Vowel resynthesis workshop

Daniel Lawrence
School of Philosophy, Psychology & Language Sciences
The University of Edinburgh

THE UNIVERSITY *of* EDINBURGH

E·S·R·C ECONOMIC & SOCIAL RESEARCH COUNCIL

Motivation

Speech perception research often calls for fine-grained control over parameters of speech stimuli

Trade-off between naturalness and control

Detailed description and evaluation of speech synthesis methods lacking

Often limited to one sentence or less in description of methods, despite wide range of experimenter degrees of freedom

'Naturalness' may be particularly important for sociophonetic perception research

Outline

Logistics

Possible to re-group based on experience with Praat scripting?

Demo scripts available at:
`https://github.com/danielplawrence/synthesis_workshop`

Latest Praat available at:
`http://www.fon.hum.uva.nl/praat/`

I will set small tasks for you to try using these scripts

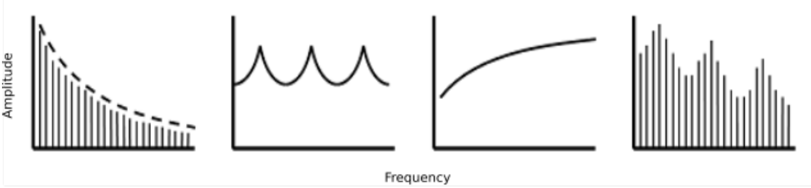
Disclaimer: Praat scripts are notoriously unstable across versions and platforms!

Logistics

Source-filter theory

Source-filter theory (Fant, 1960):

Treat the speech signal as a combination of an excitation source and vocal tract resonances, plus the effect of radiation from the lips:



Source (represents glottal flow)

Filter (represents vocal tract setting)

Radiation characteristic (represents effect of sound passing out of vocal tract and into the world)

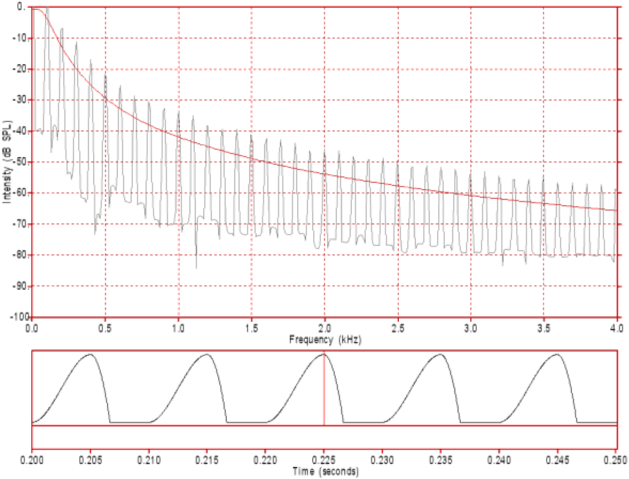
Speech spectrum

Source-filter theory

To synthesize speech, we need to simulate an excitation source and boost/attenuate the spectrum to simulate the vocal tract resonances.

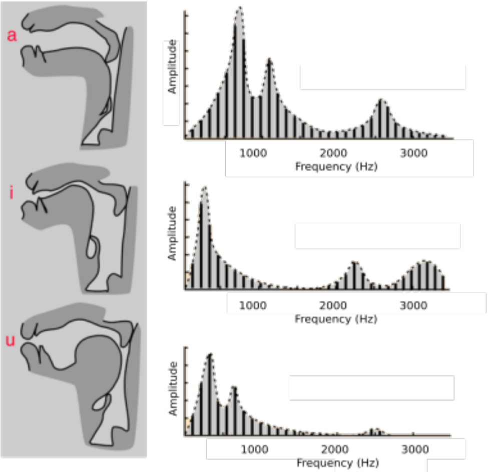
The acoustic characteristics of the voicing source and vocal tract resonances can be simulated digitally.

The source is simulated as a series of repeating pulses.



This produces a complex wave with an amplitude peak at the fundamental frequency and relatively smooth amplitude decay with increasing frequency.

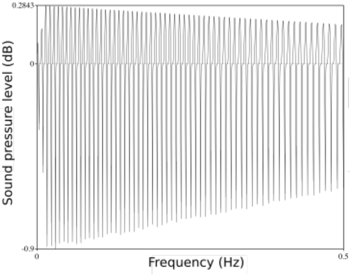
The vocal tract resonances are represented by a digital filter, which boosts and attenuates frequencies of the source.

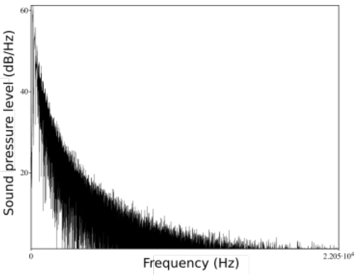


First, we generate an impulse train using Praat's *PointProcess* object:

```
pitchTier = Create PitchTier: "source", 0, 0.5
Add point: 0.0, 150
Add point: 0.5, 100
pulses = To PointProcess
source = To Sound (phonation): 44100, 0.6, 0.05,
    0.7, 0.03, 3.0, 4.0
removeObject: pitchTier, pulses
selectObject: source
View & Edit
```

Acoustic synthesis using Praat





Acoustic synthesis using Praat

Now, we create a vocal tract transfer function using a FormantGrid object:

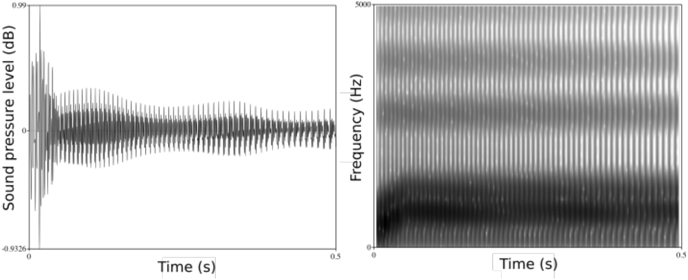
```
filter=Create FormantGrid: "filter", 0, 0.5, 10,
    550, 1100, 60, 50
Remove formant points between: 1, 0, 0.5
Add formant point: 1, 0.00, 350
Add formant point: 1, 0.05, 700
Remove formant points between: 2, 0, 0.5
Add formant point: 2, 0.00, 700
Add formant point: 2, 0.05, 1100
selectObject: filter
View & Edit
```

Acoustic synthesis using Praat

Filtering the FormantGrid and the source together produces a vowel-like sound:

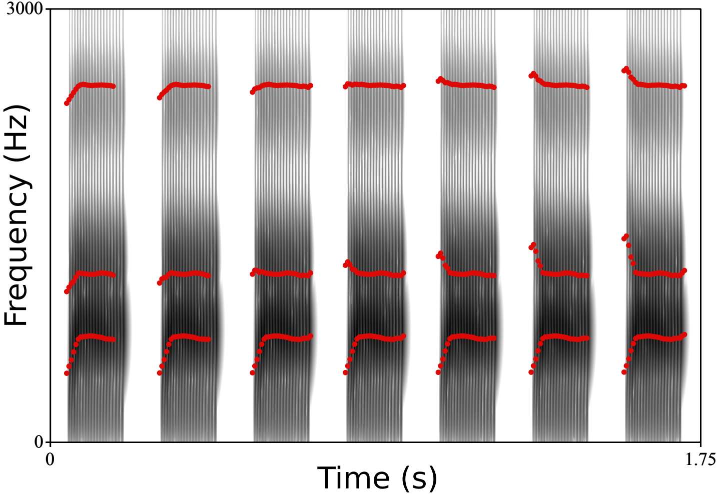
```
selectObject: source, filter
vowel=Filter
selectObject: vowel
View & Edit
```

Acoustic synthesis using Praat



Acoustic synthesis using Praat

Practical example: ba-da continuum



The Klatt synthesizer

Praat also includes an implementation of the Klatt synthesizer

To use this, specify parameters as a function of time in a ‘KlattGrid’ object

```
#Create a KlattGrid
Create KlattGrid... aa 0 0.5 6 1 1 6 1 1 1
```

Logistics
Source-filter theory
Acoustic synthesis using Praat
Spectral envelope estimation and inverse filtering
Inverse filtering using Praat
Advanced inverse filtering methods
The TANDEM-STRAIGHT vocoder

The Klatt synthesizer

Klatt synthesis in Praat:

```
#Add voicing amplitude, vowel formants, and pitch
    targets
Add voicing amplitude point... 0.0 0
Add voicing amplitude point... 0.04 90
Add voicing amplitude point... 0.25 90
Add voicing amplitude point... 0.5 90
Add pitch point... 0.0 150
Add pitch point... 0.5 150
```

Klatt synthesis in Praat:

```
Add oral formant frequency point... 1 0.1 750
Add oral formant bandwidth point... 1 0.1 70
Add oral formant frequency point... 2 0.1 1250
Add oral formant bandwidth point... 2 0.1 120
Add oral formant frequency point... 3 0.1 2500
Add oral formant bandwidth point... 3 0.1 200
Add oral formant frequency point... 4 0.1 3900
Add oral formant bandwidth point... 4 0.1 300
#Synthesis
Play
To Sound
```

Review

Try modifying the code in source_filter.praat to alter the pitch contour of the vowel

The vowel produced by source_filter.praat has a unnaturally flat intensity contour. Could you modify this using an IntensityTier? (see ba-da_continuum.praat for an example)

Try modifying the code in source_filter.praat to create different vowels (there's a list of formant frequencies at `http://bit.ly/1RPZizv`)

Try modifying the code in ba-da_continuum.praat to create a different kind of continuum (e.g. [i] to [u])

Try running klatt_cardinal.praat to generate a set of vowels using the Klatt synthesizer

Spectral envelope estimation and inverse filtering

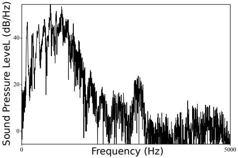
Some issues with the approach we have used so far:

- Modeled formant contours as straight lines
- Modeled the source as regular and fully periodic

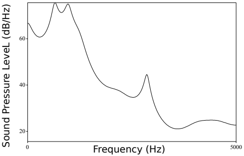
...if we could extract these characteristics from natural speech, we could manipulate them independently and recombine them.

Spectral envelope estimation and inverse filtering

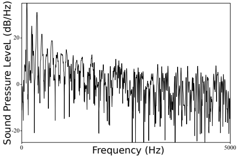
Intuition: estimate the spectral envelope of a natural sample, then use it as an *inverse filter* to ‘undo’ the filtering effect of the supralaryngeal vocal tract.



(a) Natural speech



(b) Spectral envelope

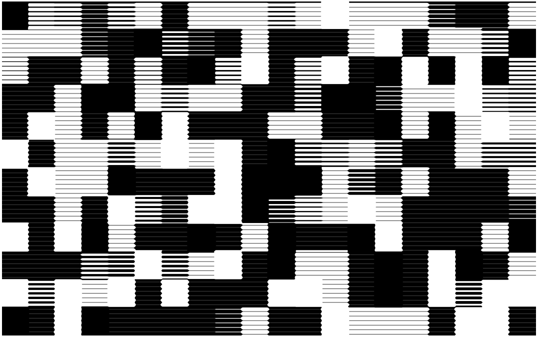


(c) Inverse filtered spectrum

Linear Predictive Coding

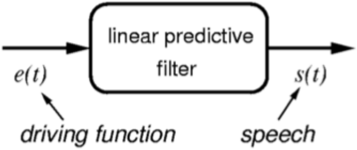
A technique for estimating the spectral envelope of a time-varying speech signal.

Origins in early television encoding – black and white TV images are 3D grids (x,y,brightness) which vary over time.



However, values do not vary randomly – the brightness of any point is likely to be related to the adjacent points

Linear Predictive Coding



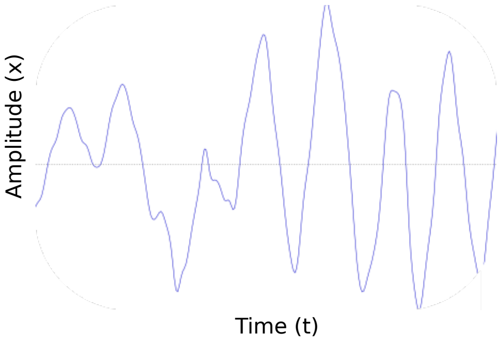
$$\hat{s}(t) = \sum_{j=1}^{p} a_j s(t-j)$$

$$s(t) = e(t) + \sum_{j=1}^{p} a_j s(t-j)$$

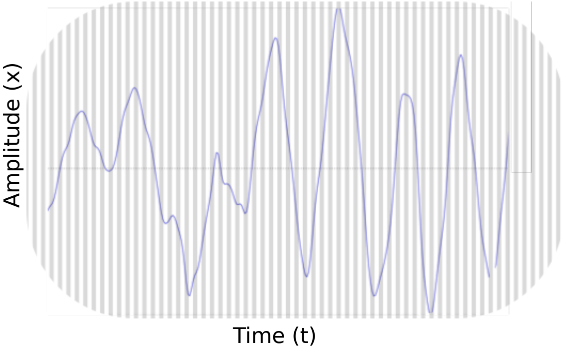
t = discrete time; p = filter order

$=$'Estimate the value at time t as a linear combination of p previous samples'

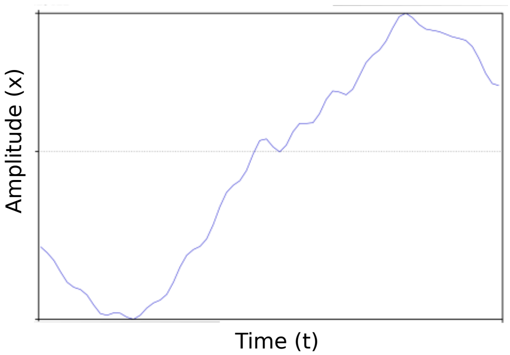
Linear Predictive Coding



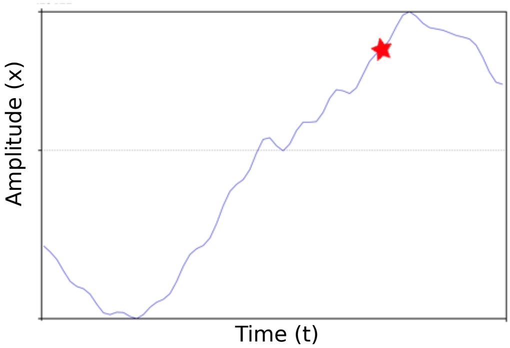
Linear Predictive Coding



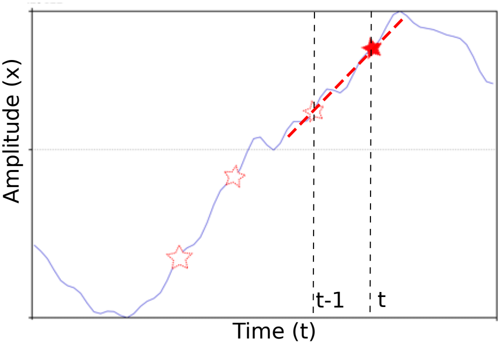
Linear Predictive Coding



Linear Predictive Coding

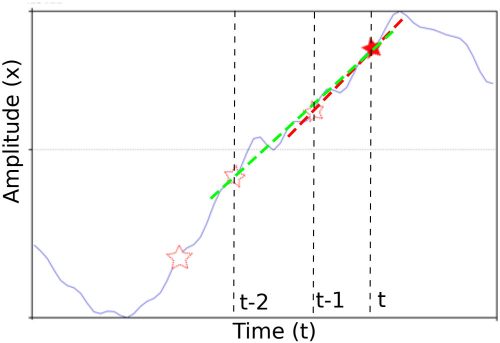


Linear Predictive Coding



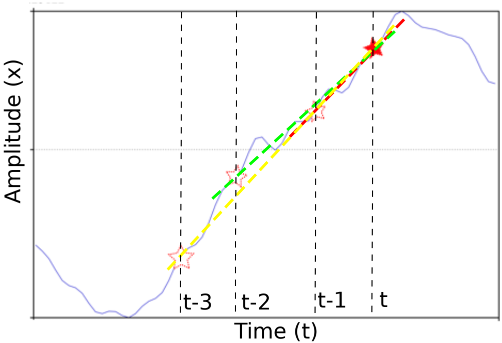
$$x(t) = ax(t-1)$$

Linear Predictive Coding



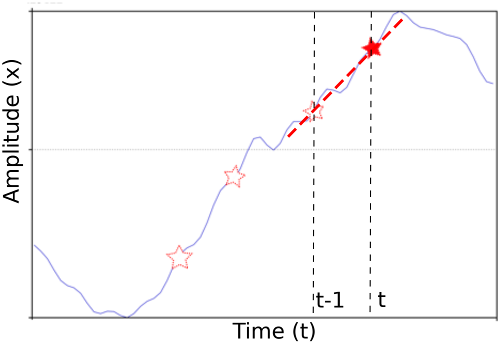
$$x(t) = a_1 x(t-1) + a_2 x(t-2)$$

Linear Predictive Coding



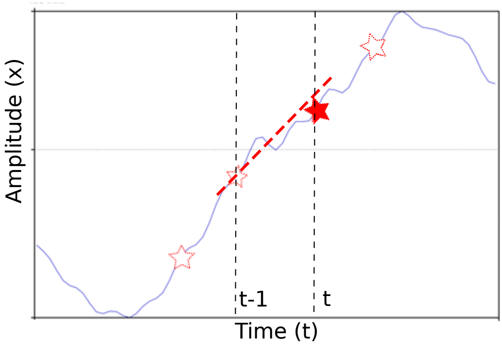
$$x(t) = a_1 x(t-1) + a_2 x(t-2) + a_3 x(t-3)$$

Linear Predictive Coding

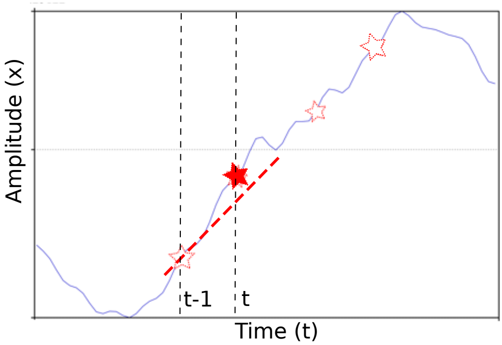


$$x(t) = as(t-1)$$

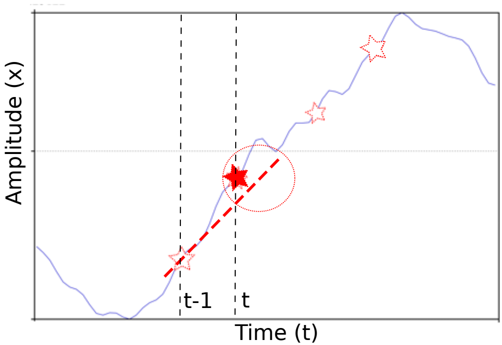
Linear Predictive Coding



Linear Predictive Coding

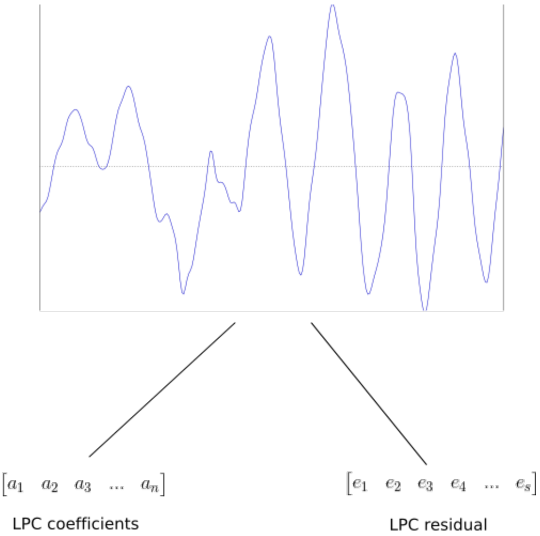


Linear Predictive Coding

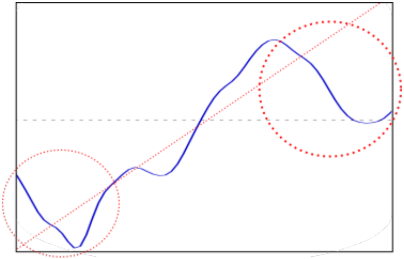


$$x(t) = \sum_{j=1}^{p} a_j s(t-j) + \mathbf{e(t)}$$

Linear Predictive Coding

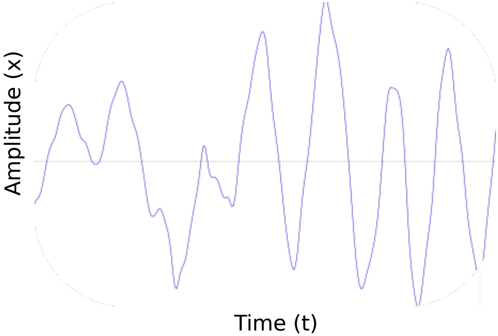


Linear Predictive Coding

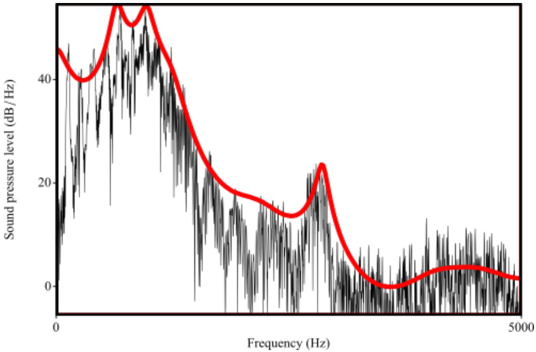


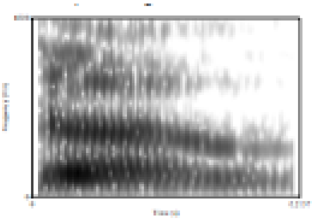
Largest errors tend to be where the signal changes the fastest, so the peaks of the residual approximate pitch periods.

Linear Predictive Coding

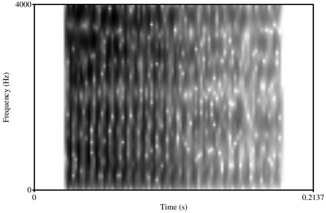


Linear Predictive Coding

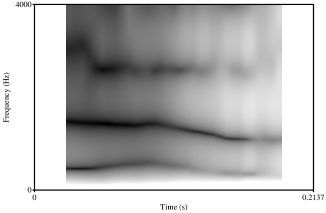




(a) Natural vowel token

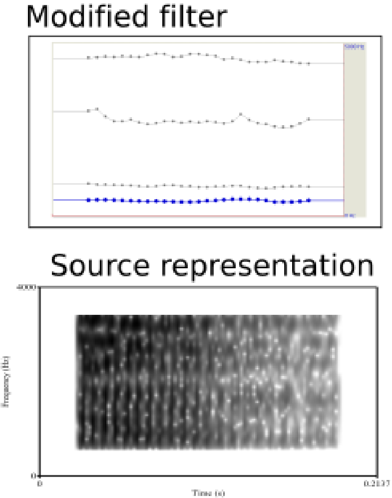


(b) Inverse-filtered source



(c) LPC spectrogram

Now we can excite a digital filter with our natural source representation



Inverse filtering using Praat

1) Resample to around 10000 Hz ($\sim 2 * nf * 1000$)

2) Perform LPC analysis with order $\sim 2 * nf + 2$

3) Inverse filter resampled original with LPC object

4) Estimate formants using 'To Formant...'

5) Manipulate formants using 'To FormantGrid'

6) Filter modified FormantGrid with inverse-filtered source

Inverse filtering using Praat

LPC inverse-filtering in Praat:

```
#Estimate the LPC filter for a selected sound
#First we need to resample
Resample: 10000, 50
To LPC (burg): 10, 0.025, 0.005, 50
```

Inverse filtering using Praat

```
#Take the inverse of this filter to get a representation of the source
selectObject: "Sound untitled_10000"
plusObject: "LPC untitled_10000"
Filter (inverse)
```

Inverse filtering using Praat

```
#Generate a formant object and add 400 Hz to F2
selectObject: "LPC untitled_10000"
To Formant
selectObject: "Formant untitled_10000"
Formula (frequencies): "if row = 2 then self + 400
    else self fi"
```

Inverse filtering using Praat

```
#Combine the source and filter representations to
    make a new vowel
selectObject: "LPC untitled_10000"
selectObject: "Sound untitled_10000"
plusObject: "Formant untitled_10000"
Filter
Play
```

Advanced inverse filtering methods

Some issues with inverse filtering:

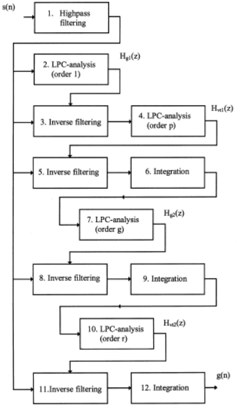
a) Difficult to get completely ‘white’ source representation

b) We lose the higher-frequency component of the sound

Advanced inverse filtering methods

a) Difficult to get completely ‘white’ source representation

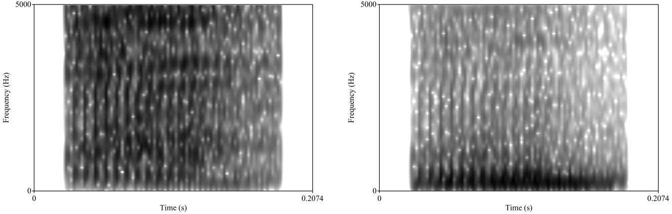
Solution: Following Alku (1992), filter iteratively, estimating spectral characteristics of glottal flow and vocal tract with different LPC orders



Advanced inverse filtering methods

a) Difficult to get completely ‘white’ source representation

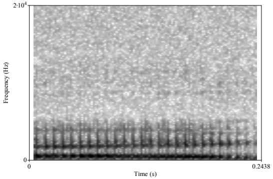
Solution: Following Alku (1992), filter iteratively, estimating spectral characteristics of glottal flow and vocal tract with different LPC orders



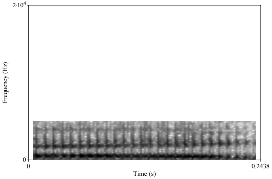
Advanced inverse filtering methods

b) We lose the higher-frequency component of the sound

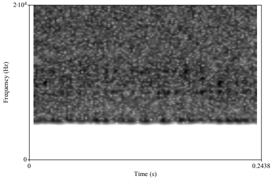
Solution: Restore the HF component of the original sound after synthesis



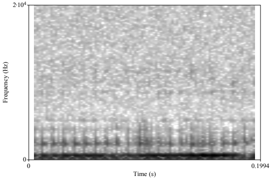
(a) Natural token @20Khz



(b) LF component



(c) HF component



(d) Semisynthetic token

Review

a) Try recording yourself producing a vowel (ctr+r). Select your vowel token and run 'LPC_cardinal_iaif.praat'. Compare the output with that of 'klatt_cardinal.praat

b) Experiment with the 'formant_manipulation.praat' script:

i. Create a 'ba-da' continuum

ii. Create an [i]-[u] continuum

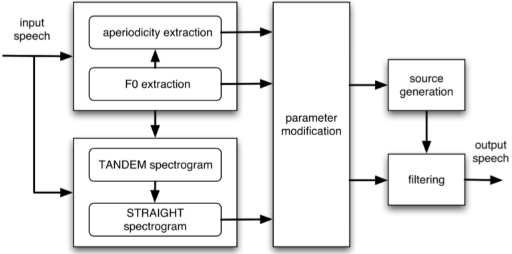
iii. Create a continuum between a diphthong and monophthong

The TANDEM-STRAIGHT vocoder

`http://www.wakayama-u.ac.jp/~kawahara/STRAIGHTadv/index_e.html`

Used in many commercial text-to-speech applications

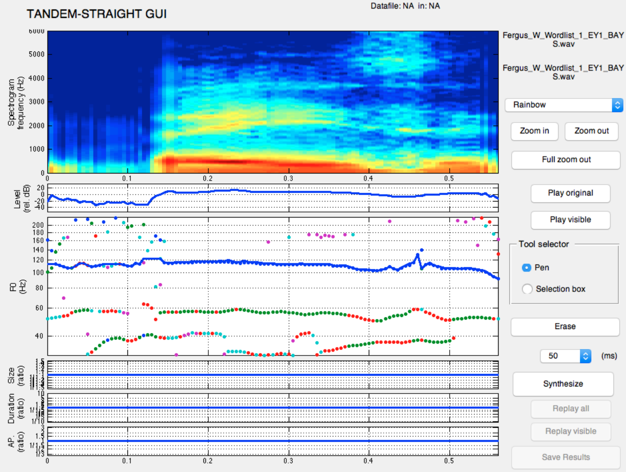
Uses advanced techniques to estimate the glottal flow and spectral envelope, as well as the aperiodic components of natural speech.



The TANDEM-STRAIGHT vocoder

Morphing GUI allows interpolation between two samples

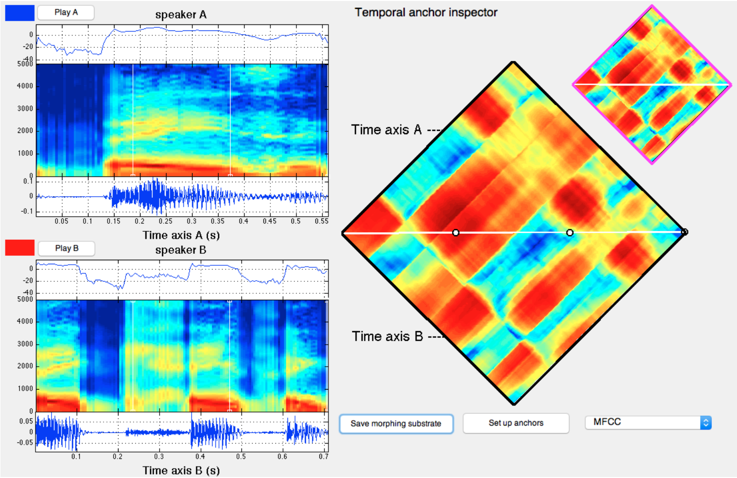
TANDEM-STRAIGHT GUI:



The TANDEM-STRAIGHT vocoder

Morphing GUI allows interpolation between two samples

TANDEM-STRAIGHT GUI:



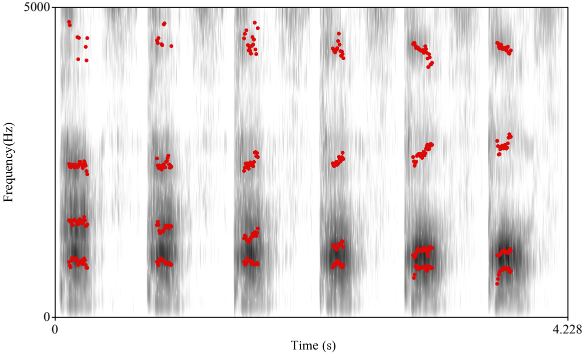
The TANDEM-STRAIGHT vocoder

TANDEM-STRAIGHT emotional morphing demo:

`http://www.wakayama-u.ac.jp/~kawahara/Miraikandemo/index-e.html`

Less useful for manipulating individual formants (but still possible)

Example TANDEM-STRAIGHT continuum



Download and install TANDEM-STRAIGHT

Full package:
`https://www.dropbox.com/s/jq1butk6shnnox1/LctrEdinburgh2012.zip`

Morphing GUI only:
`www.github.com/danielplawrence/synthesis_workshop`

MATLAB runtime:
`http://uk.mathworks.com/products/compiler/mcr/`

Use the instructions on the handout to generate a continuum

References

Alku, P. (1992). Glottal wave analysis with pitch synchronous iterative adaptive inverse filtering. *Speech communication*, 11(2-3), 109-118.

Fant, G. (1970). *Acoustic theory of speech production: with calculations based on X-ray studies of Russian articulations*. Mouton & Company.

Kawahara, H., Morise, M., Takahashi, T., Nisimura, R., Irino, T., & Banno, H. (2008). TANDEM-STRAIGHT: A temporally stable power spectral representation for periodic signals and applications to interference-free spectrum, F0, and aperiodicity estimation. In *IEEE International Conference on Acoustics, Speech and Signal Processing*. 2008. (pp. 3933-3936).