IMAGE PROCESSING (ERR041)

STUDIO EXERCISES

Lecture 2, Image Enhancement, Part I

On the windows system to use m-files written for the course you must in windows after starting MATLAB type

```
addpath \\mfil.me.chalmers.se\course-err041
```

If you are using the linux operating system after starting MATLAB you should type

```
addpath /chalmers/groups/course-err041
```

EX 1. Histograms and Thresholding

In MATLAB type **load('acc_scan.mat')**, then **acc=X;**, display with **imshow(acc, [ ])**. This is a black and white or binary image which has been affected by problems when it was scanned so that one side is darker than the other. Use the MATLAB command **imhist(acc)** to examine the image histogram of this image. We can recover the original image by image thresholding, so that all pixels below a certain value are converted to 0 and all pixels above this value become 1. This can be done using the matlab command **im2bw**, and selecting the correct threshold limit (Type **help im2bw** to find out about this command).

EX 2. Image Histograms and Contrast Stretching

Inside MATLAB type **imadjdemo**. Start with the *circuit* image. Experiment with the +/-Brightness and +/- Contrast buttons. **Question** How do these effect the transfer function? Do you understand how changing the transformation function changes the image and its histogram?

Now load the 'Pout' image. Adjust the transfer function manually by clicking on the top and bottom yellow circles. Experiment with different transform functions. **Questions** Which transform function produces the best image? What does the image histogram look like in this case? Try other images.

EX 3. Histogram Equalisation

In the same demo as Ex 2, apply the 'Histogram Equalisation' function by clicking on the 'Operations' box. Try histogram equalisation for all of the images. **Question** Which images are improved the most and why? Which image is improved least and why?

EX 4. Implementing Histogram equalisation (class+homework)

Write a matlab m-file to implement histogram equalisation using standard MATLAB image processing toolbox commands. Load the integer format 'Pout' image **pout = imread('pout.tif')**. Display with **figure; imshow(pout,[])**. This image has 8bits/pixel or 256 levels (gray level 0 to 255). Create the 256 element vector **trans** which accomplishes histogram equalisation on this image. The n'th element of this vector should represent the output gray level (0-255) that input gray level n-1 maps to.

Procedure. Create a vector containing the input image histogram with command **imhist** (type 'help imhist' for usage). Normalise this histogram by the number of pixels in the image (using **numel**) and from this form the transform function. Useful matlab commands are **cumsum, round** etc. Finally the transformed image can be made via **eqpout = trans(pout+1)** and then displayed using **figure; imshow(eqpout,[])**.

Bug Warning. Note commands **uint8, double** to convert to integer and double precision floating point respectively. The first format is needed for matrix indices and the second to do arithmetic operations like addition etc on images. Note also the difference between vector indices which must be none zero (1 -256) and the gray level they represent (0 - 255).

EX 5. Implementing Histogram specification (Homework - Optional)

Load the coin image **coin = imread('quarter.tif')**. As we saw in exercise 3, histogram equalisation does a bad job of enhancing this image. Instead we can use image histogram specification as described in the lecture notes. Write a matlab m-file to accomplish this.

The program should first display the image histogram using **imhist** without outputs. Then prompt **coinlim = input('Maximum gray level (0-255) associated with coin')**. Then prompt **coinrang = input('Maximum gray level (0-255) for coin in output image')**. The program should find a grey level transfer function that takes pixels in the range (0 - coinlim) in the input image and distributes then aprox uniformly in the range (0 - coinrang) in the output, and likewise takes any pixels in the brightness range (coinlim, 255) in the input and distributes them approx uniformly over the range (coinrang - 255).

Procedure: First calculate the target normalised histogram (a step function) and also find the input image normalised histogram (using **imhist**). Calculate the transfer function by comparing the cumulative sums of these histograms as described in the lecture notes. Useful commands might be **abs, min, find** and **for...end** loops, apply the transformation, finally display transformed image and display its histogram.

Bug warning: As for Ex 4.