Image Processing: Assignment #5

Problem 1 – Template matching

key components:

- Non-Maximum Suppression: A method used to thin out the set of detected features or patterns in an image. It does so by retaining only the local maxima above a certain threshold and that are a certain distance away from each other, effectively filtering out weaker, closely clustered detections.
- Scale Down Function: This function reduces the resolution of an image or a pattern by performing a Fourier transform, cropping the transform in the frequency domain, and then applying an inverse Fourier transform to obtain a scaled-down image.
- Normalized Cross-Correlation (NCC): A technique for template matching, which involves sliding the template pattern across the image to find the area that. matches most closely with the template. It does so by calculating the correlation between the template and every possible position of the same size in the image, resulting in a score that indicates the similarity at each position. High scores represent a high similarity between the image segment and the template.

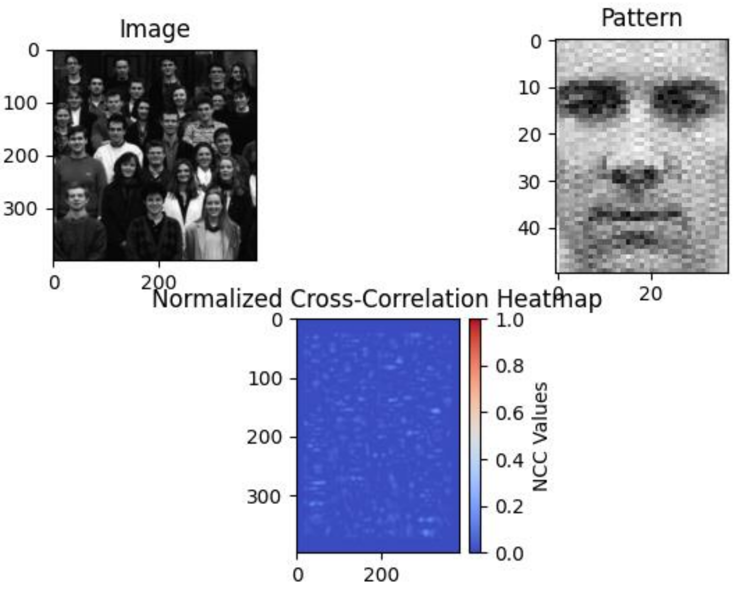
The script executes two main blocks of operations, one named "Students" and the other "Crew," each applying the process to different images. For both blocks, the process involves:

- Scaling down the pattern to match the scale of features in the image.
- Finding matches using NCC and non-maximum suppression with specific thresholds and minimum distance parameters.
- Drawing rectangles around the matches in the original images and displaying these annotated images.

Results:

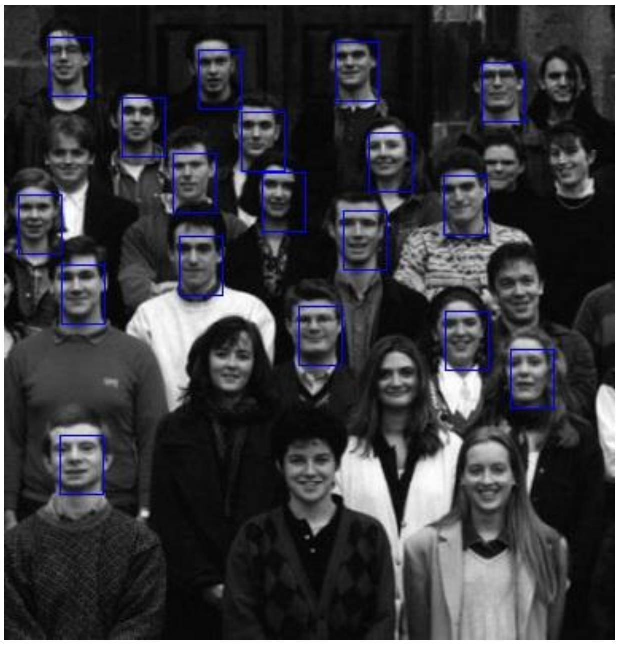
- Students

We choose to scale down the pattern image with resize ratio of 0.5.

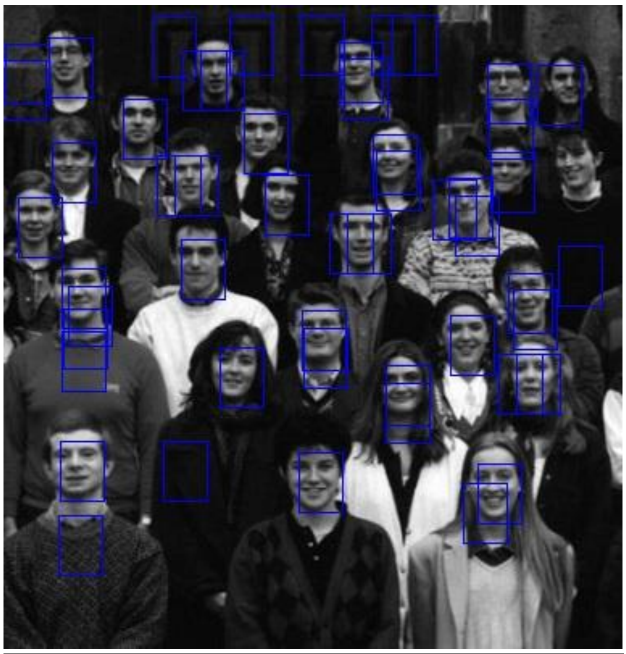


We tried several options of thresholds and below are the results:

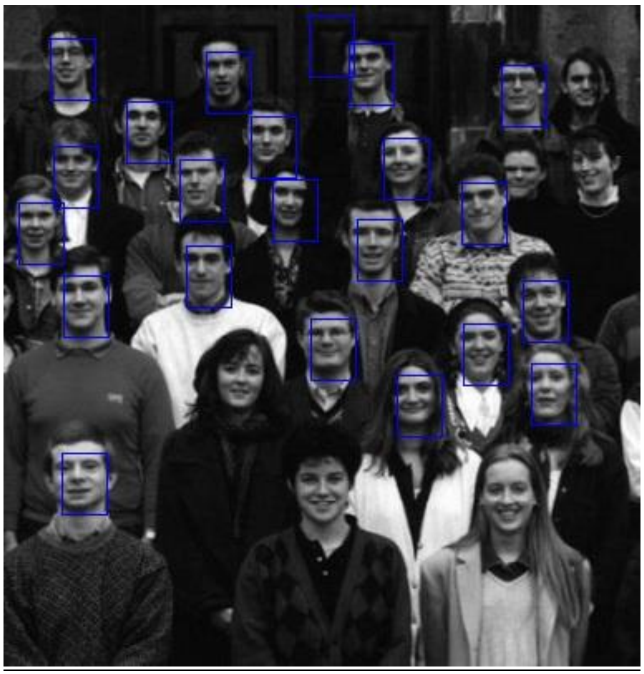
- For threshold 0.6 we found less matches, but we have no false positives.



- For a threshold of 0.4 we found almost all possible matches, but we have many false positives.

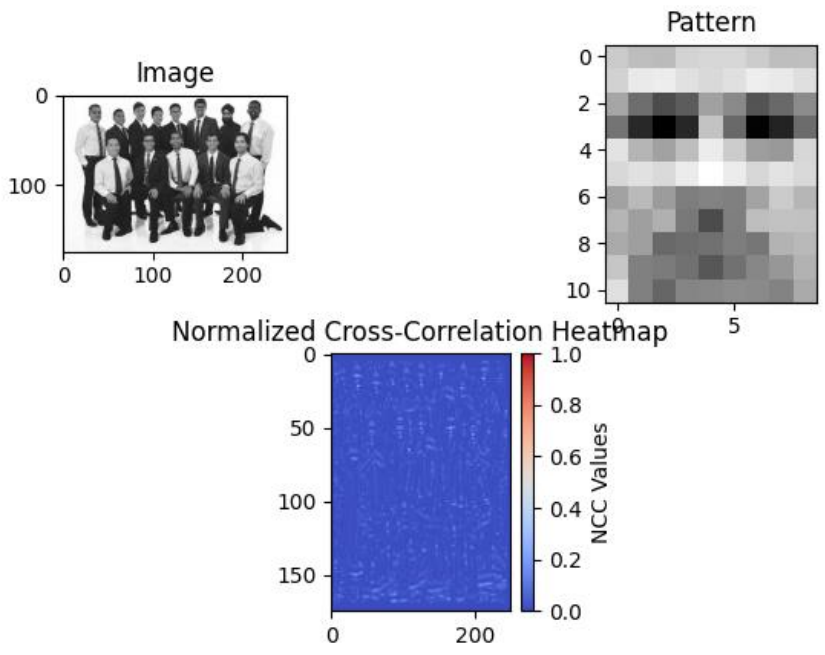


- For a threshold of 0.55 we found a balance between the number of matches and the number of false positives.



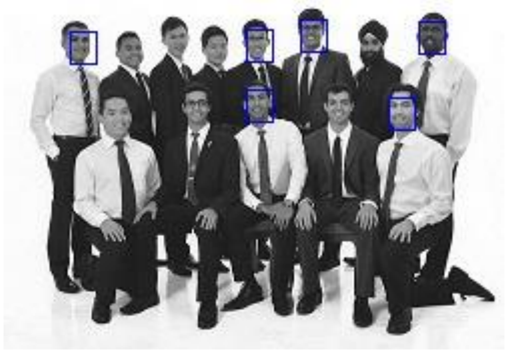
- Crew

We choose to scale down the pattern image with resize ratio of 0.23.

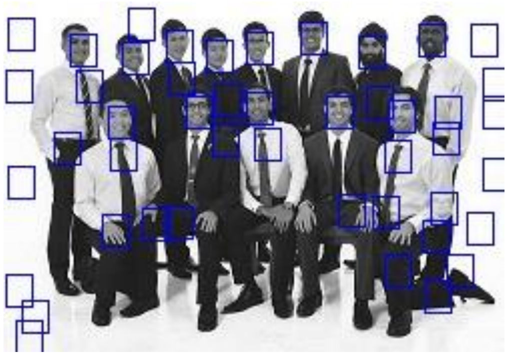


We tried several options of thresholds and below are the results:

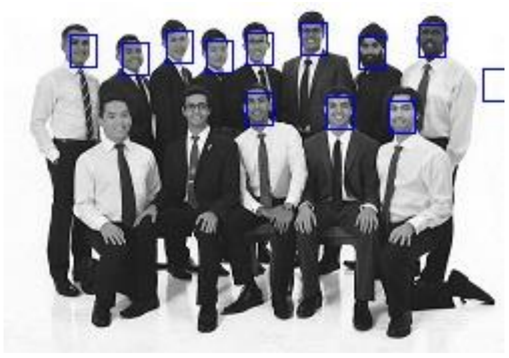
- For a threshold of 0.5 we found less matches, but we have no false positives.



- For a threshold of 0.33 we found almost all possible matches, but we have many false positives.



- For a threshold of 0.45 we found a balance between the number of matches and the number of false positives.



<u>Problem 2 – Multiband blending</u>

Building the Laplacian Pyramid:

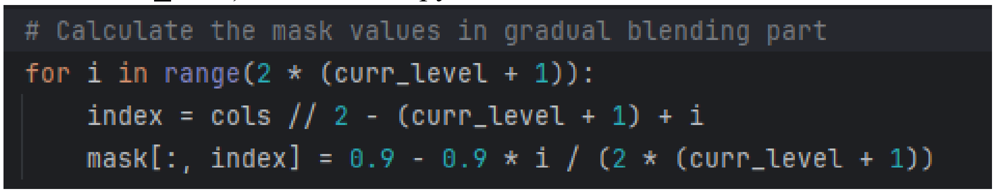
The get_laplacian_pyramid function constructs a Laplacian pyramid for a given image. We start with a Gaussian pyramid by applying a Gaussian blur with kernel size 5X5 with cv2.GaussianBlur and scaling down with cv2.resize with given resize ratio (0.5) iteratively. For each level, except the last, we upscale with cv2.resize the next level and subtract it from the current one with cv2.subtract to create the Laplacian pyramid. The last level of the Gaussian pyramid is added directly to the Laplacian pyramid.

Restoring Images:

The restore_from_pyramid function rebuilds the original image from its Laplacian pyramid. It starts from the smallest image in the pyramid and progressively upscales (with cv2.resize) and adds back (with cv2.add) the detail from the Laplacian levels. This is done in reverse order, starting from the second-to-last level back to the first.

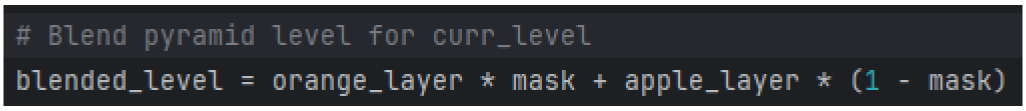
Blending Two Images:

The blend_pyramids function takes two Laplacian pyramids and blends them at each level using a dynamic mask. The mask is initialized with ones up to transition start point ((0.5 * width - curr_level) based on the pyramid level, calculate the gradual blending part in mask by:



```
# Calculate the mask values in gradual blending part
for i in range(2 * (curr_level + 1)):
    index = cols // 2 - (curr_level + 1) + i
    mask[:, index] = 0.9 - 0.9 * i / (2 * (curr_level + 1))
```

And then we blended pyramid level for curr_level:

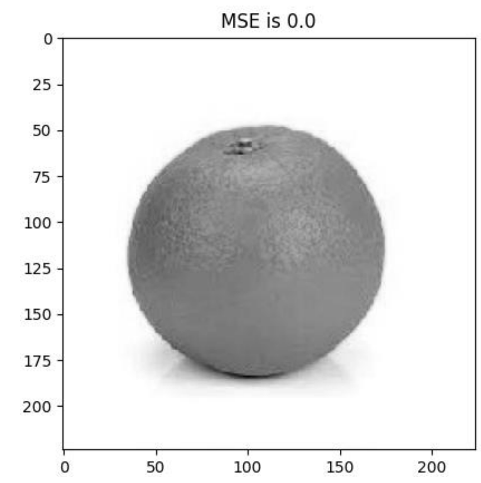


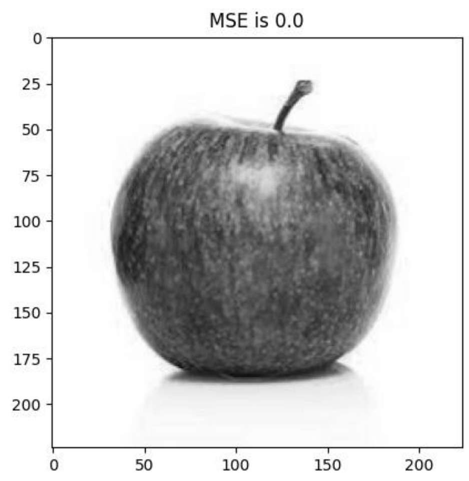
```
# Blend pyramid level for curr_level
blended_level = orange_layer * mask + apple_layer * (1 - mask)
```

We used 5 levels in our pyramids.

Results:

In validate operation the Laplacian pyramid and the image restore works correctly:





Blended image: