IMAGE PROCESSING LAB 4

AIM: Understanding Image Histograms and implementing histogram equalization and matching.

IMAGE HISTOGRAM:

- An image histogram is a **plot of the gray-level frequencies** (i.e., the number of pixels in the image that have that gray level).
- Divide frequencies by total number of pixels (m x n image size) to represent as probabilities.

HISTOGRAM EQUALIZATION:

- To **improve the contrast** of an image.
- To transform an image in such a way that the transformed image has a nearly uniform distribution of pixel values.

HISTOGRAM MATCHING:

- Histogram equalization yields an image whose pixels are uniformly distributed among all gray levels.
- Sometimes, this may not be desirable. Instead, we may want a transformation that **yields an output image** with a pre-specified histogram. This technique is called histogram matching.

IMPORTANT FUNCTIONS:

- Plot histogram for RGB image.

```
function void = plot_hist(r)
  red_channel = r(:,:,1);
  green_channel = r(:,:,2);
  blue_channel = r(:,:,3);

  [yRed, x] = imhist(red_channel);
  [yGreen, x] = imhist(green_channel);
  [yBlue, x] = imhist(blue_channel);

  plot(x, yRed, x, yGreen, x, yBlue);
  legend("Red","Green","Blue");
endfunction
```

- Histogram equalization of grayscale image.

```
function [s,final] = imequalizehist(r)
  L=256;
  [m,n] = size(r);
  hist = zeros(size(L-1));
  for i=0:(L-1),
    hist(i+1) = sum(sum(r==i));
  endfor
  pdf = hist/(m*n);
  total(1) = pdf(1);
  for i=1:(L-1),
    total(i+1) = total(i)+pdf(i+1);
  endfor
  s = (L-1)*total;
  s = round(s);
  final = zeros(m,n);
  for i=0:(L-1),
    final = final + (r==i)*s(i+1);
  endfor
  final = uint8(final);
endfunction
```

- Histogram equalization of RGB image.

```
function [s, final] = imequalizecolorhist(r)
  [m,n,d] = size(r);
  L=256;

  red_channel = r(:,:,1);
  green_channel = r(:,:,2);
  blue_channel = r(:,:,3);

  [sr, finalr] = imequalizehist(red_channel);
  [sg, finalg] = imequalizehist(green_channel);
  [sb, finalb] = imequalizehist(blue_channel);

  s=zeros(d,L);
  s(1,:)=sr;
  s(2,:)=sg;
  s(3,:)=sb;
  s = uint8(s);

  final = zeros(size(r));
  final(:,:,1) = finalr;
  final(:,:,2) = finalg;
  final(:,:,3) = finalb;
  final = uint8(final);
endfunction
```

- Histogram matching of grayscale image.

```
function [s,final] = imhistmatch(r,ref)
  L=256;
  [m,n] = size(r);
  [s ,final1] = imequalizehist(r);
  [G, final2] = imequalizehist(ref);
  for i=0:(L-1),
    [val ind(i+1)] = min(abs(G-s(i+1)));
  endfor
  ind = ind-1;
  s=ind;
  final  = zeros(m,n);
  for i=0:(L-1),
    final = final + (r==i)*ind(i+1);
  endfor
  final = uint8(final);
endfunction
```

EXERCISE:

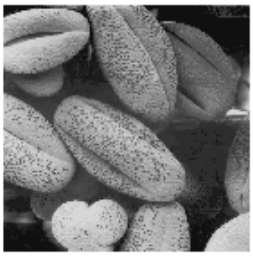
1. Can two visually different images have the same histogram? If yes, synthesize two grayscale images which are visually different but have the same histogram and also show the histogram. If no, justify your answer.

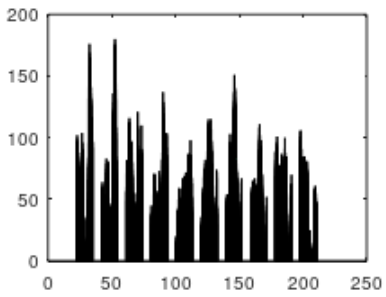
- Yes, two visually different images can have the same histogram. Each column in the histogram represents how many pixels in the photograph have the pixel value represented by the column. Histogram does not tell you where those pixels are located within the image. As a result, two different images can result in the same histogram.
- Example for this is as below.

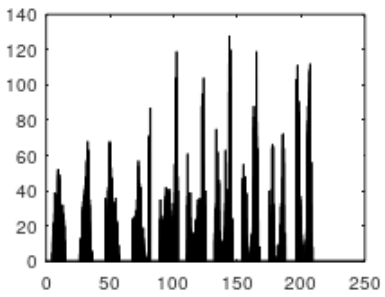
Code: -

```
img1 = imread('images/test3.tif');
img2 = imread('images/1.jpg');
ref = imread('images/2.jpg');
[s1,r1] = imhistmatch(img1,ref);
[s2,r2] = imhistmatch(img2,ref);
subplot(2,2,1);
imshow(r1);
subplot(2,2,2);
imshow(r2);
subplot(2,2,3);
hist(r1);
subplot(2,2,4);
hist(r2);
```

Output: -





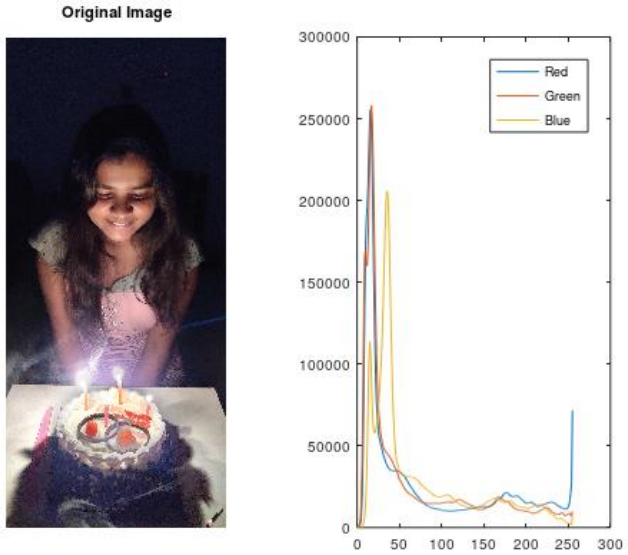


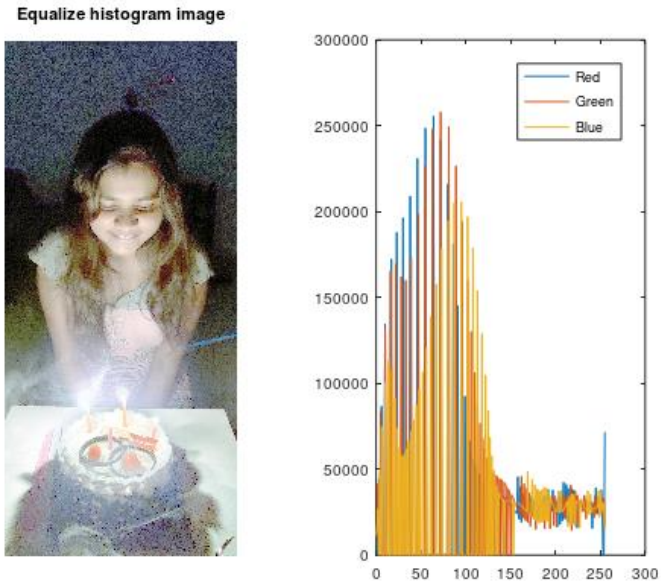
2. Take your color photograph taken in the dark. Equalize its histogram.

- Code: -

```
img = imread('images/myimg_dark.jpg');
subplot(1,2,1);
imshow(img);
title('Original Image');
red_channel = img(:,:,1);
green_channel = img(:,:,2);
blue_channel = img(:,:,3);
[yRed, x] = imhist(red_channel);
[yGreen, x] = imhist(green_channel);
[yBlue, x] = imhist(blue_channel);
subplot(1,2,2);
plot(x, yRed, x, yGreen, x, yBlue);
legend("Red","Green","Blue");
[sr, finalr] = imequalizehist(red_channel);
[sg, finalg] = imequalizehist(green_channel);
[sb, finalb] = imequalizehist(blue_channel);
[yRedf, xf] = imhist(finalr);
[yGreenf, xf] = imhist(finalg);
[yBluef, xf] = imhist(finalb);
figure;
s = zeros(size(img));
s(:,:,1) = finalr;
s(:,:,2) = finalg;
s(:,:,3) = finalb;
subplot(1,2,1);
imshow(uint8(s));
title('Equalize histogram image');
subplot(1,2,2);
plot(xf, yRedf, xf, yGreenf, x, yBluef);
legend("Red","Green","Blue");
```

- Output: -



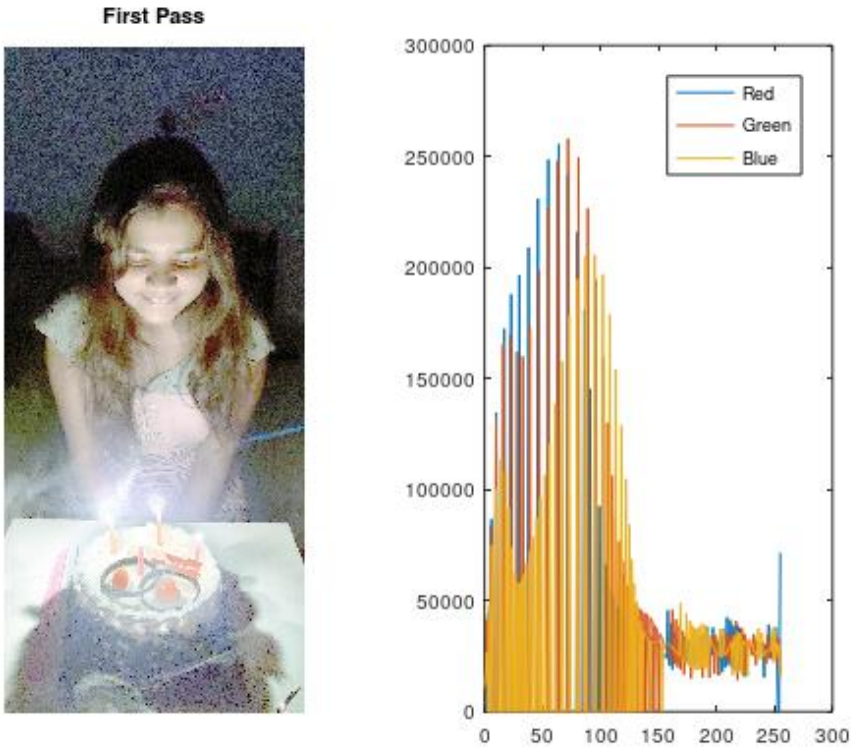


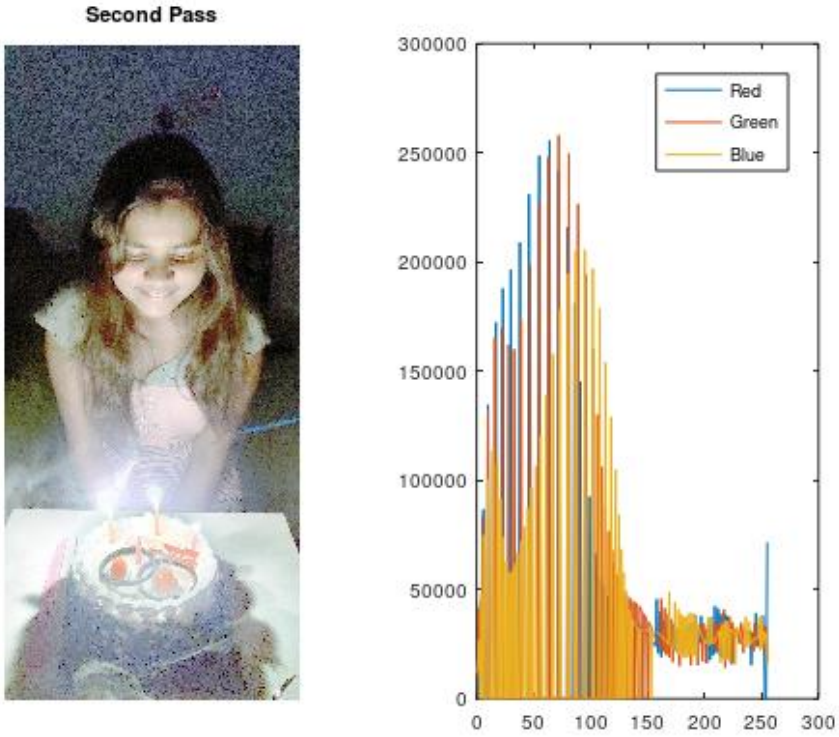
3. Perform histogram equalization of equalized image obtained. Is the second pass of the histogram equalization process useful? Justify your answer.

Code: -

```
1  img = imread('images/myimg_dark.jpg');
2  [s1, final1] = imequalizecolorhist(img);
3  [s2, final2] = imequalizecolorhist(final1);
4  subplot(1,2,1);
5  imshow(final1);
6  title("First Pass");
7  subplot(1,2,2);
8  plot_hist(final1);
9  figure;
10 subplot(1,2,1);
11 imshow(final2);
12 title("Second Pass");
13 subplot(1,2,2);
14 plot_hist(final2);
```

Output: -





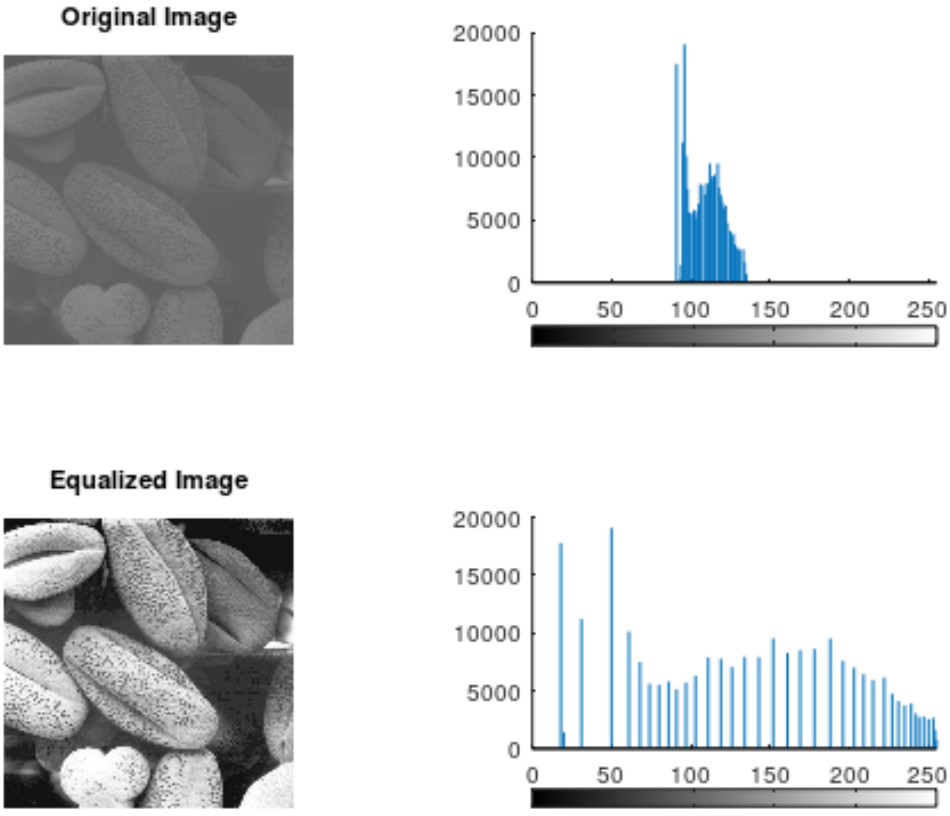
- As we see, in second pass of equalized histogram image has no importance and both image and histogram does not change.

4. Perform histogram equalization for image ‘test3.tif’.

Code: -

```
img = imread('images/test3.tif');
subplot(2,2,1);
imshow(img);
title("Original Image");
subplot(2,2,2);
imhist(img);
[s, final] = imequalizehist(img);
subplot(2,2,3);
imshow(final);
title("Equalized Image");
subplot(2,2,4);
imhist(final);
```

Output: -



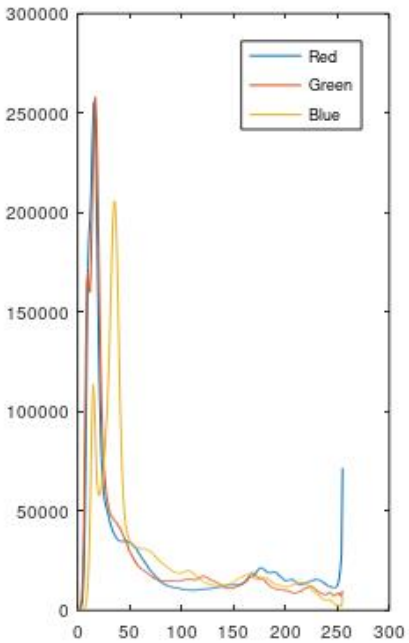
5. Take any of your photographs, match its histogram with the histogram of image ‘test4.jpg’. plot histogram of original image, template and matched image for all three channels.

Code: -

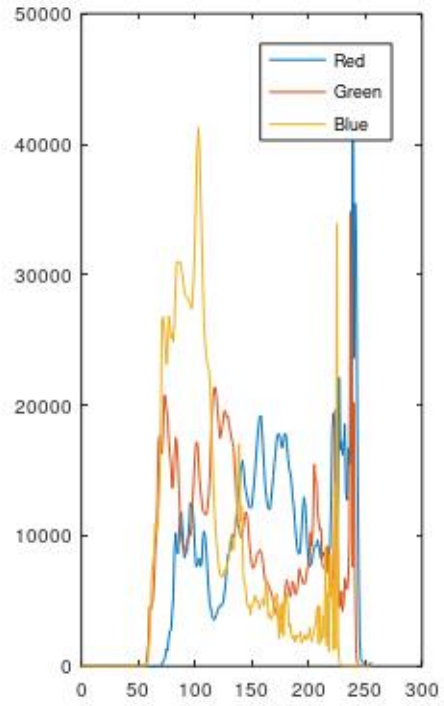
```
img = imread('images/myimg_dark.jpg');
ref = imread('images/test4.jpg');
subplot(1,2,1);
imshow(img);
subplot(1,2,2);
plot_hist(img);
figure;
subplot(1,2,1);
imshow(ref);
subplot(1,2,2);
plot_hist(ref);
figure;
L=256;
[m,n,d] = size(img);
[sf1, final(:,:,1)] = imhistmatch(img(:,:,1),ref(:,:,1));
[sf2, final(:,:,2)] = imhistmatch(img(:,:,2),ref(:,:,2));
[sf3, final(:,:,3)] = imhistmatch(img(:,:,3),ref(:,:,3));
final = uint8(final);
subplot(1,2,1);
imshow(uint8(final));
subplot(1,2,2);
plot_hist(final);
```

Output: -

- Original image



- Test4 image



- Matched image