|  |  |
| --- | --- |
| **Homework #1**  *Digital Image Processing(EEE5320), 2019-2* | Due Date: 2019. 09.30 |

1. Intensity Transformations (60pt)

**Figure 1. input1 Figure 2. Input2**

* 1. Piecewise Linear Transformation (20pt)

1. Transform ‘input1.jpg’ image using piecewise linear functions. Complete the remaining part of provided PiecewiseLinearTr.m function and run hw1\_1.m file

|  |
| --- |
| function output = PiecewiseLinearTr(input,a,b) %  % PiecewiseLinearTr(IM,A,B) applies a piecewise linear transformation to the pixel values  % of the input image INPUT, where A and B are vectors containing the x and y coordinates  % of the ends of the line segments. INPUT can be of type DOUBLE,  % and the values in A and B must be between 0 and 1 (normalized intensity values). %  % For example:  %  % PiecewiseLinearTr(x,[0,1],[1,0])  %  % simply do negative transform inverting the pixel values.  %    if length(a) ~= length (b)  error('Vectors A and B must be of equal size');  end    ***% Complete the remaining part***  % set output size equal to input size  output= zeros(size(input));  % iterate for every vector  for i = 1:length(a)-1  % set vector  a1 = a(i);  a2 = a(i+1);  b1 = b(i);  b2 = b(i+1);    % mask image  filter = (input >= a1) & (input <= a2);    % transformation  % Equation: s = ((b2-b1)/(a2-a1))(r-a1) + b1  m = (b2-b1)/(a2-a1);  % merge output  output = output + filter.\*(m\*(input-a1) + b1);  end |
| **[PiecewiseLinearTr.m]** |

|  |
| --- |
| im = im2double(imread(input1.jpg'));    im\_Tr = PiecewiseLinearTr(im, [0,1], [1,0]);  im\_Tr2 = PiecewiseLinearTr(im, [0 .25 .5 .75 1],[0 .75 .25 .5 1]);    figure(1);  x\_axis = [0:1/255:1];  subplot(1,2,1);  plot(x\_axis, PiecewiseLinearTr(x\_axis, [0,1], [1,0]));  subplot(1,2,2);  plot(x\_axis, PiecewiseLinearTr(x\_axis, [0 .25 .5 .75 1],[0 .75 .25 .5 1]));    figure(2);  subplot(1,2,1);  imshow(im\_Tr);  subplot(1,2,2);  imshow(im\_Tr2);    figure(3);  subplot(1,2,1);  imhist(im\_Tr);  subplot(1,2,2);  imhist(im\_Tr2); |
| **[hw1\_1.m]** |

1. Plot the two transformation functions.
2. Display the histograms of the output images and the output images itself.
3. Explain your implementation and discuss your results.
   1. Sigmoid Transformation (20pt)
4. Transform ‘input1.png’ using sigmoid transformation = for = [5, 10, 20, 40] (a range of x is from 0 to 1).

|  |
| --- |
| function output = SigmoidTr(input, gamma)  % Returns transformed image by sigmoid transformation with gamma where INPUT is a gray scale input image  ***% Complete the remaining part***  % transformation  output = 1 ./ (1 + exp(-1 .\* gamma .\* (input-0.5))); |
| **[SigmoidTr.m]** |

|  |
| --- |
| im = im2double(imread('input1.png'));    gamma = [5, 10, 20, 40];    figure(1);  for i = 1:length(gamma)  hold on;  x\_axis = [0:1/255:1];  f\_Tr = SigmoidTr(x\_axis, gamma(i));  plot(x\_axis, f\_Tr);  legend({'\gamma = 5', '\gamma = 10', '\gamma = 20', '\gamma = 40'}, 'Location','northwest');  end    figure(2);  for i = 1:length(gamma)  im\_Tr = SigmoidTr(im, gamma(i));  subplot(2,2,i);  imshow(im\_Tr);  title(['\gamma = ',num2str(gamma(i))]);  end |
| **[hw1\_2.m]** |

1. Plot the transformation functions.
2. Display the histograms of the output images and the output images itself.
3. Explain your implementation and discuss your results.
   1. Power Law Transformation (20pt)
4. Transform ‘input2.png’ using power law transformations for = [0.2, 0.4, 2.5, 5].

|  |
| --- |
| function output = PowerLawTr(input, gamma)  % Returns transformed image by power law transformation with gamma where INPUT is a gray scale input image  ***% Complete the remaining part***  % transformation  output= input.^(gamma); |
| **[PowerLawTr.m]** |

|  |
| --- |
| im = im2double(imread('input2.png'));    gamma = [0.2, 0.4, 2.5, 5];    figure(1);  for i = 1:length(gamma)  hold on;  x\_axis = [0:1/255:1];  f\_Tr = PowerLawTr(x\_axis, gamma(i));  plot(x\_axis, f\_Tr);  legend({'\gamma = 0.2', '\gamma = 0.4', '\gamma = 2.5', '\gamma = 5'}, 'Location','northwest');  end    figure(2);  for i = 1:length(gamma)  im\_Tr = PowerLawTr(im, gamma(i));  subplot(2,2,i);  imshow(im\_Tr);  title(['\gamma = ',num2str(gamma(i))]);  end |
| **[hw1\_3.m]** |

1. Plot the transformation functions.
2. Display the histograms of the output images and the output images itself.
3. Explain your implementation and discuss your results.
4. Histogram Equalization (40pt)



**Figure 3. input3**

* 1. Load the attatched “input3.jpg” and display the histogram of the image.

|  |
| --- |
| %2-1 Load image and Display histogram  % load image  im = im2double(imread('input3.png'));  % fig 1: histogram of the input image  figure(1);  imhist(im);  title(['2-1. Histogram of Input Image']); |
| **[HW2.m]** |

* 1. Implement a Matlab function for histogram equalization.

**(You should not use ‘histeq’ built-in function)**

|  |
| --- |
| %2-2 implement histogram equalization function;    % image size  size\_im = size(im);  MN = size\_im(1) \* size\_im(2);  % make 256bins of histogram  hist = imhist(im,256);  L = length(hist);  % calculate pdf,cdf  pdf = hist ./ MN;  cdf = cumsum(pdf);  % histogram equalized array  s\_arr = round((L-1).\*cdf);  % map input image to histogram-equalized image  output = zeros(size(im));  for i = 1:L  mask = (im >= (i-1)/L) & (im <= i/L);  output = output + mask .\* (s\_arr(i)/L);  end    % fig 2: histogram of the output image  figure(2);  imhist(output);  title(['2-2. Histogram of Output Image']);  % fig 3: Output image  figure(3);  imshow(output);  title(['2-3. Output Image']); |
| **[HW2.m]** |

* 1. Display the histogram of the input image created in step 3 and the output image itself.
  2. Explain your implementation and discuss your results

1. Histogram Matching (40pt)

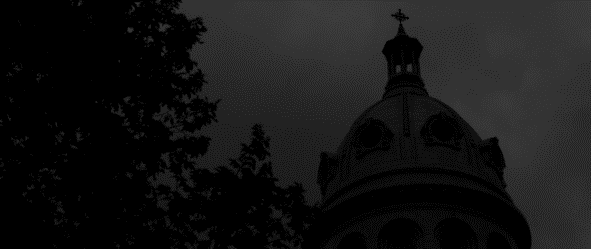
 

Figure 4. input4 Figure 5. Input4\_match

* 1. Implement a Matlab function for histogram matching to produce an image that looks like the provided image ‘input4\_match.jpg’.

**(You should not use ‘imhistmatch’ built-in function)**

* 1. Display the histogram of the images : ‘input4.jpg’, ‘input4\_match.jpg’.
  2. Display the histogram of the output image and the output image itself.
  3. Explain your implementation and discuss your results