TDI2131 Digital Image Processing (Week 3)
Tutorial 2

Note: All images used in this tutorial belong to the Image Processing Toolbox.

1. Gray Level Transformation Functions

   (a) Given the following gray level transformation functions in graph-form. Express the relation of
   \( s \) and \( r \) (in piecewise function if needed).

   (i) ![Graph for function (i)](image1.png)
   (ii) ![Graph for function (ii)](image2.png)

   (b) Perform the gray level transformations in (a) to the following 7x7 pixel, 256 gray level image
   separately.

   
<table>
<thead>
<tr>
<th>200</th>
<th>230</th>
<th>240</th>
<th>241</th>
<th>10</th>
<th>20</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>245</td>
<td>243</td>
<td>250</td>
<td>211</td>
<td>20</td>
<td>16</td>
<td>40</td>
</tr>
<tr>
<td>210</td>
<td>222</td>
<td>234</td>
<td>219</td>
<td>30</td>
<td>32</td>
<td>21</td>
</tr>
<tr>
<td>210</td>
<td>224</td>
<td>243</td>
<td>210</td>
<td>41</td>
<td>52</td>
<td>23</td>
</tr>
<tr>
<td>100</td>
<td>60</td>
<td>100</td>
<td>60</td>
<td>100</td>
<td>60</td>
<td>100</td>
</tr>
<tr>
<td>60</td>
<td>100</td>
<td>60</td>
<td>100</td>
<td>60</td>
<td>100</td>
<td>60</td>
</tr>
<tr>
<td>100</td>
<td>60</td>
<td>100</td>
<td>60</td>
<td>100</td>
<td>60</td>
<td>100</td>
</tr>
</tbody>
</table>

   What is the image obtained after each of these gray level transformations?

   (c) Write a function in Matlab to compute the transformed image and compare with the result
   obtained in part (b).

   Writing functions:
   
   ```matlab
   function [output1, output2, ... ] = function_name(input1, input2, ...)
   ```

   **Note:** You can write your code to process pixel by pixel (the same way how we process a 2D
   array in C++). However, Matlab actually provides other useful functions to avoid writing for-
   loops, and process everything in matrices instead.
2. **Power-Law Transformation**

(a) Given the Gamma correction function:

\[ s = c r^\gamma \]

where \( 0 \leq s \leq 1, 0 \leq r \leq 1 \)

Deduce the power-law transformation function, for \( s \) and \( r \) in the range of 0 to \( L - 1 \).

(b) With the deduced equation, write a Matlab function to process images with Gamma correction. Transform an image of your choice using the following parameters:

(i) \( c = 1, \gamma = 0.5 \)

(ii) \( c = 1, \gamma = 1.5 \)

3. **Contrast Stretching**

(a) To write a function to implement contrast stretching, which is defined as

\[ s = \frac{(r - r_{min})}{(r_{max} - r_{min})}(s_{max} - s_{min}) + s_{min} \]

Matlab has provided the function `imadjust` to adjust image intensity values easily. Consider the following example,

```matlab
>> im = imread('tire.tif');
>> x = imadjust(im, [0.25 0.75], [0.1 0.9]);
>> imshow(im), figure, imshow(x);
```

and try and figure out how to use it.

(b) Interestingly, the function `imadjust` also has a gamma parameter. Try to experiment with that parameter to achieve similar effects to Question 2. Or are they different?

4. **Histogram Processing**

(a) Display the histogram of an image using the function `imhist`. These are different ways to use it:

```matlab
>> imhist(im);
>> imhist(im, 128);
>> his = imhist(im);
```

`im` is the image array. Display the histogram of `pout.tif` and observe its characteristic. What can you tell by just observing this image’s histogram information?
(b) Histogram equalization can be performed by using the function `histeq`. These are different ways to use it:

```matlab
>> j = histeq(im);
>> [j,T] = histeq(im);
```

The second way of using `histeq` shown above, returns the gray level transformation values (all in `T`) that maps gray levels in the input image to gray levels in the output image `j`.

(c) Plot the graph of input gray levels vs. output gray levels that shows the transformation function for performing histogram equalization on the `pout.tif` image.

(d) What happens when you apply histogram equalization again for the second time?

5. **Histogram Processing (by hand)**

Given the following 6x6 grayscale image with 4 bits/pixel:

```
11 10 8 4 4 2
11 1 10 11 2 9
12 9 10 9 11 10
8 10 1 10 6 2
1 6 11 11 9 10
6 11 9 10 2 1
```

(a) Obtain the histogram of the image.
(b) Apply histogram equalization on the above image. (You can use both original histogram values or normalized histogram values)
(c) Determine the new intensity values of the histogram-equalized image.

**Homework**

1. Attempt coding your own version of histogram equalization in Matlab, and see if it works just like `histeq`.

2. **Image Subtraction**

   Subtracting two images (that are of the same size) results in the difference between the two images. Experiment with some images by subtracting the original image from their respective histogram-equalized image. Does it result in anything meaningful?

   *(Note: Sometimes, the resulting values could be either positive or negative value and therefore, the values need to be rescaled accordingly to see the correct picture.)*