# Machine Learning in Speech Synthesis

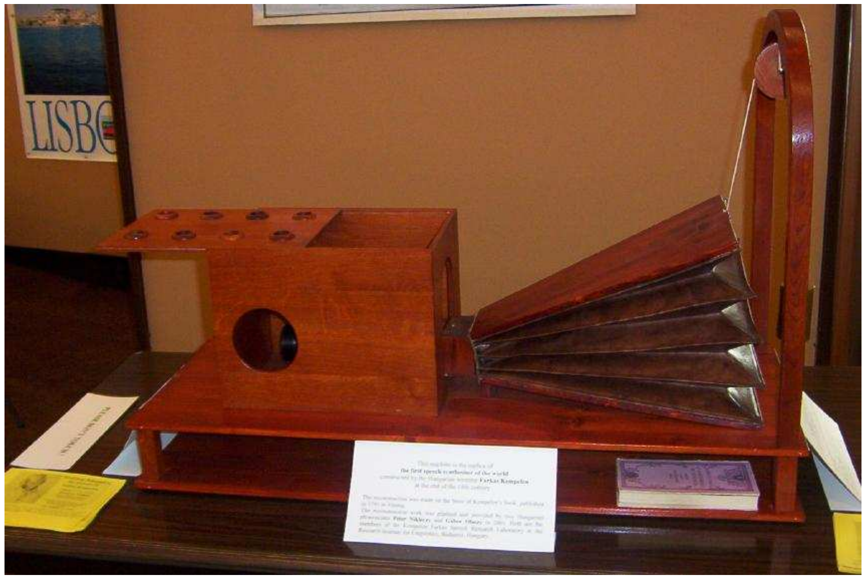
Alan W Black
*Language Technologies Institute*
*Carnegie Mellon University*
*Sept 2009*

## Overview

- Speech Synthesis History and Overview
  - From hand-crafted to data-driven techniques
- Text to Speech Processes
- Waveform synthesis
  - Unit selection and Statistical Parametric Synthesis
- Evaluation
- Voice conversion
- Project ideas

# Physical Models

- Blowing air through tubes…
  - von Kemplen's synthesizer 1791



- Synthesis by physical models
  - Homer Dudley's Voder. 1939

# More Computation – More Data

- *Formant synthesis (60s-80s)*
  - *Waveform construction from components*
- *Diphone synthesis (80s-90s)*
  - *Waveform by concatenation of small number of instances of speech*
- *Unit selection (90s-00s)*
  - *Waveform by concatenation of very large number of instances of speech*
- *Statistical Parametric Synthesis (00s-..)*
  - *Waveform construction from parametric models*

## Waveform Generation

- Formant synthesis
- Random word/phrase concatenation
- Phone concatenation
- Diphone concatenation
- Sub-word unit selection
- Cluster based unit selection
- Statistical Parametric Synthesis

## Speech Synthesis

- *Text Analysis*
  - *Chunking, tokenization, token expansion*
- *Linguistic Analysis*
  - *Pronunciations*
  - *Prosody*
- *Waveform generation*
  - *From phones and prosody to waveforms*

# Text processing

- *Find the words*
  - *Splitting tokens too e.g. “04/11/2009”*
  - *Removing punctuation*
- *Identifying word types*
  - *Numbers: years, quantities, ordinals*
  - *1996 sheep were stolen on 25 Nov 1996*
- *Identifying words/abbreviations*
  - *CIA, 10m, 12sf, WeH7200*

## Pronunciations

- *Giving pronunciation for each word*
  - *A phoneme string (plus tone, stress …)*
- *A constructed lexicon*
  - *(“pencil” n (p eh1 n s ih l))*
  - *(“two” n (t uw1))*
- *Letter to sound rules*
  - *Pronunciation of out of vocabulary words*
  - *Machine learning prediction from letters*

## Pronunciation of Unknown Words

- *How do you pronounce new words*
- *4% of tokens (in news) are new*
- *You can’t synthesis then without pronunciations*
- *You can’t recognize them without pronunciations*
- *Letter-to-Sounds rules*
- *Grapheme-to-Phoneme rules*

## LTS: Hand written

- *Hand written rules*
  - *[LeftContext] X [RightContext] -> Y*
  - *e.g.*
  - *c [h r] -> k*
  - *c [h] -> ch*
  - *c [i] -> s*
  - *c -> k*

# LTS: Machine Learning Techniques

- *Need an existing lexicon*
  - *Pronunciations:  words and phones*
  - *But different number of letters and phones*
- *Need an alignment*
  - *Between letters and phones*
  - *checked -> ch eh k t*

## LTS: alignment

- checked -> ch eh k t

| | | | | | | |
|---|---|---|---|---|---|---|
| c | h | e | c | k | e | d |
| ch | _ | eh | k | _ | _ | t |

- Some letters go to nothing
- Some letters go to two phones
  - box -> b aa k-s
  - table -> t ey b ax-l -

# Find alignment automatically

- *Epsilon scattering*
  - *Find all possible alignments*
  - *Estimate p(L,P) on each alignment*
  - *Find most probable alignment*
- *Hand seed*
  - *Hand specify allowable pairs*
  - *Estimate p(L,P) on each possible alignment*
  - *Find most probable alignment*
- *Statistical Machine Translation (IBM model 1)*
  - *Estimate p(L,P) on each possible alignment*
  - *Find most probably alignment*

## Not everything aligns

- *0, 1, and 2 letter cases*
  - *e -> epsilon   "moved"*
  - *x -> k-s, g-z   "box" "example"*
  - *e -> y-uw   "askew"*
- *Some alignment aren't sensible*
  - *dept -> d ih p aa r t m ax n t*
  - *cmu -> s iy eh m y uw*

## Training LTS models

- *Use CART trees*
  - *One model for each letter*
- *Predict phone (epsilon, phone, dual phone)*
  - *From letter 3-context (and POS)*
- *# # # c h e c -> ch*
- *# # c h e c k -> _*
- *# c h e c k e -> eh*
- *c h e c k e d -> k*

# LTS results

- Split lexicon into train/test 90%/10%
  - i.e. every tenth entry is extracted for testing

| Lexicon | Letter Acc | Word Acc |
|---|---|---|
| OALD | 95.80% | 75.56% |
| CMUDICT | 91.99% | 57.80% |
| BRULEX | 99.00% | 93.03% |
| DE-CELEX | 98.79% | 89.38% |
| Thai | 95.60% | 68.76% |

## Example Tree

```
For letter V:
if (n.name is v)
    return _
    if (n.name is #)
            if (p.p.name is t)
                return f
                return v
            if (n.name is s)
                if (p.p.p.name is n)
                    return f
                    return v
                return v
```

## But we need more than phones

- What about lexical stress
  - p r aa1 j eh k t -> p r aa j eh1 k t
- Two possibilities
  - A separate prediction model
  - Join model – introduce eh/eh1 (BETTER)

| | *LTP+S* | *LTPS* |
|---|---|---|
| *L no S* | *96.36%* | *96.27%* |
| *Letter* | *---* | *95.80%* |
| *W no S* | *76.92%* | *74.69%* |
| *Word* | *63.68%* | *74.56%* |

## Does it really work

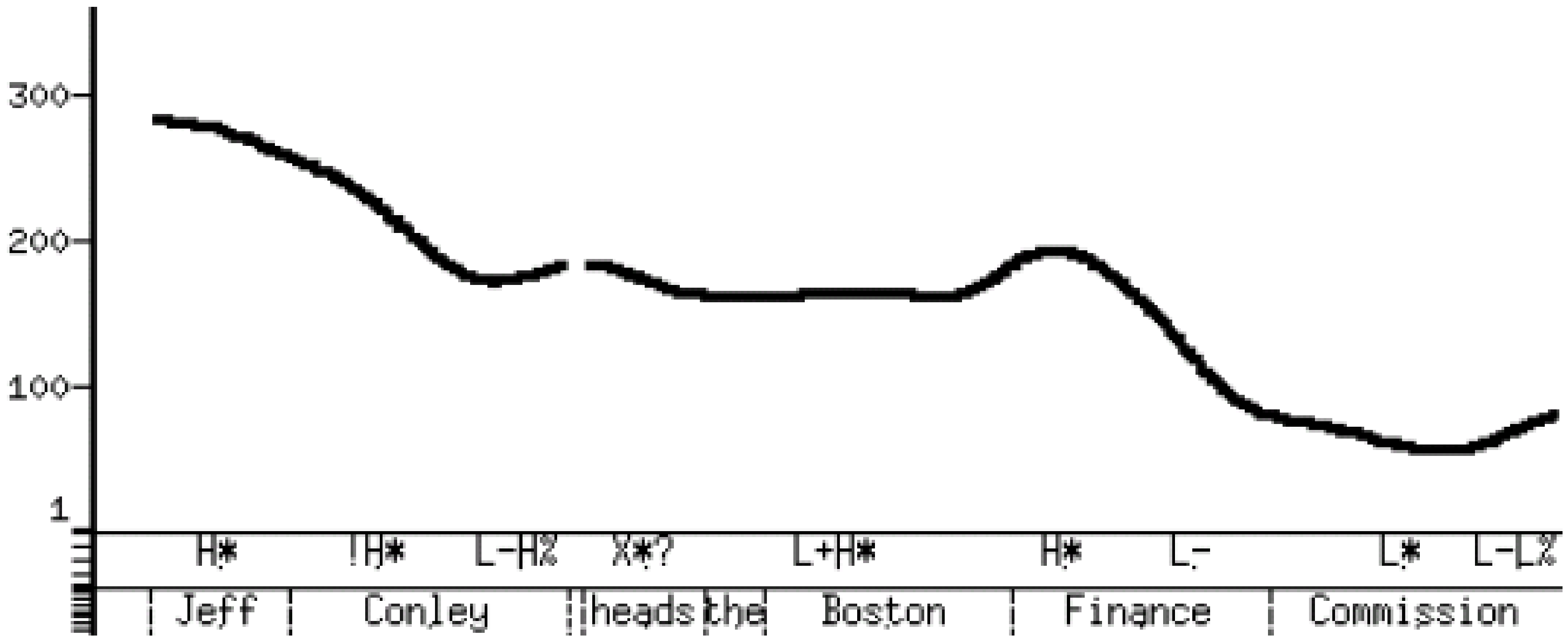
- 40K words from Time Magazine
  - 1775 (4.6%) not in OALD
  - LTS gets 70% correct (test set was 74%)

| | Occurs | % |
|---|---|---|
| Names | 1360 | 76.6 |
| Unknown | 351 | 19.8 |
| US Spelling | 57 | 3.2 |
| Typos | 7 | 0.4 |

# Prosody Modeling

- *Phrasing*
  - *Where to take breaths*
- *Intonation*
  - *Where (and what size) are accents*
  - *F0 realization*
- *Duration*
  - *What is the length of each phoneme*

# Intonation Contour



## Unit Selection vs Parametric

Unit Selection

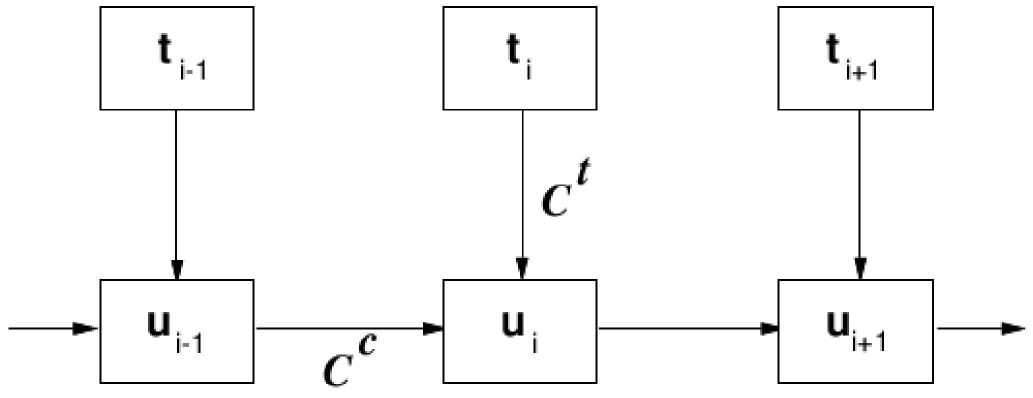
- The “standard” method
- “*Select appropriate sub-word units from large databases of natural speech*”

Parametric Synthesis:  [NITECH: Tokuda et al]

- HMM-generation based synthesis
- Cluster units to form models
- Generate from the models
- “*Take ‘average’ of units*”

## Unit Selection

- Target cost and Join cost [Hunt and Black 96]
  - Target cost is distance from desired unit to actual unit in the databases
    - Based on phonetic, prosodic metrical context
  - Join cost is how well the selected units join



# “Hunt and Black” Costs

Target distance is:

– $C^t(t_i, u_i) = \Sigma_{j=1}^{p} w_j^t C_j^t(t_i, u_i)$

For examples *in the database* we can measure

– $AC^t(t_i, u_i)$

Therefore estimate $w_{1-j}$ from all examples of

– $AC^t(t_i, u_i) \approx \Sigma_{j=1}^{p} w_j^t C_j^t(t_i, u_i)$

Use linear regression

How well does it join:

– $C^c(u_{i-1}, u_i) = \Sigma_{k=1}^{p} w_k^c C_k^c(u_{i-1}, u_i)$

– if $(u_{i-1} == \text{prev}(u_i))\ C^c = 0$

## HB Unit Selection

Find best path of units through db that minimise:

$$C(t_1^n, u_1^n) = \Sigma_{i=1}^n C^t(t_i, u_i) + \Sigma_{i=2}^n C^c(u_{i-1}, u_i) + C^c(S, u_1) + C^c(u_n, S)$$

- Use Viterbi to find best set of units
- Note
  - Finding “longest” is typically not optimal

# Clustering Units

- Cluster units [Donovan et al 96, Black et al 97]

$$Adist(U,V) = \begin{cases} \text{if } |V| > |U| \quad Adist(V,U) \\ \frac{WD*|U|}{|V|} * \sum_{i=1}^{|U|} \sum_{j=1}^{n} \frac{W_j.(abs(F_{ij}(U) - F_{(i*|V|/|U|)j}(V)))}{SD_j * n * |U|} \end{cases}$$

$|U|$ = number of frames in $U$

$F_{xy}(U)$ = parameter $y$ of frame $x$ of unit $U$

$SD_j$ = standard deviation of parameter $j$

$W_j$ = weight for parameter $j$

$WD$ = duration penalty

- Moves calculation to compile time

# Unit Selection Issues

- Cost metrics
  - Finding best weights, best techniques etc
- Database design
  - Best database coverage
- Automatic labeling accuracy
  - Finding errors/confidence
- Limited domain:
  - Target the databases to a particular application
  - Talking clocks
  - Targeted domain synthesis

# Old vs New

Unit Selection:

- large carefully labelled database
- quality good when good examples available
- quality will sometimes be bad
- no control of prosody

Parametric Synthesis:

- smaller less carefully labelled database
- quality consistent
- resynthesis requires vocoder, (buzzy)
- can (must) control prosody
- model size much smaller than Unit DB

## Parametric Synthesis

- Probabilistic Models

$$argmax(P(O|W))$$

- Simplification

$$argmax(P(o_0|W), P(o_1|W), ..., P(o_n|W))$$

- Generative model
  - Predict acoustic frames from text

# Trajectories

- *Frame (State) based prediction*
  - *Ignores dynamics*
- *Various solutions*
  - *MLPG (maximum likelihood parameter generation)*
  - *Trajectory HMMs*
  - *Global Variance*
  - *MGE, minimal generation error*

## SPSS Systems

- *HTS (NITECH)*
  - *Based on HTK*
  - *Predicts HMM-states*
  - *(Default) uses MCEP and MLSA filter*
  - *Supported in Festival*
- *Clustergen (CMU)*
  - *No use of HTK*
  - *Predicts Frames*
  - *(Default) uses MCEP and MLSA filter*
  - *More tightly coupled with Festival*

## Synthesizer

Requires:

- Prompt transcriptions (txt.done.data)
- Waveform files (well recorded)

FestVox Labelling

- EHMM (Kishore)
- Context Independent models and forced alignment
- (Have used Janus labels too).

Parameter extraction:

- (HTS's) melcep/mlsa filter for resynthesis
- F0 extraction

Clustering

- Wagon vector clustering
- for each HMM-state name

## Clustering by CART

Update to Wagon (Edinburgh Speech Tools).

- Tight coupling of features with FestVox utts
- Support for arbitrary vectors
- Define impurity on clusters of $N$ vectors

$$\left(\sum_{i=1}^{24} \sigma_i\right) * N$$

Clustering

- F0 and MCEP
- Tested jointly and separately
- Features for clustering (51):
  - phonetic, syllable, phrasal context

## Training Output

Three models:

- Spectral (MCEP) CART tree
- F0 CART tree
- Duration CART tree

F0 model:

- Smoothed extracted F0 through all speech
- (i.e. unvoiced regions get F0 values)
- Chose voicing at runtime phonetically

## CLUSTERGEN Synthesis

Generate phoneme strings (as before)

For each phone:

- Find HMM-state names: ah_1, ah_2, ah_3
- Predict duration of each
- Create empty mcep vector to fill duration
- Predict mcep values from cluster tree
- Predict F0 value from cluster tree

Use MLSA filter to regenerate speech

## Objective Score

*CLUSTERGEN*

*Mean Mel Cepstral Distortion over test set*

$$10/\ln 10\sqrt{2\sum_{d=1}^{24}\left(mc_d^{(t)} - mc_d^{(e)}\right)^2}$$

*MCD: Voice Conversion ranges 4.5-6.0*

*MCD: CG scores 4.0-8.0*

*smaller is better*

# Example CG Voices

7 Arctic databases:

1200 utterances, 43K segs, 1hr speech

| | | | |
|---|---|---|---|
| awb | 🔊 | bdl | 🔊 |
| clb | 🔊 | jmk | 🔊 |
| ksp | 🔊 | rms | 🔊 |
| slt | 🔊 | | |

## Database size vs Quality

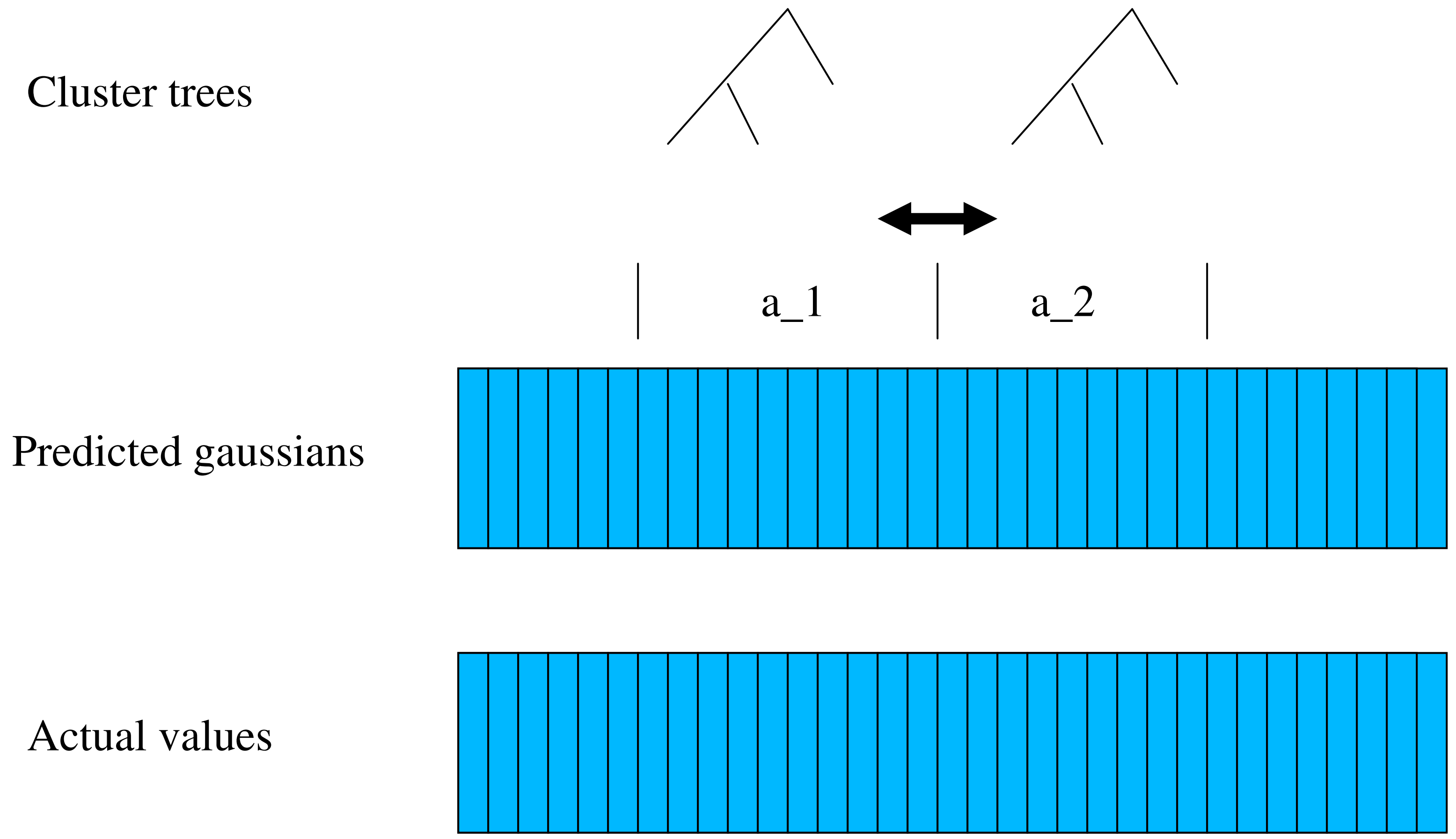
slt_arctic data size

| Utts | Clusters | RMS F0 | MCD | |
|---|---|---|---|---|
| 50 | 230 | 24.29 | 6.761 | 🔊 |
| 100 | 435 | 19.47 | 6.278 | 🔊 |
| 200 | 824 | 17.41 | 6.047 | 🔊 |
| 500 | 2227 | 15.02 | 5.755 | 🔊 |
| 1100 | 4597 | 14.55 | 5.685 | 🔊 |

## Making it Better

- *Label data, build model*
- *But maybe there are better labels*
- *So find labels that maximize model accuracy*

# Move Labels



## Move Labels

- *Use EHMM to label segments/HMM states*
- *Build Clustergen Model*
- *Iterate*
  - *Predict Cluster (mean/std) for each frame*
  - *For each label boundary*

```
If dist(actual_after,pred_before) <
  dist(actual_after,pred_after)
    Move label forward
If dist(actual_before,pred_after) <
  dist(actual_before,pred_before)
    Move label backward
```

# Distance Metric

- *Distance from predicted to actual*
  - *Euclidean*
  - *F0, static, deltas, voicing*
  - *With/without standard deviation normalization*
  - *Weighting*
- *Best choice*
  - *Static without stddev normalization*
  - *(This is closest to MCD)*

# ML with 10 iterations

- rms voice (66 minutes of speech)
  - train 1019 utts, test 113 utts (every tenth)

| **Pass** | **Move** | **+ve** | **-ve** | **MCD** | **stddev** | **F0** |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 5.247 | 1.965 | 13.990 |
| 1 | 48211 | 23162 | 25949 | 5.121 | 1.846 | 14.251 |
| 2 | 40731 | 20223 | 20508 | 5.090 | 1.794 | 14.220 |
| 3 | 35059 | 17835 | 17224 | 5.073 | 1.779 | 14.267 |
| 4 | 33083 | 16503 | 16580 | 5.061 | 1.765 | 14.260 |
| 5 | 31131 | 15518 | 15613 | 5.046 | 1.753 | 14.306 |
| 6 | 29693 | 14813 | 14880 | 5.042 | 1.754 | 14.287 |
| 7 | 28361 | 14143 | 14218 | 5.042 | 1.757 | 14.240 |
| **8** | 27571 | 13730 | 13841 | **5.035** | 1.740 | 14.239 |
| 9 | 26839 | 13457 | 13382 | 5.040 | 1.750 | 14.187 |

## Move Labels

| ***Voice*** | ***2006*** | ***2008 base*** | ***2008 ml*** |
|---|---|---|---|
| *ahw* | *-* | *5.234* | *5.057* |
| *awb* | *6.557* | *4.445* | *4.483* |
| *bdl* | *6.129* | *5.685* | *5.467* |
| *clb* | *5.417* | *4.838* | *4.698* |
| *jmk* | *6.165* | *5.398* | *5.239* |
| *ksp* | *5.980* | *5.289* | *5.140* |
| *rms* | *5.731* | *5.247* | *5.035* |
| *rxr* | *-* | *5.298* | *5.160* |
| *slt* | *5.713* | *5.170* | *4.983* |

Average improvement 0.172 (excluding awb)

# Does it **sound** better

- rms
  - abtest (10 utterances)

    **ml** 7

    **base** 1

    **=** 2

- Slt

  - abtest

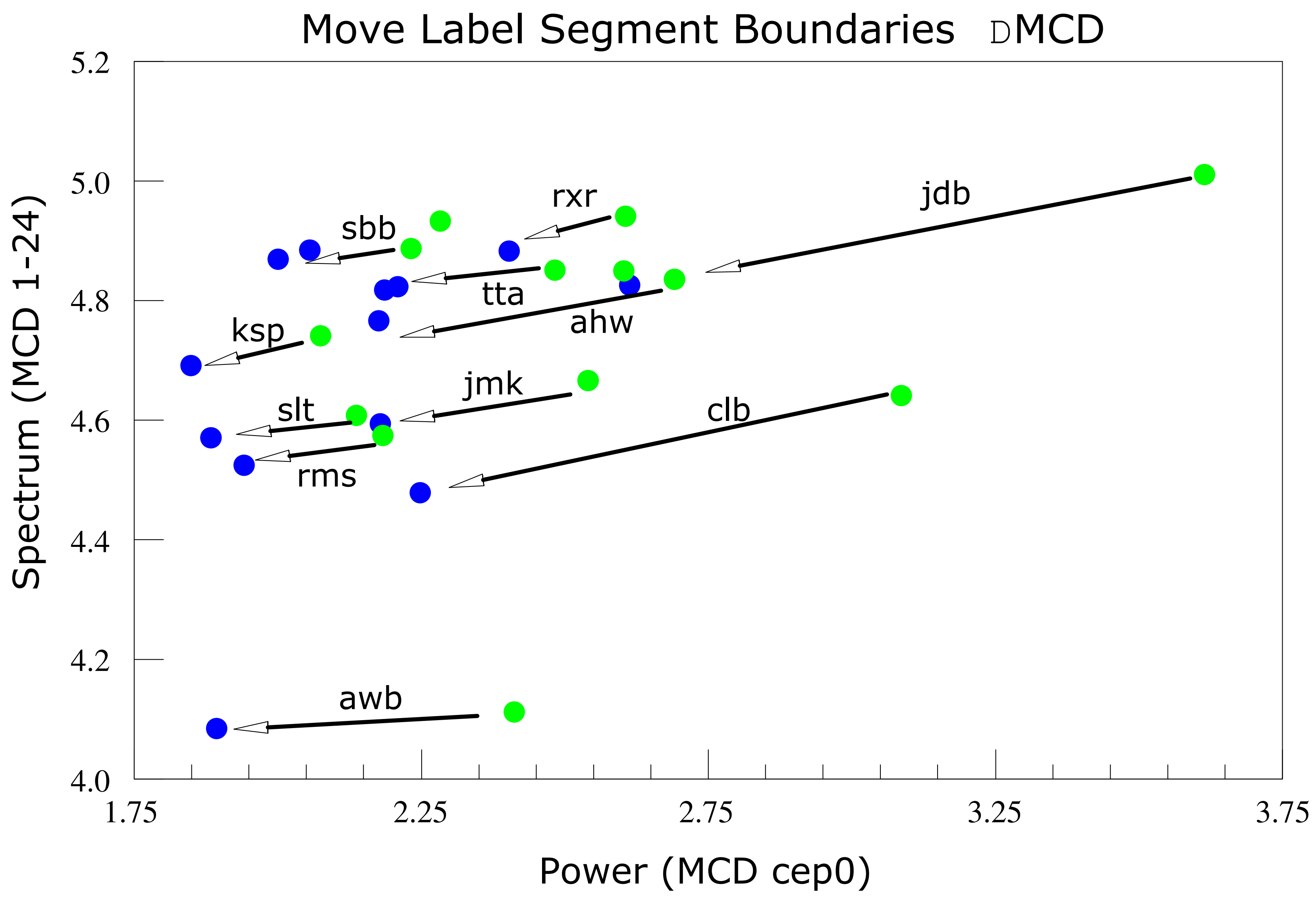
    **ml** 7

    **base** 2

    **=** 1

| | base | ml |
|---|---|---|
| | 🔊 | 🔊 |
| | 🔊 | 🔊 |
| | 🔊 | 🔊 |

# Arctic MLSB improvements



# Grapheme Based Synthesis

- *Synthesis without a phoneme set*
- *Use the letters as phonemes*
  - *("alan" nil (a l a n))*
  - *("black" nil ( b l a c k ))*
- *Spanish (easier ?)*
  - *419 utterances*
  - *HMM training to label databases*
  - *Simple pronunciation rules*
  - *Polici'a -> p o l i c i' a*
  - *Cuatro -> c u a t r o*

## Spanish Grapheme Synthesis

| Word | Castillian | gloss |
|---|---|---|
| **c**asa | /**k** a s a/ | house |
| **c**esa | /**th** e s a/ | stop |
| **c**ine | /**th** i n e/ | cinema |
| **c**osa | /**k** o s a/ | thing |
| **c**una | /**k** u n a/ | cradle |
| he**c**hizo | /e **ch** i th o/ | charm, spell |

In Spanish the letter "c" may be pronounced /k/, /ch/ and /th/ or /s/ (depending on dialect). The choice of phone is determined by the letter context.

## English Grapheme Synthesis

- Use Letters are phones
- 26 “phonemes”
  - ( “alan” n (a l a n))
  - ( “black” n (b l a c k))
- Build HMM acoustic models for labeling
- For English
  - “This is a pen”
  - “We went to the church at Christmas”
  - Festival intro
  - “do eight meat”
- Requires method to fix errors
  - Letter to letter mapping

## Common Data Sets

- *Data drive techniques need data*
- *Diphone Databases*
  - *CSTR and CMU US English Diphone sets (kal and ked)*
- *CMU ARCTIC Databases*
  - *1200 phonetically balanced utterances (about 1 hour)*
  - *7 different speakers (2 male 2 female 3 accented)*
  - *EGG, phonetically labeled*
  - *Utterances chosen from out-of-copyright text*
  - *Easy to say*
  - *Freely distributable*
  - *Tools to build your own in your own language*

## Blizzard Challenge

- *Realistic evaluation*
  - *Under the same conditions*
- *Blizzard Challenge [Black and Tokuda]*
  - *Participants build voice from common dataset*
  - *Synthesis test sentences*
  - *Large set of listening experiments*
  - *Since 2005, now in 4th year*
  - *18 groups in 2008*

# How to test synthesis

- *Blizzard tests:*
  - *Do you like it?  (MOS scores)*
  - *Can you understand it?*
    - *SUS sentence*
    - *The unsure steaks overcame the zippy rudder*
- *Can't this be done automatically?*
  - *Not yet (at least not reliably enough)*
  - *But we now have lots of data for training techniques*
- *Why does it still sound like robot?*
  - *Need better (appropriate testing)*

# SUS Sentences

- *sus_00022*
- *sus_00012*
- *sus_00005*
- *sus_00017*

# SUS Sentences

- *The serene adjustments foresaw the acceptable acquisition*
- *The temperamental gateways forgave the weatherbeaten finalist*
- *The sorrowful premieres sang the ostentatious gymnast*
- *The disruptive billboards blew the sugary endorsement*

## Voice Identity

- *What makes a voice identity*
  - *Lexical Choice:*

    *Woo-hoo,*

    *I pity the fool …*
  - *Phonetic choice*
  - *Intonation and duration*
  - *Spectral qualities (vocal tract shape)*
  - *Excitation*

## Voice Conversion techniques

- *Full ASR and TTS*
  - *Much too hard to do reliably*
- *Codebook transformation*
  - *ASR HMM state to HMM state transformation*
- *GMM based transformation*
  - *Build a mapping function between frames*

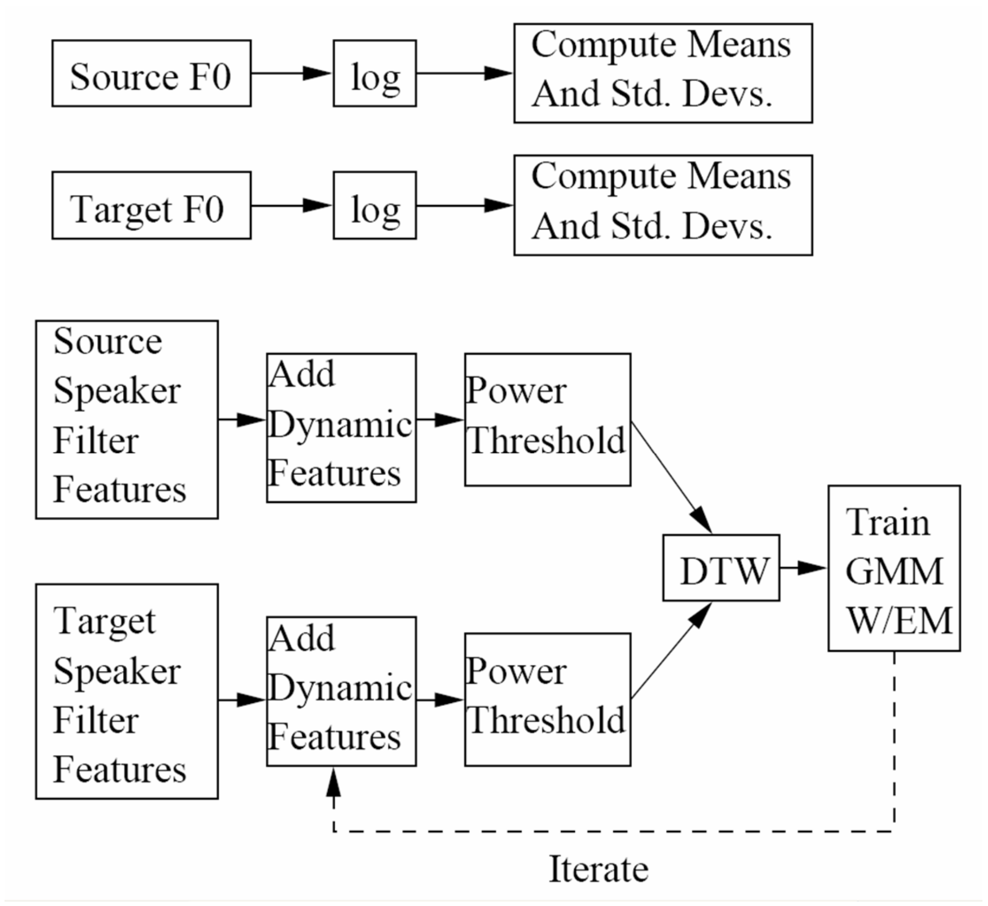
# Learning VC models

- *First need to get parallel speech*
  - *Source and Target say same thing*
  - *Use DTW to align (in the spectral domain)*
  - *Trying to learn a functional mapping*
  - *20-50 utterances*
- *"Text-independent" VC*
  - *Means no parallel speech available*
  - *Use some form of synthesis to generate it*

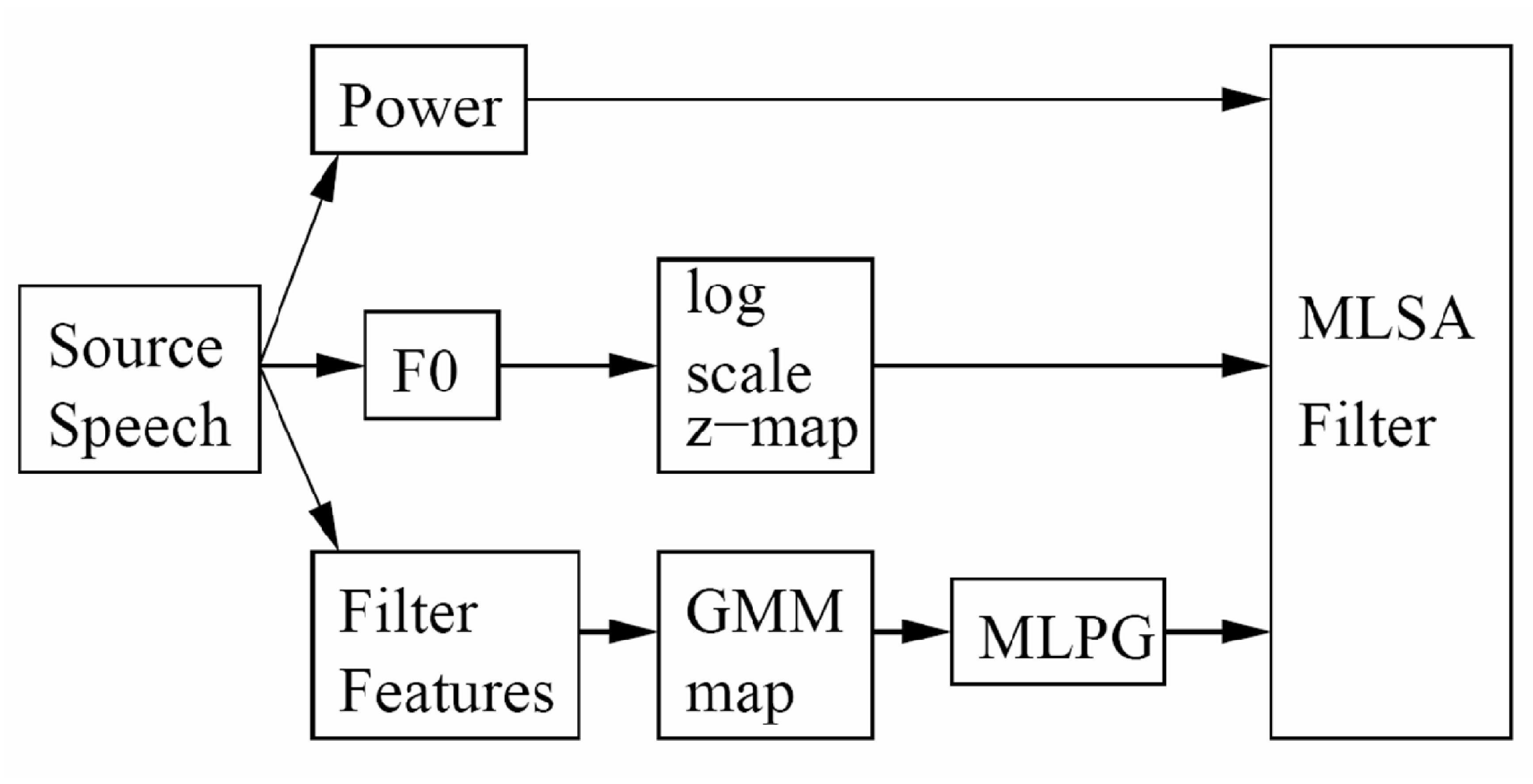
## VC Training process

- *Extract F0, power and MFCC from source and target utterances*
- *DTW align source and target*
- *Loop until convergence*
  - *Build GMM to map between source/target*
  - *DTW source/target using GMM mapping*

## VC Training process



# VC Run-time


Source Speech
Power
F0
Filter Features
log scale z−map
GMM map
MLPG
MLSA Filter

## Voice Transformation

- Festvox GMM transformation suite (Toda)

| | awb | bdl | jmk | slt |
|---|---|---|---|---|
| awb | 🔈 | 🔈 | 🔈 | 🔈 |
| bdl | 🔈 | 🔈 | 🔈 | 🔈 |
| jmk | 🔈 | 🔈 | 🔈 | 🔈 |
| slt | 🔈 | 🔈 | 🔈 | 🔈 |

## VC in Synthesis

- *Can be used as a post filter in synthesis*
  - *Build kal_diphone to target VC*
  - *Use on all output of kal_diphone*
- *Can be used to convert a full DB*
  - *Convert a full db and rebuild a voice*

## Style/Emotion Conversion

- *Unit Selection (or SPS)*
  - *Require lots of data in desired style/emotion*
- *VC technique*
  - *Use as filter to main voice (same speaker)*
  - *Convert neutral to angry, sad, happy …*

## Can you say that again?

- *Voice conversion for speaking in noise*
- *Different quality when you repeat things*
- *Different quality when you speak in noise*
  - *Lombard effect (when very loud)*
  - *“Speech-in-noise” in regular noise*

## Speaking in Noise (Langner)

- Collect data
  - Randomly play noise in person's ears
  - Normal
  - In Noise
- Collect 500 of each type
- Build VC model
  - Normal -> in-Noise
- Actually
  - Spectral, duration, f0 and power differences

# Synthesis in Noise

- *For bus information task*
- *Play different synthesis information utts*
  - *With SIN synthesizer*
  - *With SWN synthesizer*
  - *With VC (SWN->SIN) synthesizer*
- *Measure their understanding*
  - *SIN synthesizer better (in Noise)*
  - *SIN synthesizer better (without Noise for elderly)*

# Transterpolation

- *Incrementally transform a voice X%*
  - *BDL-SLT by 10%*
  - *SLT-BDL by 10%*
- *Count when you think it changes from M-F*
- *Fun but what are the uses …*

# De-identification

- *Remove speaker identity*
  - *But keep it still human like*
- *Health Records*
  - *HIPAA laws require this*
  - *Not just removing names and SSNs*
- *Remove identifiable properties*
  - *Use Voice conversion to remove spectral*
  - *Use F0/duration mapping to remove prosodic*
  - *Use ASR/MT techniques to remove lexical*

## Summary

- *Data-driven speech synthesis*
  - *Text processing*
  - *Prosody and pronunciation*
  - *Waveform synthesis*
- *Finding the right optimization*
  - *Find an objective metric that correlates with human perception*

## Potential Projects

- *Find F0 in story telling*
  - *F0 is easy to find in isolated sentences*
  - *What about full paragraphs*
  - *Storytellers use much wider range*
- *Find F0 shapes/accent types*
  - *Use HMM to recognize “types” of accents*
  - *(trajectory modeling)*
  - *Following “tilt” and Moeller model*

## Parametric Synthesis

- *Better parametric representation of speech*
  - *Particularly excitation parameterization*
- *Better Acoustic measures of quality*
  - *Use Blizzard answers to build/check objective measure*
- *Statistical Klatt Parametric synthesis*
  - *Using "knowledge-base" parameters*
  - *F0, aspiration, nasality, formants*
  - *Automatically derive Klatt parameters for db*
  - *Use them for statistical parametric synthesis*