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

Dr Charles Fox
Research Associate, Natural Speech Technology
University of Sheffield, UK
www.5m.org.uk

Talk aims

- Tell you about the world of Automated Speech Recognition research – not much overlap yet with Linux audio world. But there may be soon!
- Natural Speech Technology is different from standard ASR (such as DragonDictate), more audio/DSP focussed and needing Linux audio.
- What we are using your tools to do (thanks!)
- What we need your tools to do (please!)
  - (vs computer music requirements in particular)

ASR vs Natural speech

- Automated Speech Recognition (ASR)
  - Historically developed as a separate community from audio research (music, DSP etc)
  - Mature field, 20 years of tools
  - Large-data (eg. 1000 hours) and results driven
  - Standard features (MFCC, PLP) preprocessing
  - Then statistical models of feature data
  - Many ASR researchers don't listen to the speech!
  - Then find models and parameters to optimise word error rate (WER)
  - 10% great, 20% good, 40% normal, depends on corpus

ASR vs Natural speech

- ASR just about a solved problem for single, trained user in noiseless environment
- eg. Dragon Dictate, Siri
  - (both by Nuance Inc)
  - WER 1% obtainable
- Clean but unknown speakers
  - eg. radio news broadcasts
  - Still a research area,
  - WERs eg between 17%-70%

ASR vs natural speech

- Recent research area : extending ASR to **natural** speech (Wolfel&McDonough,2006)
  - business meetings - minuting
  - TV – subtitling
  - Interview archives – transcription
  - Telephone conversations – transcription + keywords

UK NST project

- Involves many organisations interested in NST:

Running 2011-2015. Sample applications:

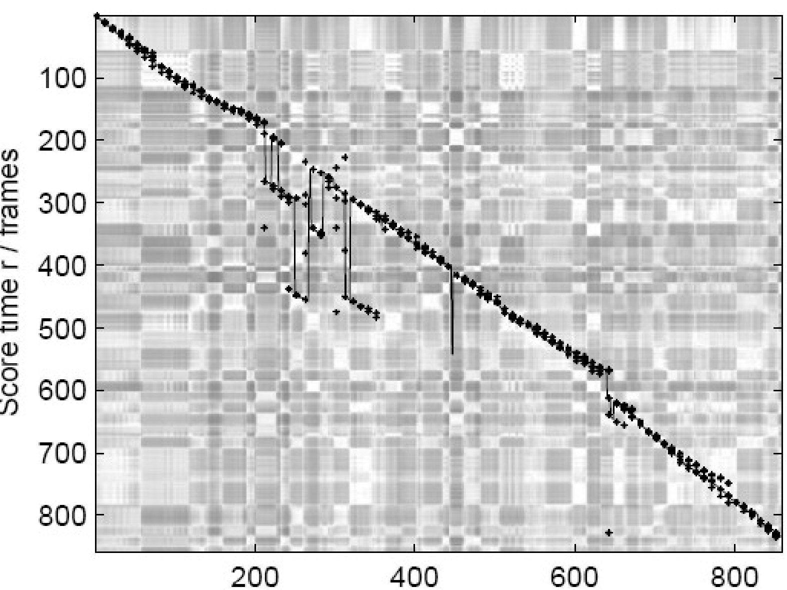
- Subtitling TV and radio programmes (BBC)
- Consumer and business meeting transcription (Nuance)
- Security (GCHQ)
- Voice control for disabled patients (NHS)

Natural speech challenges

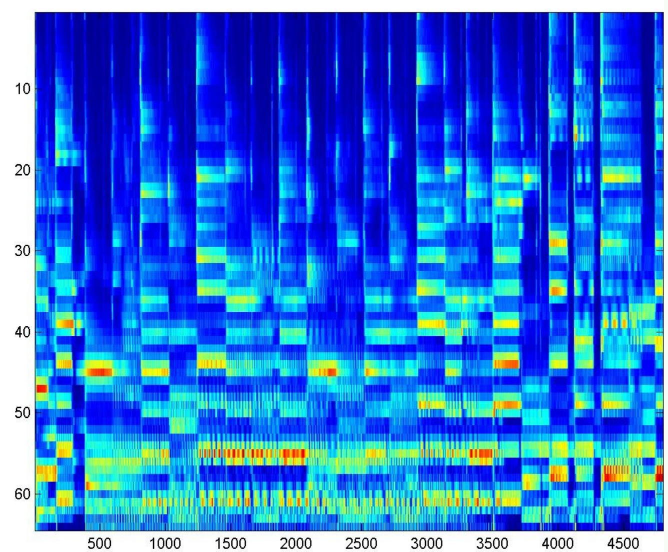
- Unlike standard ASR tasks, we have:
  - Multiple speakers talking at once
  - Speakers moving around the room
  - Significant room effects (reverb,resonance)
  - Background noise (furniture, traffic, plumbing)
  - Unknown speakers, accents/dialects
- Need to work more with DSP now
  - Begins to look more like computer music type research (eg. Music transcription, CASA)
  - Need nice audio tools
  - but different requirements from musical audio

My background – linux audio

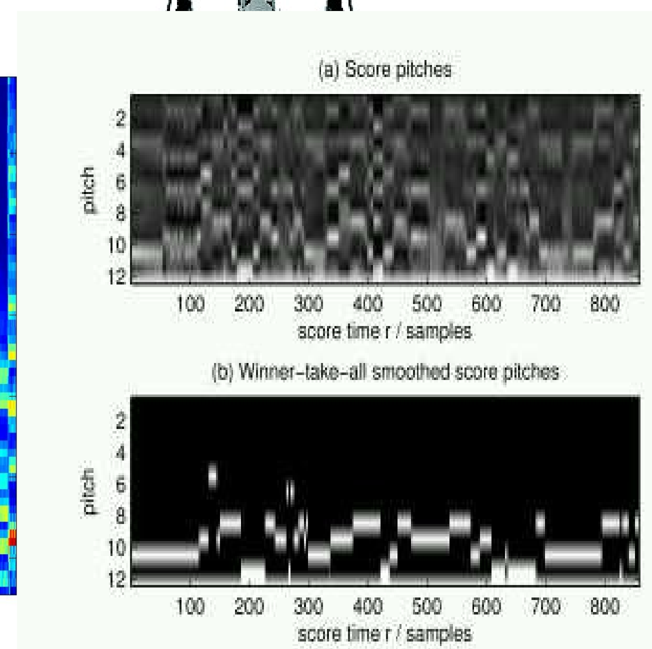
- Score following
  - Raphael Music+1
  - "How to be Lost", Fox&Quinn ICMC2008
- Linux,ALSA; Matlab+win.



Particle filter + lostness



Lyon cochlea filter

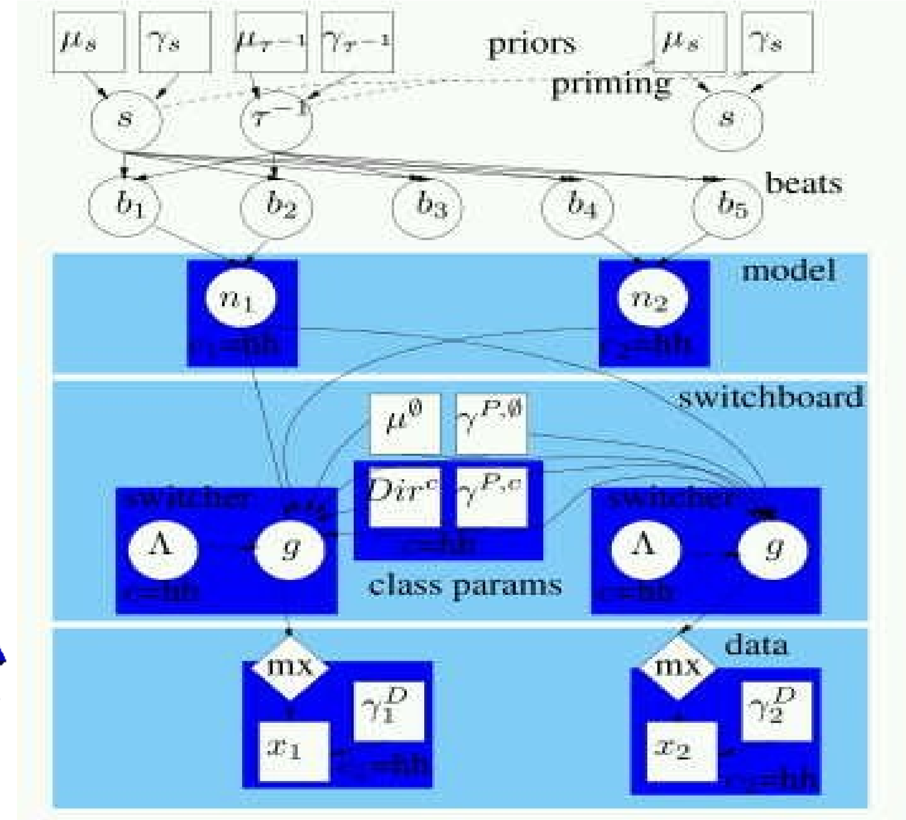


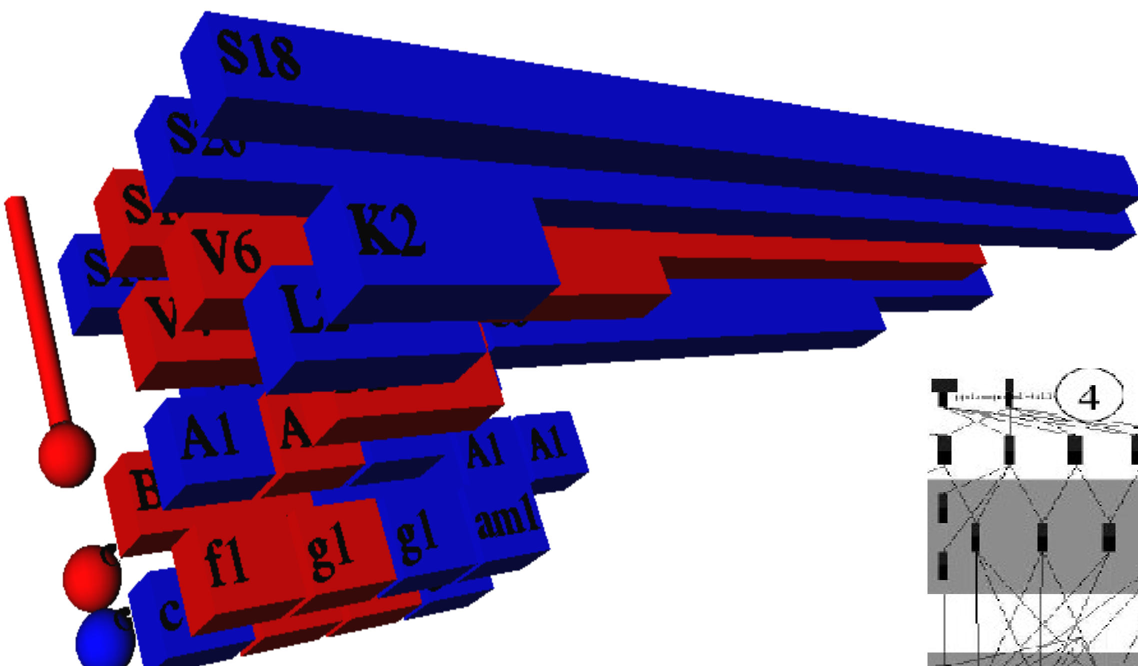
Chordbank filter HMM

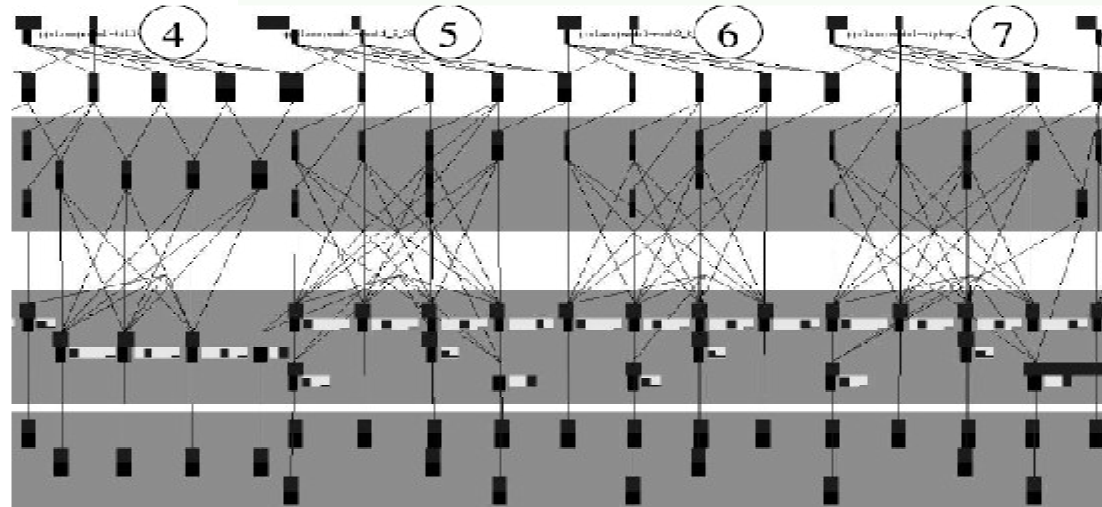
The University Of Sheffield.

Bayesian musical scenes

- Rhythms
- Structures (Hofstadter's CopyCat)
- Python, Lush Lisp, OSS midi

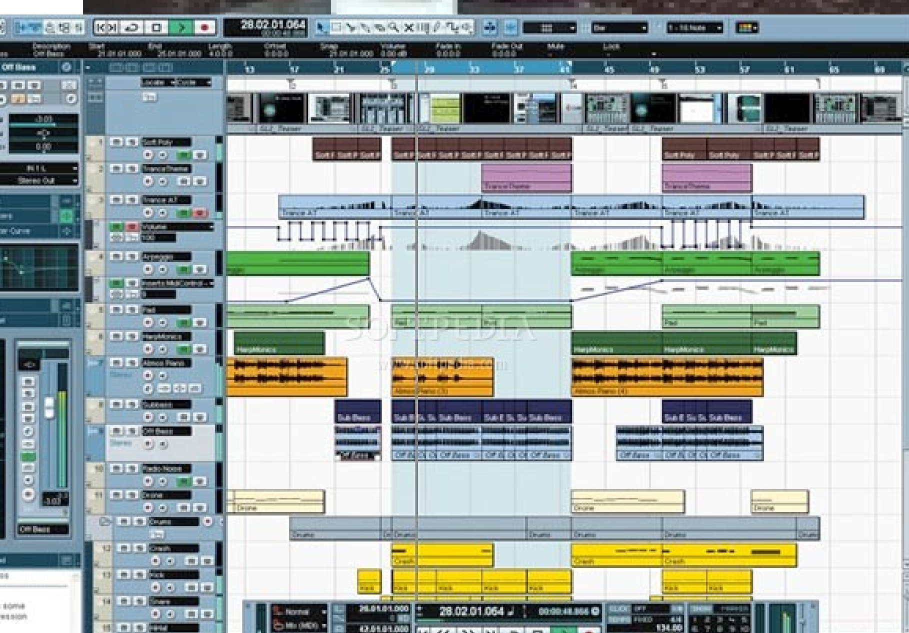




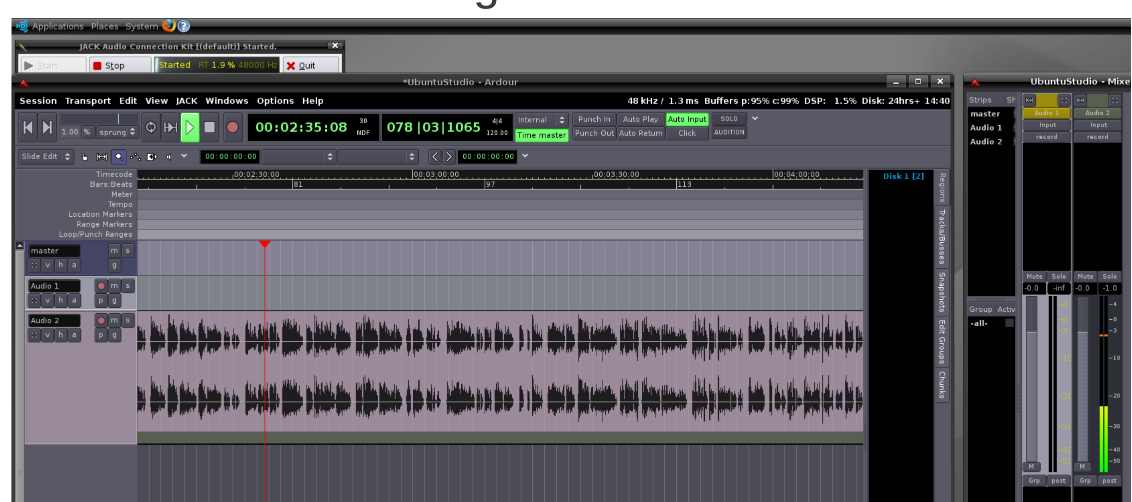


Fox,Rezek&Roberts,ICMC08
Fox&Roberts, AI Review, 2011
Murray-Brown & Fox, EvoStar09
Fox, FLAIRS2008 ; Fox, FLAIRS06

Music recording



Catch-it Kebabs (Riot Music, 2007)
£11.66 (Amazon/shops) or free direct download
www.catchitkebabs.co.uk
Studio recording still in Cubase/ProTools :-(
Some demos/writing now in Ardour/JACK



Speech vs music data

- Corpus size:
  - music album ~20 hours. music usually one track at a time, 5 mins.
  - Corpus 1000hours. Needs fast access to all at once; playback ~1 hour.
- Quality
  - Music mixdown 44.1kHz, 16bit;
  - classic ASR usually starts with 16kHz 16bit, then extracts features.
  - NST ASR may need music or better quality, eg. 48kHz 32bit.
- Channels
  - Music ~ 20 channels, typically record <10 at once
  - NST ASR, localisation and separation techniques may require large mic arrays, 16 channels common, 64+ would be nice...
- Compute power: Music usually on one PC; ASR on 100 core cluster.
- Real-time
  - Music album: latency is crucial. Classic ASR: processing done offline.
  - NST: maybe will need realtime interplay with the processing?

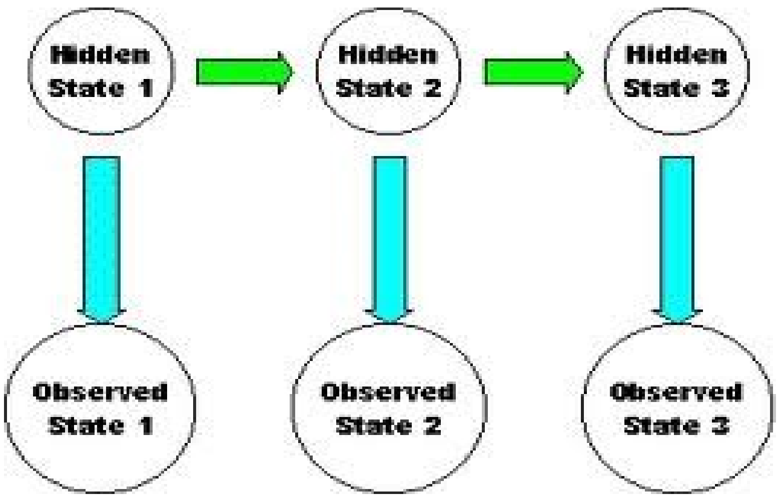
Why Linux? HTK setup

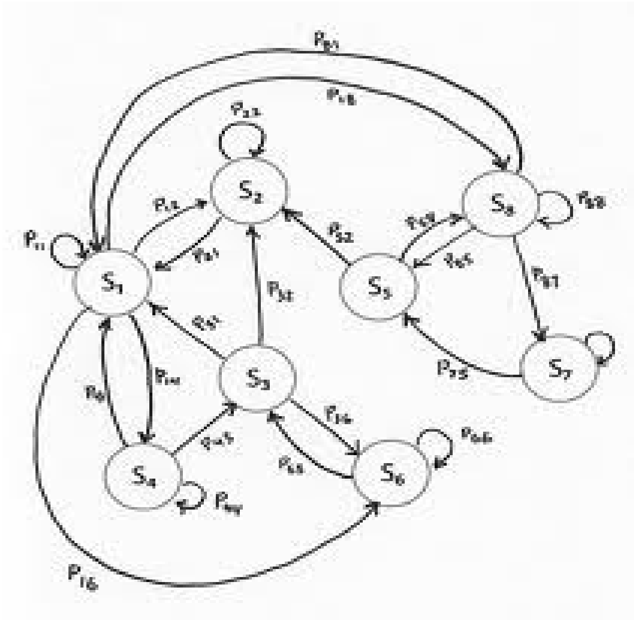
- Conventional modern ASR is done offline, on large clusters
- Our set up:
  - 100 Core cluster, running Oracle (Sun) Grid Engine
  - CentOS Linux (Community enterprise version of Fedora)
  - Encode 1000 hours of wav to features ~1 day
  - Train language models, 10M words, ~3 hours.
  - Alignment/decode ~1 day; train audio ~1 weekend.
- Crucial need to hack – NST is a new research area and we don't know in advance which parts of the tool chain will need to be opened up and modified!

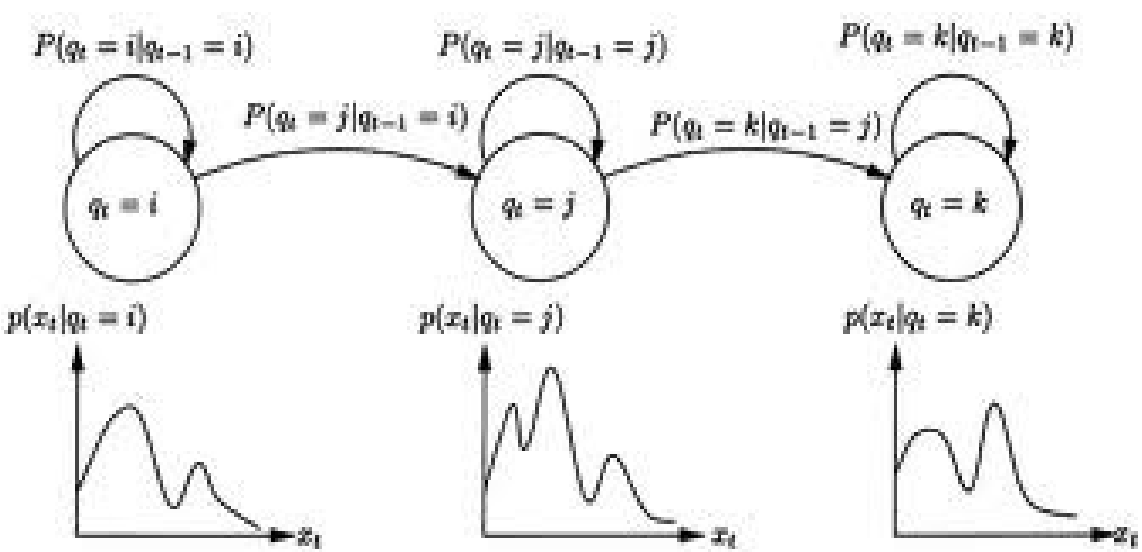
Why Linux? ASR Tools.

- Much ASR research uses the HTK (Hidden Markov model ToolKit) tools. Native to Unix/Linux (but win32 ports)
  - Under development for ~20 years, mature.
  - Tools for **building** speech recognisers, is **not** a full system
  - Began in Cambridge University; sold to spin-out Entropic; bought by Microsoft; licensed back to Cambridge.
  - Cambridge is then allowed to distribute HTK source on the net under a gratis but non-libre licence.
  - Licence allows users to modify code for own use but cannot re-distribute (eg. In a linux distro or “OpenDragonDictate”).
  - Users are “encouraged” to donate their modifications back to Microsoft for inclusion in future version
  - Microsoft could revoke the licence at any time
  - Many groups have their own mods that they don't donate back
  - A GPL alternative **Kaldi** is in progress; but HTK is dominant.

Why Linux? HTK.







Small HMMs can be written in a few lines of Python or Octave

ASR requires **very** large ones,

eg. 200,000 triphone states

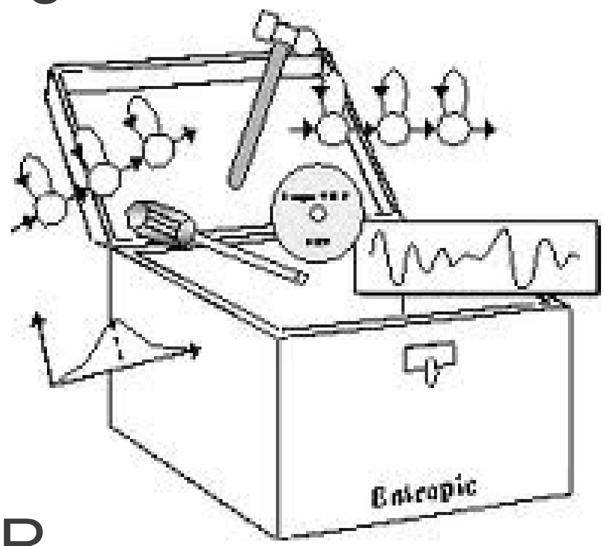
/c/ /a/ /t/ vs /b/ /a/ /t/

Transition priors from large language models, eg. 10M words.

Large models need lots of clever tricks and heuristics to run in reasonable time and memory (eg. On 100 $gb-node clusters). Hence HTK.

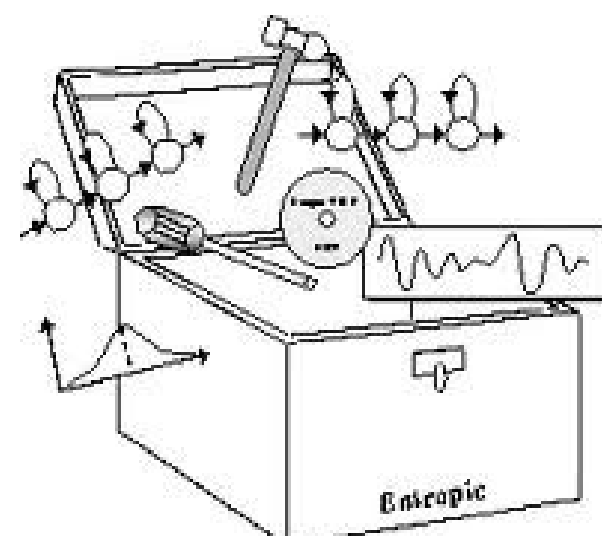
HTK command line tools

- Preprocessing
  - Listening (eg 1 hour wavs, show speech features)
  - Feature extraction (MFCC, PLP)
- Alignment
  - Line up known transcript with audio, using HMM
- Training
  - EM training of HMM
- Decoding
- Scoring
  - International NIST standard scores, WER
- Typically controlled from big tcsh or Python scripts



HTK-related ecosystem

- Sox – audio conversion and listening (GPL)
- Perl/Python.regex – transcript processing (GPL)
- HDecode – alternative HMM decoder (?)
- ARPA language modelling tools (govt/BSD?)
- NIST scoring tools (govt/BSD?)
- Padsp – makes HTK work on PulseAudio (GPL)
- OSS – native HTK audio system (GPL)
- Tcsh/Python scripting (GPL)
- Sed/awk – always useful (GPL)
- Audacity,OpenOfficeCalc for inspection.

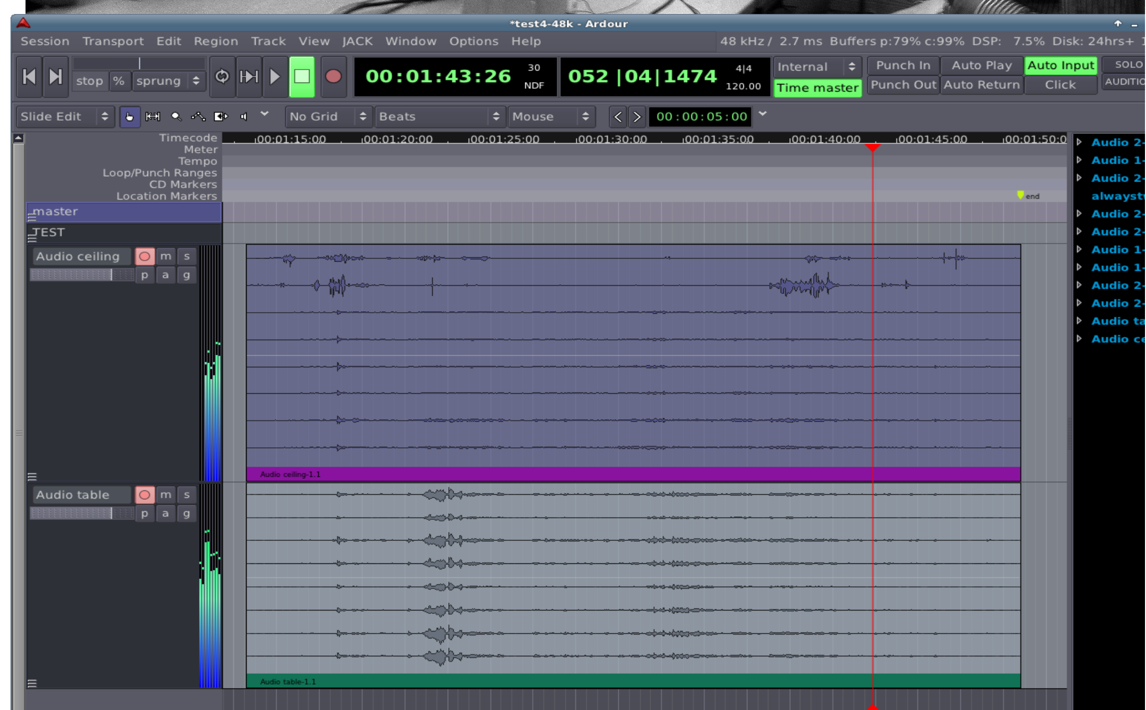


From ASR to NST

- Unlike standard ASR tasks, we have:
  - Multiple speakers talking at once
  - Speakers moving around the room
  - Significant room effects (reverb,resonance)
  - Background noise (furniture, traffic, plumbing)
  - Unknown speakers, accents/dialects

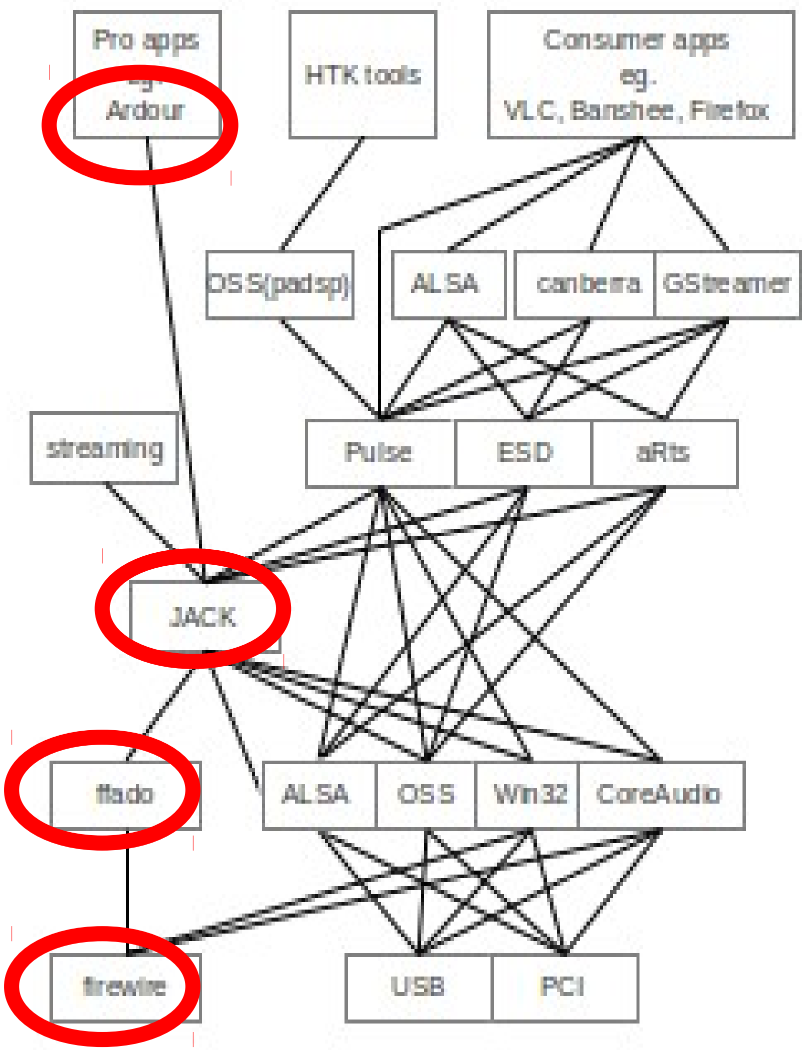
Lab set up

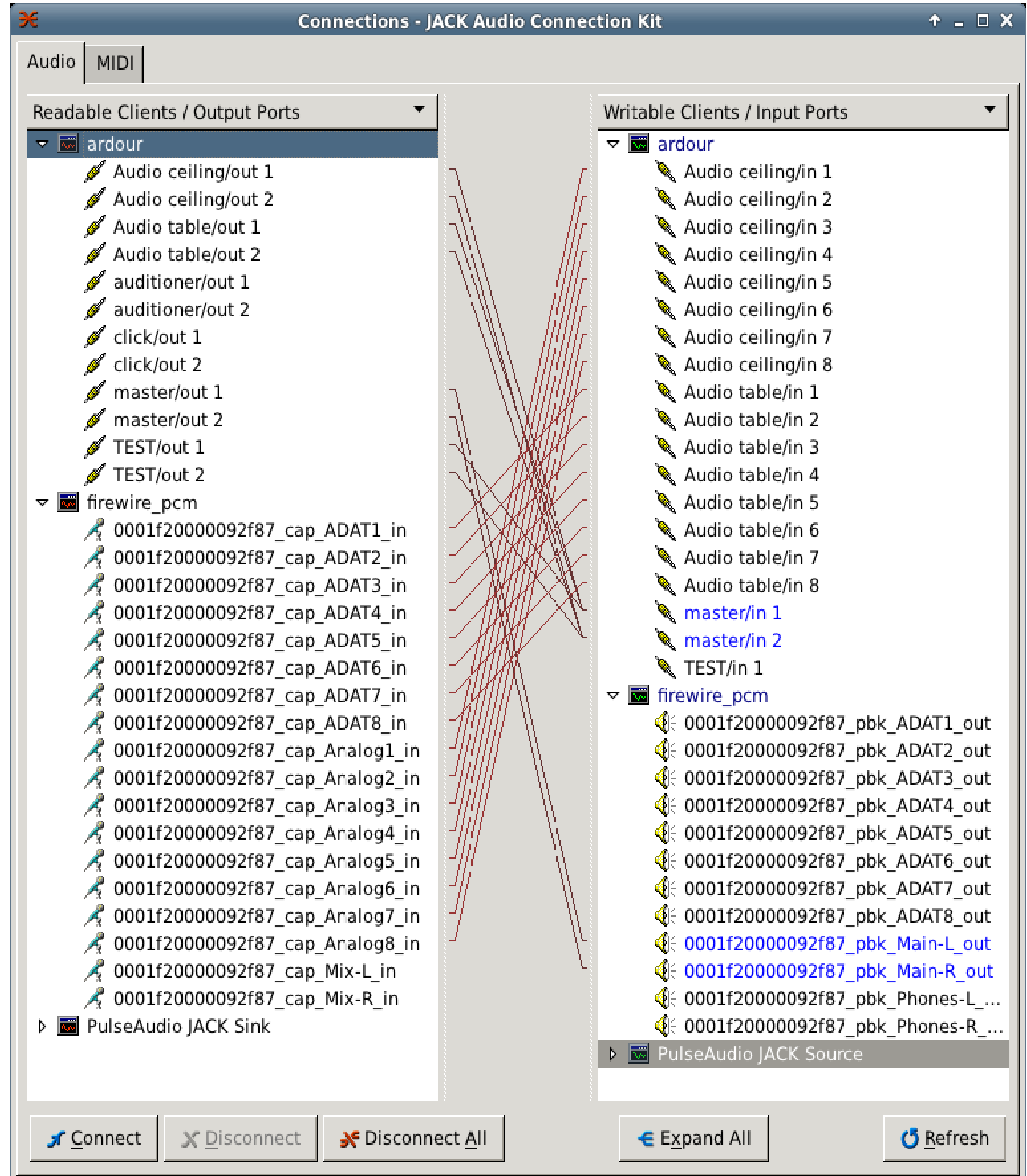
- 16 channels:
  - Up to 8 ceiling mics
  - Up to 8 table mics
  - + headsets
- Two MOTU Pre8s
- Firewire
- 3D people trackers
- 3 Cameras
- Teleconferencing (H323,sip)
- DAW: 3GHz, 4Gb,UbuntuStudio11.10
- **Record 16chs, 48kHz,316chs, 48kHz,32bit**
- **For 1 hour meetings**



Charles Fox www.5m.org.uk LAC2012

Linux meeting recording



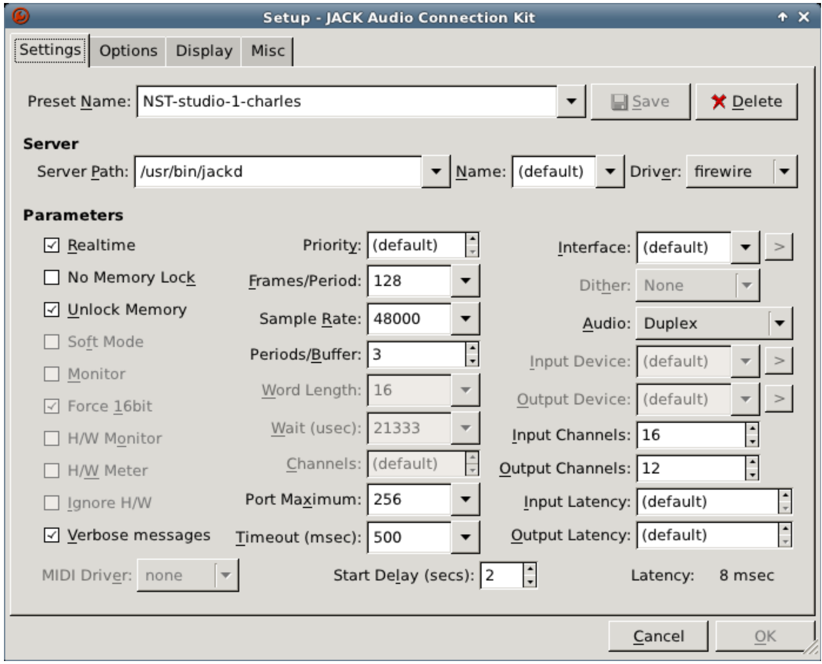


Charles Fox www.5m.org.uk LAC2012

The University Of Sheffield.

Set-up lessons learned

- ffado (firewire) with JACK2 (install Ubuntu packages: jack2d, jack2d-firewire, libffado,jackd, laditools.)
- Unlock memory.
- adduser charles audio.
- edit /etc/security/limits.d/audio.conf
  - @audio - rtprio 95
  - @audio - memlock unlimited
- Qjackctl: 128frms/per,48kHz,3per/buf.



- **Result: Just 11 xruns in 1hour.Only 25% CPU usage on 16ch/16kHz/32bit 2-core 3GHz machine.**

(So looking good for more mics!)

Multi-mic de-mixing

- Speakers may talk over one another
- And background noises interfere with them
- Use microphone arrays to pull out sources:
- Assume each speaker and noise is $x_i[t]$
- So vector of source signals, **x**[t]
- Assume mics get stationary linear mix, **y=Mx**
- Search for parameters M to make $\mathbf{x'}=\mathbf{M}^{-1}\mathbf{y}$ look like **x**
  - ICA: assume sources are independent
  - Speech priors: make **x'** have speechlike harmonics
  - Beamforming: use knowledge of locations and physics...
- **More mics → better accuracy!**

Environment modelling

- Assume each mic channel *j* picks up a source $x_i[t]$ that has been filtered by the room's reverb and resonance, $y_j(t)=\Sigma_d f_{ij}(d)x(t-d)$
- Find filter f to “de-verb” the room and retreive x.
- Again, by making prior assumptions about what we would like x to look like
- More complicated when the multiple source problem is there at the same time...

$$y_j(t)=\Sigma_i \Sigma_d f_{ij}(d)x_i(t-d)$$

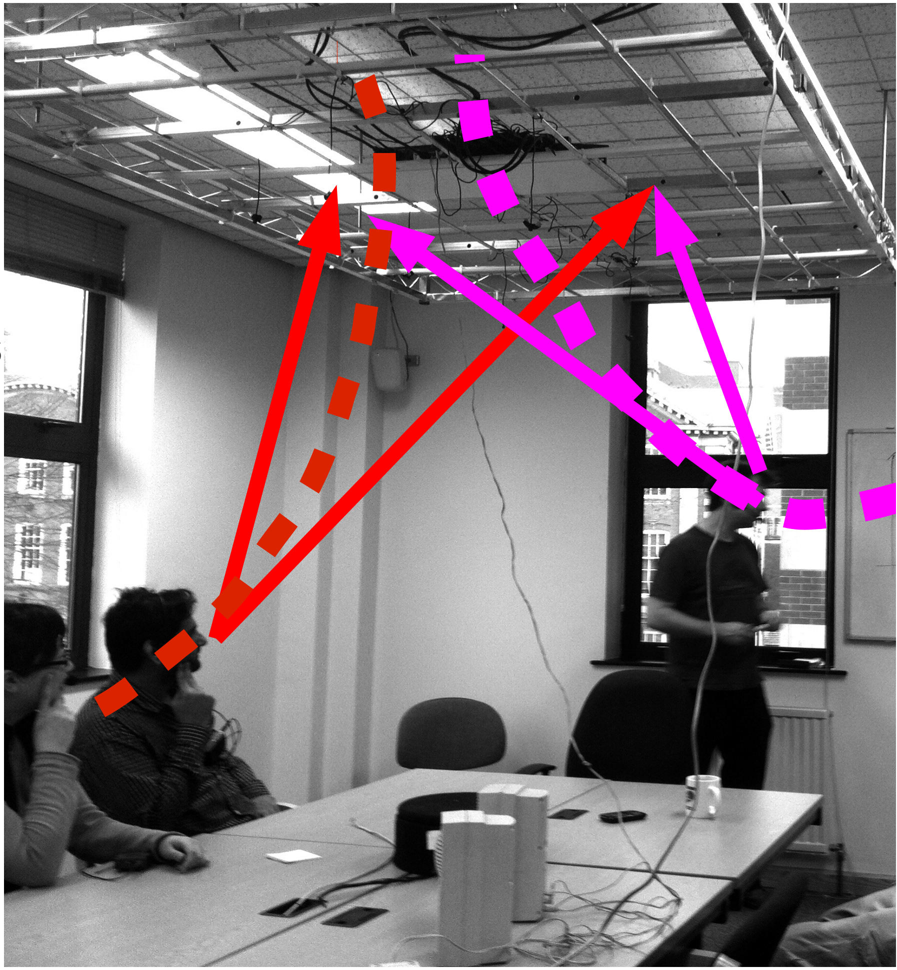
Charles Fox www.5m.org.uk LAC2012

Speaker localisation and beamforming

- Consider the time to reach different mics from the speed of sound.
- **Requires sample-perfect accuracy**.

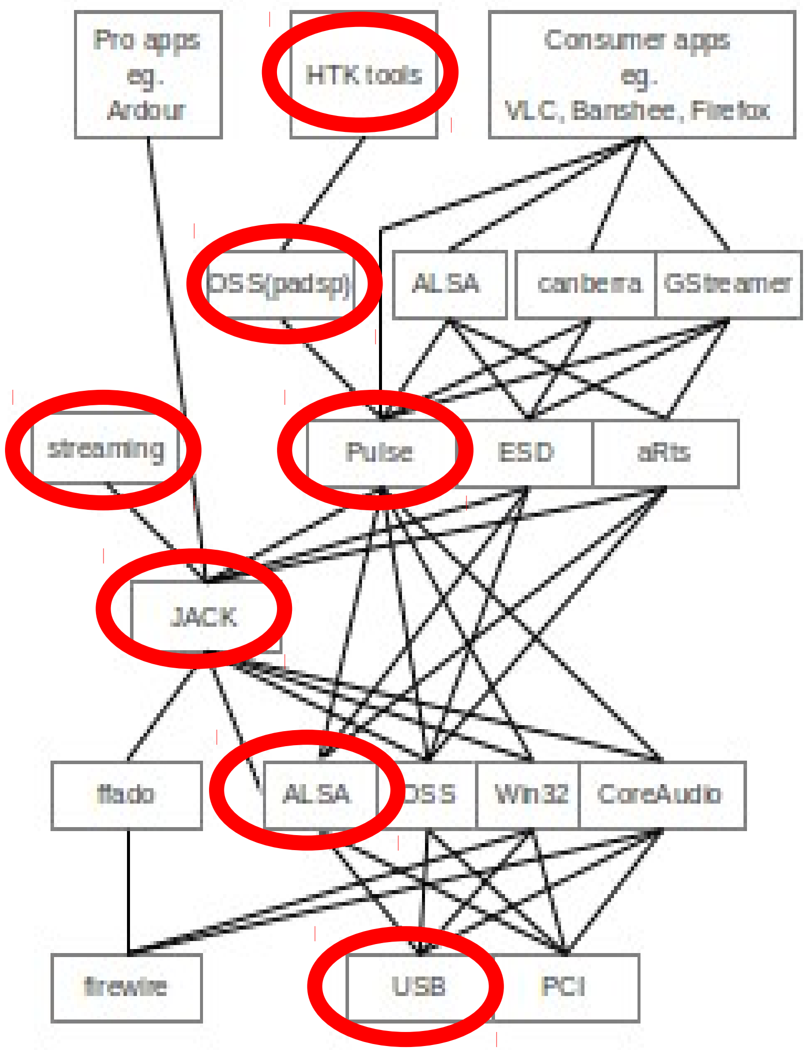
  (1meter in 1/330s = 3ms = 145samples@48kHz)
- **For testing, also sync to our people-tracker and video camera streams (JACK? GStreamer?)**
- Time delay between each pair of mics gives a set of possible locations. Intersecting sets from many pairs are probably source locations.
- Speakers may move around over time

  → Kalman like tracking (+video)
- Once we know the speaker locations, then use physics of sound to do source separation – infer back from constructive and destructive interference patterns - beamforming
- Bayesian probability is a useful framework to try solving **all** of these problems simultaneously...
- Bayes is good at chicken-and-egg problems but needs great computing power (eg. Clusters), especially if we want real-time.



The University Of Sheffield.

Remote NST recognition



Microcones, Android phones, sending speech back to our cluster for analysis.

The University Of Sheffield.

Charles Fox www.5m.org.uk LAC2012

Linux for natural speech

- Unlike standard ASR tasks, we have:
  - Multiple speakers talking at once
  - Speakers moving around the room
  - Significant room effects (reverb,resonance)
  - Background noise (furniture, traffic, plumbing)
- Linux audio requirements:
  - Large data sets (1000 hours) – mostly (entirely?) for offline work
  - Each recording typically 1 hour
  - Many simultaneous recording channels, 16 good, 64+ better ...
  - Sample-perfect synchronisation between mics for beamforms
  - Sync audio to video and 3D data streams – sample accurate
  - HTK/Microsoft licensing issues (or move to Kaldi?)
  - Linking existing HTK (or Kaldi) offline cluster tools to realtime audio – and over networks for remote users, eg. their Android phones.
- What can we contribute back to GPL Linux Audio from the NST Project?
- Please do get in touch now or after the conference: **www.5m.org.uk**

The University Of Sheffield. UNIVERSITY OF CAMBRIDGE THE UNIVERSITY OF EDINBURGH NUANCE GCHQ NHS BBC