Homework 5:

Question 1:

a) For implementing the given function scale_down(image, resize_ratio) we first will blur the image so it will reduce the high frequency of it so it will give us better scaling down and then we calculate the Fourier transform of the image so we will crop the frequencies to scall it down: fshifted_im[from_height:to_height, from_width:to_width]
and at the end we will get the image back using the inverse
and we did cv2.normalize for the final image to get the correct range of pixel values.

b) The same as scale down but now we will not do blur and we will put the frequency of the image on a larger scale.

c) No we will implement the ncc_2d the same way as we saw in the lecture :

$$\frac{\sum_{x,y\in N}\left[I(u+x,v+y)-\bar{I}_{uv}\right]\left[P(x,y)-\bar{P}\right]}{\left[\sum_{x,y\in N}\left[I(u+x,v+y)-\bar{I}_{uv}\right]^2\sum_{x,y\in N}\left[P(x,y)-\bar{P}\right]^2\right]^{1/2}}$$

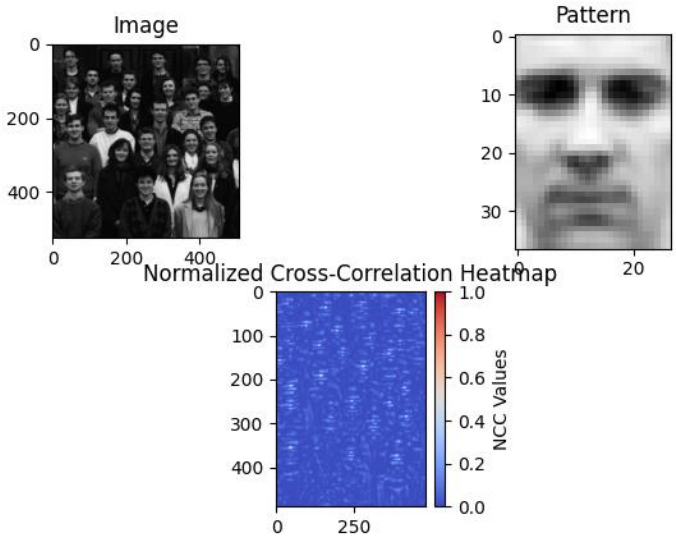
We used the np.lib.stride_tricks.sliding_window_view(image, pattern.shape) to create all the possible windows from the image and then we did the same as in the Formula given also we bayed attention the the cases denominator is 0 we maked sure to get a zero value.

d) +e

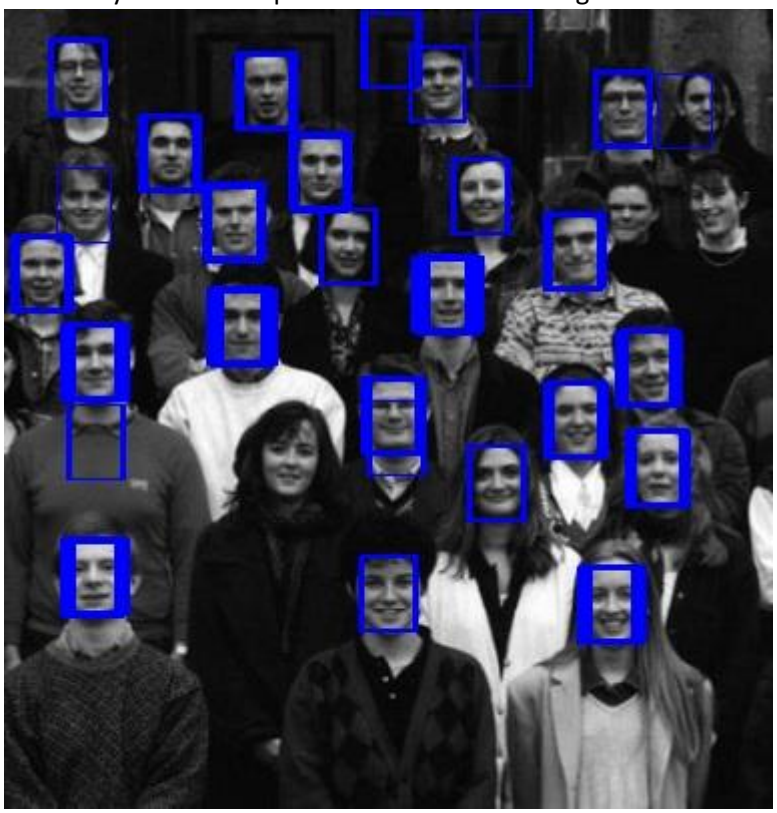
For the students picture we choose (we tried to choose values that maches the most faces at each picture did that in the painter app)

image_scaled = scale_up(students_img, resize_ratio=1.32)
pattern_scaled = scale_down(pattern, resize_ratio=0.75)

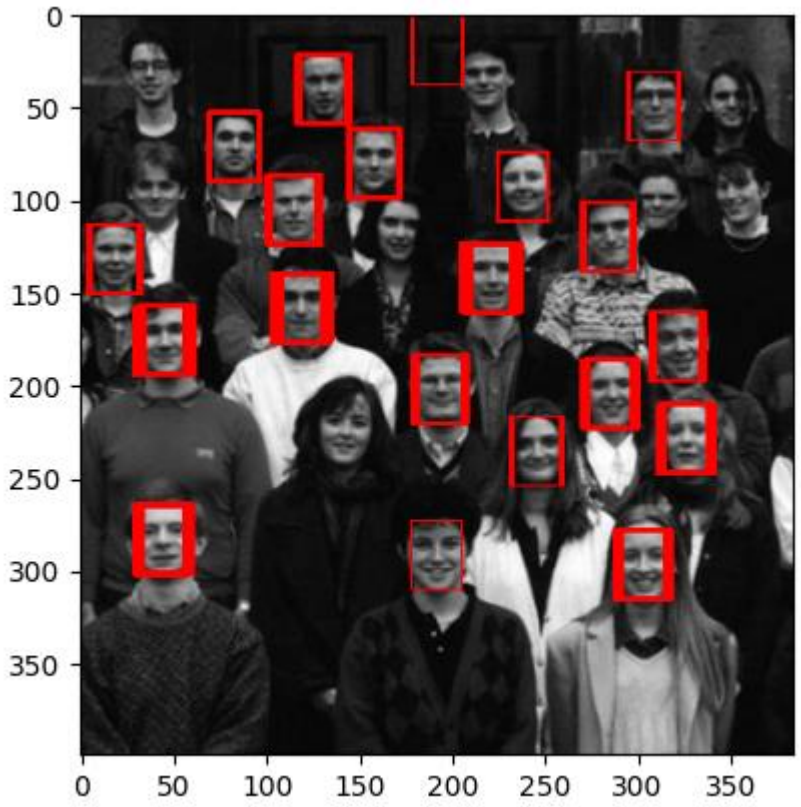
so we got this:



we clearly can see the point of matches and we got this result :



as we can see there is three false positive but we almost got all of the faces we can git ride of that by increasing the threshold a bit and we will get:

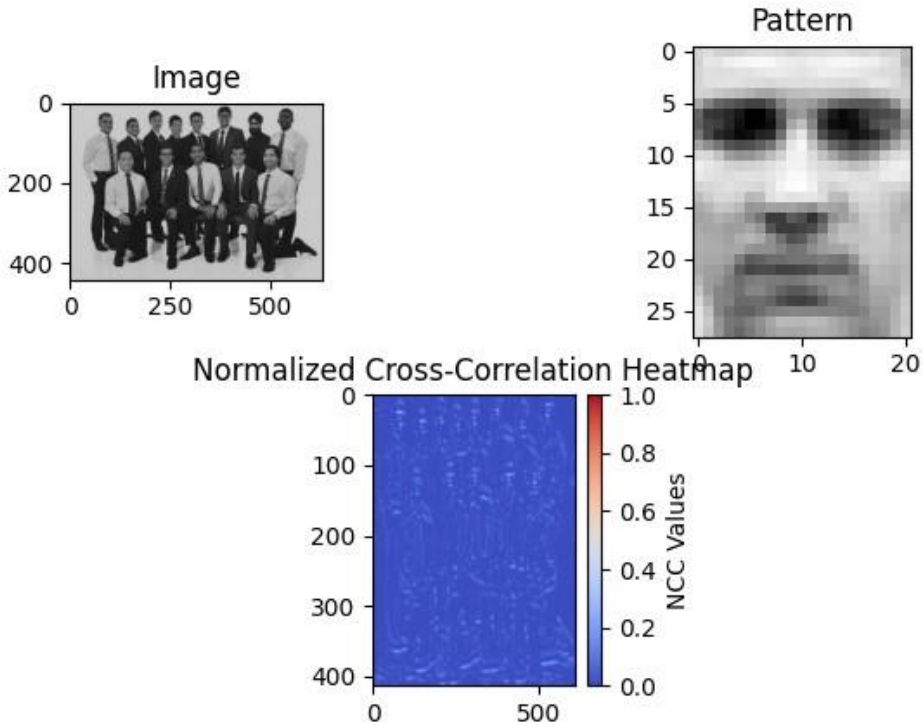


as we can see that we got less faces.

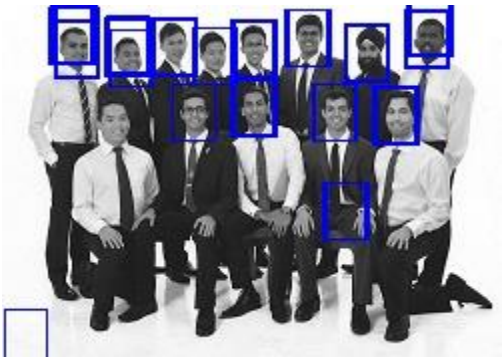
In the next picture the crew:

image_scaled = scale_up(crew_image, resize_ratio=2.52)
pattern_scaled = scale_down(pattern, resize_ratio=0.58)

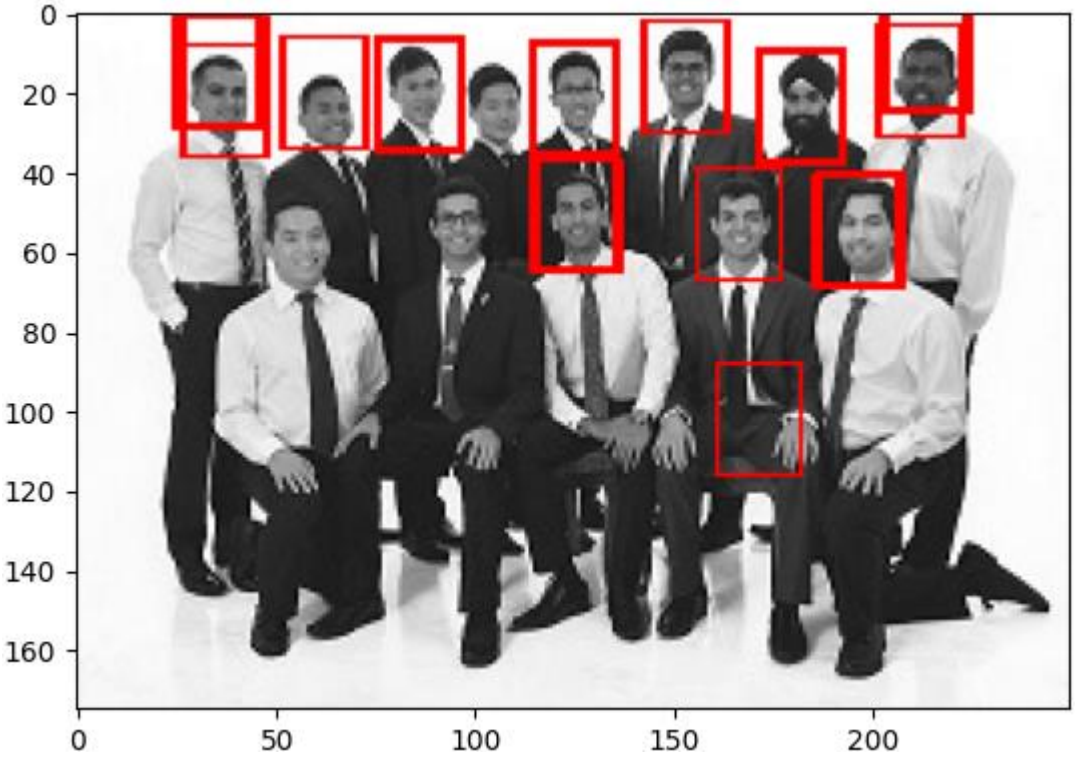
we used this values to scall the pictures we manage that using the painter to get the wanted size:



And the detected faces result like this:



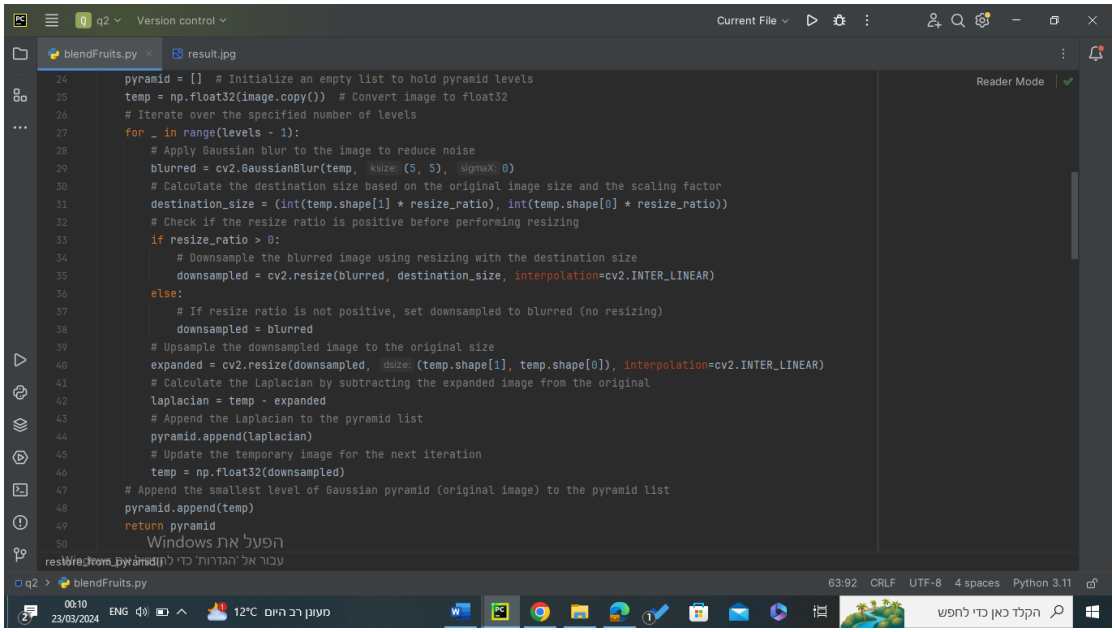
We have a small amount of false positive that we can get rid from by increasing the threshold:



Also we got less amount of detected faces.

Question 2:

a)*Implementing get_laplacian_pyramid:*



The **get_laplacian_pyramid** function constructs a Laplacian pyramid from a given image. A Laplacian pyramid is a multi-scale representation of an image that decomposes the image into different levels of detail. Here's an explanation of each part of the function:

1)Parameters:

image: This parameter represents the input image, which is expected to be a numpy array.

levels: Specifies the number of levels in the Laplacian pyramid.

resize_ratio: (Optional) It determines the ratio by which the image is resized at each level. The default value is set to 0.5, indicating that the image is resized to half of its original size at each level.

2)Returns:

It returns a list containing the laplacian pyramid levels

3)Functionality:

pyramid = []: Initializes an empty list to hold the Laplacian pyramid levels.

temp = np.float32(image.copy()): Converts the input image to **float32** format to ensure numerical stability during computations.

Iterates over the specified number of levels and does the following:

1) blurred = cv2.GaussianBlur(temp, (5, 5), 0): Applies Gaussian blur to the image to reduce noise

2)Computes the destination size for resizing based on the original image size and the resize ratio.

3)Checks if the resize ratio is positive and performs resizing accordingly

4)Upsamples the downsampled image to the original size.

5)Calculates the Laplacian by subtracting the expanded image from the original.

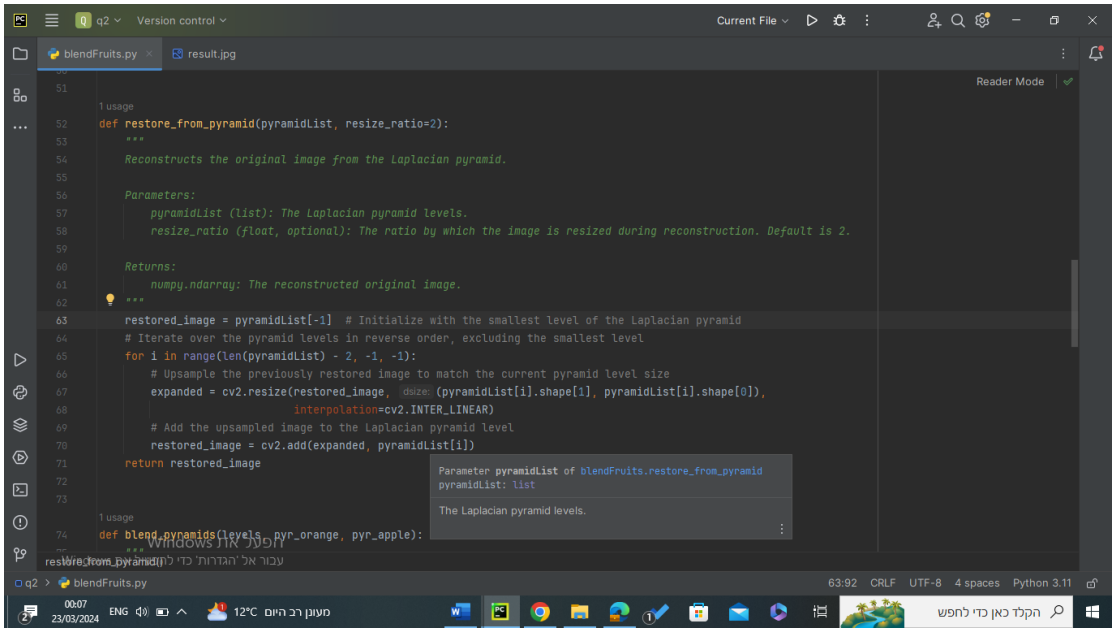
6)Appends the Laplacian to the **pyramid** list.

7)Updates the temporary image (**temp**) for the next iteration.

After we are done with the loop the function Appends the smallest level of the Gaussian pyramid (original image) to the **pyramid** list.

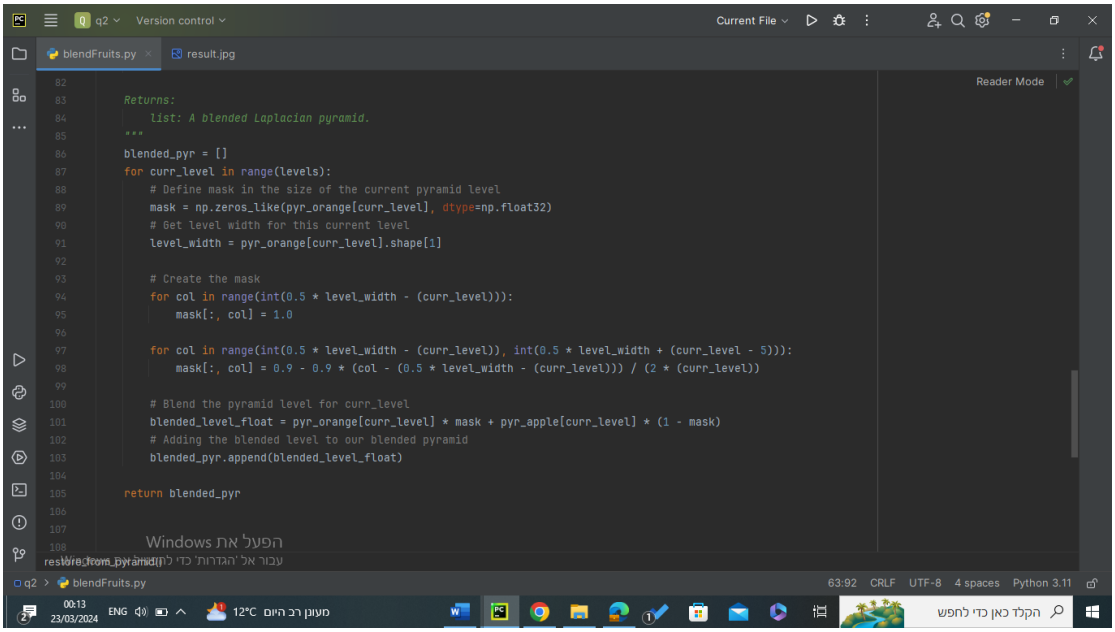
Returns the constructed Laplacian pyramid.

b)*Implementing restore_from_pyramid:*



as we can see to restore the image from the pyramid we need to do something called a collapse where we start from the smallest level and then iterate over the pyramid levels in reverse order (to get from smallest to highest), we basically use a for loop where we upsample the previously stored image and add it to the current laplacian pyramid level.

C)*implementing blend_pyramids:*



For each level(we chose 5 levels because it gives us the best answers) in the pyramids we do the following:

1)define mask the size of the current pyramid level.

2)get the level width for this level

3)creating the mask using advanced indexing techniques to get the smooth blurring between the apple and orange image.

4)then we blend the 2 pyramids on this level using this line blended_level_float = pyr_orange[curr_level] * mask + pyr_apple[curr_level] * (1 - mask)

5)we add the blended level to our blended pyramid