Final Project

Instructions

Dear Students,

As part of the assigned work for this course, you are required to use Matlab/Octave, or any other programing tool, to perform tasks related to signal processing. Please read the following instructions carefully before starting.

- **Due date:** 31 May 2020 (11:00 PM)
- **Groups:** You should work in groups of 3, 4 or 5 students.
- **Attachment:** Attached to this announcement you will find
    1. Two audio files and one image file
    2. A word document template for the project final report.
- **Deliverable:** You should deliver through G classroom (code: sgbl6uh) one uncompressed pdf file containing:
    1. All the required results and answers to questions.
    2. All the required figures. (Label your figures properly.)
    3. All the codes.
    4. The role of each team member in the project.
- **Important Note:** Neither the instructors nor the TA will guide you through projects or provide additional help.
- **Grading Criteria:** Points are given to Source Code, results and answers, figures and the report submitted according to the given template.
- **Plagiarism:** Students must not copy any material from any reference (without proper citation) or any another group's project. Plagiarism check shall be carried and the project will be considered un-valid (fail) in case of plagiarism.

Project Tasks:

Task One: Echo generation and removal

In this task, you are given a speech signal and you are required to add an echo effect to it. Then, you will remove the echo using frequency domain techniques.

a) Read the file 'audio1.wav'.

b) The required echo effect consists of the original signal and 3 delayed and shifted versions:

$$y(t) = x(t) + 0.9\,x(t - 0.25) + 0.8\,x(t - 0.5) + 0.7\,x(t - 0.75)$$

Note that the delay is in seconds, however, you will work with the discrete-time signal $x[n]$ so you will need to adjust the value of delay in the equation of $y[n]$.

c) Find and plot the impulse response of the echo generation system $h[n]$.

d) Obtain the output $y[n]$ of the echo system using convolution.

e) Now, we will remove the echo using frequency domain techniques.

Recall that the spectrum of a DT signal (the DTFT) is a continuous spectrum, periodic with a period of 2π. Therefore, it is not suitable for numerical computations.

Instead, we will use the **Discrete Fourier Transform (DFT)**, which is a sampled version of the DTFT.

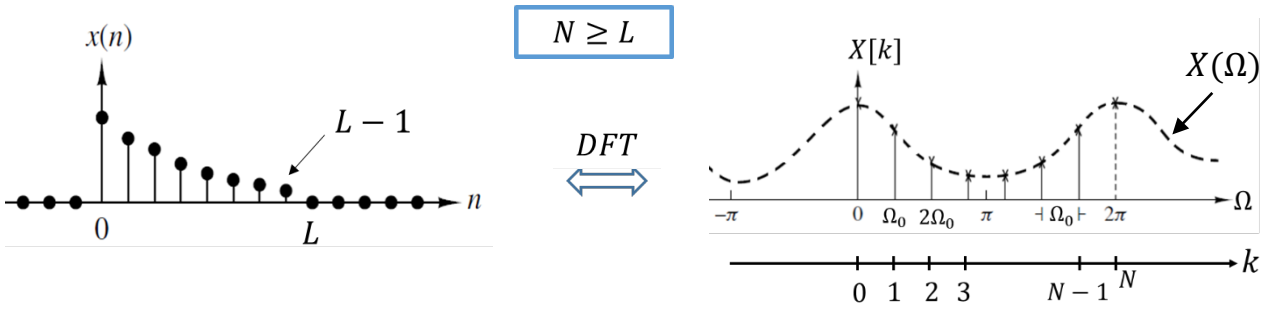

Fig. 1 DFT

DFT represents a time-domain signal of length L with N samples of its DTFT spectrum (from 0 to $N - 1$) in the range of 0 to 2π, as shown in Fig. 1. Note that N must be greater than or equal to L. This is called N-point DFT.

We say that $X[k]$ is the DFT of $x[n]$ and $x[n]$ is the inverse DFT (IDFT) of $X[k]$. These two operations are implemented in Matlab using 'fft' and 'ifft' commands, respectively. FFT is an efficient algorithm for computing the DFT.

Now, given $h[n]$ and $y[n]$, you are required to remove the echo effect and obtain $x[n]$.

We will perform this operation in the frequency domain. However, note that y[n] has length $L_y = L_x + L_h - 1$, therefore, $X[k], H[k]$ and $Y[k]$ need to be at least of length $N = L_y$.

Task Two: Audio steganography

Steganography refers to the practice of hiding a message within another message. In this task, you are given two audio files (audio1 and audio2) and you are required to hide audio1 in the audio2 file, such that a normal listener will not notice any change in audio2. Then, we will be able to extract audio1 separately.

a) Read two audio files (audio1.wav and audio2.wav)
b) Hide audio1 in audio2 as follows

$$x[n] = x_2[n] + A.x_1[n].\cos(\Omega n)$$

where, A is a suitable attenuation such that the second signal is inaudible, and Ω is a suitable frequency such that the spectra of $x_1[n]$ and $x_2[n]$ do not overlap. Recall that multiplying by cosine shifts the spectrum of the signal. Now, the two signals can be separated in the frequency domain.

c) Plot the magnitude spectrum of $x[n]$ using plot(abs(fft(X))). You may need to use log10(abs(fft(X))) to see very small values.
   Note that fft gives samples of the spectrum corresponding to the range from (0 to 2π). π corresponds to $F_s/2$ (maximum frequency in signal) and 2π corresponds to F_s, where F_s is the sampling rate in Hz.
d) Now, it is required to restore $x_1[n]$. This is done by multiplication with the same sinusoidal signal:

$$y[n] = x[n].\cos(\Omega n)$$

Then, $y[n]$ is filtered to retain the original signal $x_1[n]$. Filtering can be performed in the frequency domain. You may simply multiply the range of $Y[k]$ that you need to filter out by zeros.

Hint: You may need to increase the sampling rate of the signals before you start this task to satisfy the sampling theorem, since the spectrum of $x_1[n]$ will be shifted to a higher frequency band.

Task Three: Image compression

In this task, you will perform a simple image compression algorithm. You will read an input image and process each of its color components (red, green and blue) in blocks of 8×8 pixels. Each block will be converted into frequency domain using 2D Discrete Cosine Transform (2D DCT) and then only few coefficients are retained, while the rest will be ignored.

The 1D Fourier analysis represents the signal as a weighted sum of sinusoidals or complex exponentials with different frequencies. Similarly, the 2D DCT represents the 2D signal (e.g. an image) as a weighted sum of images with different spatial frequencies as shown in Fig. 2. Spatial frequencies refer to the rate of variation of pixel values w.r.t. space coordinates.

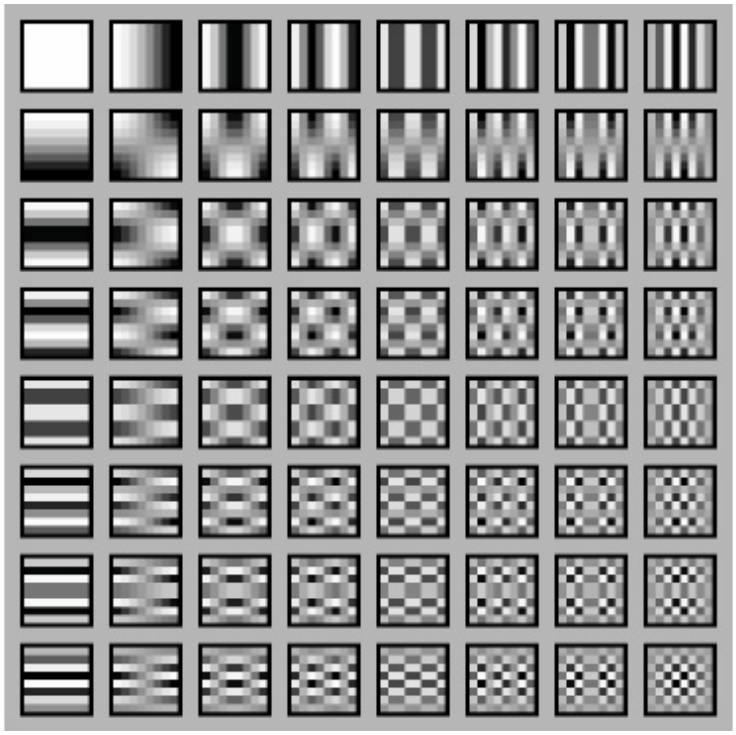

Fig. 2 DCT basis images

The top left corner is DC. Images towards the top left have low horizontal and vertical spatial frequencies. Images towards the right have higher horizontal spatial frequency. Images towards the bottom have higher vertical spatial frequency. The bottom right image has the maximum horizontal and vertical spatial frequencies.

On the one hand, most of the information of the image is concentrated in the lower spatial frequencies (the top left coefficients have significantly higher values). On the other hand, the human eye is more sensitive to lower frequencies and can't see extremely fine details. Therefore, ignoring higher spatial frequencies doesn't affect much how the eye perceives the image.

a) Read the image file 'image1.bmp'. Extract and display each of its three color components. Repeat the following steps for m=1,2,3,4.
b) To compress the image, process each color component in blocks of 8×8 pixels. Obtain 2D DCT of each block. It will have the same dimensions as the input block. Retain only the top left square of the 2D DCT coefficients of size $m \times m$. That is, if the DCT coefficients are X[1:8,1:8], retain only the top left $m \times m$ coefficients, $X[1:m, 1:m]$, assuming that the top left coefficient is $X[1,1]$. The rest of coefficients are ignored.
c) Compare the size of the original and compressed images.
d) Decompress the image by applying inverse 2D DCT to each block. Display the image.

e) The quality of the decompressed image is measured using the Peak Signal-to-Noise Ratio (PSNR), which is defined by

$$PSNR = 10 \log_{10} \frac{peak^2}{MSE}$$

where $peak$ is the peak value for the pixels according to the image datatype (e.g. for uint8 image it is 255). Mean square error (MSE) between the original image and the decompressed image is obtained by subtracting the corresponding pixel values of two images and obtaining the average of the square of all the differences. Obtain the PSNR for each value of m.
f) What are the advantages of using DCT instead of DFT for compression?