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

Charles FOX, Heidi CHRISTENSEN and **Thomas HAIN**
Center for 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 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]. 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 in Sheffield, UK, for dedicated natural speech technology 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.

This paper provides a studio report of our initial experiences setting up a Linux based studio for natural speech technology 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 – such reports are still quite rare and much needed to demonstrate the power of modern Linux audio.

2 Why Linux audio?

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

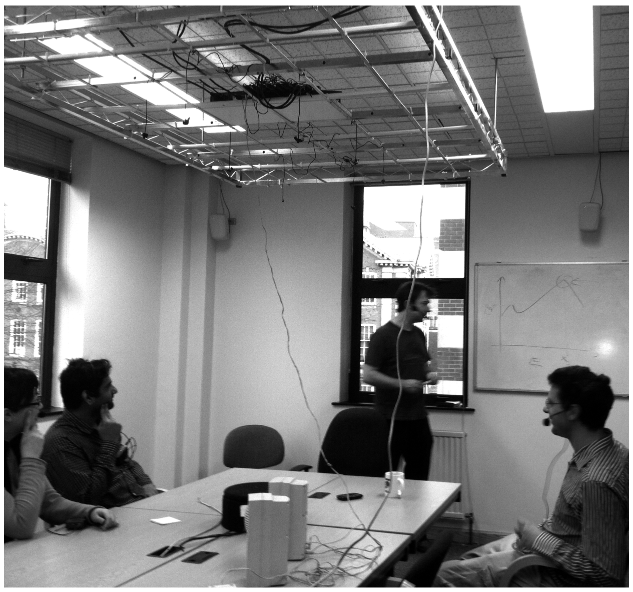


Figure 1: Meeting room recording setup. The boxes on the far wall are active-badge trackers. The frame on the ceiling and the black cylinder on the table each contain eight condenser microphones. Participants wear headsets and active badges.

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 processing 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.

3 Meeting room studio setup

Our meeting room studio, shown in fig. 1, 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 audio 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 multi-directional teleconferencing 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 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 Light-

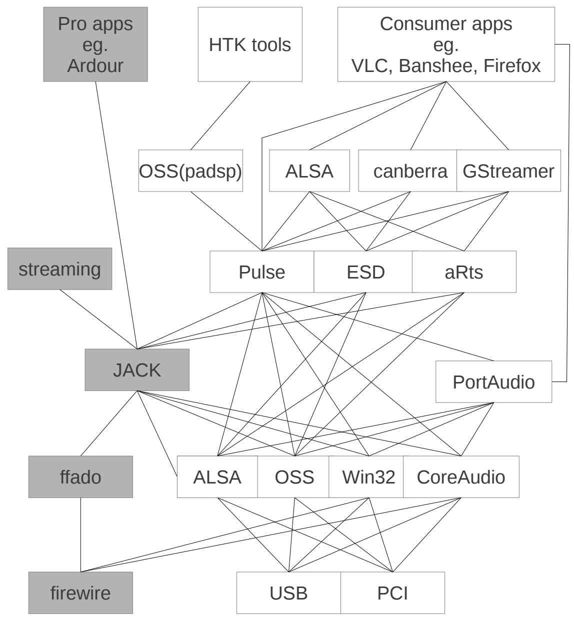


Figure 2: Audio system for recording.

pipe 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 accomodate the rest of the microphones in the room.

4 Linux audio system review

[*Note to reviewers: we are including this as we had a hard time finding a definitive account of how all these parts fit together. We hope putting it together in a peer-reviewed paper will be useful to others. Please could you help us out by checking this, and the diagrams, in detail so that we can get everything perfect for people to use in future?*]

Fig. 2 outlines the current Linux audio system and highlights the parts of the possible Linux audio stack that are used for recording in the meeting room studio. The Linux audio architecture has grown quite complex in recent years, and is reviewed here and illustrated in fig. 2 to help readers who may be using this paper as a standalone tutorial.

Historically, the OSS system was developed for Linux in the early 1990s, focused initially on the Creative SoundBlaster cards then extending to others. It was a locking system which allowed only one program at a time to access the sound card, and lacked support for modern features such as surround sound. It allowed low level access to the card, for example by `cataudio.wav` > `/dev/dsp0`. The ALSA system was designed as a modern replacement for 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 consumer-grade applications, including PulseAudio, ESD and aRts. Some of these mixers grew to take advantage of and 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 he mixer. Common media libraries such as GStreamer (which powers consumer 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 which relies on very low latencies and minimal drop-outs.

The JACK system is an alternative software mixer which fills this need. 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 CPU-hogging programs such as Gnome-Network-Monitor (which periodically searches for new wifi networks in the background) are not installed.

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. Note that the firewire backend driver (ffado) is selected rather than ALSA.

It is important to unlock memory for good JACK performance. As well as ticking the unlock 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

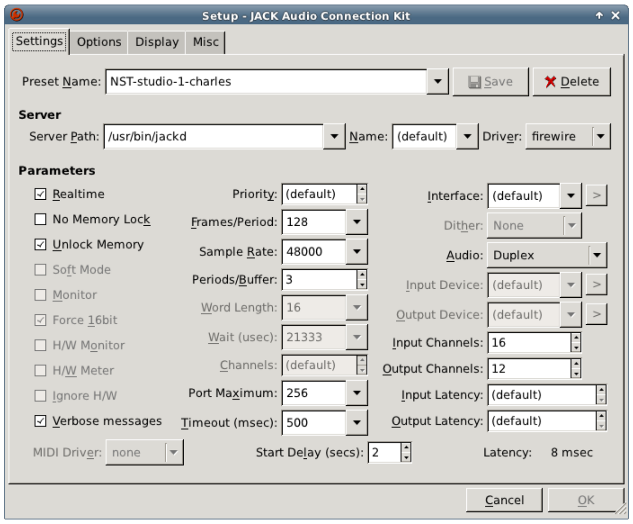
Using this setup we were able to record simultaneously from six overhead microphones, eight table-centre microphones, and two wireless headsets, as illustrated in fig. 5. We experienced no JACK xruns in a ten minute, 48kHz, 16-channel recording, and the reported JACK latency was 8ms. Total CPU usage 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 have been unable to play audio back through the Pre8 system, hearing highly 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 JACK/ALSA Linux 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 teleconferencing 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.

5.1.1 Analysis system

Analysis of our audio data is performed on a cluster of 20 Linux machines having about 80 cores (average $\tilde{2}.9$GHz) in total, running the Oracle (formerly Sun) Grid Engine and the HTK Hidden Markov Model Tool Kit. During analysis, audio playback on desktop machines is useful and is done with the setup of fig. 6. An important note for speech researchers is that the HTK tools use the OSS sound system, which may



Setup - JACK Audio Connection Kit

Settings | Options | Display | Misc

Preset Name: NST-studio-1-charles — Save — Delete

Server

Server Path: /usr/bin/jackd — Name: (default) — Driver: firewire

Parameters

- ☑ Realtime
- ☐ No Memory Lock
- ☑ Unlock Memory
- ☐ Soft Mode
- ☐ Monitor
- ☑ Force 16bit
- ☐ H/W Monitor
- ☐ H/W Meter
- ☐ Ignore H/W
- ☑ Verbose messages

Priority:	(default)	Interface:	(default)
Frames/Period:	128	Dither:	None
Sample Rate:	48000	Audio:	Duplex
Periods/Buffer:	3	Input Device:	(default)
Word Length:	16	Output Device:	(default)
Wait (usec):	21333	Input Channels:	16
Channels:	(default)	Output Channels:	12
Port Maximum:	256	Input Latency:	(default)
Timeout (msec):	500	Output Latency:	(default)

MIDI Driver: none — Start Delay (secs): 2 — Latency: 8 msec

Cancel — OK

Figure 3: 16 channel recording JACK settings.

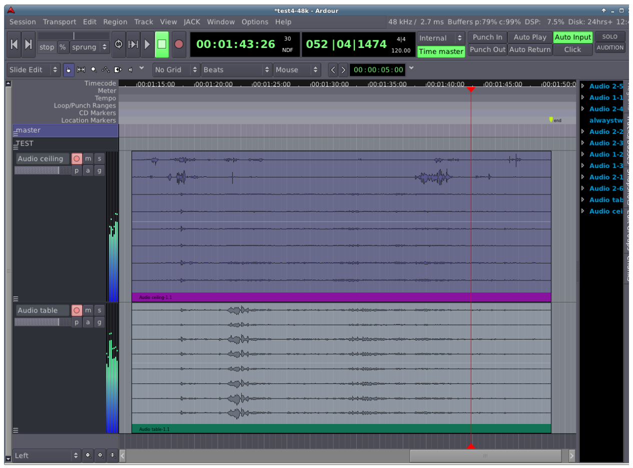
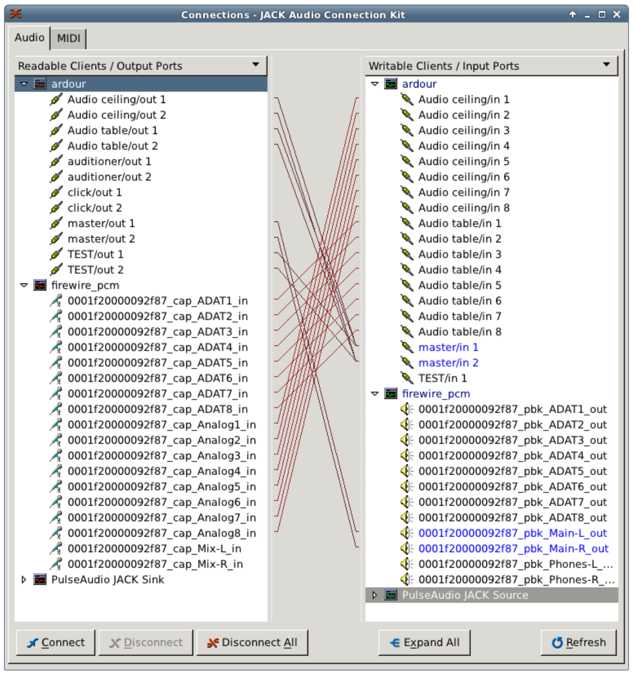


Figure 5: 16 channel meeting room recording in Ardour, using two 8-channel tracks..



Connections - JACK Audio Connection Kit

Audio | MIDI

Readable Clients / Output Ports

- ardour
  - Audio ceiling/out 1
  - Audio ceiling/out 2
  - Audio table/out 1
  - Audio table/out 2
  - auditioner/out 1
  - auditioner/out 2
  - click/out 1
  - click/out 2
  - master/out 1
  - master/out 2
  - TEST/out 1
  - TEST/out 2
- firewire_pcm
  - 0001f20000092f87_cap_ADAT1_in
  - 0001f20000092f87_cap_ADAT2_in
  - 0001f20000092f87_cap_ADAT3_in
  - 0001f20000092f87_cap_ADAT4_in
  - 0001f20000092f87_cap_ADAT5_in
  - 0001f20000092f87_cap_ADAT6_in
  - 0001f20000092f87_cap_ADAT7_in
  - 0001f20000092f87_cap_ADAT8_in
  - 0001f20000092f87_cap_Analog1_in
  - 0001f20000092f87_cap_Analog2_in
  - 0001f20000092f87_cap_Analog3_in
  - 0001f20000092f87_cap_Analog4_in
  - 0001f20000092f87_cap_Analog5_in
  - 0001f20000092f87_cap_Analog6_in
  - 0001f20000092f87_cap_Analog7_in
  - 0001f20000092f87_cap_Analog8_in
  - 0001f20000092f87_cap_Mix-L_in
  - 0001f20000092f87_cap_Mix-R_in
- PulseAudio JACK Sink

Writable Clients / Input Ports

- ardour
  - Audio ceiling/in 1
  - Audio ceiling/in 2
  - Audio ceiling/in 3
  - Audio ceiling/in 4
  - Audio ceiling/in 5
  - Audio ceiling/in 6
  - Audio ceiling/in 7
  - Audio ceiling/in 8
  - Audio table/in 1
  - Audio table/in 2
  - Audio table/in 3
  - Audio table/in 4
  - Audio table/in 5
  - Audio table/in 6
  - Audio table/in 7
  - Audio table/in 8
  - master/in 1
  - master/in 2
  - TEST/in 1
- firewire_pcm
  - 0001f20000092f87_pbk_ADAT1_out
  - 0001f20000092f87_pbk_ADAT2_out
  - 0001f20000092f87_pbk_ADAT3_out
  - 0001f20000092f87_pbk_ADAT4_out
  - 0001f20000092f87_pbk_ADAT5_out
  - 0001f20000092f87_pbk_ADAT6_out
  - 0001f20000092f87_pbk_ADAT7_out
  - 0001f20000092f87_pbk_ADAT8_out
  - 0001f20000092f87_pbk_Main-L_out
  - 0001f20000092f87_pbk_Main-R_out
  - 0001f20000092f87_pbk_Phones-L_...
  - 0001f20000092f87_pbk_Phones-R_...
- PulseAudio JACK Source

Connect — Disconnect — Disconnect All — Expand All — Refresh

Figure 4: 16 channel recording JACK connections.

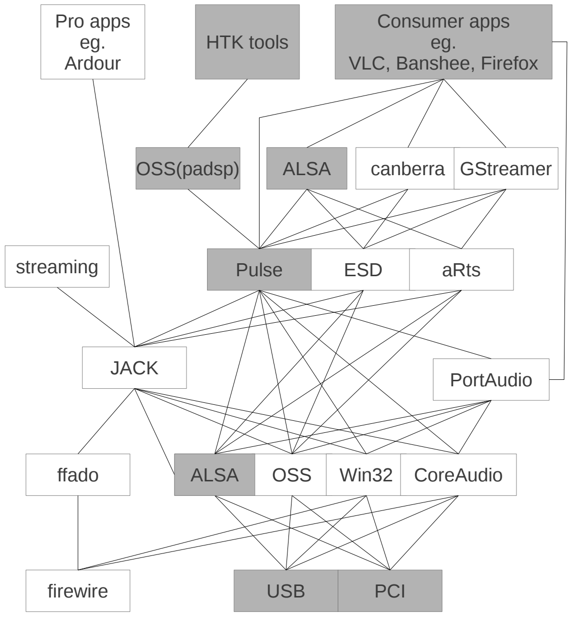


Figure 6: Audio system for data analysis.

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`.

We have used this cluster to align text transcripts to 1300 hours of two-subject, monophonic audio meetings (16kHz, 16bit audio; 450 speakers; 580,598 speaker lines; 11.7M word tokens; 51K unique words; 2.5 words/sec rate). HTK on the cluster was used to extract Perceptual Linear Predictive speech coding features [Hermansky, 1990]. Alignment was performed using the HTK tool HVite and typically takes about 2.5 hours to run on the cluster for data of this size. Using an HTK Hidden Markov Model trained on a previous meeting set, we have so far obtained alignment of

400 hours of this data.

6 Research applications

The NST project aims to further use the meeting room studio and analysis cluster to improve recognition rates in natural meeting 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 $y_t = M.x_t$.

ICA can be performed with as few microphones as there are sources, but gives improved results as the number of microphones increases. Beamforming [Trees, 2002] seeks a similar output, but makes stronger physical assumptions, including 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. [THOMAS: can you check/improve this please?]

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 localisation and tracking uses SLAM techniques from robotics, coupled with acoustic observation models, to infer the positions of moving speakers in a room. 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. For training and testing such systems it is useful to build a database with known speaker position sequences, which need to be synchronised to the audio. Speaker 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 possible setup would be to encode position data from our Ubisense active badges as MIDI messages, synchronous with the audio, and record them together for example in Ardour3.

The ultimate goal of natural speech research is to obtain a transcription of what was said in natural environments. Traditionally, source separation and denoising techniques such as the above have been treated as a separate preprocessing step, before the cleaned audio is passed to a separate recogniser 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 should seek to fuse the predictions from the transcription language model with inference and reporting of low-level audio data, for example by passing real-time probabilistic messages between HTK's transcription inference (which typically runs on a Linux computing cluster) and low-level audio processing (which typically runs on or close to the recording machines.)

In summary, the main requirements of natural speech 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

Funded by UK EPSRC Natural Speech Technology Programme Grant, EP/I031022/1.

References

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*, pages 358–361.

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.

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.