Image Enhancement Problem

Objective and Learning Outcomes

The objective of this experiment is for students to enhance or improve a visually poor quality image. From this experiment, students will acquire the following skills:

1. Learn some basic image processing techniques in image enhancement.
2. Apply image processing technique in improving poor quality images.
3. Use Matlab along with its Image Processing Toolbox to solve image enhancement problems.

Introduction

Observe the image shown below in Figure 1. This image is said to have a dark poor contrast intensity as shown by its histogram given in Figure 2. A good image normally covers the entire dynamic range of the intensity levels as shown in Figure 3. Hence, the goal of image processing task is to enhance this image so that the image has better visual quality (quality measurement will be dealt with in the next experiment).

Figure 1

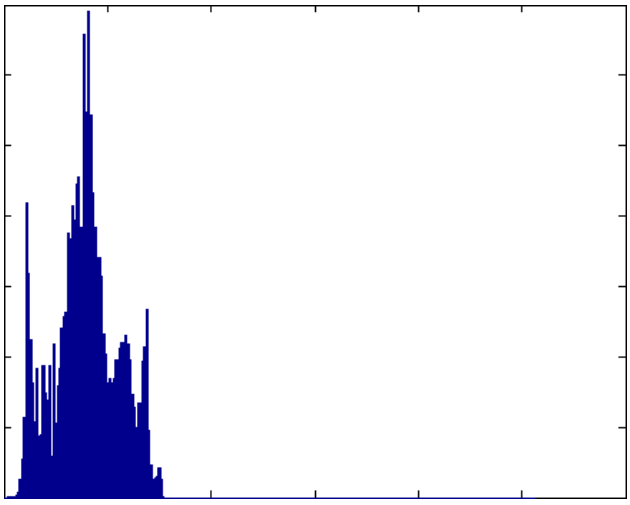
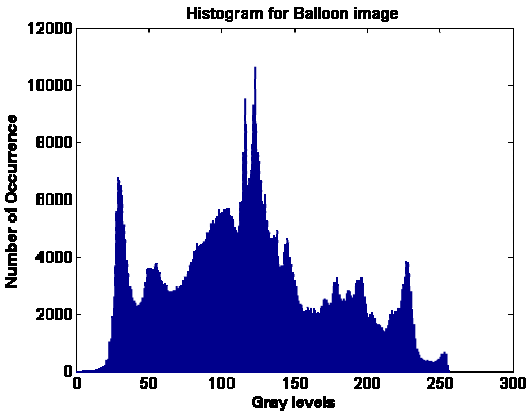


Figure 2

In doing so, there are a few approaches that one can use to enhance or improve the contrast of an image and the following are some of them.

1. Piece-wise Contrast stretching
2. Histogram Equalization
3. Histogram Matching

A good image

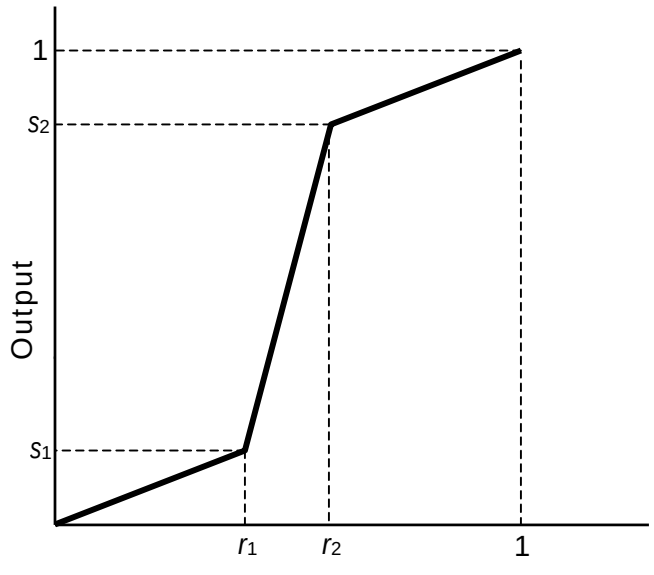


Histogram of good

Figure 3

Piece-wise Contrast Stretching

The basic idea behind contrast stretching is to linearly increase or decrease the contrast of the given image. This can be done by specifying the input/output relationship. For example, observe the following input/output relationship as shown in Figure 4 below. In this figure the intensities of the pixel have been normalized from 0 (black) to 1 (white). Input image intensity (*x*-axis) is denoted by *r* while output image intensity (*y*-axis) is denoted by *s*.



Figure

In this figure, small dynamic range input intensity (as represented by the difference between r_2 and r_1) is being mapped to a wider dynamic range at the output image (as represented by the difference between s_2 and s_1). Thus, we can see that this transformation function has stretched the contrast of the input image. In this manner we see that some of the darker or black pixels have been mapped to brighter pixels.

A few observations can be made from the mapping function shown in Figure 4.

1. If s_1 equals to r_1 **and** s_2 equals to r_2, then output image will be exactly identical to the input image. In this case there is no change in the contrast between the output and input image.

2. If $s_1 = 0$, $s_2 = 1$, and $r_1 = r_2$ then output image will consist of only black (0) and white (1) pixels. This transformation function is known as the binarizing function.

3. If $r_1 > s_1$ and $r_2 < s_2$ (as shown in Figure 3), then all pixels in between r_1 and r_2 of the input image will be stretched in between pixels s_1 and s_2 of the output image. Pixels less than r_1 in the input image will be darker in the output image and pixels greater than r_2 in the output image will appear brighter in the output image.

4. If $r_1 < s_1$ and $r_2 > s_2$, then all pixels in between r_1 and r_2 of the input image will be compressed in between pixels s_1 and s_2 of the output image. Pixels less r_1 (dark pixels) in the input image will be brighter in the output image and pixels greater than r_2 (bright pixels) in the output image will appear darker in the output image.

Histogram Equalization

The goal of histogram equalization is to spread out the contrast of a given image evenly throughout the entire available dynamic range, in this case

between 0 and 1. Figure 5 below shows the result of an image before and after histogram equalization operation.

Before Equalized

After Equalized

Figure 5

In histogram equalization technique, it is the probability density function (pdf) that is being manipulated. To make it simple, what histogram equalization technique does is that, it changes the pdf of a given image into that of a uniform pdf that spreads out from the lowest pixel value (0 in this case) to the highest pixel value (L – 1). This can be achieved quite easily if the pdf is a continuous function. However, since we are dealing with a digital image, the pdf will be a discrete function. Let suppose we have an image x, and let the dynamic range for the intensity r_k varies from 0 (black) to L – 1 (white). This pdf can be approximated using the probability based on the histogram $p(r_k)$ as follows:

$$pdf(x) = p(r_k) = \frac{total\ pixels\ with\ intensity\ r_k}{total\ pixels\ in\ image\ x} \qquad \text{Eq. (1)}$$

From this pdf, we can then obtain the cumulative density function (cdf) as follows:

$$cdf(x) = \sum_{k=0}^{L-1} p(r_k) \qquad \text{Eq. (2)}$$

where $p(r_k)$ is the probability for pixel of intensity r_k.

The output pixels from the histogram equalization operation is then equals to the cdf of the image or mathematically:

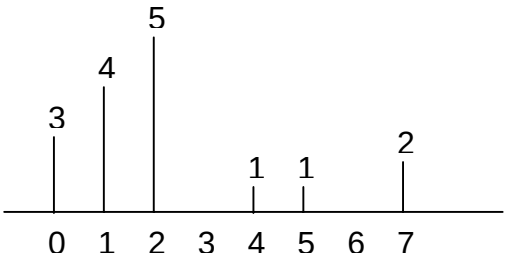
$$p(s_k) = \sum_{k=0}^{L-1} p(r_k) \qquad \text{Eq. (3)}$$

To get the value of the pixel, $p(s_k)$ need to be multiplied by $L - 1$ and then round it to the nearest integer.

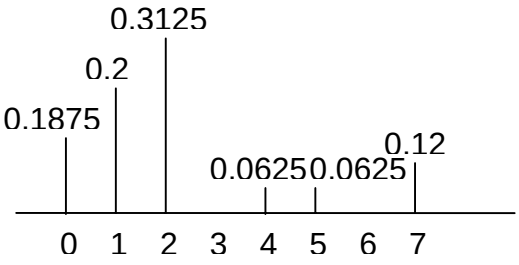
To illustrate the concept above, consider the following example with 4x4 matrix of a 3-bit image.

0	0	0	4
1	1	1	5
1	2	2	7
2	2	2	7

The histogram of this image is given as follows:



The pdf for this image can be computed simply by taking the histogram above and divided each value by the number of total pixels, i.e.



The cdf of this image matrix is then computed as an accumulation of the above histogram. Thus, using Eq. (2), we obtain the following cdf and the corresponding $p(s_k)$ sequence shown in the following table:

r	0	1	2	3	4	5	6	7
$p(r_k)$	0.1875	0.25	0.3125	0	0.0625	0.0625	0	0.125
$p(s_k)$	0.1875	0.4375	0.75	0.75	0.8125	0.875	0.875	1
s	1	3	5	5	6	6	6	7

From the above table, we see that pixel value '0' in the input (r) has been mapped to pixel value '1' in the output (s). Likewise pixel value '1' in the input has been mapped to pixel value '3' in the output. Similarly pixel values 2 and 3 in the input has been changed to pixel value 5 in the output. Thus, the table shows the pixel mapping pair until pixel value '7'. In matrix form, this is shown below:

1	1	1	6
3	3	3	6
3	5	5	7
5	5	5	7

Histogram Matching

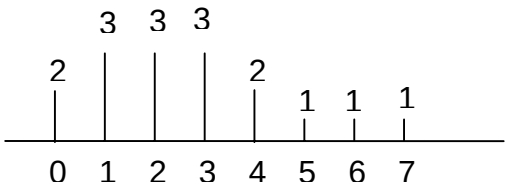
Histogram matching is an extension to the histogram equalization technique. In histogram equalization technique, what we are trying to achieve is that the output histogram should follow the uniform pdf. However, for histogram matching, we want the output histogram to follow according to the histogram we specify. To achieve this, we first histogram equalize the input image, then the pdf of this resulting equalized image will be matched to the pdf of the desired histogram.

Let r and s be the pixel values for the input and equalized image respectively, and let z be the pixel value for the desired histogram. First, the given image would be histogram equalized using the technique discussed previously. However, at this juncture the computation is only up to finding $p(s_k)$ only i.e. the actual pixel value will be done at the end of the process. Next, the cdf of the desired histogram i.e. $p(z_k)$, is computed. Finally, for every $p(s_k)$, the nearest

$p(z_k)$ is sought. The corresponding s_k then is multiplied by $L - 1$ to get the actual output value.

The following example shows the concept of histogram matching technique.

Using the same image matrix as in the previous example, let's suppose the desired histogram is as follows:



From this desired histogram, we come up with the desired cdf i.e. by accumulating the probability of the desired histogram. This is shown in the table below:

r	0	1	2	3	4	5	6	7
$p(r_k)$	0.1875	0.25	0.3125	0	0.0625	0.0625	0	0.125
$p(s_k)$	0.1875	0.4375	0.75	0.75	0.8125	0.875	0.875	1
$p(z_k)$	0.125	0.3125	0.5	0.6875	0.8125	0.875	0.9375	1
s	0	2	3	3	4	5	7	7

For $p(s_0)$ = 0.1875, the nearest value for $p(z_k)$ is 0.125, which is for z_0. Hence, the output pixel corresponding to r = 0 is s = 0. Then $p(s_1)$ = 0.4375, the nearest value for $p(z_k)$ is 0.5, which is for z_2. Hence, the output pixel corresponding to r = 1 is s = 2. Then $p(s_2)$ = 0.75, the nearest value for $p(z_k)$ is 0.6875, which is for z_3. Hence, the output pixel corresponding to r = 2 is s = 3. The process is continued until we've done the last pixel. The output matrix then will be as follows:

0	0	0	4
2	2	2	5
2	3	3	7
3	3	3	7

Notice, that in either case, the resulting histogram is not as expected. This is quite normal since we are dealing with discrete quantities while the theory was developed based on continuous values. Hence, what we have obtained is actually an approximation to the actual or desired values.

Matlab Implementation in Image Processing Problems

Among many of its basic functions, Matlab has come up with a library of dedicated subroutines for image processing m-file commands. These commands are contained in the Image Processing Toolbox. The list of these functions can be viewed by typing help images at the Matlab prompt i.e. >help images while each command has its own help information by typing help followed by the command name.

The first thing to do is to load an image file into the Matlab environment. This is achieved by using the imread command. Suppose we want to load a jpeg image called sample.jpg. This is achieved as follows:

```
>x=imread('sample.jpg');
```

The above call will load and put all the data into variable *x*. Next, to display or show this image onto the screen, the following command can be used:

```
>imshow(x);
```

We can also obtain the size of the image by calling the following function at the prompt.

```
>size(x)
```

If the image is of size 320 by 240 and a color image, then we will get the following response

```
320 240 3
```

The last number '3' indicates that the image consists of 3 bytes for 3 primary colors red, green, and blue (each primary color consists of 1 byte or 8 bits). This image will have 320 rows and 240 columns.

Another useful feature in Matlab is that we can write our own function in Matlab editor and save this function as one of the m-files functions. To do this we need to declare the function at the very beginning of the editor line such as

```
function hstg = histogramequal(image, des_hist)

the rest of the function
```

This function must be saved as histogramequal.m so that a proper call can be made to the newly created function.

For further reading on image processing please refer to:

1. Image processing class notes – DSP PBL Lab website
2. R. Gonzalez and R. Woods, "Digital Image Processing" text book published by Prentice Hall International.

For further reading on Matlab, please refer to Matlab online help.

Your Task

Download the images shown below in Figure 6 from the DSP PBL Lab website. Write your own Matlab to perform the following task. You are **not allowed** to use any function in Image Processing Toolbox except for `imread` and `imshow` functions.

1. Obtain the histogram of the image by plotting the histogram using either the plot or the bar function. Give your comment regarding the contrast of this image.

2. Perform piece-wise contrast stretching technique to improve the contrast of this image. You need to specify your own parameters for s_k and r_k. Show the result you obtain by comparing both images (before and after contrast stretching) as well as their respective histograms.

3. Perform the histogram equalization technique. Write your code as Matlab m-files function that accepts image data as an argument such as `histogramequal(x)` where x is the image data that you obtain through Matlab command `imread`. Show the resulting image as well as the histogram of this equalized image.

4. Finally, you need to perform histogram matching technique. In this experiment, the desired histogram would be the reverse pdf from the original image. In other words, suppose that the original image has more dark pixels then the desired histogram would have more bright pixels. This desired histogram can be achieved using negative function. Once again, you need to put your Matlab code as an m-files function except that this time the function accepts 2 parameters, image data and the desired histogram. Your program should be able to show both the output image as well as the resulting histogram.