20161159

Assignment 1 - Report
DSAA

Problem 1

Resizing an image increases the number of pixels by a certain factor. To make it appear as if the original image has been enlarged, these pixels have to take some value in relation to the original image pixel values otherwise they would appear black. We decide the values these pixels take using different types of interpolations on a matrix with the original pixels spread along the extremes. I implemented nearest neighbour interpolation by choosing the nearest original pixel value for every other pixel. I implemented bilinear interpolation by taking the weighted mean of two original pixels on the same line, to determine the values of all the pixels in between.

Comparison between images from both the methods : The images looks more continuous when resized using bilinear interpolation technique than when done with nearest neighbour interpolation. The reason behind that being that since in bilinear interpolation, the unknown pixel values are calculated by taking weighted average of the 4 surrounding pixels, it appears to be a smooth transition and looks more continuous. On the other hand in the nearest neighbour interpolation, the image looks distorted because any intermediate pixel replicates its nearest neighbour.

Better implementation could be to estimate a pixel value, using more than 4 values of pixels, perhaps with 16 surrounding pixels. This algorithm is called bicubic interpolation technique.

Problem 2

The image turns white at all the transitions and black in all the solid areas. This is because M is an edge-detection filter, called Sobel filter. Since the top row is negatives and the bottom row is positives of equal magnitude, in the solid areas they cancel each other and

produce black pixels. However, at transitions the top row pixels and bottom row pixels won't have equal magnitude and hence, the convolution produces a pixel of non-zero value. When the convolution is performed with M^T, the transitions are detected in the vertical direction instead of horizontal.

Problem 3

The depth of the output filter can be known in a very straightforward manner. The depth of the output filter is nothing but the number of filters which in our case is denoted by N. It can be proven in a very simple manner, if there're are N filters, each one is convolved with the image and results in a channel of the output image. Hence, the depth of the output signal is N. Now the width of the output matrix is nothing but the number of times you move the filter on your image. The effective width of the image after padding is W+2Z. Hence given the step size S, the number of times you slide it or the width of output signal is (W+2Z-F)/S+1. Similarly the height of the image is (H+2Z-F)/S+1.

Everytime you place the filter on a submatrix of the image, you're multiplying F*F elements, and you do this ((H+2Z-F)/S+1)*((W+2Z-F)/S+1) times, hence the number of multiplications that are performed are F*F*((H+2Z-F)/S+1)*((W+2Z-F)/S+1).

For the number of additions, since you add F*F, 2 at a time, you perform F*F-1 additions. You further do this ((H+2Z-F)/S+1)*((W+2Z-F)/S+1) times, hence total number of additions are (F*F-1)*((H+2Z-F)/S+1)*((W+2Z-F)/S+1).

Problem 4

The q4_generateAudioFile.m module records in the microphone for 5 seconds, and then writes the audio data to myRecording.wav. In q4.m, the myRecording.wav file is read using audioread() and stored in a local variable. I wrote the function my_resample(data,A,B) which takes the audio data as well as parameters A and B which signify to change the sample rate to sample_rate * A/B. I implemented it by iterating through the audio data with steps of B/A and sampling the data into the array which is finally returned by the function. I observed a gradual decrease in the quality of the audio upon decreasing the sampling rate. This result is due to the loss in data when we subsample. For part (c), I observed my recording

appearing to be in various different environments upon convolving with different impulse responses.

Problem 5

I created the matrix [1 1 1; 0 0 0; -1 -1 -1] for the filter. 'Input.png' image has two regions of different color. Since the top row and bottom row have same magnitude and opposite sign, the values cancel out in the solid areas and produce black pixels. At the interface, the top pixels and bottom pixels have different values and hence, the convolution produces non-zero pixels. On convolving blur.jpg with this filter, you can clearly see the transitions along the horizontal direction. Similarly, for the transpose you can see the vertical transitions. Total edge detection is obtained upon adding the two images generated.

Problem 6

I wrote a find(A,B) function which returns the frame of pixels in A which most closely matches B. My algorithm iterates through all frames of size(B) in A, and calculates the correlation coefficient at each position. The correlation coefficient indicates the degree of similarity. The function returns the pixel coordinates of the frame with highest coefficient of correlation. After that, the drawRectangle() method is invoked which draws a red rectangle around the frame.

Problem 7

Using the flip-and-shift method to convolve the input signal with a variable filter, I matched the result with the given output and obtained a system of linear equations that describe the variable filter. I used the linsolve() built-in function to solve it. As a result, I obtained the filter 1D vector.