LAB 5

Fall 2010, BESE- 13 & 14

Digital Image Processing with MATLAB

Objective

In today's lab we will explore the MATLAB functions for intensity transformation and histogram plotting of an image. Also we would see how we can develop our own code for intensity transformations. By the end of this lab, you should be able to generate image's negative, apply log and power law transformations to an image and plotting image intensity histogram.

Submission Requirements

You are expected to complete the assigned tasks within the lab session and show them to the lab engineer/instructor. Some of these tasks are for practice purposes only while others (marked as '*Exercise'* or '*Question*') have to be answered in the form of a lab report that you need to prepare. Following guidelines will be helpful to you in carrying out the tasks and preparing the lab report.

Guidelines

- In the exercises, when you are asked to display an image, you have to put the image displayed in your project report. You may either save the image as 'jpeg' (File->Save As) or add it to the report or use the 'Print Screen' command on your keyboard to get a snapshot of the displayed image. This point will become clear to you once you actually carry out the assigned tasks**.**

- Name your reports using the following convention:
    - ***Lab#_Rank_YourFullName***

    - '#' replaces the lab number
    - '*Rank'* replaces Maj/Capt/TC/NC/PC
    - '*YourFullName*' replaces your complete name.

- You need to submit the report even if you have demonstrated the exercises to the lab engineer/instructor or shown them the lab report during the lab session.

Tasks for Today

1. Intensity Transformations

Intensity transformation T is a function which is when applied to an intensity r gives a processed or transformed intensity s at that specific pixel.

1.1 Inverse Transformation:

Inverse transformation is also known as **Image Complement**.

Mathematically, it can be defined as:

$s = T(r) = L - r$

where, *r* is the current intensity at a pixel point (x, y) and *s* is the resultant intensity after the transformation function is applied onto *r*. L is the max. no of gray levels, usually $L = 2^8 - 1 = 256 - 1 = 255$.

MATLAB Function to implement inverse transformation:

img_neg = imcomplement(img);

it will compute the complement of the image *img* which can be a binary, grayscale, or RGB image. *img_neg* has the same class and size as *img*.

1.2 Power Law Transformation:

Power transformation is also known as **Gamma Transformation**.

Mathematically, it can be described as follows:

$s = T(r) = cr^{\gamma}$

where, *r* is the current intensity at a pixel coordinate (x, y) and *s* is the processed intensity after the transformation is applied. γ (gamma) specifies the how the intensities are mapped i.e. towards brighter side or darker side. If $\gamma < 1$ then mapping is weighted more towards brighter side. Otherwise, for $\gamma > 1$ more weight would be given to the darker side and so the image will be darker. It $\gamma = 1$ then it would act as a linear function.

c is a constant and is usually 1.

MATLAB Function to implement Power Transformation:

img_pwr = imadjust (img, [low_in high_in], [low_out high_out], g);

it will map the intensities in an image *img* to new intensity value such that intensity values between low_in and high_in will be mapped to the value between low_out and high_out.

Example:

1) img2 = imadjust(img, [0.25 0.5], [0 1]);

   this is useful for highlighting specific intensity band of interest.

2) img3 = imadjust(img, [ ], [ ], 2);

   this compress the higher intensity value towards darker intensities.

1.3 Logarithmic Transformation:

Logarithmic transformations are implemented as follows:

$s = T(r) = c \log(1 + r)$

where, *r* is the current intensity at a pixel point (x, y) and *s* is the resultant intensity after the transformation function c * log(1 + r) is applied onto *r*.

Example:

img2 = c * log(1 + im2double(img));

Exercise 1:

Write a function named ‘transform’ to implement intensity transformations:

INPUTS: 1) Image, 2) Transformation name i.e. ‘inverse’, ‘power’, or ‘log’.

OUTPUTS: Show your results for each of the transformations and give a comparison between log and power transformations with reference to your findings/observations. (DON’T USE BUILT-IN M-FUNCTION)

2. Plotting Graph of Functions

In MATLAB graph of a function (e.g. mathematical functions sinx etc) can be plotted using a *plot* function:

plot (horz, vert, 'color', 'linestyle', 'value', 'marker', 'value')

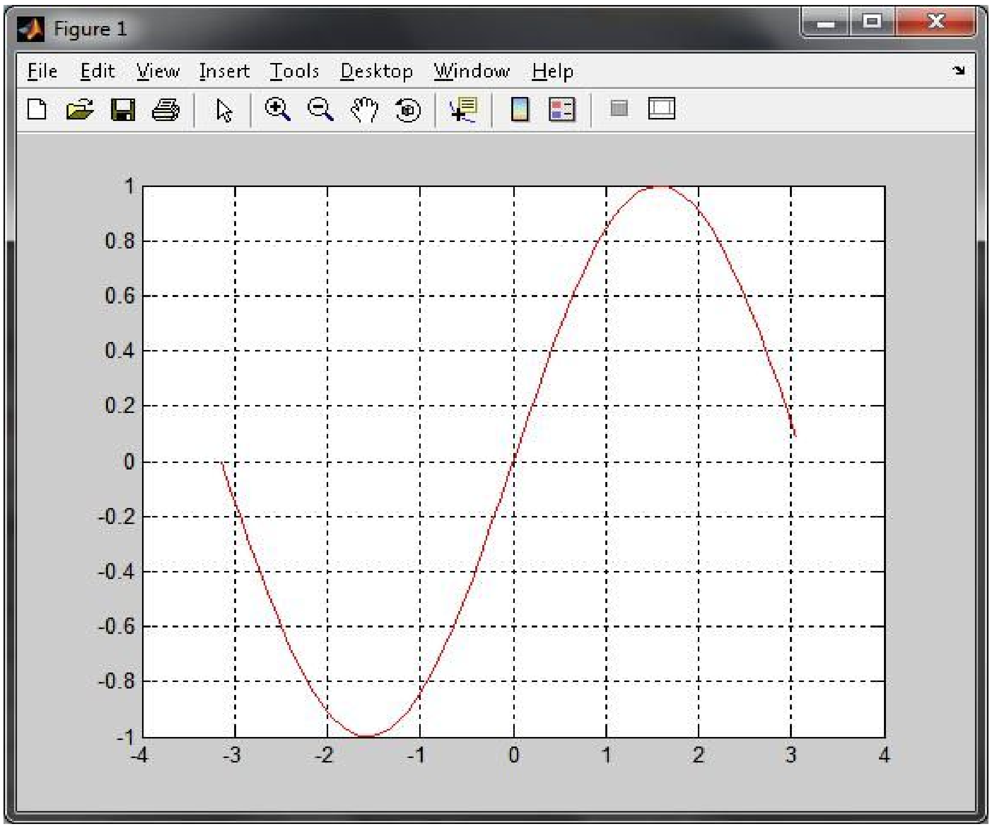
where, *horz* is a set of x values along horizontal axis (i.e. x-axis) and *vert* is set of y values, y = f(x). *color* specifies the color of line plotting the function with possible values 'r', 'g', 'b', 'm' etc. *linestyle* could be '–' (solid), ':' (dotted) etc. And *marker* can be '+' (plus sign), '*' (asterisk), 'x' (cross) etc.

NOTE: to explore all possible values of color, linestyle and marker see Book: 'Digital Image Processing with MATLAB (table3.1, Page# 93).

Example:

```
>> i = -pi: 0.1: pi;
>> plot (i, sin(i), 'r', 'linestyle', '-', 'marker', 'none'),  grid on
```

it would plot the graph for a trigonometric function *sinx*, as shown below:



To adjust range of a grid cell along horizontal and vertical axis:

```
>> set(gca, ‘xtick’, [-pi: pi]);
>> set(gca, ‘ytick’, [-1: 0.2: 1]);
```

To set the minimum and maximum values in horizontal and vertical axes:

```
% -pi to pi specifies x values range while -1 to 1 is a range for y values.
 >> axis([-pi  pi  -1  1]);
```

To display a label string along x-axis and y-axis:

```
>> xlabel(‘x-axis’, ‘fontsize’, 10);  % label horizontal axis with a string ‘x-axis’
                                      % 10 is a fontsize in points
>> ylabel(‘y-axis’, ‘fontsize’, 10);  % label vertical axis with a string ‘y-axis’
```

To set title over the graph of a plotted function:

```
    >> title(‘trigonometric function: sinx’);
```

3. Plotting Image Histograms

Histogram is a graphical chart that represents the occurrence behavior of a specific variable x in a sample space X. In image processing, histogram depicts the frequency of occurrence of a particular intensity value (e.g. gray-level) 'i' in the given image.

Mathematically, histogram of a digital image with gray levels [0, L – 1] is a discrete function $h(r_k) = n_k$, where r_k is the kth gray level and n_k is the number of pixels in the image having gray level r_k. In order to get good understanding towards this, let us consider a matrix M_{4x4} with gray-levels $2^4 = 16$ [i.e. 0 – 15]:

2	5	9	9
7	6	2	2
8	9	7	3
4	10	12	1

$f(r_k)$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
n_k	0	1	3	1	1	1	1	2	1	3	1	0	1	0	0	0

Image Processing Toolbox (IPT) in MATLAB provides a built-in function to draw an image histogram, syntax is as follows:

h = imhist(img);	plots and returns histogram of an image *img*. *imhist* automatically calculate the frequency of each intensity level L and plots its histogram accordingly.
h = imhist(img, b);	Same as above except *b* here specifies the no. of bins used in forming the histogram (if b is not specified as in above case then by default b = 256).

Example:

```
>> imhist(imread('cameraman.tif'));
```

To Manually Compute image/matrix Histogram:

An image histogram can manually be computed simply by counting the no. of times (n_k) a specific gray-level r_k has occurred, defined as follows:

```
for each intensity i at f(x, y) in an image
    increment the counter of intensity i by 1 in an array
```

After the histogram has been computed it can be displayed in following different manners:

bar(x, y, w, …)	plots histogram or graph of a function in form of bar graph. *w* is the width of bars (i.e. 0 – 1). Remaining arguments list is same as *plot* function.
stem(x, y, …)	plots image histogram or graph of a function in form of stem graph. **EXPLORE YOURSELF!**

Exercise 2:

Write a function named 'myhist' to MANUALLY compute image histogram as described in the LAB:

INPUTS: image

OUTPUTS: draw the computed histogram using bar and stem functions and compare your results with the histogram generated by imhist function.

Note:

1) *Give the title 'Bar Graph' to histogram plotted with bar function and 'Stem Graph' for histogram plotted with stem function.*
2) *Set label for x-axis to 'gray-levels' and label y-axis with 'frequency'.*
3) *Line color should 'red' for bar graph and 'blue' for stem graph.*