Image & Video Processing Resit Assignment

June 2021

Elif Yozkan
i6219160

Examiner : Alexia Briassouli

1. Frequency Transforms

1. **Code exercise :**
   3 x 3 Box filter is applied to the original image f(x,y) shown below using convolution :

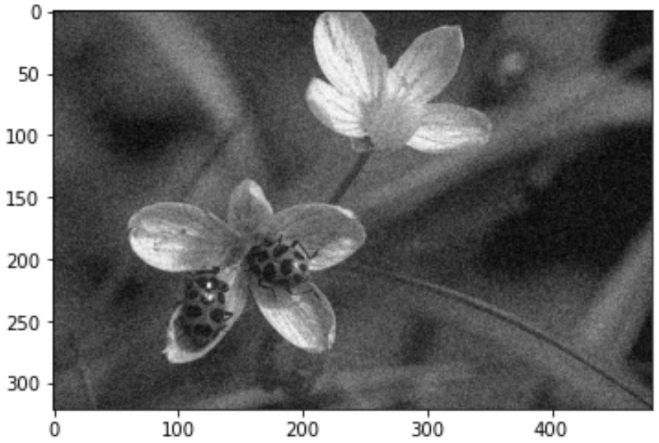


Figure 1: The original image $f(x, y)$

The convolution and filtering process involves a kernel or filter $h(x, y)$, which is a matrix smaller than the width and height of the original image $f(x, y)$. In this case, the kernel is a 3x3 box filter.

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Figure 2: 3x3 Box filter $h(x, y)$. Taken from the source: http://6.869.csail.mit.edu/fa16/lecture/lecture3linearfilters.pdf

In order to obtain the convolved or the filtered image first of all the first element of the kernel is placed at each pixel of the image. Then for each 3x3 pixel block in the image $f(x, y)$, each pixel value and its corresponding kernel value $h(x, y)$ is multiplied, then the sum of all multiplied pixels in the block is taken and then places in the center of the block. This sum represents the convolved pixel. These altered pixels all together compose the filtered image $g(x, y)$.

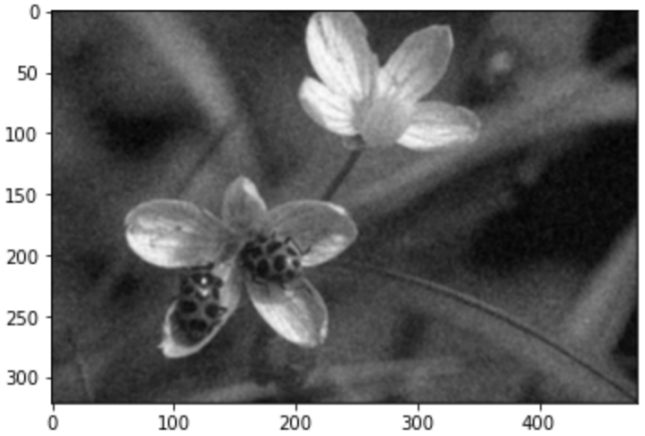


Figure 3: The filtered image $g(x, y)$using a 3x3 box filter

As seen from the resulting image $g(x, y)$In Figure 3 application of box filter applied a blur to the original image to some degree. This blur is specifically most visible in the background as the noisy background appears more smoothed out in the resulting image.

2. Maths Exercise:

The filtered image $g(x, y)$is obtained by the convolution of the original image $f(x, y)$ and the box-filter $h(x, y)$. In order to find the FT of the convolved image. The following formula can be used **:**

$$G(u,v) = \frac{1}{M}\frac{1}{N}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} g(x,y)e^{-j2\pi ux/M}e^{-j2\pi vy/N}$$

where $u = 0,\ldots, M-1$ and $v = 0,\ldots, N-1$, and M and N denotes the width and height of the image in terms of pixels and $j$denotes the complex term.

It is worth noting that the formula used for the Fourier Transform is discrete, since we are dealing with a non-infinite image.

By definition, it is known that the convolved image $g(x, y)$ is equal to the summed product of the original image and the filter :

$$g(x, y) = f(x, y) * h(x, y)$$

Therefore, it is possible to infer that, the Fourier transform of $g(x, y)$which is $G(u, v)$is also obtained from the convolution of FT of $f(x, y)$and $h(x, y)$which are $F(u, v)$and $H(u, v)$ respectively :

$$G(u,v) = F(u,v) * H(u,v)$$

If we apply the Formula of the Discrete Fourier Transform to $F(u,v) * H(u,v)$we then obtain :

$$F(u,v) * H(u,v) = \frac{1}{M}\frac{1}{N}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}(f(x,y) * h(x,y))e^{-j2\pi ux/M}e^{-j2\pi vy/N}$$

Using the Euler's Formula, which states that $e^{j\theta} = cos(\theta) + jsin(\theta)$, we could reduce the formula above to a simpler form excluding the power terms :

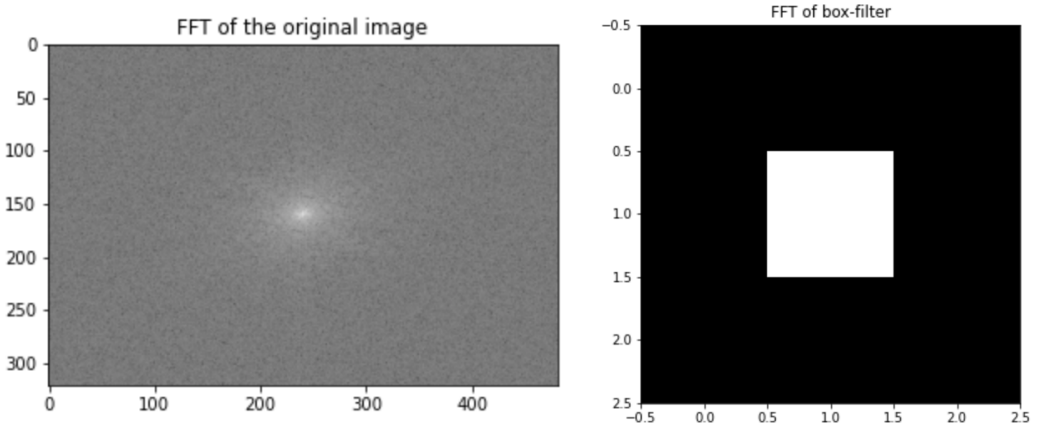
$$F(u,v)*H(u,v) = \frac{1}{M}\frac{1}{N}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}(f(x,y)*h(x,y))[cos(\frac{2\pi*(ux)}{M})-jsin(\frac{2\pi ux}{M})][cos(\frac{2\pi vy}{N})-jsin(\frac{2\pi*vy}{N})] \quad (6)$$

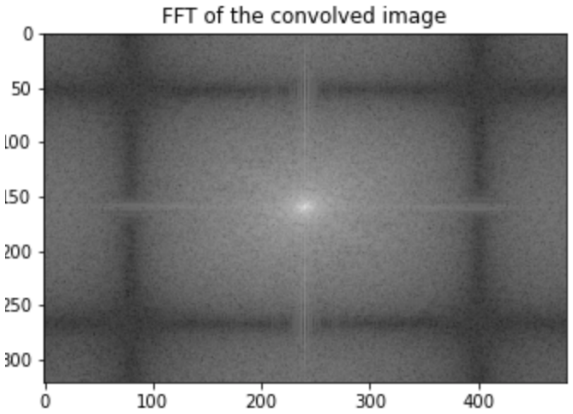
which is equivalent to the final equation below :

$$G(u,v) = \frac{1}{M}\frac{1}{N}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}(f(x,y)*h(x,y))[cos(\frac{2\pi*(ux)}{M})-jsin(\frac{2\pi ux}{M})][cos(\frac{2\pi vy}{N})-jsin(\frac{2\pi*vy}{N})] \quad (7)$$

3. Code Exercise :

Figure 4 demonstrates the original image $f(x,y)$, the filter $h(x,y)$ and the convolved image $g(x,y)$in the frequency domain as the result of Fourier Transform.





As seen in the FFT of the filter the low-frequency areas in the center have the value 1 and relatively high-frequency areas are 0, therefore the box-filter acts similar to a low-pass filter, considering the original image and the convolved image, it is apparent that box filter weakens the high spatial frequency areas within the original image and results in the convolved image. This is visible if the FFT of the original image and the FFT of the convolved image are compared. The convolved image has more zero values in the frequency domain. In terms of appearance, it is possible to say that the convolution theorem holds as the resulting image $g(x, y)$ indeed appears to be a filtered image applied on the original image $f(x, y)$ using the box filter $h(x, y)$.

4. Maths Exercise :
   We can apply inverse Fourier transform to $G(u, v)$in order to obtain $g(x, y)$.
   The inverse Fourier transform of $G(u, v)$is given below :

$$g(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} G(u, v) e^{j2\pi * ux/M} e^{j2\pi * vy/N}$$

Given that $G(u, v) = F(u, v) * H(u, v)$, the inverse Fourier transform of $G(u, v)$, which is $g(x, y)$is also the convolution of $f(x, y)$and $h(x, y)$:

$$F^{-1}(G(u, v)) = g(x, y) = f(x, y) * h(x, y)$$

If we replace the convolution of $f(x, y)$and $h(x, y)$ inside the Inverse Fourier Transform formula, then we get the following formula :

$$f(x, y) * h(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) * H(u, v) e^{j2\pi * ux/M} e^{j2\pi * vy/N}$$

Question 2: Image Restoration

1. **Maths Exercise:**

Image degradation is modeled by the convolution of the original image $f(x, y)$and the filter $h(x, y)$ and addition of the noise $n(x, y)$if exists :

$$g(x, y) = h(x, y) * f(x, y) + n(x, y)$$

and the Fourier transform of the degradation function is shown as follows:

$$G(u, v) = H(u, v) * F(u, v) + N(u, v)$$

In our case, our degradation involves a linear motion blur in each direction for constants a and b. Thus, the degradation model $g(x, y)$can be represented as :

$$g(x, y) = f(x - at, y - bt)$$

Fourier transform of this degradation then becomes :

$$G(u, v) = \frac{1}{M}\frac{1}{N}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} g(x, y)e^{-j2\pi u(x-at)/M}e^{-j2\pi v(y-bt)/N}$$

$$G(u, v) = \frac{1}{M}\frac{1}{N}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x - at, y - bt)e^{-j2\pi u(x-at)/M}e^{-j2\pi v(y-bt)/N}$$

where $u = 0,\ldots, M - 1$ and $v = 0,\ldots, N - 1$, and M and N denotes the width and height of the image in terms of pixels and $j$denotes the complex term. Using Euler's Formula, we could reduce the formula above to a simpler form excluding the power terms :

$$G(u, v) = \frac{1}{M}\frac{1}{N}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}(f(x - at, y - bt)[cos(\frac{2\pi * (u(x - at))}{M}) - jsin(\frac{2\pi u(x - at)}{M})][cos(\frac{2\pi v(y - bt)}{N}) - jsin(\frac{2\pi * v(y - bt)}{N})]$$

2. Code Exercise:

In this exercise, Motion blur filter and Gaussian Noise is applied to the image $f(x, y)$, which is displayed in Figure 5.

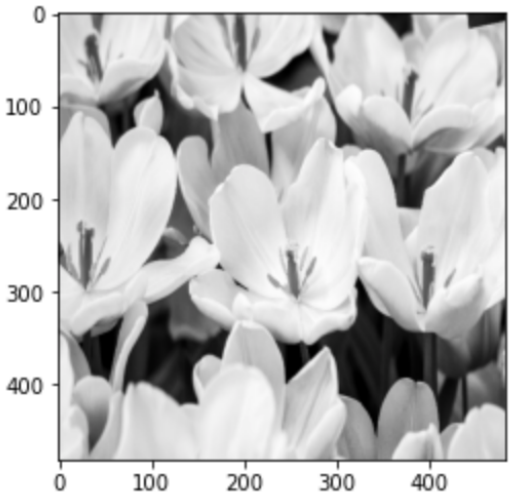


Figure 5 : The original image $f(x, y)$

After filtering image using a vertical Motion Blur filter $h(x, y)$, it is apparent in Figure 6 that the main components (flowers and two ladybugs) are less recognizable due to the smearing occured in the vertical direction in the filtered image $g(x, y) = f(x, y) * h(x, y)$. The edges of these components are also less visible and distinguishable.

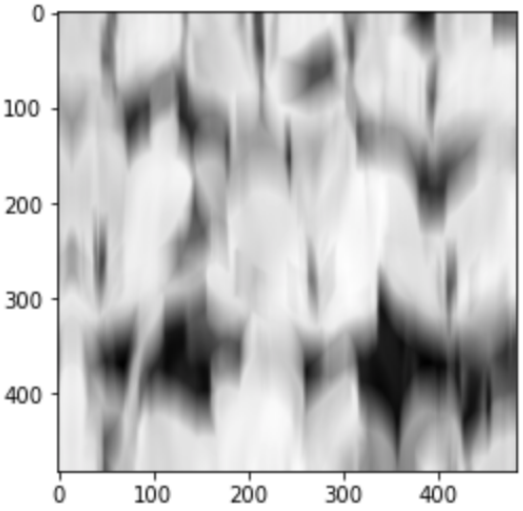


Figure 6: Motion-Blurred image

When we finally add the Gaussian noise $n(x, y)$ to our degraded image $g(x, y) = f(x, y) * h(x, y)$, it is possible to observe that in the resulting image $g(x, y) = f(x, y) * h(x, y) + n(x, y)$, the main components are even less distinguishable due to the noise.

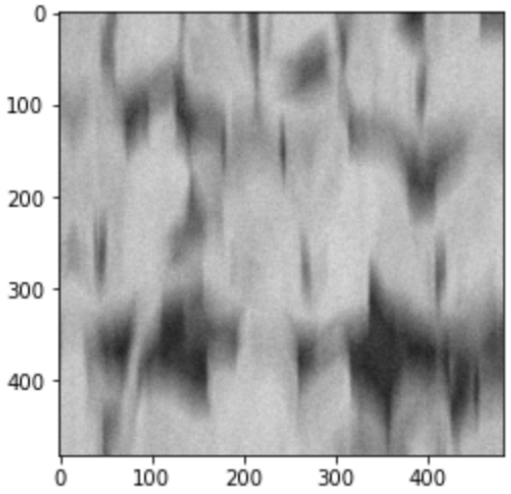
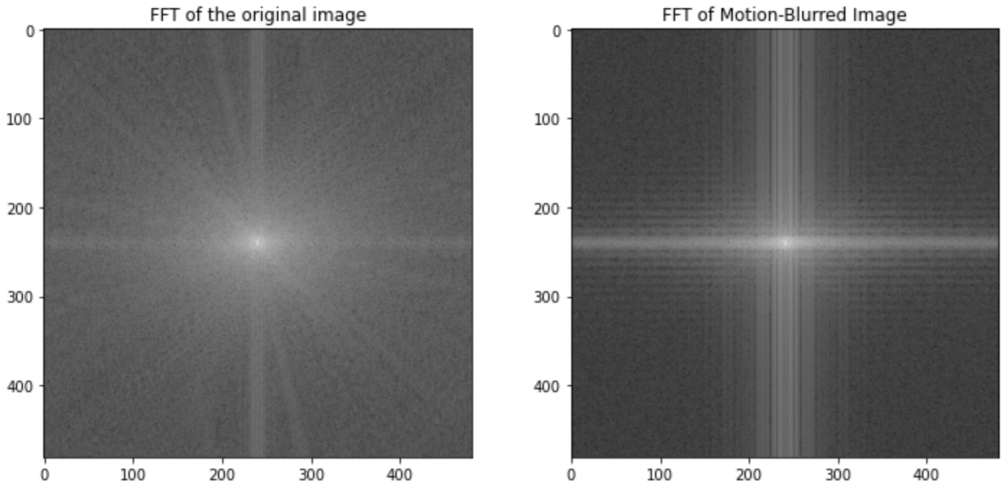


Figure 7: Image with Motion Blur + Gaussian noise

When the images are examined in the frequency domain shown in Figure 8, firstly we see that the image the slanted lines and the horizontal and vertical lines which go along the high-frequency areas, which often represent the edges in the spatial image, of the FFT image are more “homogeneously” scattered. On the other hand, if we have a look at the FFT of the Motion-Blurred Image, we see that those lines tend to line up vertically, and the high-frequency areas are weakened. This is probably due to the motion-blur filter being applied to the original image vertically. Moreover, the weakened high-frequency areas especially with the disappearance of the diagonal lines in the FFT of Motion-Blurred image can be explained by the increased blur in the image and the smoothed-out edges.

As for the FFT of the motion-blurred and noisy image, it is obvious that the dominant vertical high-frequency areas do not exist within the noisy image as a result of noise addition. Due to the noise, the clear distinction of strong high-frequency and low-frequency components also disappeared.



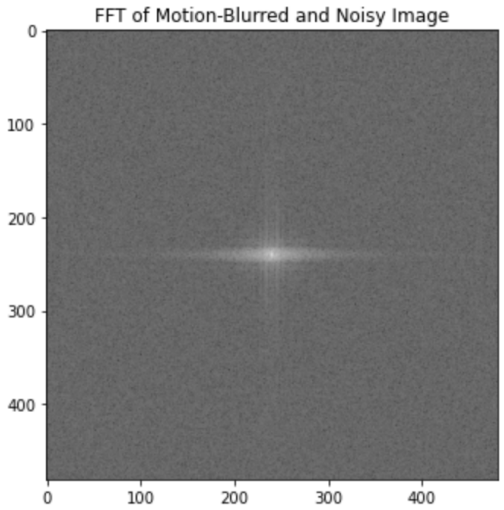


Figure 8: FFT's of the original image, motion-blurred image, and noisy image

3. Code Exercise :

The degraded image withmage$g(x, y) = f(x, y) * h(x, y) + n(x, y)$with motion-blur $h(x, y)$and Gaussian noise $n(x, y)$ is restored using Inverse and Wiener filters in this exercise.
As seen in Figure 9 , the restored image using Wiener filter, the noise added is almost not visible, and there still exists some blur to some degree. This is because Wiener filter aims to linearly estimate a sensible tradeoff between the noise reduction using Mean Squared Error.

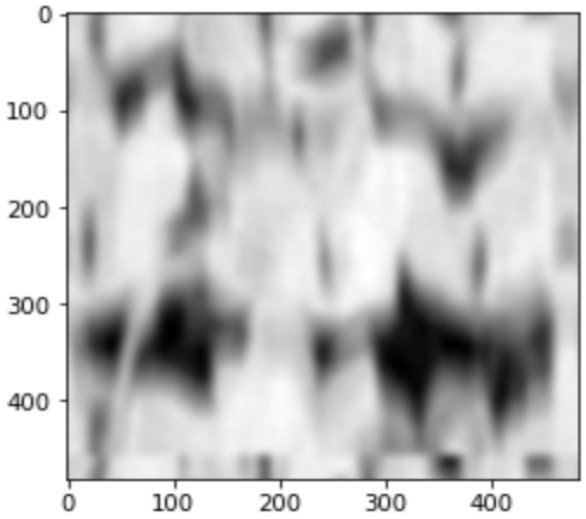


Figure 9: Degraded Image after Wiener Filter Restoration

In case of Inverse filtering, it is visible that most of the noise remains after the restoration process. Inverse filtering algorithm employs a deconvolution process to a filtered image. However as seen in the restored image in Figure 10,this deconvolution process is not quite robust to the noise added in the degraded image.

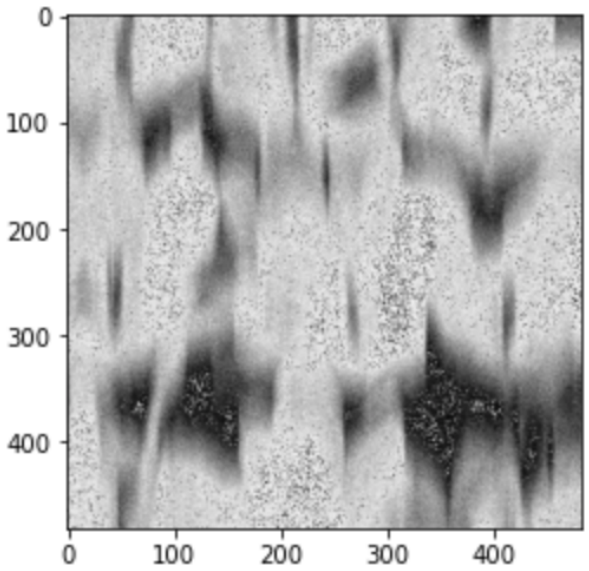


Figure 10: Degraded Image after Inverse Filter Restoration

3. **Question 3 : Morphology**

In this exercise grayscale granulometry is applied in order to find the frequency of different-sized circular components in the following images :

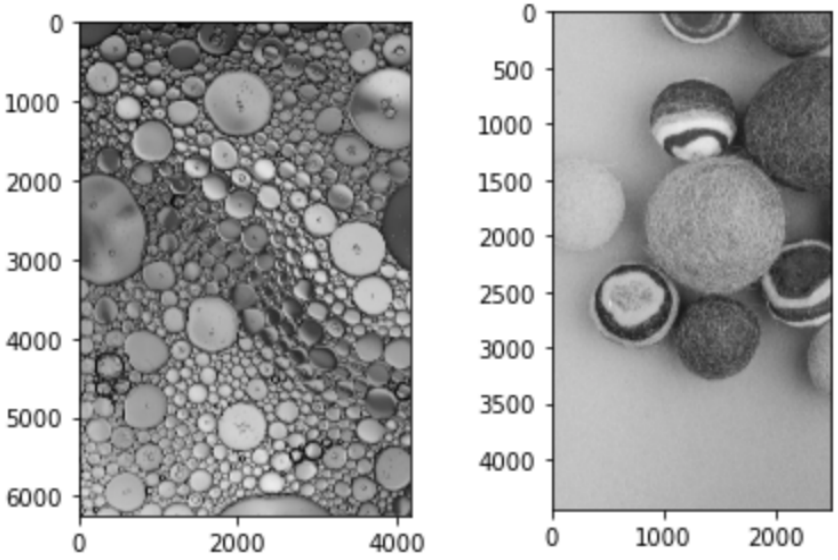


Figure 11 : Original Images

To extract the number of circular shaped components, i.e, grains in each image, Median Filtering is applied as a first step in order to reduce any potential noise that could affect the grain size detection process. The resulting images are shown in Figure 12.

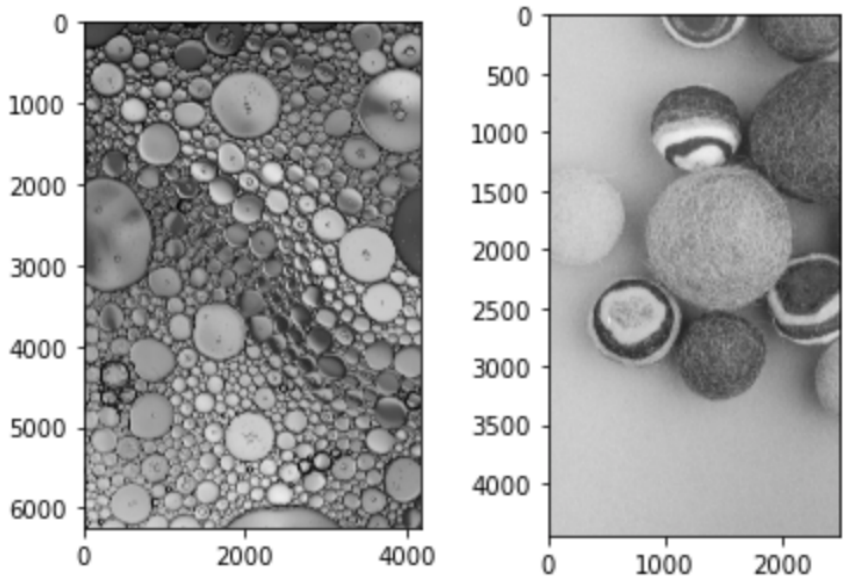


Figure 12 : Images after applying Median Filter

Then, the images are segmented and converted into binary images using thresholding to allow for easier operations afterwards. Another benefit of thresholding in this context is that it makes the components appear more distinguishable. The resulting thresholded binary images are as follows :

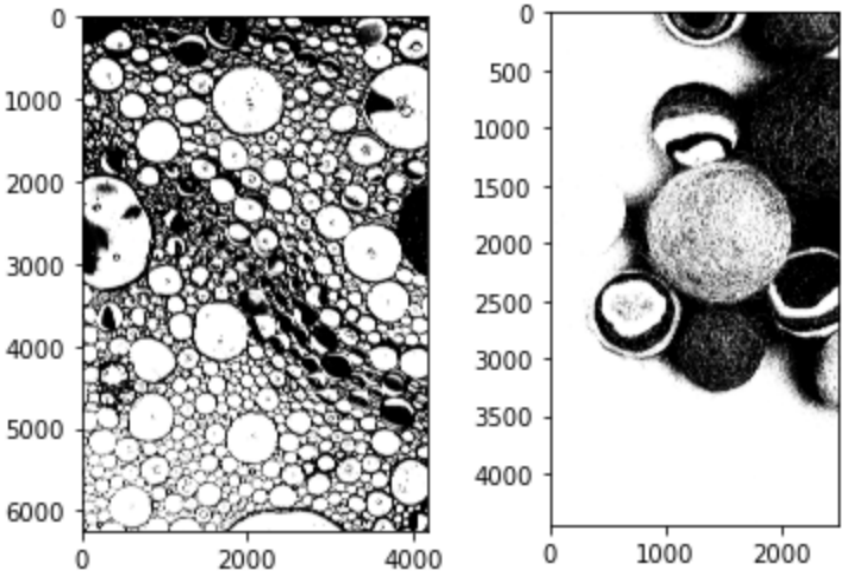


Figure 13 : Thresholded images

Afterwards, the process continued with the application of morphological opening, in other words, for each image, erosion is applied to the image and dilation process is then applied to the eroded image. As a critical note, the same structuring element which is a 3x3 matrix ,is utilized in these consecutive morphological operations. Erosion allows us to eliminate the small undesired pixels which carry no information in our context. Then by dilation we enlarge the boundaries of the white areas or our target grains. This ensures that we preserve the structure of main components in the image. Opening generally allows us to remove imperfections in the image.

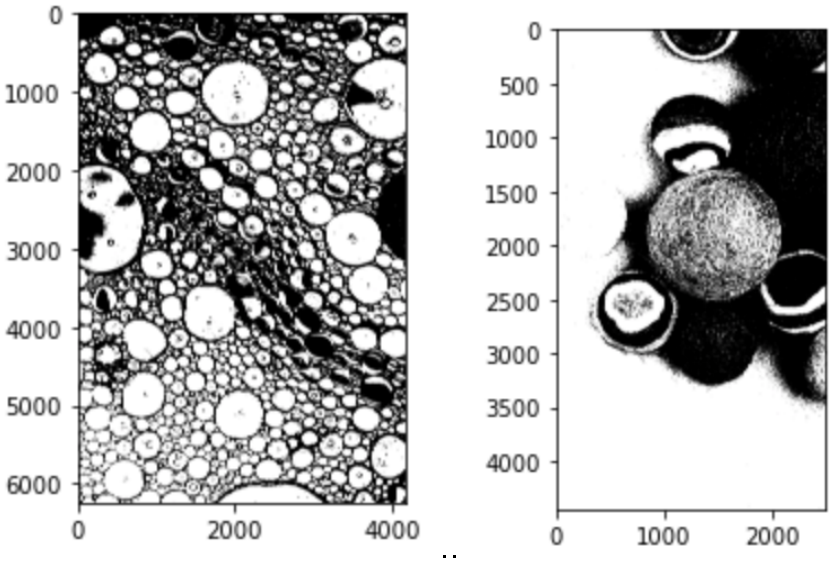


Figure 14: Resulting images after Morphological Opening

Once the morphological operations to achieve the final desirable structure of the images, connected components labeling is applied to each image , in order to detect each grain appearing in both images. As seen from the results of connected components labeling using 8-connectivity in Figure 14, we observe that the labeling process was quite successful as almost all individual grains are labeled correctly in the first image. However, concerning the image containing different balls, we observe that one of the light-colored balls is infused with the background and the whole background is labeled as a grain which is incorrect. This could be due to the thresholding application followed by erosion , the light-colored elements were labeled as 1 in balls image after thresholding operation shown in Figure 13. Erosion supported this mislabeling as it could have removed the small pixels which indicate the boundary of the light-colored ball. Another interesting outcome is that the stripe patterns also were labeled incorrectly after connected components labeling.

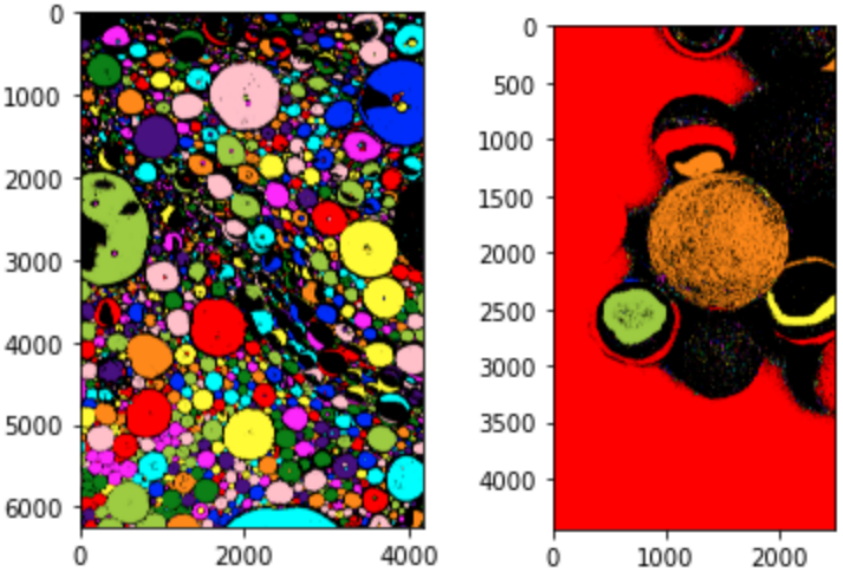


Figure 15 : Resulting images after connected-components labeling

Finally, for size frequency analysis, the area of each grain obtained from the connected-component labeling, is calculated for both images and the following information is then collected:

For Image I (which contains bubbles) :

```
The number of grains detected :  7011
Largest grain area :  703645
Smallest grain area :  6
Mean grain area :  2213.5786621024104
```

Figure 16 : Statistical information of grain sizes for Image I

As seen from the statistical data that is attained in Figure 16, we see that the range of grain sizes are extremely large, this could be due to the misclassification of a small pixel cluster as a grain, and misclassification of multiple grains as one single grain. Apart from that, the most frequent grain size is in the range of 0 and 10000, specifically in the range of 0-2000 as seen in the histogram in Figure 17.

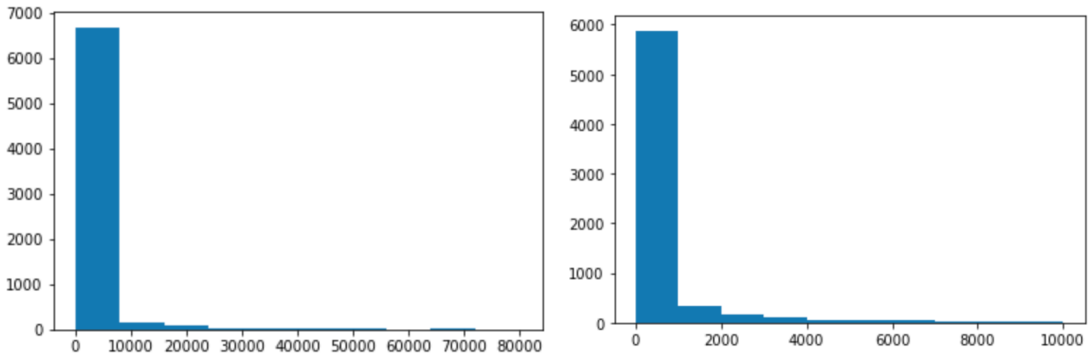


Figure 17 : Histograms of different ranges of grain size frequencies of Image I

For Image II(which contains balls):

```
The number of grains detected :  5003
Largest grain area :  6033186
Smallest grain area :  6
Mean grain area :  1483.739956026384
```

Figure 18 : Statistical information of grain sizes for Image II

For Image II we again encounter a massive range between the smallest and the largest grain area, which is due to the misclassification of the background as a large grain and the misclassification of small pixels as grains. Also if we observe the shrinked histogram of grain sizes in Figure 19, we see that the most frequent grain areas detected are between the range 0 and 100 units.

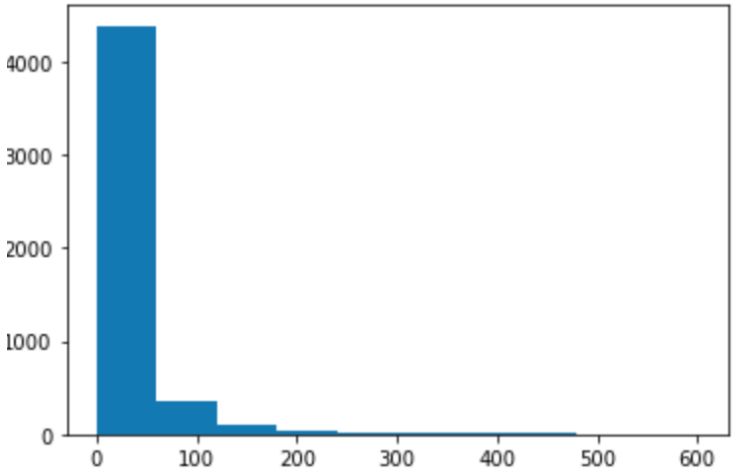


Figure 19 : Histogram of grain sizes of Image II

Question 4 : PCA Face Detection

Principal Component Analysis Algorithm for face detection is implemented in this exercise. Training images are composed of randomly generated frontal facial images of 10 different people . The training data contains the images of individuals of different skin tones, eye shapes, ages and facial expressions. First of all, the training image data is preprocessed to obtain $[N * N, 1]$ face vectors , where N is the dimension of the image in terms of pixels, to be able to compute the eigenvectors later. Following figure demonstrates the numerical values of the face vectors obtained from 10 pictures of the size $512\ x\ 512$.

```
Face vectors shape =  (262144, 10)
Face vectors :  [[162 181 180 ... 186 213 191]
 [163 180 180 ... 186 214 191]
 [163 179 180 ... 186 214 192]
 ...
 [168 163  27 ... 190  48  51]
 [168 159  32 ... 188  53  64]
 [163 152  36 ... 184  57  71]]
```

Figure 20 : Numerical values of face vectors

Then in order to eliminate common features from pictures to get more distinct features we apply normalization to the face vectors by subtracting the mean of the vectors from each face vector. So the normalized face vectors are :

$$n_i = x_i - m$$

where $m = 1/M \sum_{i=1}^{M} x_i$

the obtained normalized face vectors are shown below:

```
[[ -21.8   -2.8   -3.8 ...    2.2   29.2    7.2]
 [ -21.    -4.    -4.  ...    2.    30.     7. ]
 [ -21.1   -5.1   -4.1 ...    1.9   29.9    7.9]
 ...
 [  32.6   27.6 -108.4 ...   54.6  -87.4  -84.4]
 [  32.    23.  -104.  ...   52.   -83.   -72. ]
 [  29.1   18.1  -97.9 ...   50.1  -76.9  -62.9]]
```

Figure 21: Normalized Face Vectors

In order to find the eigenvectors for the next step, calculation of the Covariance Matrix is also needed prior to eigenvector computation. We can obtain the Covariance matrix by multiplying our normalized face vector by its transpose.

```
Covariance Matrix :  [[1043.36970612 -135.96505384 -302.21713356 -349.7
940082   208.06755594
  -212.42342156 -248.66945766  104.18551728 -111.06008815    4.50638362
]
 [-135.96505384 2793.28051742 -694.96822498  275.75636166 -148.97465375
  -307.59722549 -208.01106552  -17.09640476 -959.38616916 -597.03808157
]
 [-302.21713356 -694.96822498 2433.932098   -517.9001112  -350.4857361
  -351.65459832  -99.18778446 -136.62689268  513.4936454  -494.3852621
]
 [-349.7940082   275.75636166 -517.9001112  2666.19051277 -878.4358834
   740.51463105 -795.94235934 -683.2835648  -164.85476041 -292.25081813
]
 [ 208.06755594 -148.97465375 -350.4857361  -878.4358834  1415.74174404
  -373.99752583  173.65118488  156.07050578 -444.5364318   242.89924024
]
 [-212.42342156 -307.59722549 -351.65459832  740.51463105 -373.99752583
  1169.73111428   49.27442474 -141.5086054  -227.2570737  -345.08171977
]
 [-248.66945766 -208.01106552  -99.18778446 -795.94235934  173.65118488
    49.27442474 1675.66839948  552.4155053  -745.05183608 -354.14701134
]
 [ 104.18551728  -17.09640476 -136.62689268 -683.2835648   156.07050578
  -141.5086054   552.4155053   896.94893993 -536.57444915 -194.5305515
]
 [-111.06008815 -959.38616916  513.4936454  -164.85476041 -444.5364318
  -227.2570737  -745.05183608 -536.57444915 2433.73874895  241.48841411
]
 [   4.50638362 -597.03808157 -494.3852621  -292.25081813  242.89924024
  -345.08171977 -354.14701134 -194.5305515   241.48841411 1788.53940644
]]
```

Figure 22 : Covariance Matrix

Eigenvalues are calculated using the formula $C'u_i = \lambda_i u_i$ where C' denotes the covariance matrix and λ denotes the eigenvalues. Then the next step is to select k eigenvectors from C' with largest eigenvalues.

```
Eigen vectors :  [[-0.02515787  0.10511581 -0.18244832  0.08622524  0.3
1622777  0.3874029
   0.05326052  0.46857607 -0.68927585  0.00362489]
 [ 0.60732666  0.11680882  0.15847065  0.64189176  0.31622777  0.073213
59
  -0.14945173 -0.02273891  0.19385565  0.12074323]
 [-0.42571716 -0.07572238  0.66116415  0.15544591  0.31622777  0.046441
54
  -0.0875308   0.03209964  0.03209323 -0.48974559]
 [ 0.39486036 -0.63223212 -0.00986038 -0.24143645  0.31622777 -0.075354
57
   0.47368403 -0.0500228  -0.0343002  -0.22857254]
 [-0.05751565  0.35739019 -0.25796379  0.03522658  0.31622777 -0.107250
62
   0.09854327 -0.73908472 -0.2819559  -0.23285008]
 [ 0.14261423 -0.19643639  0.04941877 -0.45363682  0.31622777  0.023044
05
  -0.7631009  -0.12893815 -0.09275021  0.15736486]
 [ 0.0146906   0.40275202  0.21836341 -0.43055847  0.31622777  0.428901
62
   0.33223744 -0.01250493  0.35406581  0.29095451]
 [ 0.01463669  0.30325666  0.07429395 -0.13044948  0.31622777 -0.796376
88
   0.08748712  0.35180487 -0.04617312  0.13234713]
 [-0.48728922 -0.38644006 -0.10164996  0.29469625  0.31622777 -0.046214
08
   0.1033353  -0.15515723  0.0474657   0.61391695]
 [-0.17844864  0.00550745 -0.60978848  0.04259548  0.31622777  0.066192
44
  -0.14846427  0.25596616  0.51697488 -0.36778337]]
```

Figure 23:Eigen Vectors

```
K- eigenvectors :  [[-0.02515787  0.10511581 -0.18244832  0.08622524  0
.31622777  0.3874029
   0.05326052  0.46857607 -0.68927585  0.00362489]
 [ 0.60732666  0.11680882  0.15847065  0.64189176  0.31622777  0.073213
59
  -0.14945173 -0.02273891  0.19385565  0.12074323]
 [-0.42571716 -0.07572238  0.66116415  0.15544591  0.31622777  0.046441
54
  -0.0875308   0.03209964  0.03209323 -0.48974559]
 [ 0.39486036 -0.63223212 -0.00986038 -0.24143645  0.31622777 -0.075354
57
   0.47368403 -0.0500228  -0.0343002  -0.22857254]
 [-0.05751565  0.35739019 -0.25796379  0.03522658  0.31622777 -0.107250
62
   0.09854327 -0.73908472 -0.2819559  -0.23285008]
 [ 0.14261423 -0.19643639  0.04941877 -0.45363682  0.31622777  0.023044
05
  -0.7631009  -0.12893815 -0.09275021  0.15736486]
 [ 0.0146906   0.40275202  0.21836341 -0.43055847  0.31622777  0.428901
62
   0.33223744 -0.01250493  0.35406581  0.29095451]
 [ 0.01463669  0.30325666  0.07429395 -0.13044948  0.31622777 -0.796376
88
   0.08748712  0.35180487 -0.04617312  0.13234713]
 [-0.48728922 -0.38644006 -0.10164996  0.29469625  0.31622777 -0.046214
08
   0.1033353  -0.15515723  0.0474657   0.61391695]
 [-0.17844864  0.00550745 -0.60978848  0.04259548  0.31622777  0.066192
44
  -0.14846427  0.25596616  0.51697488 -0.36778337]]
```

Figure 24 : K-selected Eigenvectors

Next, we represent each k-selected eigenvector in other words the eigen face as a combination of the K eigenvectors. In order to accomplish this, we perform matrix multiplication with our normalized face vectors and the k-selected eigenvectors :

```
Weights:  [[-1.10998874e+07  1.22151289e+08 -1.25060263e+08  1.44475705
e+08
   1.09898804e+07  1.37663411e+08 -4.22274018e+07  9.39207336e+07
  -1.34346550e+08 -6.29785957e+07]
 [ 2.13179219e+08  1.68595439e+07 -6.09426100e+07 -4.79187327e+08
   4.30521731e+08 -1.73844317e+08  1.02357440e+07  2.59412585e+08
  -3.50949217e+08 -2.32266111e+07]
 [-6.14384280e+08  5.92344731e+07  8.08619257e+08  1.12742365e+08
  -7.13361948e+07 -2.33071887e+07 -8.48130461e+07  2.67194021e+08
  -1.28037289e+08 -9.72362564e+08]
 [-3.66402285e+07  3.70506111e+08  1.13157441e+08 -4.05312653e+08
   1.76715786e+08 -2.68789301e+08 -4.30031694e+08 -3.38550244e+08
   1.03211077e+08 -9.56058682e+07]
 [ 2.24351294e+08 -7.00029304e+07 -9.85726143e+07  2.34686184e+08
   9.02919839e+07  2.29063957e+08  1.56124013e+08  1.95512404e+08
   9.03037251e+07  1.84291226e+08]
 [ 3.26869272e+08 -5.24605810e+07 -1.86711176e+08 -4.68090065e+07
  -9.63755934e+07 -5.72210907e+07  1.17590591e+07 -5.16236278e+08
   5.22946151e+07  4.83958579e+08]
 [ 1.80045428e+08 -2.64715906e+08  5.53691521e+07  3.03157789e+08
   3.58100016e+07 -2.57342036e+08  1.06214570e+08  1.38980073e+08
  -4.15717478e+07 -1.69771840e+08]
 [ 3.45321796e+08 -2.20454543e+08 -1.93885477e+08  1.41418012e+08
  -1.69744922e+08  9.71305573e+06  1.13995966e+08  3.48753650e+07
  -8.75736439e+07  3.28445951e+08]
 [-5.35494162e+08  2.54867159e+07 -2.83496810e+07 -7.91196530e+05
  -3.16835190e+08  1.83851082e+08  1.10500520e+08 -2.19322279e+08
   1.33578895e+08  3.93391415e+08]
 [-9.21484519e+07  1.33958263e+07 -2.83624028e+08 -4.37987185e+06
  -9.00374819e+07  2.20212428e+08  4.82422700e+07  8.42136193e+07
   3.63090136e+08 -6.61416926e+07]]
```

Figure 25 : Projection of k-eigenvectors into eigenspace.

Next to determine if a newly introduced image $x$is a face, we obtain the weight of x from previously calculated normalized and eigenfaces and then calculate the distance between weights of x and the weights of the training images. For final comparison we take the minimum distance and compare it with the threshold which is simply the sum of the weights of x. If the distance is smaller or equal to the threshold magnitude, then the given image x is identified as a face.

When a PCA Face Detector is applied to an image of a face in 3 different lighting conditions, the detector correctly identifies the original image and the lighter image that a face existed within. However, when it is applied to the darker image it fails to identify the face. This could be because the main features of the face such as eyes, hair and mouth are not very apparent due to the lightning.

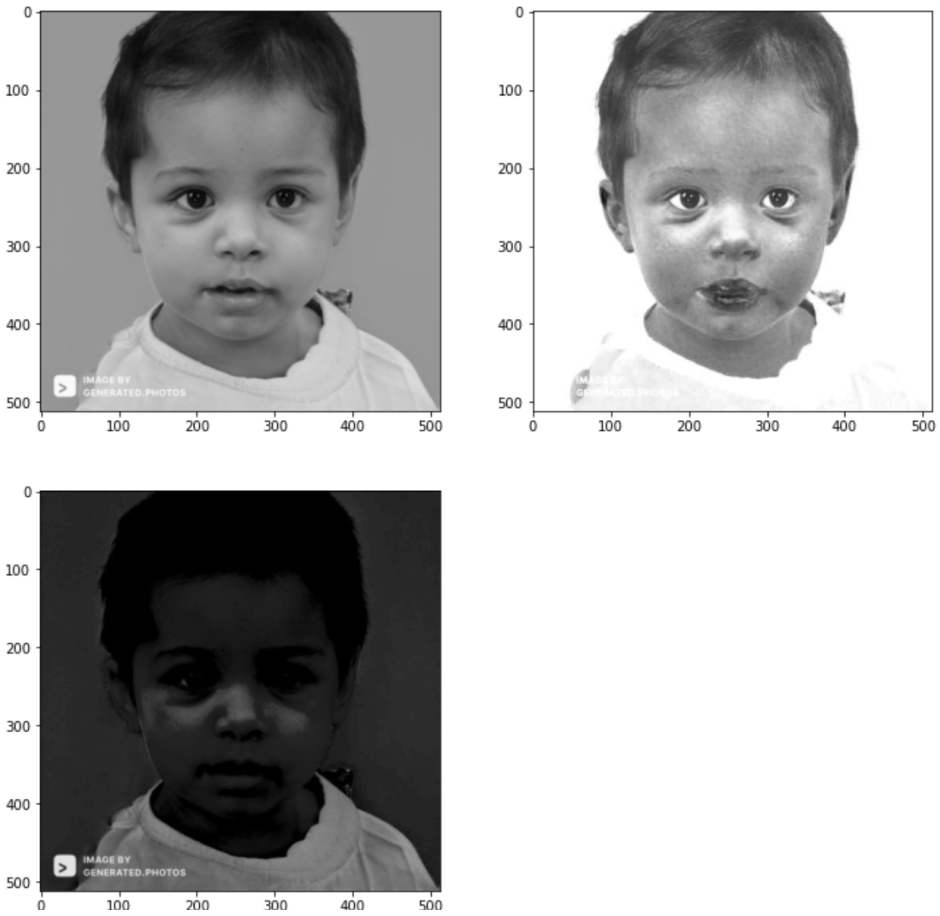


Figure 26 : Image of a face within 3 different lighting conditions

When the PCA Face Detector is applied to two face images with different skin colors, the Face Detector correctly identifies that both images contain a face. The training dataset of the face detector contained diverse faces with different skin tones, facial expressions etc.

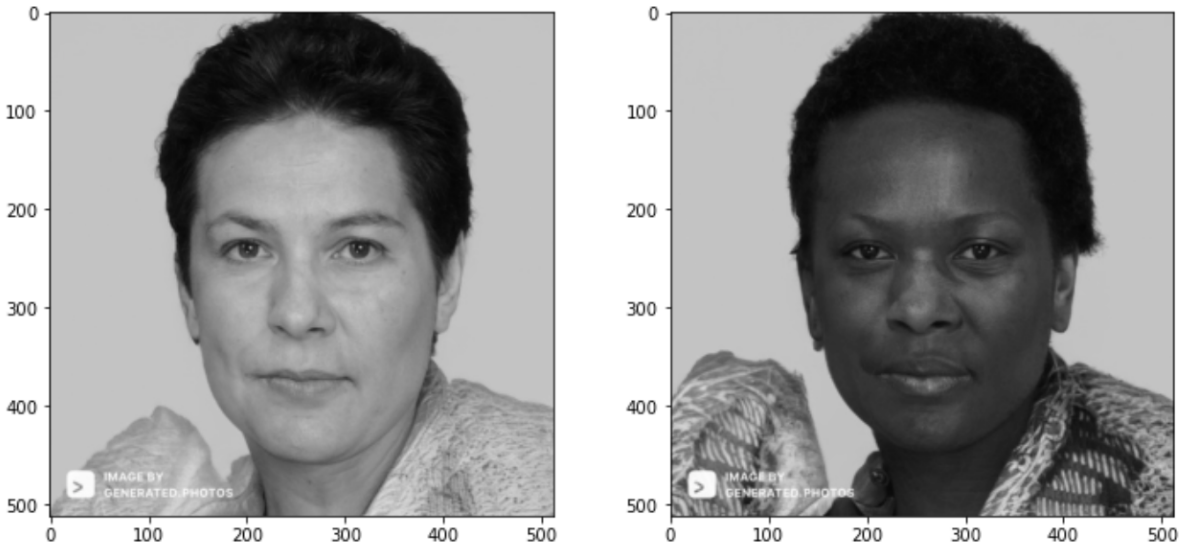


Figure 27 : Two Images of faces with different skin colors

For the case of upside-down and 90 degrees rotated images, PCA Face Detector correctly identifies that the upside down image contains a face but it fails to detect the face in the rotated image. This could be due to the alignment of face features (eyes,nose,mouth,hair) remains vertical in the upside-down image, however it completely changes in the rotated image.

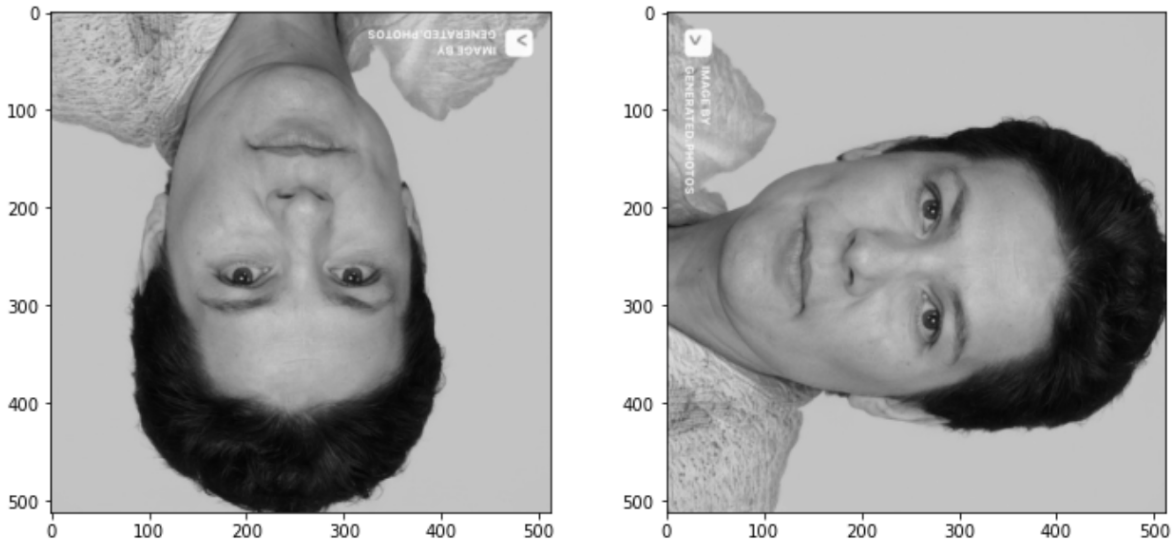


Figure 28 : Upside down and rotated face images

5. For the images which do not contain an actual face but contain face-like components, the PCA Detector failed to detect the face-like components as they were significantly different from the training face images in terms of features. It is also worth noting that the dataset which is used was relatively small, therefore, it can be improved for the detection of face-like components

Figure 29 : Various images that contain face-like components

6. For the case of images which contain occluding faces such as sunglasses, beards, face masks, etc. The PCA face detector identifies that the Image of the Person wearing sunglasses actually contains a face. However, it fails to identify the other two images as faces. This is due to the large differences between the test images and the training images.

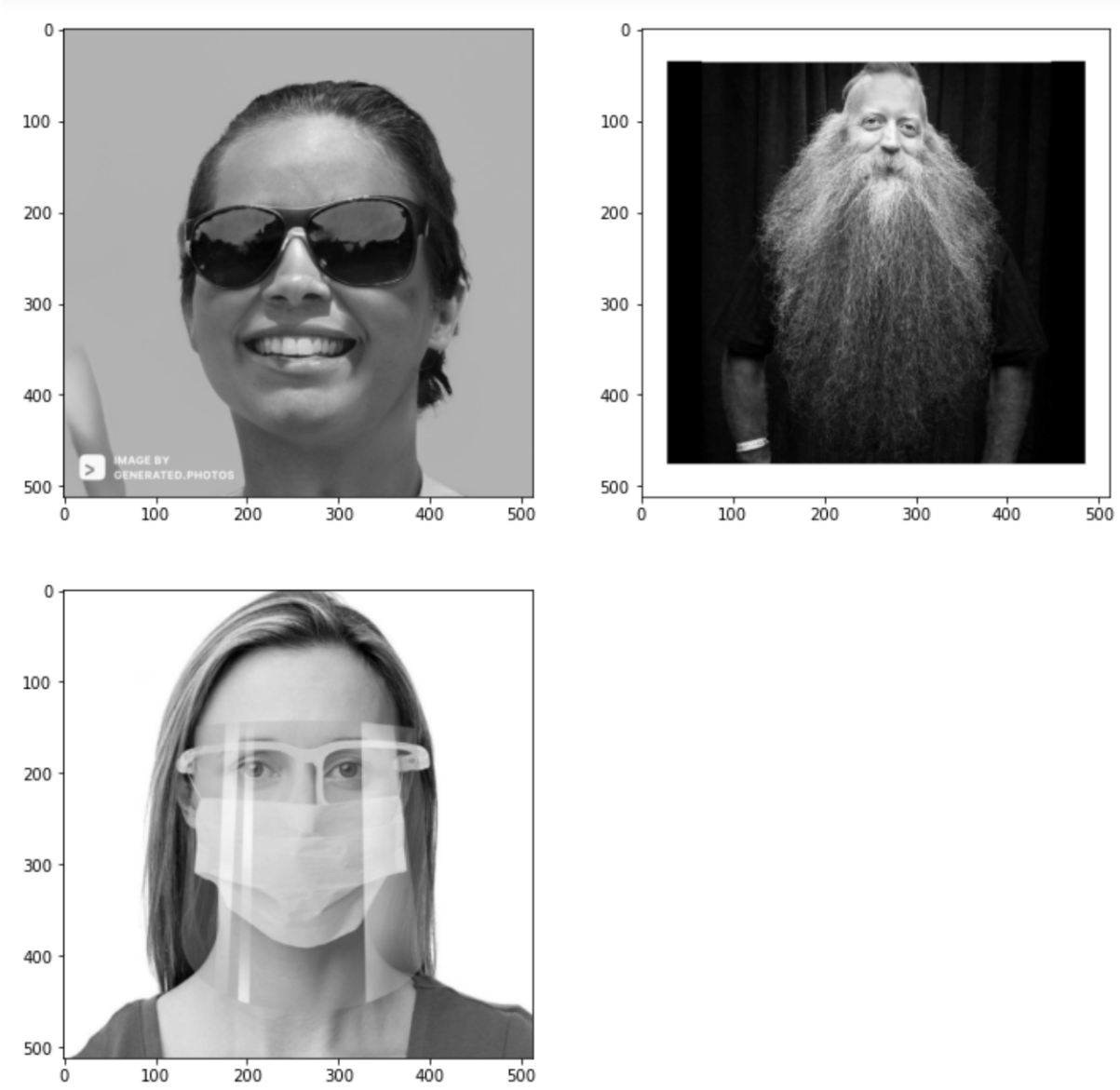


Figure 30 : Images containing images with occluding features

Question 5 : Texture

As seen in Figure 13, there are four images that have different periodic components and different repetition patterns.

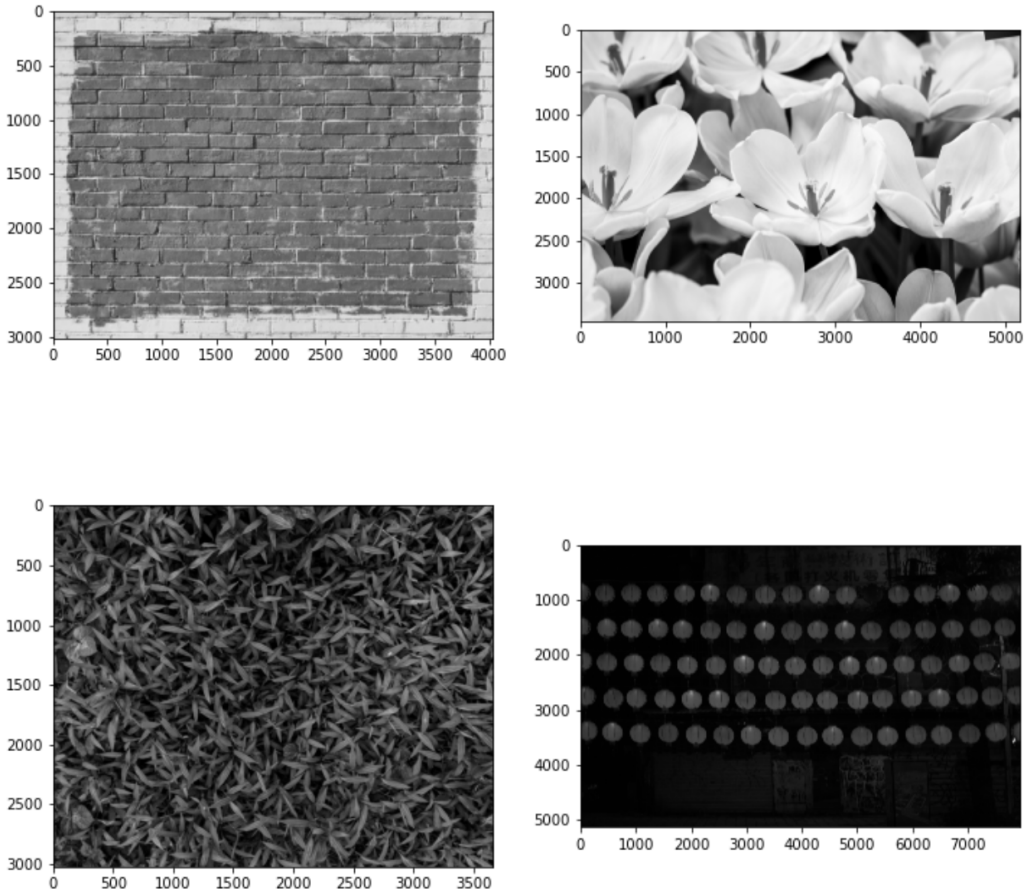


Figure 31 : Images with different periodicity.

In the plotted results of the images transformed into frequency domain, it is obvious that the FFT of the grass image which is placed at row 2 column 1 of the Figure 32, the components are less gathered around the center of the FFT image, and the edges are relatively less dominant compared to the other images. This makes sense, as the pattern is the most frequent in terms of repetition in the grass image.

For the bricks image, whose FFT is placed in row 1 column 1 of Figure 32, it is apparent that most of the frequency components are also gathered in the center but a little bit more scattered, the edges are not very strong but still they are projected in the frequency domain.

It is possible to say that, the frequency component and the pattern repetition is less dominant in bricks image than the grass image.

If we have a closer look at the FFT of the flower image which is placed at row 1 column 2 of Figure 32, we see that the frequencies are still mostly gathered around the center, ie, the low frequency areas Also there are horizontal,vertical and slanted lines which correspond to the strong edge areas. That could be because the edges around the flowers are less in quantity due to less repetition of the patterns.

Lastly for the lamps image, we can infer that the periodicity and the pattern repetition is more dominant in this image compared to the flower and bricks image. We can see in the FFT of that image (row 2, column 2 of Figure 32) that the frequency components are more spread and the edges are also visible but not as strong as the flower image. This could be because the patterns are more frequent in this image.

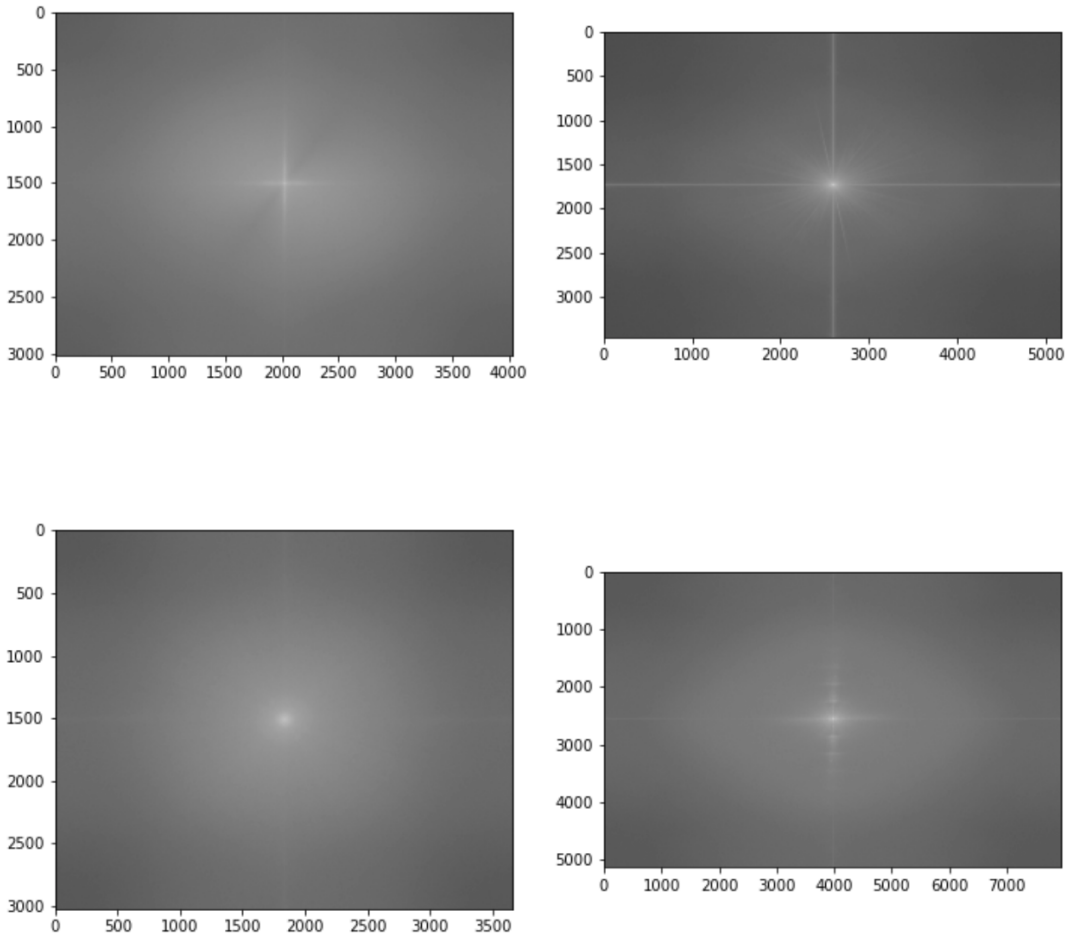


Figure 32 : FFT of the images in Figure 31 in the respective order

Power Spectrum of the Images :

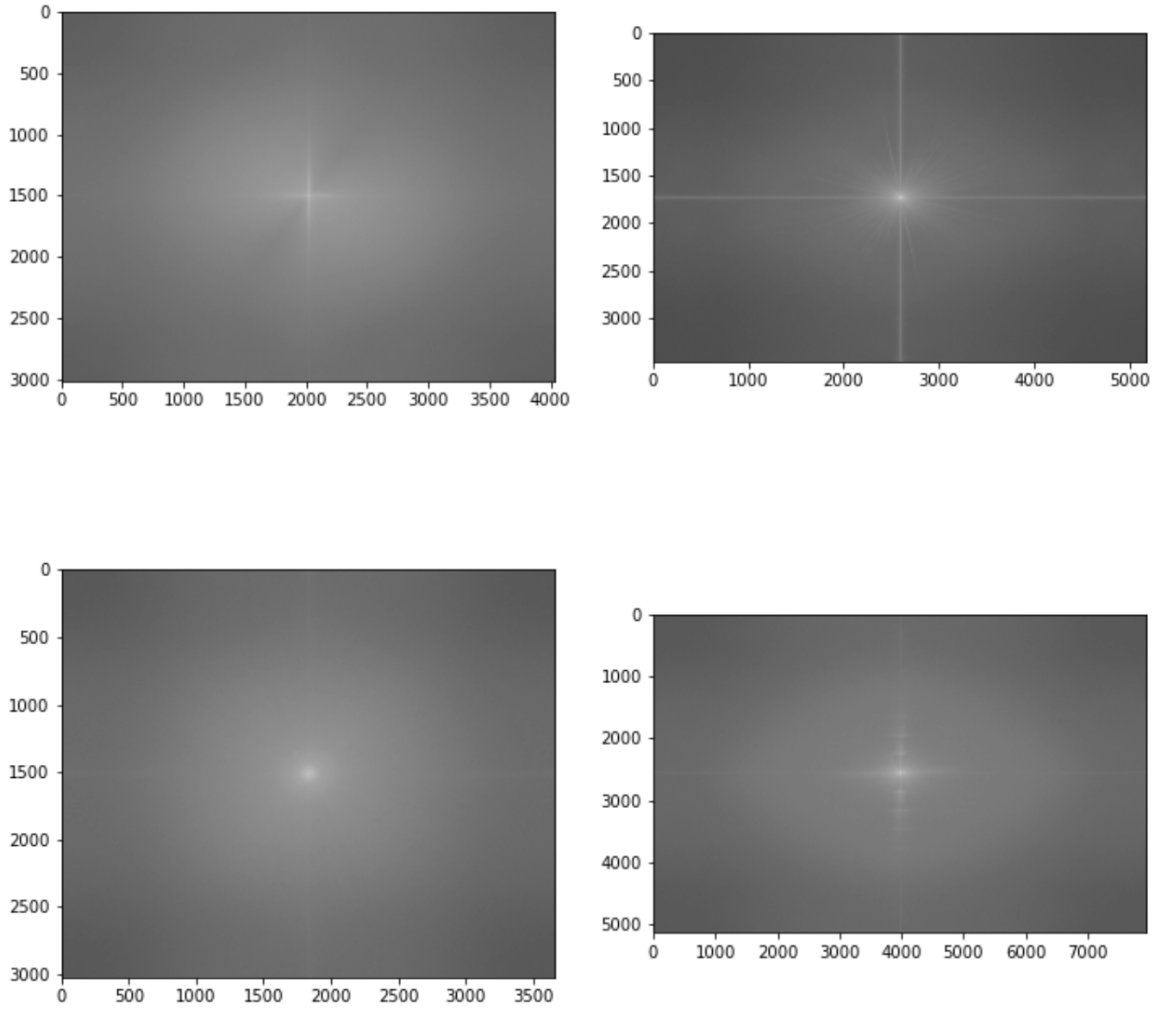


Figure 33 : Power spectrum of the images

3. Spectral Entropy gives us an idea of how regular an image is. If we observe that power spectrum entropies of each image displayed in Figure 34, we can see that the flower image has the greatest entropy value, which shows that this image is the most regular among all images, which also makes sense as it has less frequency of patterns.

```
Power spectrum entropy of bricks image :  -1.5209776123688261
Power spectrum entropy of flower image :  -1.191841225482021
Power spectrum entropy of grass image :  -5.919871444655877
Power spectrum entropy of lamps image :  -7.787276445089702
```

Figure 34 : Power Spectrum Entropy of the images