7. Matlab

7.1 Download

You can download the matlab code that implements the phase vocoder described in the previous sections.

7.2 Installation

In order to use this code, you need to unzip the following files in the folder of your choice:

- **pitchShift.m**
- **createFrames.m**
- **fusionFrames.m**

7.3 Execution

Here are the steps to shift the pitch of a wave file.

7.3.1 Launch Matlab

Well this one is kinda trivial ;)

7.3.2 Change the current directory

You need to set the current directory to the directory where you extracted the matlab files from the zip file above. You can do this with the following menu:

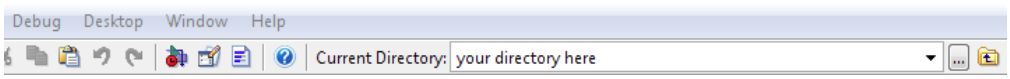

Figure 7.1: How to change the Matlab current directory

7.3.3 Load a wave file into a vector

The next step consists in loading the content of a wave file (*.wav) into Matlab. As far as I know, you cannot load directly from the mp3 format. If you want to use a mp3 file, you first need to convert it to the wave format and then load it with Matlab. To do so, you can type the following command in the Matlab command window.

```
[x Fs] = wavread('C:\MySounds\yourwavefile.wav');
```

The audio signal is loaded in the vector named x and the sampling rate is loaded in the variable Fs (this can be useful if you ignore the sampling rate used for recording the sound initially). If you have a mono signal, x will be a matrix with a single column. However, if you have a stereo signal, x will be a matrix with two columns. You can either pitch shift each left and right channel individually or average them together and use the result as the input vector. You can use the following commands to reorganise your signals:

```
leftChannel = x(:,1);
rightChannel = x(:,2);
averageBoth = (leftChannel + rightChannel) / 2;
```

7.3.4 Shift the pitch

You can then shift your signal by calling the pitchShift function. Here, you need to specify the vector which contains the signal to be shifted, the sampling rate, the frame size in samples and the hop size in samples.

We will assume that the samppling rate of the audio wave file was 48000 samples/sec, such that Fs = 48000. For this reason, as explained previously, we choose a frame size of 1024 samples for optimal results. Since we want an overlap of 75%, we set the hop size to 256. Finally, let's say we want to shift the pitch up by two steps. We want the resulting signal to be stored in a vector named y. We need to write the following command:

```
y = pitchShift(x,1024,256,2);
```

7.3.5 Listen to the result

That's it! The signal is now pitch shifted. But it would be nice to listen to it. In order to do so, use the following command. By the way, **always reduce the volume of your speaker when you perform some audio tests with Matlab**. If for some reason the operation went wrong and the power of the resulting signal is two high, you could blast your ears (happened to me a few times... then I learned my lesson).

```
wavplay(y,Fs);
```

7.4 Complete code

Here is the content of the Matlab files, which you can download in section 7.1.

pitchShift.m

```
%
% *****************************************************************
% * Authors: Laurier Demers, Francois Grondin, Arash Vakili       *
% *****************************************************************
% * Inputs:  inputVector     Vector (time-domain) to be processed *
% *          windowSize      Size of the window (ideally 1024 for a *
% *                          sampling rate at 44100 samples/sec   *
% *          hopSize         Size of the hop (ideally 256 for a window *
% *                          of 1024, in order to get 75% overlap *
```

```
% *          step            Number of steps to shift (should range       *
% *                          between -4 and 4 to preserve quality)        *
% * Outputs: outputVector    Result vector (time-domain)                  *
% ***************************************************************************
% * Description:                                                            *
% *                                                                         *
% * This function takes a vector of samples in the time-domain and shifts *
% * the pitch by the number of steps specified. Each step corresponds to  *
% * half a tone. A phase vocoder is used to time-stretch the signal and   *
% * then linear interpolation is performed to get the desired pitch shift *
% ***************************************************************************
% * DISCLAIMER:                                                             *
% *                                                                         *
% * Copyright and other intellectual property laws protect these          *
% * materials. Reproduction or retransmission of the materials, in whole  *
% * or in part, in any manner, without the prior consent of the copyright *
% * holders, is a violation of copyright law.                             *
% *                                                                         *
% * The authors are not responsible for any damages whatsoever, including *
% * any type of loss of information, interruption of business, personal   *
% * injury and/or any damage or consequential damage without any          *
% * limitation incurred before, during or after the use of this code.     *
% ***************************************************************************
%
function [outputVector] = pitchShift(inputVector, windowSize, hopSize, step)

%% Parameters

% Window size
winSize = windowSize;
% Space between windows
hop = hopSize;
% Pitch scaling factor
alpha = 2^(step/12);

% Intermediate constants
hopOut = round(alpha*hop);

% Hanning window for overlap-add
wn = hann(winSize*2+1);
wn = wn(2:2:end);

%% Source file

x = inputVector;

% Rotate if needed
if size(x,1) < size(x,2)

    x = transpose(x);

end

x = [zeros(hop*3,1) ; x];

%% Initialization

% Create a frame matrix for the current input
[y,numberFramesInput] = createFrames(x,hop,winSize);

% Create a frame matrix to receive processed frames
numberFramesOutput = numberFramesInput;
outputy = zeros(numberFramesOutput,winSize);

% Initialize cumulative phase
phaseCumulative = 0;

% Initialize previous frame phase
previousPhase = 0;

for index=1:numberFramesInput

%% Analysis

    % Get current frame to be processed
    currentFrame = y(index,:);

    % Window the frame
    currentFrameWindowed = currentFrame .* wn' / sqrt(((winSize/hop)/2));

    % Get the FFT
    currentFrameWindowedFFT = fft(currentFrameWindowed);

    % Get the magnitude
    magFrame = abs(currentFrameWindowedFFT);

    % Get the angle
    phaseFrame = angle(currentFrameWindowedFFT);

%% Processing

    % Get the phase difference
    deltaPhi = phaseFrame - previousPhase;
    previousPhase = phaseFrame;

    % Remove the expected phase difference
    deltaPhiPrime = deltaPhi - hop * 2*pi*(0:(winSize-1))/winSize;

    % Map to -pi/pi range
    deltaPhiPrimeMod = mod(deltaPhiPrime+pi, 2*pi) - pi;
```

```
    % Get the true frequency
    trueFreq = 2*pi*(0:(winSize-1))/winSize + deltaPhiPrimeMod/hop;

    % Get the final phase
    phaseCumulative = phaseCumulative + hopOut * trueFreq;

    % Remove the 60 Hz noise. This is not done for now but could be
    % achieved by setting some bins to zero.

%% Synthesis

    % Get the magnitude
    outputMag = magFrame;

    % Produce output frame
    outputFrame = real(ifft(outputMag .* exp(j*phaseCumulative)));

    % Save frame that has been processed
    outputy(index,:) = outputFrame .* wn' / sqrt(((winSize/hopOut)/2));

end

%% Finalize

% Overlap add in a vector
outputTimeStretched = fusionFrames(outputy,hopOut);

% Resample with linearinterpolation
outputTime = interp1((0:(length(outputTimeStretched)-1)),outputTimeStretched,(0:alpha:
(length(outputTimeStretched)-1)),'linear');

% Return the result
outputVector = outputTime;

return
```

createFrames.m

```
%
% ************************************************************************
% * Authors: Laurier Demers, Francois Grondin, Arash Vakili              *
% ************************************************************************
% * Inputs:  x              Vector                                       *
% *          hop            Number of samples between adjacent windows   *
% *          windowSize     Size of each window                          *
% * Outputs: vectorFrames   Resulting matrix made of all the frames      *
% *          numberSlices   Number of frames in the matrix               *
% ************************************************************************
% * Description:                                                         *
% *                                                                      *
% * This function splits a vector in overlapping frames and stores these *
% * frames into a matrix:                                                *
% *                                                                      *
% * |------------------Input vector----------------------|               *
% *                                                                      *
% * |------1------|                                                      *
% *      |------2------|                                                 *
% *          |------3------|                                            *
% *              |------4------|                                        *
% *                       ...                                          *
% *                                                                      *
% * Index            Frame                                               *
% *   1          |------1------|                                         *
% *   2          |------2------|                                         *
% *   3          |------3------|                                         *
% *   4          |------4------|                                         *
% *  ...              ...                                              *
% *                                                                      *
% ************************************************************************
% * DISCLAIMER:                                                          *
% *                                                                      *
% * Copyright and other intellectual property laws protect these         *
% * materials. Reproduction or retransmission of the materials, in whole *
% * or in part, in any manner, without the prior consent of the copyright*
% * holders, is a violation of copyright law.                            *
% *                                                                      *
% * The authors are not responsible for any damages whatsoever, including *
% * any type of loss of information, interruption of business, personal  *
% * injury and/or any damage or consequential damage without any         *
% * limitation incurred before, during or after the use of this code.    *
% ************************************************************************
%
function [vectorFrames,numberSlices] = createFrames(x,hop,windowSize)

% Find the max number of slices that can be obtained
numberSlices = floor((length(x)-windowSize)/hop);

% Truncate if needed to get only a integer number of hop
x = x(1:(numberSlices*hop+windowSize));

% Create a matrix with time slices
vectorFrames = zeros(floor(length(x)/hop),windowSize);

% Fill the matrix
for index = 1:numberSlices
```

```
    indexTimeStart = (index-1)*hop + 1;
    indexTimeEnd = (index-1)*hop + windowSize;

    vectorFrames(index,:) = x(indexTimeStart: indexTimeEnd);

end

return
```

fusionFrames.m

```
%
% ************************************************************************
% * Authors: Laurier Demers, Francois Grondin, Arash Vakili              *
% ************************************************************************
% * Inputs:  framesMatrix    Matrix made of all the frames               *
% *          hop             Number of samples between adjacent windows  *
% * Outputs: vectorTime      Resulting vector from overlapp add          *
% ************************************************************************
% * Description:                                                         *
% *                                                                      *
% * This function overlap adds the frames from the input matrix          *
% *                                                                      *
% * Index            Frame                                               *
% *   1          |------1------|                                         *
% *   2          |------2------|                                         *
% *   3          |------3------|                                         *
% *   4          |------4------|                                         *
% *  ...              ...                                               *
% *                                                                      *
% * |------1------|                                                      *
% *   + |------2------|                                                 *
% *       + |------3------|                                             *
% *           + |------4------|                                         *
% *               +      ...                                           *
% *                                                                      *
% * |-----------------Input vector---------------------|                *
% *                                                                      *
% ************************************************************************
% * DISCLAIMER:                                                          *
% *                                                                      *
% * Copyright and other intellectual property laws protect these         *
% * materials. Reproduction or retransmission of the materials, in whole *
% * or in part, in any manner, without the prior consent of the copyright *
% * holders, is a violation of copyright law.                            *
% *                                                                      *
% * The authors are not responsible for any damages whatsoever, including *
% * any type of loss of information, interruption of business, personal  *
% * injury and/or any damage or consequential damage without any         *
% * limitation incurred before, during or after the use of this code.    *
% ************************************************************************
%
function vectorTime = fusionFrames(framesMatrix, hop)

sizeMatrix = size(framesMatrix);

% Get the number of frames
numberFrames = sizeMatrix(1);

% Get the size of each frame
sizeFrames = sizeMatrix(2);

% Define an empty vector to receive result
vectorTime = zeros(numberFrames*hop-hop+sizeFrames,1);

timeIndex = 1;

% Loop for each frame and overlap-add
for index=1:numberFrames

    vectorTime(timeIndex:timeIndex+sizeFrames-1) =
vectorTime(timeIndex:timeIndex+sizeFrames-1) + framesMatrix(index,:)';

    timeIndex = timeIndex + hop;

end

return
```

Previous: Performances