150001C
G. Abarajithan

Assignment 1

1. Gamma Correction

We use power law transformation for the gamma correction. This is usually used to bring out details in areas of highlights or shadows. Transformation is: (imOut = c * imIn .^ gamma;) with c = 1 in general.

Gamma is obtained from user. Gamma > 1 makes the image darker and brings out details in highlights / whites. Gamma < 1 makes image brighter and brings out details in shadows / blacks.

```
result = (c255) * (imGrayDNormal .^ gamma);
```

0.9

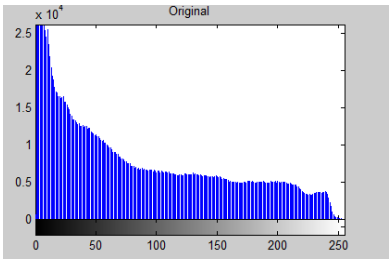
Original

1.8

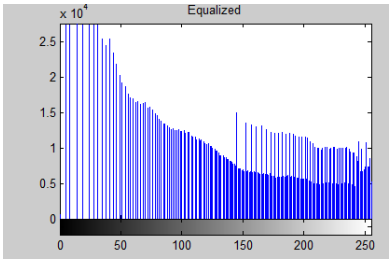
2. Histogram Equalization

Equalizing histogram improves contrast. This is achieved by making the histogram like a uniform distribution.

The transfer function is derived from cumulative probability density of input histogram.



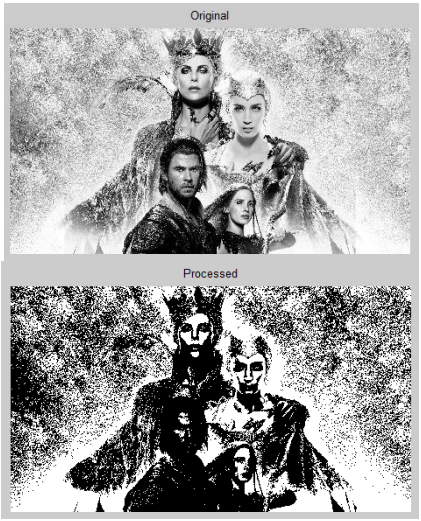
Original

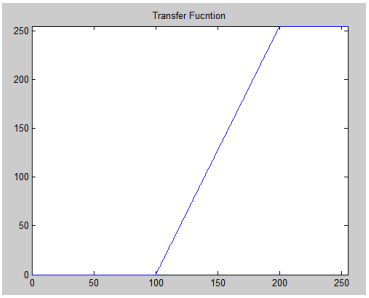


Equalized

3. Intensity Windowing

Used to isolate the data in specific range of intensities. This is done by linearly stretching the original image intensities using a transfer function. I have done this by writing a code to generalize contrast stretching and using (100,0) and (200,255) as points.






```
% main fucntion

m1 = (y1/x1);
c1 = 0;
m2 = (y2-y1)/(x2-x1);
c2 = -x1*m2 +y1;
m3 = (255-y2)/(255-x2);
c3 = -x2*m3 +y2;
```

```
% Plotting the Transfer Function

inp = 0:1:255;
outp = 0:1:255;

for i = 1:255
    m = 0;
    c = 0;
    p = inp(i);

        if p < x1
            m = m1;
            c = c1;
        elseif p < x2
            m = m2;
            c = c2;
        else
            m = m3;
            c = c3;
        end
    outp(i) = m* p + c;
end
```

4. Gaussian Blur / Smoothing

Spatial filter with a gaussian kernel. We define the size of the kernel. Blurring is increased with the increase of standard deviation (sigma). Pixels are weighed by the gaussian kernel and summed to obtain the result.

```
% Gaussian Kernel
H = fspecial('gaussian', hsize, sigma);

% Apply filter

imOut = imfilter(imIn, H, 'replicate');
```

Original Image

Size = 40, Sigma = 10

Size = 40, Sigma = 160

Size = 10, Sigma = 10

Size = 40, Sigma = 40

5. Zoom

Nearest neighbor algorithm simply fills the gaps with the intensity of the nearest pixel. This results in an obviously pixelated image.

Bi linear interpolation fills the spaces with the predicted / interpolated values. This results in a smoother image.

Original Image | Nearest Neighbor x 4 | Bilinear Interpolation x 4

```
function ZoomNear2(imName, scale)

% Reading the Image
imIn = imread(imName);

[h,w,d] = size(imIn);

imOut = uint8(zeros(h*scale, w*scale, d));

for i = 1:h*scale
    for j = 1:w*scale
        imOut(i,j,:) = imIn( min(floor(i/scale)+1, h), min(floor(j/scale)+1, w),
    end
end
```

Nearest Neighbor

```
for k = 1:3
    for i = 1:m-1
        for j = 1:n-1

            A = im(i,j,k);
            B = im(i,j+1,k);
            C = im(i+1, j, k);
            D = im(i+1, j+1, k);

            for jj = 0:factor
                ix = double(A+ (B-A) * (jj/factor));
                jx = double(C+ (D-C) * (jj/factor));

                for ii = 0:factor
                    imOut ((i-1)* factor +1 +ii+1, (j-1)*factor+1+jj+1,k) = double(ix+11*(jx-ix)/factor);
                end
            end
        end
    end
end
```

Bilinear Interpolation

5. Rotate by Angle in degrees

A mathematical transformation using polar coordinates is used here. A mesh grid of image is rotated in the polar form and then converted to cartesian form. This is the most common algorithm in use. At the end, size of the new image must be calculated.

```
[y,x] = meshgrid(1:n, 1:n);
[t,r] = cart2pol(x-midx, y-midy);
t1 = radtodeg(t+deg);
t = degtorad(t1);
[x,y] = pol2cart(t,r);

tempx = round(x +midx);
tempy = round(y +midy);
```