NANYANG TECHNOLOGICAL UNIVERSITY

CZ4003

Computer Vision

Lab 1 Report

Koh Tong Liang (U1921928D)

1. Contrast Stretching

a. Input the image into a MATLAB matrix variable by executing:

```
Name         Size                Bytes  Class    Attributes

Pc         320x443x3            425280  uint8
```

From reading the image we can identify that the original image is 320 by 443 dimension with RGB channels. Size of the image is 425280 bytes. We run rgb2gray() function to convert it to greyscale and check the images attributes:

```
Name         Size                Bytes  Class    Attributes

P          320x443               141760  uint8
```

We notice that the dimension is only 320 by 443 as the image is now in greyscale.

b. View this image using imshow. Notice that the image has poor contrast. Your initial task will be to investigate different methods for improving the image appearance using point processing. In point processing, the image transformation is carried for each pixel independently of neighbouring pixels.

c. Check the minimum and maximum intensities present in the image:

```
ans =

  uint8

   13


ans =

  uint8

   204
```

By applying contrast scaling, we scale the minimum value 13 to minimum and maximum value 204 to 255.

d. Next, write two lines of MATLAB code to do contrast stretching. Hint: This involves a subtraction operation followed by multiplication operation (note that for images which contain uint8 elements, you will need to use imadd, imsubtract, etc. for the arithmetic instead of the usual operators; alternatively you can convert to a double-valued matrix first using double, but you will need to convert back via uint8 prior to using imshow — see below). Check to see if your final image P2 has the correct minimum and maximum intensities of 0 and 255 respectively by using the min and max commands again.

```
ans =

  uint8

   0


ans =

  uint8

   255
```

Result of min and max after contrast scaling is performed.

e. Finally, redisplay the image

Original:

Result from imshow(P2):

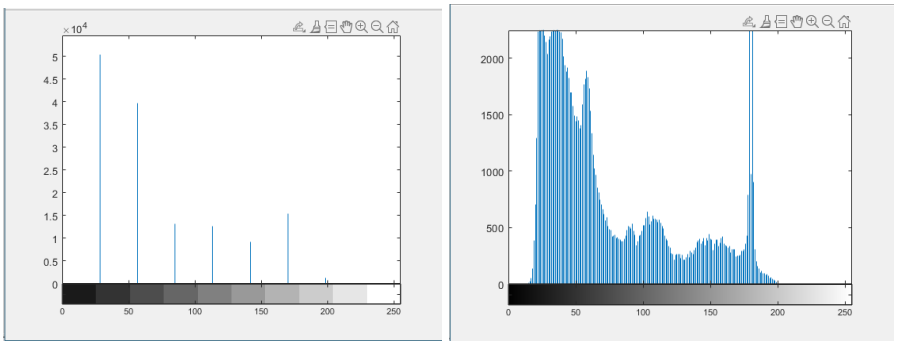
Result from imshow(P2,[]):

From the result, we can see that after contrast stretching, the new image has a better contrast as opposed to the original image.

2. Histogram Equalization

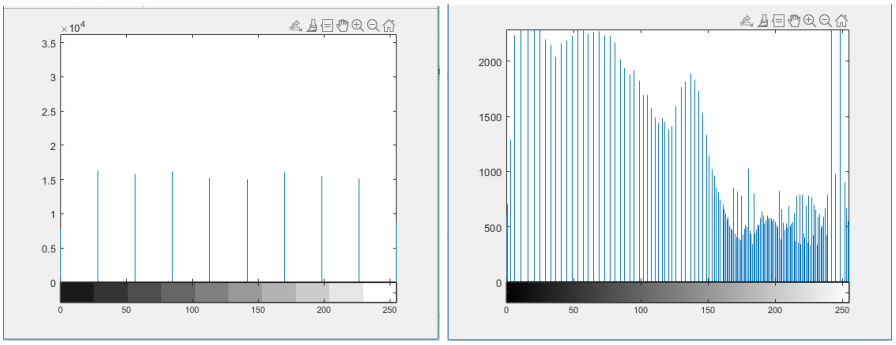
a. Display the image intensity histogram of P using 10 bins by. Next display a histogram with 256 bins. What are the differences?

10 bins vs 255 bins:



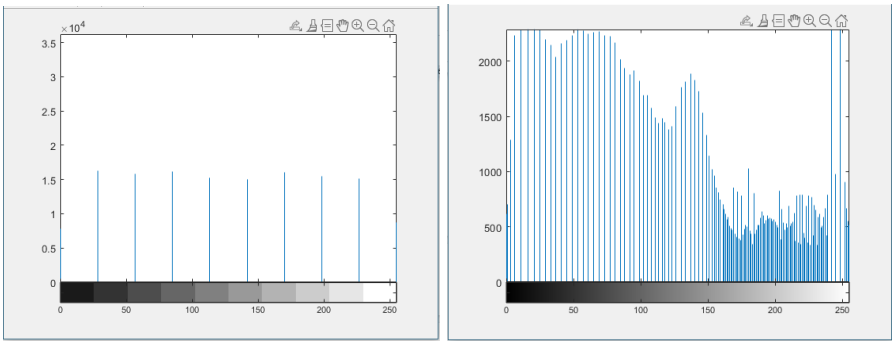
We can observe that 10 bins gives a more generalized histogram as compared to a more detailed histogram shown by 255 bins. The distribution in general across the 2 different histograms is the same however.

b. Next, carry out histogram equalization as follows:



Histogram equalization is performed and now we observe that the results in 10 bins and 255 bins are different. In 10 bins the histogram is more or less equalized whereas in 255 bins the histogram is not equalized (albeit it is more evenly distributed than the previous result). The reason for this result is because the image used is in grayscale so the histogram is only approximately flat.

c. Rerun the histogram equalization on P3. Does the histogram become more uniform? Give suggestions as to why this occurs.

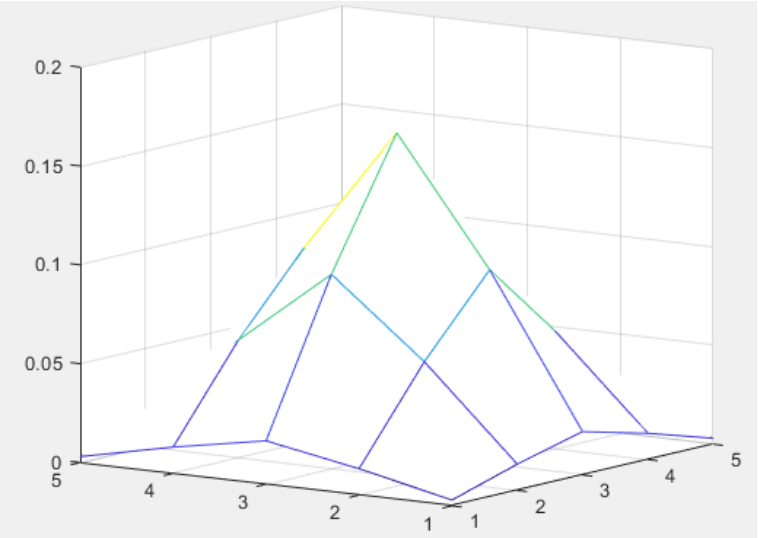


Histogram is already equalized, running histeq() will note change anything. Image comparison (left: original, right: after histeq()):

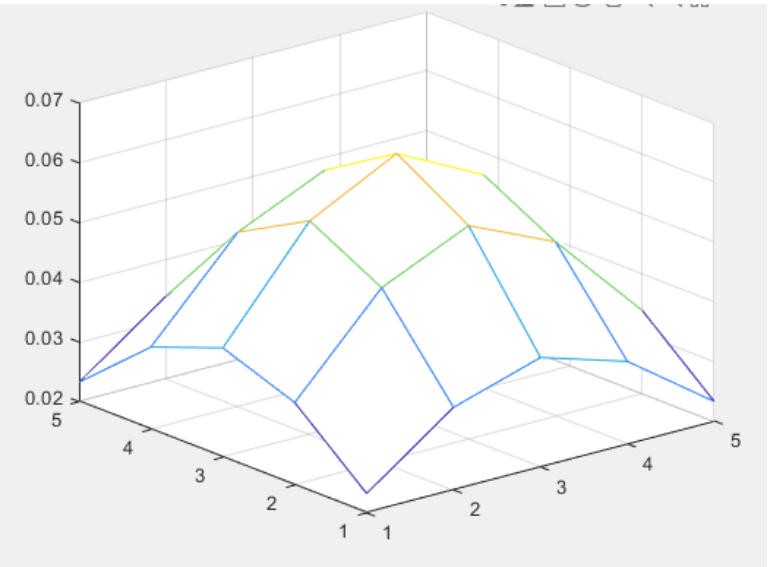
3. Linear Spatial Filtering

a. Generate the following filters:

i.



ii.



b. Download the image ‘ntu-gn.jpg’ from NTULearn and view it. Notice that this image has additive Gaussian noise.

c. Filter the image using the linear filters that you have created above using the conv2 function, and display the resultant images. How effective are the filters in removing noise? What are the trade-offs between using either of the two filters, or not filtering the image at all?

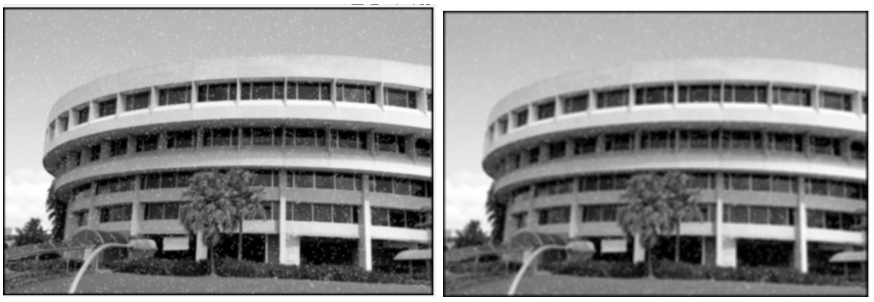
Comparison of result (left: filter i, right: filter ii):

Applying the second filter seems to result in the image becoming visibly blurer. This could be due to the higher std value which results in higher noise filtering but increased blurriness in the resulting image. In conclusion, there is a need to strike a balance between the amount of noise to filter versus the amount of details to preserve.

d. Download the image 'ntu-sp.jpg' from NTULearn and view it. Notice that this image has additive speckle noise.

e. Repeat step (c) above. Are the filters better at handling Gaussian noise or speckle noise?

Comparison of result (left: filter i, right: filter ii):



Similar results, more filtering leads to lesser details preserved. Upon further observation we can observe tiny speckles still showing up in the filtered images. Gaussian averaging filtering relies on the distribution of pixel intensity to be smooth with respect to the individual pixel's neighbour in order to achieve the desired filtering result. However in the cases of the pictures with speckles, the pixel intensity in the speckles differs sharply to its neighbours thus causing the filter to perform poorly. We can deduce that gaussian averaging filter is not suited for this speckle filtering task.

4. Median Filtering

a.

b.

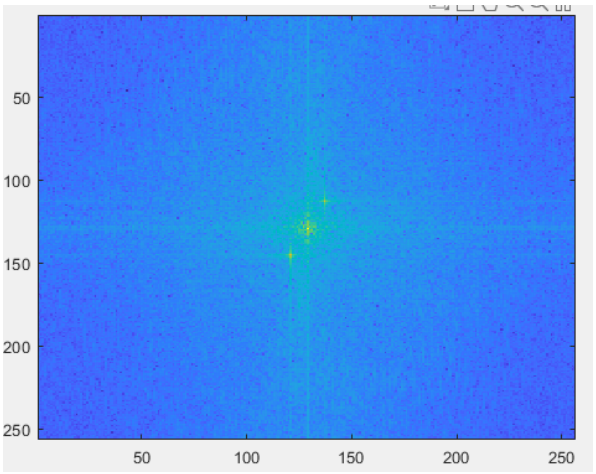
From the results alone it is clear that median filtering performs significantly better than gaussian averaging filtering in the speckle filtering task. The interesting observation is that the median filter creates a sharper image after filtering images with speckle while producing blurry images after filtering images with additive gaussian noise. The reason for this is that the median filter will ignore the pixels with sharp intensity compared to its neighbours because it looks primarily at the median value. The same tradeoff of details vs filter strength applies.

To summarise, different tasks would require different filters. In our case here we determine that gaussian averaging filter works best at filtering additive gaussian noise while median filtering works best at filtering speckle noise. We must first understand the nature of the noise present in the image before deciding the best approach to designing a filter.

5. Suppressing Noise Interference Patterns

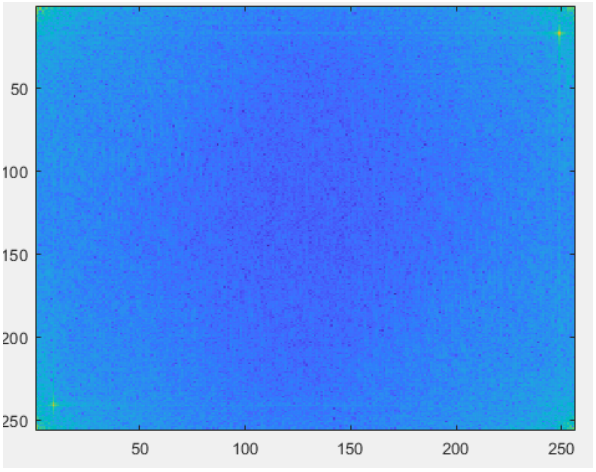
a. Download the image 'pck-int.jpg' from NTULearn and display it from MATLAB. Notice the dominant diagonal lines destroying the quality of the image.

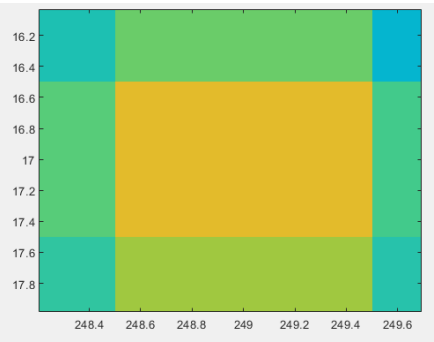
b. Obtain the Fourier transform F

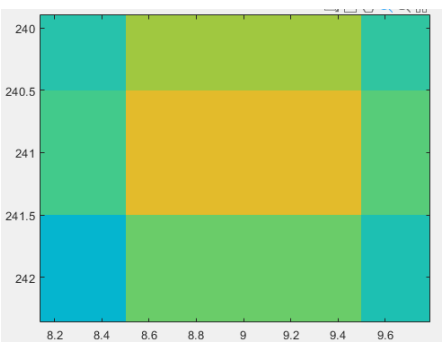


We can observe from the spectrum map that there are 3 sharp specks. We can deduce that the interference are of similar pixel values due to the pattern shown.

c. Redisplay the power spectrum without fftshift. Measure the actual locations of the peaks. You can read the coordinates off the x and y axes, or you can choose to use the ginput function.



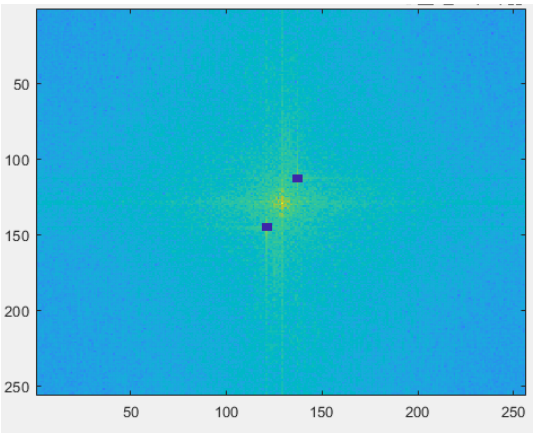




x = [16.5,17.5], y = [248.5, 249.5] x = [240.5, 241.5], y = [8.5, 9.5]

(x,y) = (17, 249) (x,y) = (241, 9)

d.



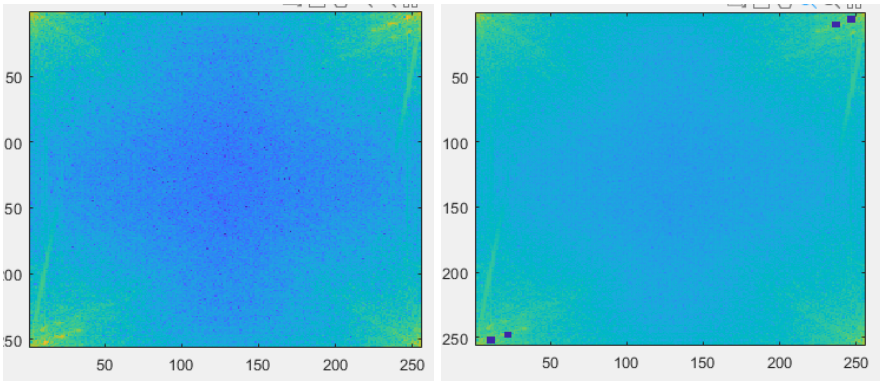
Frequency peak is removed corresponding to the interference of the original image.

e. Compute the inverse Fourier transform using ifft2 and display the resultant image. Comment on the result and how this relates to step (c). Can you suggest any way to improve this?

Interference is significantly reduced as frequency corresponding to the noise is removed. The remaining noise corresponds to the frequency not belonging to the peaks. Because we only removed the peaks, they still remain when transformed back to the image. We can potentially find more peaks to be removed to reduce the interference further.

f. Download the image `primate-caged.jpg' from NTULearn which shows a primate behind a fence. Can you attempt to "free" the primate by filtering out the fence? You are not likely to achieve a clean result but see how well you can do.

Identifying the peaks



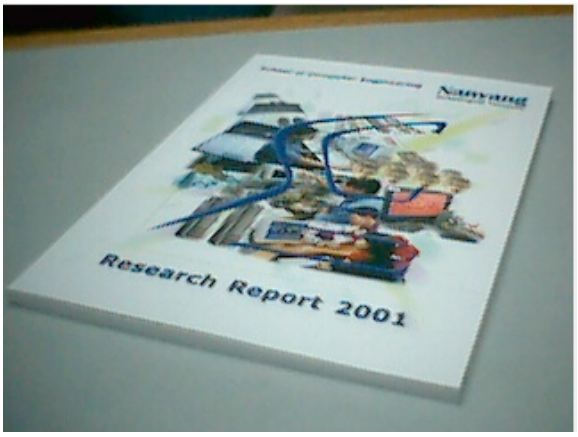
Peak1: (6, 247) Peak2: (10, 237) Peak3: (252, 11) Peak4: (248, 22)

Result comparison:

We can observe that the cage is significantly less visible after removing the peaks identified. However, the lines can still be made out leading to the conclusion that a more complex and powerful technique is required to tackle the tasks. It can also be observed from the spectrum map that the peaks are less obvious when compared to the previous television example. This suggests that the fourier transform is unable to detect the line patterns as clearly. In the previous scenario it is easy to discern the peak unlike in this scenario. Fourier transform may not be the most suitable method to tackle this particular task.

6. Undoing Perspective Distortion of Planar Surface

a. Download `book.jpg' from the NTULearn website as a matrix P and display the image. The image is a slanted view of a book of A4 dimensions, which is 210 mm x 297 mm.



b. The ginput function allows you to interactively click on points in the figure to obtain the image coordinates. Use this to find out the location of 4 corners of the book, remembering the order in which you measure the corners.

c.

d.

e.

Result is as expected. Whichever portion of the original image that could not be discerned clearly will have the same issue in the resulting image (top left words; blurry graphics etc.). Overall, the performance is good as there is no visible loss in quality and details. If given a high resolution image, I believe the result will be even better.