Lab 1: Basics of Image Processing

Brian Hosler & Sarah Peachey

January 18, 2018

Abstract

Stuff about images and processing them

Contents

1 Contrast Enhancement

This section involved contrast enhancement, and a comparison of various different techniques.

1.1 Gamma Correction

First, a function, `Gcorrection.m`, was made, to do contrast enhancement through gamma correction. Given a grayscale image, and a value for gamma, the function rescales every pixel to a value between 0 and 1. Each pixel value is then raised to the power of gamma before being cast back to an intiger between 0 and 255. The code is shown below.

```
function [ img_out ] = Gcorrection(img_in, gama)     1
%Does gamma correction using the equation:           2
%   new=255*(old/255)^gamma                           3
    img_out=uint8(255*(double(img_in)/255).^gama);    4
end                                                   5
```

1.2 Effects of Gamma Correction

The Gcorrection function was then used with varying gamma arguments on the same photo to show the effects of gamma being larger than, smaller than, and equal to, unity. When gamma was set to 1, the mean squared error between the original and altered image was computed, to show that when gamma is equal to 1, every value is mapped to itself.

```
%read in the image                                        1
pout=imread('Assignment_1_Files/pout.tif');               2
figure                                                    3
%Plot .4 enhanced image and histogram                     4
subplot(2,3,1)                                            5
imshow(Gcorrection(pout,.4))                              6
title('\gamma=0.4')                                      7
subplot(2,3,4)                                            8
imhist(Gcorrection(pout,.4))                              9
                                                         10
%Plot unenhanced image and histogram                     11
subplot(2,3,2)                                            12
imshow(Gcorrection(pout,1))                               13
title(sprintf('\\gamma=1\nMSE_from_original:_%d',immse(pout, 14
    Gcorrection(pout,1))))
subplot(2,3,5)                                            15
imhist(Gcorrection(pout,1))                               16
                                                         17
%Plot 2.1 enhanced image and histogram                   18
subplot(2,3,3)                                            19
imshow(Gcorrection(pout,2.1))                             20
title('\gamma=2.1')                                      21
subplot(2,3,6)                                            22
imhist(Gcorrection(pout,2.1))                             23
```

1.3 Histogram Equalization

Finally, Gamma correction was compared to histogram equalization using the photo MoonPhobos.tif. This photo has a bimodal distribution, with a concentration of pixel values near zero, and another near 255.

```matlab
%Read in new photo                                                    1
moonHobos=imread('Assignment_1_Files/MoonPhobos.tif');               2
figure                                                               3
%Plot the enhanced image                                             4
subplot(1,2,1)                                                       5
imshow(Gcorrection(moonHobos,.3))                                    6
title('\gamma=.3')                                                  7
subplot(1,2,2)                                                       8
imshow(histeq(moonHobos,256))                                       9
title('HistEQ')                                                     10
                                                                    11
figure                                                              12
%Plot histograms of the enhanced image                              13
subplot(1,2,1)                                                      14
imhist(Gcorrection(moonHobos,.3))                                   15
title('\gamma=.3')                                                 16
subplot(1,2,2)                                                      17
imhist(histeq(moonHobos,256))                                      18
title('HistEQ')                                                    19
```

The Gcorrection function is given an input γ less than 1, greater than 1 and equal to one. As seen in Figure 4, when the $\gamma = 0.4 < 1$ the picture becomes much lighter. Also, the image histogram is shifted to higher values and it's middle and high values are compressed. This results in the picture having higher grey values over a smaller range. When $\gamma = 1$ the original image is shown, therefore the image histogram can be used for reference. This is because any value raised to the first power is itself. When $\gamma = 2.1 < 1$ the image pixel values are lower but occupy a slightly larger range than the original image. The original image has most of it's pixel values concentraited in the center, around 127. Gamma correction does not take advantage of the full range of pixel values available in this instance, because it performs a unidirecitonal shift of the histogram, instead of spreading the concentration at the center.
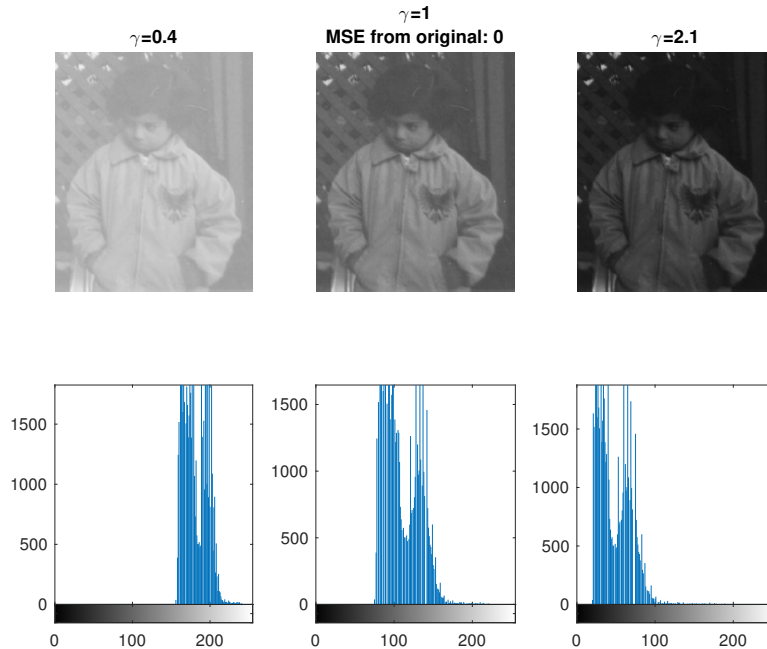

Figure 1: Varying gamma in Gcorrection function

The Gcorrection function was used to edit moonPhobos.tiff and the best percieved version of the picture was at $\gamma = 0.3$. For comparison the MATLAB built in function histeq was also used. The results of both can be see in 2. As seen in 3, the Gcorrection actually seemed to spread out the values of historgram better, but by making the pixel values lower, the image became much darker, so some of the details are lost. Whereas histeq does a better job of making sure the darker details and lighter details are maintained.
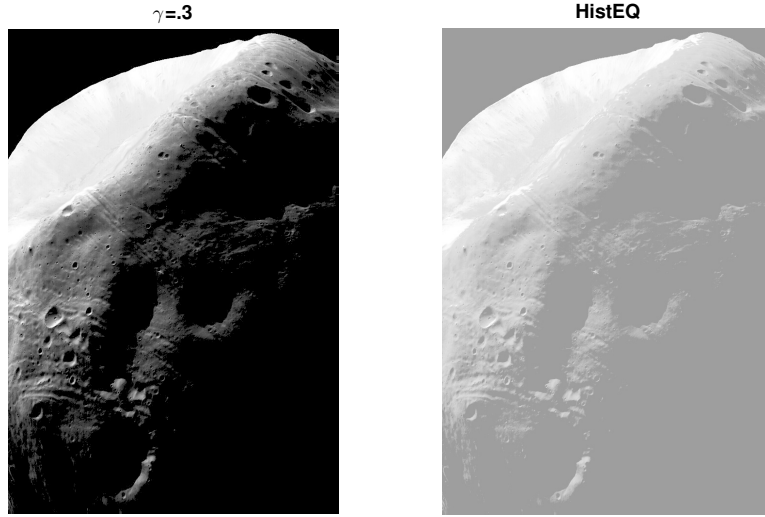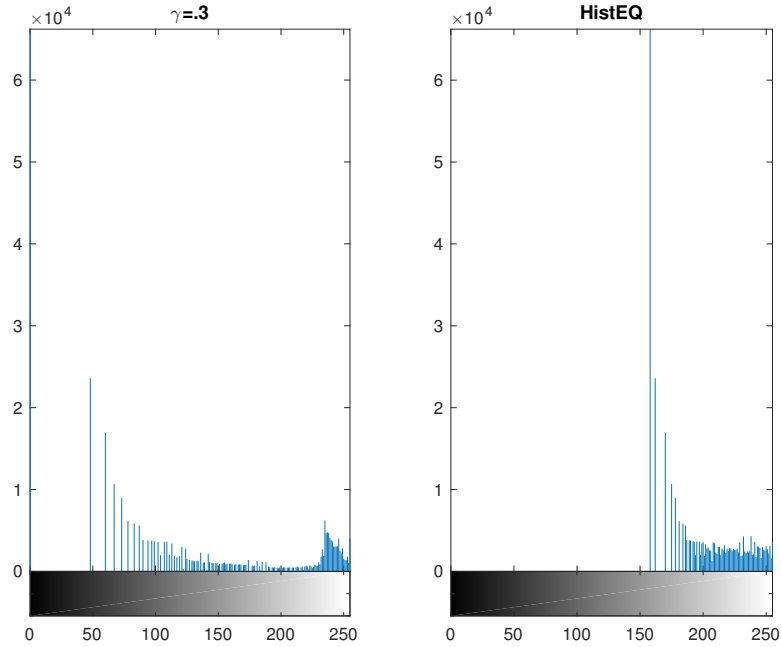

Figure 2: Gcorrection vs Histeq

Figure 3: Histograms of Gcorrection vs Histeq

2 High Frequency Content

This section investigates the effect that high-frequency content has on an image.

2.1 High Boot Filtering

High Boost filtering is a technique that increases high-frequencies. This is achieved by extracting the "sharpness" using a laplacian filter, and adding back to the original image a magnified version of that sharpness. The code to do so is shown below.

```matlab
function [ img_out ] = HBfilt(img_in, alph)       1
%High boost filtering using a laplaccian filter   2
    img_out=img_in+uint8(alph*conv2(double(img_in),[0 -.25 0;   3
        -.25 1 -.25; 0 -.25 0],'same'));
end                                               4
```

2.2 Effect of High Boost Filtering

An image of the moon is filtered using the HBfilt function to better see the surface texture.

```matlab
%Read in and filter the moon image                1
moon=imread('Assignment_1_Files/moon.tiff');      2
figure                                            3
imshow(HBfilt(moon,2.4))                          4
title('\alpha=2.4')                               5
```

2.3 High Boost Filtering to Sharpen an Image

Finally, an out-of-focus image is sharpened using High Boost filtering. Various alpha values were used to find the sharpest image.

```matlab
%Read in a blurry image and high-boost filter it  1
oof=imread('Assignment_1_Files/outoffocus.tif');  2
figure                                            3
imshow(HBfilt(oof,4))                             4
title('\alpha=4')                                 5
%High frequency noise added with increasing alpha(7)  6
```
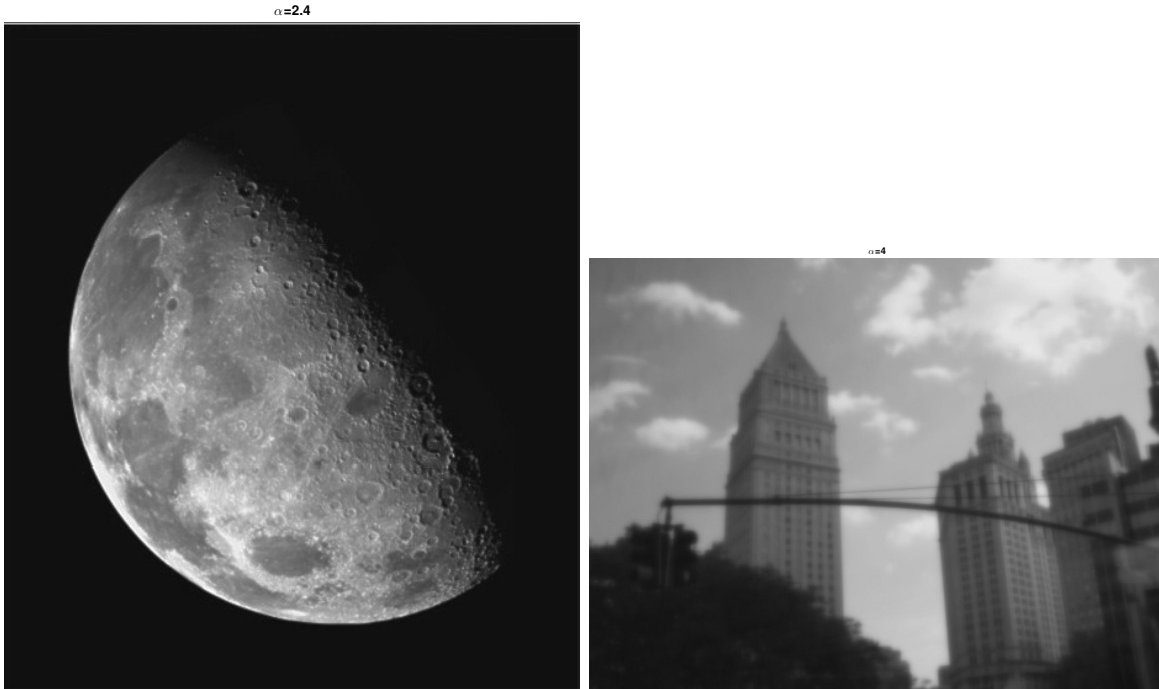
Figure 4: Varying gamma in Gcorrection function

3   Filtering Noise

3.1   Average and Median Filtering

```matlab
%Read in two noidy images                                          1
pep1=imread('Assignment_1_Files/peppersNoise1.tiff');              2
pep2=imread('Assignment_1_Files/peppersNoise2.tiff');              3
figure                                                             4
                                                                   5
%Denoise images with a 3x3 median filter                           6
subplot(4,2,1)                                                     7
imshow(medfilt2(pep1,[3,3]))                                       8
title(sprintf('peppersNoise1\nMedian_3x3'))                       9
subplot(4,2,2)                                                     10
imshow(medfilt2(pep2,[3,3]))                                       11
title(sprintf('peppersNoise2\nMedian_3x3'))                       12
                                                                   13
%Denoise images with a 5x5 median filter                           14
subplot(4,2,3)                                                     15
imshow(medfilt2(pep1,[5,5]))                                       16
title('Median_5x5')                                               17
subplot(4,2,4)                                                     18
imshow(medfilt2(pep2,[5,5]))                                       19
title('Median_5x5')                                               20
                                                                   21
%Denoise images with a 3x3 averaging filter                        22
subplot(4,2,5)                                                     23
imshow(uint8(filter2(ones(3,3)/9,pep1)))                           24
title('Averaging_3x3')                                            25
subplot(4,2,6)                                                     26
imshow(uint8(filter2(ones(3,3)/9,pep2)))                           27
```

```matlab
title('Averaging_3x3')                                        28
                                                              29
%Denoise images with a 5x5 averaging filter                   30
subplot(4,2,7)                                                31
imshow(uint8(filter2(ones(5,5)/25,pep1)))                     32
title('Averaging_5x5')                                        33
subplot(4,2,8)                                                34
imshow(uint8(filter2(ones(5,5)/25,pep2)))                     35
title('Averaging_5x5')                                        36
```

3.2 Edge Detection and Noise

```matlab
%Save the average and median filtered images                  1
pep1avg=uint8(filter2(ones(3,3)/9,pep1));                     2
pep1med=medfilt2(pep1,[3,3]);                                 3
figure                                                        4
th=60000;                                                     5
subplot(1,2,1)                                                6
sx=filter2([-1,0,1;-2,0,2;-1,0,1],pep1avg).^2;%Xgradient      7
sy=filter2([-1,0,1;-2,0,2;-1,0,1].',pep1avg).^2;%Ygradient    8
imshow((sx+sy)>th)%Magnitude squared                          9
subplot(1,2,2)                                                10
sx=filter2([-1,0,1;-2,0,2;-1,0,1],pep1med).^2;%Xgradient      11
sy=filter2([-1,0,1;-2,0,2;-1,0,1].',pep1med).^2;%Ygradient    12
imshow((sx+sy)>th)%Magnitude squared                          13
```

peppersNoise1
Median 3x3

peppersNoise2
Median 3x3

Median 5x5

Median 5x5

Averaging 3x3

Averaging 3x3

Averaging 5x5

Averaging 5x5