Lab 3

Introduction

Histograms are one way to display the frequency for which each pixel intensity appears in the image. A histogram cna be produced from an image by simply summing up the number of pixels that appear at each possible pixel intensity. Once a histogram has been taken from an image, there are a number of operations that we can perform on the histograms. The first of these operations is a simple operation called histogram equalization.

Histogram equaliation can be performed on an image to cause the histogram to become flat and uniform. The idea is that this will cause the probability of image intensities to become more uniform across all intensities. especially for images with large ranges of values, and

details clustered about certain intensities. This will greatly improve the contrast and improve the details visible in the image.

Equalizing histograms is a fairly simple operation that requires analyisis of the image and producing a histogram which is then equalized using a normalizing function

However it is also possible to modify the image to be matched up against any kind of histogram distrobution. This is a technique called histogram equalization. This involves producing a normalizing function for either image and producing a nearest-neighbour match from one

image to another. This is then applied to a normalized image and can be used to match the histogram of one image to another.

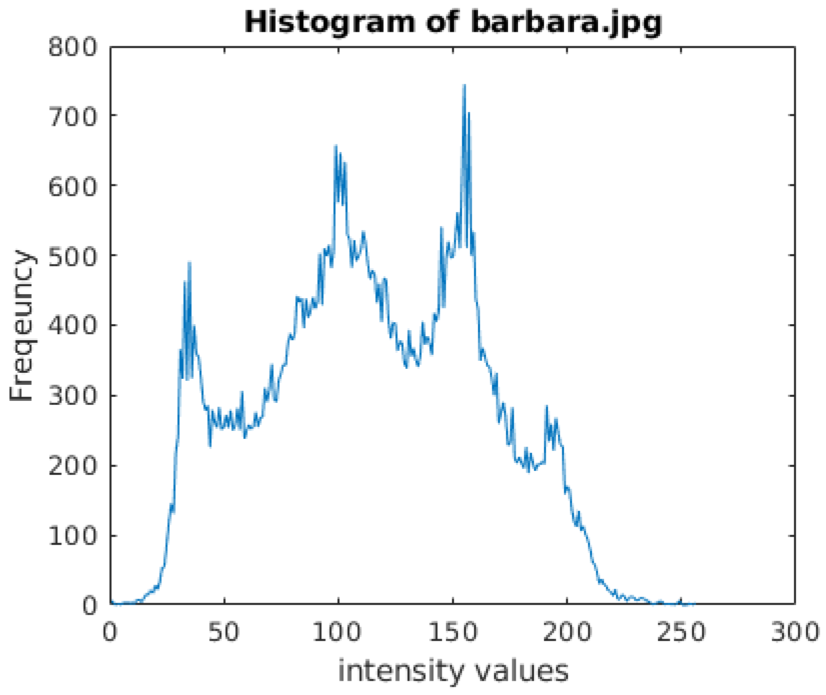
Histograms are just another incredibly useful tool in the field of image processing and over the course of this lab we explore a couple of these histogram equalizing functions as well explore our own.

Histograms

Histograms is a way to show the frequency of each pixel intensity of the image. We can perform many operaitons on an image, given the histogram of the image, For example, we can equalize the histogram tog reatly improve the contrast.

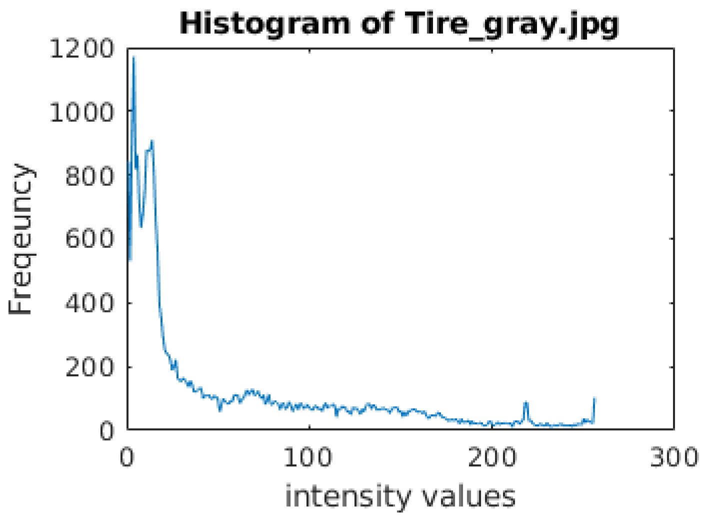
```
I = imread('barbara.jpg');
imshow(I)
```

```
plot(imhist(I));
title('Histogram of barbara.jpg')
ylabel('Freqeuncy')
xlabel('intensity values')
```



```
I = imread('Tire_gray.jpg');
S = imhist(I);
imshow(I)
```

```
plot(imhist(I));
title('Histogram of Tire\_gray.jpg')
ylabel('Freqeuncy')
xlabel('intensity values')
```

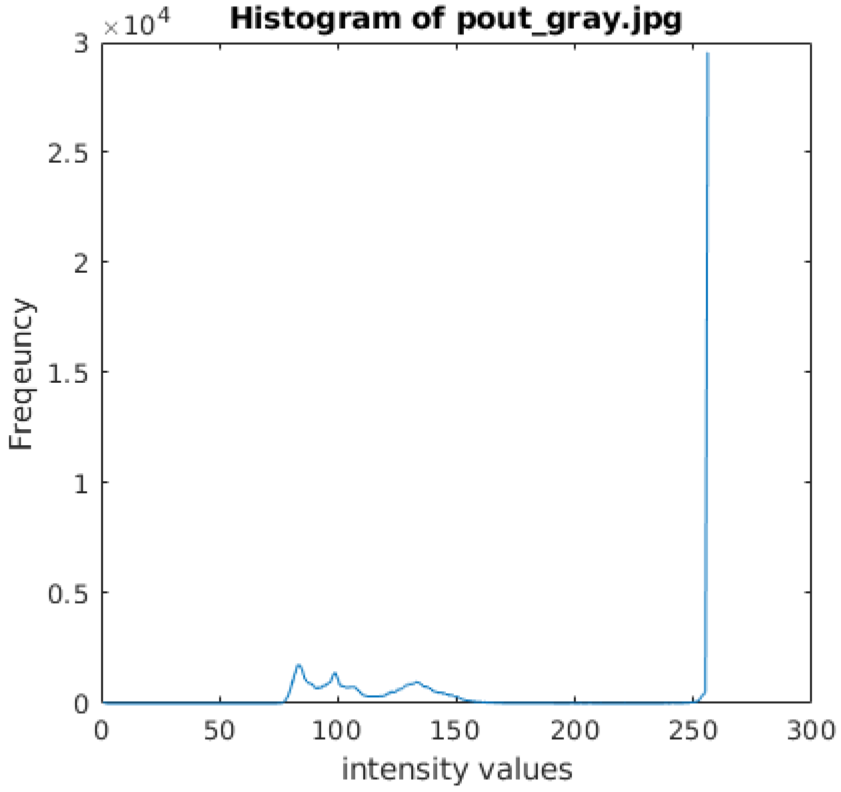


We can see that tire.jpg is a significantly lower on the histogram as this is a significantly darker image

```
I = imread('pout_gray.jpg');
S = imhist(I);
imshow(I)
```

Original

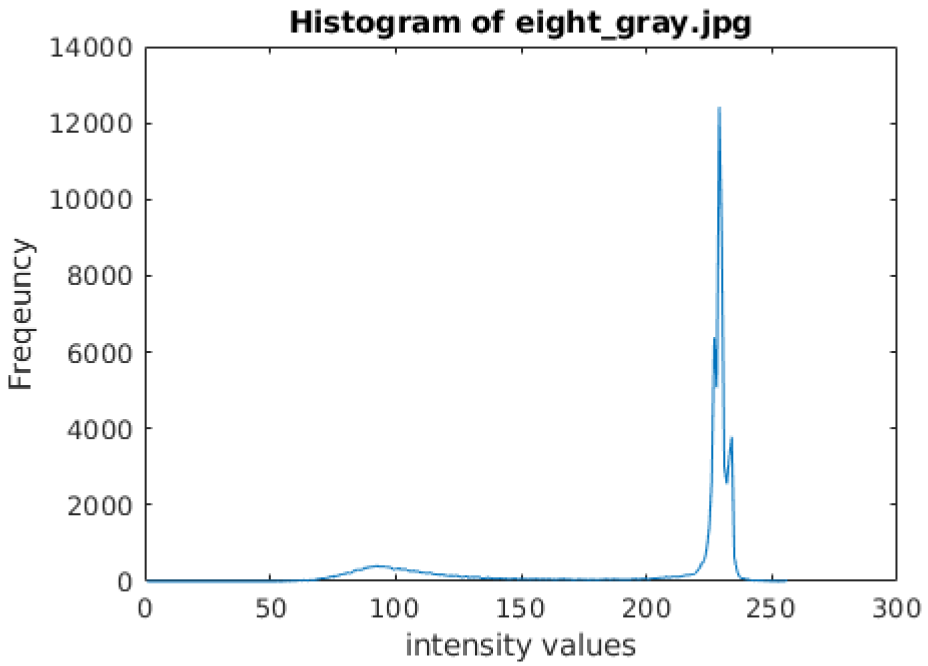
```
plot(imhist(I));
title('Histogram of pout\_gray.jpg')
ylabel('Freqeuncy')
xlabel('intensity values')
```



I expected this image to be fairly gray but the result was quite surprising due to the white border of the image inflating the value at 255

```
I = imread('eight_gray.jpg');
S = imhist(I);
imshow(I)
```

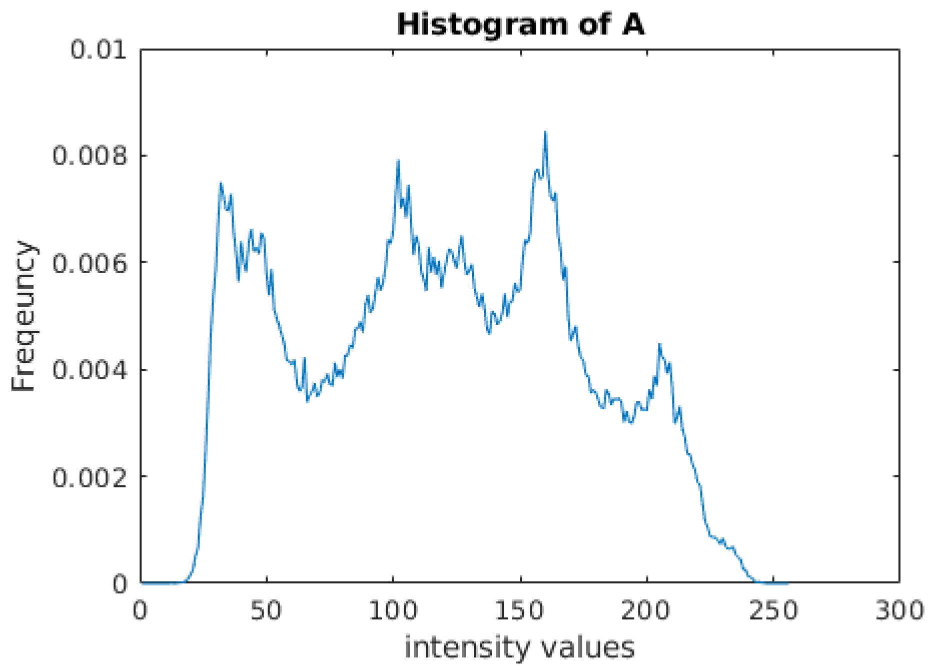
```
plot(imhist(I));
title('Histogram of eight\_gray.jpg')
ylabel('Freqeuncy')
xlabel('intensity values')
```



as can be seen in the image the prevalance of the white background causes the histogram to be high at a certain value.

```
A = imread('barbara.png');
% my_imhist(A)
```

```
hist(256) = 0;
for i = 1:1:numel(A);
    ind = A(i) + 1;
    hist(ind) = hist(ind) + 1;
end
% my_normhist(A)
[x,y] = size(A);
hist = hist/(x*y);
plot(hist)
title('Histogram of A')
ylabel('Freqeuncy')
xlabel('intensity values')
```



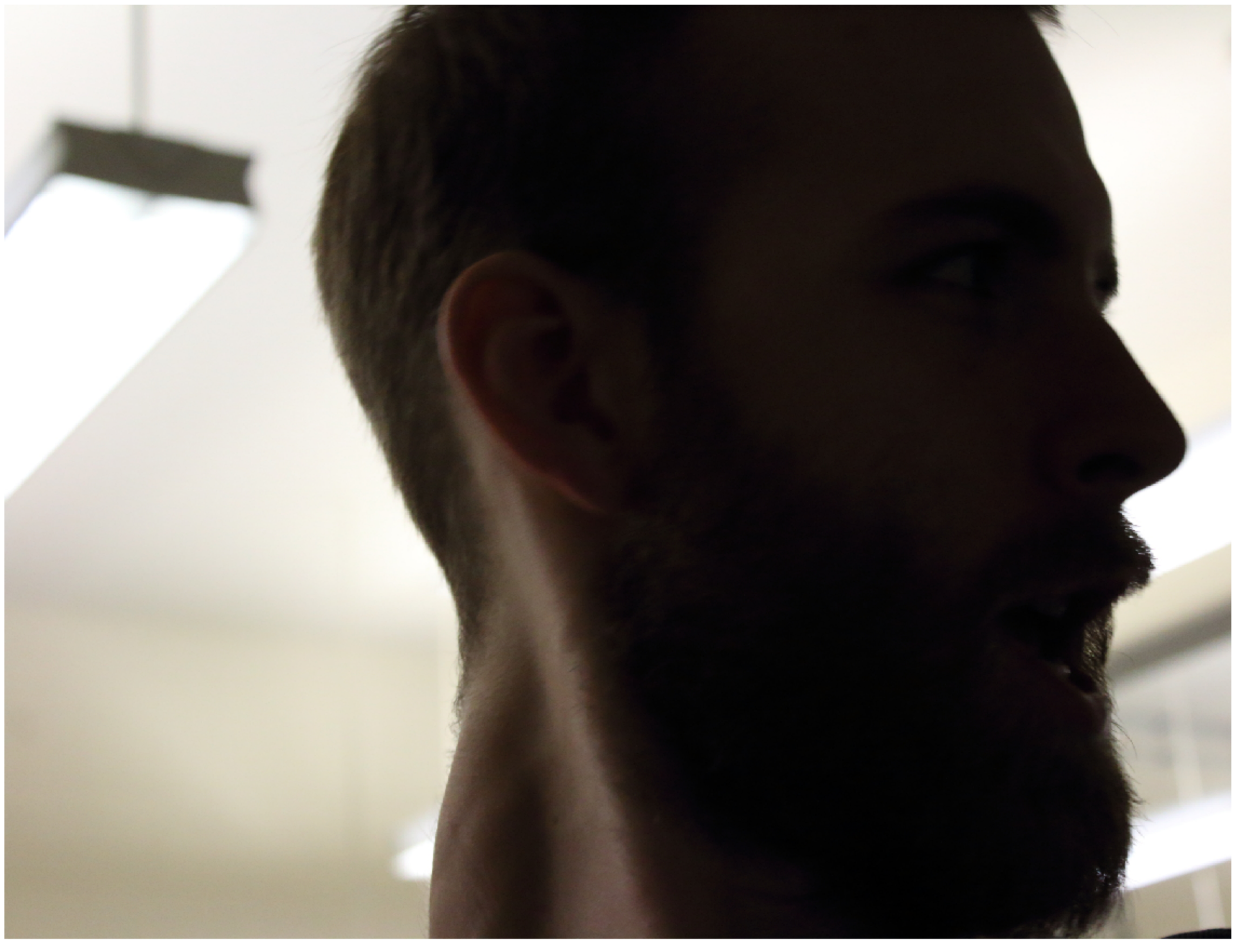
This is my own Histogram function.

```
A = imread('mat.jpg');
% myhisteq(A)
hist = my_normhist(A);
s_k(256) = 0;
const = (255 - 1);
for i = 1:1:256
    s_k(i) = const*sum(hist(1:i));
end
S = zeros(size(A));
for i = 1:1:numel(A)
    S(i) = uint8(s_k(A(i)+1));
end
S = uint8(S);
% End
```

The original image is shown here

```
imshow(A)
```

```
Warning: Image is too big to fit on screen; displaying at 50%
```



With histogram equalization.

```
imshow(S)
```

```
Warning: Image is too big to fit on screen; displaying at 50%
```

Histogram Matching

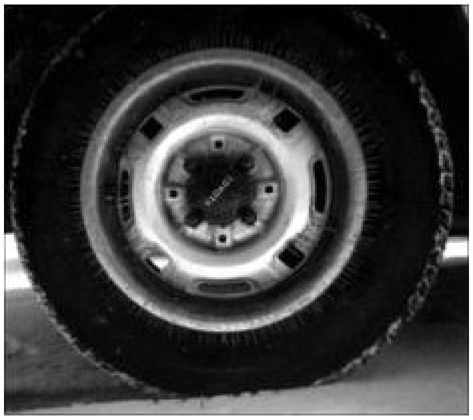
In some Applications, it is needed to change the histogram of an image to a specific histogram in this case we can specify the shape of desired histogram for the processed image. This will generate a second processed image that has a specified histogram called histogram matching.

it happens in 4 steps

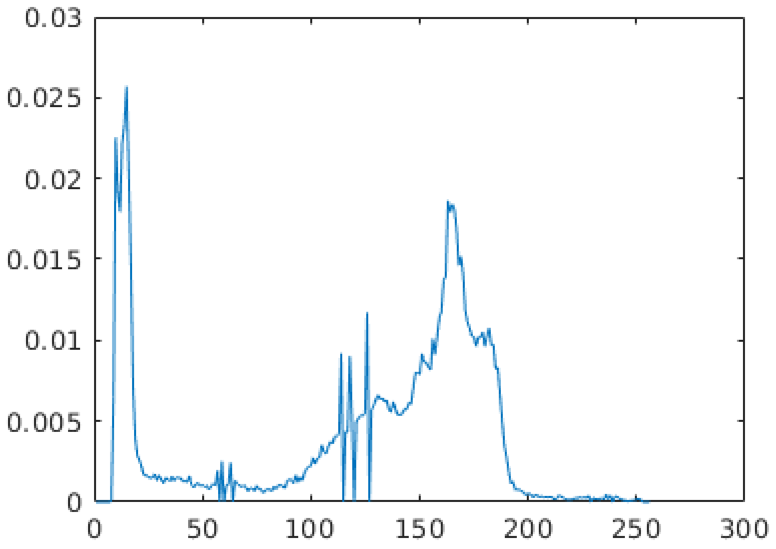
step 1. Find the histogram of P_r of the input image and determine it's equalization image

```
A = imread('Cameraman256.png');
C = imread('tire_orig.jpg');
% equalize A

B = myhisteq(A);
% get the transformation fcns for each of the image equalizations
T = round(get_sk(A));

G = round(get_sk(C));
% inverse transfer function
r = 1:1:256;
```

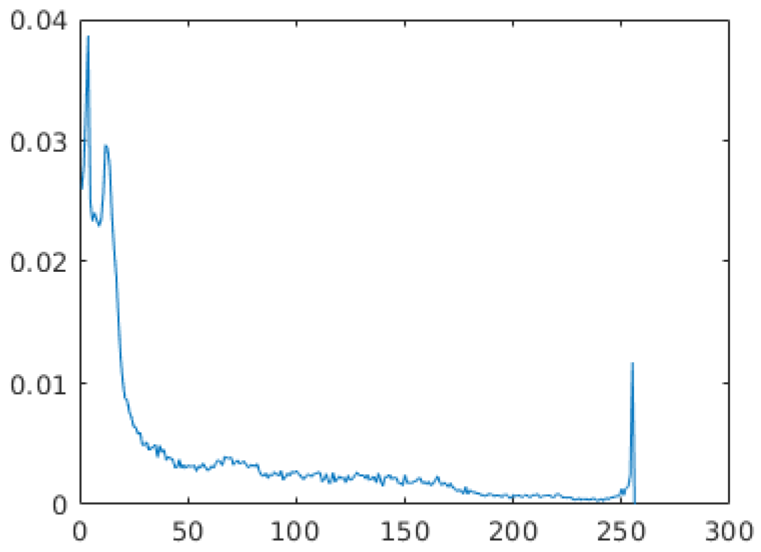
```
G_1(256) =0;
for i = 1:1:256
    [val, ind] = min(abs(i - G));
    G_1(i) = (ind);
end

S= zeros(size(B));

for i = 1:1:numel(B)
    S(i) = G_1(B(i) + 1);
end
S=uint8(S);
imshow(C);
```



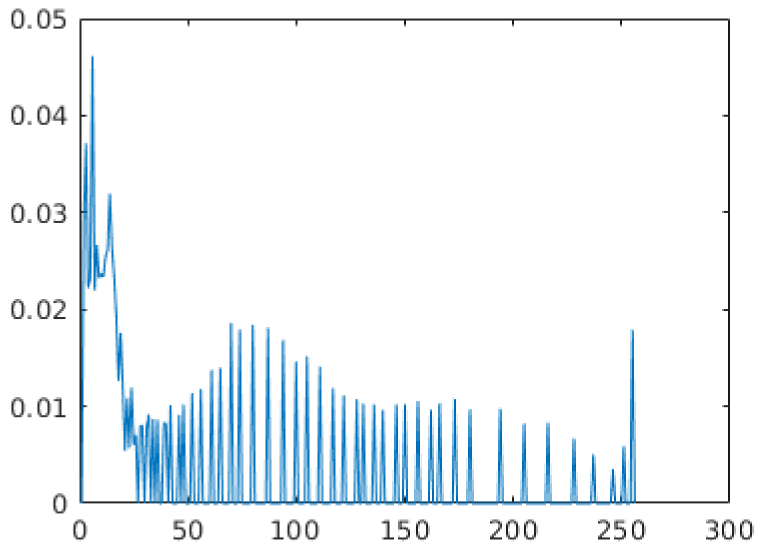
```
plot(my_normhist(A));
```



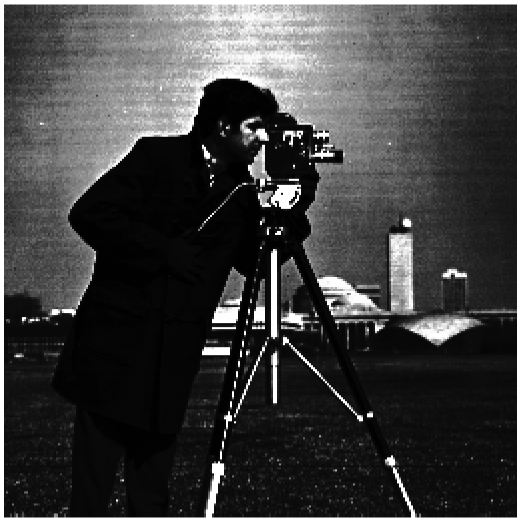
```
plot(my_normhist(C));
```



```
plot(my_normhist(S));
```



```
imshow(S);
```



So what happened here, given the two images, Cameraman256 (read in as A) we produced a histogram equalized version of A called (B) and then read in the image tire as C. The goal here is to produce match A's histogram to C. This takes 4 steps to do

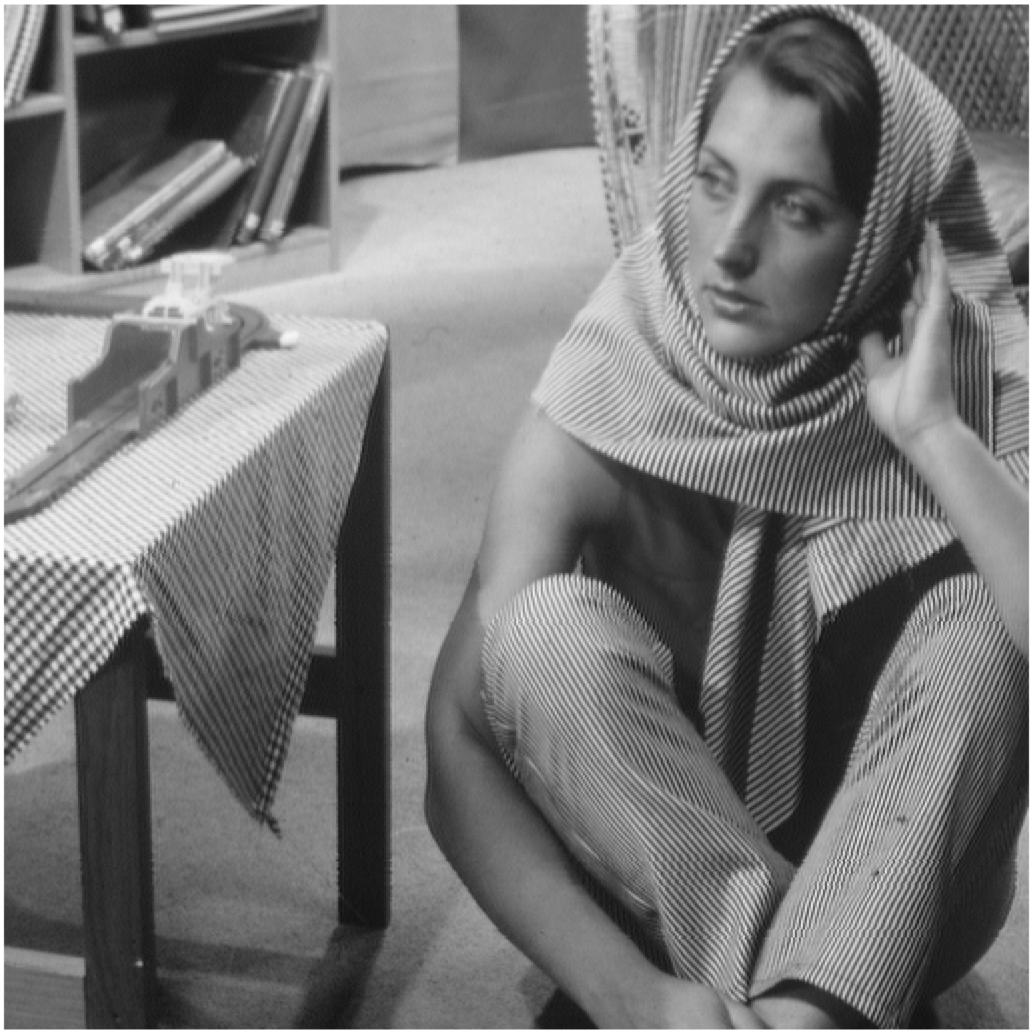
1. get T for A
2. Normalize the histogram for A with T and produce B
3. Generate function G for normalizing B, then invert G and get G_1
4. Apply transformation G_1 to B

The result is the normalized image S.

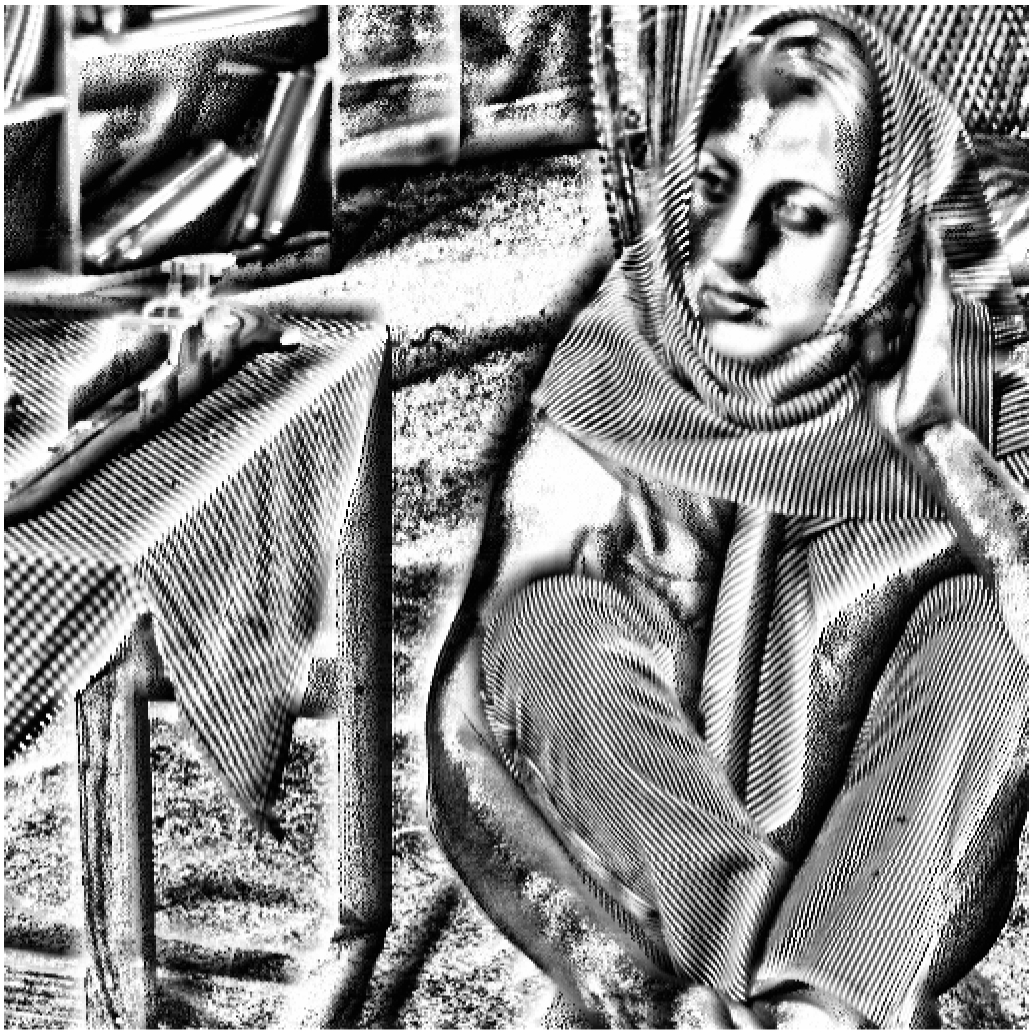
Histogram Windowing Function.

We can perform histogram equalization on a picture by appling a window size of MxN In this example you can see how this is done with a window size of MxN for a 12x12 image.

```
% read A
A = imread('barbara.png');
imshow(A)
```



```
M = 12;
N = 12;
x_off = 1;
y_off = 1;
[Y, X] = size(A);
for j = 1:1:(Y)
    for i = 1:1:(X)
        histI = A(y_off:min(M+y_off,Y),x_off:min(x_off+N, X));
        histI = myhisteq(histI);
        S(y_off:min(M+y_off,Y),x_off:min(x_off+N, X)) = histI;
        x_off = x_off + 1;
    end
    x_off = 1;
    y_off = y_off + 1;
end

imshow(S);
```



Here we can see that applying the histogram of a window of a small size causes tons of details to be revealed on the image. This doesnt result in a very good looking image, but it reveals all the individual details that are present in the image.

Conclusions

Over the course of this lab we showed how histograms can be used as an approximation of an image's probability density function, and how we are able to utilize transformations on these histograms to perform functions such as histogram equalization which are used to reveal details of an image by equalizing the probability of each pixel intensity. We also showed how we can perform other tramsformations such as matching the histogram profile of one image to another one. Additionally we are able to reveal localized details of an image by using histogram equalization in a window. revealing lots of details of an image locally for each area of the image.