CE4003 Computer Vision
LAB REPORT

Student Name: Wang Xinyu
Matriculation Number: U1920825J

Load target folder

Source Code:

```
cd /Users/ricardo/Desktop/CE4003/Lab1/images
```

Q2.1 Contrast Stretching

Source Code:

```
Pc = imread('mrt-train.jpg');
subplot(1,2,1);
imshow(Pc);
title('Original Image');
P = rgb2gray(Pc);
P = cast(P,'double');
minP = min(P(:));
maxP = max(P(:));
P2 = P;
P2 = imsubtract(P2,minP);
P2 = immultiply(P2,255/(maxP-minP));
P2 = cast(P2, 'uint8');
min(P2(:)), max(P2(:))
subplot(1,2,2);
imshow(P2);
title('Result Image');
```

Result Images:

maxP	204
minP	13

*In greyscale, The intensity of brightest pixel is 204 and the intensity of darkest pixel is 13. This shows that the original image doesn't occupy the whole interval of histogram(0-255).

Comments:

The histogram of the original image is not spread in the whole interval(0-255), thus there are very few pixels for highlights and shadow portions of the image, which yields a relatively low contrast. By mapping the maxP to 255 and minP to 0, we manually stretch the histogram into scale of 0-255, which increased the contrast of the original image and the result image now looks better(less greish)

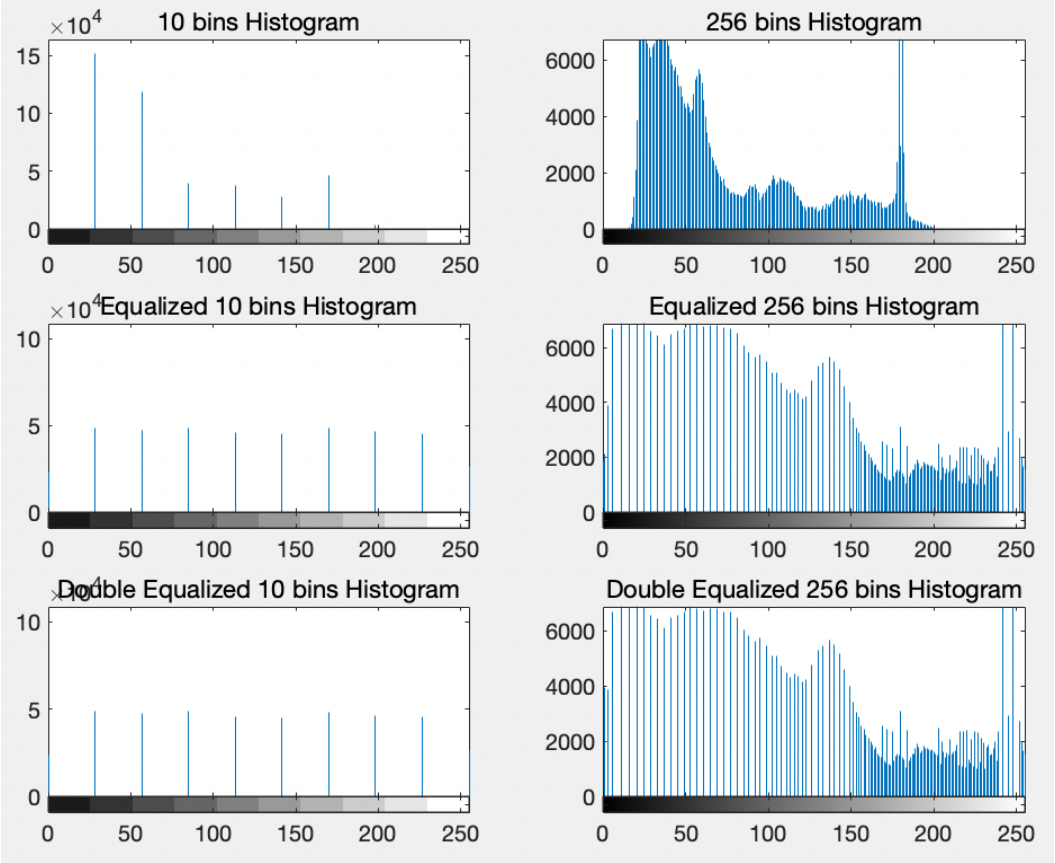
Q2.2 Histogram Equalization

Source Code:

```
subplot(3,2,1);
imhist(Pc,10);
title('10 bins Histogram');
subplot(3,2,2);
imhist(Pc,256);
title('256 bins Histogram');
P3 = histeq(Pc,255);
subplot(3,2,3);
imhist(P3,10);
title('Equalized 10 bins Histogram');
subplot(3,2,4);
imhist(P3,256);
title('Equalized 256 bins Histogram');
P3_refine = histeq(P3,255);
subplot(3,2,5);
imhist(P3_refine,10);
```

```
title('Double Equalized 10 bins Histogram');
subplot(3,2,6);
imhist(P3_refine,256);
title('Double Equalized 256 bins Histogram');
```

Result Images:



Comments:

Given the original image with unequal histogram bins, the result is drastically improved after the first implication of *histeq*. But after the second time of *histeq,* the improvement is almost invisible. The reason for it might be that when we apply *histeq* for the first time, the function has already outputted the best result, thus the second time of implication would not have any influence on it.

Q2.3 Linear Spatial Filtering

Source Code:

```
filter1 = getUnormalizedFilter(5,1.0);
filter1_nor = getNormalizedFilter(filter1);
% mesh(filter1_nor)
```

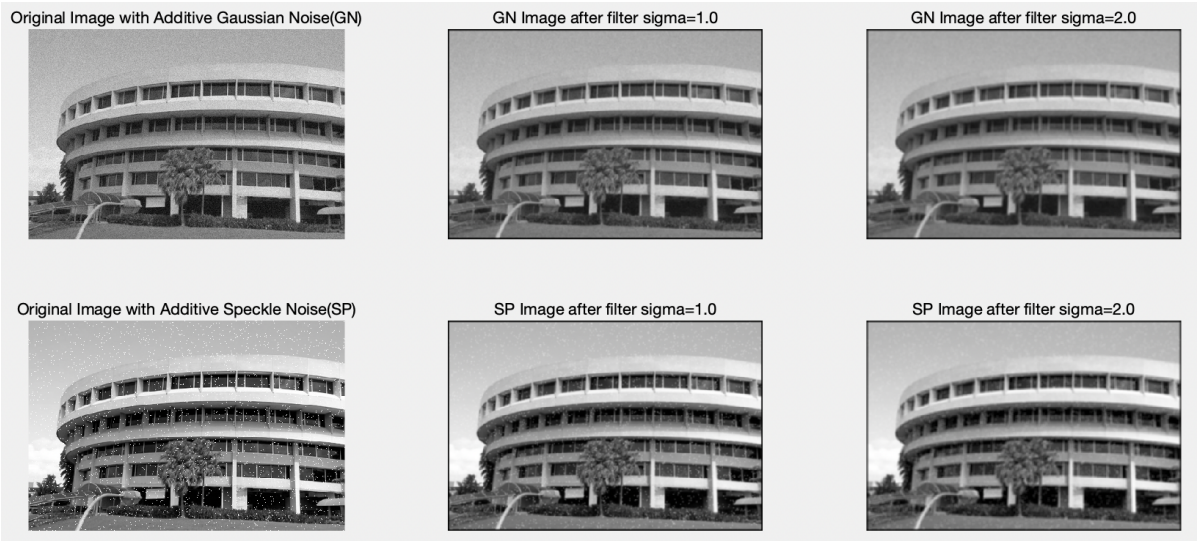
```
% title('filter 1');

filter2 = getUnormalizedFilter(5,2.0);
filter2_nor = getNormalizedFilter(filter2);
% mesh(filter2_nor)
% title('filter 2');

P_gn = imread('ntugn.jpeg');
P_sp = imread('ntusp.jpeg');
subplot(2,3,1);
imshow(P_gn);
title('Original Image with Additive Gaussian Noise(GN)')
subplot(2,3,4);
imshow(P_sp);
title('Original Image with Additive Speckle Noise(SP)')
gn_result_img_filter1 = cast(P_gn,'double');
gn_result_img_filter1 = conv2(gn_result_img_filter1, filter1_nor);
gn_result_img_filter1 = cast(gn_result_img_filter1, 'uint8');
subplot(2,3,2);
imshow(gn_result_img_filter1);
title('GN Image after filter sigma=1.0')
sp_result_img_filter2 = cast(P_gn,'double');
sp_result_img_filter2 = conv2(sp_result_img_filter2, filter2_nor);
sp_result_img_filter2 = cast(sp_result_img_filter2, 'uint8');
subplot(2,3,3);
imshow(sp_result_img_filter2);
title('GN Image after filter sigma=2.0')
sp_result_img_filter1 = cast(P_sp,'double');
sp_result_img_filter1 = conv2(sp_result_img_filter1, filter1_nor);
sp_result_img_filter1 = cast(sp_result_img_filter1, 'uint8');
subplot(2,3,5);
imshow(sp_result_img_filter1);
title('SP Image after filter sigma=1.0')
sp_result_img_filter2 = cast(P_sp,'double');
sp_result_img_filter2 = conv2(sp_result_img_filter2, filter2_nor);
sp_result_img_filter2 = cast(sp_result_img_filter2, 'uint8');
subplot(2,3,6);
imshow(sp_result_img_filter2);
title('SP Image after filter sigma=2.0')
```

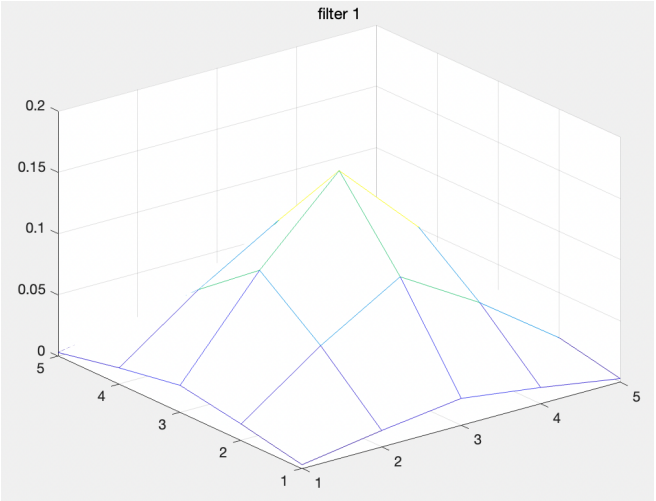
Function Declaration:

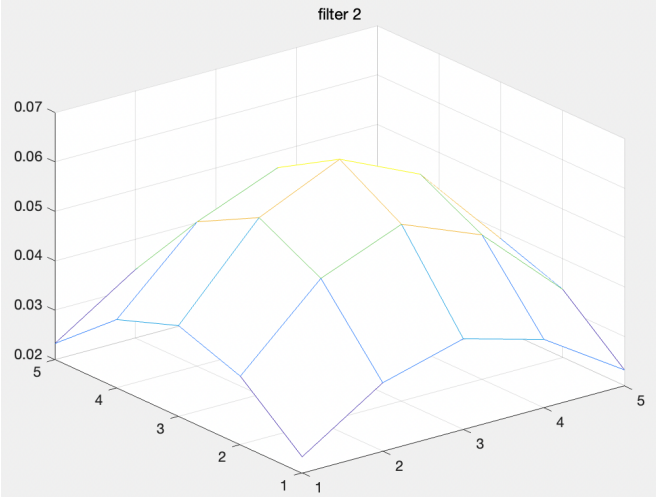
```
function filter_normalized = getNormalizedFilter(filter_unormalized)
   filter_normalized = filter_unormalized;
   filter_sum = sum(filter_normalized,"all");
   filter_normalized = immultiply(filter_normalized, 1/filter_sum);
end

function filter_unormalized = getUnormalizedFilter(dimension, variance)
   filter_unormalized = zeros(dimension);
   for x = 1:dimension
       for y = 1:dimension
```

```
            filter_unormalized(x,y) =
getGaussianValue(variance,x-(dimension+1)/2,y-(dimension+1)/2);
        end
    end
end

function h = getGaussianValue(variance, x, y)
    v2 = variance.^2;
    h = 1/(2*pi*v2)*exp(-(x.^2 + y.^2)/(2*v2));
end
```

Result Images:







Comments:

The larger the dimension of the filter is, the stronger the power of “blur” would be. By results, our filters are more suitable to deal with Gaussian Noise as it is more evenly distributed and unlike speckle noise with sudden change, it is more gentle. And since our filters are actually a simulation of a gaussian filter, they should have similar performance as a gaussian filter as well.

Q2.4 Median filtering

Source Code:

```
P_gn = imread('ntugn.jpeg');
P_sp = imread('ntusp.jpeg');
subplot(2,3,1);
imshow(P_gn);
title('Original Image with Additive Gaussian Noise(GN)');
```

```
subplot(2,3,4);
imshow(P_sp);
title('Original Image with Additive Speckle Noise(SP)');
subplot(2,3,2);
gn_medianfilter_result_3x3 = medfilt2(P_gn,[3,3]);
imshow(gn_medianfilter_result_3x3);
title('GN Image after median filter 3x3');
subplot(2,3,3);
gn_medianfilter_result_5x5 = medfilt2(P_gn,[5,5]);
imshow(gn_medianfilter_result_5x5);
title('GN Image after median filter 5x5');
subplot(2,3,5);
sp_medianfilter_result_3x3 = medfilt2(P_sp,[3,3]);
imshow(sp_medianfilter_result_3x3);
title('SP Image after median filter 3x3');
subplot(2,3,6);
sp_medianfilter_result_5x5 = medfilt2(P_sp,[5,5]);
imshow(sp_medianfilter_result_5x5);
title('SP Image after median filter 5x5');
```

Result Images:

Comments:

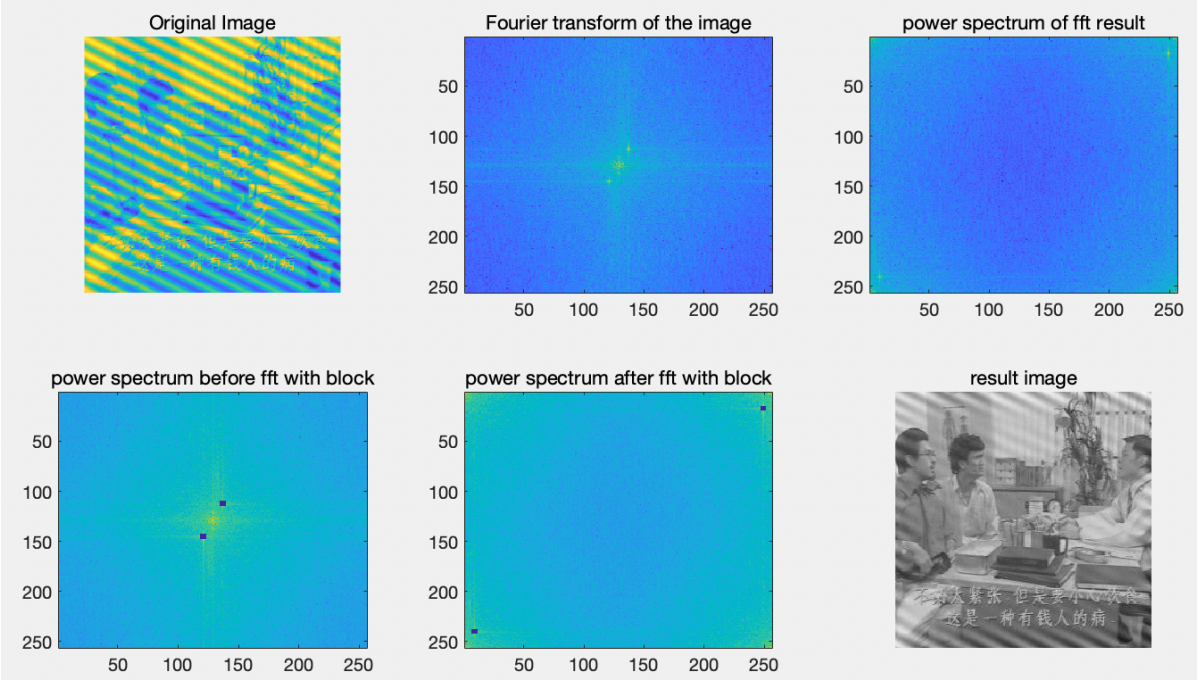
Median Filter is good to deal with speckle noise, or more generally, it is good to deal with any noise which is sharp and suddenly changed. But the disadvantage of using a median filter is it would blur the edges in images, making the sharpness of image decreased, which would yield an undesired result sometimes.

Q2.5 Suppressing Noise Interference Patterns Part a-e

Source Code:

```
P_int = imread('pckint.jpeg');
subplot(2,3,1);
imshow(P_int);
title('Original Image');
F = fft2(P_int);
S = abs(F);
subplot(2,3,2);
imagesc(fftshift(S.^0.1));
colormap('default');
title('Fourier transform of the image');
subplot(2,3,3);
imagesc(S.^0.1);
title('power spectrum of fft result');
F(238:242,7:11) = 0;
F(15:19, 247:251) = 0;
S = abs(F);
subplot(2,3,4);
imagesc(fftshift(S.^0.1));
colormap('default');
title('power spectrum before fft with block');
subplot(2,3,5);
imagesc(S.^0.1);
title('power spectrum after fft with block')
result_img = ifft2(F);
subplot(2,3,6);
result_img = cast(result_img,'uint8');
imshow(result_img);
title('result image');
```

Result Images:



Comments:

Part e: By filtering out the peaks other than origin in the power spectrum, we filter out certain patterns in the original images(the diagonal strips). There are still some more strips left other than the central area. If we filter out more areas in the power spectrum, we should be able to get a better result.

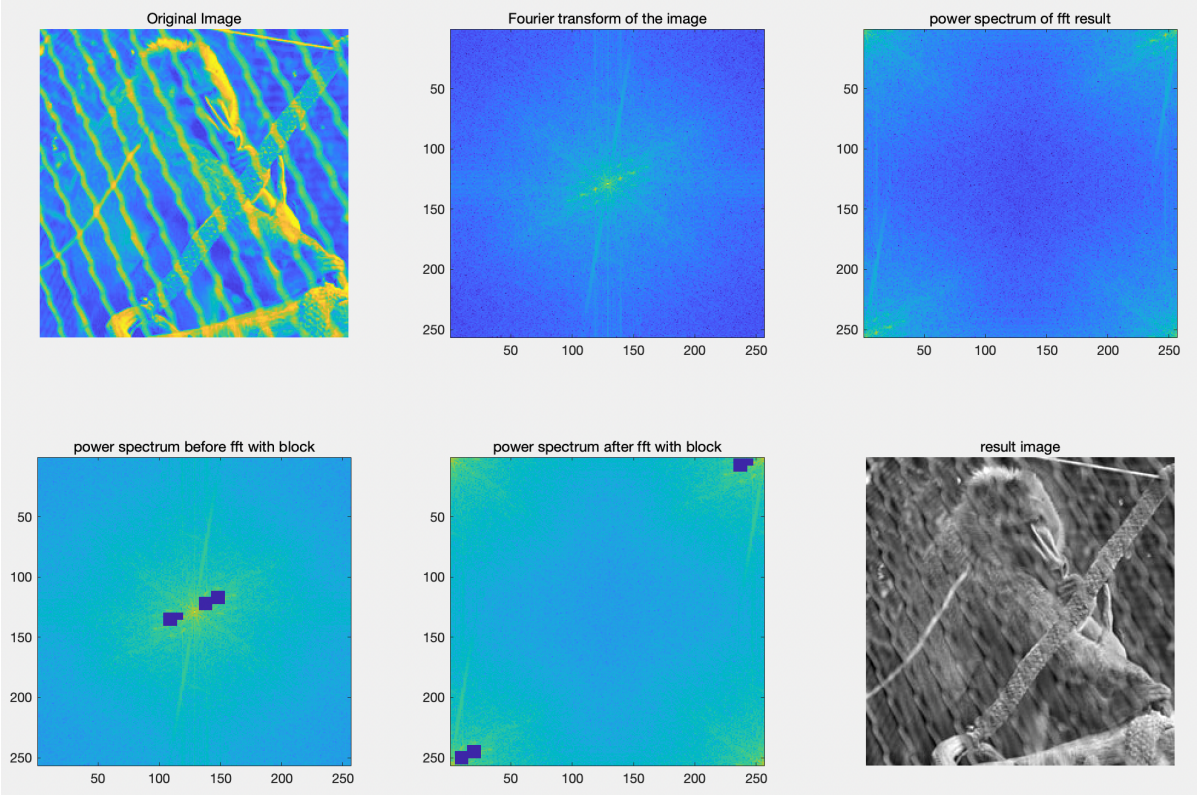
Q2.5 Suppressing Noise Interference Patterns
Part f

Source Code:

```
P_int = imread('primatecaged.jpeg');
P_int = rgb2gray(P_int);
P_int = cast(P_int, 'uint8');
subplot(2,3,1);
imshow(P_int);
title('Original Image');
F = fft2(P_int);
S = abs(F);
subplot(2,3,2);
imagesc(fftshift(S.^0.1));
colormap('default');
title('Fourier transform of the image');
subplot(2,3,3);
imagesc(S.^0.1);
```

```
title('power spectrum of fft result');
F(240:250,15:25) = 0;
F(245:255,5:15) = 0;
F(2:12,232:242) = 0;
F(2:7,242:247) = 0;
%[X,Y] = ginput(4);
S = abs(F);
subplot(2,3,4);
imagesc(fftshift(S.^0.1));
colormap('default');
title('power spectrum before fft with block');
subplot(2,3,5);
imagesc(S.^0.1);
title('power spectrum after fft with block')
result_img = ifft2(F);
subplot(2,3,6);
result_img = cast(result_img,'uint8');
imshow(result_img);
title('result image');
```

Result Images:



Comments:

With some attempts to block certain areas in the power spectrum, the resulting image is slightly better than the original. But since the fence is a real object, it is not solely in one pattern, thus it is not represented in one peak of frequency in the power spectrum. So although after blocking the major four peaks, the fence is still visible.

Q2.6 Undoing Perspective Distortion of Planar Surface

Source Code:

```
%part a
P = imread('book.jpeg');
subplot(1,2,1);
imshow(P);
title('Original Image');
%part b
%[X,Y] = ginput(4);
X = [2.3850,256.6357,308.2097,143.5349];
Y = [158.9543,215.5047,46.7582,27.7573];
X_transformed = [0,210,210,0];
Y_transfromed = [297,297,0,0];
%part c
A = matrixA(X,Y,X_transformed,Y_transfromed);
v = matrixv(X_transformed, Y_transfromed);
u = A \ v;
U = reshape([u;1],3,3)';
w = U*[X;Y;ones(1,4)];
w = w./(ones(3,1)*w(3,:));
%part d
T = maketform('projective',U');
P_double = cast(P,'double');
P2 = imtransform(P_double,T,'XDATA',[0 210],'YDATA',[0 297]);
P2 = cast(P2,'uint8');
%part e
subplot(1,2,2);
imshow(P2);
title('Transformed Image');
```

Function Declaration:

```
function A = matrixA(X,Y,X_transformed,Y_transformed)
   A = zeros(8,8);
   for idx = 1:1:4
       row_idx = idx * 2 - 1;
       A(row_idx,1) = X(idx);
       A(row_idx,2) = Y(idx);
       A(row_idx,3) = 1;
       A(row_idx,7) = -1*X_transformed(idx)*X(idx);
       A(row_idx,8) = -1*X_transformed(idx)*Y(idx);
       even_idx = idx*2;
       A(even_idx,4) = X(idx);
       A(even_idx,5) = Y(idx);
       A(even_idx,6) = 1;
```

```
        A(even_idx,7) = -1*Y_transformed(idx)*X(idx);
        A(even_idx,8) = -1*Y_transformed(idx)*Y(idx);
    end
end

function v = matrixv(X_transformed,Y_transformed)
    v = zeros(8,1);
    for idx = 1:1:4
        v(idx*2-1,1) = X_transformed(idx);
        v(idx*2,1) = Y_transformed(idx);
    end
end
```

Result Images:

Comments:

The resulting image is the expected result. But due to blur of further parts of the book in the original image, the quality of the upper part of the wrapped image is not ideal.

Q2.6 part f

Source Code:

```
subplot(1,2,1);
```

```
imshow(P2);
img_converted = cast(P2,'double');
title('Original Image');
img_new = 255*zeros(size(img_converted));
[H,W,dimension] = size(img_converted);
for i = 1:H
   for j = 1:W
       if img_converted(i,j,1)>=158 && img_converted(i,j,1)<=225 ...
           && img_converted(i,j,2)>=105 && img_converted(i,j,2)<=176 ...
           && img_converted(i,j,3)>=61 && img_converted(i,j,3)<=171
           img_new(i,j,1) = img_converted(i,j,1);
           img_new(i,j,2) = img_converted(i,j,2);
           img_new(i,j,3) = img_converted(i,j,3);
       end
  end
end
for i = 1:H
   for j = 1:W
       if i<157 || i>184 || j<148 || j>177
           img_new(i,j,:) = 0;
       end
   end
end
subplot(1,2,2);
img_new = cast(img_new, 'uint8');
imshow(img_new);
title('Captured Image');
```

Result Images: