Synthesis of time-varying speech

- Articulators move relatively slowly, much more slowly than the sampling rates used to represent speech sounds.

- Vocal tract can be treated as stationary for chunks of time approximately 10 ms long.

- Speech can be synthesized by dividing the filtering into 10 ms frames that have fixed formant frequencies (F) and bandwidths (BW).
Technique

1. Calculate source for entire utterance

2. Filter through VT cascade
   - Outer Loop: Fn to F1
     - Inner Loop: frame 1 to frame N.
     - for each frame, take window of samps_per_frame from input
     - get F, BW values for that frame from table.
     - calculate filter coefficients and perform filtering
     - concatenate filter output with output of previous frames

3. Filter through radiation filter
samps_per_frame = srate * (frame_dur / 1000);
% syn1.m
% <Your Name>
% October 2013
% formant synthesizer
% version 1
% [signal, t] = syn1(srate,f0,frame_dur,nf,Ftable)
%
% input arguments:
% srate          sampling rate (in Hz)
% f0             fundamental frequency (in Hz)
% frame_dur      duration of each frame in milliseconds
% nf             number of formants
% Ftable        character string containing filename of F table
%                formant frequencies and bandwidths
%                first nrows/2 contain frequencies
%                second nrows/2 contain bandwidths
% returned arguments:
% signal      vector with synthesized waveform samples
% t           vector with time values (in ms) of each successive sample

% set parameters
FBW = get_FBW(Ftable);
nframes = size(FBW,2);
dur = nframes * (frame_dur / 1000); % total utterance duration in seconds
samps_per_frame = srate * (frame_dur / 1000);

% generate source
source = make_buzz(srate,f0,dur);

% filter thru of fixed glottal resonator
RG = 0; % RG is the frequency of the Glottal Resonator
BWG = 100; % BWG is the bandwidth of the Glottal Resonator
[b_glo,a_glo]=resonance(srate,RG,BWG);
source=filter(b_glo, a_glo , source);

% filter successive frames of source through VT cascade

% <YOUR CODE HERE>
% <THE OUTPUT SHOULD BE IN A VECTOR CALLED signal>

% set coefficients of high pass radiation filter and filter
% this calculates volume velocity at a distance from the the mouth,
b_rad = [1 -1];
a_rad = 1;
signal = filter(b_rad,a_rad,signal);
subplot (3,1,1)
% your code here
% plot the values of F1, F2, and F3 as function of frame No.
xlim ([1 max(frames)]);
xlabel ('Frame No.'

subplot (3,1,2)
% your code here
% plot the synthesized signal as a function of t in ms
% make sure you calculate the t vector based on the number of actual
% samples in signal
xlim ([1 max(t)]);
xlabel ('Time in milliseconds');

%plot the spectrogram of the synthesized signal as a function of t in ms in the bottom panel
make_spect2(signal', srate,6);
Controlling synthesis: FBW table

• To a first approximation speech can be synthesized as a sequence of steady-state epochs (acoustic segments) that show little change internally, and intervals of transition that connect these epochs.

• Transitions can be modeled as linear ramps between adjacent epochs.

• Control table specifies values at beginning and end of steady-state epochs. Intervening values can be calculated through linear interpolation between adjacent specified values.

<table>
<thead>
<tr>
<th>C</th>
<th>6</th>
<th>trans</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>we.txt</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

F 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
F1 1 400 22 400
F2 1 1000 6 1000 11 2500 22 2500
F3 1 2400 6 2400 11 3200 22 3200
F4 1 4000 22 4000
B1 1 70 6 70 11 50 22 50
B2 1 100 8 100 11 200 22 200
B3 1 150 8 150 11 400 22 400
B4 1 200 22 200
% get_FBW
% Louis Goldstein
% 22 October 2013
% interpolate across any blank columns
% Input arguments:
% filename      name of .xls file containing FBW table
% Output arguments:
% FBW           FBW table

function FBW = get_FBW(filename)
d = importdata(filename);
for i = 2:size(d.data,1)              % do each row separately
    rowdat = d.data(i,:);
    rowdat = rowdat(~isnan(rowdat));
    col = (rowdat(1:2:end-1));
    val = rowdat(2:2:end);
    nframes = col(end);
    FBW(i-1,:) = zeros(1,nframes);
    for j = 2:length(col)
        if col(j) - col(j-1) >0
            slope = (val(j) - val(j-1)) ./ (col(j) -
            col(j-1));
            for k = 0:col(j)-col(j-1)
                FBW(i-1, (col(j-1)+k)) = val((j-1)) +
                k*slope;
            end
        end
    end
end