"Heaven's Light is Our Guide "

Rajshahi University of Engineering & Technology

Submitted To:

Rizoan Toufiq

Assistant professor of

Department of Computer Science & Engineering

Course Title : Digital Image Processing

Course No : CSE 4106

Submitted By:

Sabit Ahmed

Roll:1503056

Sec: A

Department of Computer Science & Engineering

Rajshahi University of Engineering & Technology

TABLE OF CONTENTS

2. Image Interpolation

2.1 Bilinear Interpolation

Theory

In mathematics, **bilinear interpolation** is an extension of linear interpolation for interpolating functions of two variables (e.g., *x* and *y*) on a rectilinear 2D grid.

An interpolation technique that reduces the visual distortion caused by the fractional zoom calculation is the bilinear interpolation algorithm, where the fractional part of the pixel address is used to compute a weighted average of pixel brightness values over a small neighborhood of pixels in the source image. Bilinear interpolation produces pseudoresolution that gives a more aesthetically pleasing result, although this result is again not appropriate for measurement purposes.

Resampling Through Bilinear Interpolation

Let **I** be an $R \times C$ image.
We want to resize **I** to $R' \times C'$.
Call the new image **J**.

Let $s_R = R / R'$ and $s_C = C / C'$.

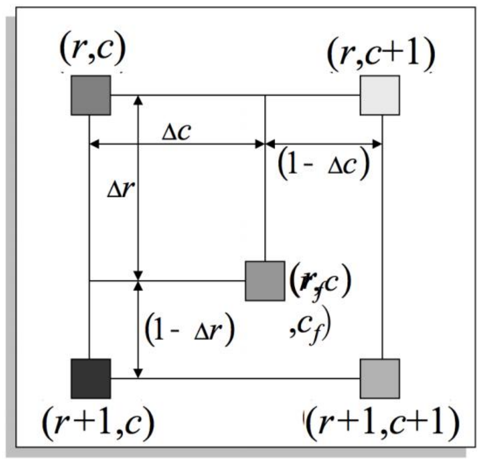
Let $r_f = r' \cdot s_R$ for $r' = 1, \ldots, R'$

and $c_f = c' \cdot s_C$ for $c' = 1, \ldots, C'$.

Let $r = \lfloor r_f \rfloor$ and $c = \lfloor c_f \rfloor$.

Let $\Delta r = r_f - r$ and $\Delta c = c_f - c$.

$$\begin{aligned}\text{Then } \mathbf{J}(r',c') = {} & \mathbf{I}(r,c)\cdot(1-\Delta r)\cdot(1-\Delta c) \\ & +\mathbf{I}(r+1,c)\cdot\Delta r\cdot(1-\Delta c) \\ & +\mathbf{I}(r,c+1)\cdot(1-\Delta r)\cdot\Delta c \\ & +\mathbf{I}(r+1,c+1)\cdot\Delta r\cdot\Delta c.\end{aligned}$$



Code

Raw code

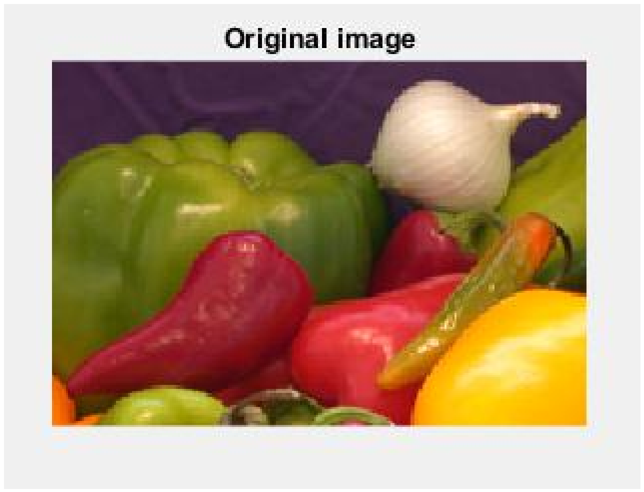
```
function [out] = bilinearInterpolation(x, out_dims)

    im = imread(x);

    in_rows = size(im,1);
```

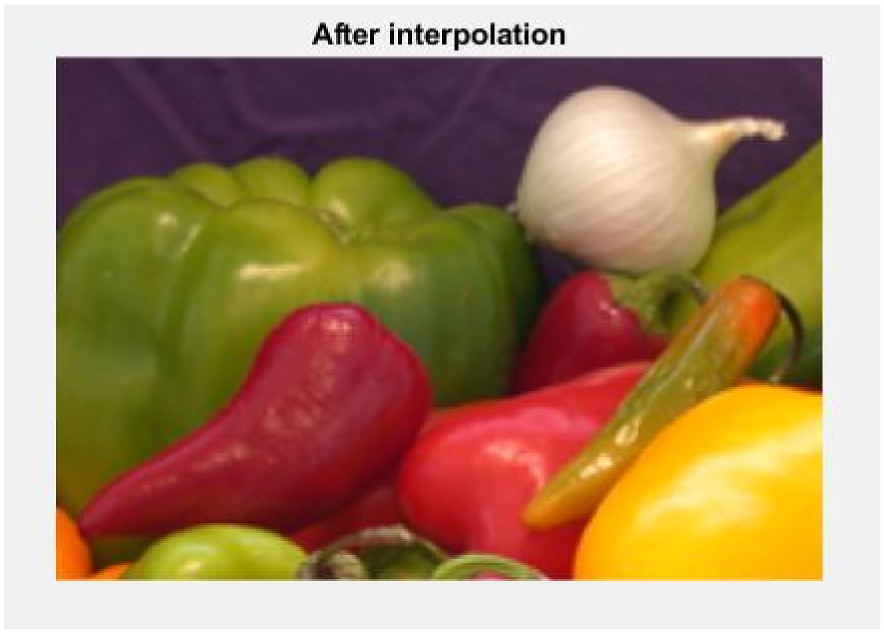
```
in_cols = size(im,2);

out_rows = out_dims(1);

out_cols = out_dims(2);


%// Let S_R = R / R'

S_R = in_rows / out_rows;

%// Let S_C = C / C'

S_C = in_cols / out_cols;


%// Generate (x,y) pairs for each point in our image

[cf, rf] = meshgrid(1 : out_cols, 1 : out_rows);


%// Let r_f = r'*S_R for r = 1,...,R'

%// Let c_f = c'*S_C for c = 1,...,C'

rf = rf * S_R;

cf = cf * S_C;


r = floor(rf);

c = floor(cf);


r(r < 1) = 1;

c(c < 1) = 1;

r(r > in_rows - 1) = in_rows - 1;

c(c > in_cols - 1) = in_cols - 1;
```

```
    delta_R = rf - r;

    delta_C = cf - c;


    in1_ind = sub2ind([in_rows, in_cols], r, c);

    in2_ind = sub2ind([in_rows, in_cols], r+1,c);

    in3_ind = sub2ind([in_rows, in_cols], r, c+1);

    in4_ind = sub2ind([in_rows, in_cols], r+1, c+1);


    out = zeros(out_rows, out_cols, size(im, 3));

    out = cast(out, class(im));


    for idx = 1 : size(im, 3)

        chan = double(im(:,:,idx));

        tmp = chan(in1_ind).*(1 - delta_R).*(1 - delta_C) + ...

            chan(in2_ind).*(delta_R).*(1 - delta_C) + ...

            chan(in3_ind).*(1 - delta_R).*(delta_C) + ...

            chan(in4_ind).*(delta_R).*(delta_C);

        out(:,:,idx) = cast(tmp, class(im));

    end

figure;

imshow(im);

title('Original image');

figure;

imshow(out);

title('After interpolation');
```

```
end
```

Output





Discussion

The above code will interpolate the image by increasing each dimension by twice as much, then show a figure with the original image and another figure with the scaled up image.

Similarly, the image can be shrunk down by half as much.

3. Some Basic Intensity Transformation Functions

3.1 Contrast Stretching

Theory

Contrast stretching (often called normalization) is a simple image enhancement technique that attempts to improve the contrast in an image by ‘stretching’ the range of intensity values it contains to span a desired range of values, e.g. the the full range of pixel values that the image type concerned allows.

The equation for contrast stretching is as follows:

$$P_{out} = (P_{in} - c)\left(\frac{b-a}{d-c}\right) + a$$

For more simplification:

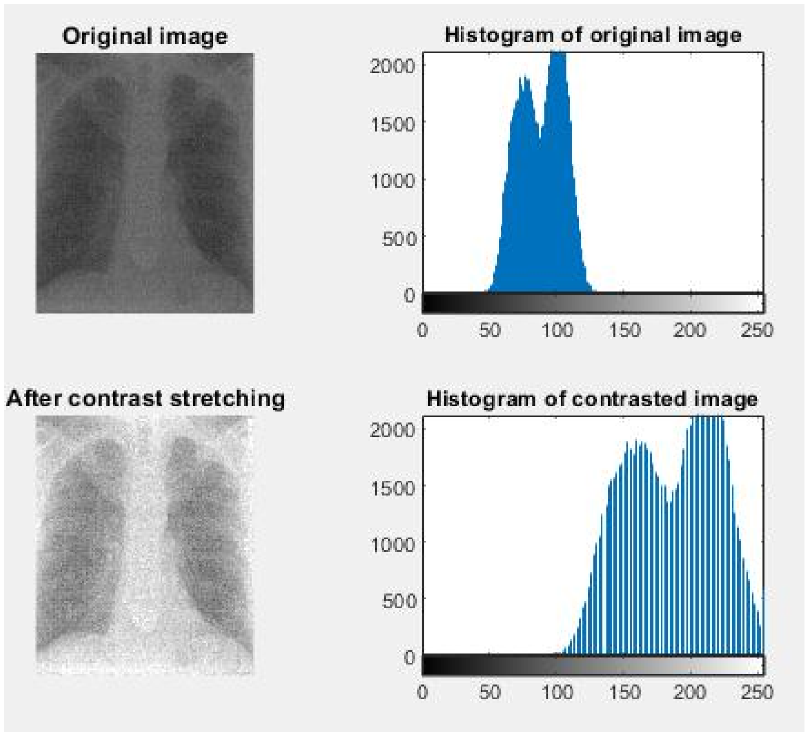
$$I_N = (I - \mathrm{Min})\frac{\mathrm{newMax} - \mathrm{newMin}}{\mathrm{Max} - \mathrm{Min}} + \mathrm{newMin}$$

Code

```
function contrast_stretching()

x=imread('peppers.png');

f=rgb2gray(x);

a=input('Enter the lower range, a = ');

b=input('Enter the higher range, b = ');

al=input('Enter alpha value = ');

be=input('Enter beta value = ');

ga=input('Enter gamma value = ');

va=al*a;

vb=be*(b-a)+va;

[M,N]=size(f);
```

```
          for x = 1:M
               for y = 1:N
                    if(f(x,y)<a)
                         g(x,y)=al*f(x,y);
                    elseif(f(x,y)>=a && f(x,y)<b)
                         g(x,y)=be*(f(x,y)-a)+va;
                    else
                         g(x,y)=ga*(f(x,y)-b)+vb;
                    end
               end
          end
subplot(1,2,1),imshow(f);title('original image');
subplot(1,2,2),imshow(g);title('After contrast stretching');
end
```

Output


Original image
Histogram of original image
2000
1500
1000
500
0
0
50
100
150
200
250
After contrast stretching
Histogram of contrasted image
2000
1500
1000
500
0
0
50
100
150
200
250

Discussion

From the above figure, output of basic contrast stretching on the X-Ray image shows the stretching on some portions of the images.

3.2 Log Transformation

Theory

Log transformation maps a narrow range of low intensity values in the input into a wider range of output levels.We use a transformation of this type to expand the values of dark pixels in an image while compressing the higher-level values.

The log transformations can be defined by this formula

$s = c \log(r + 1)$.

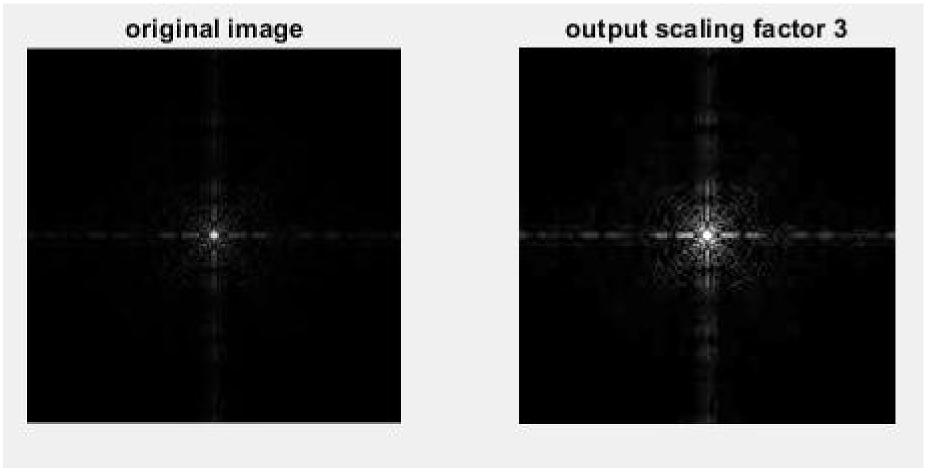
Where s and r are the pixel values of the output and the input image and c is a constant.

The opposite is true of the inverse log transformation.

Code

```
function log_transformation()

x='log.jpg';

a=importdata(x);

g=rgb2gray(a);

dv=im2double(g);

o3=5*log(1+dv);

subplot(1,2,1),imshow(g);title('original image');

subplot(1,2,2),imshow(o3);title('output scaling factor 5');

End
```

Output



Discussion

During log transformation, the dark pixels in an image are expanded as compare to the higher pixel values. The higher pixel values are kind of compressed in log transformation. The value of c in the log transform adjust the enhancement.

3.3 Gamma Correction

Theory

Power Law Transformation is called as Gamma Correction. As in the case of the log transformation, power-law curves with fractional values of map a narrow range of dark input values into a wider range of output values, with the opposite being true for higher values of input levels. Unlike the log function, however, we notice here a family of possible transformation curves obtained simply by varying γ.

These transformations can be given by the expression:

$s=cr^\gamma$

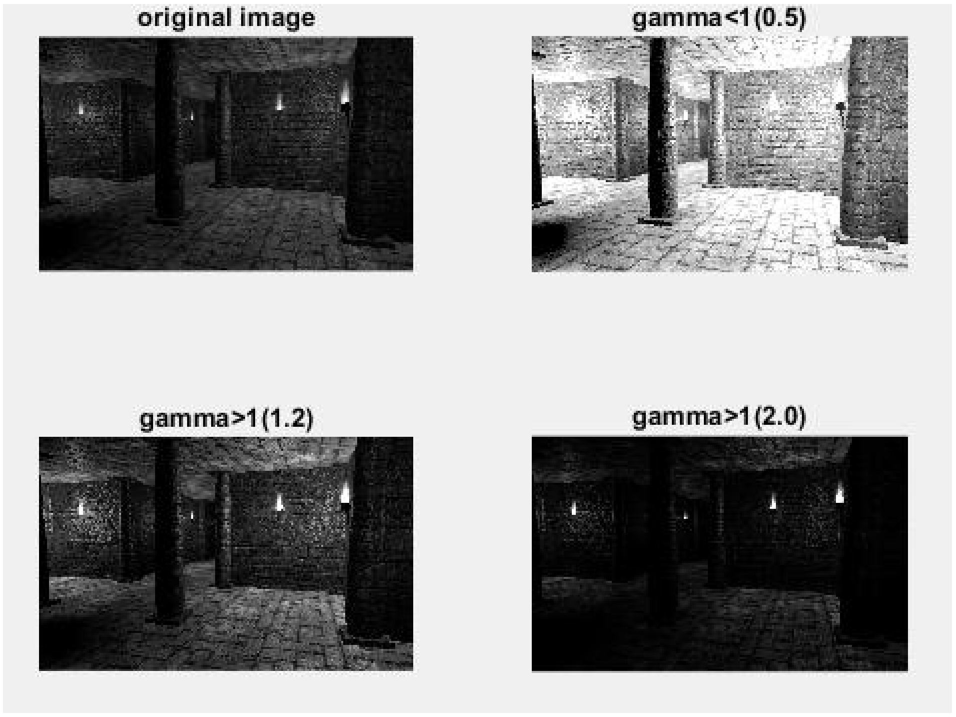
Where c and γ are positive constant. By convention, the exponent in the power-law equation is referred to as gamma. The process used to correct these power-law response phenomena is called gamma correction.

Code

```
function gamma_correction()

x='gamma.jpg';

a=importdata(x);

g=rgb2gray(a);

dv=im2double(g);

o1=2*(dv.^0.5);

o2=2*(dv.^1.2);

o3=2*(dv.^2.0);

  subplot(2,2,1),

imshow(g);

title('original image');

subplot(2,2,2),

imshow(o1);
```

```
title('gamma<1(0.5)');

subplot(2,2,3),imshow(o2);title('gamma>1(1.2)');

subplot(2,2,4),imshow(o3);title('gamma>1(2.0)');

end
```

Output



Discussion

This type of transformation is used for enhancing images for different type of display devices. The gamma of different display devices is different. We see that the best enhancement in terms of contrast was obtained with gamma=1.2

4. .Some Basic Intensity Transformation Functions

4.1 Histogram Equalization

Theory

The process of adjusting intensity values can be done automatically using histogram equalization. Histogram equalization involves transforming the intensity values so that the histogram of the output image approximately matches a specified histogram.

It accomplishes this by effectively spreading out the most frequent intensity values, i.e. stretching out the intensity range of the image. This method usually increases the global contrast of images when its usable data is represented by close contrast values. This allows for areas of lower local contrast to gain a higher contrast.

Code

Raw code

```
function hist_eq()

x = imread('hist.jpg');

Im = rgb2gray(x);

[height,width] = size(Im);



%calculate the number of pixels with the same grey level

PixelNumSame = zeros(1,256);%there are 256 grey levels 0-255

for j = 1:height

    for k = 1:width

        PixelNumSame(Im(j,k)+1) = PixelNumSame(Im(j,k)+1)+1;

    end

end
```

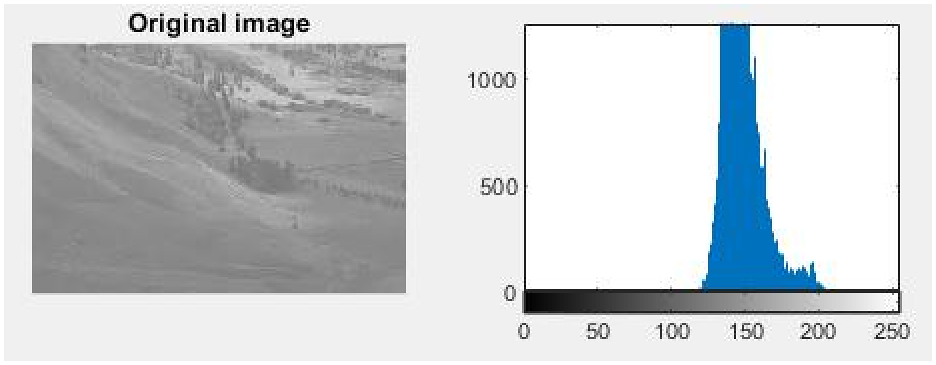
```
%calculate the probability density

ProDensity = zeros(1,256);

for i = 1:256

        ProDensity(i) = PixelNumSame(i) / (height * width);

end


%calculate the probability density after the equalization

ProDenAfter = zeros(1,256);

for i = 1:256

        if i == 1

                ProDenAfter(i) = ProDensity(i);

        else

                ProDenAfter(i) = ProDensity(i)+ProDenAfter(i-1);

        end

end


%calculate the equalized grey level

GreyLevelEq = uint8(ProDenAfter*255);


%map the image

for j = 1:height

        for k = 1:width

                Im2(j,k) = GreyLevelEq(Im(j,k)+1);

        end
```

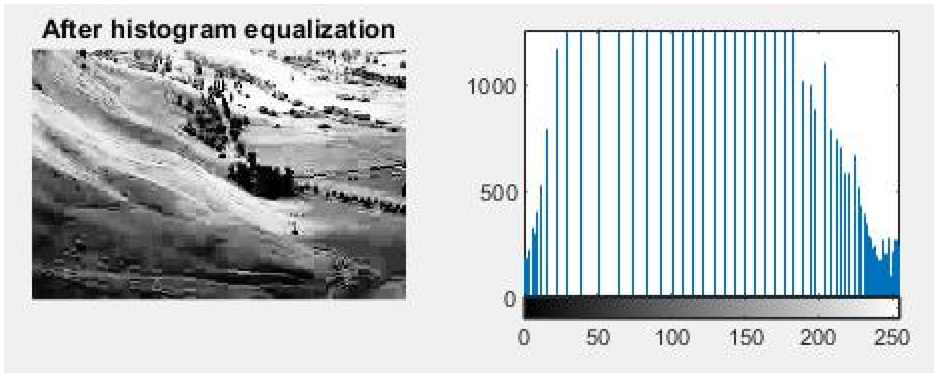
```
end


figure;

subplot(2,2,1)

imshow(Im);

title('Original image');

subplot(2,2,2);

imhist(Im);

figure;

%show the new image and histogram

subplot(2,2,1);

imshow(Im2);

title('After histogram equalization');

subplot(2,2,2);

imhist(Im2);
```

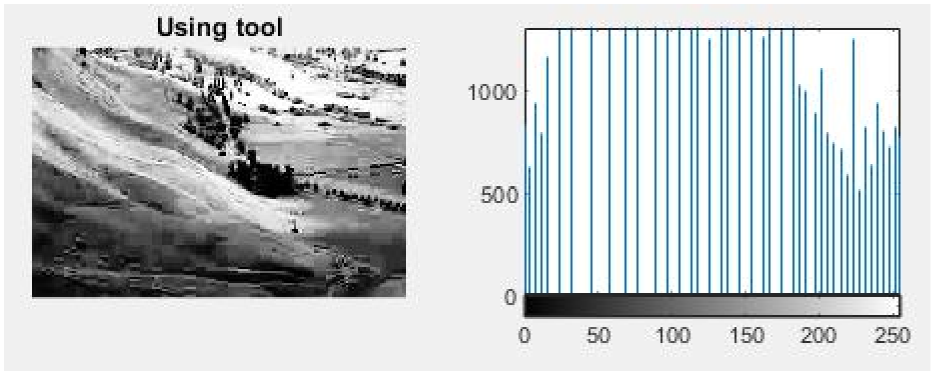
With toolbox

```
x=histeq(Im);

figure;

subplot(2,2,1);

imshow(x);

title('Using tool');

subplot(2,2,2);

imhist(x);

end
```

Output







Discussion

A color histogram of an image represents the number of pixels in each type of color component. Histogram equalization cannot be applied separately to the Red, Green and Blue components of the image as it leads to dramatic changes in the image's color balance. However, if the image is first converted to another color space, like HSL/HSV color space, then the algorithm can be applied to the luminance or value channel without resulting in changes to the hue and saturation of the image. In the figure above, toolbox is slightly efficient than the raw code.

4.2 Histogram Matching

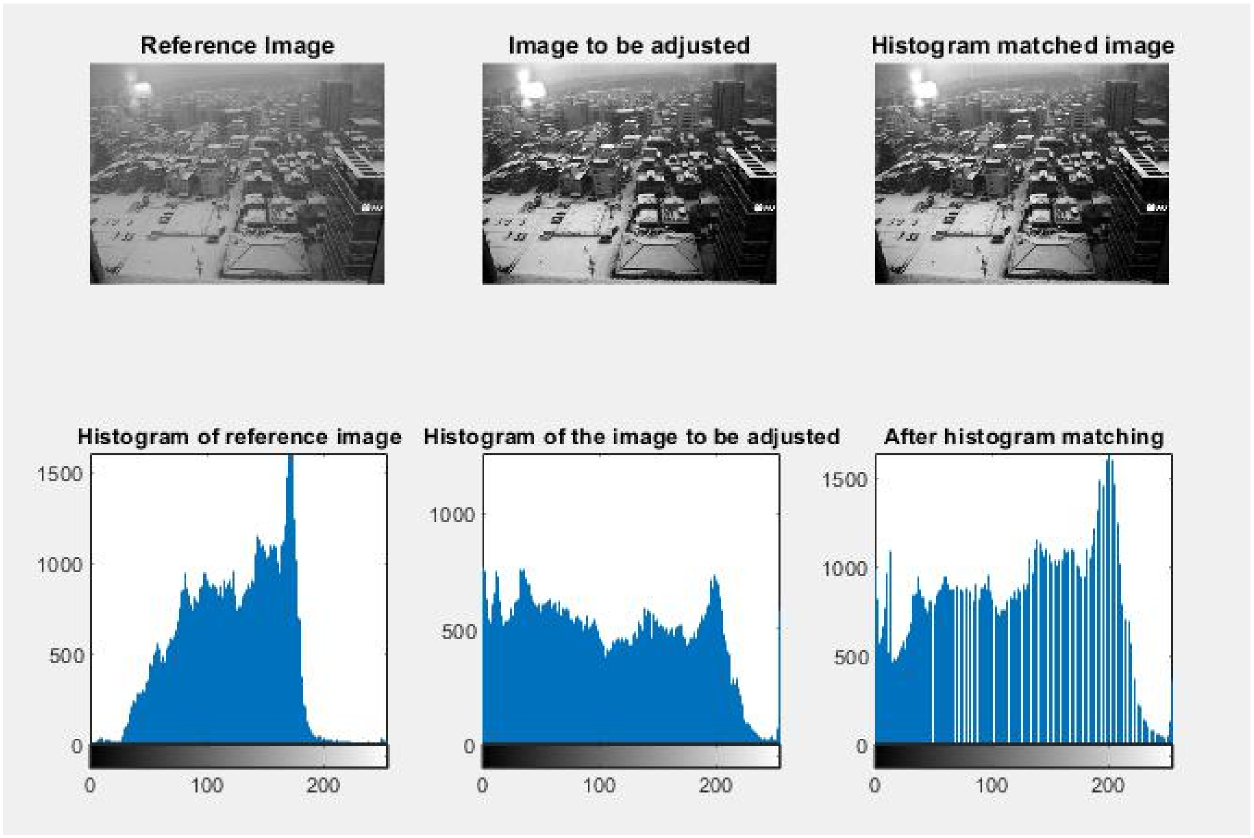
Theory

Histogram matching is a process where a time series, image, or higher dimension scalar data is modified such that its histogram matches that of another (reference) dataset. A common application of this is to match the images from two sensors with slightly different responses, or from a sensor whose response changes over time.

Code

```
function hist_match()

im1 = imread('cameraman.tif');

im2 = imread('board.tif');

M = zeros(256,1,'uint8');

hist1 = imhist(im1);

hist2 = imhist(im2);

cdf1 = cumsum(hist1) / numel(im1);

cdf2 = cumsum(hist2) / numel(im2);

  for idx = 1 : 256

        diff = abs(cdf1(idx) - cdf2);

        [~,ind] = min(diff);

        M(idx) = ind-1;

end

out = M(double(im1)+1);

subplot(2,3,1),imshow(im1);title('Reference Image');

subplot(2,3,2),imshow(im2);title('Image to be adjusted');

subplot(2,3,3),imshow(out);title('Histogram matched image');

subplot(2,3,4),imhist(im1);title('Histogram of reference image');
subplot(2,3,5),imhist(im2);title('Histogram of the image to be adjusted');

subplot(2,3,6),imhist(out);title('After histogram matching');

end
```

Output



Discussion

In the discussion here the two datasets are assumed to have the same range of values. The size of the two datasets do not need to be the same.

As can be seen in the examples above, the matching introduces gaps in the histograms. This is to be expected since the histograms are being distorted. This is especially so for discrete datasets such as images, it can often be reduced for continuous data by creating finer histogram bins.

5. Main Function

```
%2.1 bilinear interpolation

x = 'onion.png';

out_dims = [405 594];

out = bilinearInterpolation(x, out_dims)

pause(2)


%3.1 contrast stretching

contrast_stretching()

pause(2)


%3.2 log transformation

log_transformation()

pause(2)


%3.3 gamma correction

gamma_correction()

pause(2)


%4.1 histogram equalization
```

```
hist_eq()

pause(2)


%4.2 histogram matching

hist_match()
```