

Lab 1: Basics of Image Processing

Brian Hosler & Sarah Peachey

January 17, 2018

Contents

- Part 1
- Part 2
- Part 3

```
%Assignemnt 1
%Brian Hosler and Sarah Peachey
```

Part 1

```
%1
%The Gcorrection function
type('Gcorrection.m')

%2
%read in the image
pout=imread('Assignment_1_Files/pout.tif');
figure
%Plot .4 enhanced image and histogram
subplot(2,3,1)
imshow(Gcorrection(pout,.4))
title('\gamma=0.4')
subplot(2,3,4)
imhist(Gcorrection(pout,.4))
%Plot unenhanced image and histogram
subplot(2,3,2)
imshow(Gcorrection(pout,1))
title(sprintf('\gamma=1\nMSE from original: %d',immse(pout,Gcorrection(pout,1))))
subplot(2,3,5)
imhist(Gcorrection(pout,1))
%Plot 2.1 enhanced image and histogram
subplot(2,3,3)
imshow(Gcorrection(pout,2.1))
title('\gamma=2.1')
subplot(2,3,6)
imhist(Gcorrection(pout,2.1))

%3
%Read in new photo
moonHobos=imread('Assignment_1_Files/MoonPhobos.tif');
figure
%Plot the enhanced image
subplot(1,2,1)
imshow(Gcorrection(moonHobos,.3))
title('\gamma=.3')
subplot(1,2,2)
imshow(histeq(moonHobos,256))
title('HistEQ')

figure
%Plot histograms of the enhanced image
```

```

subplot(1,2,1)
imhist(Gcorrection(moonHobos,.3))
title('\gamma=.3')
subplot(1,2,2)
imhist(histeq(moonHobos,256))
title('HistEQ')

function [ img_out ] = Gcorrection(img_in, gama)
%Does gamma correction using the equation:
% new=2558(old/255)^gamma
img_out=uint8(255*(double(img_in)/255).^gama);
end

```

The Gcorrection function is given an input γ less than 1, greater than 1 and equal to one. As seen in Figure 1, when the $\gamma = 0.4 < 1$ the picture becomes much lighter. Also, the image histogram is shifted to higher values and it shrank in width. Meaning that the picture has higher grey values over a smaller range. When $\gamma = 1$ the original image is shown, therefore the image histogram can be used for reference. When $\gamma = 2.1 > 1$ the image pixel values are lower but occupy a larger range than the original image.

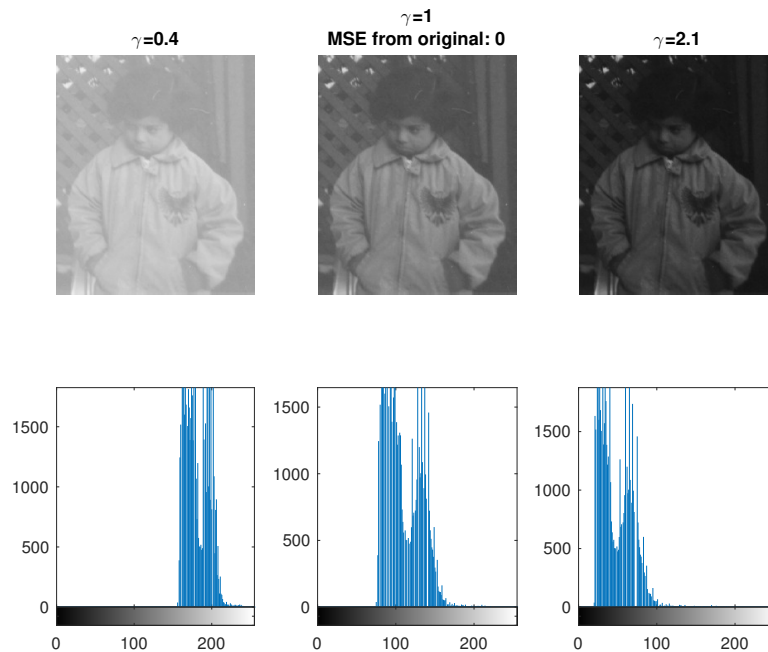


Figure 1: Varying gamma in Gcorrection function

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

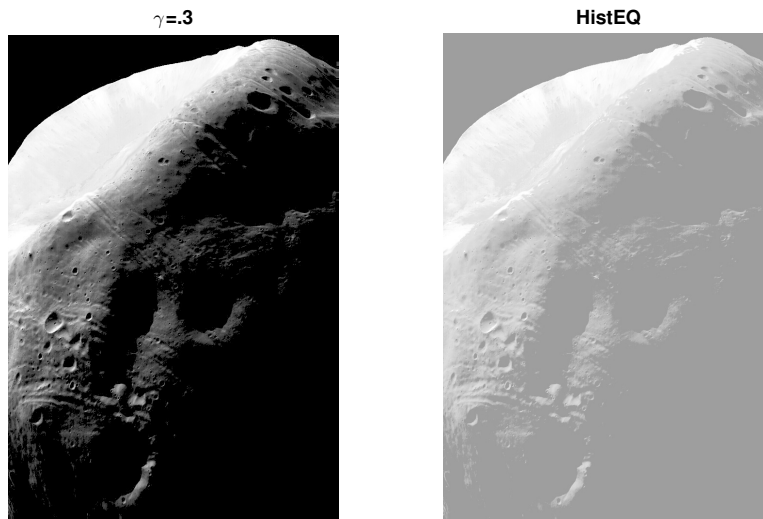


Figure 2: Gcorrection vs Histeq

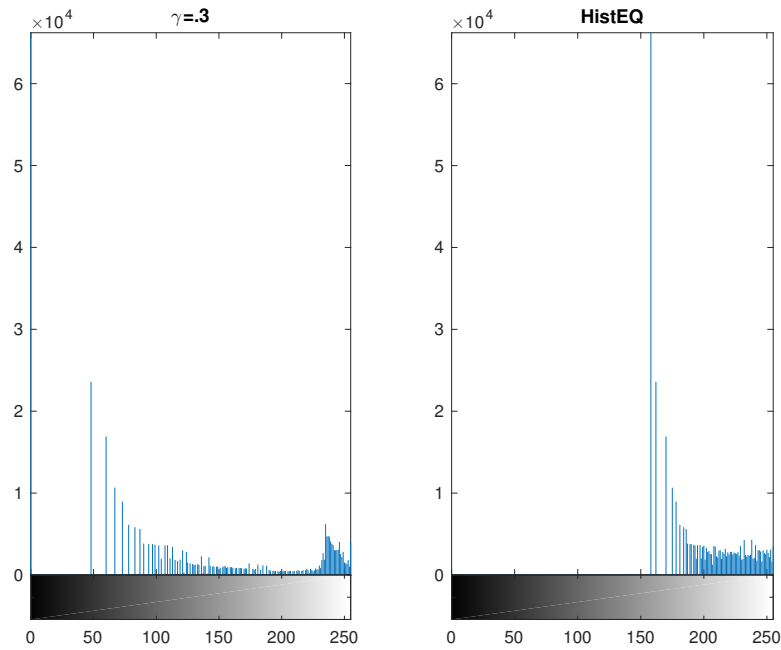


Figure 3: Histograms of Gcorrection vs Histeq

Part 2

```
%1
%Our High-Boost filter function
type('HBfilt.m')

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

%3
%Read in a blurry image and high-boost filter it
oof=imread('Assignment_1_Files/outoffocus.tif');
figure
imshow(HBfilt(oof,4))
title('\alpha=4')
%High frequency noise added with increasing alpha(7)

function [ img_out ] = HBfilt(img_in, alph)
```

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

Warning: Image is too big to fit on screen; displaying at 67%

$\alpha=2.4$



$\alpha=4$



Part 3

```
%1
%Read in two noisy images
pep1=imread('Assignment_1_Files/peppersNoise1.tiff');
pep2=imread('Assignment_1_Files/peppersNoise2.tiff');
figure
%Denoise images with a 3x3 median filter
subplot(4,2,1)
imshow(medfilt2(pep1,[3,3]))
title(sprintf('peppersNoise1\nMedian 3x3'))
subplot(4,2,2)
imshow(medfilt2(pep2,[3,3]))
title(sprintf('peppersNoise2\nMedian 3x3'))
%Denoise images with a 5x5 median filter
subplot(4,2,3)
imshow(medfilt2(pep1,[5,5]))
title('Median 5x5')
subplot(4,2,4)
imshow(medfilt2(pep2,[5,5]))
title('Median 5x5')
%Denoise images with a 3x3 averaging filter
subplot(4,2,5)
imshow(uint8(filter2(ones(3,3)/9,pep1)))
title('Averaging 3x3')
subplot(4,2,6)
```

```

imshow(uint8(filter2(ones(3,3)/9,pep2)))
title('Averaging 3x3')
%Denoise images with a 5x5 averaging filter
subplot(4,2,7)
imshow(uint8(filter2(ones(5,5)/25,pep1)))
title('Averaging 5x5')
subplot(4,2,8)
imshow(uint8(filter2(ones(5,5)/25,pep2)))
title('Averaging 5x5')

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

```


peppersNoise1
Median 3x3



Median 5x5



Averaging 3x3



Averaging 5x5



peppersNoise2
Median 3x3



Median 5x5



Averaging 3x3



Averaging 5x5

