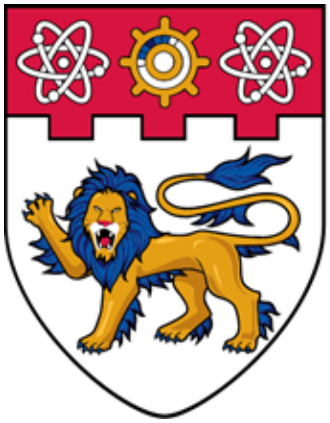

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

CE4003: Computer Vision

Lab 1:

Point Processing + Spatial Filtering + Frequency Filtering + Imaging Geometry

Prepared by:

Venkat Subramanian (U1921075D)

Table of Contents

1. Contrast Stretching

Contrast Stretching is a point processing image enhancement method. In point processing, each pixel's new grey level does not depend on the other pixel's grey level. Contrast stretching attempts to improve the contrast in an image by spreading the range of intensity value it contains to a desired range of values. In this experiment, we will be stretching it to the full range of pixel values which is 0 to 255.

The original image's minimum intensity value was 13 and the maximum intensity value was 204. This range does not fully occupy the intended range of pixel intensity values.

Figure 1:Original MRT image

To stretch the pixels to their intended range, each pixel will be subjected to this formula:

$$New\ Pixel\ Intensity\ value = \frac{255(Pixel\ intensity - \min Image\ Intensity)}{max\ Image\ Intensity - \min \text{Image Intensity}}$$

Where min Image intensity and max Image Intensity refer to the original image's global minimum grey and maximum grey level respectively. The MATLAB the code is as such

```
P_double = cast(P,"double");
P2 = ( 255 .* ( P_double - min(P_double(:)) ) ) ./ ...
                ( max(P_double(:)) - min(P_double(:))) ;
P2 = cast(P2,"uint8");
```

Figure 2: MATLAB code for contrast Stretching

To check if the contrast stretching is done properly, we can check the global minimum and maximum of the grey levels.

```
>> min(P2(:)),max(P2(:))

ans =

  uint8

    0


ans =

  uint8

    255
```

Figure 3:Comparing the effects of contrast stretching

From Figure 2, it is observable that the darker areas (such as the face of the MRT and mosque) in the original image are much brighter in the new image. The MATLAB code for this entire section please refer to Figure 25 in the appendix.

2. Histogram Equalization

In section 1, we discussed contrast stretching which makes the image use the entire range of intensity values. Another way the image can be enhanced is through histogram equalisation.

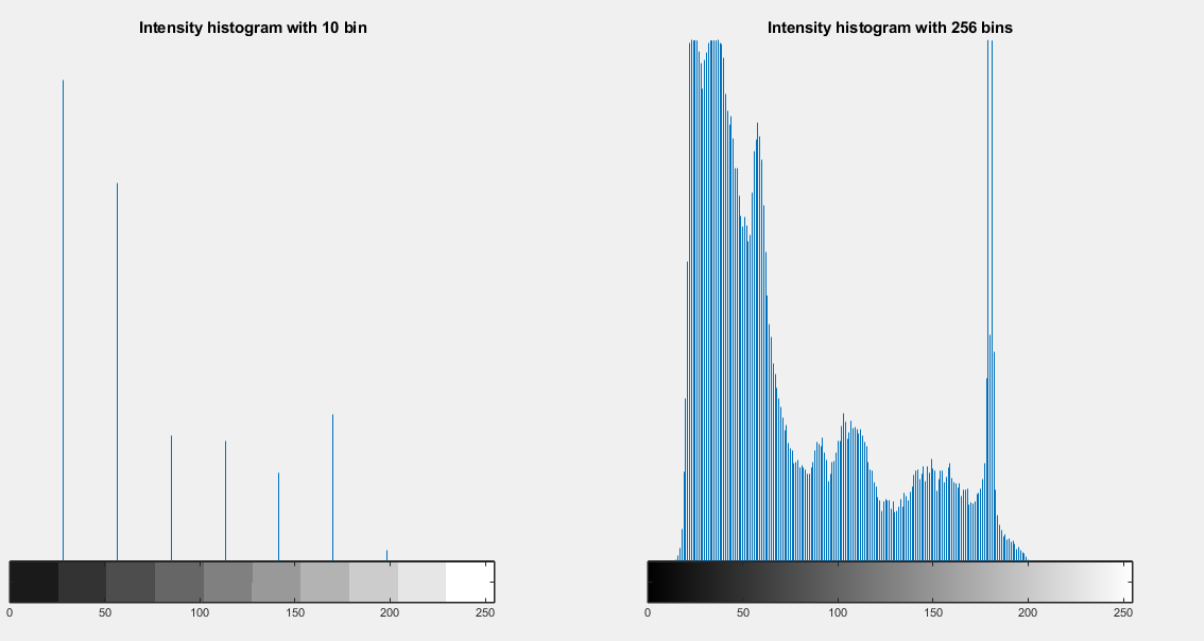

Figure 4:Intensity histogram of the original image

The intensity distribution of the pixels in the MRT image can be seen in the 2 histograms. Most of the pixel values are in the lower region explaining why the image appears to be very dark. The significant difference between the 2 types of histograms is the degree of fluctuation. Since the histogram with 10 bins averages all the pixels the fluctuations are not very stark. But if we break it down to 256 bins, we can see the fluctuations, especially at around 170. Histogram equalisation will

try to flatten the grey-level histogram as much as possible. By doing so the cumulative probability distribution will be linear. After applying the histogram equalisation in MATLAB the resulting histograms are shown below.

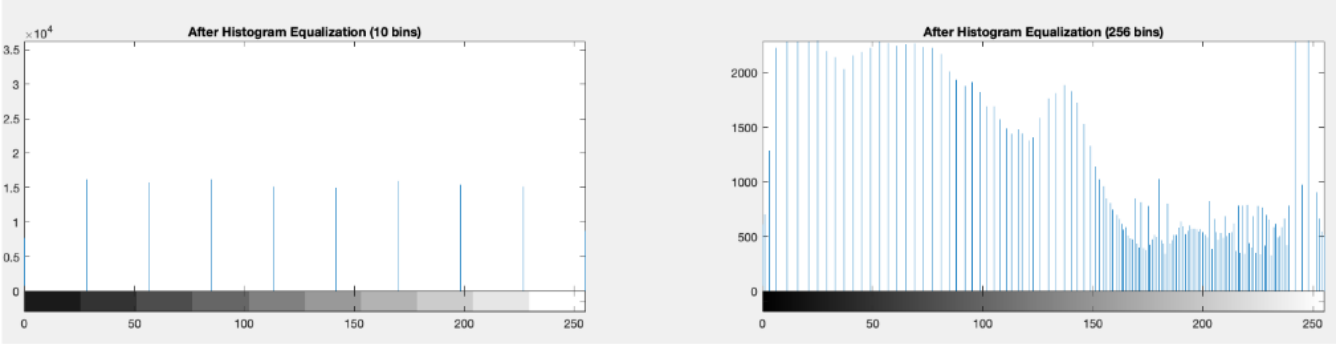

Figure 5: Intensity histogram of enhanced image

Both histograms generally have a lower degree of fluctuations when compared to the original image. The difference between these histograms will be that the 10-bin histogram is relatively stable whereas the 256-bin histogram still fluctuates. But on closer inspection, the fluctuating bins are closer together while those relatively stable bins are placed further apart.ucing As a result, the cumulative distribution of the pixels is almost linear, and it gives a perception that the 256-bin histogram is unstable. Overall, we can see better results than in section 1. A reason for this could be that histogram equalisation is a non-linear mapping function and it will spread the low-intensity bins while contrast stretching is a linear mapping function.

Figure 6: Original Image vs Equalized Image

Histogram equalization is idempotent meaning if it is operated on itself the value remains unchanged. As mentioned previously, histogram equalization will try to reorder the bins such that the cumulative probability distribution of the intensities is as linear as possible. The new position of the bin depends on the cumulative distribution of the previous bins before it. As a result, reapplying histogram equalization will not change the results as it is already in an ordered manner. The code for this section is found in Appendix figure 26.

3. Linear Spatial Filtering

In sections 1 and 2, we discussed 2 point processing methods to enhance images. In this section, we will be discussing the spatial processing method. In particular, we will be looking at how spatial filtering is used to remove unwanted noise from an image.

The image that we will be working on is added with additive white gaussian noise. To remove the noise we will be experimenting with 2 ,5 by 5 gaussian filters one with a standard deviation of 1 and another with 2.

To generate these filters a nested loop is used. Since the centre value of the kernel has to have an index of 0 the range of x and y are -2 to 2. The value at each index is given by the gaussian distribution thus the name gaussian filter.$h(x,y) = \frac{\left(e^{-\frac{x^2+y^2}{2\sigma^2}}\right)}{2\pi\sigma^2}$. After computing the values at each index, we need to normalize these values so that the sum of all elements in the kernel adds up to 1

```
sd = 1 ;
var = sd ^2;
convolutional_filter_1 = zeros(5,5);
for x = -2:2
    for y = -2:2
        convolutional_filter_1(x+3,y+3) = ( 1/ (2 * pi * var) ) * ...
                                    exp( -1*(x^2 + y^2)/(2*var)) ;
    end
end
convolutional_filter_1 = convolutional_filter_1 ./ sum(convolutional_filter_1(:));
figure;
mesh(convolutional_filter_1);
```

Figure 7: Code snippet of gaussian filter generation

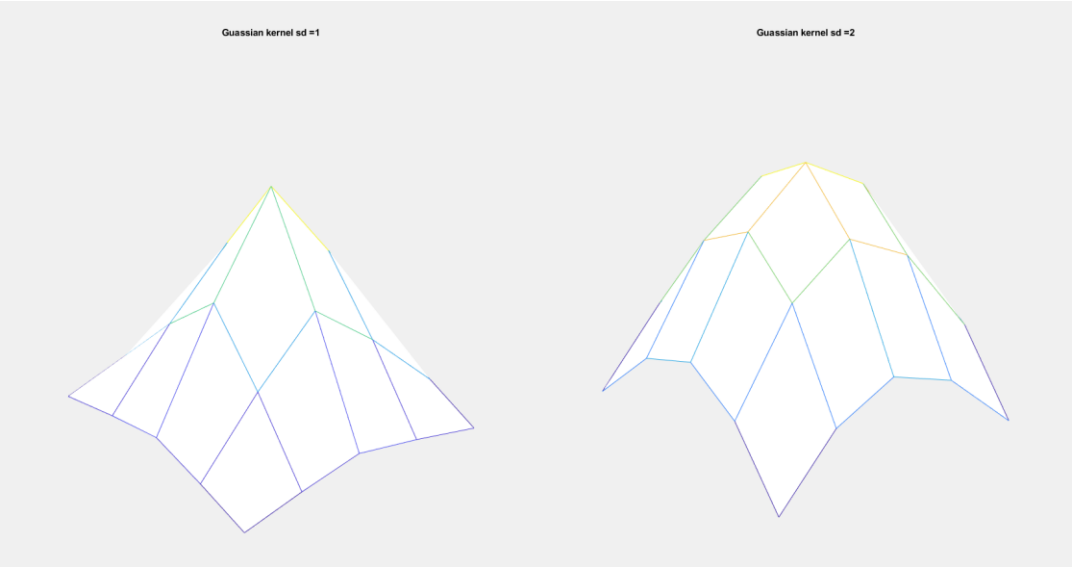

Figure 8: Gaussian kernels generated

As expected, the kernel with a standard deviation of 2 has a larger spread than the kernel with a standard deviation of 1. When these filters are convoluted with the original image the results vary.

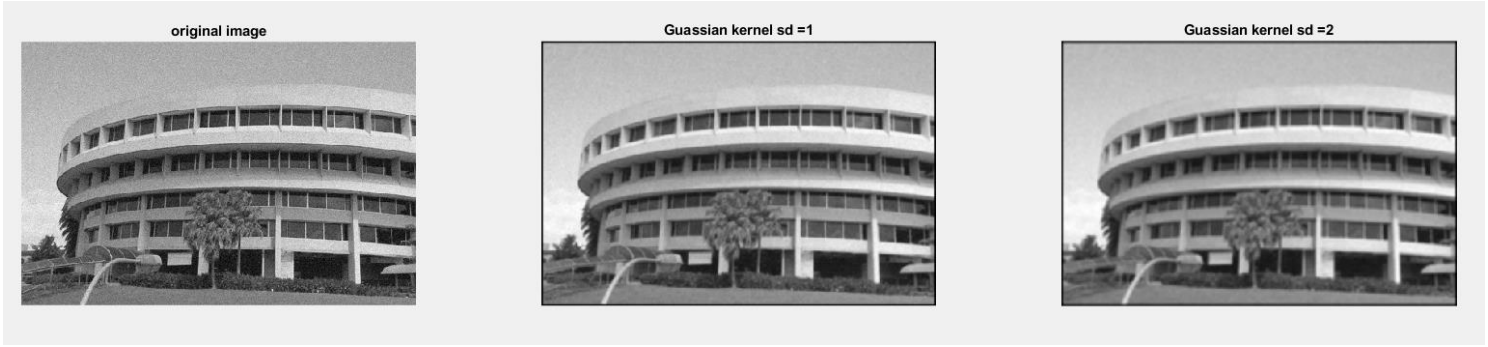

Figure 9:Comparing different Gaussian filters

By increasing the standard deviation of the gaussian filters, the image is blurred and the borders are thickened. Standard deviation represents the variation from the mean, so a lower standard deviation would mean that the weights will be focused around the centre and less around the edges. In the case of edges where there is a sharp transition of gradient, increasing the sigma will smoothen this sudden change resulting in a loss of detail. On the other hand, increasing the sigma value will generate the image's overall trend, removing the effect of noise. Hence when sigma is increased, there is a trade-off between sharpness and noise reduction.

Other than additive white gaussian noise, an image can be corrupted with speckle noise. Speckle noise is a multiplicative noise that affects pixels in a grey-scale image. It is also known as salt-pepper noise since the noise appears like salt and pepper on the image.

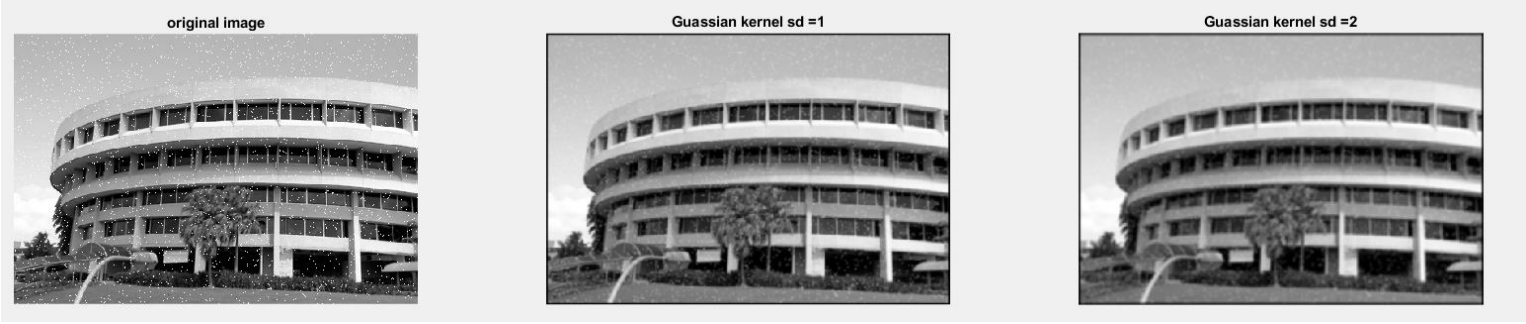

Figure 10: Comparing the effects of gaussian filter on speckle noise

By increasing the sigma value of the kernel, it can remove more noise, but the effect of the speckle noise is still evident in the image. Therefore, gaussian filters are not very suitable for speckle noise, they should be used to filter gaussian noise. The code for this section is found in Figure 27 of the Appendix.

4. Median Filtering

A suitable filter to remove the speckle noise will be the median filter. A median filter is a non-linear filter which can remove impulse noise while preserving the edges. Unlike the gaussian filter, the median filter will not be convoluted with the image. Instead, the median filter will try to find the median of the pixel values and replace the current pixel value with that median. In this section we will be experimenting with 2 sizes of median filter, 3 by 3 and 5 by 5, on the image with speckle noise

Figure 11:Comparing the effect of median filter on speckle image

From Figure 11, the 3 by 3 median filter can remove most of the speckle noise but some of it can still be seen (those points that are within the red circles). But when we increase the filter size can remove all the noise. When the kernel size is increased the output captures the trend of the image such as the building and the structures surrounding it. But it is not able to capture the details of the image such as the windowpane edges. A possible reason for these phenomena is that a larger median filter will encompass a larger set of pixels therefore the median will deviate more from the mean.

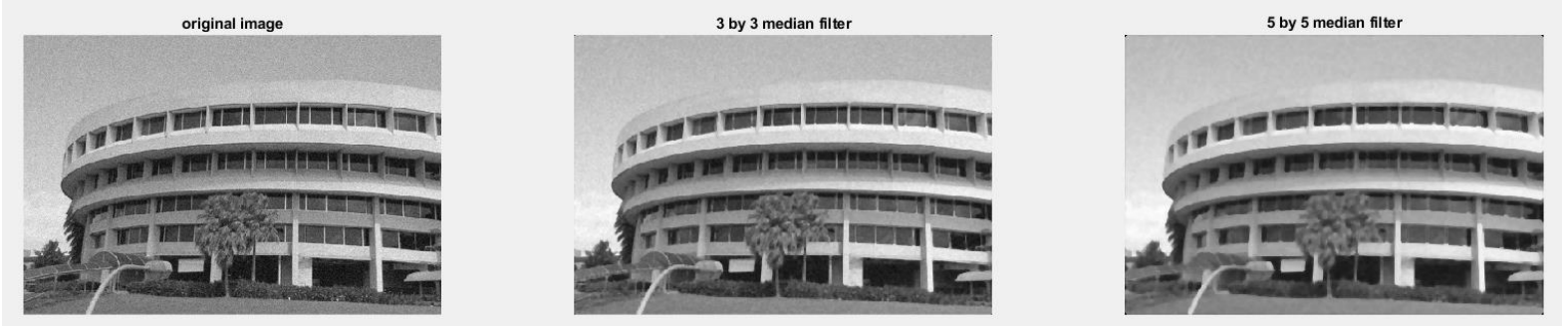

Figure 12: Comparing the effects of median filter on gaussian noise

From figure 12, it can be observed that the Median filters are not very effective on the gaussian noise since the noisy pixels will be similar to the surrounding pixels. Therefore, these noisy pixels can still be selected as the median, unlike speckle noise which has a higher value than its surrounding pixel values. The code for this section can be found in Appendix figure 28.

5. Suppressing Noise Interference

Sometimes images can also be corrupted with interference patterns. To remove these patterns, gaussian and median filters might not be very suitable. Instead, a bandpass filter is more appropriate to suppress such interference. In this section, we will be discussing how Fourier transform can be used to filter the frequencies of unwanted patterns

The first image that we will be experimenting with is shown in figure 13. Bandpass filtering aims to remove the diagonal lines that spread across the image. To remove this interference pattern, we must first identify the frequency of these patterns so that we only remove the interference pattern and not the base image itself. The frequencies of the image can be obtained by Fourier transform. To easily identify the peak frequency components a power of 0.1 is used to nonlinearly scale the power spectrum (which is the absolute of Fourier transform)

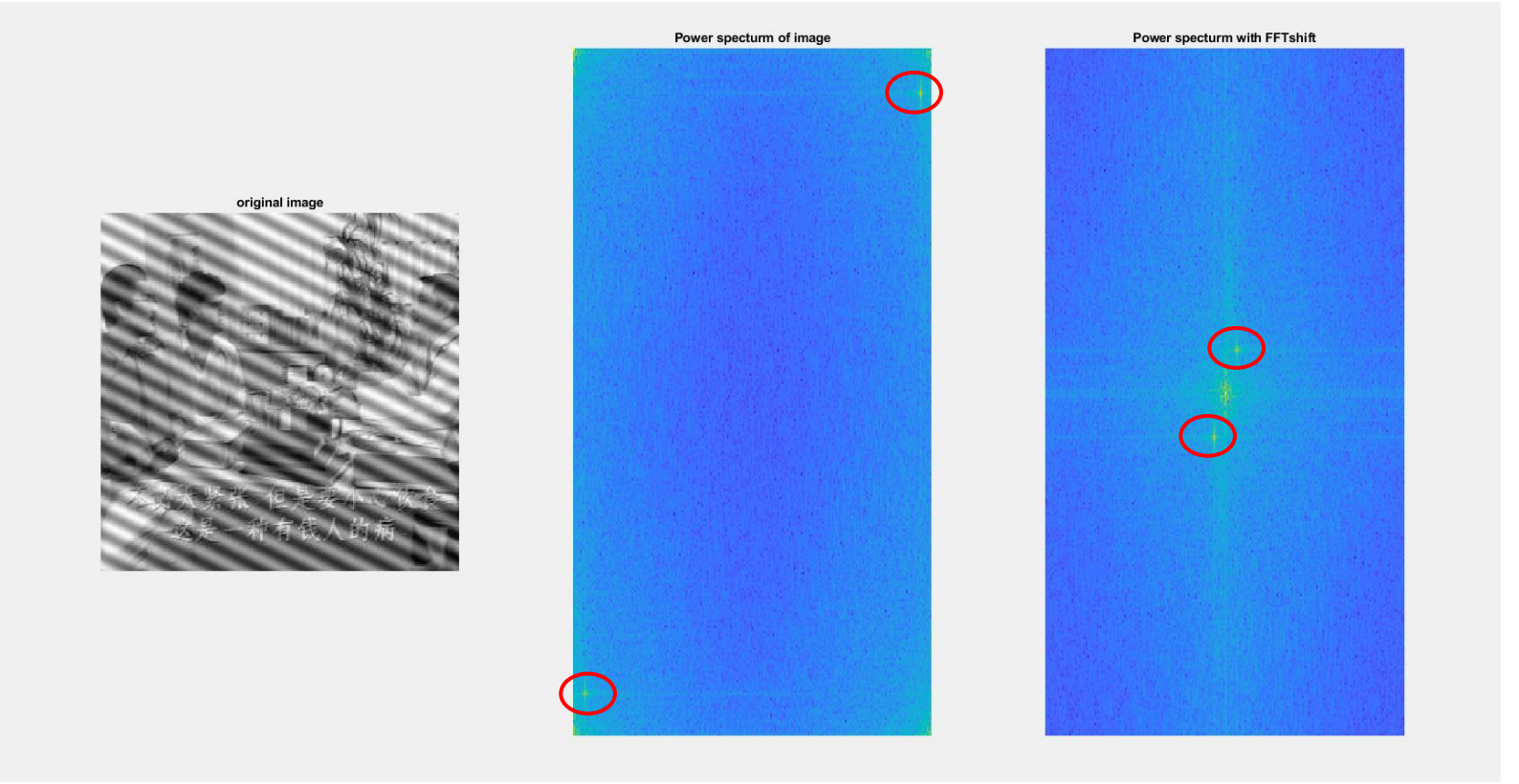

Figure 13: Image with Interference and its spectrum

From the fftshift of the power spectrum, we can identify 2 peaks surrounding the centre. These 2 peaks could infer the frequencies of the interference pattern. The 2 peaks occur at (241,9) and (17,249). In MATLAB the horizontal axis represents the Y axis while the vertical axis represents the X axis, therefore, the coordinates of the peaks are swapped when compared to conventional coordinates.

To remove the interference pattern, we try to set the 5x5 neighbours around the located power peaks to 0 (in the Fourier transform matrix). The new power spectrum is shown in figure 14

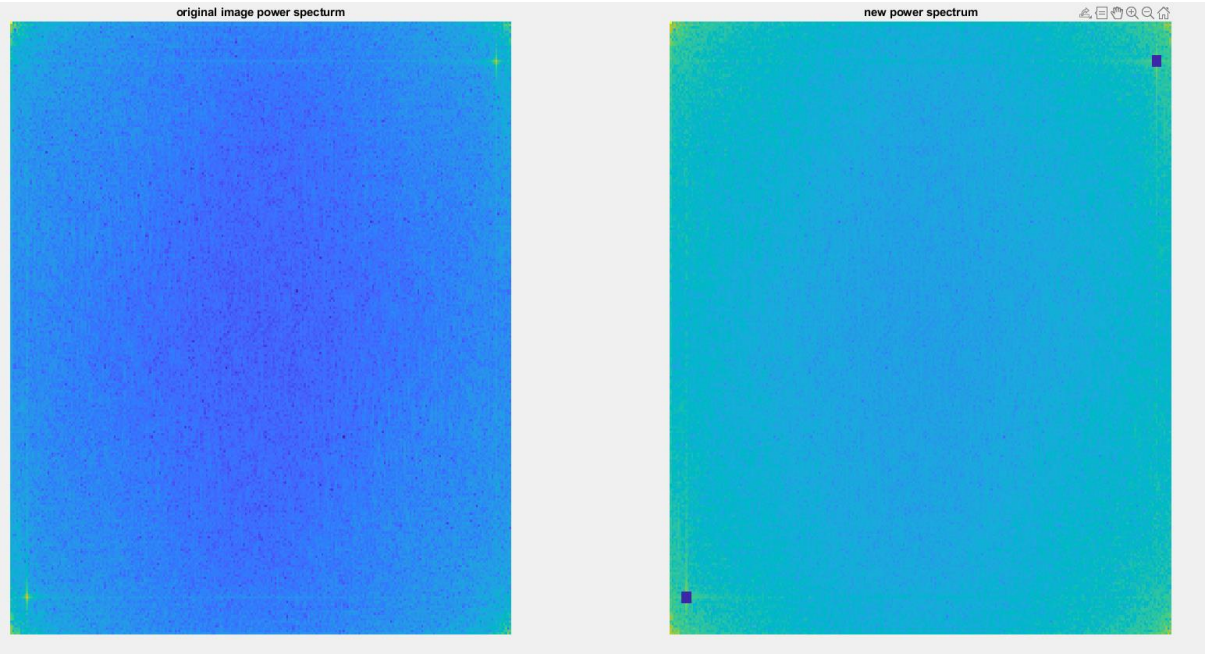

Figure 14: Removing Interference frequency

The next step is to check if the above process managed to remove the interference pattern. This is done by doing an inverse Fourier transform to convert the spatial frequency matrix back into the spatial domain

Figure 15: Before and after image of removing interference

By removing the frequency peaks, we were able to remove most of the interference but some remain. This could be the result of the trailing lines from the frequency peaks as shown in figure 16 (highlighted in red). Other than the trailing edges, 5x5 neighbouring might not be enough. If we refer to figure 14 we see some frequency peaks located outside the box. So to enhance the image we will remove these weak frequency peaks that might be related to the interference pattern and increase the padding from 5 to 6 and set the 6x6 neighbouring frequency elements to 0.

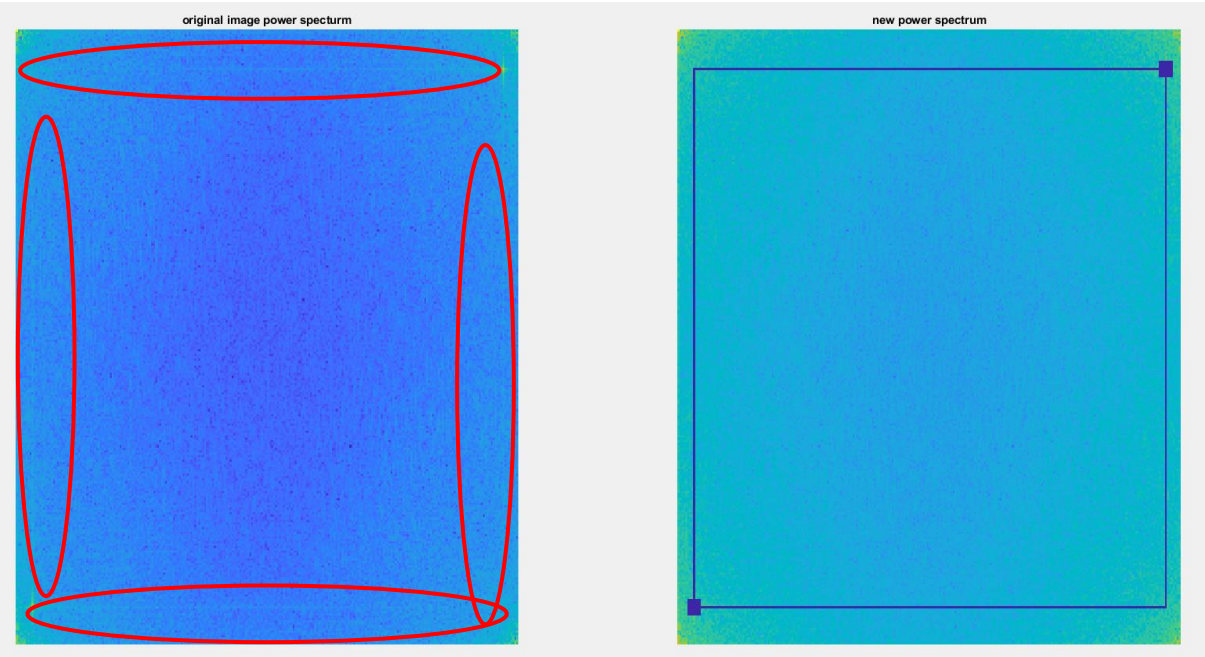

Figure 16: Enhanced Power spectrum vs the original one

Figure 17:Comparing Enhanced image with the previous images

From figure 17, the enhanced image has a lower interference (it doesn't completely remove)when compared to the image obtained earlier (when removing the 2 peaks). And at the same time, the original image is kept intact, and the details of the image are not lost/blurred. The code for this experiment is found in Figure 29 of the Appendix.

The other image will be a caged primate. Here we are trying to remove the unwanted elements of the image which is the cage. By utilizing the same method as discussed previously, we will identify the frequency peaks of the power spectrum and change their neighbours to a frequency value of 0. There is an additional step before that which is to convert the RGB colour image of the primate to greyscale. Similar to the previous image, we will not be able to remove the interference completely.

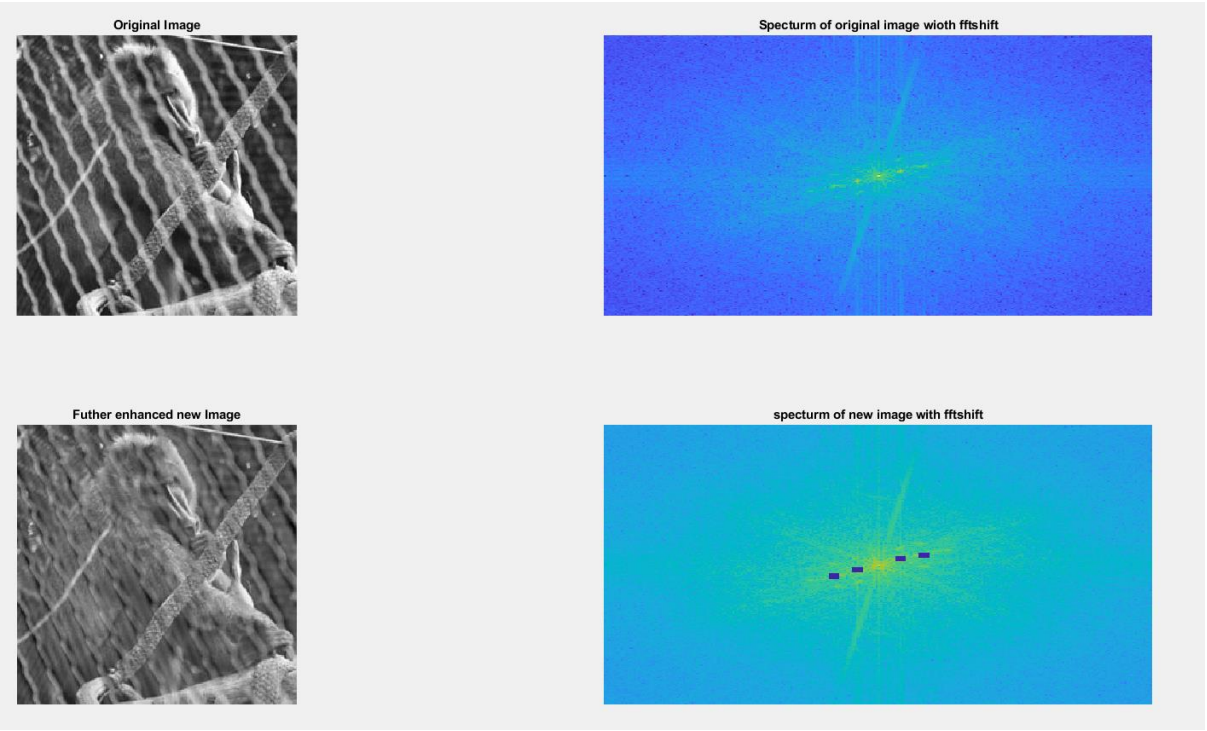

Figure 18: The primate and its spectrum before and after enhancement

Another method is to find a threshold and select only frequencies above or below the threshold. From the analysis of the spectrum, most of the high-frequency values lie above 3.3. So we should only select the elements that have a frequency value that is below 3.3.

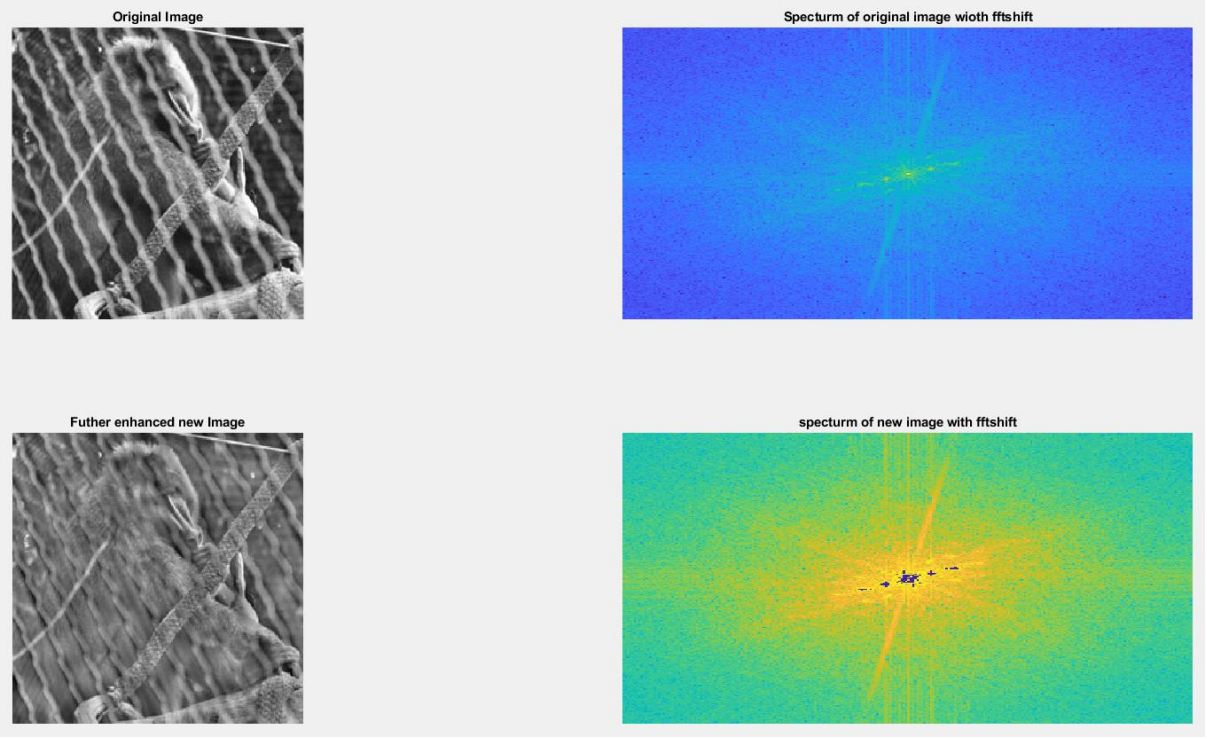

Figure 19: primate image and its spectrum after low pass filter

Figure 20: Comparing the output of the 2 different methods

When the output of the images are compared together, it is observable that the details of the primate such as the whiteness of the fur, and the back of the primate is lost when we carry out the second method. The second method will remove all the elements that have a frequency that is higher than 3.3 and not all of them might correspond to the interference pattern. Some of these elements might be part of the primate itself. But hand picking out the frequency peaks might be a time-consuming process while the second method will be more automated. So there is a tradeoff between automation and details of the image. The full MATLAB code for this section can be found in Appendix Figures 30 and 31.

6. Undoing Perspective Distortion of Planar Surface

In this section, we will be exploring how computer vision can be used to change the perception of the object. We can transform the image by multiplying the coordinates of a point in the original image with a 3x3 matrix to obtain the coordinates of the transformed image. The equation on the right shows when xim and yim are made the subject.

$$\begin{bmatrix} kx_{im} \\ ky_{im} \\ k \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ 1 \end{bmatrix}$$

Figure 22:Planar transformation formula

$$x_{im} = \frac{m_{11}X_w + m_{12}Y_w + m_{13}}{m_{31}X_w + m_{32}Y_w + 1}, y_{im} = \frac{m_{21}X_w + m_{22}Y_w + m_{23}}{m_{31}X_w + m_{32}Y_w + 1}$$

Figure 21:Making xim the subject

The equation has 8 unknowns (the m variables) so they can be broken into a multiplication of 2 matrix Au = v where V is the coordinates of the projected image and A is the coordinates of the actual image.

We will be experimenting with a slanted image of a book with dimensions of 210x 297mm. So v will be [1 1 210 1 210 297 1 297] starting from the top corner and going in an anti-clockwise manner. A will be obtained by selecting the corner of the book in the same order as the v matrix.

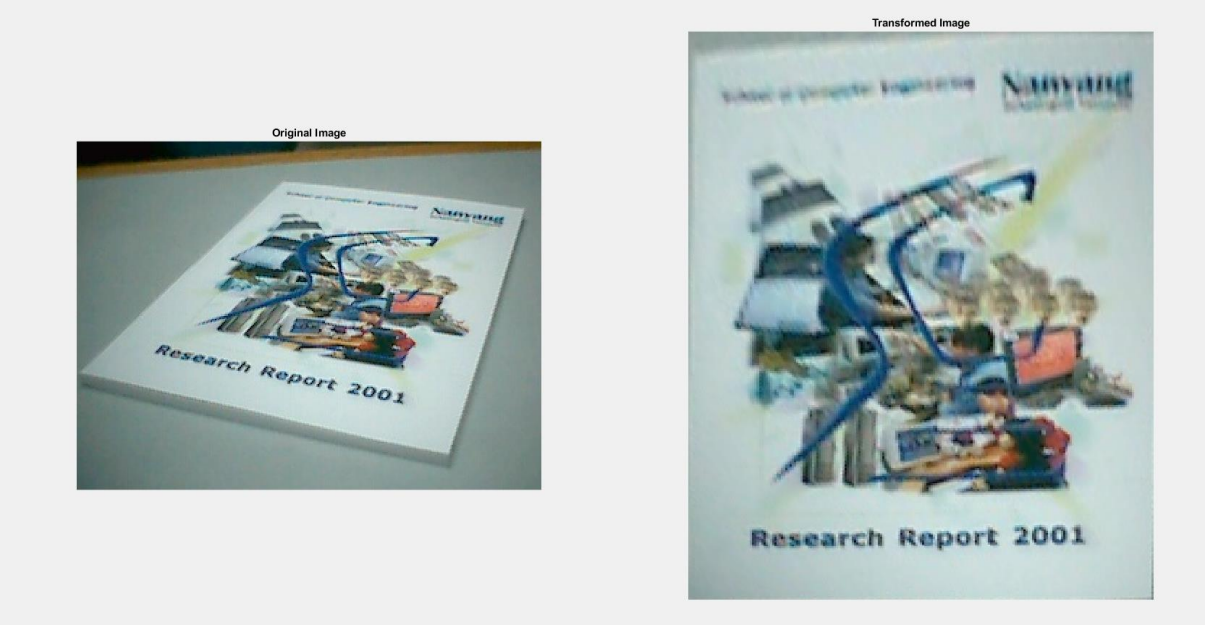

Figure 23:Comparing the original and planar transformed image

The successful transformation is shown in figure 23, the slanted book is now standing upright. Even though the planar transformation worked the new image appears to be blurry at the top while being clearer at the bottom of the book. One of the possible reasons for this discrepancy could be that the bottom of the book is closer to the camera in the original image and appear to be clearer while the top of the book is positioned further away from the camera missing out a few details. Therefore, the planar transformation is clearer for the bottom half of the image and not the top. Refer to Figure 32 of the Appendix for the full code of this experiment.

Other than the planar transformation, we are also keen to know if we can extract the pink computer screen from the transformed image. This can be done in 2 ways, 1 is to manually find the corners of the computer and crop the image based on these 4 corners. This is a manual process; if there are multiple images, it might be laborious to manually select the 4 corners in each image. So the other method is to make use of colour segmentation to segment the pink screen out.

This can be done using K-means clustering. First, we need to select a suitable number of clusters such that only the pink screen falls within a cluster. After multiple tries, it was found that 8 clusters will be suitable to extract the pink screen. This is classifying the colours in the RBG colour space. After classifying it we have to convert the RBG to L*a*b* colour space using the rgb2lab() function. After which we need to classify the colours again in the a*b* space. Then we can get the different colour masks. The pink computer screen falls under the 2nd cluster.

In the 2nd cluster, some additional objects are being segmented. These colours mainly come from the girl's shirt and the chair she's sitting on. These colours are darker than the computer screen so we can segment the dark red from the light red/pink using the L* layer. The overall process and the final mask are given in figure 24. For a clear image of the final mask refer to the appendix figure 33 and the full code of this experiment in figure 34.

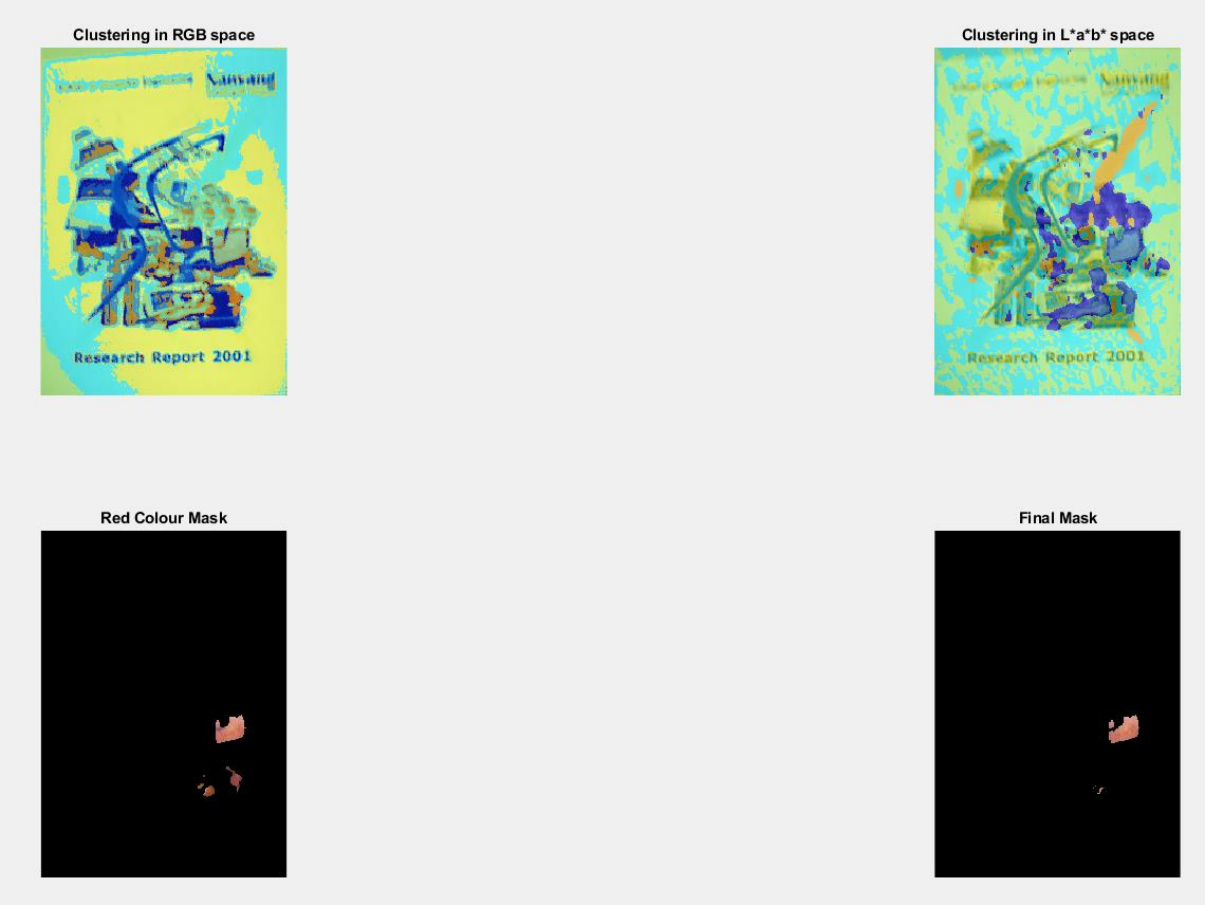

Figure 24:Image segmentation

Conclusion

In this lab 1 experiment, we have discussed point processing methods such as contrast stretching and histogram equalisation. We have also experimented with spatial processing techniques to filter out gaussian and speckle noise. In addition to those, we explored the use of Fourier transform to remove unwanted interference and finally changed the perspective of images and delved slightly into colour segmentation of images.

Appendix

```
%% Section 2.1
clear; close all;
Pc = imread('mrttrainbland.jpg');
whos Pc
P = rgb2gray(Pc);
whos P
figure;
imshow(P);
min(P(:)), max(P(:))

P_double = cast(P,"double");
P2 = ( 255 .* ( P_double - min(P_double(:)) ) ) ./ ...
                ( max(P_double(:)) - min(P_double(:))) ;
P2 = cast(P2,"uint8");
figure;
imshow(P2);
subplot(1,2,1);imshow(P);axis off; title("Original Image");
subplot(1,2,2);imshow(P2);axis off;title("Constrast Stretched Image");

min(P2(:))
max(P2(:))
```

Figure 25:Code for section 1

```
%% section 2.2 Histogram equalisation
figure;
subplot(1,2,1);imhist(P,10);axis off; title("Intensity histogram with 10 bin");
subplot(1,2,2);imhist(P,256);axis off;title("Intensity histogram with 256 bins");
% so with 256 bins each bin will represent the pixel and we can get a better repre
P3 = histeq(P,256); % the 256 refer to the number of bins you want to equalise so,
figure;
subplot(1,3,1);imshow(P);axis off; title("Original Image");
subplot(1,3,2);imshow(P2);axis off;title("Stretched Image");
subplot(1,3,3);imshow(P3);axis off;title("Equalised Image");
figure;
subplot(1,2,1);imhist(P3,10);axis off; title(" Equalised histogram with 10 bin");
subplot(1,2,2);imhist(P3,256);axis off;title("Equalised histogram with 256 bins");
% what is the similarity -> both have lesser fluctations from before
% but the one with 10bins has lesser fluctuations that 256 WHY IS THAT
P3 = histeq(P3,256);
figure;
imshow(P3);
figure;
imhist(P3,10);
figure;
imhist(P3,256);
```

Figure 26:Code for section 2

```
%% section 2.3
sd = 1 ;
var = sd ^2;
convolutional_filter_1 = zeros(5,5);
for x = -2:2
    for y = -2:2
        convolutional_filter_1(x+3,y+3) = ( 1/ (2 * pi * var) ) * ...
                                    exp( -1*(x^2 + y^2)/(2*var)) ;
    end
end
convolutional_filter_1 = convolutional_filter_1 ./ sum(convolutional_filter_1(:));
sd = 2 ;
var = sd ^2;
convolutional_filter_2 = zeros(5,5);
for x = -2:2
    for y = -2:2
        convolutional_filter_2(x+3,y+3) = ( 1/ (2 * pi * var) ) * ...
                                    exp( -1*(x^2 + y^2)/(2*var)) ;
    end
end
convolutional_filter_2 = convolutional_filter_2 ./ sum(convolutional_filter_2(:));
figure;
subplot(1,2,1);mesh(convolutional_filter_1);axis off; title("Guassian kernel sd =1");
subplot(1,2,2);mesh(convolutional_filter_2);axis off;title("Guassian kernel sd =2");
image = imread('ntugn.jpg');
whos image
figure;
subplot(1,3,1);imshow(image);axis off; title("original image");
subplot(1,3,2);imshow(conv2(image,convolutional_filter_1),[]);axis off;title("Guassian kernel sd =1");
subplot(1,3,3);imshow(conv2(image,convolutional_filter_2),[]);axis off; title("Guassian kernel sd =2");
image = imread('ntusp.jpg');
figure;
subplot(1,3,1);imshow(image);axis off; title("original image");
subplot(1,3,2);imshow(conv2(image,convolutional_filter_1),[]);axis off;title("Guassian kernel sd =1");
subplot(1,3,3);imshow(conv2(image,convolutional_filter_2),[]);axis off; title("Guassian kernel sd =2");
```

Figure 27:Code for section 3

```
%% section 2.4
image = imread('ntusp.jpg');
figure;
subplot(1,3,1);imshow(image);axis off; title("original image");
subplot(1,3,2);imshow(medfilt2(image,[3 3]),[]);axis off;title("3 by 3 median filter");
subplot(1,3,3);imshow(medfilt2(image,[5 5]),[]);axis off; title("5 by 5 median filter");


image = imread('ntugn.jpg');
figure;
subplot(1,3,1);imshow(image);axis off; title("original image");
subplot(1,3,2);imshow(medfilt2(image,[3 3]),[]);axis off;title("3 by 3 median filter");
subplot(1,3,3);imshow(medfilt2(image,[5 5]),[]);axis off; title("5 by 5 median filter");
```

Figure 28:Code for section 4

```
%% section 2.5
image = imread("pckint.jpg") ;
F = fft2(image);
S = abs(F);
figure;
subplot(1,3,1);imshow(image);axis off; title("original image");
subplot(1,3,2);imagesc((S.^0.1));axis off;title("Power specturm of image");
subplot(1,3,3);imagesc(fftshift(S.^0.1));axis off;title("Power specturm with FFTshift");
% the 2 freq are (241,9) and (249,17)
x1 = 241;x2 = 17;y1 = 9;y2 = 249;
% now we need to check if those 2 areas are indeed the high freq
figure;
imagesc(S(x1-2:x1+2,y1-2:y1+2).^0.1);
figure;
imagesc(S(x2-2:x2+2,y2-2:y2+2).^0.1);
%part d
F(x1-2:x1+2,y1-2:y1+2) = 0;
F(x2-2:x2+2,y2-2:y2+2) = 0;
figure;
subplot(1,2,1);imagesc((S.^0.1));axis off; title("original image power specturm");
subplot(1,2,2);imagesc((abs(F).^0.1));axis off;title("new power spectrum");
figure;
subplot(1,2,1);imshow(image);axis off;title("Original Image");
subplot(1,2,2);imshow(ifft2(F),[]);axis off;title("new Image");
F1 = fft2(image);
F1(x2:x1,y2) = 0;
F1(x2:x1,y1) = 0;
F1(x1,y1:y2) = 0;
F1(x2,y1:y2) = 0;
% needs abit more padding at the corners
F1(x1-3:x1+3,y1-3:y1+3) = 0;
F1(x2-3:x2+3,y2-3:y2+3) = 0;
figure;
figure;
subplot(1,2,1);imagesc((S.^0.1));axis off; title("original image power specturm");
subplot(1,2,2);imagesc((abs(F1).^0.1));axis off;title("new power spectrum");
figure;
subplot(1,3,1);imshow(image);axis off;title("Original Image");
subplot(1,3,2);imshow(ifft2(F),[]);axis off;title("new Image");
subplot(1,3,3);imshow(ifft2(F1),[]);axis off;title("Futher enhanced new Image");
```

Figure 29: Code for section 5 (TV part)

```
%% section 2.5 part e
image =  imread('primatecaged.jpg');
whos image
% convert to grayscale
image = rgb2gray(image);
F = fft2(image);
temp = abs(F) .^ 0.1;
% so the cut off freq will be 3.2, anything greater than that has to be
% removed
temp = temp < 3.3 ;
templ = temp.* F ;

subplot(2,2,1);imshow(image);axis off;title("Original Image");
subplot(2,2,2);imagesc(fftshift(abs(F).^0.1));axis off;title("Specturm of original image wioth fftshift");
subplot(2,2,3);imshow(ifft2(templ),[]);axis off;title("Futher enhanced new Image");
subplot(2,2,4);imagesc(fftshift(abs(templ).^0.1));axis off;title("specturm of new image with fftshift");
%%
figure;
subplot(1,3,1);imshow(image);axis off;title("Original Image");
subplot(1,3,2);imshow(ifft2(Fl),[]);axis off;title("Enhanced Image using previous method");
subplot(1,3,3);imshow(ifft2(templ),[]);axis off;title("Enhanced Image using low pass filter");
```

Figure 30: Low pass filter method to remove primate in section 5

```
%% section 2.5 but another method which doesnt blur the monkey
image =  imread('primatecaged.jpg');
whos image
% convert to grayscale
image = rgb2gray(image);
F = fft2(image);
Fl = F;
x1 = 251;x2 = 5;y1 = 11;y2 = 247;
x3 = 248;x4 = 11;y3 =22;y4 = 236;
Fl(x1-2:x1+2,y1-2:y1+2) = 0;
Fl(x2-2:x2+2,y2-2:y2+2) = 0;
Fl(x3-2:x3+2,y3-2:y3+2) = 0;
Fl(x4-2:x4+2,y4-2:y4+2) = 0;

subplot(2,2,1);imshow(image);axis off;title("Original Image");
subplot(2,2,2);imagesc(fftshift(abs(F).^0.1));axis off;title("Specturm of original image wioth fftshift");
subplot(2,2,3);imshow(ifft2(Fl),[]);axis off;title("Futher enhanced new Image");
subplot(2,2,4);imagesc(fftshift(abs(Fl).^0.1));axis off;title("specturm of new image with fftshift");
```

Figure 31: Code for removing cage from primate

```
%% section 2.6
image =imread('book.jpg');
imshow(image);
[X, Y] = ginput(4);
x = [ 1 210 210 1];
y=  [ 1 1 297 297];

% A is the actual corner data while v is the potential corner data
%%
% getting the v matrix
for i = 1: 8
    if(mod(i,2)==1)
        v(i) = x( (i+1) /2 );
    else
        v(i) = y(i/2);
    end
end
v = v' ;
% now get the A matrix
count = 1;
for i = 1:8
    if(mod(i,2)==0)
        A(i,:) = [ 0 0 0 X(count) Y(count) 1 -y(count)*X(count) -y(count)*Y(count)] ;

        count = count+1;
    else
        A(i,:) = [ X(count) Y(count) 1 0 0 0 -x(count)*X(count) -x(count)*Y(count)] ;
    end
end

u = A\v;
U = reshape([u;1], 3, 3)';
w = U*[X'; Y'; ones(1,4)];
w = w ./ (ones(3,1) * w(3,:));
T = maketform('projective', U');
P2 = imtransform(image, T, 'XData', [0 210], 'YData', [0 297]);
figure;
subplot(1,2,1);imshow(image);axis off;title("Original Image");
subplot(1,2,2);imagesc(P2);axis off;title("Transformed Image");
```

Figure 32: Code for Planar Transformation of the book in section 6

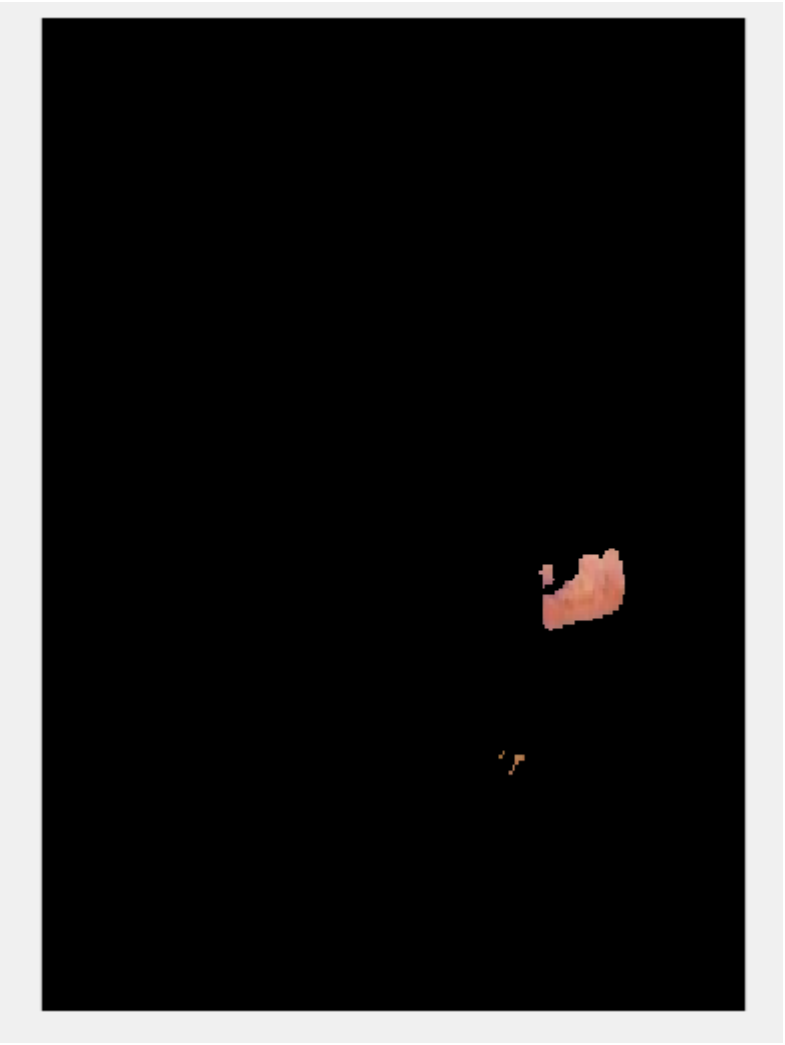

Figure 33: Segmented Image

```
numColors = 8;
L = imsegkmeans(P2,numColors);
B = labeloverlay(P2,L);


lab_he = rgb2lab(P2);
ab = lab_he(:,:,2:3);
ab = im2single(ab);
pixel_labels = imsegkmeans(ab,numColors);
B2 = labeloverlay(P2,pixel_labels);


mask3 = pixel_labels == 2;
cluster3 = P2.*uint8(mask3);

% but there is some other stuff that needs to be remoevd
L = lab_he(:,:,1);
L_red = L.*double(mask3);
L_red = rescale(L_red);
idx_light_red = imbinarize(nonzeros(L_red));
red_idx = find(mask3);
red_mask = zeros(298,211);
red_mask(red_idx(idx_light_red)) = 1;

red_image = P2.*uint8(red_mask);

subplot(2,2,1);imshow(B);axis off;title("Clustering in RGB space");
subplot(2,2,2);imshow(B2);axis off;title("Clustering in L*a*b* space");
subplot(2,2,3);imshow(cluster3);axis off;title("Red Colour Mask");
subplot(2,2,4);imshow(red_image);axis off;title("Final Mask");
```

Figure 34: Code for colour segmentation in section 6