Studio report: Linux audio for multi-speaker natural speech technology

Charles FOX, Heidi CHRISTENSEN and **Thomas HAIN**
Speech and Hearing
Department of Computer Science
University of Sheffield , UK

charles.fox@sheffield.ac.uk

Abstract

The Natural Speech Technology (NST) project is the UK's flagship research programme for speech recognition research in natural environments. NST is a collaboration between Edinburgh, Cambridge and Sheffield Universities; public sector institutions the BBC, NHS and GCHQ; and companies including Nuance, EADS, Cisco and Toshiba. In contrast to assumptions made by most current commercial speech recognisers, natural environments include situations such as multi-participant meetings, where participants may talk over one another, move around the meeting room, make non-speech vocalisations, and all in the presence of noises from office equipment and external sources such as traffic and people outside the room. To generate data for such cases, we have set up a meeting room / recording studio equipped to record 16 channels of audio from real-life meetings, as well as a large computing cluster for audio analysis. These systems run on free, Linux-based software and this paper gives details of their implementation as a case study for other users considering Linux audio for similar large projects.

Keywords

Studio report, case study, speech recognition, diarisation, multichannel

1 Introduction

The speech recognition community has evolved into a niche distinct from general computer audio and Linux audio in particular. It has its own large collection of tools, some of which have been developed continually for over 20 years such as the HTK Hidden Markov Model ToolKit [Young et al., 2006][1]. We believe there could be more crosstalk between the speech and Linux audio worlds, and to this end we present a report of our experiences in setting up a new Linux-based studio for dedicated natural speech research.

In contrast to assumptions made by current commercial speech recognisers such as Dragon Dictate, natural environments include situations such as multi-participant meetings [Hain et al., 2009], where participants may talk over one another, move around the meeting room, make non-sentence utterances, and all in the presence of noises from office equipment and external sources such as traffic and people outside the the room. The UK Natural Speech Technology project aims to explore these issues, and their applications to scenarios as diverse as automated TV programme subtitling; assistive technology for disabled and elderly health service users; automated business meeting transcription and retrieval, and homeland security.

The use of open source software is practically a prerequisite for exploratory research of this kind, as it is never known in advance which parts of existing systems will need to be opened up and edited in the course of research. The speech community generally works on offline statistical, large data-set based research. For example corpora of 1000 hours of audio are not uncommon and require the use of large compute clusters to process them. These clusters already run Linux and HTK, so it is natural to extend the use of Linux into the audio capture phase of research. As speech research progresses from clean to natural speech, and from offline to real-time processing, it is becoming more integrated with general sound processing [Wolfel and McDonough, 2009], for example developing tools to detect and classify sounds as precursors to recognition. The use of Bayesian techniques in particular emphasises the advantages of considering the sound process-

[1] Currently owned by Microsoft, source available *gratis* but not *libre*. Kaldi is a *libre* alternative currently under development, (kaldi.sourceforge.net).

ing and recognition as tightly coupled problems, and using tightly integrated computer systems. For example, it may be useful for Linux cluster machines running HTK in real-time to use high level language models to generate Bayesian prior beliefs for low-level sound processing occurring in Linux audio.

This paper provides a studio report of our initial experiences setting up a Linux based studio for NST research. Our studio is based on a typical meeting room, where participants give presentations and hold discussions. We hope that it will serve as a self-contained tutorial recipe for other speech researchers who are new to the Linux audio community (and have thus included detailed explanations of relatively simple Linux audio concepts). It also serves as an example of the audio requirements of the natural speech research community; and as a case study of a successful Linux audio deployment.

2 Research applications

The NST project aims to use a meeting room studio, networked home installations, and our analysis cluster to improve recognition rates in natural environments, with multiple, mobile speakers and noise sources. We give here some examples of algorithms relevant to natural speech, and their requirements for Linux audio.

Beamforming and ICA are microphone-array based techniques for separating sources of audio signals, such as extracting individual speakers from mixtures of multiple speakers and noise sources. ICA [Roberts and Everson, 2001] typically makes weak assumptions about the data, such as assuming that the sources are non Gaussian in order to find a mixing matrix M which minimises the Gaussian-ness over time t of the latent sources vector x_t, from the microphone array time series vectors y_t, in $y_t = Mx_t$.

ICA can be performed with as few microphones as there are sound sources, but gives improved results as the number of microphones increases. Beamforming [Trees, 2002] seeks a similar output, but can include stronger physical assumptions - for example known microphone and source locations. It then uses expected sound wave propagation and interference patterns to infer the source waves from the array data. Beamforming is a high-precision activity, requiring sample-synchronous accuracy between recorded channels, and often using up to 64 channels of simultaneous audio in microphone arrays (see for example the NIST Mark-III arrays [Brayda et al., 2005]).

Reverberation removal has been performed in various ways, using single and multi-channel data. In multi-channel settings, sample-synchronous audio is again used to find temporal correlations which can be used to separate the original sound from the echos. In the iPhone4 this is performed with two microphones but performance may increase with larger arrays [Watts, 2009].

Speaker tracking may use SLAM techniques from robotics, coupled with acoustic observation models, to infer positions of moving speakers in a room (eg. [Fox et al., 2012], [Christensen and Barker, 2010]). This can be used in conjunction with beamforming to attempt retrieval of individual speaker channels from natural meeting environments, and again relies on large microphone arrays and sample-accurate recording.

Part of the NST project called 'homeService' aims to provide a natural language interface to electrical and electronic devices, and digital services in people's homes. Users will mainly be disabled people with conditions affecting their ability to use more conventional means of access such as keyboard, computer mouse, remote control and power switches. The assistive technology (AT) domain presents many challenges; of particular consequence for NST research is the fact that users in need of AT typically have physical disabilities associated with motor control and such conditions (e.g. cerebral palsy) will also often affect the musculature surrounding the articulatory system resulting in slurred and less clear speech; known as *dysarthric* speech.

3 Meeting room studio setup

Our meeting room studio, shown in fig. 1, is used to collect natural speech and training data from real meetings. Is centred on a six-person table, with additional chairs around the walls for around a further 10 people. It has a whiteboard at the head of the table, and a presentation projector. Typical meetings involve participants speaking from their chairs but also getting up and walking around to present or to use the whiteboard. A 2×2.5m aluminium frame is suspended from the ceiling above the table and used for mounting au-

Figure 1: Meeting room recording setup. The boxes on the far wall are active-badge trackers. The frame on the ceiling and the black cylinder on the table each contain eight condenser microphones. Participants wear headsets and active badges.

dio equipment. Currently this consists of eight AKG C417/III vocal condenser microphones, arranged in an ellipse around the table perimeter. A further eight AKG C417/IIIs are embedded in a 100mm radius cylinder placed in the table centre to act similarly to eight-channel multidirectional tele-conferencing recorder. The table can also include three 7-channel DevAudio Microcones (www.dev-audio.com), which are commercial products performing a similar function. The Microcone is a 6-channel microphone array which comes with propriety drivers and an API. A further 7th audio channel contains a mix of the other 6 channels as well as voice activity detection and sound source localisation information annotation. Some noise reduction and speech enhancement capabilities are provided, although details of the exact processing are not made public.

There are four Sennheiser ew100 wireless headsets which may be mounted on selected participants to record their speech directly. The studio will soon also include a Radvision Scopia XT1000 videoconferencing system, comprised of a further source-tracking steered microphone and two 1.5m HD presentation screens. In the four upper corners of the meeting room are mounted Ubisense infrared active badge receivers, which may be used to track the 3D locations of 15 mobile badges worn by meeting participants. (The university also has a 24-channel surround sound diffusion system used in an MA Electroacoustic music course [Mooney, 2005], which may be useful for generating spatial audio test sets.)

Sixteen of the mics are currently routed through two MOTU 8Pre interfaces, which take eight XLR or line inputs each. Both currently run at 48kHz but can operate up to 96kHz. The first of these runs in A/D conversion mode and sends all 8 digitised channels via a single ADAT Lightpipe fibre optic cable to the second 8Pre. The second 8Pre receives this input, along with its own eight audio channels and outputs all 16 channels to the DAW by Firewire 400 (IEEE 1394, 400 bits/sec). (The two boxes must be configured to have (a) the same speed, (b) 1x ADAT protocol and (c) be in be in converter/interface mode respectively.) Further firewire devices will be added to the bus later to accommodate the rest of the microphones in the room.

4 Linux audio system review

Fig. 2 outlines the Linux audio system and highlights the parts of the many possible Linux audio stacks that are used for recording in the meeting room studio. The Linux audio architecture has grown quite complex in recent years, so is reviewed here in detail.

OSS (Open Sound System) was developed in the early 1990s, focused initially on Creative SoundBlaster cards then extending to others. It was a locking system allowing only one program at a time to access the sound card, and lacked support for features such as surround sound. It allowed low level access to the card, for example by `cataudio.wav > /dev/dsp0`. ALSA (Advanced Linux Sound Architecture) was designed to replace OSS, and is used on most current distributions including our Ubuntu Studio 11.10. PortAudio is an API with backends that abstract both OSS and ALSA, as well as sound systems of non-free platforms such as Win32 sound and Mac CoreAudio, created to allow portable audio programs to be written. Several software mixer systems were built to resolve the locking problem for

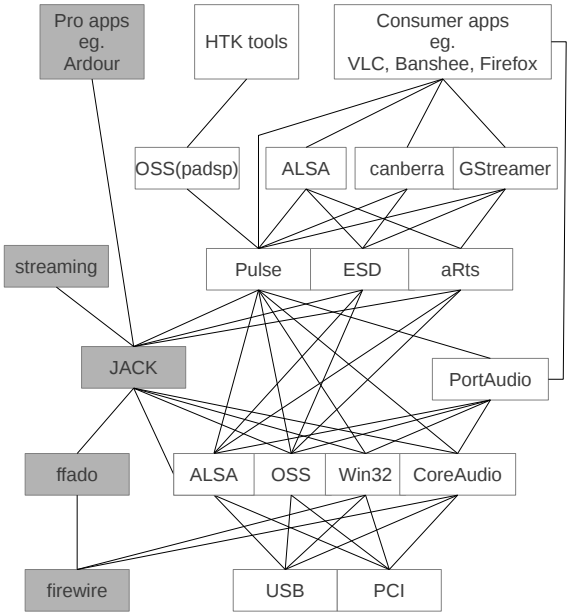

Figure 2: Audio system for recording.

consumer-audio applications, including PulseAudio, ESD and aRts. Some of these mixers grew to take advantage of and to control hardware mixing provided by sounds cards, and provided additional features such as network streaming. They provided their own APIs as well as emulation layers for older (or mixer-agnostic) OSS and ALSA applications. (To complicate matters further, recent versions of OSS4 and ALSA have now begun to provide their own software mixers, as well as emulation layers for each other.) Many current Linux distributions including Ubuntu 11.10 deploy PulseAudio running on ALSA, and also include an ALSA emulation layer on Pulse to allow multiple ALSA and Pulse applications to run together through the mixer. Media libraries such as GStreamer (which powers consumer-audio applications such as VLC, Skype and Flash) and libcanberra (the GNOME desktop sound system) have been developed closely with PulseAudio, increasing its popularity. However, Pulse is not designed for pro-audio work as such work requires very low latencies and minimal drop-outs.

The JACK system is an alternative software mixer for pro-audio work. Like the other soft mixers, JACK runs on many lower level platforms – usually ALSA on modern Linux machines. The bulk of pro-audio applications such as Ardour, zynAddSubFx and qSynth run on JACK. JACK also provides network streaming, and emulations/interfaces for other audio APIs including ALSA, OSS and PulseAudio. (Pulse-on-JACK is useful when using pro and consumer applications at the same time, such as when watching a YouTube tutorial about how to use a pro application. This re-configuration happens automatically when JACK is launched on a modern Pulse machine such as Ubuntu 11.10.)

5 Software setup

Our DAW is a relatively low-power Intel E8400 (Wolfdale) duo-core, 3GHz, 4Gb Ubuntu Studio 11.10-64-bit machine. Ubuntu studio was installed directly from CD – not added as packages to an existing Ubuntu installation – this gives a more minimalist installation than the latter approach. In particular the window manager defaults to the low-power XFCE, and resource-intensive programs such as Gnome-Network-Monitor (which periodically searches for new wifi networks in the background) are not installed. Ubuntu was chosen for compatibility and familiarity with our other desktop machines. (Several other audio distributions are available including PlanetCCRMA(Fedora), 64Studio(Debian), ArchProAudio(Arch)).

The standard ALSA and OSS provide interfaces to USB and PCI devices below, and to JACK above. However for firewire devices such as our Pre8, the `ffado` driver provides a *direct* interface to JACK from the hardware, bypassing ALSA or OSS. (Though the latest/development version provides an ALSA output layer as well.) Our DAW uses `ffado` with JACK2 (Ubuntu packages: `jack2d, jack2d-firewire, libffado, jackd, laditools`. JACK1 is the older but perhaps more stable single-processor implementation of the JACK API) and fig. 3 shows our JACK settings, in the `qjackctl` tool. The firewire backend driver (`ffado`) is selected rather than ALSA.

We found it useful to unlock memory for good JACK performance.[2] As well as ticking the un-

[2]By default, JACK locks all memory of its clients into RAM, ie. tells the kernel not to swap their pages to virtual memory on disc, see mlock(2). Unlock memory relaxes this slightly, allowing just the large GUI components of clients

lock memory option, the user must also be allowed to use it, eg. `adduser charles audio`. Also the file `/etc/security/limits.d/audio.conf` was edited (followed by a reboot) to include

```
@audio - rtprio 95
@audio - memlock unlimited
```

These settings can be checked by `ulimit -r -l`.

The JACK sample rate was set to 48kHz, matching the Pre8s. (This is a good sample rate for speech research work as it is similar to CD quality but allows simple sub-sampling to power-of-two frequencies used in analysis.)

Fig. 4 shows the JACK connections (again in `qjackctl`) for our meeting room studio setup. The eight channels from the converter-mode Pre8 appear as ADAT optical inputs, and the eight channels from the interface-mode Pre8 appear as 'Analog' inputs, all within the firewire device. Ardour was used with two tracks of eight channel audio to record as shown in fig. 5.

5.1 Results

Using this setup we were able to record simultaneously from six overhead microphones, eight table-centre microphones, and two wireless headsets, as illustrated in fig. 5. We experienced no JACK xruns in a ten minute, 48kHz, 32-bit, 16-channel recording, and the reported JACK latency was 8ms. A one hour meeting recording with the same settings experienced only 11 xruns. Total CPU usage was below 25% at all times, with `top` listing the following typical total process CPU usages: `jack` 11%, `ardour` 8%, `jack.real` 3%, `pulseaudio` 3%.

However, we were unable to play audio back through the Pre8s, hearing distorted versions of the recording. For our speech recognition this is relatively unimportant, and can be worked around by streaming the output over the network with JACK and playing back on a second machine. The `ffado` driver's support for the Pre8 hardware is currently listed as 'experimental' so work is needed here to fix this problem.

The present two Pre8 system is limited to 16 audio channels, we plan to extend it with further firewire devices to record from more audio sources around the meeting room and from tele-

to perform swapping, leaving more RAM free for the audio parts of clients.

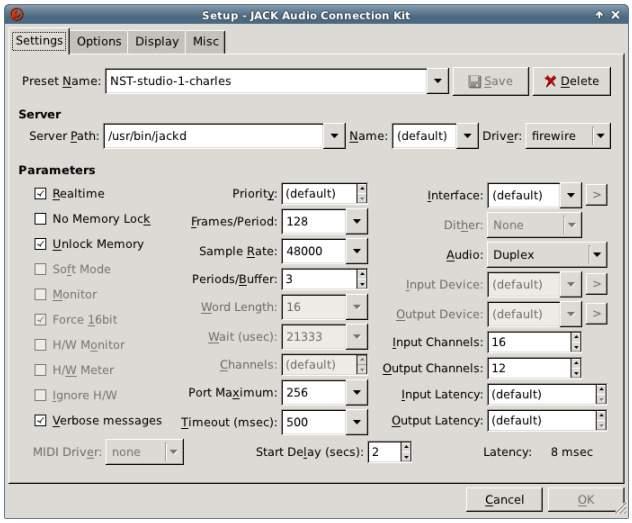

Figure 3: 16 channel recording JACK settings.

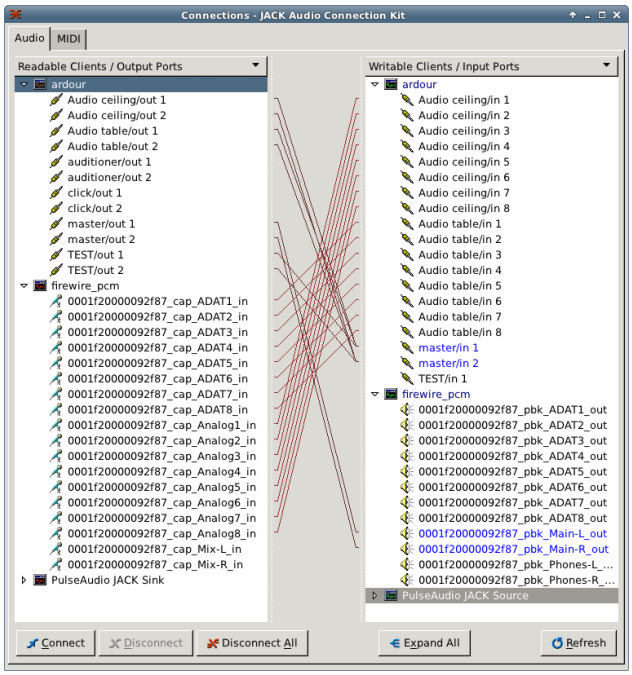

Figure 4: 16 channel recording JACK connections.

conferencing channels in future. We have not yet needed to make further speed optimisations, but we note that for future, more-channel systems, two speedups include disabling PulseAudio (adding `pulseaudio -kill` and `pulseaudio -start` to qjackctl's startup and shutdown option is a simple way to do this); and installing the real-time `rt-linux` kernel.

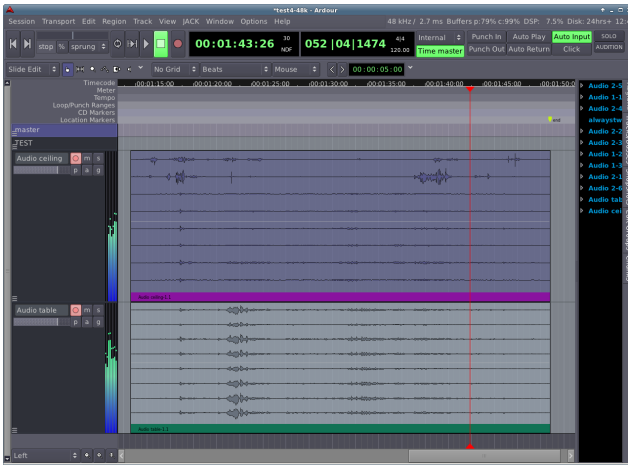

Figure 5: 16 channel meeting room recording in Ardour, using two 8-channel tracks.

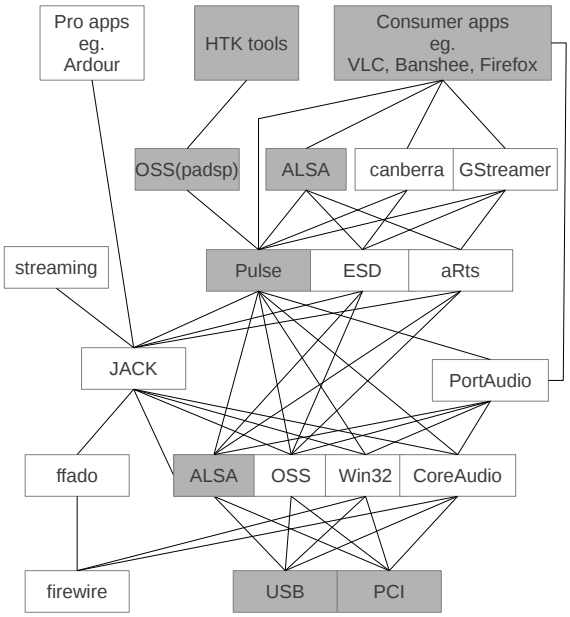

Figure 6: Audio system for data analysis.

5.1.1 Audio analysis

Analysis of our audio data is performed on a compute cluster of 20 Linux nodes with 96 processor cores in total, running the Oracle (formerly Sun) Grid Engine, the HTK Hidden Markov Model Tool Kit [Young et al., 2006] and the Juicer recognition engine [Moore et al., 2006]. During analysis, audio playback on desktop machines is useful and is done with the setup of fig. 6. For direct audio connections the HTK tools make use the OSS sound system, which may be emulated on PulseAudio on a modern machine, by installing the `padsp` tool (Ubuntu package `pulseaudio-utils_0.9.10-1ubuntu1_i386`) then prefixing all audio HTK commands with `padsp`. Similarly, Juicer allows the use of an OOS front, the implementation of a JACK plugin is in progress.

The cluster may be used both for online and offline processing. An example of online processing can be found in the description of an online transcription system for meetings [Garner et al., 2009]. Such systems distinguish between far and near-field audio sources and employ beamforming and echo cancellation procedures [Hain et al., 2009] for which either sample synchronicity or at least minimal latency between channels is of utmost importance. For both offline and online processing typically audio at 16kHz/16bit is required, to be converted into so-called feature streams (for example Perceptual Linear Predictive (PLP) features [Hermansky, 1990]) of much lower bit rate. Even higher sampling frequencies (up to 96kHz are commonplace) are often required for cross-correlation based sound source localisation algorithms to provide sufficient time resolution in order to detect changes in angles down to one degree. In offline mode the recognition of audio typically operates at 10 times real-time (i.e. 1 hour of audio in many channels takes 10 hours to process). However, the grid framework allows the latency of processing to drop to close to 1.5 times real-time using massive parallelisation.

6 Future directions

The ultimate goal of NST is to obtain transcriptions of what was said in natural environments. Traditionally, the source separation and denoising techniques of sec. 2 have been treated as a separate preprocessing step, before the cleaned audio is passed to a separate transcriber such as HTK. However for challenging environments this is suboptimal, as it reports only a single estimate of the denoised signal rather than Bayesian information about its uncertainty. Future integrated systems could fuse the predictions from transcription language models with inference and reporting of low-level audio data, for example by passing real-time probabilistic messages between HTK's transcrip-

tion inference (on a Linux computing cluster) and low-level audio processing (on desktop or embedded Linux close to the recording hardware.)

For training and testing speaker location tracking systems it is useful to build a database of known speaker position sequences, which need to be synchronised to the audio. Positions change at the order of seconds so it is wasteful to use audio channels to record them – however we note that JACK is able to record MIDI information alongside audio, and one possibility would be to encode position from our Ubisense active badges as MIDI messages, synchronous with the audio, and record them together for example in Ardour3. It could also be useful to – somehow – synchronise video of meetings to JACK audio.

The homeService system has two major parts, namely hardware components that are deployed in people's homes during trial (Android tablet, Microcone, and Linux box responsible for audio capture, network access, infrared and blue tooth communication), as well as our Linux computing cluster back at the university running the processes with particularly high demands in terms of processing power and memory usage. The main audio capturing will take place on a Linux box in the users' home, and we plan to develop a setup which will enable the Microcone and JACK to work together and provide – if needed – live streaming of audio over the network.

The main requirements of NST research for Linux audio are support for sample-synchronous, many (e.g. 128 or more) channel recording, and communication with cluster computing and speech tools such as HTK. As natural speech technology makes closer contact with signal processing research, we expect to see more speech researchers moving to Linux audio in the near future, and we hope that this paper has provided some guidance for those who wish to make this move, as well as a guide for the Linux audio community about what technologies are important to this field.

Acknowledgements

The research leading to these results was supported by EPSRC Programme Grant EP/I031022/1 (Natural Speech Technology).

References

L. Brayda, C. Bertotti, L Cristoforetti, M. , Omologo, and P. Svaizer. 2005. Modifications on NIST MarkIII array to improve coherence properties among input signals. In *Proc. of 118th Audio Engineering Society Conv.*

H. Christensen and J. Barker. 2010. Speaker turn tracking with mobile microphones: combining location and pitch information. In *Proc. of EUSIPCO.*

C. Fox, M. Evans, M. Pearson, and T. Prescott. 2012. Tactile SLAM with a biomimetic whiskered robot. In *Proc. ICRA.*

P. N. Garner, J. Dines, T. Hain, A. el Hannani, M. Karafiat, D. Korchagin, Mike L., V. Wan, and L. Zhang. 2009. Real-Time ASR from Meetings. In *Interspeech'09*, pages 2119–2122.

T. Hain, L. Burget, J. Dines, P. N. Garner, A. el Hannani, M. Huijbregts, M. Karafiat, M. Lincoln, and V. Wan. 2009. The AMIDA 2009 meeting transcription system. In *Proc. Interspeech.*

H. Hermansky. 1990. Perceptual linear predictive analysis for speech. *J. Acoustic Society of America*, pages 1738–1752.

J. Mooney. 2005. *Sound Diffusion Systems for the Live Performance of Electroacoustic Music.* Ph.D. thesis, University of Sheffield.

D. Moore, J. Dines, M. Magimai Doss, J. Vepa, O. Cheng, and T. Hain. 2006. Juicer: A weighted finite state transducer speech decoder. In *Machine Learning for Multimodal Interaction*, pages 285–296. Springer-Verlag.

S. Roberts and R. Everson, editors. 2001. *Independent Components Analysis: Principles and Practice.* Cambridge.

H. L. Van Trees. 2002. *Optimum array processing.* Wiley.

L. Watts. 2009. Reverberation removal. In *United States Patent Number 7,508,948.*

M. Wolfel and J. McDonough. 2009. *Distant Speech Recognition.* Wiley.

S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, XA Liu, G. Moore, J. Odell, D. Ollason, D. Povey, et al. 2006. The HTK book.