Table of Contents

Skylar Tamke, Homework 4 - Phase Vocoder

The most I can expand my TIMIT wave is by 5 times its original playback speed. Alternatively I can shrink my TIMIT wave by 2.2 times before it runs too fast for me to understand.

```
% A phase phase vocoder is a tool used to compress or expand an audio
 wave
% so that changes in playback speed result in the same pitch output.
% Meaning that if you playback the audio at a faster rate you should
 be
% able to hear the same pitch as the original rather then what
 normally
% happens, which is the pitch increases and sounds like a tiny person.
clc
clear

warning('off','all')

%file that says 'Artificial intelligence is for real.'
filename = 'SX29.WAV';
phntable = readtable("SX29phoneam.txt");

%code provided on handout by Snider
fid = fopen(filename,'r');
status = fseek(fid, 1024, -1);
[wave,count] = fread(fid,inf,'int16');
fclose(fid);
Fs = 16000;

%to change the wave length into a nice number to divide into
count = length(wave)-3;

% Change this to change playback speed (how many time faster
% playbackspeed = .2          % slowest
playbackspeed = 2.2             % fastest

% Determining playback speed
SynthesisLen = floor(128*(1/playbackspeed));


WindowLen = 256;
AnalysisLen = 64;
Hopratio = SynthesisLen/(WindowLen/2);

numWindows = 316;
```

```
windowSize = 256;

%as of Ross's recommendation this only needs to be around 16
coefCount = 16;                                                 %how many
 coef per section, will probably go up for end result


Clength = windowSize;
tempcoefCount = coefCount;

windows = zeros(numWindows,windowSize);
h = zeros(numWindows,(Clength-coefCount),coefCount);
b = [];
       %if the number of coefficients change this does too, so b and h
 have equal m vs n lengths
atotal_ls = zeros(numWindows,coefCount);

windows(1,:) = wave(1:windowSize);                    %overlapping
 windows for each section
windowOverlap = floor(windowSize/2);                 %floor to keep
 safe,

% Create a vector of overlapping windows
% This part is important since if this is done improperly the output
 will
% sound choppy.  If the windows are not overlapped the output will
 only
% show the changes between the phoneams, which is where the choppyness
% comes from.  When the windows are overlapped and combined correctly
 later
% the windows will blend the changes together keeping some of the
% choppyness out.
for i = 1:numWindows
    if i-1 == 0
        windows(i,:) = wave(1:windowSize);
    else
        windows(i,:) = wave(((i-1)*windowOverlap+1):
(((i-1)*windowOverlap)+windowSize));
    end
end

fftwindows = [];
windowWeight = window(@hanning,windowSize);


yprevwin = zeros(1,WindowLen-SynthesisLen);
gain = 1/(WindowLen*sum(hanning(WindowLen,'periodic').^2)/
SynthesisLen);
unwrapdata = 2*pi*AnalysisLen*(0:WindowLen-1)'/WindowLen;
firsttime = true;

ysangle = zeros(numWindows,WindowLen);
yunwrap = zeros(numWindows,WindowLen);
yprevangle = zeros(numWindows,WindowLen);
```

```
yangle = zeros(numWindows,WindowLen);



for i = 1:numWindows
   fft_windows(i,:) = fft(windows(i,:));
   ymag(i,:) = abs(fft_windows(i,:));
   if i == 1
       yprevangle(i,:) = yprevangle(i,:);
   else
       yprevangle(i,:) = yangle(i,:);
   end
   yangle(i,:) = angle(fft_windows(i,:));
   yunwrap(i,:) = (yangle(i,:) - yprevangle(i,:)) - unwrapdata';
   yunwrap(i,:) = yunwrap(i,:) - round(yunwrap(i,:)/(2*pi))*2*pi;
   yunwrap(i,:) = (yunwrap(i,:) + unwrapdata') * Hopratio;

   if i == 1
       ysangle(i,:) = yangle(i,:);

   else
       ysangle(i,:) = ysangle(i,:) + yunwrap(i,:);
   end
   ys(i,:) = ymag(i,:) .* complex(cos(ysangle(i,:)),
 sin(ysangle(i,:)));
   ywin(i,:) = real(ifft(ys(i,:)));
end



winWeightSize = windowSize;
windowWeight = window(@hanning,windowSize);

outputlen = round(count*(SynthesisLen/windowOverlap))+1

output = zeros(outputlen,1);

for i = 1:numWindows-10
    if i == 1
        output(1:length(ywin(i,:))) = ywin(i,:)' .* windowWeight;
    else
        start = (i-1)*SynthesisLen;
        stop = start + windowSize-1;
        initial = output(start:stop) .* windowWeight;
        convolution = ywin(i,:)' .* windowWeight;
        out = initial + convolution;
        output(start:stop) = out;
    end
end
output(:) .*gain;


playbackspeed =
```

```
    2.2000
```

```
outputlen =

       18421
```

Plotting - commented out since this wasn't needed for deliverable

just a plot of the original wave and the resulting waveform after the process

figure(1) hold off plot(wave,'b') hold on plot(output,'r') hold off

legend("original wave","shrunken wave") title("Phase vocoder")

```
newFs = Fs*(1/(Hopratio));
```

Playback

sounds are played back at the new sample rate to show difference

```
soundsc(output,Fs)
% pause(5)
% soundsc(wave,newFs)
```

```
fileout1 = 'phase_vocoder_output_fastest.wav';
audiowrite(fileout1,output, Fs);
```

Published with MATLAB® R2018b