (2008). “Wavescript: A case-study in applying a distributed stream-processing language”. In: *system* 1.2008/1, p. 31.
- Norcomp Inc. (URL provided, accessed: May 2013). *Female 44-pin HD connector datasheet*. URL: <http://www.norcomp.net/rohspdfs/Connectors/18Y/181/181-YYY-213RYY1.pdf>.
- O’Farrell, Michael J, Bruce W Miller, and William L Gannon (1999). “Qualitative identification of free-flying bats using the Anabat detector”. In: *Journal of Mammalogy*, pp. 11–23.
- OpenSSH developers (URL provided, accessed: Feb 2012). *The ssh command*. URL: <http://www.openbsd.org/cgi-bin/man.cgi?query=ssh&sektion=1>.
- Page, Melvin E. (URL provided, accessed: July 24 2008). In “*A brief citation guide for internet sources in history and the humanities*”. URL: <http://www.h-net.org/~africa/citation.html>.
- Patashnik, Oren et al. (URL provided, accessed: July 24 2008). *BibTeX: Your BibTeX resource*. URL: <http://www.bibtex.org/>.
- Pendec, Michael et al. (URL provided, accessed: Apr 2012). *Wireless Weather Station Reader*. URL: <http://meteolinger.lowlevel.cz/tracker/wiki>.
- Petterson Elektronik AB (URL provided, accessed: Dec 2011). *Petterson Elektronik AB ultrasound detectors*. URL: <http://www.batsound.com/?p=2>.
- Pi, Raspberry (URL provided, accessed: May 2013). *An ARM GNU/Linux box for \$25*. URL: <http://www.raspberrypi.org>.
- Raymond, Eric S. (1999). *The Cathedral and the Bazaar*. Ed. by Tim O’Reilly. 1st. Sebastopol, CA, USA: O’Reilly & Associates, Inc. ISBN: 1565927249.
- Rich, B.R. (2004). “Clarence Leonard (Kelly) Johnson”. In: *National Academy of Sciences Biographical Memoirs*. URL: <http://www.nap.edu/html/bio/mems/cjohnson.html>.

- Russ, JM and PA Racey (2007). “Species-specificity and individual variation in the song of male Nathusius’ pipistrelles (*Pipistrellus nathusii*)”. In: *Behavioral Ecology and Sociobiology* 61.5, pp. 669–677.
- Russo, Danilo and Gareth Jones (1999). “The social calls of Kuhl’s pipistrelles *Pipistrellus kuhlii* (Kuhl, 1819): structure and variation (Chiroptera: Vespertilionidae)”. In: *Journal of Zoology* 249.4, pp. 476–481.
- Samotronic (URL provided, accessed: Sep 2012). *Motor controller type SE2*. URL: <http://dk.rs-online.com/web/p/products/4430102/>.
- Schnitzler, Hans-Ulrich and Elisabeth K. V. Kalko (2001). “Echolocation by Insect-Eating Bats: We define four distinct functional groups of bats and find differences in signal structure that correlate with the typical echolocation tasks faced by each group”. In: *Bioscience* 51.7, pp. 557–569.
- Schnitzler, Hans-Ulrich, Elisabeth Kalko, et al. (1994). “Fishing and echolocation behavior of the greater bulldog bat, *Noctilio leporinus*, in the field”. In: *Behavioral Ecology and Sociobiology* 35.5, pp. 327–345.
- SensComp (URL provided, accessed: Aug 2012). *Series 7000 Transducer*. URL: <http://www.senscomp.com/pdfs/series-7000-transducer.pdf>.
- Shannon, C.E. (July 1948). “A mathematical theory of communication”. In: *Bell System Technical Journal* 27,
- Skalak, Samuel L, Richard E Sherwin, and R Mark Brigham (2012). “Sampling period, size and duration influence measures of bat species richness from acoustic surveys”. In: *Methods in Ecology and Evolution* 3.3, pp. 490–502.
- Smith, Steven W (2003). *The Scientist and Engineer’s Guide to Digital Signal Processing*. FreeTech Books.
- Sorainen, E. and E. Rytkönen (2002). “Noise level and ultrasound spectra during burring”. In: *Clinical oral investigations* 6.3, pp. 133–136.
- SoX developers (Feb. 2012). *The SoX command*. URL: <http://sox.sourceforge.net/sox.html>.

- Spiesberger, J.L. (2004). “Geometry of locating sounds from differences in travel time: Isodiachrons”. In: *The Journal of the Acoustical Society of America* 116, p. 3168.
- Surlykke, Annemarie and Elisabeth K. V. Kalko (2008). “Echolocating bats cry out loud to detect their prey”. In: *PLoS one* 3.4, e2036.
- Tange, O. (Feb. 2011). “GNU Parallel - The Command-Line Power Tool”. In: *login: The USENIX Magazine* 36.1, pp. 42–47. URL: <http://www.gnu.org/s/parallel>.
- TASCAM (URL provided, accessed: Dec 2011). *Ultrasound recorder HD-P2*. URL: <http://tascam.com/product/hd-p2/specifications>.
- the UK TeX enthusiasts (URL provided, accessed: July 24 2008). *URLs in BibTeX bibliographies*. URL: <http://www.tex.ac.uk/cgi-bin/texfaq2html?label=citeURL>.
- Tiemann, Michael et al. (URL provided, accessed: July 2011). *About the Open Source Initiative*. URL: <http://www.opensource.org/about>.
- Various (URL provided, accessed: Sep 2012[a]). *Bibliographies in LaTeX using BibTeX for sorting only*. URL: <http://ctan.org/pkg/biblatex>.
- Various (URL provided, accessed: Sep 2012[b]). *GNU screen – terminal multiplexer*. URL: <http://www.gnu.org/software/screen/>.
- Various (URL provided, accessed: Sep 2012[c]). *GNU screen – terminal multiplexer*. URL: <http://tmux.sourceforge.net/>.
- Various (URL provided, accessed: Sep 2012[d]). *Player Stage framework*. URL: <http://playerstage.sourceforge.net/>.
- Via (Nov. 2001). *New Mini-ITX Mainboard Specification White Paper*. URL: http://www.via.com.tw/en/downloads/whitepapers/initiatives/spearhead/ini_mini-itx.pdf.
- Wikipedia (URL provided, accessed: Apr 2012). *SCSI - Small Computer System Interface*. URL: <http://en.wikipedia.org/wiki/SCSI>.
- Wildlife Acoustics (URL provided, accessed: Apr 2012). *Ultrasound recorder SM2BAT+*. URL: <http://www.wildlifeacoustics.com/products/ultrasonic-monitoring/sm2bat>.
- Wu, J. and I. Stojmenovic (2004). “Ad hoc networks”. In: *COMPUTER-IEEE COMPUTER SOCIETY- 37.2*. an introduction to mesh concepts

can be found at http://en.wikipedia.org/wiki/Wireless_mesh_network, pp. 29–31.

List of Figures

2.1	Block diagram of a distributed microphone array	19
3.1	First array module prototype	24
3.2	Second prototype	27
4.1	Transfer function for a G.R.A.S. $\frac{1}{4}$ inch microphone model 40BF	30
4.2	Multi-channel recording	32
4.3	Microphone protection strategies	35
4.4	Two implementations of the echolocation sensor	37
4.5	Echolocating robot	38
4.6	Diagram of echolocating robot control	39
4.7	Objects to be located by the echolocating robot	40
4.8	Typical recording when locating objects – left ear	41
4.9	Typical recording when locating objects – right ear	42
4.10	Calibration setup	44
4.11	The B&K turntable used for finding echo-profiles of objects	46
4.12	Motor-controller and digital IO used with turntable	46
4.13	Emitted signal	48
4.14	Before emission x	50
4.15	After emission – x convolved with h ($x * h$)	51
4.16	Emission of x deconvolved with h	52
4.17	Measurements from array-local weather station	55
5.1	A tetrahedron mounted with four Knowles microphones	58
5.2	The relation between programs linked with Unix pipes	60
5.3	A typical sorted cumulative density plot	63
5.4	Quality measure comparison – bandwidth	64
5.5	Quality measure comparison – frequency entropy	65
5.6	Quality measure comparison – minimum frequency	66
5.7	ROC curves comparing scaling effects on SVM parameters	70

6.1	Temporal distribution of recorded data	75
6.2	Histogram of recording file sizes	77
6.3	Events per recording per second and recording event ratio . . .	78
6.4	Temporal distribution of detected events	80
6.5	Temporal distribution of bat events	81
6.6	Precipitation measured 2.5 km away from the survey site	82
6.7	Precipitation measured 5 km away from the survey site	83
6.8	Example of typical events classified during the tally	85
7.1	Copy of Figure 6.5 for convenience	99
7.2	Cloudcover and lunar cycle	100
7.3	Temperature measurements during ecological study	101
7.4	Custom box with ultrasonic speaker and transducer	107
7.5	Published and measured ScanSpeak transfer functions	108
B.1	Variable amplification test of box6 – 20 mV at 30, 50, 70 kHz .	130
B.2	Variable amplification test of box6 – 40 mV at 30, 50, 70 kHz .	131
B.3	Variable amplification test of box6 – 80 mV at 30 kHz, 50 kHz, and 70 kHz	132
B.4	Transfer functions of microphones 3, 6, and 11	133
B.5	Transfer functions of microphones 2, 10, and 16	134
B.6	Transfer functions of microphone 4 and 18	135
B.7	Transfer functions of microphones 18 and 30	136
B.8	Transfer functions of microphones 8, 9, and 19	137
B.9	Transfer functions of microphones 1, 20, and 21	138
B.10	Transfer functions of microphones 13 and 14	139
C.1	GNU Radio spectrogram waterfall screenshot	142

List of Tables

2.1	Design criteria for the array system	8
2.2	Comparison of the different recording systems	15
4.1	Summary of medium-term recordings	31
6.1	Manually identified bat events	84
7.1	Per 4 channel array comparison	96

List of Source Code Listings

5.1 Start A/D conversion and signal-detection	60
7.1 Advanced pipe example	105
A.1 <code>adac</code> – one way to continuously sample the DAQ-2010	113
A.2 <code>signal_detect</code> – signal detection for long-term recordings . . .	119
A.3 <code>process.m</code> – calculations of quality measures	123
C.1 GNU Radio block program: spectrogram waterfall view	142
C.2 Multiplexing files with <code>SoX</code>	145
C.3 Multiplex streams from two named pipes <code>fifo1</code> and <code>fifo2</code>	146
C.4 Demultiplex a stream from <code>tee</code> with <code>SoX</code>	146
C.5 Example of how to forward a stream	146
C.6 Example of how to amplify a 4 channel stream	147
C.7 Example of how to bandpass filter a 4 channel stream	147
C.8 Example of how to select a subset of channels from a stream . .	147
C.9 Example of conditional pipelines	148
D.1 Sum and average	150
D.2 Source code for finding x deconvolved with h	151
D.3 Trimming many files with <code>SoX</code>	151
D.4 Apply a transfer function to a signal	152
E.2 Main robot control program – header file	153
E.1 Makefile for compiling against the KNI library	154
E.3 Main robot control program – source file	155
E.4 Mobile robot control	157

List of Algorithms

- 4.1 Align and average algorithm. See Source Listing D.1 on page 150
for implementation details of this procedure 49
- 4.2 Apply transfer function to a signal. See Source Listing D.4 on
page 152 for implementation details of this procedure. 49

ARTICLE

Ultrasonic recording limits without intrinsic limits

The article is in press to be published in JASA.

Ultrasonic recording system without intrinsic limits

Tórir Andreassen^{a)} and Annemarie Surlykke

Institute of Biology, University of Southern Denmark, Campusvej 55 5230 Odense M Denmark

John Hallam and David Brandt

*Mærsk McKinney Møller Institute, University of Southern Denmark, Campusvej 55
5230 Odense M Denmark*

Today state-of-the-art bioacoustic research requires high-sample-rate, multi-channel, and often long-term recording systems. Commercial systems are very costly. This paper proposes and demonstrates an ultrasonic recording system (URS) design that is arbitrarily scalable. The system is modular and based on retail components and open source software/hardware. Each module has four microphones and modules can be combined to extend the coverage area, obtain higher spatial recording resolution and/or add recording redundancy. The system is designed to have no inherent scalability limits.

The system has been deployed in four different test settings. The first setup tests the system's ability to make medium-term recordings (1-2 minutes) with many microphones. The second setup tests the robustness of the system, being deployed throughout the Danish winter with only minor issues. The third setup integrates the system in a mobile robot as an echolocating guidance system, while the fourth setup demonstrates full-spectrum transducer calibration.

In most respects this system's hardware specification surpasses all competitors on the market at a quarter of the price. Tests demonstrate that large deployments are feasible and accurate ultrasonic measurements can be obtained.

PACS numbers: 43.60.Fg, 43.58.Ta, 43.60.Vx

I. INTRODUCTION

Today state-of-the-art bioacoustic research requires high-sample-rate, multi-channel, and often long-term recording systems. One application of long-term recordings has been when planning large structures, because there are increasing concerns about the impact of such structures on local bat populations. The concerns involve, for example, habitats, foraging grounds and migratory routes. Many methods of assessing impact are used: carcass counting post-construction (Fiedler, 2004; Barclay *et al.*, 2007), radar tracking (Bruderer and Popa-Lisseanu, 2005; Kunz *et al.*, 2007), infrared thermography tracking (Hristov *et al.*, 2008) and passive acoustical measurements, usually with one or more bat-detectors. Each of these methods has benefits and drawbacks.

When trying to survey a large area, none of these methods covers all of the following important aspects: 1) estimating how many bats have passed by 2) estimating what species have passed by 3) estimating their flight paths and 4) supporting automatic unsupervised long-term survey. Carcass counting is done too late and does not reveal the flight behavior, death rate and reaction to the structure. Radar tracking and stereo infrared thermography can give suitable data, at least for large species, but availability and price make them infeasible; multiple bat-detectors could, with some effort, be used to automatically count passing bats and estimate their species — but unless they are synchronized, flight-path extraction is difficult.

In principle, an acoustical array would provide all the

information needed as bats echolocate when commuting or hunting. Bats hunting for insect prey out in the open emit high-intensity calls (Jensen and Miller, 1999; Surlykke and Kalko, 2008). The multi-microphone array technique proposed exploits this fact. This gives the system an inherent bias towards loud bats; but windmills present the largest danger to bats out in the open hunting or migrating (Baerwald and Barclay, 2011; Barclay *et al.*, 2007) and since those bats are among the loudest this bias might not be a large issue.

The URS (Ultrasonic Recording System) market of today can only cover these requirements at high cost, and therefore we propose this new design. This new microphone array system is designed to fulfill the above needs at a fraction of the price of commercial or research alternatives.

A. Criteria for an ultrasonic recording system

An important goal of this work is that the system should be applicable to a wide range of experiments. The design chosen is therefore open-ended and the first prototype implementation is based on a versatile analog-to-digital and digital-to-analog platform. The design criteria we set up to ensure the system is useful for the anticipated applications and goals are listed below. The criteria are divided into three categories: *critical* criteria are considered essential — that is, if one of these is not fulfilled the system would not be useful; *important* criteria are features that the system might work without, but much emphasis has been put on achieving those goals; the *nice* criteria were implemented if convenient (Table I).

^{a)}Electronic address: thor@biology.sdu.dk

critical	1. Record ultrasound. 2. Easily adaptable to new experiments. 3. Record for long periods. 4. No inherent scalability limits. 5. Pre-trigger recording. 6. Data to estimate flight-paths.
important	7. Real-time access to signal data from computer programs. 8. Use generic mass-market components leading to easy upgrade and low cost. 9. Use well defined protocols and standards to facilitate interoperability.
nice	10. Platform agnostic. 11. Playback of ultrasound. 12. Save data in well-defined format. 13. Generate real-time event log of data.

TABLE I. System requirements for an ultrasonic recording system. The criteria are ordered according to importance and placed into three priority groups: *critical*, *important*, and *nice*.

1. Critical Criteria

The system should be: capable of recording ultrasound for long periods and easily adaptable to new experiments. It should have no inherent scalability limits, which implies that all aspects of the system should be extensible i.e. more microphones, more storage, and more computers can be added with a linear increase in spatial coverage and performance.

Pre-trigger recording, i.e. recording of events happening n seconds prior to triggering requires that n seconds of recording are continuously stored in a round-robin or ring buffer. The data in the buffer is permanently stored upon a trigger event. This is essential for recording unpredictable events such as animal sounds.

In order to recover flight paths, a minimum of four synchronized channels is needed. This follows from the time delay of arrival equations (Madsen and Wahlberg, 2007; Spiesberger, 2004) from which one recovers the 3D position of the emitting source.

2. Important Criteria

Real-time access to the data during long-term recordings enables the system to record only bits that satisfy pre-programmed criteria, e.g. sections identified by a matched filter bank of relevant bat calls.

The system should use generic mass-market components leading to easy upgrade and low cost. It should communicate with well-defined publicly available formats

and standards to facilitate interoperability, both in direct inter-system communication and indirectly through stored data.

3. Nice Criteria

The design of the system should preferably be platform agnostic, so it can be implemented on the available hardware. It should be capable of ultrasound playback. It would be convenient if the system could generate a real-time event log of recorded data: interesting events should be time-stamped, analyzed, and recorded in a streamable log, which would facilitate real-time display and analysis of array recordings.

II. RELATED WORK

Recording ultrasound requires hardware which is capable of sampling data at sufficiently high rates (Luke, 1999; Shannon, 1948) but this generates large data volumes. Thus, the system should be able to handle large data volumes, while at the same time being versatile and rugged.

When recording ultrasound in the field there are limited options with regards to equipment. The system described here is compared to several commercial solutions and one university-developed solution called VoxNet (Allen *et al.* (2008)) with respect to the design criteria listed in section I.A.

The existing commercial solutions can be divided into two categories: multi-channel triggered arrays, and single- or dual-channel long-term recording systems. The comparison of these systems and the one proposed in this paper is presented in Table II.

The commercial solutions perform fairly well with regards to some of our criteria, but not all. Either they are too simple (like bat detectors) or, if they can deal with multiple channels, they are inflexible and become prohibitively expensive when scaling them up. The VoxNet system is both versatile and arbitrarily scalable, sharing some important design choices with the new system described here, but VoxNet could not record ultrasound without a major re-implementation and partial re-design.

A. Hardware

Within the **triggered** array category, a popular system among biologists is the Avisoft AD converter, e.g. the USG1216H (Avisoft Bioacoustics, 2012). Similar systems from other companies, such as National Instruments, Tucker/Davis etc., are commonly in use. Such systems are used for obtaining many synchronized acoustic samples from many places at the same time, though usually for short periods. The AD converter is fed from an analog processing chain comprising analog microphones, filters, pre-amplifiers, and amplifiers — such as the Avisoft analog chain used in Gillam *et al.* (2007) and Dzal *et al.* (2011) or a chain based on G.R.A.S./B&K equipment

as used in Sorainen and Rytönen (2002), Kirchner and Röschard (1999), and Surlykke and Kalko (2008). The exact choice of system and optional hardware depends on the requirements and resources available.

There are now a number of different **long-term** recording systems, which are primarily used for surveys assessing bat activity at an area of interest, e.g. when planning to build large constructions. Systems evaluated include the SM2BAT+ from Wildlife Acoustics (2012), HD-P2 from TASCAM, a division of TEAC America, Inc. (2012), the Batcorder from ecoObs GmbH (2012) and detectors from Petterson Elektronik AB (2012). These are single- or dual-channel systems which are often acoustically triggered and store their recordings on-board. They are usually rugged and can be deployed outdoors to capture data continuously for months.

The academic solution referred to is the VoxNet embedded sensing platform (Allen *et al.*, 2008) and its predecessor the ENSBox (Girod *et al.*, 2006b), used in bioacoustics by, for example, Ali *et al.* (2007) and Collier (2011). VoxNet and ENSBox are based on networked nodes where each node records data and, if appropriate, propagates it back to a network master. Thus these two use a different design from any of the systems described above. For networking they rely on a mesh network topology, a type of networking where each node must not only handle its own network communication but also serve as a network relay for other nodes (Wu and Stojmenovic, 2004). This means that as long as each node can “see” enough other nodes, it can communicate with *all* other nodes. Each node has on-board storage, a processing unit, and a wireless link to nearby nodes. The system scales very well, is flexible and cheap (easily adaptable to new problems). Long-term recording to local storage is possible, and recording can be triggered by analyzing the incoming data stream. The system includes its own stream programming language Newton *et al.* (2008). Depending on the iteration of the system, it is based on a Slauson Single Board Computer or the smaller Mica Mote and a fair amount of custom-made electronics and protocols.

VoxNet’s design is modular and distributed, like the one presented here. However, it cannot record ultrasonic signals in its present form and therefore does not satisfy our critical criteria; its extensive dependence on custom components means that to adapt it for ultrasonic recording would be a major redesign.

B. Software

Recording acoustical data in real-time means handling continuous streams of data. Most of the systems mentioned above have developed their own proprietary solutions to handle the data streams, normally excluding any outside access. That is, if you have a concrete task involving a criterion you wish to apply to the real-time data (such as online classification, saving only some bits of the stream, triggering only when a matched filter hits, etc.) it is usually impossible without involving the vendor of your product. Only the vendor’s accompanying software

	Record ultrasound (1)	Flexibility (2)	Long-term (3)	Scalability (4)	Post-trigger (5)	Minimum channels (6)	Real-time access (7)	Generic (8)	Interoperable (9)	Platform agnostic (10)	Emit ultrasound (11)	Event log (13)
triggered	✓	(✓)	✓	✓	✓	✓						
long-term	✓		✓		✓						✓	
VoxNet		✓	✓	✓		✓	✓	✓	✓	✓		
mclurs	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	

TABLE II. Comparison of the different recording systems, with the criteria from Table I as keys in the table column (numbers refer to the corresponding point in Table I). The compared systems are shortened thus: **triggered** refers to commercial triggered arrays, **long-term** to mono- or dual-channel long-term commercial recording systems, **VoxNet** is a flexible network system and **mclurs** is the proposed system described in this paper. Check marks in parenthesis, i.e. (✓), are qualified statements as follows. (**triggered**, flexibility): although technically possible, these systems are usually hard to apply to problems that they were not designed for. (**triggered**, scalability): these systems are in principle scalable — if multiple AD converters are connected with a common trigger distributed recording is possible; however the recording length would be limited by local storage capacity. This option soon becomes prohibitively expensive if a large number of channels are needed.

handles real-time streams, and the user is restricted to the built-in software solution.

This real-time restriction does not hold for the Voxnet system, where external real-time access was designed in from the beginning (Girod *et al.*, 2006b,a; Newton *et al.*, 2008).

III. DESIGN

As guiding design principles, we have taken inspiration from Kelly Johnson (Skunk Works) and Douglas McIlroy (of Unix fame). Johnson said *always use the simplest solution possible* (Rich, 1995, p. 221 and 231) and McIlroy recommends (slightly modified) *creating components that do one thing and do it well* (McIlroy *et al.*, 1978).

It is also helpful to attain a higher level of abstraction in design, making it possible to ignore underlying complexity; this necessitates that the possibilities and limitations of the component interfaces are well understood. Another desirable trait is that the design should be platform agnostic, so it can be implemented in any operating system and architecture.

A schematic of the design can be seen in Figure 1. The system consists of multiple ultrasonic microphone array recordings systems, modules 1 to n . In each module, the transducers (microphones) convert pressure changes impinging on them into a voltage signal, which is amplified and filtered. The signal is then digitized in an analog-to-digital converter (ADC or AD converter). Finally the

signal is passed to a computer, which further processes, stores, and/or forwards it.

As illustrated in Figure 1, all modules can be considered as *one* array recording system provided they all have a common time reference. In this way we can obtain an arbitrarily scalable transducer recording system. As the figure also shows, this system can then be controlled through standard networking technologies, such as LAN (IEEE 802.3), Wi-Fi (IEEE 802.11), etc., by a computer, smartphone or other device supporting the control protocol. The system can be set up to trigger acoustically and data can be forwarded to a Network Attached Storage box, i.e. a network-accessible hard disk. When recording continuously, the network bandwidth will limit how many channels can operate simultaneously, so to achieve arbitrary scalability local storage must be used.

IV. IMPLEMENTATION

A. Analog filter and amplification

Designing custom hardware is not a trivial task, and we have sought to limit the exercise and make the system as generic as possible. This entails simple open design utilizing cheap off-the-shelf and easily replaceable components.

Attaching ultrasonic sensors to a standard computer requires some specialized amplification and power supply; some customized hardware has therefore been developed. This is not needed for some high-end microphones, which are sold with filters, pre-amplifiers and amplifiers; though for these devices signal conditioning might still be needed to convert voltage levels to those supported by the analog-to-digital converter. We have emphasized making the system as open as possible, which is a well-understood concept in the software world. Open source hardware, on the other hand, is not so widespread although it is not a new phenomena, see e.g. Ackerman (2008).

Our custom-designed hardware includes bandpass filters and analog amplification for each channel (Brandt, 2012). The analog bandpass filter for the system is a 2nd order Bessel biquad filter with 6 dB high-pass at 15 kHz and 6 dB low-pass at 150 kHz. The analog amplification on each input channel is programmable, which allows each channel to be attenuated or amplified individually from -20 dB to +60 dB. The hardware supports voltage signals up to $\pm 10V$ peak, so many different types of transducer are directly supported.

B. Microphone

The microphone used in our prototypes is the FG23329-PO7 from Knowles Acoustics with sensitivity, ≈ -73 dB re 1 VPa $^{-1}$ (Knowles Acoustics, 2012). Originally designed for cell phone use, this microphone is usable for ultrasonic reception as it is sensitive up to approximately 140 kHz. It is also cheap (circa 40 USD) and small (2.54 mm in diameter with a 1 mm acoustic

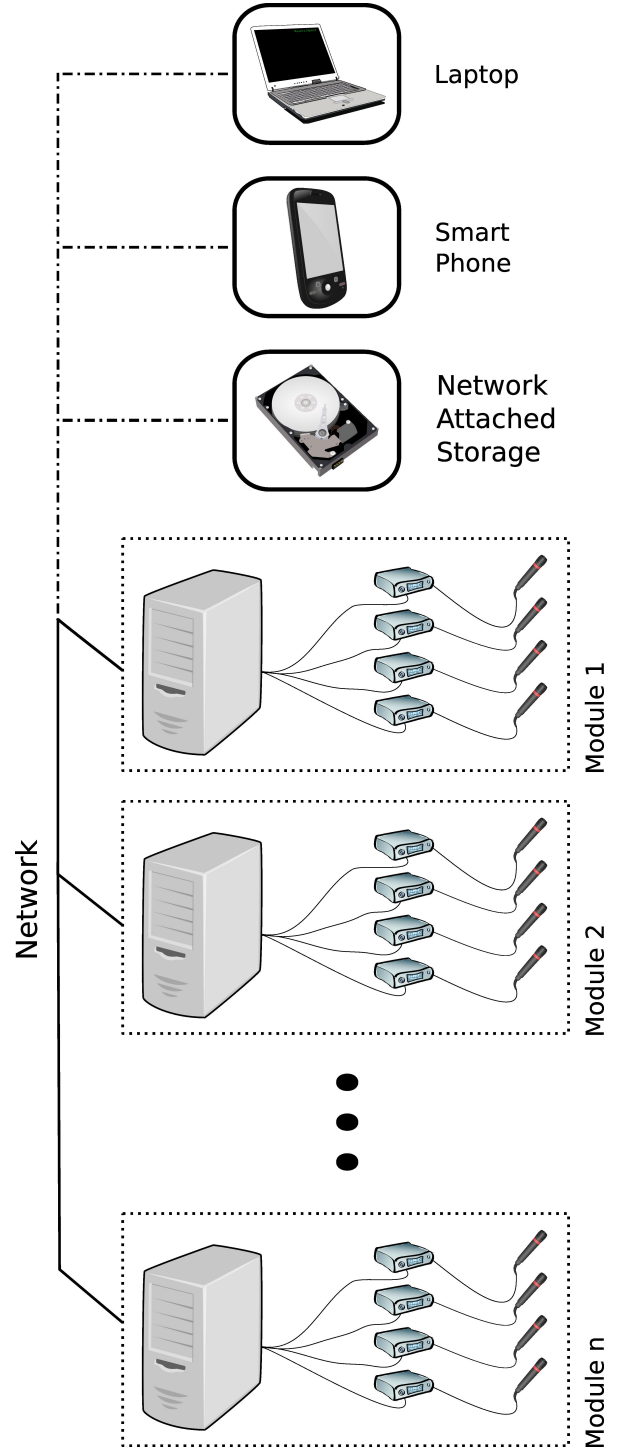


FIG. 1. (color online) Block diagram of a generic modular ultrasonic recording system. Each module consists of a computer connected to 4 filter/amplifier boxes each with an ultrasonic microphone attached. Note that even though the modules have a limited number of microphones, no restrictions are put on the total number of modules. The analog voltage from microphones is filtered and amplified and AD converted into digital information which is then handled by a computer. The recording system can be controlled through standard network protocols.

aperture), making it a near ideal choice given the criteria for the system.

As illustrated in Figure 2, 4 microphones are connected to the filter/amplifier box, using the connectors which have been used for all prototypes/versions of the analog part of the system. To make the system rugged we used IP68-certified connectors so that the joints are dust proof and submersible in water to below 1 m (Hirose Electric type LF07WBJ-6P and 6S). The microphone cables are commonly available shielded cable (Carol Brand type C0743A.21.10).

C. Analog to digital converter

For AD conversion we used a DAQ-2010 data acquisition board from Adlink Technology Inc. (Adlink Technology, Inc., 2012), chosen for its hardware specification and relatively open driver software. It provides 4 channels with an analog-to-digital conversion rate of 2 MSample/s/channel at 14 bit/sample. In addition, it has 8 digital input channels and two 12 bit digital-to-analog channels with a maximum sample rate of 1 MHz/s/channel. It comes with proprietary binary Linux drivers.

D. System computer

The smallest computer that provides the required PCI port for the AD board is a Single Board Computer with the Mini-ITX form factor (motherboard size of 17x17 cm) defined by VIA Technologies (Via Technologies, Inc., 2001). Figure 2 shows the first prototype. From the node computer the customized hardware is connected to the AD board through an off-the-shelf cable.

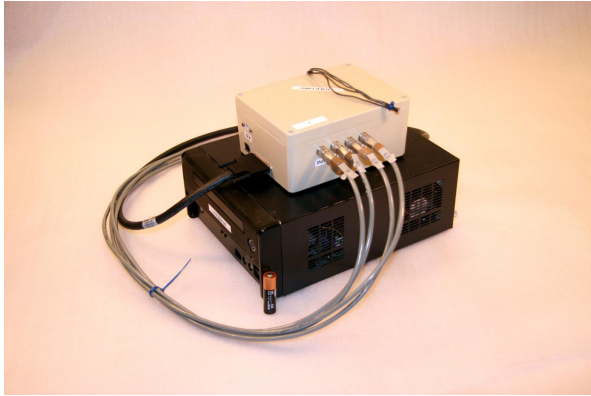


FIG. 2. (color online) First prototype, AA batteries show the scale. The computer and AD converter are in the black box, whilst the custom filter and amplifier hardware is in the light box. Four Knowles microphones are bundled and placed on top of the filter/amplifier box.

V. RESULTS

The system has been deployed in four different setups which demonstrate its capabilities, such as high number of channels, long-term recordings, synchronized playback/recording and operation under harsh conditions — recording right before the wet season in Panama with well above 90% relative humidity or recording during the Danish winter with sometimes freezing temperatures and winds as high as 20 ms^{-1} .

We deployed this prototype with several different transducers in the different setups and we took specific measures to deal with the large amount of data generated. In one setup we also attached a weather station to the prototype.

A. Multi-channel pre-triggered recording

The system was deployed to record *Noctilio leporinus* and *Micronycteris microtis*, flying in two outdoor flight rooms at the Smithsonian Tropical Research Institute's field station on Barro Colorado Island, Panama in March 2010. Three modules were used, so a total of 12 channels were available. Figure 3 shows the spectrogram of a 150 ms long clip taken out of a 1 minute *M. microtis* recording. 10 microphones were used; the eleventh channel recorded a synchronization signal for off-line synchronization with high speed video recordings.

N. leporinus and *M. microtis* have very different hunting behaviors. *N. leporinus* is a trawling bat, while *M. microtis* scans leaves for prey. Thus recording these two bat species poses different demands on the system. *N. leporinus* searches for prey on or over smooth water surfaces and once detected swoops down and catches the prey. To capture a whole hunting sequence requires a recording length of about 2–3 seconds with a 2 seconds pre-trigger time to be sure to capture the sequence from the very beginning. *M. microtis* can scan leaves for up to 2 minutes before finally taking the prey. This posed no problem for this system as it was a matter of increasing the pre-trigger buffer length to a couple of minutes.

B. Robust long-term multi-channel recording

Weatherproofing the electronics was achieved by mounting it inside a rain and dust proof IP65-rated box (Fibox type EK-W). The Knowles microphones used are weather resilient; pointing them downwards with a water guide gives them some water protection.

A four channel system was deployed outdoors from summer 2010 to summer 2011. There were only minor disruptions, and only one failing microphone, even though the system remained outdoors in Denmark throughout the winter. When the system is weather-proofed as described above, it is indeed seen to be very robust.

Long-term recording with the system was demonstrated using the same setup. The system was deployed

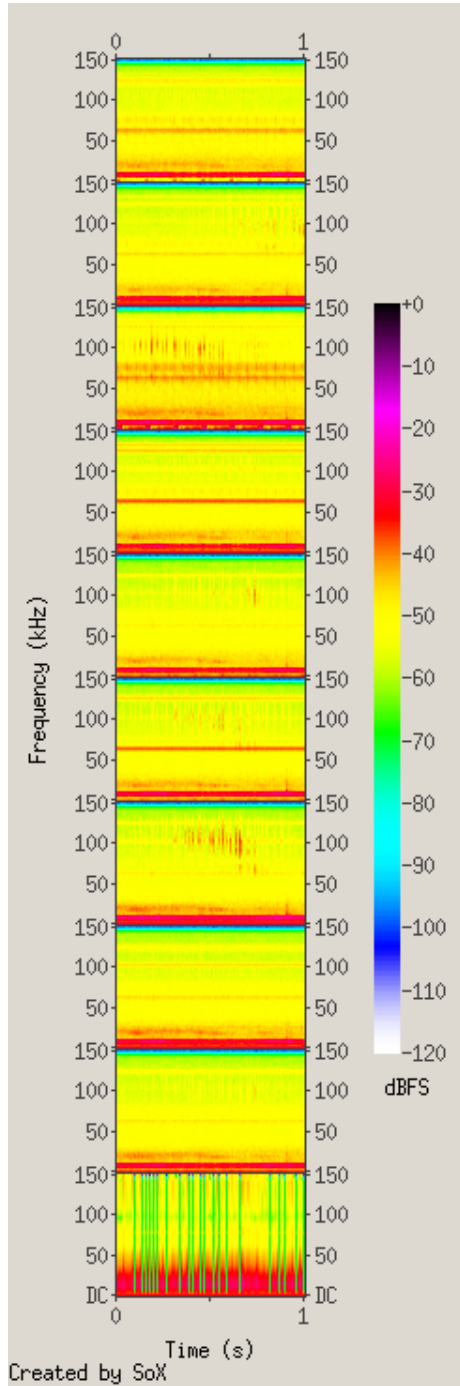


FIG. 3. Multi-module recording done in Panama in March of 2010 (session 9 recording 21). Note that only microphone 3 and 7 have clear reception. These are calls of *Micronycteris microtis* scanning a leaf in search of prey. The recording consists of 11 synchronous channels. The 11th channel recorded a synchronization signal.

for more than three months and recorded for approximately two of those months. Operational failures were due to accidental disconnection from the power grid by university ground staff.

One of the interesting recordings from the long-term setup was captured during the night between Sept. 26

and Sept. 27 2011. Figure 4 shows 2 seconds of processed data from the 2nd channel of this recording, 1 second long pre- and post-trigger buffers. The recording shows high frequency echolocation calls with main energy around 55 kHz interspersed with low frequency (down to ca. 20 kHz) social calls of a *Pipistrellus pygmaeus* (Barlow and Jones, 1997).

Long-term recordings generate large amounts of data, e.g. for a node with 4 channels sampling at 500 kHz:

$$4 \text{ channels} \cdot 2 \frac{\text{bytes}}{\text{channel}} \cdot 500\text{K} \frac{\text{samples}}{\text{seconds}} \cdot 10 \text{ hours} = 144 \text{ GB}$$

Analyzing such an amount of data manually would be very time-consuming, so automated methods are needed. We used a simple recording paradigm to reduce the amount of stored data: the system was set to a wakeup/shutdown cycle where it started recording at 8 PM and stopped at 6 AM. This timing was experimentally determined to cover the hunting period of the local bats.

When recording, only signals which exceed a triggering threshold are stored. The threshold was set at 5 times the noise floor of the topmost microphone. To trigger a recording, the threshold had to be exceeded for at least half a millisecond; at that moment the system stores the previous second's buffered data and continues recording until 1 second of silence occurs. More advanced triggers are being developed, for example acoustic profiles for a matched filter bank, which can then be used as acoustical triggers.

C. The echolocating robot

As the system readily supports synchronized playback and recording it lends itself to other applications. One of these was a sensor capable of echolocation implemented on a robot (Figure 5). The robot uses a Polaroid speaker and two Knowles microphones embedded in a 1.5 scale 3D print of a *Myotis daubentonii* head; the microphones are placed 1 cm inside each ear canal. The sensor successfully guided the robot to objects lying on a floor. Efforts are currently being made not only to find but also to identify these objects. This application demonstrates the flexibility of this AD/DA platform.

D. Calibration setup

The system has additionally been tested as a tool for calibrating microphones. Synchronization of input and output using the DAQ-2010 lends itself readily to linked emission and recording, which is not only useful for echolocation as described above but also makes averaging of data from many emission-recording cycles trivial. The frequency response of the emitter is recorded with a calibrated microphone in a reasonably anechoic place. By replacing the calibrated microphone with the microphone that needs to be calibrated, we can automatically

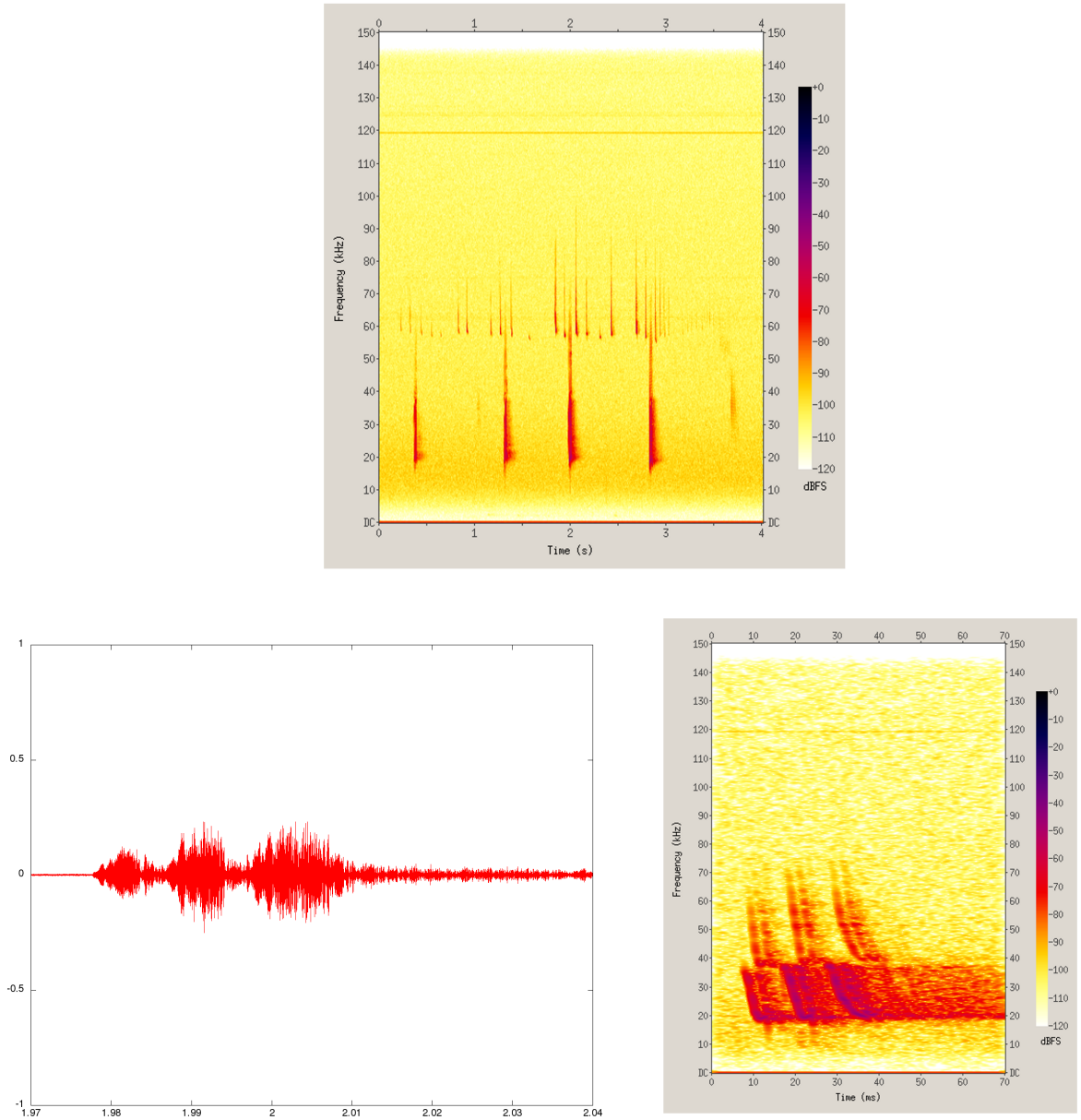


FIG. 4. (color online) These plots show one of the recordings made with an outdoor setup on the roof of a University garage (Sep. 26, 2011). The top figure is a spectrogram of the whole recording (Hanning window with 4096 points), showing echolocation calls interspersed with social calls. Bottom figures show a small segment of the recording, from 7.5 ms before 2 s to 10 ms after, left is the amplitude plot and right shows the spectrogram (Hanning window with 512 points). Both spectrograms have been digitally low-pass filtered with f_c at 150 kHz. Note that there are significant reflections within the three chirps, which probably come from the rippled roof of the garage.

acquire data to compute the transfer function of the microphone needing calibration.

The recording system will work with input from any transducer delivering voltages of ± 10 V peak, e.g. ultrasonic microphones from Knowles Acoustics, Avisoft, B&K, and G.R.A.S.

The analog output channels have a maximum voltage range of ± 10 V and have been used to drive a number

of different speakers. The output impedance was chosen to be 50Ω to be compatible with the input for our custom-designed power amplifier for Senscomp's Series 7000 (22 mm) Polaroid transducer (SensComp, 2012). The digital-to-analog converter has also been tested using a regular dynamic speaker and a ScanSpeak ultrasonic speaker from Avisoft.

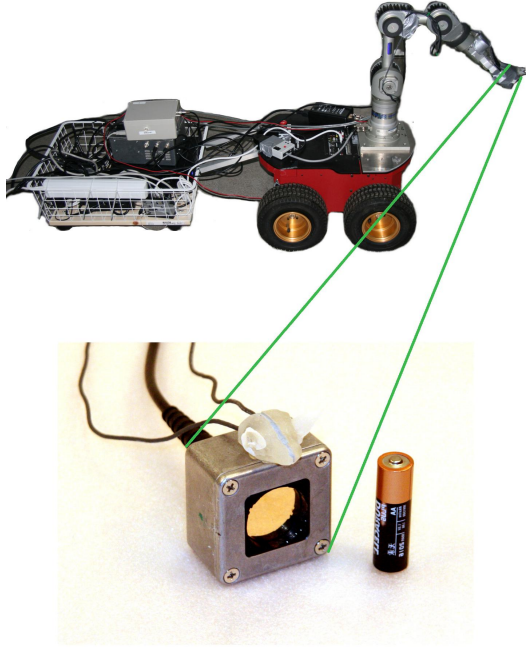


FIG. 5. (color online) Echolocating robot, a demonstration of synchronized playback and recording. The robot consists of two robots joined together — a Neuronics Katana 450 robot arm with 4 degrees of freedom while the mobile robot is a Pioneer 3-AT robot from Adept Technology Inc. A blowup of the binaural echolocation sensor can be seen below, scale is shown with an AA battery.

E. Easy upgrade

As a demonstration of the easy upgrade path offered by this design, we produced a second prototype which is portable with low power consumption. This prototype is based on the open source/hardware AD converter USB-DUXFAST (Incite Technology Ltd., 2012) and the DreamPlug computer (Globalscale Technologies, Inc., 2012a,b) which together offer a platform with very low power consumption and good technical specifications.

This second prototype can be seen in Figure 6. All custom-built components from the first implementation were reused with small modifications. The only substantive modification is a change to the connector cable between the custom analog box and the AD converter, necessitated by the different termination connector on the USB-DUXFAST.

This prototype was optimized for portability, low-power usage, and many channels, thus making it useful as a multi-channel recorder for use in the field, or as a module in a distributed ultrasonic recording system. There are, however, some limitations with this prototype compared to the first one: the maximum sample rate is 750 kHz with 4 channels instead of 2 MHz; the bit depth is 12 bit compared to 14 bit; and the USB-DUXFAST does not include a digital-to-analog converter so playback is not possible. None of these limitations are significant for

What (per ch.)	Avisoft	Proto. 1	Proto. 2
Sample rate	750 kHz	2 MHz	750 kHz
Bits per sample	12 bits	14 bits	12 bits
Variable gain	0-40 dB	-20-60 dB	-20-60 dB
Volume	240 cm ³	2200 cm ³	140 cm ³
Weight	260 g	1000 g	170 g
Power consumption	5.4 W	10.4 W	2.8 W
Price	\$2200	\$950	\$600

TABLE III. Three array systems are compared based on their per-channel parameters: a system based on an Avisoft AD converter which needs a computer running Microsoft Windows, the first prototype based on the Adlink AD converter and the second prototype based on the USB-DUXFAST AD converter. Power consumption calculation of the Avisoft array includes a laptop using 20 W.

the ultrasonic recording application.

Table III compares the two prototypes and a commercial system based on the Avisoft UltraSoundGate type 1216H AD converter.



FIG. 6. (color online) Portable, low-power prototype, scale shown with AA batteries. The computer is in the black and yellow box, while the AD converter is in the black and green box. The interconnection between the AD converter and filter/amplifier box still needs to be adjusted.

F. Logging other data

Each module includes a standard retail Single Board Computer, which will accept input from most available retail transducer hardware. As a demonstration of this, we sampled data from a weather station, the wireless Fine Offset WH1090 (Fine Offset Electronics, 2012), using the open source software *wwsr* (Pendec *et al.*, 2012). The recorded data includes time-stamped weather measurements such as wind speed, wind direction, precipitation, air pressure, temperature, and relative humidity, and is time-synchronized with the acoustic recordings.

Recording local weather conditions together with acoustical data enables us to correlate various climatic

parameters with bat activity as well as providing the basic information (temperature, pressure and relative humidity) needed to calculate the local speed of sound. The system also allows easy integration of other types of data, for example light intensity, cloud coverage, wind turbine blade speed, etc.

G. Advanced use of the system

Using a Unix-inspired operating system on the modules' local computers makes available a large set of advanced high-level tools and paradigms. As an example, consider operating system pipes, which abstract data handover between programs and readily handle infinite data streams in a sensible way. Using such a paradigm it becomes possible to separate acquisition, analysis, multiplexing, de-multiplexing, and forwarding of the streams.

Below is a short working code example of what this might look like. The vertical bar (|) indicates a pipe, which links the output of one program to the input of another, running independently in parallel. The example uses linked programs to detect signals, to convert recorded data into Microsoft Wave PCM (Microsoft, Inc., 1991) format, and to forward channel 1 and 3 to a different computer for further analysis. What follows is an actual working program chain; long command lines are split, for clarity, and the splits are marked with \ tokens:

```
$ adac -r 500e3 \
| tee raw_data \
| signal_detect \
| sox -r 500e3 -c4 -t s16 - -t wavpcm remix 1 3 \
| ssh -e none OTHER_HOST \
    'further_analysis > outfile'
```

The chain works like this:

1. **adac**, a custom program, takes one argument, namely the sample rate. It acquires samples from the AD converter at the specified rate, 500 thousand samples per second per channel in this case, and emits them on its output stream.
2. **tee**, a standard GNU coreutils program, is an input replicator. It saves one copy of the data stream to a file named **raw_data**, whilst sending a second copy forward to the next linked program.
3. **signal_detect**, a custom program, conducts **local** real-time analysis; this is where matched filters and other data selection criteria might be applied, and where threshold-triggering is implemented when that is used. Data judged interesting by this program is passed along the chain; uninteresting data is dropped (but a copy remains in the **raw_data** file from the previous step).
4. **sox** is a powerful audio processing program with more than 20+ years of usage and active development. Here it converts channels 1 and 3 of the (interesting) raw data to Wave PCM format. Note that **sox** needs to be told the sample rate (**-r**

500e3), number of channels in the input stream (**-c4**) and sample format (**-t s16**) i.e. little-endian signed 16 bit. This information is encoded in the Wave PCM file format header.

5. **ssh**, a standard communication program, connects to the remote computer **OTHER_HOST** and forwards the Wave PCM-encoded data stream to a custom remote analysis program **further_analysis**.

The complete command chain is built up piecewise, offering flexibility and choice to the user, while allowing rather complex analysis to be accomplished with a chain of individually-simple steps.

VI. DISCUSSION

The proposed system design has been tested in various settings, and the results show that it is also applicable for doing large-scale recording. The framework is based on an open-ended design which has allowed the development of two markedly different systems without changing either the basic design or most of the software components. Another hallmark of the current design is the ease with which the configuration of prototype 1 was changed from, for example, multichannel recording of short events like the 1–2 second echolocation sequence of a trawling *Noctilio leporinus*, to long-term recordings of bat activity over a whole season and finally to function as an input/output system deployed for two different purposes, robot echolocation and microphone calibration.

The ultrasonic recording system we propose is designed to achieve an unprecedented flexibility: as the interface is based on widely used open web standards, few limits are put on what type of equipment can communicate with the system. For example any system which has a web browser installed will be able to control and get feedback from the recording system — which implies, for example, that the system can be controlled by smartphones allowing flexible, wireless, universal access to the data. This has been tested by successfully running a terminal program on an Android smartphone. Simple preview, analysis, and control interfacing can be achieved by installing GNU Octave on the same type of smartphone; alternatively, more sophisticated analysis could be carried out on the module's local computer and the smartphone be used only for display and control.

A. Benefits of the openness

Due to the openness of the system, there are no limits or restrictions to possible modifications and additions. Any user interface could potentially be used to control the system. The stream format is publicly available, so anyone can implement and integrate a user interface that would work with the ultrasonic recording system. Below we outline some of the perspectives in such an open, modular, scalable, and cheap system.

When scaling the system to arbitrary size, keeping track of individual channels or modules of the system

becomes a book-keeping challenge. However, with an open framework everything is potentially programmable, and the system can be extended to deal automatically with this task. Electronically tagging microphones, filters and computers is straightforward and would allow system software to keep track of which components were in use in what combinations.

One useful possibility is the automatic determination of microphone array geometry, which is essential for computation of flight paths and is a major contributor to the effort of deploying a large-scale array. Using a small, loud impulsive sound source — such as an electrical spark — the geometry of an already-installed array can be calculated using time-of-flight computations to determine the positions of the (perhaps many) microphones while simultaneously calculating the positions of the sparks. Such software could be integrated into the array module local computers, making the array almost self-calibrating.

As shown by the tests we performed, the versatility of the system was sufficient to enable application to all of the experiments we intended. But an extra advantage of the open design is that it allows others to modify, remove or add components to the system and apply it straightforwardly to many other tasks which we have not yet anticipated.

B. Calibrated recording with interchangeable microphones

A second benefit of the possibility of electronically-tagged system components accrues if the tags can store acoustic calibration data. Each microphone could carry its own calibration information, as could each analog channel in each module. Plugging a microphone into a channel could then cause the automatic application of compensating (digital) signal processing so that a calibrated recording transfer function was realised independent of which microphone and channel were actually used. By this means, relatively inexpensive microphones might achieve the same recording tasks as expensive, stable, calibrated microphones.

A further question implied by this possibility is the stability of the microphone with aging and environmental conditions. The functionality of the Knowles microphones after being subjected to the Danish outdoor environment has been verified, but the stability of their transfer function over time and with use still needs to be determined. In particular, it is not known whether (or how much) the transfer function varies with temperature, humidity, and wind. Tagging microphones with their calibration data would go some way toward answering this question too.

C. Other possible applications

There is a number of other applications that this system could handle; some possibilities are outlined below.

Automatic flight tracking can be integrated in several different ways. In principle it should be sufficient to have 4 microphones placed in a non-planar arrangement, from

which full 3D source position can be calculated. A tetrahedral structure's spherical symmetry offers position accuracy independent of direction, which may be important for environmental monitoring where it is unclear from which directions animals may approach. Adding multiple 4 microphone sets would improve position estimation for distant targets.

As noted by Guarato and Hallam (2009), the call of a flying bat can be reconstructed with reasonable accuracy if the bat's position can be determined, its emission directivity pattern is known (or can be approximated) and it emits its broadband call towards the center of a suitable array of calibrated microphones. As the proposed system is an arbitrarily scalable array, it would be possible to place a sufficient number of calibrated microphones in the environment to allow call reconstruction.

VII. CONCLUSION

We have demonstrated a modular ultrasonic recording system design that is useful, inexpensive and versatile, and fulfills all our criteria for use in tasks involving recording or playback of ultrasonic signals for animals or animats.

The implemented system has been tested in a variety of application settings, and is able to record and emit ultrasound at sufficient rates and process the resulting data volumes. The use of open design allows for an easy upgrade path, distributed deployment, long-term recording and emission, low-cost implementations based on retail components and open source software/hardware.

This system allows exploration of new areas of the ultrasonic soundscape. Multi-channel long-term surveys become possible, making it feasible to obtain more accurate assessments of presence and activity of bats or other sound-emitting animals. The flexibility of the system may also lead to applications not yet anticipated.

When building tools for academic exploration, open source software and hardware provide transparency and ease of design sharing, thus increasing the possible proliferation of successful ideas.

Acknowledgments

Part of this project was funded by the EU 7th Framework Programme (FP7), through the ChiRoPing project (ICT-2007-1 STREP project 215370). A special thanks goes to the OSS community which has brought about many good and useful things, many of which have been used in this project.

We also thank the Smithsonian Tropical Research Institute for allowing us to use their facilities on Barro Colorado Island, and also the biologists who helped us with many of the recordings.

We thank Ali Shekarchi who assisted with the debugging and use of the echolocator and microphone calibration setups.

APPENDIX A: HOW TO BUILD A COPY OF THIS SYSTEM

Taking the steps listed below would duplicate one of the described modules (not necessarily in order)

- Fetch schematics and gerber files from Brandt (2012) and send them to a PCB production company.
- Order components on the Bill of Materials list.
- Solder the filter/amplifier.
- Order AD board and PC.
- Order microphones, microphone cables, microphone plugs.
- Assemble the device.
- Install GNU/Linux on the PC.
- Install the proprietary driver from Adlink.
- Fetch and install the `adac` software from Andreassen (2012).

The authors are happy to assist if issues arise.

- Ackerman, J. (2008). “Toward open source hardware”, U. Dayton L. Rev. **34**, 183.
- Adlink Technology, Inc. (2012). “DAQ-2010 data acquisition board”, URL http://www.adlinktech.com/PD/web/PD_detail.php?pid=12, date last viewed 10/9/2012.
- Ali, A., Yao, K., Collier, T., Taylor, C., Blumstein, D., and Girod, L. (2007). “Acoustic source localization using the acoustic ENSBox”, in *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*, 555–556 (IEEE).
- Allen, M., Girod, L., Newton, R., Madden, S., Blumstein, D., and Estrin, D. (2008). “VoxNet: An interactive, rapidly-deployable acoustic monitoring platform”, in *Proceedings of the 7th international conference on Information processing in sensor networks*, 371–382, IEEE Computer Society (IEEE Computer Society Washington, DC, USA).
- Andreassen, T. (2012). “Software used with the system: ADC driver, GNU/Linux setup files, signal detection program”, URL http://array.chiroping.org/long_term_recording_array-software.tar.gz, date last viewed 10/9/2012.
- Avisoft Bioacoustics (2012). “AD converter UltraSoundGate 1216H”, URL <http://avisoft.com/usg/usg1216h.htm>, date last viewed 10/9/2012.
- Baerwald, E. and Barclay, R. (2011). “Patterns of activity and fatality of migratory bats at a wind energy facility in alberta, canada”, *The Journal of Wildlife Management* **75**, 1103–1114.
- Barclay, R., Baerwald, E., and Gruver, J. (2007). “Variation in bat and bird fatalities at wind energy facilities: assessing the effects of rotor size and tower height”, *Canadian Journal of Zoology* **85**, 381–387.
- Barlow, K. E. and Jones, G. (1997). “Differences in songflight calls and social calls between two phonic types of the vesperilionid bat *pipistrellus pipistrellus*”, *Journal of Zoology* **241**, 315–324.
- Brandt, D. (2012). “Amplifier/filter schematics, gerber files, and system software”, URL http://array.chiroping.org/long_term_recording_array-hardware.tar.gz, date last viewed 10/9/2012.
- Bruderer, B. and Popa-Lisseanu, A. (2005). “Radar data on wing-beat frequencies and flight speeds of two bat species”, *Acta Chiropterologica* **7**, 73–82.
- Collier, T. (2011). “Wireless sensor network-based acoustic localization for studying animal communication in terrestrial environments”, Ph.D. thesis, University of California, Los Angeles.
- Dzal, Y., McGuire, L., Veselka, N., and Fenton, M. (2011). “Going, going, gone: the impact of white-nose syndrome on the summer activity of the little brown bat (*Myotis lucifugus*)”, *Biology Letters* **7**, 392–394.
- ecoObs GmbH (2012). “ecoObs batcorder products”, URL <http://www.ecoobs.com/cnt-batcorder.html>, date last viewed 10/9/2012.
- Fiedler, J. (2004). “Assessment of bat mortality and activity at buffalo mountain windfarm, eastern tennessee”, Ph.D. thesis, University of Tennessee, Knoxville.
- Fine Offset Electronics (2012). “WH1090 - Professional weather center with PC interface”, URL http://www.foshk.com/Weather_Professional/WH1090.htm, date last viewed 10/9/2012.
- Gillam, E., Ulanovsky, N., and McCracken, G. (2007). “Rapid jamming avoidance in biosonar”, *Proceedings of the Royal Society B: Biological Sciences* **274**, 651.
- Girod, L., Jamieson, K., Mei, Y., Newton, R., Rost, S., Thiagarajan, A., Balakrishnan, H., and Madden, S. (2006a). “Wavescope: a signal-oriented data stream management system”, in *Proceedings of the 4th international conference on Embedded networked sensor systems*, 421–422 (ACM).
- Girod, L., Lukac, M., Trifa, V., and Estrin, D. (2006b). “The design and implementation of a self-calibrating distributed acoustic sensing platform”, in *Proceedings of the 4th international conference on Embedded networked sensor systems*, 71–84 (ACM).
- Globalscale Technologies, Inc. (2012a). “Dreamplug – hardware specification”, URL <http://www.globalscaletechnologies.com/t-dreamplugdetails.aspx>, date last viewed 10/9/2012.

- Globalscale Technologies, Inc. (2012b). “Dreamplug – user guide”, URL http://dreamplug.googlecode.com/files/DreamPlug%20User%20Guide_US_v1.1.pdf, date last viewed 10/9/2012.
- Guarato, F. and Hallam, J. (2009). “Determining sound source orientation from source directivity and multi-microphone recordings”, *Bioinspired Applications in Artificial and Natural Computation* **5602**, 429–438.
- Hristov, N., Betke, M., and Kunz, T. (2008). “Applications of thermal infrared imaging for research in aeroecology”, *Integrative and Comparative Biology* **48**, 50–59.
- Incite Technology Ltd. (2012). “The usbdxfast AD converter”, URL http://www.linux-usb-daq.co.uk/prod2_duxfast, date last viewed 10/9/2012.
- Jensen, M. and Miller, L. (1999). “Echolocation signals of the bat *Eptesicus serotinus* recorded using a vertical microphone array: effect of flight altitude on searching signals”, *Behavioral Ecology and Sociobiology* **47**, 60–69.
- Kirchner, W. and Röschard, J. (1999). “Hissing in bumblebees: an interspecific defence signal”, *Insectes sociaux* **46**, 239–243.
- Knowles Acoustics (2012). “Microphone type FG23329-P07 datasheet at Farnell”, URL <http://www.farnell.com/datasheets/49984.pdf>, date last viewed 10/9/2012.
- Kunz, T., Arnett, E., Cooper, B., Erickson, W., Larkin, R., Mabee, T., Morrison, M., Strickland, M., and Szewczak, J. (2007). “Assessing impacts of wind-energy development on nocturnally active birds and bats: A guidance document”, *The Journal of Wildlife Management* **71**, 2449–2486.
- Luke, H. (1999). “The origins of the sampling theorem”, *Communications Magazine, IEEE* **37**, 106–108.
- Madsen, P. and Wahlberg, M. (2007). “Recording and quantification of ultrasonic echolocation clicks from free-ranging toothed whales”, *Deep Sea Research Part I: Oceanographic Research Papers* **54**, 1421–1444.
- McIlroy, M., Pinson, E., and Tague, B. (1978). “Unix time-sharing system forward”, *The Bell system technical journal* **57**, 1902.
- Microsoft, Inc. (1991). *Microsoft Windows multimedia programmer’s reference* (Microsoft Press), URL <http://download.microsoft.com/download/9/8/6/9863C72A-A3AA-4DDB-B1BA-CA8D17EFD2D4/RIFNEW.pdf>, date last viewed 10/17/2012.
- Newton, R., Girod, L., Craig, M., Madden, S., and Morrisett, G. (2008). “Wavescript: A case-study in applying a distributed stream-processing language”, Technical Report, Massachusetts Institute of Technology.
- Pendec, M., Skafte, S., Pribyl, A., Jarvis, B., Zvonar, L., and Zitny, P. (2012). “Wireless Weather Station Reader”, URL <http://meteolinger.lowlevel.cz/>, date last viewed 10/9/2012.
- Petterson Elektronik AB (2012). “Petterson Elektronik AB ultrasound detectors”, URL <http://www.batsound.com/?p=2>, date last viewed 10/9/2012.
- Rich, B. (1995). “Clarence leonard (kelly) johnson”, *National Academy of Sciences – Biographical Memoirs* **67**, URL <http://www.nap.edu/html/biomems/cjohnson.pdf>, date last viewed 10/23/2012.
- SensComp (2012). “Seris 7000 Transducer”, URL <http://www.senscomp.com/pdfs/series-7000-transducer.pdf>, date last viewed 10/9/2012.
- Shannon, C. (1948). “A mathematical theory of communication”, *Bell System Technical Journal* **27**, 379–423 and 623–656.
- Sorainen, E. and Rytönen, E. (2002). “Noise level and ultrasound spectra during burring”, *Clinical oral investigations* **6**, 133–136.
- Spiesberger, J. (2004). “Geometry of locating sounds from differences in travel time: Isodichrons”, *The Journal of the Acoustical Society of America* **116**, 3168.
- Surlykke, A. and Kalko, E. (2008). “Echolocating bats cry out loud to detect their prey”, *PLoS one* **3**, e2036.
- TASCAM, a division of TEAC America, Inc. (2012). “Ultrasound recorder hd-p2”, URL <http://tascam.com/product/hd-p2/specifications>, date last viewed 10/9/2012.
- Via Technologies, Inc. (2001). “New mini-itx main-board specification white paper”, URL http://www.via.com.tw/en/downloads/whitepapers/initiatives/spearhead/ini_mini-itx.pdf, date last viewed 10/17/2012.
- Wildlife Acoustics (2012). “Ultrasound recorder sm2bat+”, URL <http://www.wildlifeacoustics.com/products/ultrasonic-monitoring/sm2bat>, date last viewed 10/9/2012.
- Wu, J. and Stojmenovic, I. (2004). “Ad hoc networks”, *Computer-IEEE Computer Society* **37**, 29–31, an introduction to mesh concepts can be found at http://en.wikipedia.org/wiki/Wireless_mesh_network (date last viewed 10/9/2012).

ARTICLE

Semi-automatic long-term ecological surveying: A case study with bats

The article has been submitted for review.

Semi-automatic long-term ecological surveying: A case study with bats

Tórrur Andreassen^a Annemarie Surlykke^a
John Hallam^b

May 30, 2013

^aInstitute of Biology
University of Southern Denmark, SDU, DK-5230
Odense M, Denmark

^bMaersk-McKinney-Moeller Institute
University of Southern Denmark, SDU, DK-5230
Odense M, Denmark

Abstract

As automatic ultrasonic surveillance becomes common, automatic analysis is increasingly important. We developed an automatic method using Simple Vector Machine (SVM) models that are able to separate bat events from non-bat events. Thresholding and the SVM method reduce our original 236 GiB acoustic dataset to a mere 162 MiB of good bat-calls recorded at full bandwidth, sample rate 500 kHz, with an accuracy of qualitatively estimated at circa 70 %.

The automatic method provided recordings of bats that have not been observed in the area before, as well as an abundance of social calls. We discuss the future applications including even higher accuracy in determining bat calls and exploiting the multiple microphones to localize individual bats.

1 Introduction

In the past decade worries about decline of biodiversity have fostered growing interest in technology for environmental monitoring, using surveys of animals as diverse as insects, birds and bats, which can automate species recognition based on acoustic communication [Blumstein et al., 2011]. For bats and

birds, special concerns have been linked to the growing number of large scale wind turbine parks [Kunz et al., 2007b].

Bats are protected and some species are endangered; loss of foraging grounds and roosting places poses a serious threat to them [Cryan et al., 2012]. Major new building projects not only disturb local bats during construction, but may also add to the decline in suitable habitat. The University of Southern Denmark, where we are situated, is undergoing major re-construction with plans for tripling the built-up area and introducing substantial changes to the surroundings. We are interested to know what effect such changes may have on our local bat populations.

In the above and many other cases bat surveillance is an important field. Many methods of surveying are currently in use: carcass counting [Fiedler, 2004, Barclay et al., 2007], radar tracking [Bruderer and Popa-Lisseanu, 2005, Kunz et al., 2007a], infrared thermography tracking [Hristov et al., 2008] and passive acoustical measurements, usually with one or more bat-detectors [Ahlén et al., 2007, Baerwald and Barclay, 2009]. Each of these methods has some drawbacks: carcass counting is done too late and does not reveal the bats' flight behavior. Radar tracking and stereo infrared thermography can give suitable data, at least for large species, but availability and price make them infeasible for wide deployment. Multiple bat-detectors could, with some effort, be used to count automatically passing bats and estimate their species — but unless they are synchronized one cannot easily combine information from the independent detectors, for example to localize an individual bat precisely relative to the recording sites. Thus a synchronized multi-channel solution is to be preferred. Furthermore, to obtain a fair picture of local bat activity one needs to survey over a long period, ideally continuously; if that is impractical, carefully timed short-term surveys may suffice.

In this paper we only consider acoustical surveillance, but we investigate automated long-term multi-channel recordings. This approach allows us to record bat activity over months, and to correlate bat behavior over time to several parameters including change of season, day/night, sunset/sunrise, lunar luminance, and weather data. One can also study the distribution of social versus echolocating calls, and in principle identify sequences which predict particular behaviors (such as foraging) and see how these are distributed over time.

Acoustical surveyors, especially ultrasonic surveyors, are limited today by the available equipment. This problem concerns both the area that can be surveyed and the duration over which the survey can be performed. To ameliorate these shortcomings we developed an inexpensive scalable array recording system using commodity-computing components [Andreassen et al., res]. The system can be deployed for months while automatically recording and

saving data on ultrasonic signals in its vicinity. The system is generic and adaptable to a wide range of other applications; this deployment of the system was partly to carry out a long-term survey of our local bats, but also partly to test and demonstrate its robustness. The key technical problem raised by our approach is what to do with the potentially huge volume of data collected.

2 Materials and Methods

2.1 Study site

As mentioned above, our goals were both to assess bat activity but also to test the novel prototype recording array, so convenience of access was important. Therefore we placed the system at a site on the Odense campus of the University of Southern Denmark ($55^{\circ}21'59.63''$ N, $10^{\circ}25'54.10''$ E) with low anthropogenic activity, high bat activity, power grid access, and convenient physical access. The campus is located outside the city of Odense in Denmark in an open area with patches of old forest, large open fields and small streams. The array was placed on the roof of a garage building, about 5 meters off the ground, close to a small old forest where we had often observed bats of various species flying along the tree-line and inside the forest.

2.2 The acoustic recording system and its setup

Four miniature microphones, type FG23329-PO7 (Knowles Acoustics, Itasca, Illinois, USA), were mounted on a tetrahedral structure with 30 cm sides, which was positioned on the roof the garage building. The tetrahedron can be seen in the left image of figure 1. The omnidirectional microphones were directed downwards and protected by a water-guide, see right image in figure 1. The recording system comprises a standard PC running GNU/Linux equipped with a PCI-bus analog-to-digital board DAQ-2010 (Adlink Technologies inc., Taipei City, Taiwan). This board is capable of simultaneously recording 4 analog channels with a sample rate of 2 MHz per channel. The microphones interface to the analog-to-digital converter board via custom-made signal-conditioning and programmatically controllable amplification electronics. The recording system components fit comfortably in a sturdy IP65 compliant plastic box (Fibox type EK-W). Power to the system was supplied directly from the grid to avoid changing batteries. The system is configured to record four synchronized channels with a 500 kHz sampling rate per channel at 14 bits precision. See [Andreassen et al., *ress*] for further

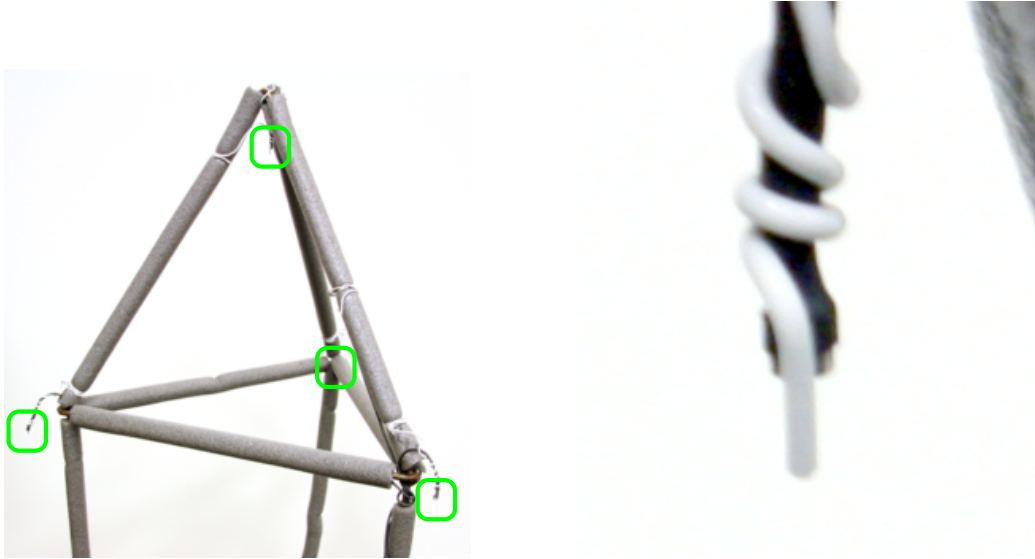


Figure 1: **Left:** The microphone assembly: a tetrahedron mounted with four Knowles microphones with water-guides. The microphones are picked out by green rounded rectangles. **Right:** A close-up of the topmost microphone, showing its water-guide (the wire that projects below the bottom of the microphone).

technical information on the array recording system.

2.3 Recording paradigm

We configured the system to wake up and start recording at 8:00 PM. This was set through the Real Time Clock on the computer motherboard. The system automatically switched itself off at 6:00 AM through the operating system scheduler. Upon return from power-failure the previous wake-up/shutdown cycle is recovered.

The system was configured with a repeating trigger and a two-second pre-trigger buffer. The trigger was activated when the average energy of the signal exceeded a threshold for more than half a millisecond. The trigger is deactivated again when the average energy has been below threshold for more than two seconds. The recording system saves the pre-trigger buffer and up to and including the final two seconds of sub-threshold samples to permanent storage each time the trigger is activated. Each such recording generates one data file on the recording system computer.

2.4 Event marking

Upon the completion of this survey, the system had recorded 236 GiB of data. The volume of data is the key technical problem with our approach: such an amount of data requires a systematic (and automatic, if possible) approach to analyze. Our objective was to discover and classify as many bat calls as possible in the dataset, with a high detection success, and a low false positive rate.

The goal of automatic processing is to identify interesting segments of the recorded signals, which we shall call *events*, within which bat calls are likely to be found. Note that in general not every bat call will result in an event, since not every call will pass the event-energy threshold we set. The idea is to identify likely candidates first, then search for quiet calls later once good examples have been identified. The likely candidates can then be used to extract the weaker signals, for example with cross-correlation.

2.4.1 Event detection and marking

In the first step of event detection, the data was corrected for DC shift, then squared and convolved with a normalized Gaussian moving average filter of length $100\ \mu s$ ($\mathbf{G}$) in order to smooth the power series and construct a measure of the local intensity of the signal at each time. This will not remove interesting events as those are all much longer than the filter length.

The above can be expressed mathematically like this

$$\mathbf{e}(\mathbf{t}) = \text{diag}[(\mathbf{x} - x_{avg}\mathbf{1})(\mathbf{x} - x_{avg}\mathbf{1})] * \mathbf{G} \quad (1)$$

where $x_{avg}\mathbf{1}$ is the vectorized average for the channel and $*$ represents convolution.

Events were detected based on energy thresholding. Those samples of the energy signal that exceed a chosen threshold mark the presence of candidate events. Events are then defined as starting at a mark preceded by a certain interval ($500\ \mu s$ is used here) without marks. The end of an event is defined by a mark followed by a certain interval without marks. In this way, a cluster of high energy values, each generating a mark, are grouped into a single event. Furthermore, each event has a front and back “porch” (region) where the signal energy is below the event threshold.

The final form of an event is an interval of the recorded data that begins $500\ \mu s$ before the start mark and ends $500\ \mu s$ after the end mark. Further analysis of these events will classify them as comprising noise alone, or one or more bat calls. Events shorter than $1100\ \mu s$ (including the front and back porches) were ignored.

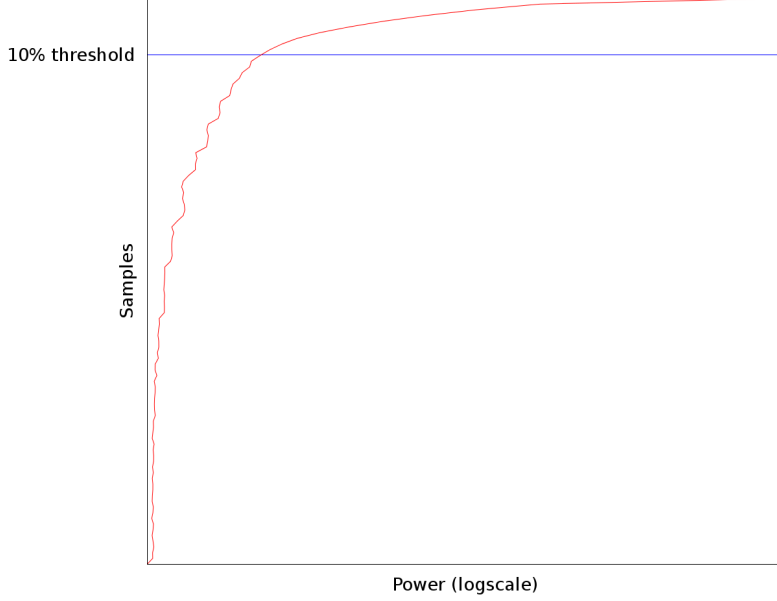


Figure 2: A typical sorted cumulative density plot. The x-axis is sample value; the y-axis runs from zero to one and shows for given abscissa the proportion of samples in the set exceeded by that value.

An empirically estimated energy threshold was determined by selecting those portions of the recordings where the signal power was in the top $\Theta\%$. This was found by sorting the samples in a recording into ascending order and setting the sample at the $1 - \Theta\%$ percentile in the order as the threshold. This is closely related to finding the cumulative distribution function. Figure 2 shows a typical sorted cumulative density of a recording. Formally the threshold was chosen to be:

$$\text{th} = \text{cumdensity}^{-1}(1 - \Theta) \quad (2)$$

We want the inverse because we are mapping a specific density Θ to an energy sample value. In our analyses we used $\Theta = 0.1$, i.e. 10%.

2.5 Event classification

Given the event extraction method, the problem becomes to one of classification. We define a set of quantitative features that capture relevant aspects of the properties of bat call spectrograms and use the vectors of their values as inputs to a suitable classification algorithm. There are many ways to go about this; preliminary research suggested that the well-established and

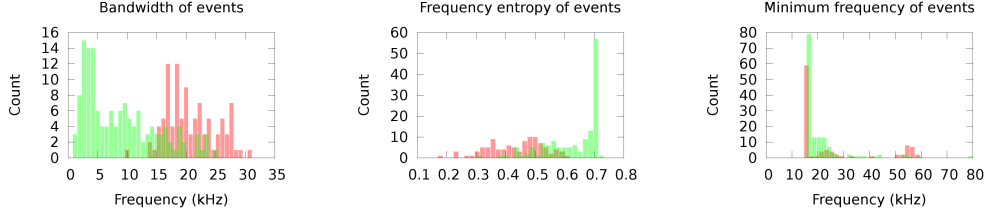


Figure 3: How three of the parameters or quality measures — bandwidth, energy and minimum frequency — are divided amongst bat (red) and non-bat (green) events. The less overlap the better separation the quality measure carries.

well-understood Support Vector Machine (SVM) [Boser et al., 1992] would be suitable for separating events into bat and non-bat classes. We used the LIBSVM [Chang and Lin, 2011] implementation of the method, which is a library that supports training and applying support vector machine models to a dataset. For the two-class case that we are considering, the SVM model can be envisioned as a hypersurface that separates a parameter space into two hypervolumes, one volume for each class.

2.5.1 Event quality measures

The parameter space in our case is constituted by the features we choose to describe events: the event quality measures. To classify the events we used a combination of temporal and spectral analyses. Bat calls are characterized by their relatively short duration and stereotypical spectrographic structure [Schnitzler and Kalko, 2001]. Consequently, features such as duration, energy, bandwidth and so forth might be expected to provide good discriminants. After some experimentation, we found the measures described below to be good classifier features. Three of them are plotted in figure 3 and give a feel for how well they separate the data. The quality measures were combined in a composite bat-call classifier described later. First, we formally define the quality measures used.

Duration The duration of event j was calculated in milliseconds as

$$d_j = \frac{\text{end}(ev_j) - \text{start}(ev_j)}{sr/1000} \quad (3)$$

where sr is the recording sample rate; $\text{start}(\cdot)$ and $\text{end}(\cdot)$ return the indices of the first and last samples belonging to the event supplied as argument.

Average power The average power of event j was calculated as the sum of squares of the samples comprising the event divided by the duration of the event:

$$W_j = \frac{1}{d_j} \sum_{i=\text{start}(ev_j)}^{i=\text{end}(ev_j)} ev_j(i)^2 \quad (4)$$

where $ev_j(i)$ is the i^{th} sample of event j .

Minimum and maximum frequency The most energetic frequency content was determined in a similar manner to the marks in the time domain which delimit events. The FFT of the event is constructed so that each frequency bin is always 50 Hz, this makes events more spectrally comparable. The resulting frequency bins are thresholded to mark the top 10%. Adjacent marked bins are merged into the same cluster if they were less than 300 Hz apart. The minimum $f_j^{\min}$ and maximum $f_j^{\max}$ were determined by the start and end marks of the largest spectral cluster.

Bandwidth Bandwidth follows directly from the minimum and maximum detected frequencies:

$$bw_j = f_j^{\max} - f_j^{\min} \quad (5)$$

Spectral entropy In order to detect the bat calls even under noisy conditions, we devised a feature that highlights the much more orderly time-frequency structure of bat calls relative to noise. This enabled us to detect bat calls even when masked by moderate noise in the same bandwidth. Entropy is a generic statistical measure [Shannon and Weaver, 1948] that is high for unordered data and low for ordered data. As noise tends to have a broad unordered spectrum, entropy is a potentially good measure for separating bats from noise.

The spectral entropy was calculated by first determining the probability distribution of spectral power. This was done by creating 5 kHz frequency bins, summing the power in each bin and dividing by total power of the event:

$$P_j(m) = \frac{1}{d_j W_j} \sum_{m=1}^M EV_j(m)^2 \quad (6)$$

where M is the number of frequency bins used computing the Fourier transform of event j , and the value of the i^{th} bin of the transform is $EV_j(i)$. The entropy of the event is then calculated as:

$$E_j^{\text{freq}} = - \sum_{m=1}^M P_j(m) \log P_j(m) \quad (7)$$

2.5.2 Training the model

Classification works by building a model based on vectors of the quality measures of manually classified events to determine an appropriate separatrix function [Boser et al., 1992]. The events are treated as labeled points in a high-dimensional space, and the classifier training algorithm determines a surface that separates the two clouds of labeled points, assuming this to be possible. The SVM algorithm in particular also maximizes the distance from the separatrix to the members of the two classes [Boser et al., 1992], producing thereby the “best” separating surface in the sense of allowing maximal clearance between the surface and the class members. The distance to the separatrix is a measure of how “bat-like” an event is.

Building the classifier requires labeled data, i.e. samples of input grouped into each of the classes to be learned. This data was prepared manually, thus: the events were randomized, the quality measures were calculated for all the events, these and accompanying spectrogram, spectrum and amplitude plots were visually inspected, and events which were clearly identifiable as bat calls that exceeded 20 dB signal-to-noise ratio were marked as bat events. During this process many non-bat events were encountered and marked as such, to provide the training set entries for the non-bat class. In total the training set consisted of 99 bat call events and 141 non-bat events, 240 in total. The feature value vectors and their labels were provided to the LIBSVM training algorithm to construct the SVM classifier.

As the individual feature value ranges are quite disparate when compared to each other, feature rescaling is often recommended to improve the classification model’s accuracy. Three classifiers were built using three scaling approaches: no scaling, relative linear scaling, and customized absolute scaling. In the first case, the raw feature values were used unaltered. Relative linear scaling finds the global maximum and minimum values of each feature in the entire dataset, and uses that to scale the feature component in the

SVM input parameter vector to the range -1 to 1 . This scaling depends on the data set available for training (hence “relative”) since it will differ for different data sets. A model trained on one set will thus not be applicable to another dataset if the minimum and/or maximum differ from the original.

The third scaling type rescales each feature based on our *a priori* knowledge of the expected range of values for that feature. It is independent of the actual feature values in the training data set, hence “absolute”. In this approach, there are three cases. First, if the feature has a fixed minimum and maximum (for instance, spectral entropy lies between 1 and $\log M$) then it is scaled linearly to ± 1 . If the feature has a fixed lower bound but a soft upper bound — that is, the *a priori* minimum value is never crossed while the maximum might be exceeded (though that is unlikely) — the feature value is rescaled using a logistic squashing function (sigmoid) to map the minimum to -1 and the soft maximum to 0.9 . Outliers are the compressed into the $(0.9, 1]$ range. If the feature’s maximum and minimum are both soft, it is rescaled using the same squashing function to map the minimum and maximum to ± 0.9 respectively.

The performance of the three scaling approaches can be compared using the receiver operating curve plots for the classifiers built from those feature vectors. The greater the area under the curve, the better is the performance of the classifier. The ROC curves for unscaled, relative linear scaling, and customized absolute scaling are shown in figure 4. These ROC curves were calculated from the labeled training set, and suggest that the raw feature values are such that feature rescaling provides no advantage for this problem.

The actual model used by the SVM is based on a Radial Basis Function (RBF) kernel. The RBF is a real-valued function whose value depends only on the distance between a feature vector and a specified center for the function. It is the most common kernel function used with SVMs [Chang and Lin, 2011]. The RBF layer in LIBSVM has two parameters, *gamma* and *cost*. The optimal value for these is found by running *svm-grid* program and visually adjusting *gamma* and *cost* to optimize correct separation into correct classes.

When assessing the accuracy of the SVM classifier, N -fold cross-validation is used. The manually classified training events are divided into N blocks. N SVM classifiers are constructed by training using $N - 1$ blocks of the data, while the final block is used for testing the performance of the model constructed on the other $N - 1$. Each block is therefore used as testing set, while the rest is used as the basis for the model, giving N independent estimates of the classifier’s accuracy and an indication of the variability of accuracy as a function of training data.

The goal of the classifier is to identify, with as high an accuracy as possi-

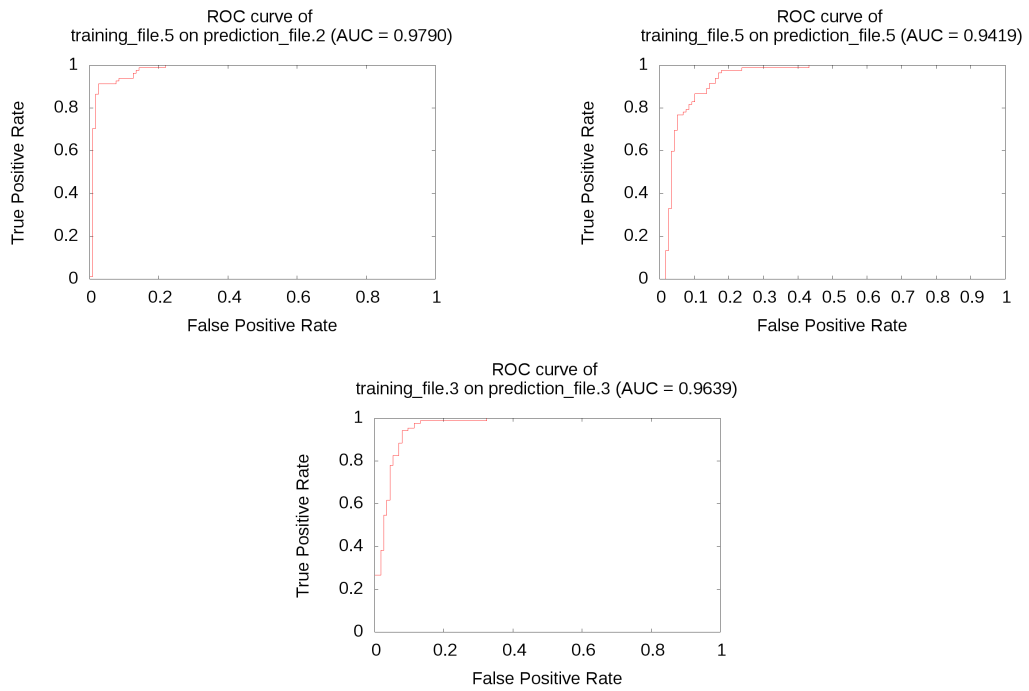


Figure 4: ROC curves comparing the accuracy of SVM models based on unscaled, relative linear scaled and custom absolute scaled SVM parameters. **Left:** Unscaled, **Right:** Relative linear scaling, **Bottom:** Custom absolute scaling (see text).

ble, events that contain bat calls; in case of errors, we prefer false negatives over false positives. The best classifier obtained from training and cross-validation is used to classify all available events into candidate bat-calls, meriting attention from a biologist, and non-bat events (which may in fact contain bat calls, of course). From that point onward, in the current system design, data analysis requires human judgment.

3 Results

3.1 General notes and observations

The system was first set up for automatic data capture from 20:00 to 6:00 on June 23rd 2011. The top plot in figure 5 shows the distribution of recorded data volume over time. The main panel shows data volume plotted with respect to day and time-of-night counted using 2 minute time bins; the top panel shows the total data volume for that day in bytes. The bottom plot shows a histogram of the sizes of data capture files; notice that most files are four seconds in length (pre-trigger buffer plus post-trigger silence), but there is a long tail of much bigger files shown in more detail in the inset plot. Overall, the system captured 236 GiB of raw data in a total of 11059 files of varying sizes.

Unfortunately the system was accidentally disconnected from the grid on June 27th 2011. This happened again on July 6th 2011 and this time the system remained off-line for almost a month, reflected in figure 7 by the many empty days during July. These disconnections always happened during working hours, i.e. between 8:00 h and 17:00 h, so the system was never interrupted during a recording run. The result is that we have sporadic recording days in June and July. When the system was restarted on August 1st 2011 it remained operational until the survey period was over on September 27th 2011. As we are interested in patterns over time, we have chosen to focus on the last period only, i.e. August 1st through June 27th, in the results and analysis below.

Although we recorded four channels in this survey, this paper only considers data from channel one as that is sufficient for our analyses here. There are improvement possibilities and other analyses available with multi-channel recordings, which are discussed in more detail in Section 4.

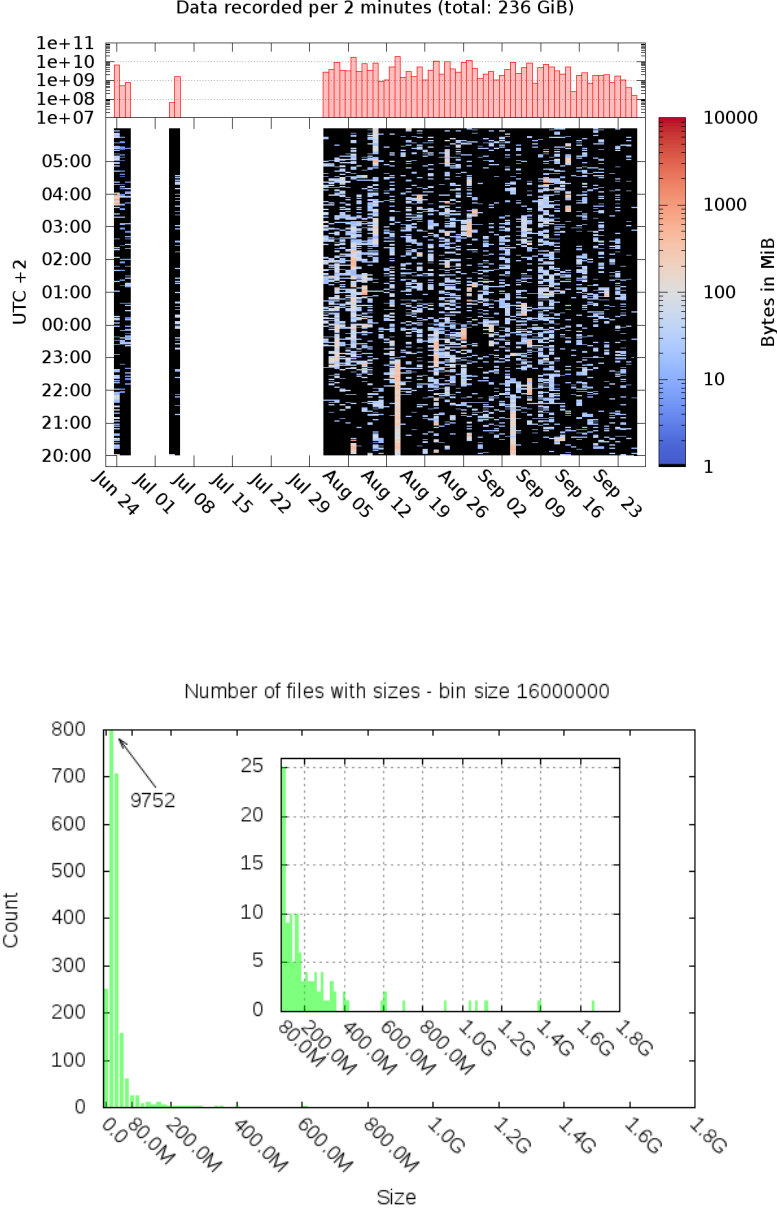


Figure 5: **Top:** Data distribution over time and size. The histogram along the top shows daily totals in number of bytes. **Bottom:** Histogram of recording file sizes. It shows that most of the recordings were the size of two post-trigger buffers, 4 seconds or roughly 4 MiB. There is a long “tail” on the histogram of a few long recordings. This part is zoomed in by the inset plot, with the size range limited to 80 MiB to 1.8 GiB.

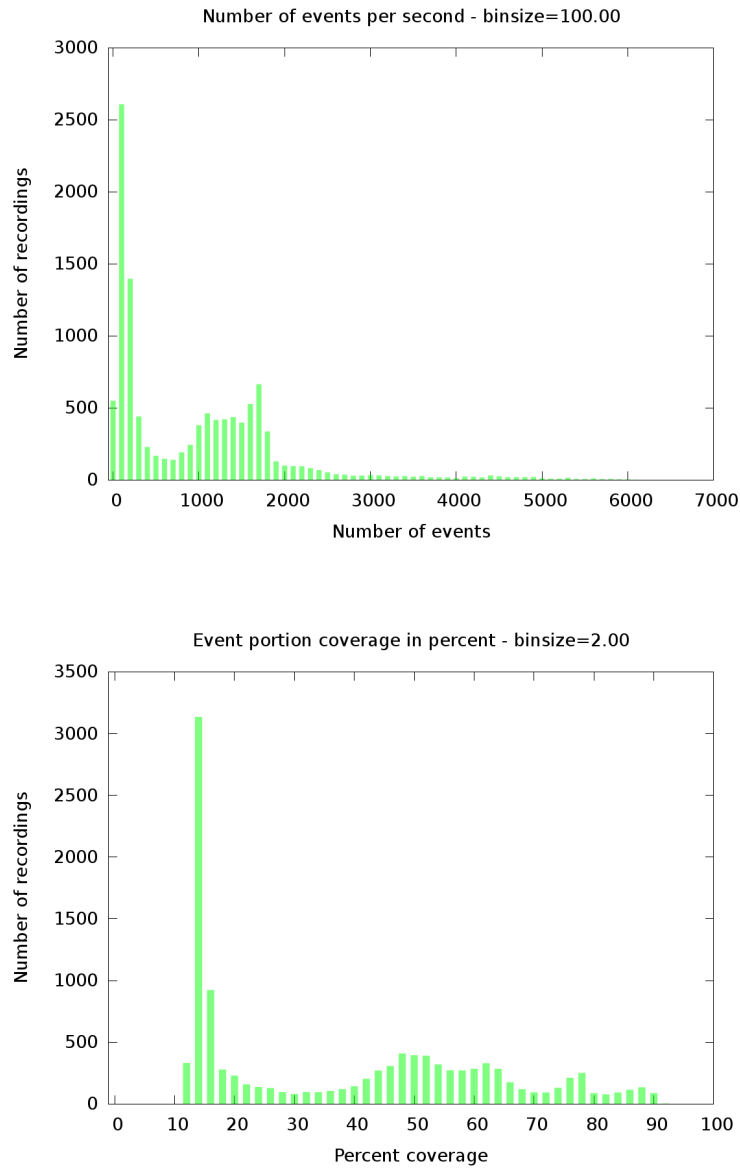


Figure 6: **Top:** Histogram of number of events per recording per second. **Bottom:** Histogram of event coverage in each recording file.

3.2 Event marking

After applying the threshold and event marking methods described the data can be viewed as an event distribution with a total of 11,777,675 events. This number is inflated by the fact that a number of test signals were also included in this pre-analysis. These automatically-identified regions of the recordings are where we hope to find bat calls. Figure 6 shows how the data separates into number of events per second (top panel) and the percentage of the whole recording that an event covers (bottom panel). The top panel shows that most of our recordings contain fewer than 500 events, while the bottom panel shows that in most files the events cover less than 15 % of the samples.

A plot of the event data analogous to the temporal distribution of recordings is shown in figure 7 (top panel). The color of each block represents the number of events occurring during each 2 minute bin. Note that this overview map plots all events, bat and non-bat alike: at this stage, we have not classified the events. When narrowing down to only look at events found during August through September we consider a total of 8,413,249 events. There are clear high incidence areas both in the recordings plot (figure 5 (top panel)) and the event plot (figure 7 (top panel)) — for instance early evening on August 14th — and there are also areas which have few events, such as the later night and early morning of that same session. The lines plotted on the figure are the times for nautical sunset and sunrise (gold colored) and end and beginning of nautical twilight (goldenrod colored) for the given day. The distribution of events seems to peak within the inner sunset-sunrise lines, apart from bright areas such as during August 10th morning, 14th evening and September 4th evening.

Applying the SVM classifier to the detected events results in the bottom plot in figure 7, a total of 115,456 events. The remaining events are all candidate bat events, their number is considerably smaller, and many of the bright areas that were present in the top plot have disappeared. Inspection of the events in the previously-bright areas reveals that the majority of them are noise recordings, and correlation with weather data for the region suggests that the noise is caused by local rain. The SVM classifier rejects a large proportion of the rain-noise events, though given their density and the accuracy of the classifier it is unreasonable to expect it to eliminate all of them.

There are several interesting features to notice in these distribution plots. Much of the noise has been removed, but regions of relatively high activity remain. Sunset and sunrise appear to have a substantial impact on activity since the candidate bat events tend to occur between the sunset and sunrise boundaries. Throughout the recording period there appears to be both high

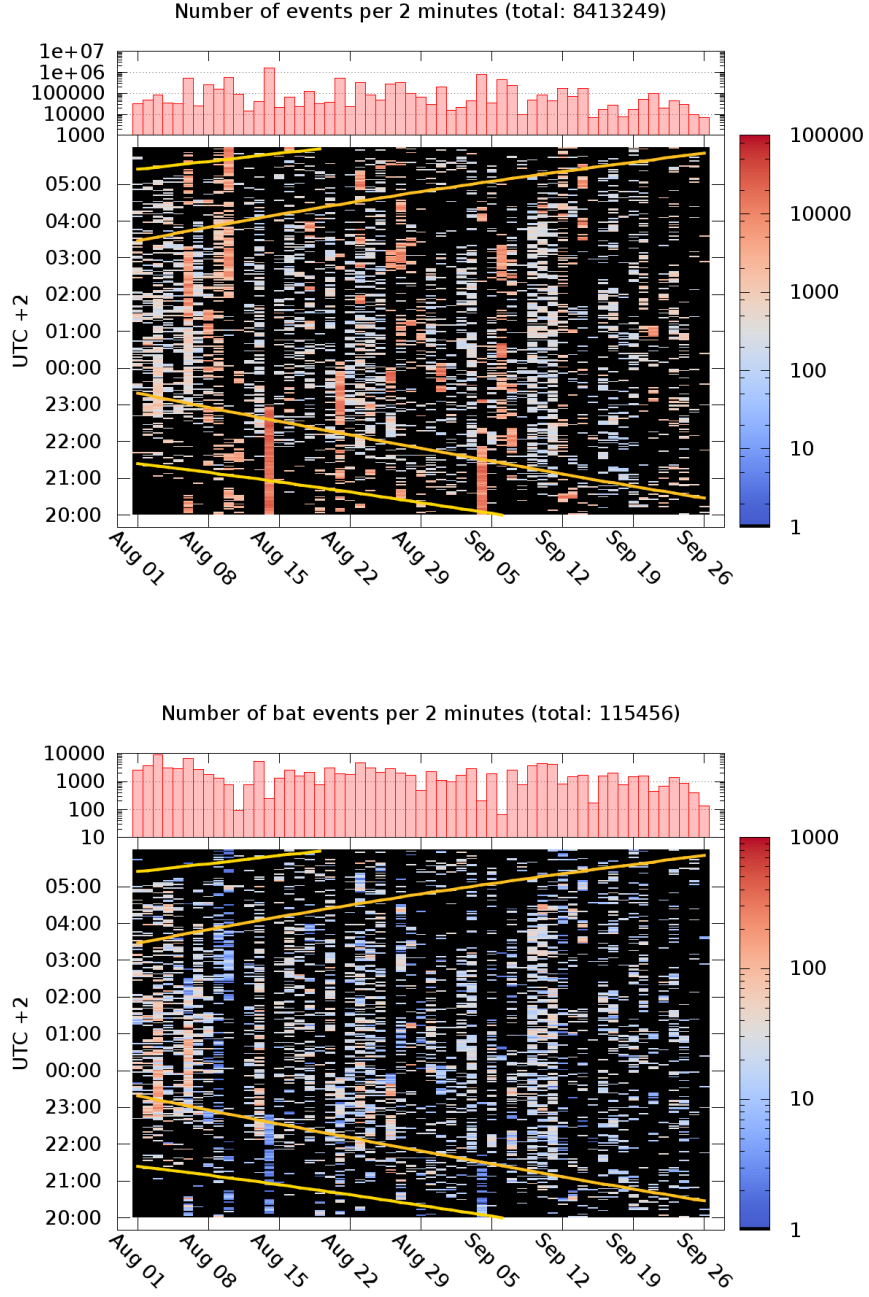


Figure 7: **Top:** Temporal distribution of events. **Bottom:** temporal distribution of classified bat events. Notice that the color scales are different by two orders of magnitude.

	Aug 3	Aug 11	Aug 24	Aug 26	Sep 4	Sep 10	Sep 14
<i>Eptesicus serotinus</i>	145	0	34	58	0	20	0
<i>Nyctalus noctula</i>	12	0	31	6	0	3	1
<i>Pipistrellus pygmaeus</i>	16	10	38	13	0	47	31
<i>Pipistrellus nathusii</i>	0	0	2	0	0	5	1
<i>Myotis daubentonii</i>	0	0	5	0	0	1	0
Non-bat event	16	0	36	104	31	50	62

Table 1: Manually identified bat events in a sample of events labeled as candidate bat events by the custom scaled SVM classifier.

activity days and low activity days. The start and end of the recording time window could be expanded in the end of the period (from around September 1st) to better capture the longer nights in late summer and fall. There appear to be both high activity days and low activity days.

The events that are plotted in the lower plot in figure 7 are those that the SVM classifier classed as candidate bat calls: as yet we do not know what proportion of them really are bat calls. To address this question, we inspected a number of events by looking at their spectrograms, signal-to-noise ratio compared to the rest of the recording, and duration. Because of time restrictions we chose to look at 7 different days and only every n^{th} event in each day so as to keep the event count to be inspected below 200 per day. In total we reviewed just fewer than 1400 events. Table 1 shows the results from this tally where we have identified a number of different bat species with high degree of confidence. Figure 8 shows some of the typical events counted in the table.

Clearly, there are still many non-bat events present in reduced collection, but when looking chronologically through the events, it is obvious that there are clusters of non-bat events, presumably once again related to rain. We are convinced that these can fairly easily be spotted and removed by improved automatic analysis.

Another interesting feature of this tally is that we found evidence of two species that have not been registered before in the survey area, namely *P. nathusii* and *M. dasycneme* [Baagøe, 2001].

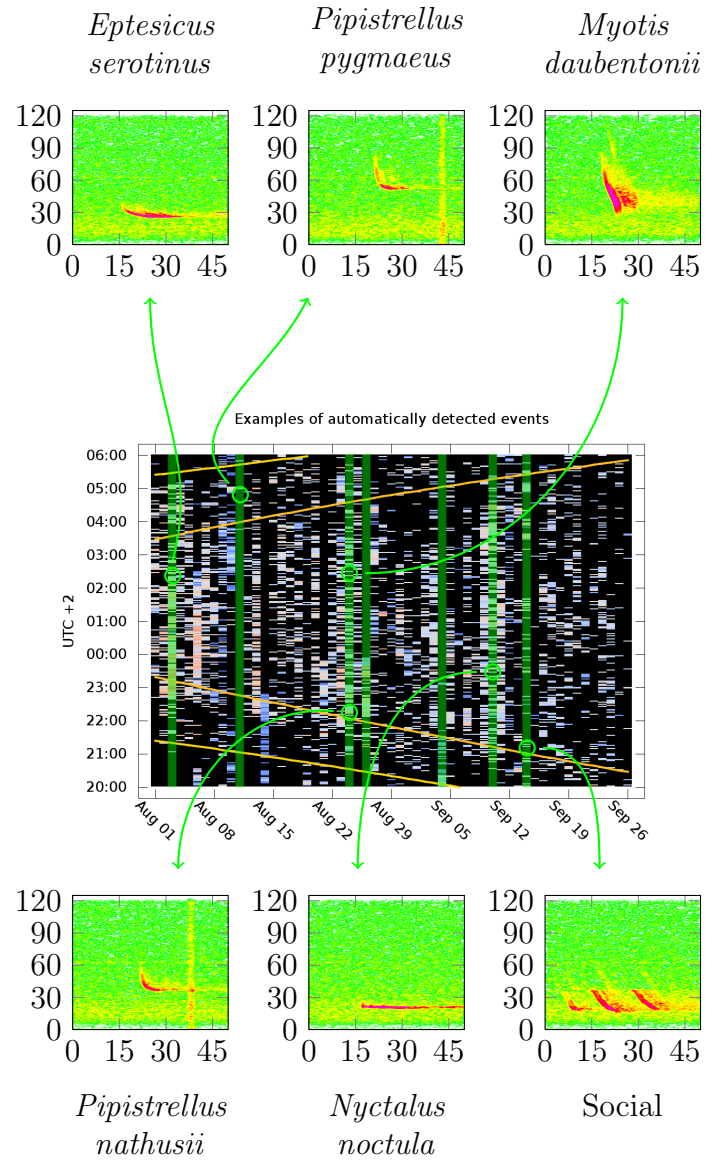


Figure 8: Example of typical events classified during the tally.

4 Discussion

The survey described, and the methodology used to effect it, suggest a number of interesting discussion points, both biological and technological. We discuss some of them below.

Firstly, from a technical viewpoint, the data processing has been designed to favor simplicity whenever that provides adequate functionality. This reflects the emphasis of the current work on proving the concept of this approach rather than squeezing the last drop of power from it. For example, the single-threshold approach to event marking tends to label isolated regions of a recording file, and the mark clustering then aggregates nearby marks to make the events. A dual-threshold approach — a high threshold for the initial mark, and a lower threshold to determine the start and end of the candidate event — might be preferred because this would include low intensity calls from the beginning and end of a sequence of calls from a passing bat.

The support vector machine was chosen as a robust and versatile classifier, but there may be techniques better suited to the specific class discrimination problem posed by our data. In particular, adjusting the classifier parameters depending on the presence or absence of rain would be sensible, given the large number of rain events highlighted by the initial marking process: the prior distribution of the two classes is markedly different on rainy and dry days.

The question of scaling the feature vector for the classifier is also open. For the training data set, unscaled data clearly results in a very good classifier (figure 4, left); absolute rescaling causes a small loss of performance but incorporates more biological knowledge about the reasonable range of values for the chosen features (figure 4, bottom). It remains to be seen whether it is better to use the absolute scaling trained classifier when running on the bulk of events, as we did here, or to keep to the unscaled data trained classifier that performs better on the training set. One might also envisage refinements of the chosen features, or further features that would increase the discriminating power of the classification process. Finally, given the initial biological survey carried out here, the available pool of labeled training data can be expanded using all the candidate bat events examined by the biologists, since their true class is known after inspection [Obrist et al., 2004].

The analysis presented here used data from a single channel in the recorded data and also treated events as isolated occurrences. Given the four channels recorded, one can in principle recover the spatial position of the sound source relative to the recording array (using time-difference-of-arrival techniques [Chan and Ho, 1994]). This would allow the system automatically to

recover spatial data for the bats detected. Correlating event marking between the four channels might improve event detection. Temporal patterns in the single channel event stream could also be extracted: bats do not generally emit isolated calls, but structure their call patterns temporally to reflect the activities in which they are engaged. One might therefore recover data not only about which bat species were present but also what they were doing, e.g. insects caught using buzz sequence counting. These analyses entail technical difficulties relating to signal quality and signal matching, but there are many approaches in the literature worth trying ([O’Farrell et al., 1999, Gorresen et al., 2008, Skalak et al., 2012]).

Finally, it would be advantageous to augment the survey recording equipment with a small weather station able to monitor at least local temperature, humidity, illumination and rainfall. The former two data are needed to compute an accurate value for the speed of sound — for calculating spatial data — while all data are also relevant to bat behavior. Adding such a sensor suite and synchronizing it to the acoustic recordings presents no difficulties, given the open design of the recording system [Andreassen et al., *ress*].

From a biological viewpoint, there are a number of interesting ecological results from the survey. First, the variety of species sampled at the recording site includes frequent appearances by individuals of at least two species have not been registered in the area before [Baagøe, 2001]. Thus this technique proves very promising to survey species that are rare, and thus difficult rarely sampled in short random surveys or transects. One advantage of a longitudinal survey such as this is that one potentially observes everything that occurs, rather than relying on careful judgment and some luck to be surveying at the right time, which will prove useful for estimating the true occurrence of less common species, and also determine areas of importance for passing events as for example migration.

Another, surprising, observation from the subset of survey data inspected so far is that social calls were frequent. Compared to the large number of echolocation studies, there are only few systematic studies of social calls in the field in temperate areas ([Lundberg and Gerell, 1986, Barlow and Jones, 1997a, Barlow and Jones, 1997b, Russo and Jones, 1999, Russ and Racey, 2007, Georgiakakis and Russo, 2012]). Recordings of social calls emitted by bats may provide data for assessing cryptic diversity and investigating reproductive strategies and foraging ecology. Further, social calls reveal areas of importance for reproduction and roosting. Occupancy may help evaluate reaction to anthropogenic impact, not only through exploitation of land resources, but also through climate change and other changes to the habitat.

One may consider the extent to which the survey is a reliable indicator of bat population and activity. A full answer to this question awaits a more

complete analysis of the collected data, but initially we can say that the observed bat activity certainly underestimates what is really present. The automatic processing focuses on minimizing false positives, but may generate many false negatives to be caught by more sophisticated automatic processing not yet implemented. Thus, while inspections suggest that around 30% of candidate bat-call events proposed by the current signal-processing chain will turn out to be misclassified noise, there are certainly bat calls present in the large bulk of data not highlighted for human examination.

Of course, no survey methodology is unbiased [Flaquer et al., 2007]. In our case, the principal bias favors loud bats since they will record calls that more easily pass our energy thresholds. This is partially an artifact of the simple signal processing chain we have presented. Having extracted all the loud bats, one could continue and search for quieter ones in the remaining data — especially if given models of the types of call made by species of interest. Such a search could also be automated, but would be much more time-consuming than the process adopted for this paper. On the other hand, technology aside, louder bats will always be detectable in a larger volume than quieter bats and this will inevitably bias population estimates in favor of the louder species. However, given spatial data for each call, extracted as outlined above, one could straightforwardly normalize for volume and derive bat population density measurements in units of individuals per cubic meter. Such analysis would be hard or impossible to do without the array recording technique underpinning the present survey.

A rather more subtle (absence of) bias is suggested by the observation that bat calls are detected in our survey very close to the end of periods of rain. Given that weather can be a very local phenomenon and that the nearest weather recording station is 2 km from our survey site, we can only speculate; but it appears that bats may either fly immediately after rain or indeed during rain. Careful analysis of the data from rainy periods may reveal bat calls partially masked by the rain noise. It could perhaps be that the generally held belief that bats do not fly in rainy weather reflects a preference of biologists (for not recording in the wet) rather than of bats. The serious point is that the methodology we demonstrate avoids this kind of potential observer bias, simply by observing all the time. The long-term automatic survey not only permits sampling of data that are un-biased by weather conditions, but is also less biased by difficulty of access to a potential bat habitat.

Correlation with weather and ephemeris information suggests that further interesting observations may be possible. For instance, the data plotted in the lower panel in figure 7 shows nights with little bat activity followed by nights with much activity (e.g. September 8–11). Correlation with weather data

(temperature and wind, principally), lunar phase and cloud cover (measured some 15 km from the survey site) suggests that the quiet nights in question were cloudless and therefore cool with bright moonlight, while the following nights were cloudy and warmer. This type of analysis would be facilitated by the local weather sensing addition mentioned above. It is also apparent in the same plot that the bat activity varies across the sunrise and sunset periods in a systematic way, and one could construct plots of the density of bat calls (in total or per species) against the time relative to sunrise and sunset to investigate those phases of the bat’s night.

As a final comment, the principal difficulty of the survey methodology we advocate here is the sheer volume of data generated. A full season of recording bats, from say April to October, would generate nearly 1 TiB of recorded data. Storing such a data volume is trivial nowadays, but analyzing it by hand is infeasible. The automated processing developed here, with the improvements discussed above (and others), can reduce this data volume to something manageable for a busy biologist. For specific questions such as species density it may be possible to automate the answers completely. Our survey demonstrates the growing possibility of carrying out ecologically interesting comprehensive long-baseline studies of acoustically active animals without dedicating the whole of one’s working life to the project.

5 Conclusion

The goals of this survey were two: to investigate and document local bat population behavior in the vicinity of the University of Southern Denmark; and to test and demonstrate that an automatic array recording system could be deployed long-term and the resulting voluminous data could be analyzed effectively.

Our inexpensive scalable array recording system [Andreassen et al., *ress*] was used for the survey, and was at the survey site for just over three months. Despite some unexpected power interruptions, the system functioned largely as intended and we were able to capture two continuous months’ worth of recorded data totaling nearly 240 GiB. Automatic processing of the data successfully identified around 115,000 candidate bat “events” so-labeled by a support vector machine classification algorithm, of which we expect some 70% to be recordings of actual bat calls. This represents a small percentage of the total volume of data which requires human judgment for further analysis. A number of plausible improvements to the automatic processing may eliminate more of the data from initial consideration; conversely, improvements may also retrieve less conspicuous bat calls from the data rejected by the current

analytic methods. We conclude that automatic long-term recording is both feasible and practical, given our experience with this system.

From a biological perspective, the survey also reveals some interesting results. Examination of a sample of events from specific days shows a variety of bat species observed at the survey site, including a number of rare or infrequently observed species. A surprisingly large proportion of the total calls inspected turned out to be social calls rather than echolocation calls. There are suggestions of correlations between bat activity and local weather: busy activity on warm nights especially following cloudless moonlit nights, and bat activity surprisingly close to the end of rain showers. These tentative observations require further detailed analysis to confirm them, and the data collected supports such analyses.

References

- [Ahlén et al., 2007] Ahlén, I., Bach, L., Baagøe, H. J., and Pettersson, J. (2007). Bats and offshore wind turbines studied in southern scandinavia. Technical report, Swedish Environmental Protection Agency.
- [Andreassen et al., res] Andreassen, T., Surlykke, A., Hallam, J., and Brandt, D. (In press). Ultrasonic recording system without intrinsic limits. *The Journal of the Acoustical Society of America*.
- [Baagøe, 2001] Baagøe, H. J. (2001). Danish bats (mammalia: Chiroptera): Atlas and analysis of distribution, occurrence, and abundance. *Steenstrupia*, (26, 1):1–117.
- [Baerwald and Barclay, 2009] Baerwald, E. F. and Barclay, R. M. (2009). Geographic variation in activity and fatality of migratory bats at wind energy facilities. *Journal of Mammalogy*, 90(6):1341–1349.
- [Barclay et al., 2007] Barclay, R., Baerwald, E., and Gruver, J. (2007). Variation in bat and bird fatalities at wind energy facilities: assessing the effects of rotor size and tower height. *Canadian Journal of Zoology*, 85(3):381–387.
- [Barlow and Jones, 1997a] Barlow, K. E. and Jones, G. (1997a). Differences in songflight calls and social calls between two phonic types of the vesper-tilionid bat pipistrellus pipistrellus. *Journal of Zoology*, 241(2):315–324.
- [Barlow and Jones, 1997b] Barlow, K. E. and Jones, G. (1997b). Function of pipistrelle social calls: field data and a playback experiment. *Animal behaviour*, 53(5):991–999.

- [Blumstein et al., 2011] Blumstein, D. T., Mennill, D. J., Clemins, P., Girod, L., Yao, K., Patricelli, G., Deppe, J. L., Krakauer, A. H., Clark, C., Cortopassi, K. A., et al. (2011). Acoustic monitoring in terrestrial environments using microphone arrays: applications, technological considerations and prospectus. *Journal of Applied Ecology*, 48(3):758–767.
- [Boser et al., 1992] Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM.
- [Bruderer and Popa-Lisseanu, 2005] Bruderer, B. and Popa-Lisseanu, A. G. (2005). Radar data on wing-beat frequencies and flight speeds of two bat species. *Acta Chiropterologica*, 7(1):73–82.
- [Chan and Ho, 1994] Chan, Y. and Ho, K. (1994). A simple and efficient estimator for hyperbolic location. *Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on]*, 42(8):1905–1915.
- [Chang and Lin, 2011] Chang, C.-C. and Lin, C.-J. (2011). Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27.
- [Cryan et al., 2012] Cryan, P. M., Jameson, J. W., Baerwald, E. F., Willis, C. K., Barclay, R. M., Snider, E. A., and Crichton, E. G. (2012). Evidence of late-summer mating readiness and early sexual maturation in migratory tree-roosting bats found dead at wind turbines. *PloS one*, 7(10):e47586.
- [Fiedler, 2004] Fiedler, J. (2004). *Assessment of bat mortality and activity at Buffalo Mountain Windfarm, eastern Tennessee*. PhD thesis, University of Tennessee, Knoxville.
- [Flaquer et al., 2007] Flaquer, C., Torre, I., and Arrizabalaga, A. (2007). Comparison of sampling methods for inventory of bat communities. *Journal of Mammalogy*, 88(2):526–533.
- [Georgiakakis and Russo, 2012] Georgiakakis, P. and Russo, D. (2012). The distinctive structure of social calls by hanák’s dwarf bat pipistrellus hanaki. *Acta Chiropterologica*, 14(1):167–174.
- [Gorresen et al., 2008] Gorresen, P. M., Miles, A. C., Todd, C. M., Bonaccorso, F. J., and Weller, T. J. (2008). Assessing bat detectability and

- occupancy with multiple automated echolocation detectors. *Journal of Mammalogy*, 89(1):11–17.
- [Hristov et al., 2008] Hristov, N., Betke, M., and Kunz, T. (2008). Applications of thermal infrared imaging for research in aeroecology. *Integrative and Comparative Biology*, 48(1):50–59.
- [Kunz et al., 2007a] Kunz, T., Arnett, E., Cooper, B., Erickson, W., Larkin, R., Mabee, T., Morrison, M., Strickland, M., and Szewczak, J. (2007a). Assessing impacts of wind-energy development on nocturnally active birds and bats: A guidance document. *The Journal of Wildlife Management*, 71(8):2449–2486.
- [Kunz et al., 2007b] Kunz, T. H., Arnett, E. B., Erickson, W. P., Hoar, A. R., Johnson, G. D., Larkin, R. P., Strickland, M. D., Thresher, R. W., and Tuttle, M. D. (2007b). Ecological impacts of wind energy development on bats: questions, research needs, and hypotheses. *Frontiers in Ecology and the Environment*, 5(6):315–324.
- [Lundberg and Gerell, 1986] Lundberg, K. and Gerell, R. (1986). Territorial advertisement and mate attraction in the bat pipistrellus pipistrellus. *Ethology*, 71(2):115–124.
- [Obrist et al., 2004] Obrist, M. K., Boesch, R., and Flückiger, P. F. (2004). Variability in echolocation call design of 26 swiss bat species: consequences, limits and options for automated field identification with a synergetic pattern recognition approach. *Mammalia mamm*, 68(4):307–322.
- [O’Farrell et al., 1999] O’Farrell, M. J., Miller, B. W., and Gannon, W. L. (1999). Qualitative identification of free-flying bats using the anabat detector. *Journal of Mammalogy*, pages 11–23.
- [Russ and Racey, 2007] Russ, J. and Racey, P. (2007). Species-specificity and individual variation in the song of male nathusius pipistrelles (pipistrellus nathusii). *Behavioral Ecology and Sociobiology*, 61(5):669–677.
- [Russo and Jones, 1999] Russo, D. and Jones, G. (1999). The social calls of kuhl’s pipistrelles pipistrellus kuhlii (kuhl, 1819): structure and variation (chiroptera: Vespertilionidae). *Journal of Zoology*, 249(4):476–481.
- [Schnitzler and Kalko, 2001] Schnitzler, H.-U. and Kalko, E. K. (2001). Echolocation by insect-eating bats: We define four distinct functional groups of bats and find differences in signal structure that correlate with

the typical echolocation tasks faced by each group. *Bioscience*, 51(7):557–569.

[Shannon and Weaver, 1948] Shannon, C. E. and Weaver, W. (1948). A mathematical theory of communication.

[Skalak et al., 2012] Skalak, S. L., Sherwin, R. E., and Brigham, R. M. (2012). Sampling period, size and duration influence measures of bat species richness from acoustic surveys. *Methods in Ecology and Evolution*, 3(3):490–502.