1. **Edge Detection**

a) Sobel, Prewitt, canny and LoG filters on the Y channel only

```
I = imread("~/Documents/inputImages/med.jpg");
Iy = rgb2ycbcr(I);
y = Iy(:,:,1);
s = edge(y,'Sobel');
p = edge(y,'Prewitt');
c = edge(y,'Canny');
l = edge(y,'log');
```

Sobel

Prewitt

Canny

Log

Clearly from the above thresholded images we can observe that Canny Edge detection gives the best result.

Explanation: Methods Sobel and Prewitt are based on the global gradient of the image Therefore many weak edges are missed.

Canny Edge detection uses the local gradient of the image. It first uses a derivative of Gaussian to remove noise. It uses two threshold values, one for detecting stronger edges and one for the weak edges.

Log Edge detection extracts the laplacian of the gaussian filtered image. It then checks for the Zero-crossings i.e. whenever the sign value changes.

b) Add Gaussian noise to the luminance image and find edges

```
L = rgb2gray(I);
J = imnoise(L,'gaussian');
s1 = edge(J,'Sobel');
p1 = edge(J,'Prewitt');
c1 = edge(J,'Canny');
l1 = edge(J,'log');
```

Sobel and Prewitt are slightly affected but Canny and Log are completely messed up because both methods are sensitive to noise. That is why the Gaussian filtering is done at the beginning of both the methods.

To overcome this we can increase the standard deviation (**sigma**) of the Gaissian filter for both the methods so the noise is reduced and edges are better detected. Updated Code:

```
c2 = edge(J,'Canny', [], 2.1);
l2 = edge(J,'log', [], 2.5);
```

Canny sigma = 2.1

Log sigma = 2.5

Canny gives better result at correct sigma value.

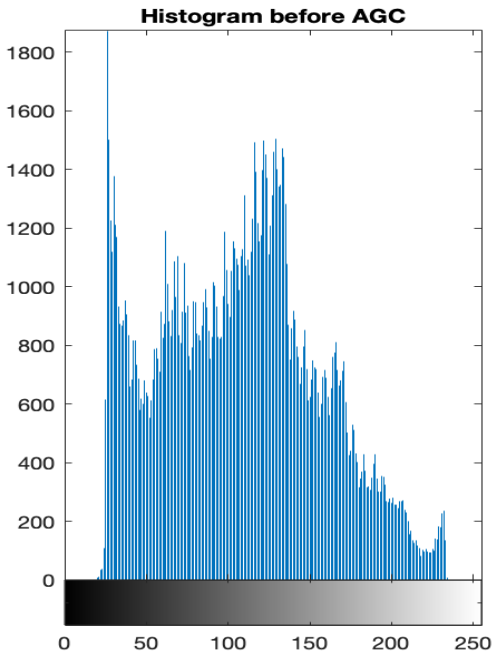
2. **Contrast enhancement of Color Images**

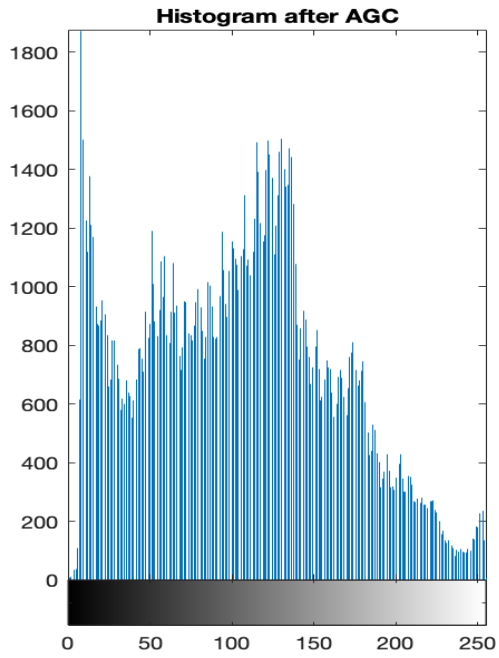
```
I = imread("~/Documents/inputImages/geisha.jpg");
Iy = rgb2ycbcr(I);
y1 = Iy(:,:,1);
y = Iy(:,:,1);
```

a) Adaptive Gain Control.

```
A = double(min(y(:)));
B = double(max(y(:)));
K = double(256);
alpha = double(double(K-1)/(B-A));
beta = double((-1)*A*(double(K-1)/(B-A)));
y1 = uint8(alpha * double(y) + beta);
Iy1 = Iy;
Iy1(:,:,1) = y1;
rgb = ycbcr2rgb(Iy1);
```

The lowest level of the image is 19 and the highest level is 234 so no much enhancement can be done in the case of this image where 19 is mapped to 0 and 234 is mapped to 255.





The histogram is stretched from [19, 234] to [0, 255].

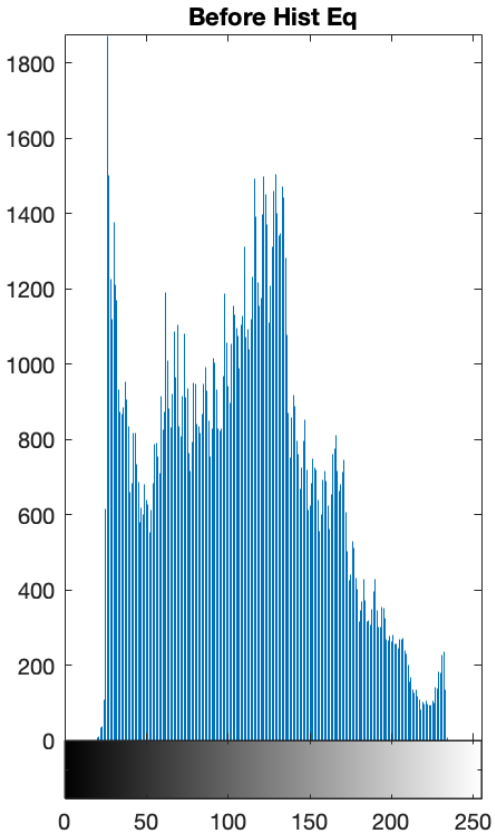
Result:

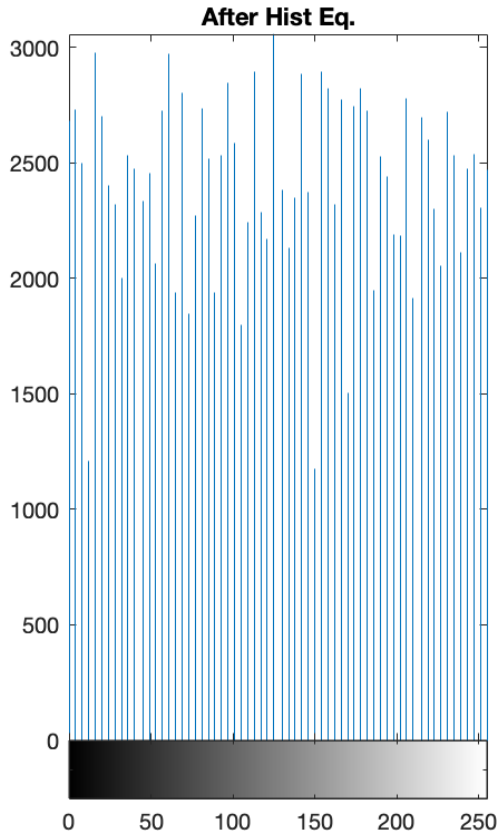
The major difference observed is that the hair became slightly darker and the dress became slightly brighter.

b) Histogram Equalization

```
yh = Iy(:,:,1);
yh1 = histeq(yh);
Iyh = Iy;
Iyh(:,:,1) = yh1;
rgb1 = ycbcr2rgb(Iyh);
```

Histogram Equalization remaps histogram to have a uniform probability density function. The transformed image has an almost flat histogram.





Result:

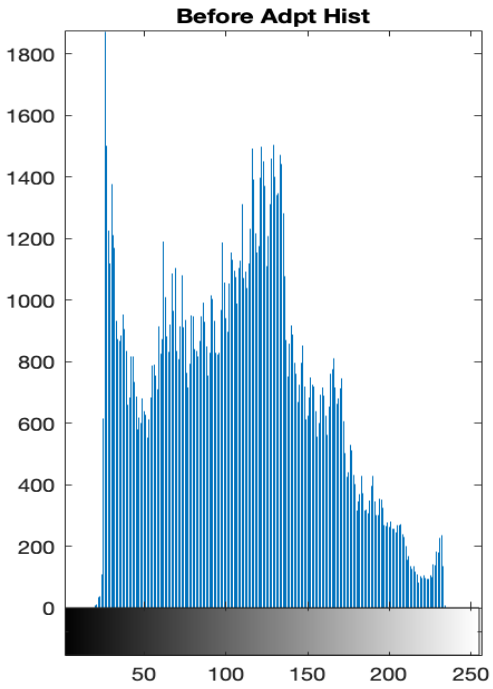
Before HistEq

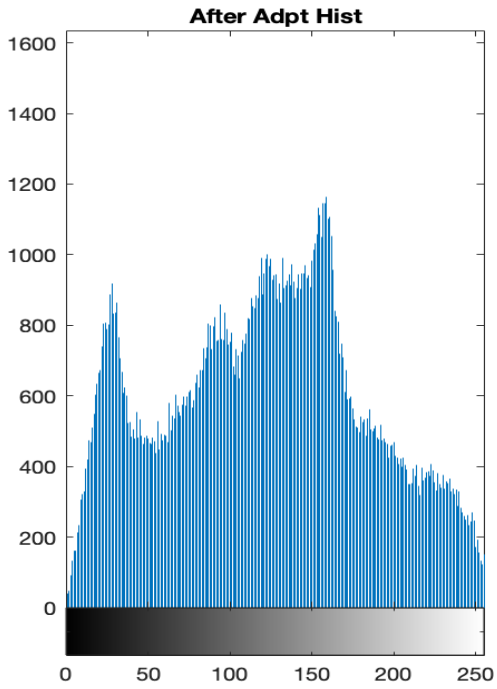
After HistEq

This image has a bimodal histogram. By the process the modes will be flattened and spread. The intensities of both the peaks are equalized.

c) Adaptive Histogram Equalization

```
ya = Iy(:,:,1);
ya1 = adapthisteq(ya);
Iya = Iy;
Iya(:,:,1) = ya1;
rgb2 = ycbcr2rgb(Iya);
```


Before Adpt Hist
1800
1600
1400
1200
1000
800
600
400
200
0
50
100
150
200
250


After Adpt Hist
1600
1400
1200
1000
800
600
400
200
0
0
50
100
150
200
250


Before Adpt Hist


After Adpt hist

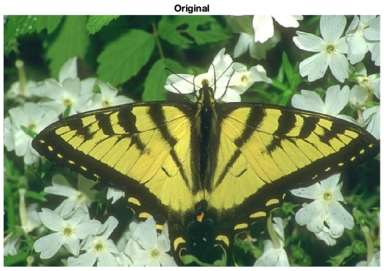
In adaptive histogram equalization each mode is evaluated in a sliding window and independent histogram equalization is executed.

As a result the contract is enhanced and details are also preserved.

3. Sharpness Enhancement of Color Images

a) Apply unsharp masking to the Y component only

```
I = imread("~/Documents/inputImages/butterfly.jpg");
Iy = rgb2ycbcr(I);
y = Iy(:,:,1);
Iy1 = Iy;
Iy2 = Iy;
Iy3 = Iy;
yd = double(y);
h = fspecial('gaussian',5);
yl = imfilter(yd,h);
B1 = double(2);
B2 = double(4);
B3 = double(8);
ys1 = uint8(yl + B1*(yd - yl));
ys2 = uint8(yl + B2*(yd - yl));
ys3 = uint8(yl + B3*(yd - yl));
Iy1(:,:,1) = ys1;
Iy2(:,:,1) = ys2;
Iy3(:,:,1) = ys3;
rgb1 = ycbcr2rgb(Iy1);
rgb2 = ycbcr2rgb(Iy2);
rgb3 = ycbcr2rgb(Iy3);
```



b) Apply unsharp masking to the Y, Cr and Cb components

```
I = imread("~/Documents/inputImages/butterfly.jpg");
Iy = rgb2ycbcr(I);
B = double(8);
y = Iy(:,:,1);
cr = Iy(:,:,2);
cb = Iy(:,:,3);
yd = double(y);
crd = double(cr);
cbd = double(cb);
h = fspecial('gaussian',5);
yl = imfilter(yd,h);
crl = imfilter(crd,h);
cbl = imfilter(cbd,h);
ys = uint8(yl + B1*(yd - yl));
crs = uint8(crl + B1*(crd - crl));
cbs = uint8(cbl + B1*(cbd - cbl));

Iy1 = Iy;
Iy1(:,:,1) = ys;
Iy1(:,:,2) = crs;
Iy1(:,:,3) = cbs;
rgb1 = ycbcr2rgb(Iy1);
figure;imshow(rgb1);
```

USM on Y component

USM on all YCbCr

For ycrcb image, Y part contains the Luminance information and Cr and Cb contains Chrominance information. Most of the textural information is present in the Y component. Therefore Unsharp masking on only Y components or on Y Cr Cb doesn’t make much difference.