Lab 3: Filter Banks, the STFT, and the Phase Vocoder

Due Tuesday 04/08/19

Overview:

- This assignment should be completed with your chosen lab partner(s).
- Each group must turn in a report composed using a word processor (e.g., Word, Pages, LaTeX, etc...) including a cover page with full names of all group members. The remaining pages should contain (in order) the answers and MATLAB scripts for the exercises. MATLAB figures can be pasted into the document or saved as PDF files. When working on the project, please follow the instructions and respond to each item listed. Your project grade is based on: (1) your MATLAB scripts, (2) your report (plots, explanations, etc. as required), and (3) your final results. For all labs, you must clearly write the problem number next to your solution and label the axes on all plots to get full credit. Submission can be done electronically in PDF format or on paper.
- Plagiarism is a very serious offense in Academia. Any figures in the paper not generated by you should labeled "Reproduced from [...]". Any portions of any simulation code (e.g., MATLAB, C, etc...) not written by you be clearly marked in your source files. The original source of any mathematical derivation or proof should be explicitly cited.

1 Overview

In this lab, we will explore filter banks, the STFT, and the phase vocoder.

2 Exercises

2.1 Warm-Up

The goal of this warm-up is to recall the symmetry exhibited by Fourier transforms of real signals.

(a) For $N = 256$, let the signal $x[n]$ be a standard Gaussian random variable for $n = 0, 1, \ldots, N-1$. In Matlab, one can write `x=randn(1,256)`. Consider the quantity $X[-k]^* - X[k]$? In Matlab, you can compute `X=fft(x)` and `conj(fliplr(X(2:end)))-X(2:end)`. What value does this quantity have and why?

(b) Now, let the DFT $Y[k]$ be a standard complex Gaussian random variable for $n = 0, 1, \ldots, N-1$. In Matlab, one can write `Y=(randn(1,256)+1i*randn(1,256))/sqrt(2)`. Consider the quantity $Z[k] = Y[k] + Y[-k]^*$ or `Z=Y+conj([Y(1) fliplr(Y(2:end))])` . What can you say about its inverse DFT $z[n]$?

2.2 Short-Time Fourier Analysis

The goal of this part of the lab to implement the short-time Fourier analysis algorithm using length-N blocks overlapped by $N - M$ samples. To simplify some expresions, we will assume that **N is even and that M divides N**. Write and test Matlab code to implement each of steps listed below.

(a) For a length-L input signal, zero-pad its length so that $L-(N-M)$ is a multiple of M. Extract the overlapped sections $x_m[n] = x[mM + n]$ from $x[n]$ and stack them into the matrix `X` of dimension $N \times m_{\max}$, where $m_{\max} = \lceil (L - N)/M \rceil + 1$ sections suffice for an input signal of length. Hint: Use help to investigate built-in function `buffer(x,N,N-M,'nodelay')`.

(b) Window each section using the window defined by

$$w[n] = \sqrt{\frac{2M}{N}} \sin\left(\frac{\pi}{N}\left(n + \frac{1}{2}\right)\right).$$

Let the resulting matrix be denoted by `Xw`. Hint: To apply the same window to every column of a matrix, investigate the built-in function `bsxfun(@times,X,w)`.

(c) Use the `fft` command to compute the column-wise DFT of the windowed matrix `Xw` to get $X_m[k] \ \forall \ m, k$. Since audio signals have real values, the resulting FFT has conjugate symmetry and therefore is redundant. Define the matrix `Xwf` to contain the first $N/2+1$ rows of the FFT of the windowed `X`, which are non-redundant. Note that, in the `fft` function, the second argument specifies the dimension of the matrix along which the FFT should be computed.

(d) Then, from `Xwf` extract the energy and phase into distinct matrices `E` and `P` based on (for $k \in \{0, 1, \ldots, N/2\}$)

$$E_m[k] = |X_m[k]|^2$$
$$P_m[k] = \angle X_m[k].$$

Exercise 1. For the audio file "singing8.wav", compute the STFT with $N = 256$ and $M = 64$. Use the command `image` to plot the `E` matrix using a separable scaling (e.g., $a \log(x) + b$) that exposes the harmonic structure in the signal.

2.3 Short-Time Fourier Synthesis

The goal of this part of the lab to implement the short-time Fourier synthesis algorithm, which inverts the previous STFT algorithm based on length-N blocks overlapped by $N - M$ samples. Write and test Matlab code for each step in the process outlined below.

(a) First, reconstruct an estimate

$$\hat{X}_m[k] = \sqrt{E_m[k]}e^{jP_m[k]}$$

of $X_m[k]$ for $k \in \{0, 1, \ldots, N/2\}$. Then, let `hXwf` denote the matrix with N rows whose m-th row is given by $\hat{X}_m[k]$ for $k \in \{0, 1, \ldots, N/2\}$ and $\hat{X}_m^*[N-k]$ for $k \in \{N/2+1, \ldots, N-1\}$. This regenerates the conjugate symmetric portion of the signal.

(b) Next, we reconstruct the signal

$$\hat{x}_m[n] = \frac{1}{N} \sum_{k=0}^{N-1} \hat{X}_m[k] e^{2\pi jkn/N}$$

by letting `hXwfi` equal the column-wise `ifft` of the matrix `hXwf`. Due to numerical roundoff, one may need to keep only the real part. Note: A good test here is to check if $\hat{x}_m[n]$ is essentially real (i.e., max absolute value of imaginary is less than 10^{-8}).

(c) Next, we combine the different observations of $x[n]$ together to form the signal reconstruction

$$\hat{x}[n] = \sum_{m=\lfloor n/M \rfloor - N/M+1}^{\lfloor n/M \rfloor} w[n - mM]\hat{x}_m[n - mM].$$

This can be computed by starting with a zero vector of length $m_{\max}M + (N - M)$ and then using a for-loop to window and accumulate each column of `hXwfi` into the appropriate place in that vector. This corresponds to starting with $\hat{x}[n] = 0$ and iterating, for $m = 0$ to $m = m_{\max}$, the update

$$\hat{x}[n + mM] = \hat{x}[n + mM] + w[n]\hat{x}_m[n].$$

Exercise 2. For the audio file "notawhisper.wav", use your code from the previous section to compute the STFT with $N = 512$ and $M = 256$. Then, use the command `P=2*pi*rand(size(P))` to replace the true phase by a uniform random phase over $[-\pi, \pi]$ (note: `rand` generates uniform random variables). Now, reconstruct the modified signal using the STFT synthesis algorithm. How does the synthetic whisper signal sound in comparison to the natural "whisper.wav"? Use `pwelch(x,ones(1,256))` to compare the power spectral density of the two signals. What type of post-processing might make the synthetic whisper sound more realistic.

2.4 Phase Vocoder

The goal of this section is to build a phase vocoder using STFT analysis, resampling, and STFT synthesis. The result will be a function MATLAB function `phaseVocoder.m` whose arguments are the input signal $x[n]$, the time-scaling factor α, the window length N, and the shift length M. The function will return the time-scale modified signal $\hat{x}[n]$ whose length is roughly $\frac{1}{\alpha}$ times the length of $x[n]$. The following is the overall flow for one kind of implementation of the phase vocoder.

Note: Make sure that you have tested your STFT analysis and synthesis algorithms before attempting this part. Also, it suggested to start by focusing on $\alpha = 1/2$, $N = 1024$, and $M = N/2$. Once you have successfully implemented this case, it will be easier to generalize your function to arbitrary (N, M, α).

(a) Use your STFT analysis function implement the filter bank and compute `E` and `P`.

(b) Now construct a new matrix `E1` whose rows are the linearly interpolated energies for factor α. In particular, the ℓ-th column (for $\ell \in \{1, \ldots, \lfloor (m_{\max} - 2)/\alpha \rfloor - \lceil \frac{1}{\alpha} \rceil\}$) corresponds to time $t = 1 + \alpha(\ell - 1)$ (relative to blocks in the input signal) using the formula

$$\tilde{E}_t[k] = (\lceil t \rceil - t) E_{\lfloor t \rfloor}[k] + (1 - \lceil t \rceil + t) E_{\lceil t \rceil}[k].$$

Note that $t \in [1, m_{\max} - 2 - \alpha]$ implies $\lfloor t \rfloor \geq 0$ and $\lceil t \rceil \leq m_{\max} - 1$. The built-in `interp1` function can also be useful for linear interpolation.

(c) Now, construct the m-th column of the delta-phase matrix `D` using

$$D_m[k] = (P_m[k] - P_{m-1}[k] - 2\pi k M/N) \bmod 2\pi$$

for $m \in \{1, \ldots, m_{\max} - 1\}$, using $z \bmod 2\pi$ =`mod(z+pi)-pi`. It can help to think of $D_m[k]$ as an estimate of the derivative of phase evaluated at time $t = m - \frac{1}{2}$ (relative to the input signal). The built-in function `diff` may also be useful here.

(d) Next, construct the ℓ-th column of the interpolated delta-phase matrix `D1` for $\ell \in \{1, \ldots, \lfloor (m_{\max} - 2)/\alpha \rfloor - \lceil \frac{1}{\alpha} \rceil - 1\}$ by linear interpolation of the delta-phase matrix to $t_\ell = 1 + \alpha(\ell - \frac{1}{2})$ using the formula

$$\tilde{D}_t[k] = \left(\left\lceil t + \frac{1}{2} \right\rceil - t - \frac{1}{2} \right) D_{\lfloor t + \frac{1}{2} \rfloor}[k] + \left(1 - \left\lceil t + \frac{1}{2} \right\rceil + t + \frac{1}{2} \right) D_{\lceil t + \frac{1}{2} \rceil}[k].$$

Note that $t_\ell \in [1 + \frac{\alpha}{2}, m_{\max} - 2 - \frac{3\alpha}{2}]$ implies $\lfloor t_\ell + \frac{1}{2} \rfloor \geq 0$ and $\lceil t_\ell + \frac{1}{2} \rceil \leq m_{\max} - 1$. Finally, for $\ell \in \{1, \ldots, \lfloor (m_{\max} - 2)/\alpha \rfloor - 1\}$, the ℓ-th column of the accumulated-phase matrix `P1` is constructed by accumulating the phase for each interpolated block. For $\ell = 1$, we choose the first column $\tilde{P}_1[k] = P_1[k]$ and, for $\ell \geq 2$, we use

$$\tilde{P}_\ell[k] = \tilde{P}_{\ell-1}[k] + \tilde{D}_{t_{\ell-1}}[k] + 2\pi k M/N.$$

(e) Use your STFT synthesis function to reconstruct the signal from `E1` and `P1` (i.e., $\tilde{E}$ and $\tilde{P}$).

Exercise 3. Test your function on the file `singing44.wav` by setting $\alpha = 1/2$, $N = 1024$, and $M = N/2$. Once you have successfully implemented this case, try and generalize your function for arbitrary (N, M, α). Include the resulting sound file (in wav format) with your submission.

Exercise 4. Repeat the previous exercise for the same sound file with $\alpha = 17/18$, $N = 1024$, and $M = 128$. The result $\hat{x}[n]$ must is the sound slowed down to 18/17 times its length with the pitch unchanged. Now, resample this signal by a suitable factor using band-limited interpolation so that the output signal $y[n]$ (almost) preserves the time-scale of the original sound file. Use the `resample(P,Q,100)` built-in function to do this and briefly explain the steps that the function executes to achieve resampling. This should increase the pitch of the original signal by a factor of $2^{1/12} \approx 18/17$ while preserving the overall time-scale. How does the resulting clip sound? Include the resulting sound (in wav format) file with your submission.