Digital Image Processing

Shervin Halat

98131018

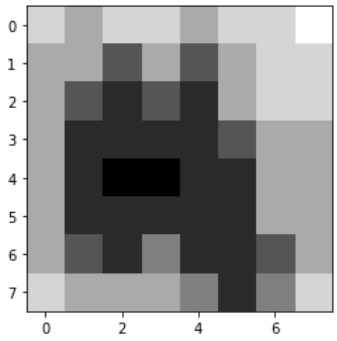
Homework2

1.

a.

The only operation needed is subtract values from 7:

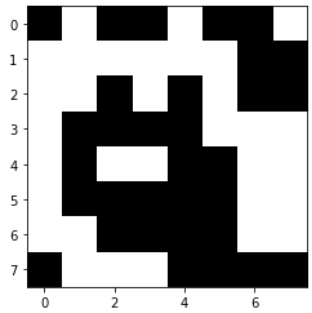
```
[5 4 5 5 4 5 5 6]
[4 4 2 4 2 4 5 5]
[4 2 1 2 1 4 5 5]
[4 1 1 1 1 2 4 4]
[4 1 0 0 1 1 4 4]
[4 1 1 1 1 1 4 4]
[4 2 1 3 1 1 2 4]
[5 4 4 4 3 1 3 5]
```



b.

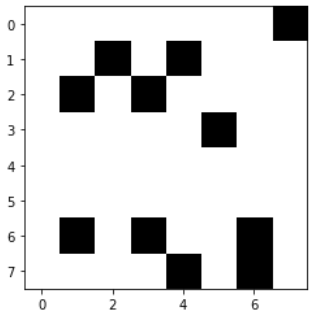
bit-plane (0):

```
[0 1 0 0 1 0 0 1]
[1 1 1 1 1 1 0 0]
[1 1 0 1 0 1 0 0]
[1 0 0 0 0 1 1 1]
[1 0 1 1 0 0 1 1]
[1 0 0 0 0 0 1 1]
[1 1 0 0 0 0 1 1]
[0 1 1 1 0 0 0 0]
```



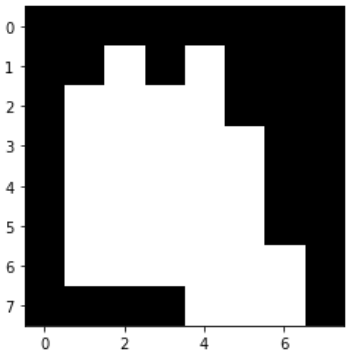
bit-plane (1):

```
[1 1 1 1 1 1 1 0]
[1 1 0 1 0 1 1 1]
[1 0 1 0 1 1 1 1]
[1 1 1 1 1 0 1 1]
[1 1 1 1 1 1 1 1]
[1 1 1 1 1 1 1 1]
[1 0 1 0 1 1 0 1]
[1 1 1 1 0 1 0 1]
```



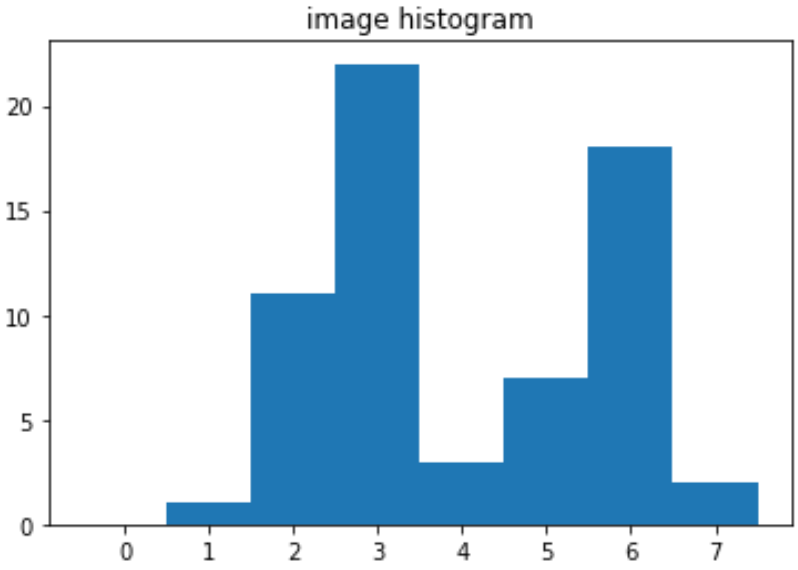
bit-plane (2):

```
[0 0 0 0 0 0 0 0]
[0 0 1 0 1 0 0 0]
[0 1 1 1 1 0 0 0]
[0 1 1 1 1 1 0 0]
[0 1 1 1 1 1 0 0]
[0 1 1 1 1 1 0 0]
[0 1 1 1 1 1 1 0]
[0 0 0 0 1 1 1 0]
```



c.

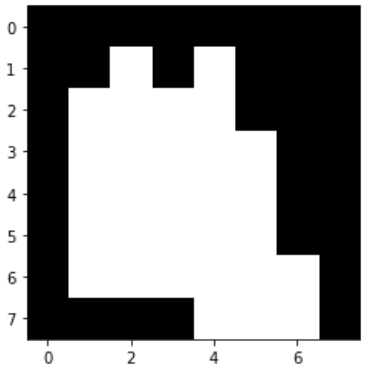
computed by counting each pixel value from 0 to 7:



d.

The ‘valley’ is on the pixel value of ‘4’. Applying threshold on this value of ‘3.5’ leads to the following values:

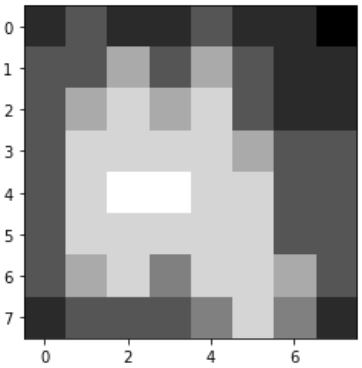
```
[0 0 0 0 0 0 0 0]
[0 0 1 0 1 0 0 0]
[0 1 1 1 1 0 0 0]
[0 1 1 1 1 1 0 0]
[0 1 1 1 1 1 0 0]
[0 1 1 1 1 1 0 0]
[0 1 1 1 1 1 1 0]
[0 0 0 0 1 1 1 0]
```



e.

Dividing pixel values by ‘8’ then multiplying them by ‘256’ results in:

```
[ 64  96  64  64  96  64  64  32]
[ 96  96 160  96 160  96  64  64]
[ 96 160 192 160 192  96  64  64]
[ 96 192 192 192 192 160  96  96]
[ 96 192 224 224 192 192  96  96]
[ 96 192 192 192 192 192  96  96]
[ 96 160 192 128 192 192 160  96]
[ 64  96  96  96 128 192 128  64]
```



f.

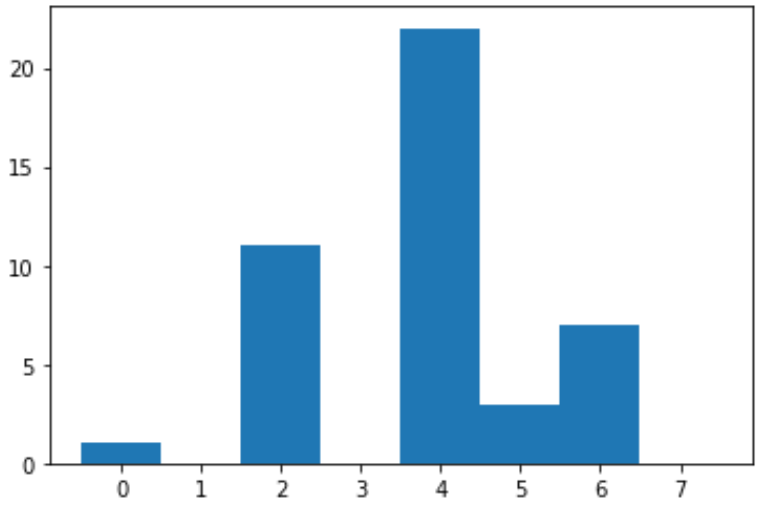
probabilities of pixel values corresponding to histogram are as follows:

```
[0.   , 0.125, 1.375, 2.75 , 0.375, 0.875, 2.25 , 0.25 ]
```

Therefore, S_k values are as follows:

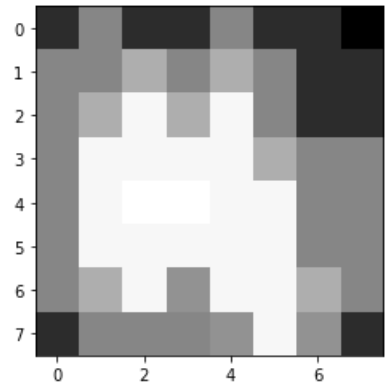
```
[0.0, 0.015625, 0.1875, 0.53125, 0.578125, 0.6875, 0.96875, 1.0]
```

Hence, the histogram would be as follows:



And the resultant matrix would be:

```
[1.5   4.25  1.5   1.5   4.25  1.5   1.5   0.125]
[4.25  4.25  5.5   4.25  5.5   4.25  1.5   1.5  ]
[4.25  5.5   7.75  5.5   7.75  4.25  1.5   1.5  ]
[4.25  7.75  7.75  7.75  7.75  5.5   4.25  4.25 ]
[4.25  7.75  8.    8.    7.75  7.75  4.25  4.25 ]
[4.25  7.75  7.75  7.75  7.75  7.75  4.25  4.25 ]
[4.25  5.5   7.75  4.625 7.75  7.75  5.5   4.25 ]
[1.5   4.25  4.25  4.25  4.625 7.75  4.625 1.5  ]
```



g.

???

h.

???

2.

Convolution:

a.

```
[-14  42  42  -7  -6  11   5  11]
[-21  32 -34 -16 -15   2  32   5]
[-21  39   0  18   8   1  -2  11]
[-14  39 -27 -19 -22  38 -22  -9]
[-10 -24  42 -20   8   2   8  -6]
[ 17 -31  42 -30   9   0   6 -12]
[ 47 -30  46 -20   8   2  -2  11]
[ 14   7 -10 -10 -13  47 -23  50]
```

b.

```
[ 14   7 -14  -7   6  10   0 -10]
[ 21  10 -18  -7   6  10   0 -13]
[ 21   3 -18   0  10   7  -7 -10]
[ 14  10 -11  -4  10   0 -10  -6]
[  7  11  -7  -8  13   0 -13  -6]
[  0  11   0 -12   9   0  -6  -9]
[  3   1  -3  -8  13   0  -3  -6]
[  3  -3  -3  -4  10   0   0  -3]
```

c.

```
[ 14  14  -7  -7   6  13  10   0]
[ 14   3 -15  -1   9  10   0 -10]
[ 14   3 -11   0   7   4 -10 -13]
[  7   0 -10  -3   7   0 -10  -6]
[  3   7   0  -1   9  -1  -7  -3]
[  7  11   0  -8   6   0  -3  -3]
[  3   0  -7 -11   7   1   1   1]
[ -4 -10 -10  -7   4  -6  -6  -6]
```

d.

```
[ -7  14  21  -7  -3   6  -1   6]
[ -7  14 -17  -3  -6  -1  16  -1]
[ -7  11   2   6   6  -4  -1   6]
[ -7  21 -17  -3 -13  22 -13  -3]
[ -3 -14  21 -10   6  -4   6  -3]
[  5 -10  14 -10   3   0   3  -6]
[ 22 -17  21 -10   6  -4   3   2]
[  2   9 -10   0 -10  25 -17  25]
```

e.

```
[14 21 21  7  6 16 22 16]
[21 31 34 16 15 25 31 22]
[21 24 27  9 19 26 29 16]
[14 24 27 19 22 25 22  9]
[10 24 21 20 19 25 19  6]
[10 31 21 30 18 27 21 12]
[16 30 17 20 19 25 29 16]
[13 20 10 10 13 16 23 13]
```

f.

```
[  7  21  21  14   3   2   5   2]
[  0 -10 -13 -16  -6   5   8   8]
[  0   6  12  18   8   1  -2   2]
[  7  -3  -6 -19  -1  -4  -1  -9]
[-10  -3   0  10  -1   2  -1   3]
[ -1  -1   0   0   0   0  -3  -3]
[  5  12   4  10  -1   2  -2   2]
[  5  -2  -1 -10   8   5  19   8]
```

g.

```
[ -7   0  14   7  -3   0  10  13]
[-14  -7  10   6   0  -7   0  10]
[-14  -3  11   0  -3  -3  -3   0]
[-14 -10   8   1 -10   0   7   0]
[ -4 -11   7  11  -7  -1   9   6]
[  7  -4   0   8  -6   0   6   9]
[  0  -1  -4   1  -9   1   4  10]
[-10  -4  -4  -3 -13  -6  -6  -3]
```

h.

```
[-14  21   0  -7  -6   2   8   2]
[-21  32  -4  -7  -6   2   8   5]
[-21  39 -18   0 -10  13   1   2]
[-14  18   3 -10  -4  14  -4   0]
[ -1  -3  21 -20  -1  14  -1  -6]
[ 20 -31  42 -30   9   0   6  -3]
[ 23 -21  25 -20  -1  14 -11  14]
[ 17 -11  11 -10  -4  14 -14  17]
```

Deconvolution:

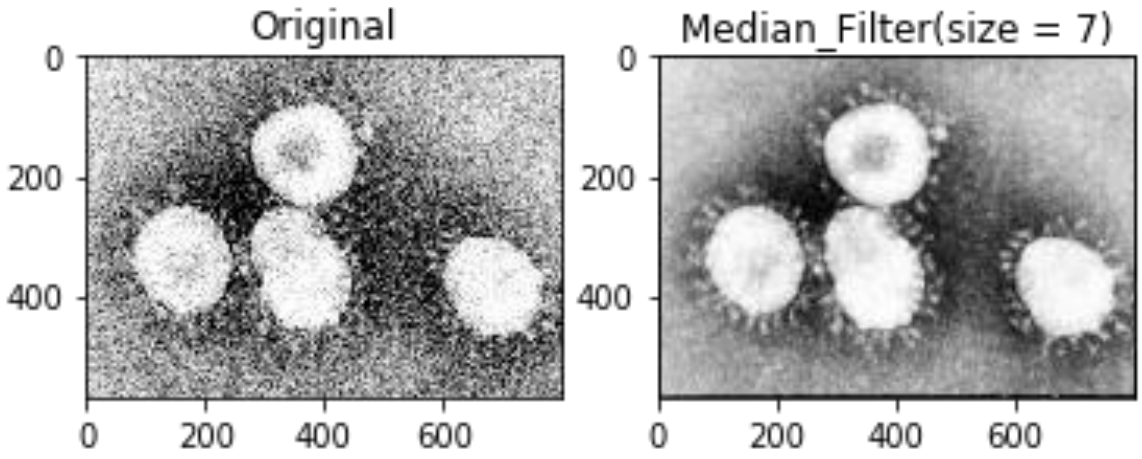
The main approach here is to see each value of the intended filter as a variable and consider the whole process as a system of linear equations. Now, we just need to find appropriate equations by considering some pixels of filtered image and its corresponding pixel on main image. The tip here is to find the mentioned pixel wisely so that the least number of variables appear in each equation. Therefore, the results are as follows:

3.

a.

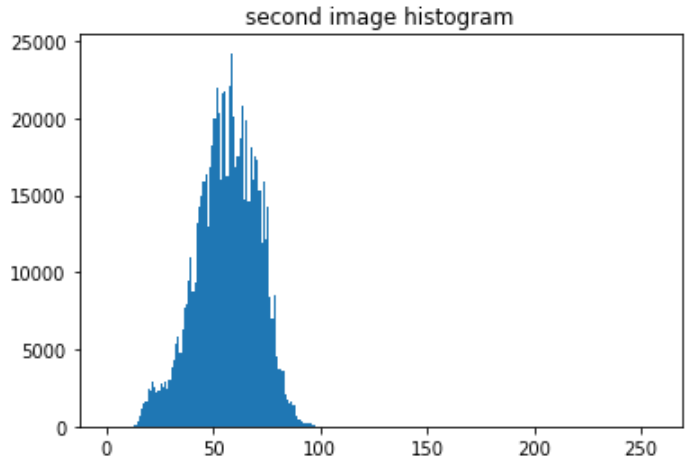
1st image:

Due to salt and pepper noises Median Filter was implemented:

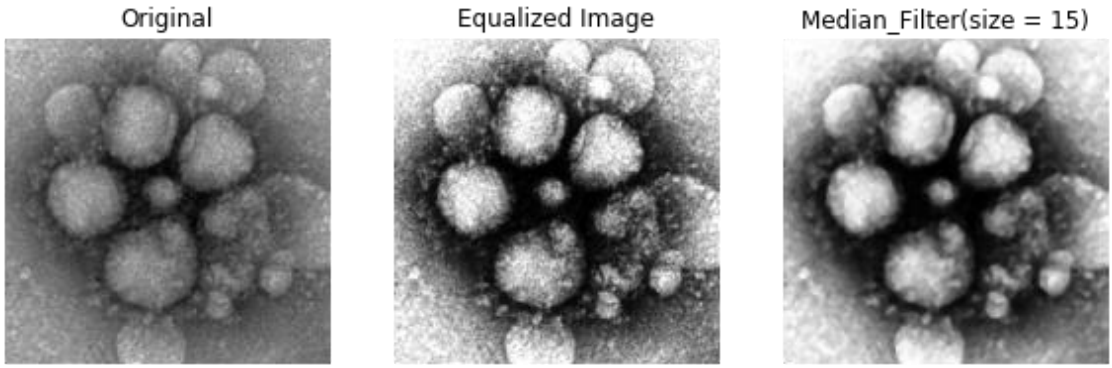


2nd image:

Considering histogram of second image, which is as follows, the image is low-contrast:

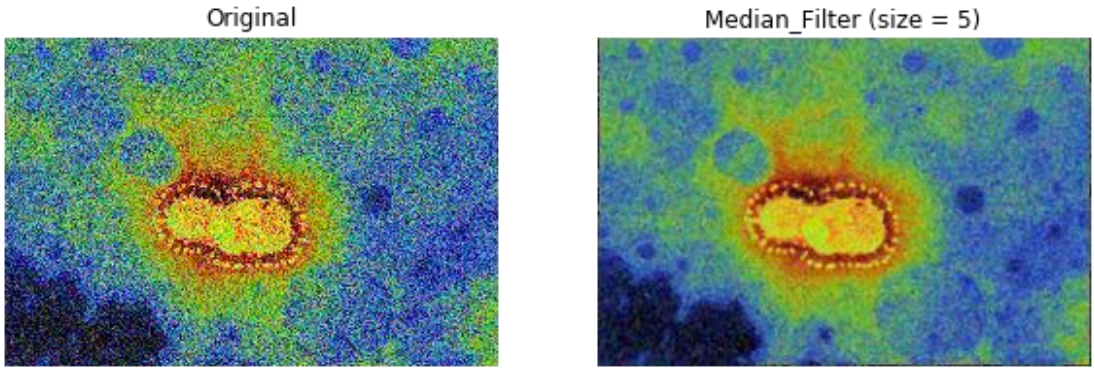


Therefore, at first the histogram of image is equalized, then, to overcome salt and pepper noises median filter of size 15 was implemented:



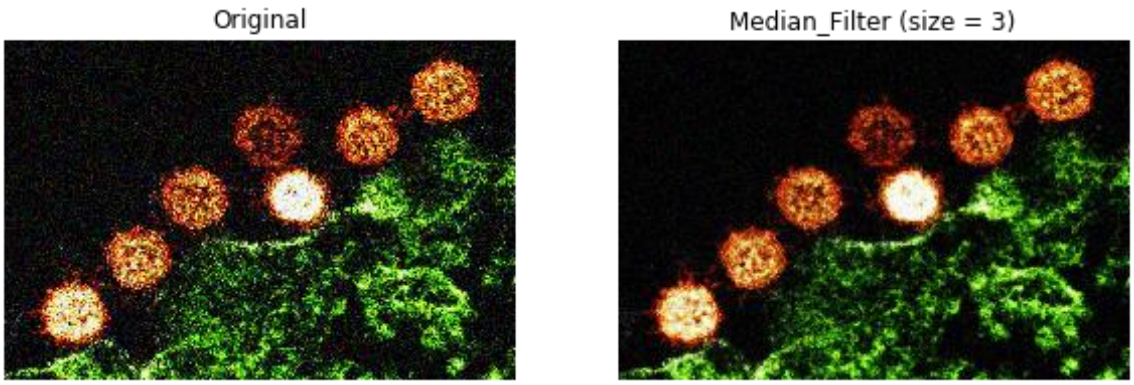
3rd image:

Since this image is RGB, denoising was implemented on each channel separately. Since this image noise is salt and pepper as well as previous images, median filter was implemented:



4th image:

The fourth image has salt and pepper noises as well therefore by implementing median filter we have:



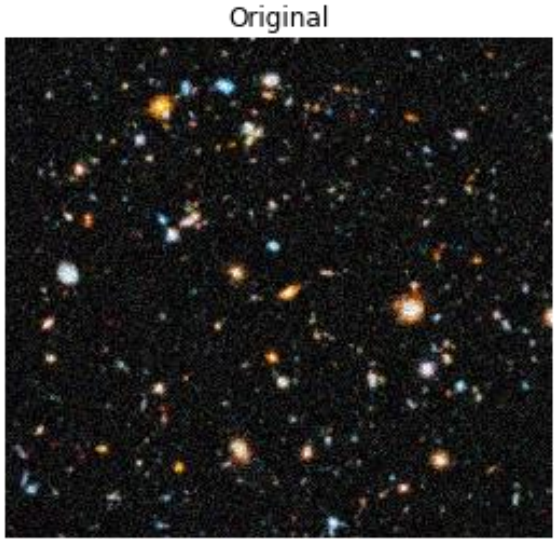
b.

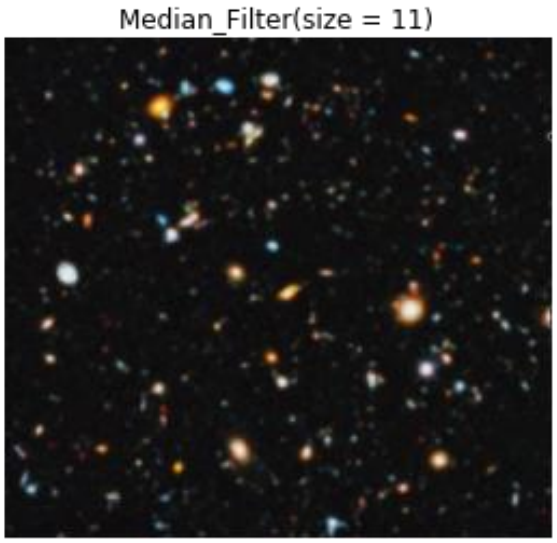
No meaningful degradation of images was found!

c.

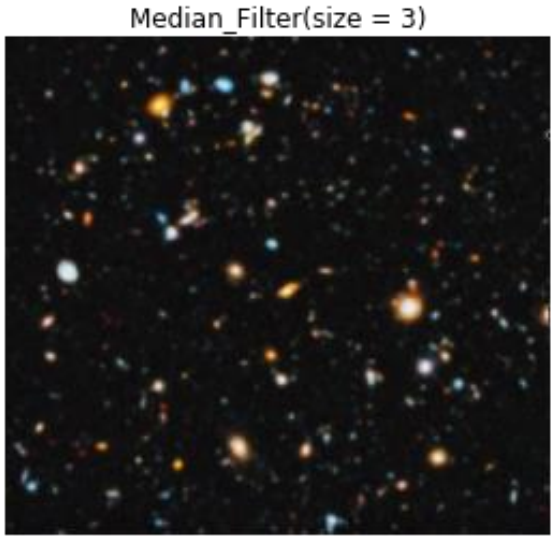
1st image:

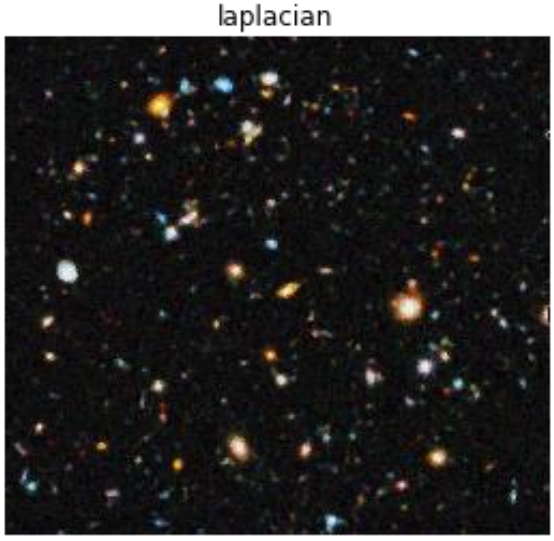
In order to overcome salt and pepper noises, median filter of size 11 was applied in the first place:





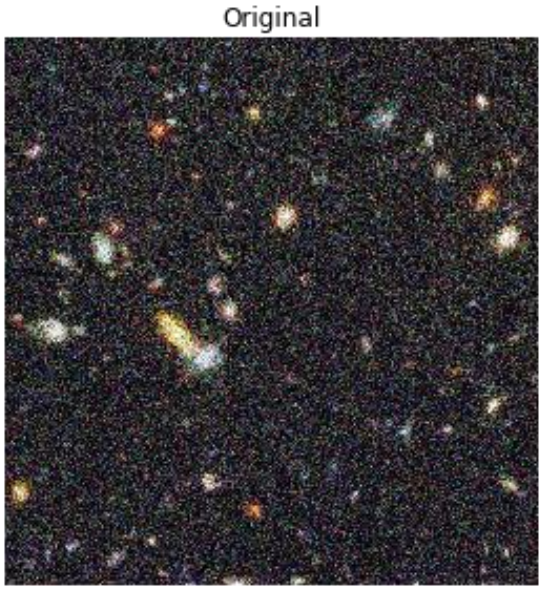
Next, to overcome the blur of edges due to median filter, the obtained image will be added to its Laplacian mask which results in:

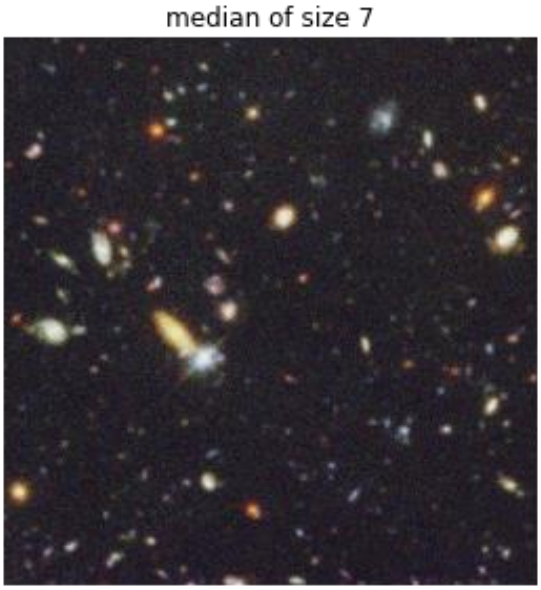




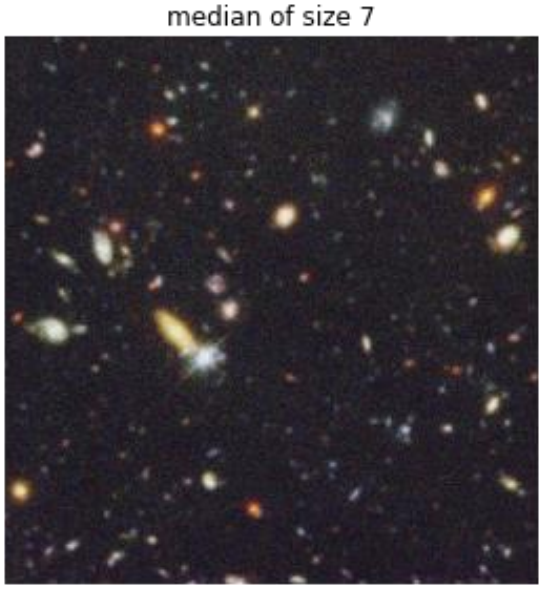
2nd image:

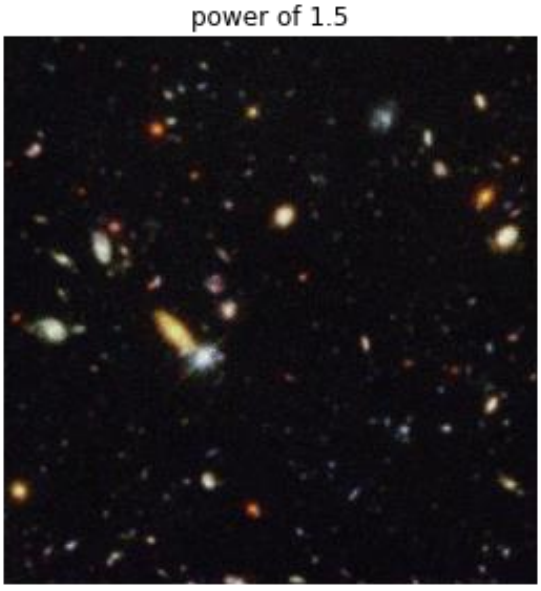
Again, In order to overcome salt and pepper noises, median filter of size 7 was applied in the first place:





Then, to enhance the obtained image, specifically decreasing the slight bright noises on the black background of the image, n^{th} power transformation was applied to the image:



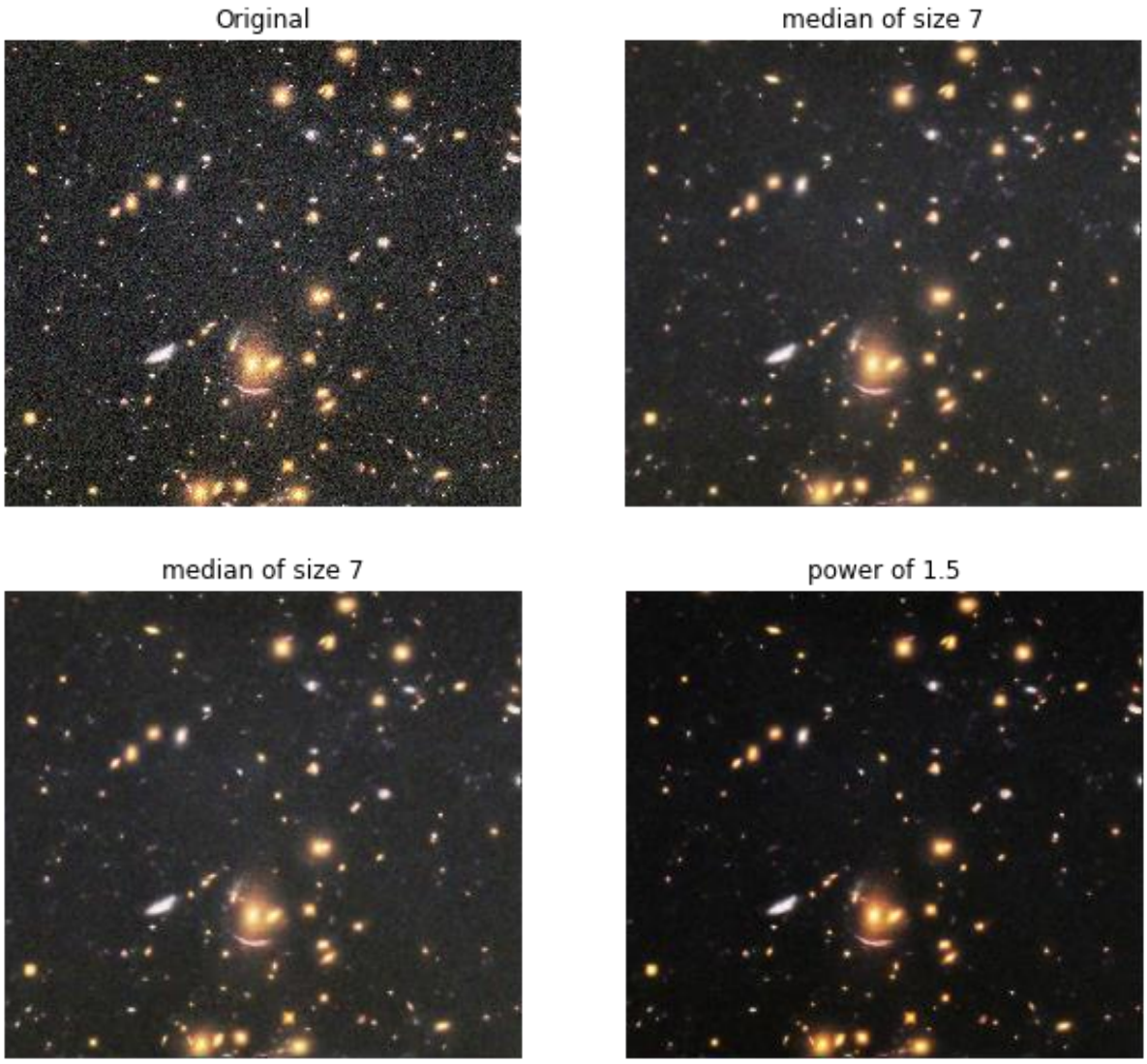


3rd image:

Applying n^{th} power transformation (n = 1.9) for decreasing the brightness of the center:

4th image:

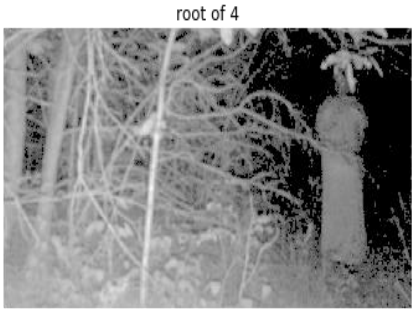
Due to similar types of degradation of this image to the second image of this part, the exact same steps were taken:

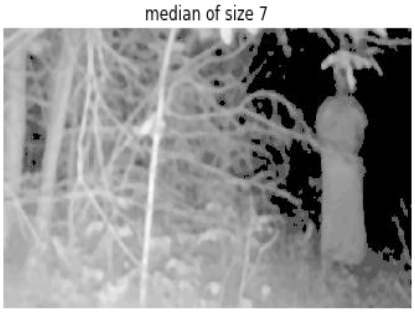


d.

1st image:

To reveal the ghost root transformation of 4 was applied to the image then a median filter of size 7 applied to denoise the image.





2nd image:

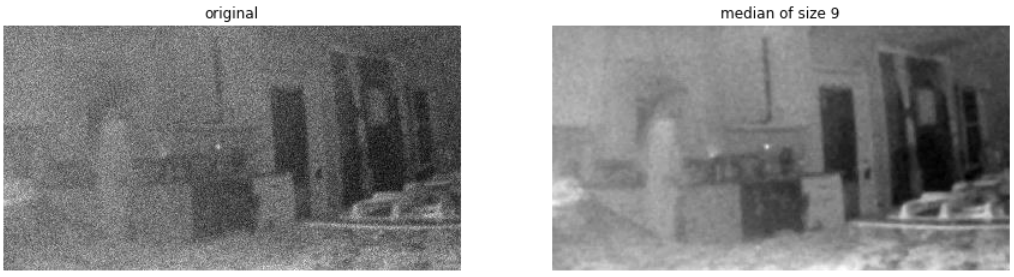
The n^{th} power transformation of 1.99 was applied to lower the brightness of the image:

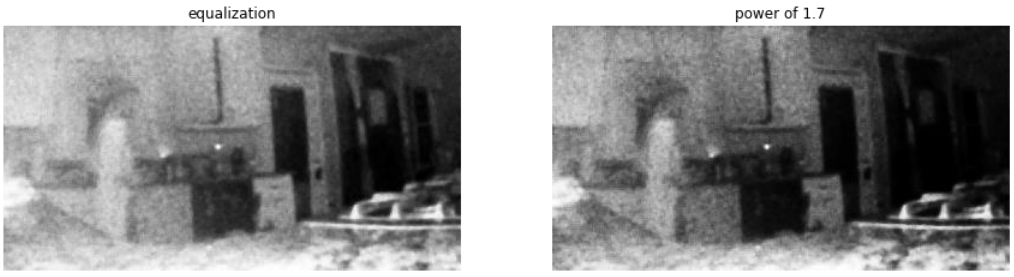
3rd image:

Firstly, equalization was applied to enhance contrast, afterwards, root of 2 transformation was applied to brighten too dark pixels:

4th image:

Steps which were taken are captioned above each image:

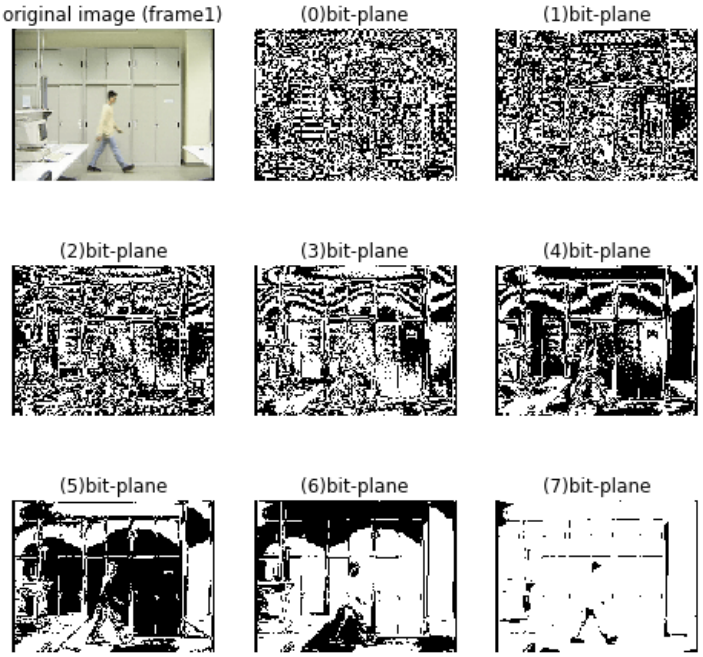




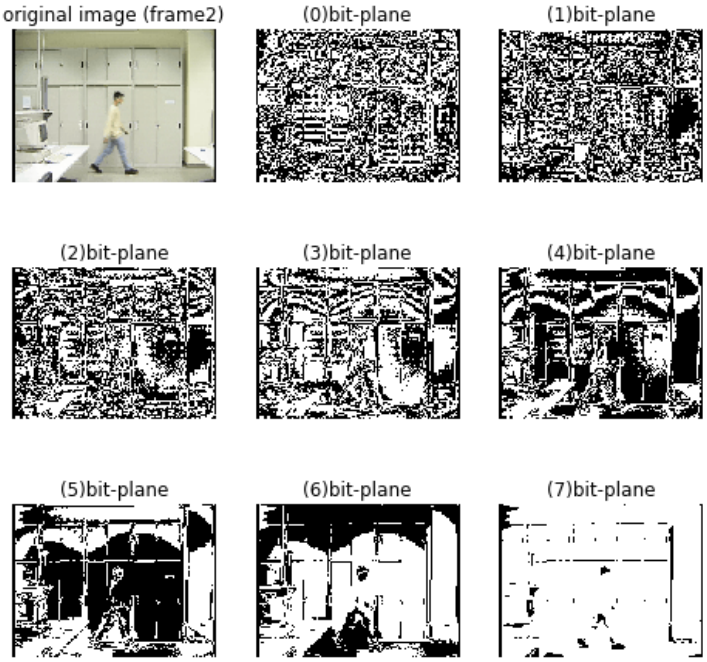
4.

a.

bit-plane slices of ‘frame1’:

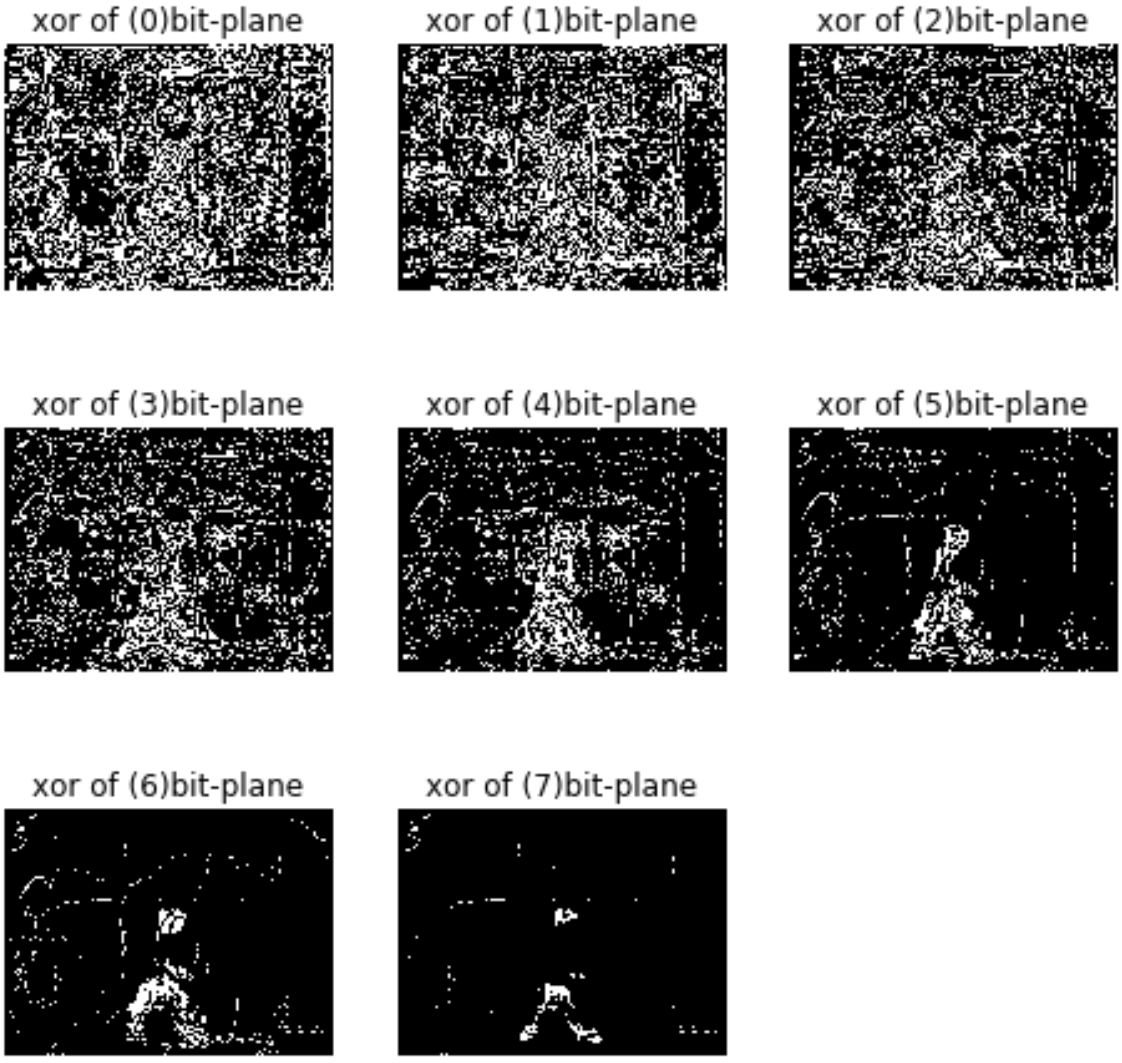


bit-plane slices of ‘frame2’:



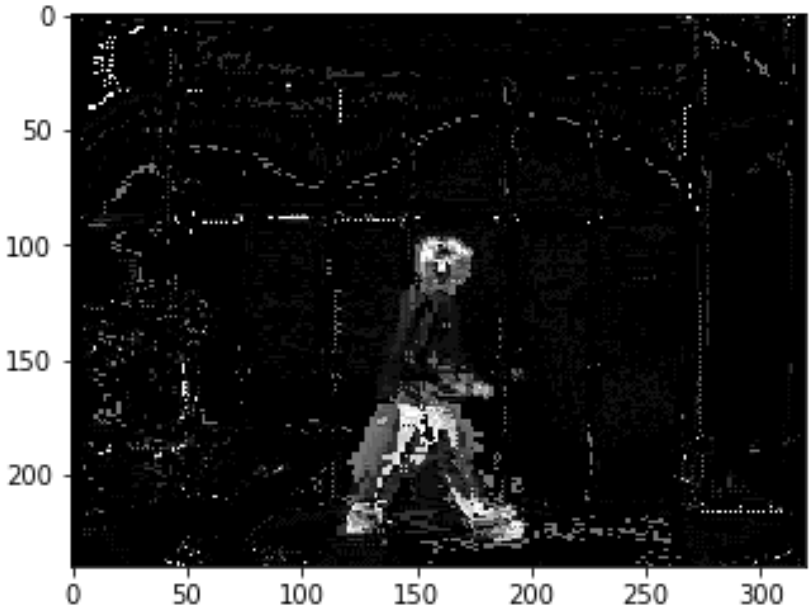
b.

XOR of corresponding bit-plane slices of ‘frame1’ and ‘frame2’:



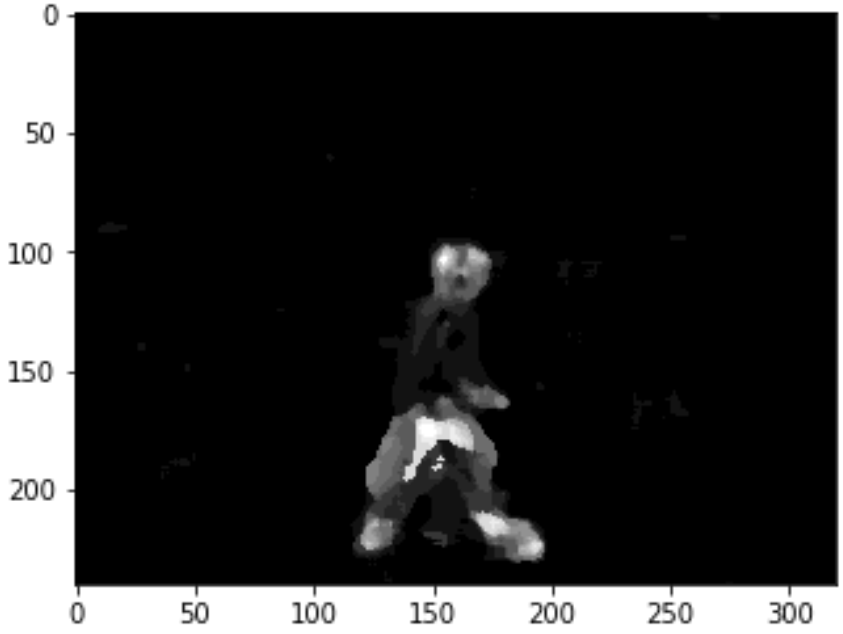
c.

Sum of valued 4 highest bit-planes:



d.

Applying median filter (of size 7) results in:

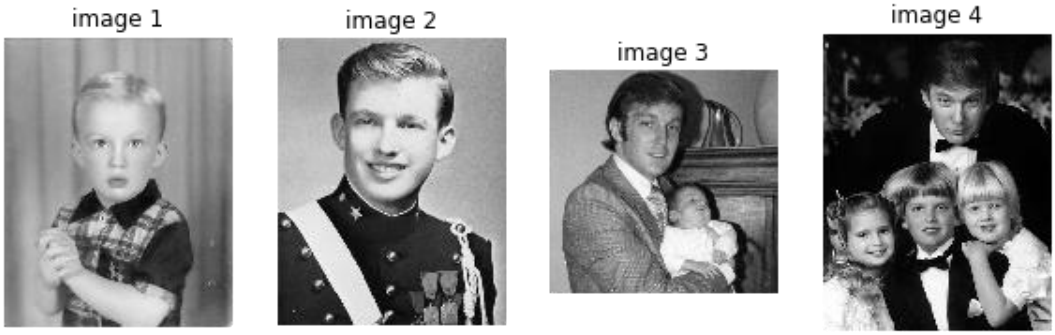


5.

|.

a.

Grayscale conversion:



b.

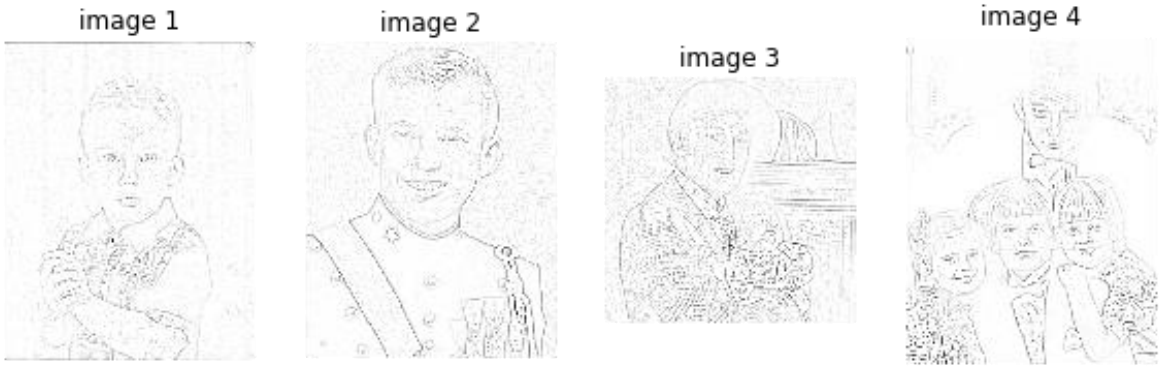
Finding image negative:

c.

Blurring the image:

d.

Color dodging



e.

Smoothing kernel size:

Following images are obtained by applying median filter of size 3:

As the size of median filter increases some details will be disappear but some lines become more obvious and clearer. Also, noises get bigger in size by increasing filter sizes! Following images are obtained by filter of size 7 and 9 respectively:

||.

f.

Grayscale conversion:

g.

Convolution operation:

(applying mean filter of size 5)

h.

Most frequent local pixels:

(Mask size of 3)

i.

Updating intensity values:

j.

Applying median filter of size 5:

k.

Although applying median filter in the first place improves variety of colors of the resultant image (which is obvious in the following image) but it decreases its smoothness therefore seems more amateur oil painting:

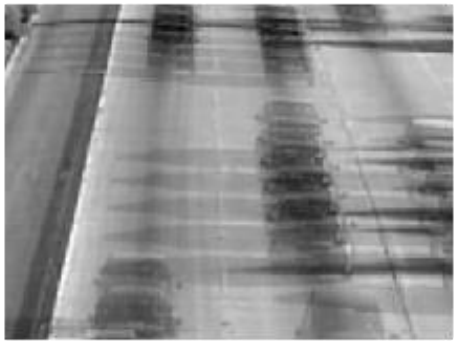
Also using filters of bigger sizes leads to decrease in some details for example elimination of Golf stick!

6.

Mean Operation:

a.

average of first 5 images



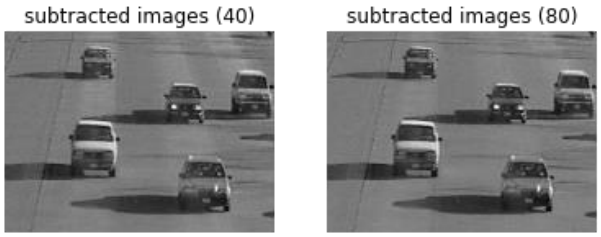
average of first 10 images

average of first 20 images

average of first 40 images

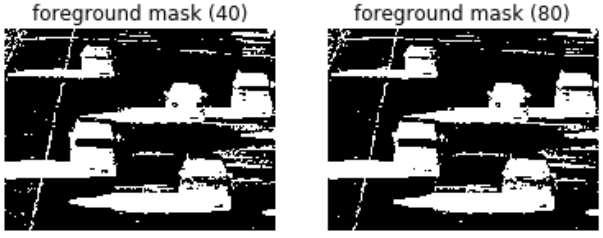
average of first 80 images

b.

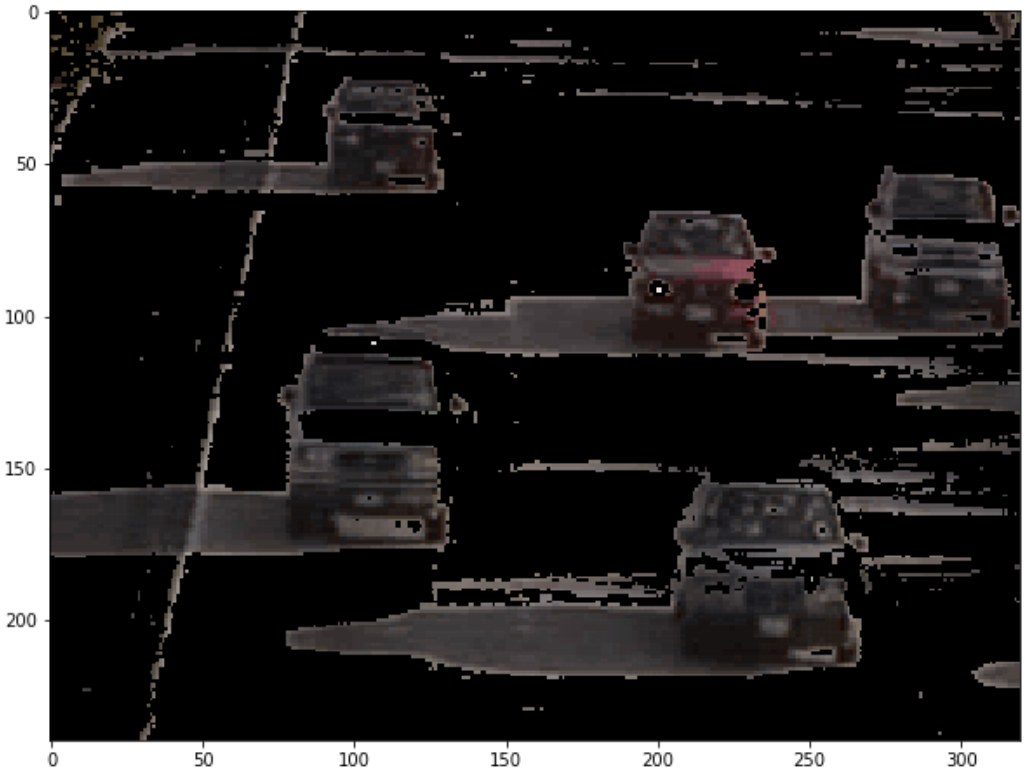


c.

applying binary threshold (130)



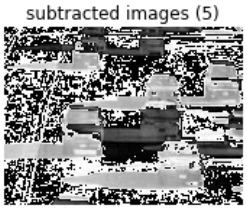
d.

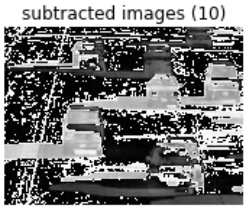


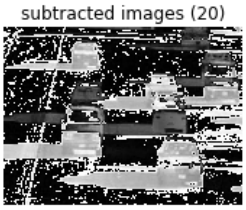
e.

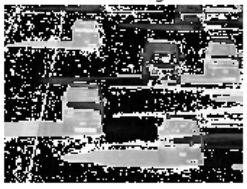
Median Operation:

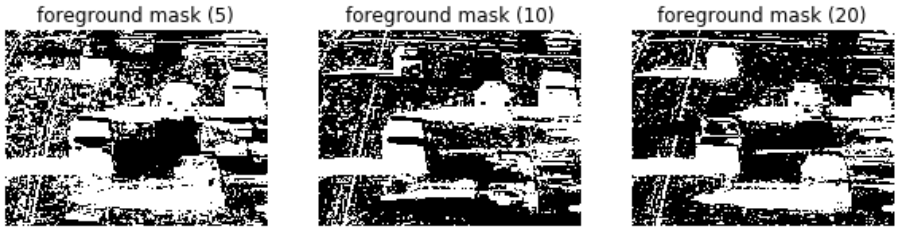
(all steps are as follows:)

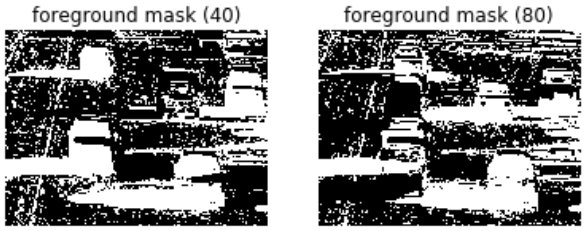


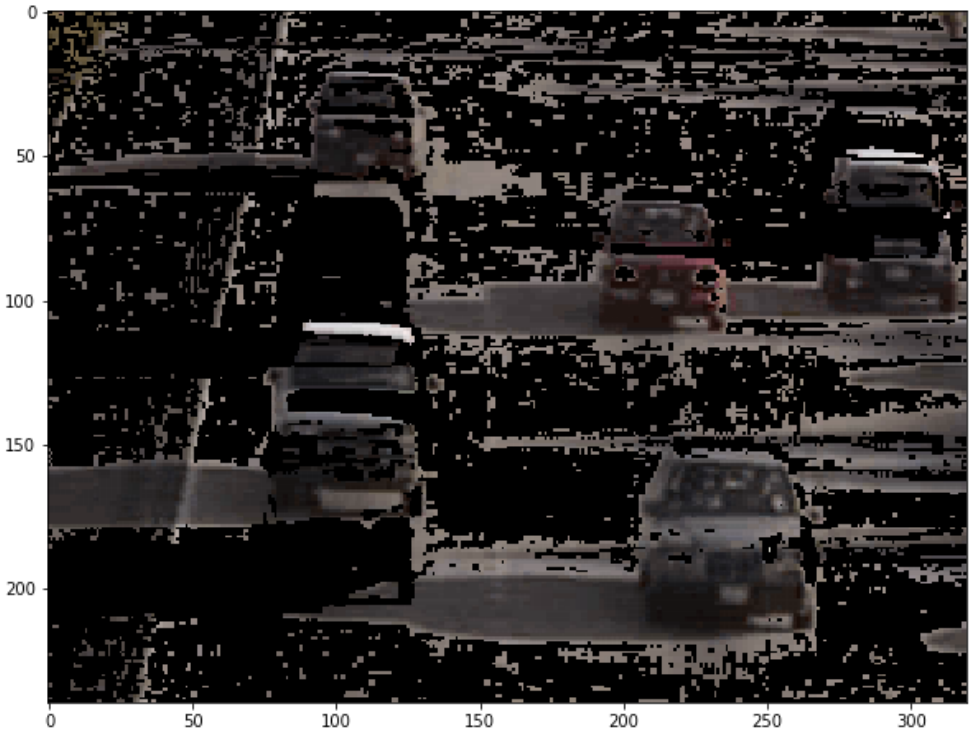







foreground mask (5)
foreground mask (10)
foreground mask (20)


foreground mask (40)
foreground mask (80)


0
50
100
150
200
0
50
100
150
200
250
300

7.

a.

Implementation of ‘bilateral_filter’ function:

b.

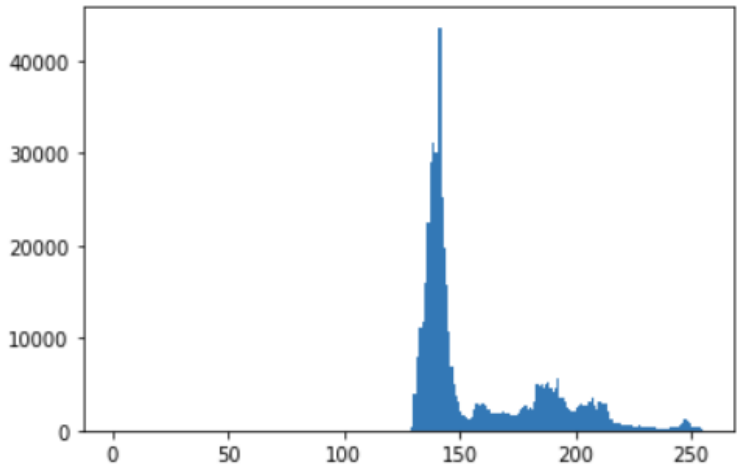
Not performed due to high runtime of bilateral_filter function!!!

c.

For Trump's image:

Considering the following histogram of Trump's grayscale image then considering each peak of the histogram as the center of two consecutive thresholds, leads to the following thresholds:

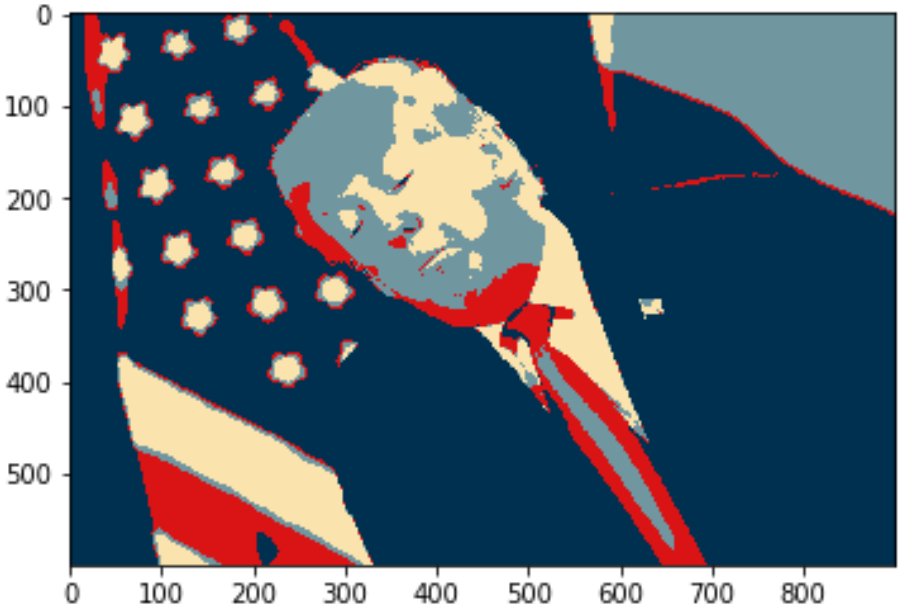
```
plt.hist(gray_trump.ravel(),256,[0,256])
plt.show()
```



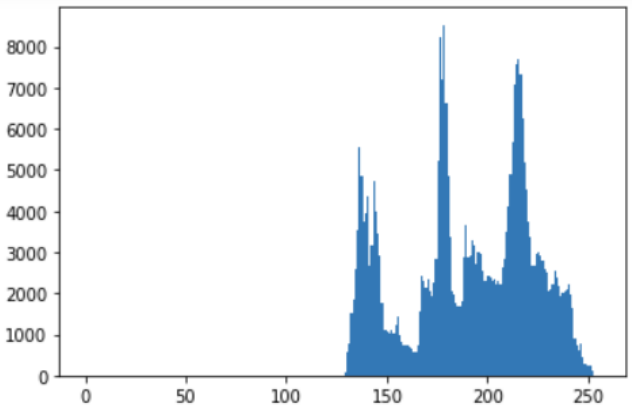

```
x , y = gray_trump.shape
color_trump = np.zeros((x,y,3))
#gray_trump = np.array(img)
for i in range(x):
    for j in range(y):
        if gray_trump[i,j]<= 155:
            color_trump[i,j] = [0,48,80]
        elif 155<gray_trump[i,j]<= 175:
            color_trump[i,j] = [218,20,21]
        elif 175<gray_trump[i,j]<= 200:
            color_trump[i,j] = [112,150,160]
        elif 200<gray_trump[i,j]<= 255:
            color_trump[i,j] = [250,227,173]

```

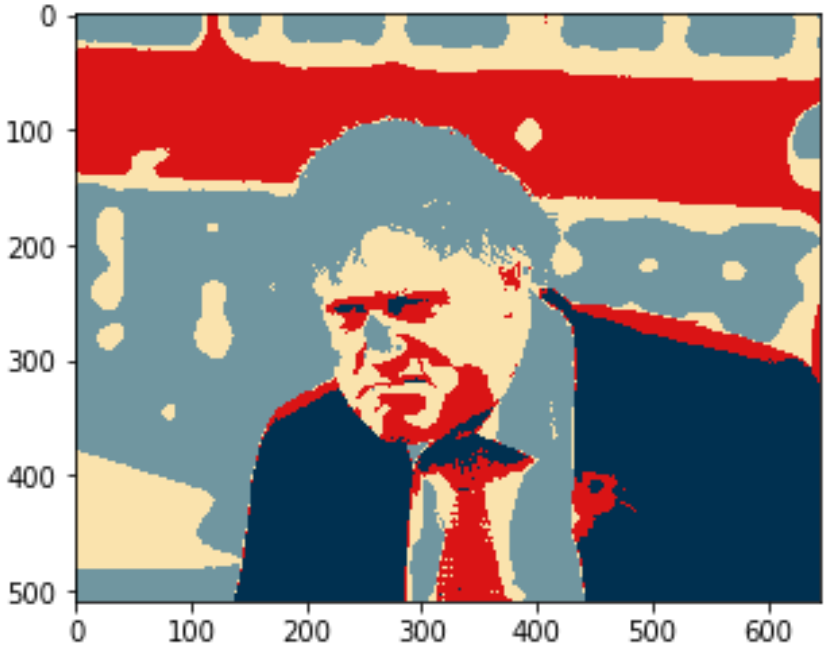
The obtained image by the thresholds defined above is:



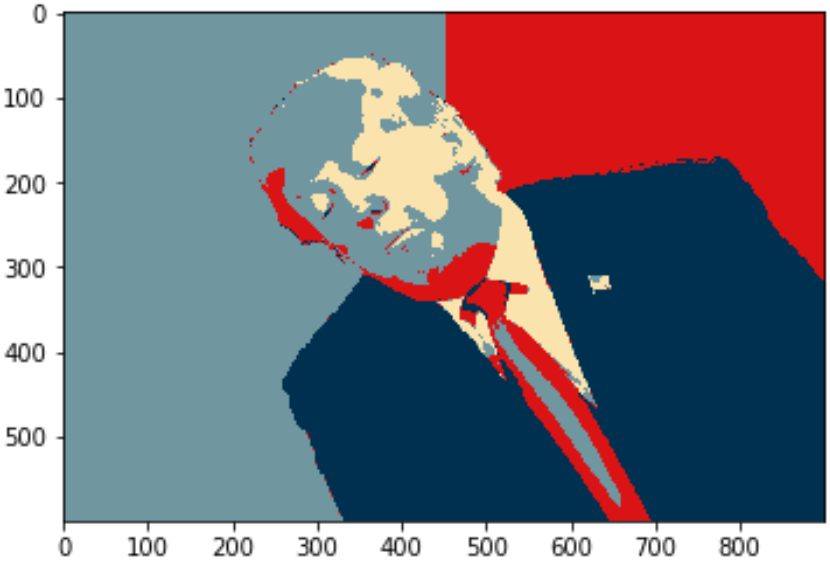
Similarly, for Johnson's image:

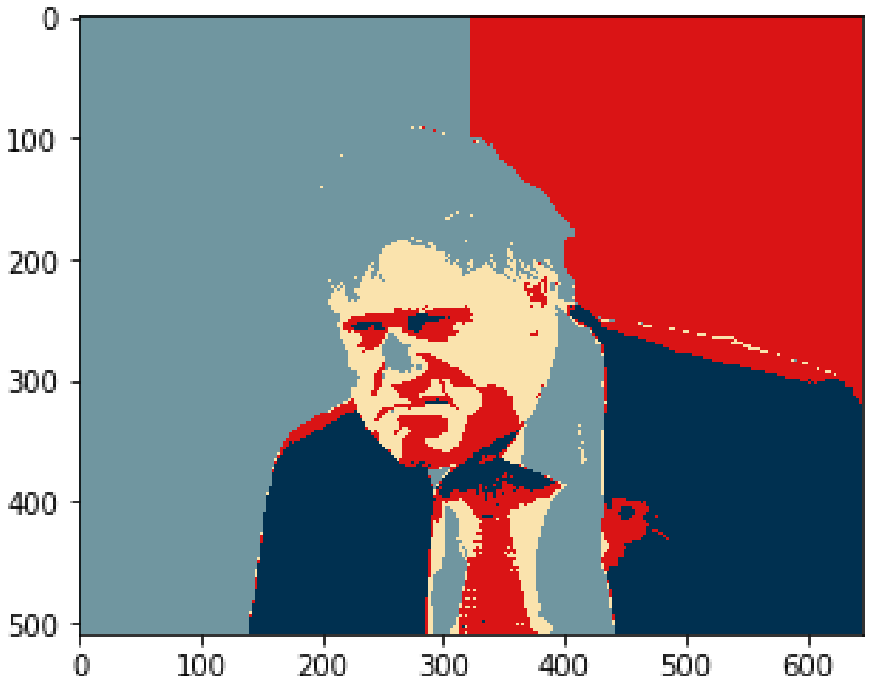



```
x , y = gray_johnson.shape
color_johnson = np.zeros((x,y,3))
for i in range(x):
    for j in range(y):
        if 0<gray_johnson[i,j]<= 162:
            color_johnson[i,j] = [0,48,80]
        elif 162<gray_johnson[i,j]<= 187:
            color_johnson[i,j] = [218,20,21]
        elif 187<gray_johnson[i,j]<= 210:
            color_johnson[i,j] = [250,227,173]
        elif 210<gray_johnson[i,j]<= 255:
            color_johnson[i,j] = [112,150,160]
```


0
100
200
300
400
500
0
100
200
300
400
500
600

d.


0
100
200
300
400
500
0
100
200
300
400
500
600
700
800



e.

The importance of values of thresholds is that these parameters control some details in the image such as some separating lines or may define outlines of some shapes. Therefore, they are important in creation of meaningful images.

8.

???

9.

a.

Because of the fact that this process is obtained from its general proof for continuous space and actually it is an approximated approach and its equations cannot be proven in general but in contrast in continuous space it produces an exact flat histogram; it is due to the fact that in the continuous domain there is an infinite number of values in any interval.

b.

Considering my personal experiment on some images, lastly images converge to a situation at which applying equal 3x3 smoothing filter does not change images anymore or at least a sensible change.

c.

LSB:

If the LSB bit plane is set to zero, in general, the number of bins or components in the histogram of the image will be reduced. Hence, number of pixels in each bin or component will increase substantially. This phenomenon results in increase in the height of a few peaks existed in histogram. Also, it may slightly shift the histogram to the left as many of the pixels' values will be decreased.

MSB:

If the MSB bit plane is set to zero, in general, much more bins or components in the histogram of the image will be reduced compared to LSB. Hence, number of pixels in each bin or component will increase substantially. This phenomenon

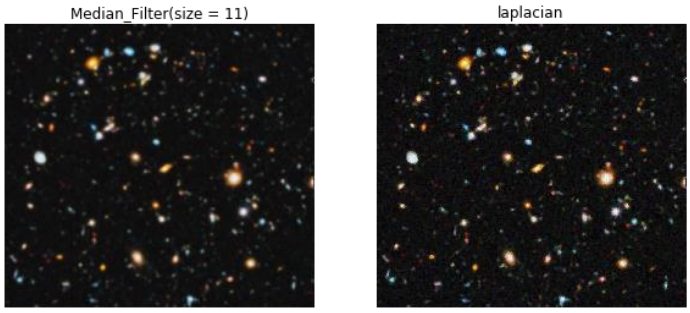
results in increase in the height of a many peaks existed in histogram. Also, it may remarkably shift the histogram to the left as many of the pixels' values will be decreased.

d.

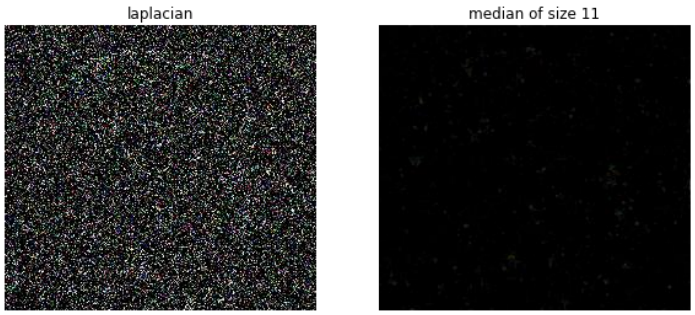
Since derivative filters are very sensitive to noise, it is common to smooth the image (especially using a Gaussian filter) before applying the Laplacian.

e.

In the first two images median filter was applied firstly then the Laplacian filter:



On the contrary, in the next two images Laplacian was applied in the first place then the median:



It seems that applying Laplacian filter without smoothing may break lots of image's shapes into salt and pepper noises most of which may be vanished after applying smoothing filter, hence, some main parts of the image may be vanished.