**CSE 348**

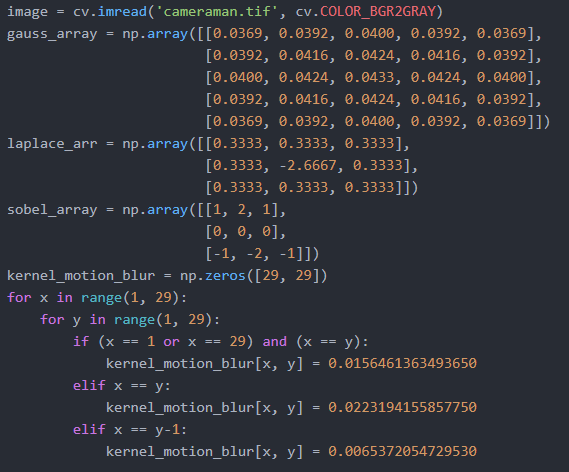
**HOMEWORK – 2**

**MUHAMMET ŞERAMET – 150115069**

**TOLGA SAVAŞ – 150115004**

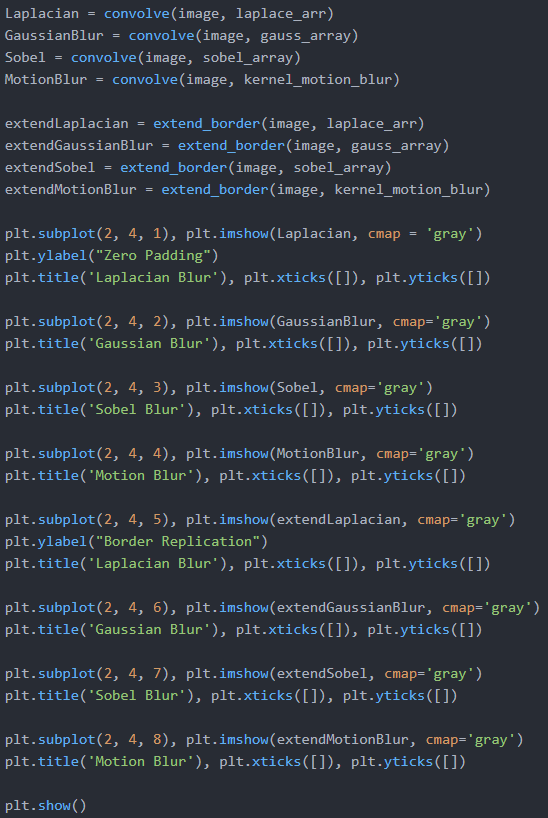
**1)** In this question we were asked to apply several kernels (Gaussian, Laplace, Motion and Sobel) on a given photo (cameraman.tif). To do so we will use open libraries like opencv and numpy.

To start with, we need to load our image and create our kernels. Kernels values are taken from Matlab with the help of fspecial function.

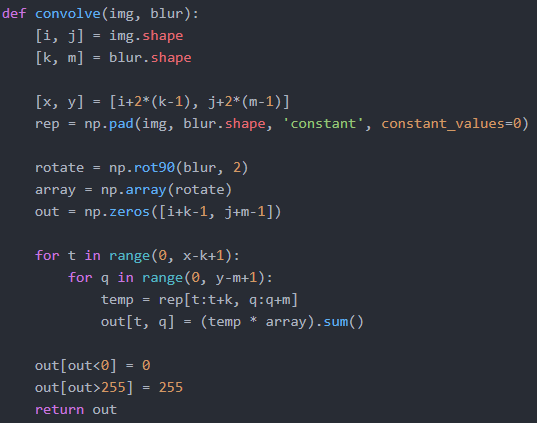


After creation we will convolve this kernels via our function called convolve and extend\_border.

In the main the code will look like this:



We are just calling the functions with original images and kernel parameter then plot the results. In convolve function:

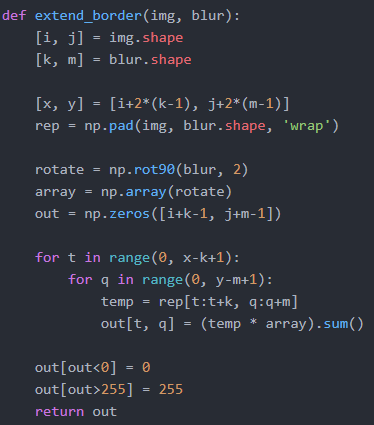


Now in the code part above, firstly we need to know what size are the parameters is. To do so we call the shape method. Then we are trying to find the new size of the blurred image. After that we are creating our 2d matrix with zeros in the extended part and image values on the inside. This is called **zero-padding**.

After zero-padding array has created, we need to apply our kernel on it. To do that we need to know what value will be in it. To calculate that we need to iterate all possible elements on the zero-padded array and multiply them with our kernel and add to the previous value.

After all the calculations has been done we need to be sure that our RGB pixels do not be negative or greater than 255 (because all values in RGB is between 0 and 255). To do that, we need to iterate our new array one more time to make sure of that.

Now for the **border replication** part, we will almost do exactly like we done in the zero-padding part. But the only difference will be, instead of all zeros outside of the image, we will extend our borders with border replication. To do that we will use the numpy library.



Now the code is almost identical to the zero padding. The only difference is in the line 65 where we use pad function in numpy. What is does is, after creating our zero extended array, we will fill all values with images values on it. This is called border replication.

After the execution of the program the output will look like this:



**2)** For second question in this homework we was asked to find faces in the given image. To do so we will use 3 templates.

First one is called kid.jpg:



Second one is called man.jpg:



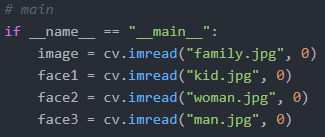
Third one is called woman.jpg:



And one image to search faces called family.jpg:



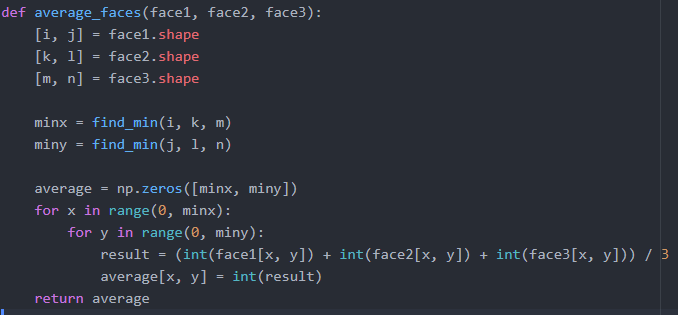
To read all of the images from our program we will use imread method from opencv:



Then we need to find averages on this images. In main:



In method called average\_faces:

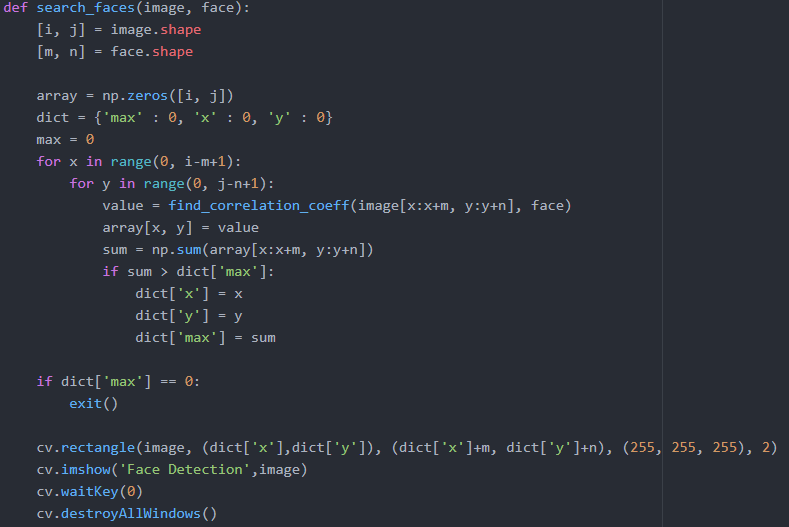


This code part is just doing one simple thing which is to iterate all the arrays and find and return the average of them.

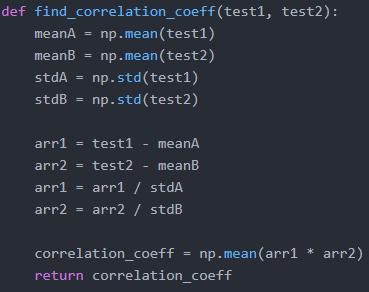
Then in main :



We are calling our face search method with two parameters. In the search\_faces:



In this part of the code, we are just iterating our face on the image and while iterating we are calculating our **correlation coefficient** by the function called find\_correlation\_coeff. We will look this function later. While calculating correlation coefficient we check whether one of them is greater than 0.67. If it is we draw a rectangle on top of it. Then we show the image. Now in the find\_correlation\_coeff function:



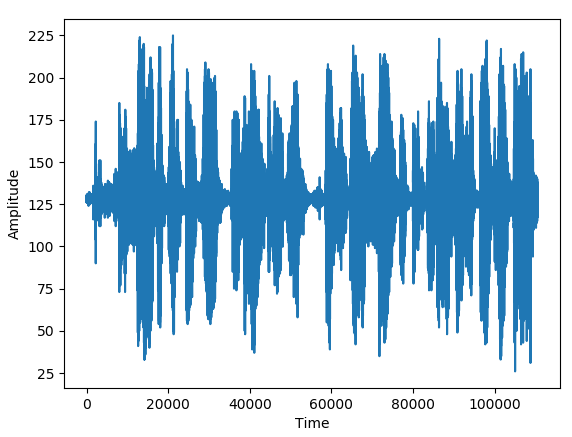
In the code part above we are calculating the correlation coefficient that we supposed to calculate with the given formula in the homework description. To calculate the formula we need to find mean and standard derivation of these matrixes. Numpy simply does this for us. Then we need to extract means from each matrixes. Then we divide them with standard derivations. After that all we need to calculate the mean of the multiplication of this matrixes. Then we return the value.

After execute our code the output image is:

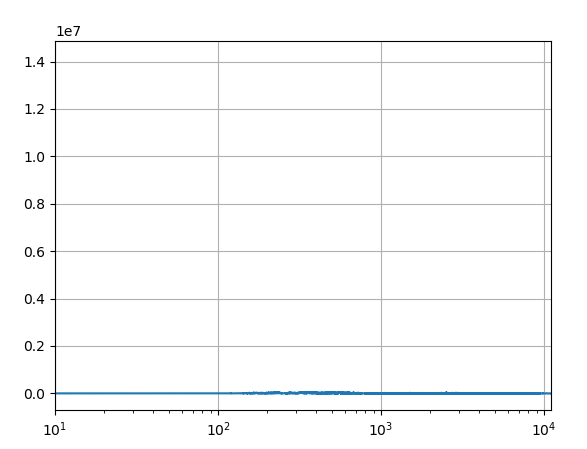
**3)**

In this question, we opened the wav file firstly.

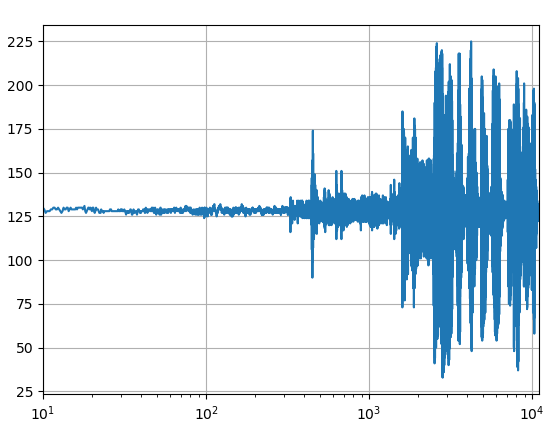
Here is our initial sound file frequency plot:



After that we took the data from wav file. Then, we took the ﬀt of frequencies of sound file. Here is the plotted sound file frequency:



After fft operation we took ifft of of frequencies of sound file. Here is the plotted sound file frequency:



At the end of implementation, sound didn’t change when we played it again.