CS3570 Introduction to Multimedia Technology

Homework #2

Due: 11:59pm, 4/17/2023

1. DCT image compression (50%)

Transform the two images ***"Barbara.jpg", "cat.jpg"*** from spatial domain to frequency domain with DCT for compression and reconstruct the compressed image using inverse DCT with reduced numbers and bits of DCT coefficients.

DCT compression process:

- Divide the image into blocks of 8 × 8 pixels and apply 2D DCT for each block.
- For each block, only keep the lower-frequency (i.e. upper-left n-by-n) coefficients in the 2D DCT domain by setting the remaining coefficients to zero.
- Apply uniform quantization for DC and AC coefficients with suitable quantization tables and save each coefficient with m bits.
- Reconstruct the image by taking inverse 2D DCT with the modified DCT coefficients for each block.

(a) Implement the simplified DCT compression process above for n = 2, 4 and m = 4,8 respectively, and then apply it to the attached image.

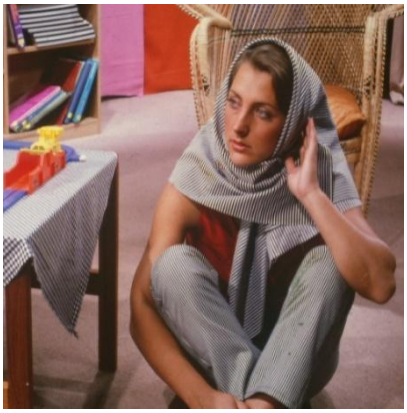
1. Show the reconstructed images for these four different cases. **[2*4 images]**
2. Compute the compression ratios and the PSNR values of these four reconstructed images and discuss the best rate-distortion choice.

(b) Use the same process in (a) with image transformed to YCbCr color space with 4:2:0 chrominance subsampling.

1. Show the reconstructed images in RGB space. **[2*4 images]**
2. Compute the compression ratios and the PSNR values of the four reconstructed images and discuss the best rate-distortion choice at the report.

(c) Report:

Compare the differences between the results with DCT compression performed in two color spaces in (a) and (b) and the results of the two given images.



2. Create your own FIR filters to filter audio signal (40%)

"***HW2_Mix.wav"*** is a mix of 3 songs. Based on the ideal impulse responses given in *slide #72* and the windowing functions in *slide #75*, you need to design and apply different FIR filters in time domain to separate the three audio signals from the given audio file. You can refer to the algorithm on *slide #76*. Next, you are asked to reduce the sampling rates of signals and compare them. Finally, the output audio signals are too simple so you should apply one-fold echo and multiple-fold echo (*slide #69*) to produce wonderful music. Please following the steps given below:

1. Transform the input signal into frequency domain.
2. Implement 3 different FIR filters to separate the three audio signals with Blackmann window function (You have to pick the appropriate window size and cut-off frequency). Please Implement 1-D convolution on the input signal of the given audio with your filters by yourself. (You are not allowed to use **scipy.signal.convolve or other python package function**).
3. Reduce the sampling rates of the three separated songs to 2000Hz.
4. Apply one-fold echo and multiple-fold echo on the audio signal that pass through the low-pass filter from step1. (Please use the audio files before reducing sampling rates)

For question2 you need to turn:

(a) Image results:

1. The spectrum of the input signal.
2. The spectrums of the output signals. (Before echo.) [3 images]
3. The spectrums of the filters. [3 images]
4. The shapes of the filters (time domain) [3 images]

(b) Audio results:

1. Store the three filtered audio files before reducing the sampling rates and name "Filter1Name_[para1]_[para2].wav". "Filter2Name_[para1]_[para2].wav". "Filter3Name_[para1]_[para2].wav".
2. Store the filtered audio files after reducing the sampling rates and name "Filter1Name_[para1]_[para2]_2kHZ.wav". "Filter2Name_[para1]_[para2] _2kHZ.wav". "Filter3Name_[para1]_[para2] _2kHZ.wav".
3. Store the echo audio files and name "Echo_one.wav" and "Echo_multiple.wav".

(c) Report:

1. Discuss how you determine the filters.
2. How you implement the filter and convolutions to separate the mixed song and one/multiple fold echo?
3. Compare spectrum and shape of the filters.
4. Briefly compare the difference between signals before and after reducing the sampling rates.

Reminder

- You should not use any function which can generate the result directly in each step.
- Your code should display and output your results so that we can judge if your code works correctly.
- You should provide a README file about your execution instructions.
- Please compress your code, input images, result images, report and README in a zip file named HW2_{Student-ID}.zip and upload it to eeclass.
- If you encounter any problem, please post your problems/questions on eeclass.
- Please follow the file structure below:

```
\---HW2_123456789
    |   README.md
    |   report.pdf
    |
    +---Q1
    |   |   1.py
    |   |   cat.jpg
    |   |   barbara.jpg
    |   |
    |   \---otuput
    |           bar_n2m2_a.png
    |           bar_n2m2_b.png
    |           bar_n2m4_a.png
    |           bar_n2m4_b.png
    |           bar_n4m2_a.png
    |           bar_n4m2_b.png
    |           bar_n4m4_a.png
    |           bar_n4m4_b.png
    |           cat_n2m2_a.png
    |           cat_n2m2_b.png
    |           cat_n2m4_a.png
    |           cat_n2m4_b.png
    |           cat_n4m2_a.png
    |           cat_n4m2_b.png
    |           cat_n4m4_a.png
    |           cat_n4m4_b.png
    |
    \---Q2
        |   2.py
        |   HW2_Mix.wav
        |
        \---output
                Filter1Name_[para1]_[para2].wav
                Filter1Name_[para1]_[para2]_2kHZ.wav
                Filter1Name_shape.png
                Filter1Name_spectrum.png
                Filter2Name_[para1]_[para2].wav
                Filter2Name_[para1]_[para2]_2kHZ.wav
                Filter2Name_shape.png
                Filter2Name__spectrum.png
                Filter3Name_[para1]_[para2].wav
                Filter3Name_[para1]_[para2]_2kHZ.wav
                Filter3Name_shape.png
                Filter3Name__spectrum.png
                input.png
                output1.png
                output2.png
                output3.png
```