PRACTICAL 1

IMAGE ENHANCEMENT

TASK 1: DISPLAYING IMAGES AND HISTOGRAMS

Following the last task (Task 10) from the previous practical, we will start by displaying the any given image. Please remember to create a new script for each task in this practical rather than using the command line directly, so you can keep track of your progress and rerun the program after errors.

STEP 1: Download the images `'boat256.jpg'` and `'dome256.jpeg'` and load them as two different variables using imread.

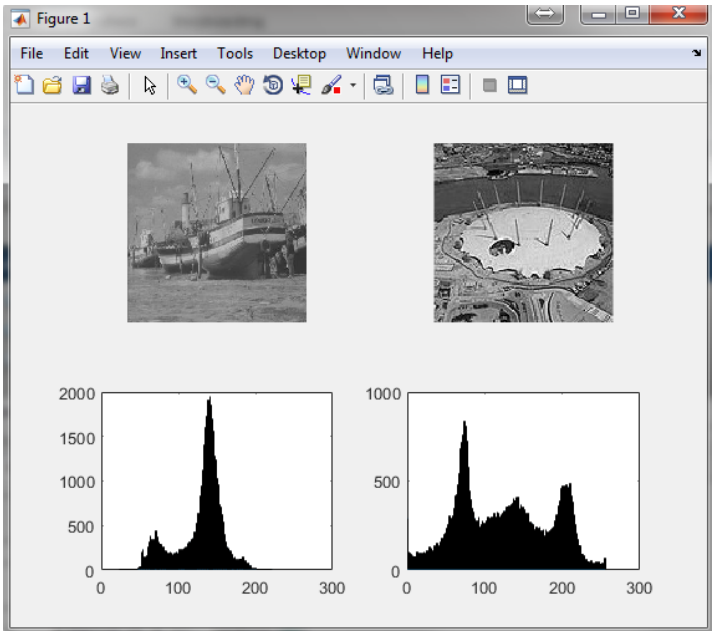
STEP 2: Using `subplot`, create a 4-plot figure (2x2). Display the two images that you just loaded in the first 2 plots using the display command of your choice.

In order to calculate the histogram, Matlab provides several functions (`hist, imhist, histogram`, ...), with slight differences.

STEP 3: Using the function `histogram`, display the corresponding histograms of 256 bins for both pervious images right under the image.

```
histogram(I,'BinLimits',[0 256],'BinWidth',1);
```

The output should look as follows:



Please remember that the two hour duration of a practical class will probably not be enough time to complete all sections. There is an expectation that you will work outside of class as an independent learner.

STEP 4: The histogram function can also be used to give you back the calculated histograms as follows

```
H=histogram(….);
h=H.Values;
```

Extract the array of 256 values of one of the histograms.

STEP 5 (OPTIONAL): Create your own function `myHistogram`, that returns the histogram `h` of the input image `I` as a simple array of 256 values. Compare the output with the output from step 5 to be sure your calculations are correct.

```
function h = myHistogram(I)
```

[HINT: You could use a double nested for to pass by every single pixel]

TASK 2: BRIGHTNESS ENHANCEMENT

Pixel operations are used for brightness and contrast enhancement. In a pixel operation each input pixel value is mapped to an output pixel value using a transfer function.

We can adjust the overall brightness of an image by adding a constant c to each pixel value. The transfer function is

```
output value = input value + c
```

If c > 0 the overall brightness is increased, if c < 0 it is decreased.

Since in image processing, pixel values are in the range 0 to 255 we must ensure that output values are not less than 0 or greater than 255. Our transfer function therefore becomes

```
if input value < -c
    output value = 0
else if input value > 255 – c
    output value = 255
else
    output value = input value + c
```

STEP 1: Create a new function:

```
function Lut = brightnessLUT(c)
```

that will create a Look-up table `Lut` containing the transfer function for increasing/decreasing the brightness of a figure in a given amount `c`. `Lut` is an array of 256 elements indexed from 1 to 256 (in Matlab index the first element is index 1 unlike in java or c++). Thus, the index of an element represents the **input pixel value +1** and the value stored in the element is the corresponding output value.

To be sure the look-up table is of the same type of data that the pixels (shorts) , use the instruction:

```
Lut=uint8(Lut);
```

as last line in your function

STEP 2: Create a new figure and using `subplot`, divide it in 2x3 plots. Display the `'boat256.jpg'` image in the first position and its histogram right under. Display the `Lut` using the command `plot` to the right of the image boat (position 2) for a value `c` of your choice (e.g. c=50)

STEP 3: Now you should be able to run the provided function:

```
function Iout = enhanceBrightness(Iin,c)
```

which calls the function you just created in step 1. Open the function and analyse how it works. The command `inlut` allows us to perform a pixel operation where it applies the `Lut` to the input image. The parameters passed into the method are the original image and the LUT. The enhanced image is returned.

STEP 4: Using `enhanceBrightness`, display the enhance image in position 4 and its corresponding enhance histogram in position 6. Try varying the value of c (positives and negatives) and observed the effect.

TASK 3: CONTRAST ENHANCEMENT USING LINEAR STRETCHING

STEP 1: Write the function to enhance the contrast of an image using linear stretching:

```
function Iout = enhanceContrastLS(Iin,m,c)

function Lut = contrast_LS_LUT(m,c)
```

This function can be easily created by modifying the functions from the previous task. The method should create a LUT, use the pixelop method to perform the pixel operation, and return the enhanced buffered image.

STEP 2: Apply the previous functions to the boat image and display the results, histograms and transfer function in a new figure. Choose the values of m and c manually so it optimises the histograms as much as possible.

STEP 3: Using the magnifying lens in the toolbar, make zoom in the histograms. Do you notice something unusual in the enhance boat's histogram? What can this be due to?

STEP 4: Create a modified version of the contrast enhancement function call:

```
function [Iout, Lut] = enhanceContrastALS(Iin)
```

where the values of m and c are automatically generated internally without the user input. To do so, the initial `(i1)` and end point `(i2)` of the histogram should be detected and used to calculated m and c.

[HINT: To avoid noise in the histogram, you can consider the histogram starts and ends the first and last time that the bin values goes above 10 pixels. The function `find()` can be of use.]

TASK 4: CONTRAST ENHANCEMENT USING POWER LAW

For power law the transfer function is

$$\text{output value} = (\text{short})[(\text{input value})^{\gamma} / 255^{\gamma-1}]$$

STEP 1: Create the required functions in order to enhance the contrast of an image using linear stretching:

```
function Iout = enhanceContrastPL(Iin,gamma)
```

```
function Lut = contrast_PL_LUT(gamma)
```

STEP 2: Apply the previous functions to the given images and display the results, histograms and transfer function in a new figure. Vary the value for gamma. Try 0.5, 2 and 3.

The strategy to implement the pixel operation as a LUT first which is then applied to every pixel is an excellent option that will work in any programming language, even low level ones such as Java or C, and even in embedded implementation. In Matlab, however, due to being high level and designed for matrix operations, the previous enahncemenst in Task 2,3 and 4 can be simply written as a one line instruction. For instance, brightness enhancement can be simply written as:

```
Iout = Iin+c;
```

obtaining exactly the same results. This is due to Matlab detecting overflow for variables defined as `uint8` and capping their values to 0 and 255 automatically (without requiring the if conditions). It also detects that we are trying to add a number to a matrix and applies the sum to every single pixel for you.

STEP 3: Knowing this, replace the calls to the functions `enhanceContrastLS and enhanceContrastPL` by a single instructions that produce exactly the same effect.

[HINT: you many need to convert between uint8 and doubles to be sure the calculation is correct]

TASK 5: CONTRAST ENHANCEMENT USING HISTOGRAM EQUALISATION

STEP 1: Matlab has its own function to perform histogram equalisation, call `histeq`(). Using this function, display the result of for boat given images (boat and dome) as well as the calculated transfer function and resulting histogram

STEP 2: Implement your own histogram equalisation functions:

```
function Iout = enhanceContrastHE(Iin)
```

```
function Lut = contrast_HE_LUT(Iin)
```

so that the transfer function is

```
output value = max {0,(short)[256 * CH(input value) / (number of pixels in image)] - 1}
```

where CH is the cumulative frequency function. Compare the results with Step 1. Are the identicals? Check the documentation and the transfer function to figure it out why.

[HINT: The function `cumsum`() in matlab can be useful for this task. Remember that the indexs in Matlab starts in 1 rather than 0, so you may have to modify **very** slightly the formula above].