DIGITAL IMAGE FORMATION AND ENHANCEMENT

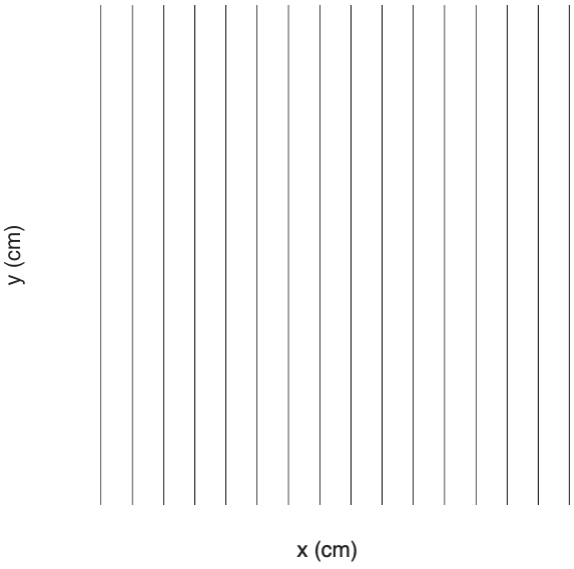
Submitted by: Jonabel Eleanor B. Baldres

Section: App Physics 157 WFY FX 2

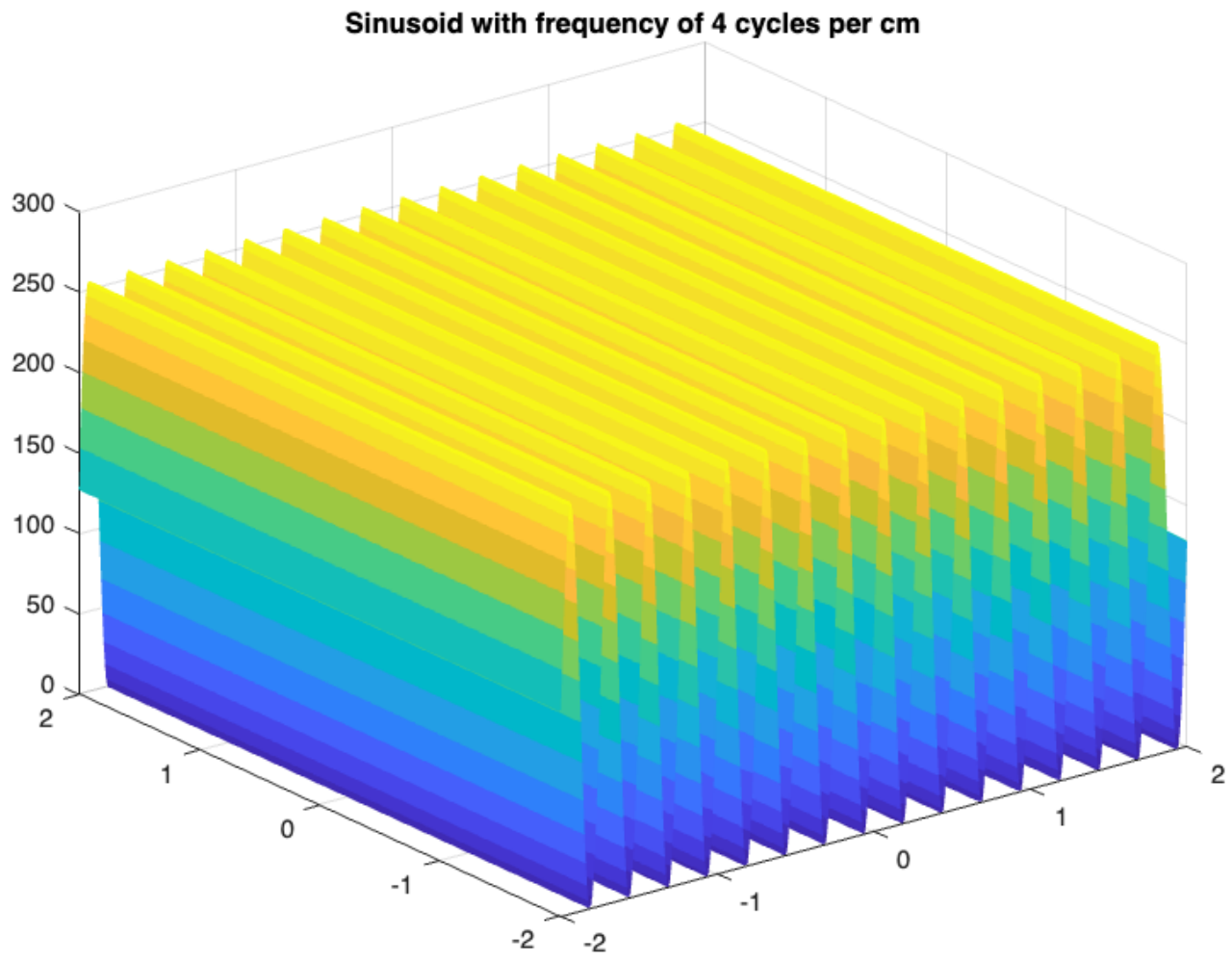
Activity 1.1 Image DIY

Sinusoid along the x-direction with frequency of 4 cycles per centimeter.

```
N = 500;
x = linspace(-2,2,N);
y = x;
[X,Y] = meshgrid(x,y);
R = sin(8*X*pi);
B = rescale(R,0,255);
figure;
imshow(B); % rescaled
xlabel("x (cm)")
ylabel("y (cm)")
```

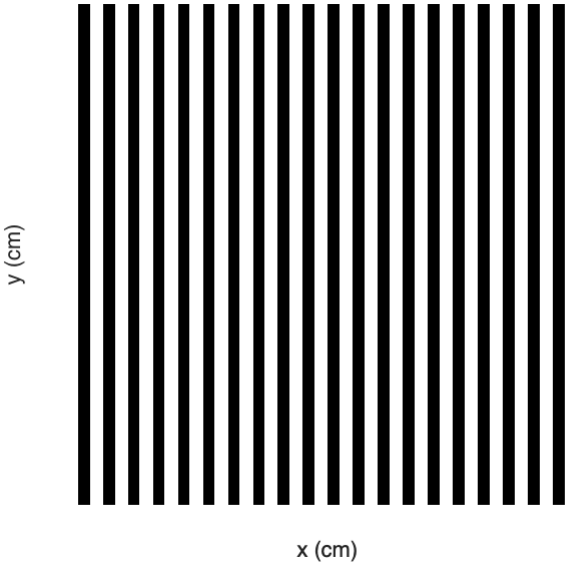



```
figure;
mesh(x,y,B) %rescaled
title("Sinusoid with frequency of 4 cycles per cm")
```

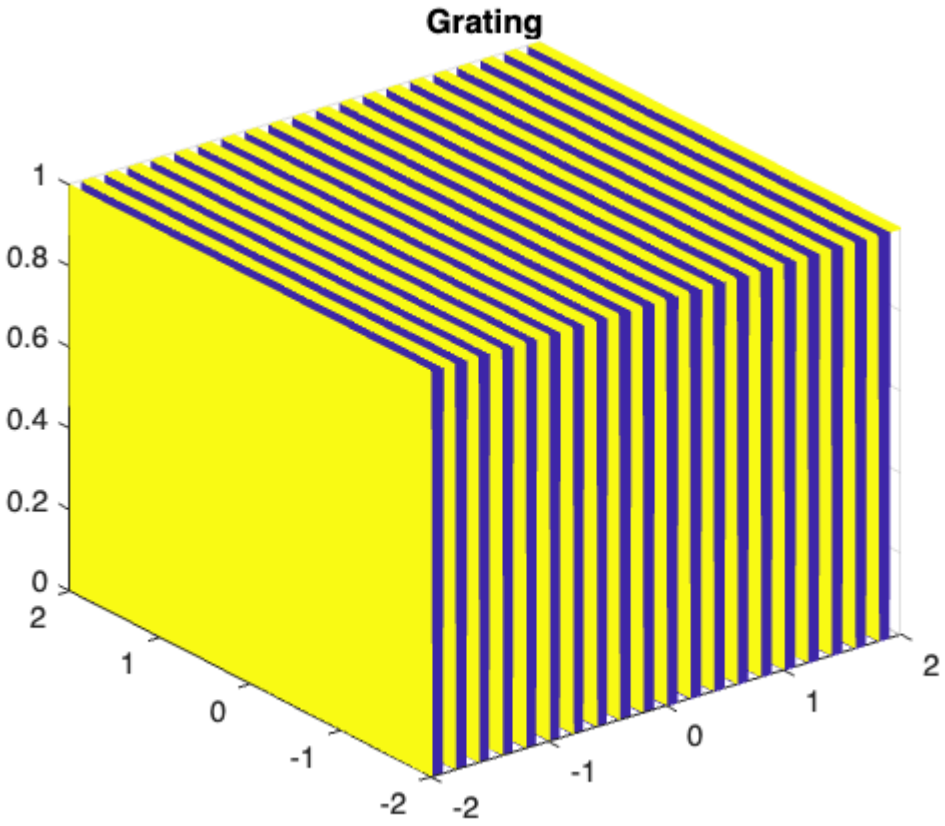


Grating with frequency of 5 line pairs per centimeter

```
N = 500;
x = linspace(-2,2,N);
y = x;
[X,Y] = meshgrid(x,y);
R = sin(10*X*pi);
A = zeros(size(R));
A(R<0.1) = 1;
figure(1);
imshow(A);

xlabel("x (cm)")
ylabel("y (cm)")
```

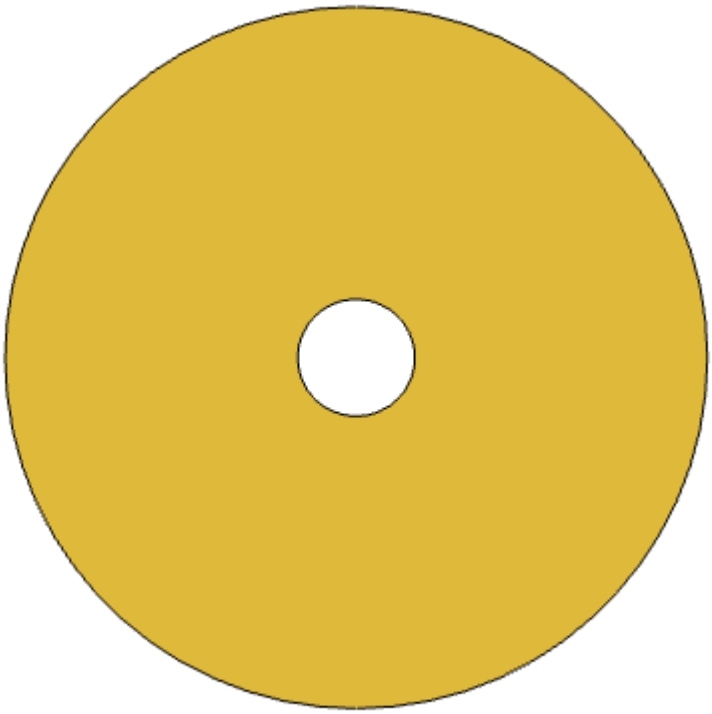



```
figure(2);
mesh(x,y,A);
title("Grating")
```

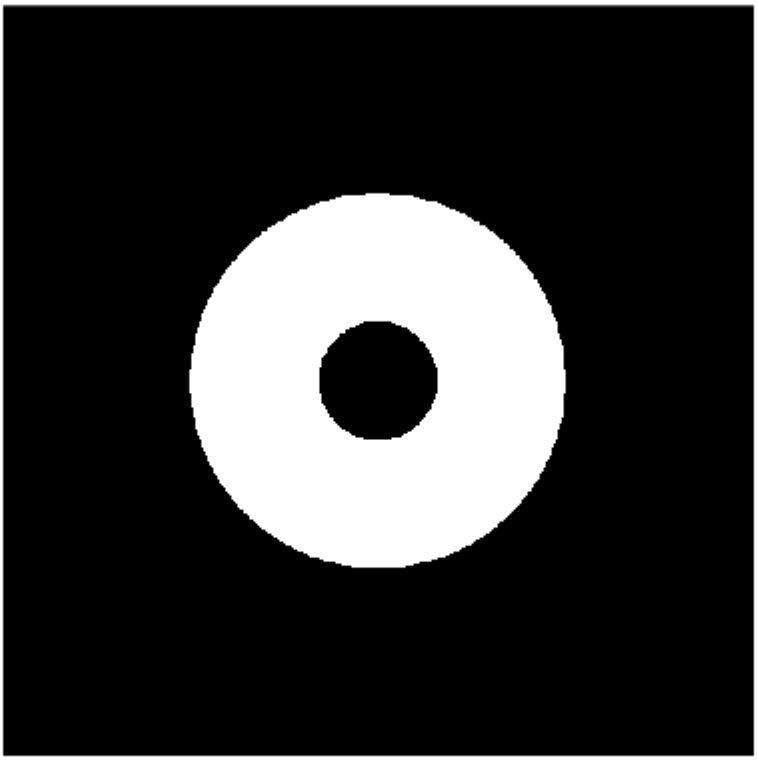


Hubble's Primary Mirror

```
% First method is by using the ringAnnular package
hubble = ringAnnular(InnerRadius=1,Width=5);
show(hubble)
axis off
```



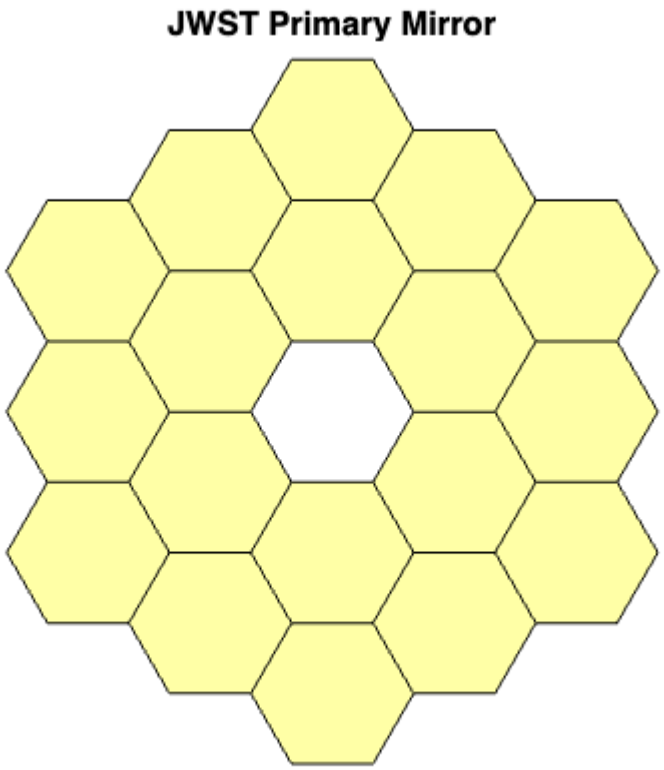
```
% Second method is by creating an equation that results to an annulus
N = 500;
x = linspace(-2,2,N);
y = x;
[X,Y] = meshgrid(x,y);
R = (X.^2 + Y.^2);
A = zeros(size(R));
A(R<1) = 1;
A(R< 0.1) = 0;
figure(1);
imshow(A);
```



Hexagon Array

```
%used nsidedpoly to create the hexagon
hex2 = nsidedpoly(6, 'Center', [0,3.4641], 'SideLength', 2);
hex3 = nsidedpoly(6, 'Center', [0,6.9282], 'SideLength', 2);
hex4 = nsidedpoly(6, 'Center', [0,-3.4641], 'SideLength', 2);
hex5 = nsidedpoly(6, 'Center', [0,-6.9282], 'SideLength', 2);
hex6 = nsidedpoly(6, 'Center', [3,-1.73205], 'SideLength', 2);
hex7 = nsidedpoly(6, 'Center', [3,1.73205], 'SideLength', 2);
hex8 = nsidedpoly(6, 'Center', [3, -5.19615161512], 'SideLength', 2);
hex9 = nsidedpoly(6, 'Center', [3, 5.19615161512], 'SideLength', 2);
hex10 = nsidedpoly(6, 'Center', [-3,-1.73205], 'SideLength', 2);
hex11 = nsidedpoly(6, 'Center', [-3,1.73205], 'SideLength', 2);
hex12= nsidedpoly(6, 'Center', [-3, -5.19615161512], 'SideLength', 2);
hex13= nsidedpoly(6, 'Center', [-3, 5.19615161512], 'SideLength', 2);
hex14 = nsidedpoly(6, 'Center', [-6,3.4641], 'SideLength', 2);
hex15 = nsidedpoly(6, 'Center', [-6,0], 'SideLength', 2);
hex16= nsidedpoly(6, 'Center', [-6,-3.4641], 'SideLength', 2);
hex17 = nsidedpoly(6, 'Center', [6,3.4641], 'SideLength', 2);
hex18 = nsidedpoly(6, 'Center', [6,0], 'SideLength', 2);
hex19= nsidedpoly(6, 'Center', [6,-3.4641], 'SideLength', 2);
plot([ hex2 hex3 hex4 hex5 hex6 hex7 hex8 hex9 hex10 hex11 hex12 hex13 hex14 hex15 hex
axis equal
```

```
axis off
title("JWST Primary Mirror")
```

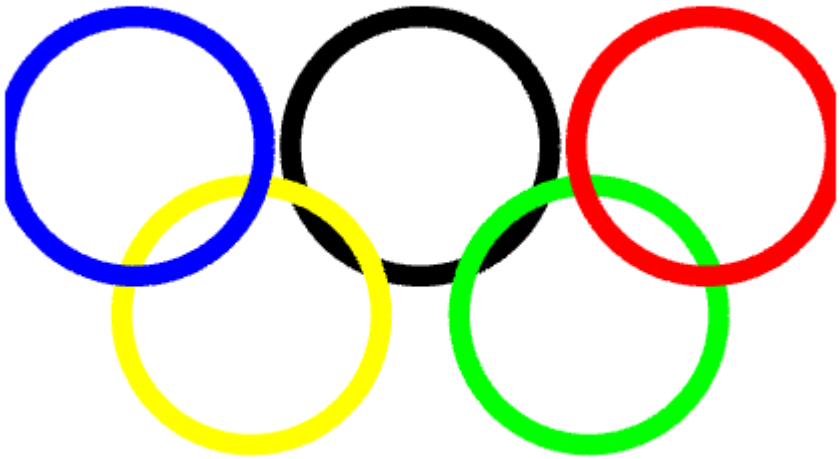


Activity 1.2 Color Image

Mathematically recreating the Olympics logo as an image

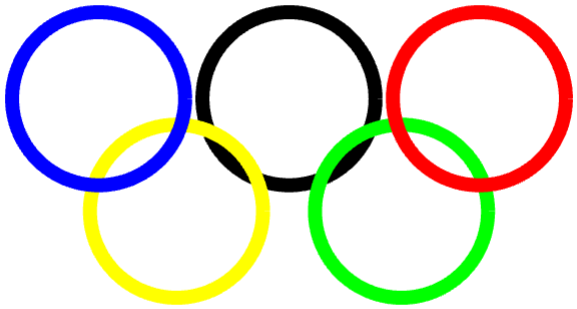
```
a = linspace(0,2*pi);
% black ring
x= cos(a);
y = sin(a);
% yellow ring
x1 = cos(a) - 1.3 ;
y1 = sin(a) - 1.3 ;
%green ring
x2 = cos(a) + 1.3;
y2 = sin(a) - 1.3 ;
%red ring
x3 = cos(a) + 2.2;
y3 = sin(a);
%blue ring
x4 = cos(a) - 2.2 ;
y4 = sin(a);
```

```
plot(x,y,"LineWidth",7, "Color", "k");
hold on;
plot(x1,y1, "LineWidth", 7, "Color", "y");
plot(x2,y2, "LineWidth", 7, "Color", "g");
plot(x3,y3, "LineWidth", 7, "Color", "r");
plot(x4,y4, "LineWidth", 7, "Color", "b");
axis equal;
axis off;
hold off
```



Saving the image using different file formats

```
saveas(gcf, "OlympicRings.png");
saveas(gcf, "OlympicRings.jpg");
saveas(gcf, "OlympicRings.tif");
saveas(gcf, "OlympicRings", "bmp");
```



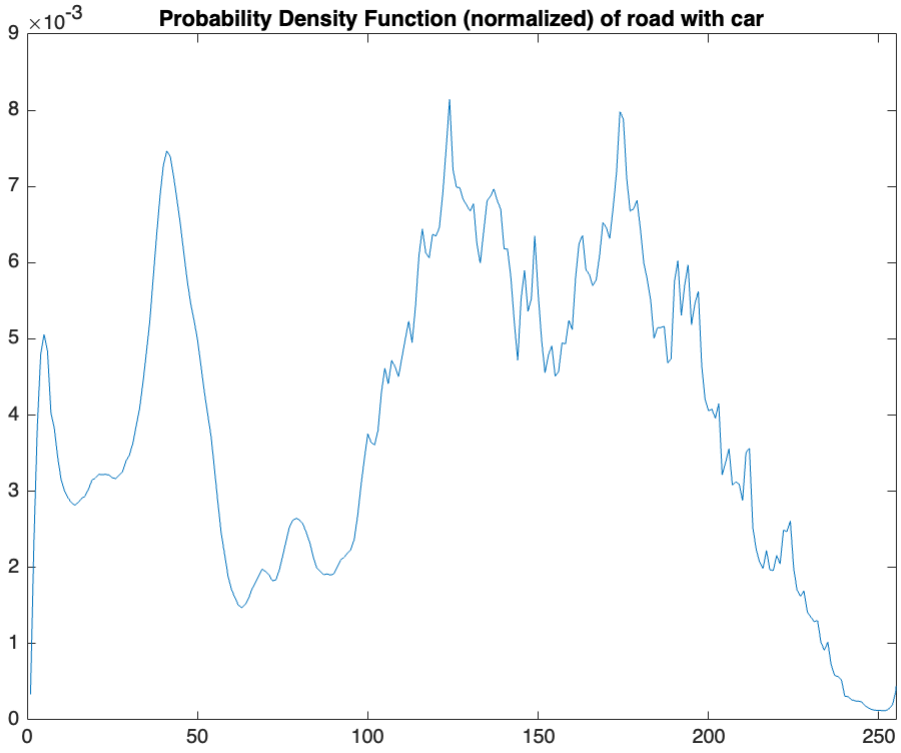
Activity 1.3 Altering the Input-Output Curve

Link for the Activity

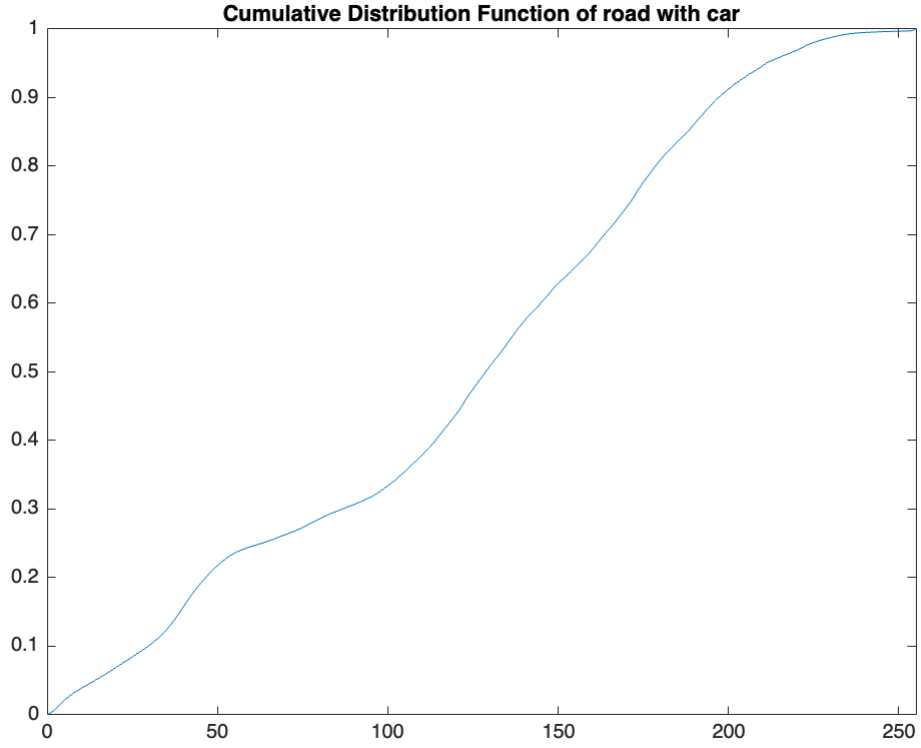
Activity 1.4 Histogram Backprojection on Grayscale Images

The images from left to right: Original Image, Image with linear desired CDF, Image with quadratic desired CDF, Image formed using histogram equalization function, and the Image formed using the adaptive histogram function.

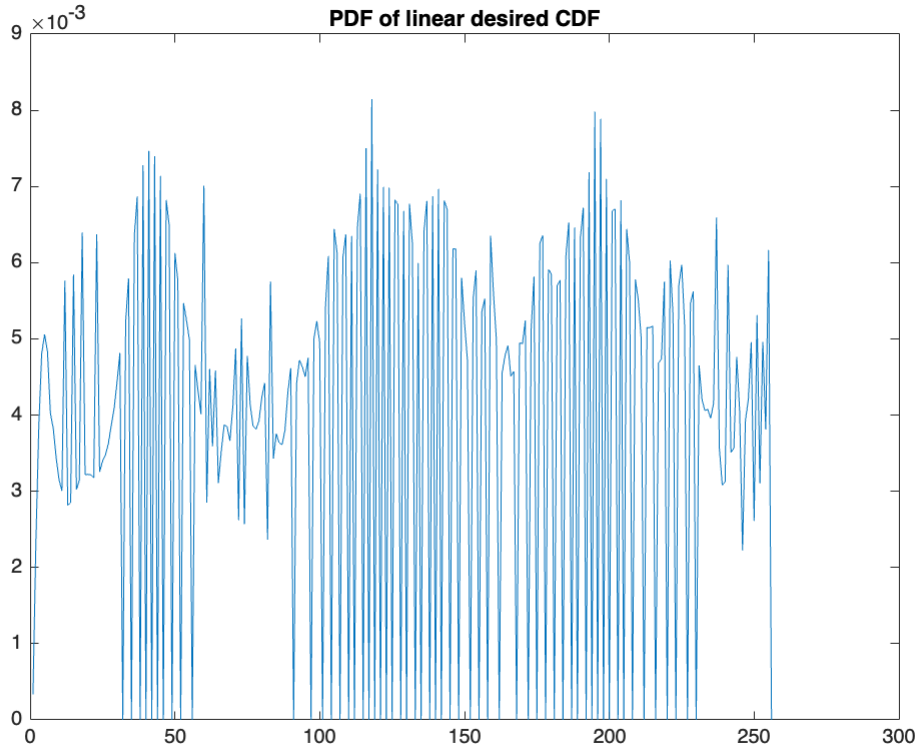
```
Igray = rgb2gray(imread("IMG_0202.png"));
PDF = hist(Igray(:),0:255) / numel(Igray);
CDF= cumsum(PDF);
figure; plot(PDF)

xlim([0 255])
ylim([0.00000 0.00900])
title("Probability Density Function (normalized) of road with car")
```



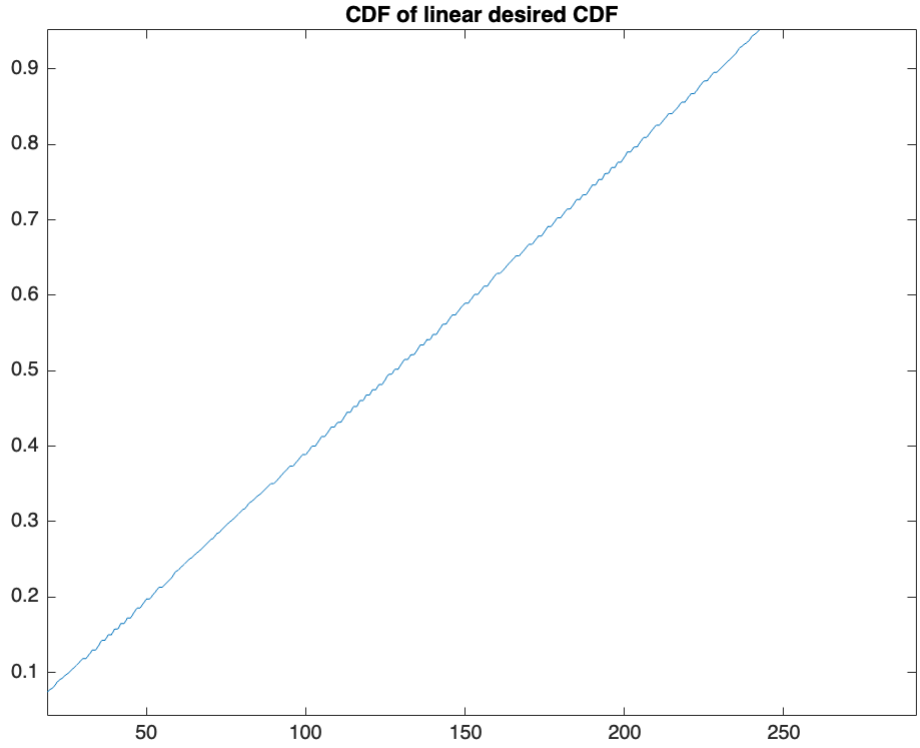

```
figure; plot(0:255, CDF);

xlim([0 255])
ylim([0.000 1.000])
title("Cumulative Distribution Function of road with car")
```



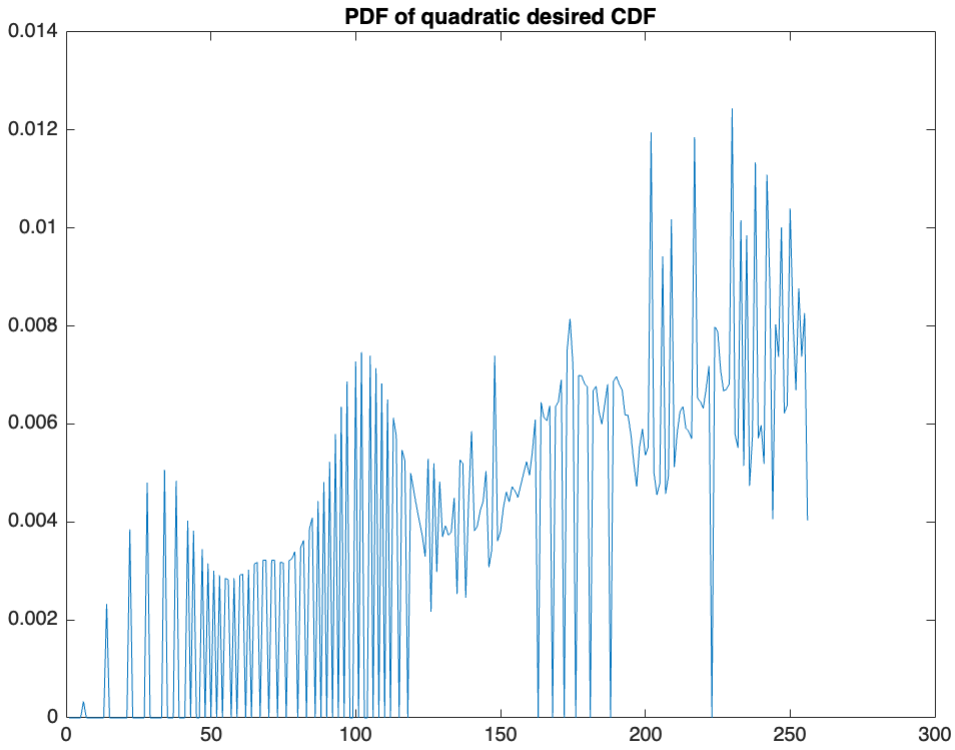

```
x = 0:255;
desiredCDF_linear = (1/255) .* x ;
desiredCDF_quad = ((1/255) .* x ) .^2 ;
newGS_linear = interp1(desiredCDF_linear, x, CDF(Igray(:)+1));
newGS_quad = interp1(desiredCDF_quad,x,CDF(Igray(:)+1));
Igraynew_linear = reshape(newGS_linear, size(Igray));
Igraynew_linear_uint8 = uint8(Igraynew_linear); %
Igraynew_quad = reshape(newGS_quad, size(Igray));
Igraynew_quad_uint8 = uint8(Igraynew_quad);
Igray_histogram = histeq(Igray);
Igray_histogram_uint8 = uint8(Igray_histogram);
Igray_adapt_histogram = adapthisteq(Igray);
Igray_adapt_histogram_uint8 = uint8(Igray_adapt_histogram);


PDF_linear = hist(Igraynew_linear(:), 0:255) / numel(Igraynew_linear);
CDF_linear = cumsum(PDF_linear);
figure; plot(PDF_linear)
title("PDF of linear desired CDF")
```



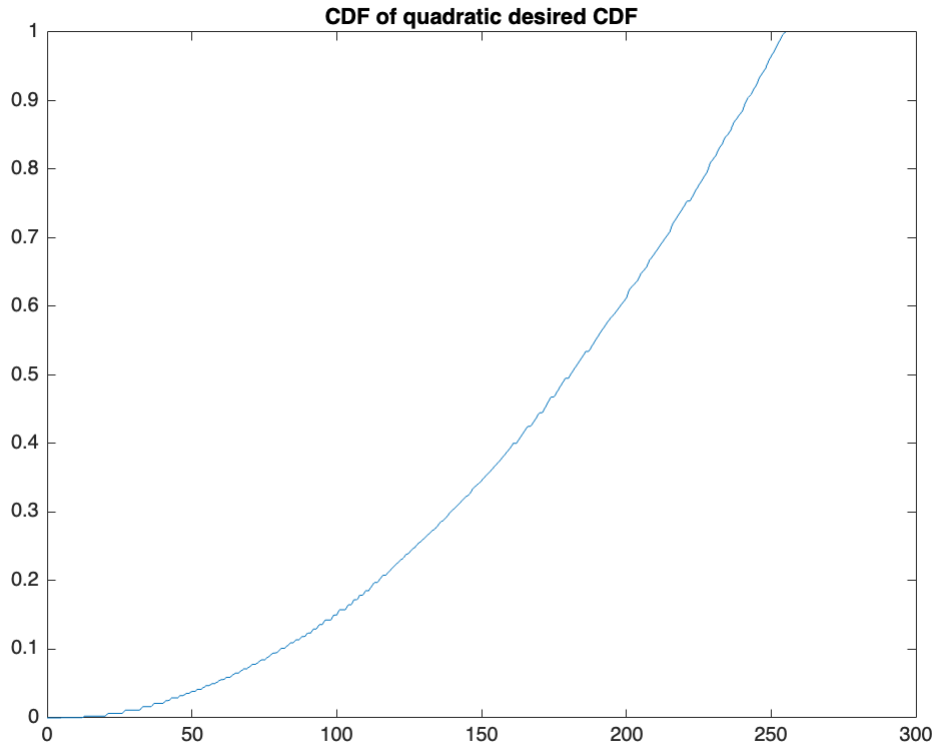

```
figure; plot(0:255,CDF_linear)
title("CDF of linear desired CDF")
```



```
PDF_quad = hist(Igraynew_quad(:), 0:255) / numel(Igraynew_quad);
CDF_quad = cumsum(PDF_quad);
figure; plot(PDF_quad)
title("PDF of quadratic desired CDF")
```




```
figure; plot(0:255,CDF_quad)
title("CDF of quadratic desired CDF")
```



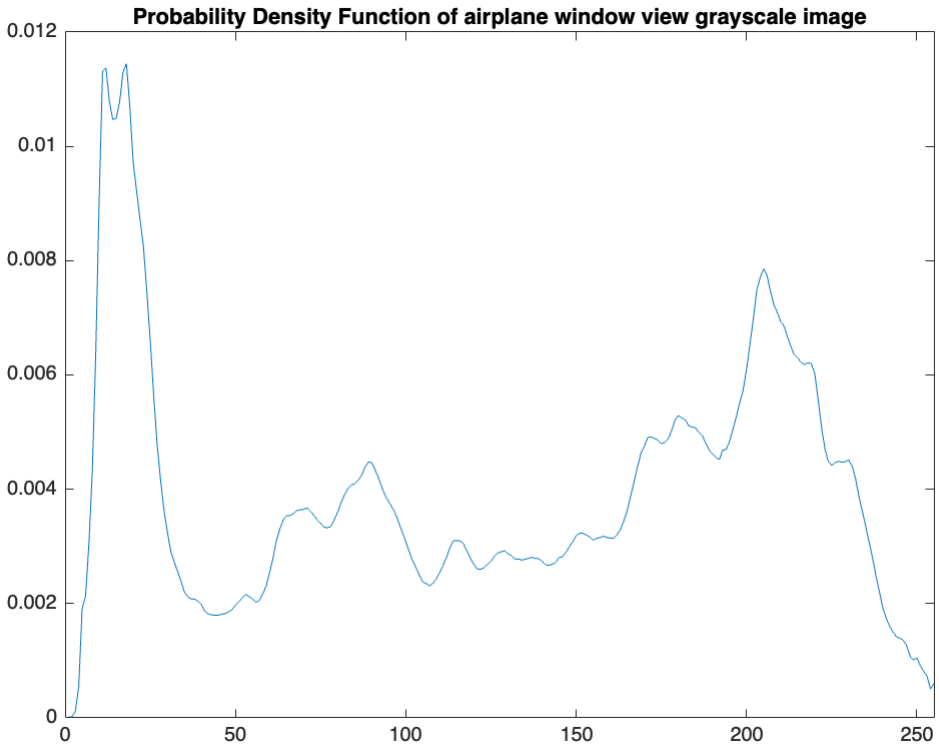

```
montage({Igray,Igraynew_linear_uint8, Igraynew_quad_uint8,Igray_histogram_uint8,Igray_
```

Activity 1.5 Contrast Enhancement

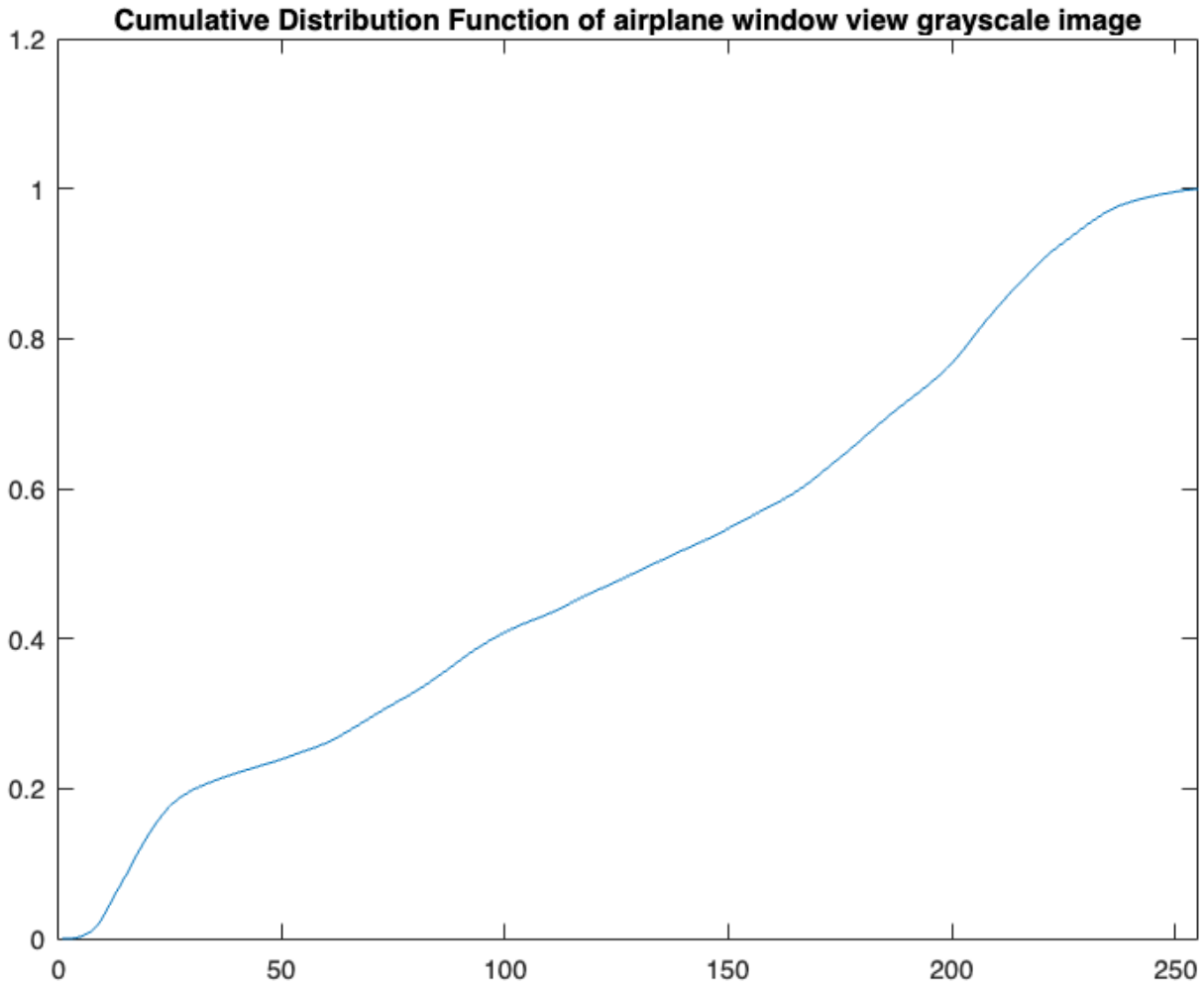
In this activity, I compared an contrast the result of contrast enhancement given the minimum and maximum values of the image and given different percentiles in the image.

```
Image = rgb2gray(imread("IMG_6653.JPG"));
PDF_Image = hist(Image(:),0:255) / numel(Image);
CDF_Image= cumsum(PDF_Image);
figure; plot(PDF_Image)

xlim([0 255])
```

```
ylim([0.00000 0.012])
title("Probability Density Function of airplanw window view grayscale image")

title("Probability Density Function of airplane window view grayscale image")
```

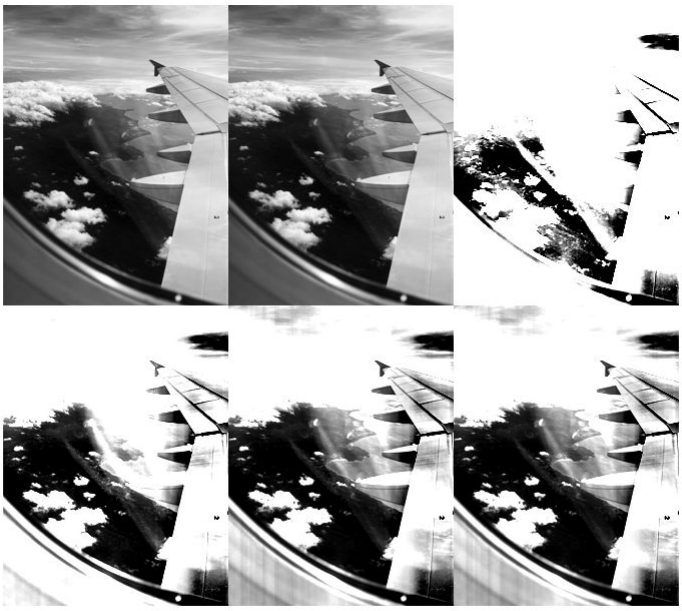



```
figure; plot(CDF_Image)

xlim([0 255])
ylim([0.00 1.20])
title("Cumulative Distribution Function of airplane window view grayscale image")
```




```
dImage = im2double(Image);
minf =@(dImage) min(dImage(:));
maxf =@(dImage)max(dImage(:));
dImg_min = minf(dImage);
dImg_max = maxf(dImage);
prct_t1_min = prctile(dImage, 10);
prct_t1_max = prctile(dImage, 20);
prct_t2_min = prctile(dImage, 10);
prct_t2_max = prctile(dImage, 40);
prct_t3_min = prctile(dImage, 10);
prct_t3_max = prctile(dImage, 60);
prct_t4_min = prctile(dImage, 10);
prct_t4_max = prctile(dImage, 80);
dImg_new_minmax= (dImage - dImg_min) ./ (dImg_max - dImg_min);
dImg_new_t1= (dImage - prct_t1_min) ./ (prct_t1_max - prct_t1_min);
dImg_new_t2= (dImage - prct_t2_min) ./ (prct_t2_max - prct_t2_min);
dImg_new_t3= (dImage - prct_t3_min) ./ (prct_t3_max - prct_t3_min);
dImg_new_t4= (dImage - prct_t4_min) ./ (prct_t3_max - prct_t4_min);
```

```
montage({dImage, dImg_new_minmax, dImg_new_t1, dImg_new_t2, dImg_new_t3, dImg_new_t4})
```



```

```

Activity 1.6 Restoring Faded Colored Photographs

Contrast Stretching

```
I= imread("ken.png");
I = im2double(I);
[R, G, B] = imsplit(I);
montage({R,G,B}, 'size', [1 NaN]);
```

```
minf =@(R) min(R(:));
maxf =@(R)max(R(:));
R_old = R;
R_min = minf(R);
```

```
Prct_minR = prctile(R_min, 50);
R_max = maxf(R);
Prct_maxR = prctile(R_max, 85);
R_contrast = ((R_old - Prct_minR)./(Prct_maxR - Prct_minR));

minf =@(G) min(G(:));
maxf =@(G)max(G(:));
G_old = G;
G_min = minf(G);
Prct_minG = prctile(G_min, 0);
G_max = maxf(G);
Prct_maxG = prctile(G_max, 80);
G_contrast = ((G_old - Prct_minG)./(Prct_maxG - Prct_minG));

minf =@(B) min(B(:));
maxf =@(B)max(B(:));
B_old = B;
B_min = minf(B);
Prct_minB = prctile(B_min, 42);
B_max = maxf(B);
Prct_maxB = prctile(B_max, 90);
B_contrast = ((B_old - Prct_minB)./(Prct_maxB - Prct_minB));

montage({R_contrast, G_contrast, B_contrast}, 'size', [1 NaN])
```

```
I_restored_cs(:,:,1) = R_contrast;
I_restored_cs(:,:,2) = G_contrast;
I_restored_cs(:,:,3) = B_contrast;


montage({I,I_restored_cs}, 'size', [1 NaN])
```

Gray World Algorithm

```
I = imread("ken.png");
I= im2double(I);
[R,G,B] = imsplit(I);

R_ave = mean2(R);
G_ave = mean2(G);
B_ave = mean2(B);


Rwb = R./R_ave;
Gwb = G./G_ave;
Bwb = B./B_ave;

I_restored_gw(:,:,1) = Rwb;
I_restored_gw(:,:,2) = Gwb;
I_restored_gw(:,:,3) = Bwb;

montage({I, I_restored_gw})
```

White Patch Algorithm

```
I = imread("ken.png");
I = im2double(I);
[R,G,B] = imsplit(I);

%white patch
%obtained from using imcrop on I
white = imread('white.tif');
white = im2double(white);
[R_w, G_w, B_w] = imsplit(white);
WR = R_w;
WG = G_w;
WB = B_w;

WRave = mean2(WR);
WGave = mean2(WG);
WBave = mean2(WB);

%original image
Rwb = R./WRave;
Gwb = G./WGave;
Bwb = B./WBave;

I_restored_wp(:,:,1) = Rwb;
I_restored_wp(:,:,2) = Gwb;
I_restored_wp(:,:,3) = Bwb;
montage({I, I_restored_wp})
```

Comparison of the White Balancing Algorithms

Compared and contrast the different white balancing algorithms on the rightmost part showing the image produced when using imlocalbrighten function.

```
I_brighten= imlocalbrighten(I);
montage({I, I_restored_cs, I_restored_gw, I_restored_wp,I_brighten}, 'size', [1 NaN])
```