MARMARA UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

CSE 348

HOMEWORK #2

ZAFER EMRE OCAK - 150113075

OZAN GÜLHAN-150114013

Question #1:

This was the original image we're asked to use.

This is the effect of "Laplacian Filter" with *zero padding*. It basically highlights the edges on the image.

This is the effect of “Laplacian Filter” with *border replication* technique. It basically highlights the edges on the image.

This is the effect of “Gaussian Filter” with *zero padding*. It blurs the original image a little.

This is the effect of “Gaussian Filter” with *border replication* technique. It blurs the original image a little.

This is the effect of “Sobel Filter” with *zero padding*. It highlights the edges thicker than Laplace Filter.

This is the effect of “Sobel Filter” with *border replication* technique. It highlights the edges thicker than Laplace Filter.

This is the effect of “Motion Filter” with *zero padding*. It blurs the image more than Gaussian Filter and it also gives the illusion as the image is moving.

This is the effect of "Motion Filter" with *border replication* technique. It blurs the image more than Gaussian Filter and it also gives the illusion as the image is moving.

Question #2:

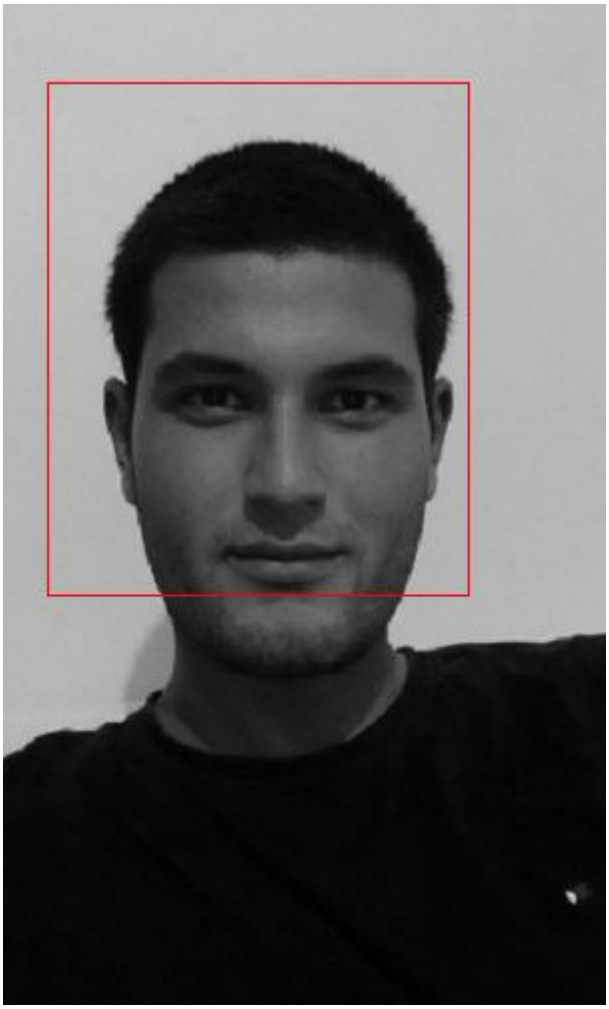
In this question, firstly, we successfully obtained the average template image.

This image is achieved by summing up 3 different face images which is [*256x210*] in size.

Later on, we managed to implement the template matching algorithm as it has been given in the documentation. The “test” and “template” images are given in the main function and function **detect** divides the test image into chunks with a size of the template image and iterates over each submatrice.

calcDif function calculates the amount of correlation for each sub-test image with the template image and returns the result. All results are appended into *coords* list and in **detect** function, we get maximum correlated indices and pass it to **drawFace** function. The latter draws the red rectangle with size given as a parameter and program terminates.

However, we have encountered a problem which is the total required time for output results. With a test image of size [500x301], we almost waited for one hour for one test image. Due to the restricted remaining time, we couldn’t test more than 1 image.The output result is:



We can say that the performance of the algorithm was average depending on the test images. If our program can get enough template images, the result is going to be more accurate.

Question #3:

We noticed, as we trimmed more and more frequency range at this part, we lost more and more data but since the trimmed parts were away from the center of our signal, we managed to understand it. The reason is that the most meaningful information was within the center area of our frequency range.

We trimmed our transformed discrete signal from 1000 lower, 1000 upper to 415000 lower and 415000 upper signals by increasing total trim range 10000 each stride. After 415000, the voice was barely understandable and we decided on leaving it on that range after all.

We have benefited from "*scipy.fftpack*" library to take Fourier and inverse Fourier transform of our signal.

Lastly, we plotted the signal in time and frequency domain to see how it looks...