Pixel Operations[1]

Dr. François Pitié

Electronic and Electrical Engineering Dept.

pitief@tcd.ie www.sigmedia.tv

THIS LABORATORY SESSION COUNTS FOR 3% OF YOUR OVERALL 4C8 MARK

1 Lab Information

This is the first lab assignment and accounts for 3% of the total 4C8 mark out of a total Continuous Assessment Mark of 25%. In this lab, you will be required to compile a report that answers specific instructions included in the report. There are other instructions that do not require answers in the report but are necessary to complete the lab.

IN THIS LABORATORY HANDOUT SPECIFIC INSTRUCTIONS ARE INDICATED WITH THIS SYMBOL : □ .

ENUMERATED INSTRUCTIONS REQUIRE ANSWERS IN YOUR SUBMITTED REPORT.

1.1 Submission Requirements

Labs Reports should be typed up and submitted electronically using the **PDF file format** via the module's blackboard page by the specified deadline. Remember to put your name and student number on the top of your reports.

Penalties for Late Submission

- less than 15 minutes past the deadline - no penalty.
- greater than 15 minutes and less than 2 hours past the deadline - loss of 25% of the final mark.
- greater than 2 hours past the deadline - loss of 50% of the final mark.

[1] This lab was originally written by Prof. Anil Kokaram and ammended by Dr. David Corrigan.

Plagiarism Policy

Any submitted code or answers that can be seen to be plagiarised from other sources will result in 0 marks being awarded for that section of the lab and may result in you being awarded 0 marks for the entire assignment. Anti-plagiarism software will be used on submitted materials.

2 The EESERVER Resources (at EEE, Trinity College)

Test images can be found in `y:/image_stills` `y:/sequences`. Raw video for restoration can be found in `y:/sequences/dirty`. All data without `.bmp; .jpg; .mpg etc` extensions are stored in RAW BINARY FORMAT. Grey scale images are 8 bits per pixel (each pixel is stored as **Unsigned Chars**). See the read me files in each directory to work out how big the images are. Ask a demonstrator where the data is located in case you have a different set up.

3 Images and Matlab

You can read raw data into Matlab using the `fread3` function. See the help file for details. You can write data back out using the `fwrite3` function. These functions call the basic matlab `fopen, fseek` etc functions. You should remember this in case you want to write your own function with different behaviour. Note that `fread3 and fwrite3` are not standard Matlab files and are only available in the installation on the EESERVER.

Remember that images are just 2-d arrays of numbers. For 8bit grey scale, the image pixels take on values from 0 to 255.

You can display images in matlab easily. Say that the image is in the matrix K (lets assume it is of size 256×256. Then

» `figure(1);image(K);colormap(gray(256));axis off;axis image;`

Will display the image in figure 1 as an 8 bit grey scale value.

Because an image is simply a matrix we can access any subblock of it using Matlab notation. Thus

» `K(20:40,20:27)`

is a block of size 21 rows and 8 columns starting from the pixel at the 20th row of the 20th column of K. Try it.

Remember that in Matlab when you do things with `for` loops, it tends to be very slow. Use Matrix manipulation wherever possible.

4 Basic image loading and display

□ Create an .m script which loads a raw image and then assigns that image to the matrix `pic`. Since the image is just raw, 8 bits per pel in a single file with no header, you have to know beforehand what size it is. That is a pain in general since pictures can be any size. This is the reason why most image formats in common use e.g. .bmp, .tif, .jpeg , have header information which included the size of the image. Here is an example that you can use for loading the `lenna.256` image. Note that in Matlab the % symbol is used to indicate a comment used to explain the code.

```
%clear the work space and close all open figure windows
clear; clear all; close all;

%declare the size of your pictures
hres = 256;  vres = 256;

%define the picture name
name = ['lenna.256'];
%ask your demonstrator exactly what filename
%(including directory addressing) to use here

%open the file
fin = fopen(name,'rb');

%read in the picture data
% note the transpose operation ()' is used since the picture is stored
% in row scan order from top to bottom, but Matlab reads data into
% columns
pic = double(fread(fin,[256 256])');

%double() converts the image values to floating point
%double images for mathemtical use

% display the image in a figure window
figure(1);image(pic);colormap(gray(256));shg;

%image(pic) actually shows the picture in figure(1)
%colormay gray(256) makes sure the 8 bit picture is
%displayed such that 0=black and 255=white
%shg makes the figure come to the top of your windows desktop,
% and hence shows it to you. Its short for "show graphics"
```

☐ Add a line in your .m file which adds 128 to the image using `newpic = pic+128;`. Now display the resulting picture in Figure window 2.

1. Explain what you see compared to `pic`. Why has much of the picture turned white?

☐ Add a line in your .m file which subtracts 128 from the image using `newpic = pic-128;`. Now display the resulting picture in Figure window 3.

2. Explain what you see compared to `pic`. Why has much of the picture turned black?

4.1 Matlab and Data Types

Note, when loading images, by default Matlab assigns the data type to the matrix that is native to the file data type. As most images represent intensities with unsigned 8 bit integers (uint8) that means that the matrix into which the image is loaded uses the uint8 datatype. It is convenient to cast the datatype as double floating point precision since there are some operations that are not implemented for uint8. In the previous code snippet, this is achieved by the use of the `double` function.

5 Histograms

Histograms summarise the grey scale distribution of an image. It is a useful tool for informing subsequent analysis. The histogram plots on the x-axis values of greyscale, and for each value it plots on the y axis the number of pixels in the image having that value. Gradation in levels over which the frequency of pixels are counted are called *bins*. Thus `hist(pic(:),[0:255]);` in Matlab generates a histogram of the image `pic` using 256 bins starting at 0 and moving up to 255 in increments of 1 grey level.

☐ Write a Matlab script that loads an image from a user defined filename, and generate the 2D histogram of the image. Use the picture `sigmedia06907.tif`, and assign it to `pic`. Note that the inbuilt Matlab function `imread` can read in all common image formats like *bmp, jpeg, tiff etc.*. Also note that `pic` now has 3 color planes, as Matlab assigns the Red pixels to `pic(:,:,1)`. Green and Blue get assigned to the second and third planes respectively. We are only working on gray scale images here so the .m file will need to convert this RGB image to grayscale using the Matlab function `rgb2gray`. Use `help rgb2gray` to see how to use it. Your pictures should look something like those shown in Figure 1.

1. Type `size(pic)` in Matlab and explain the output.

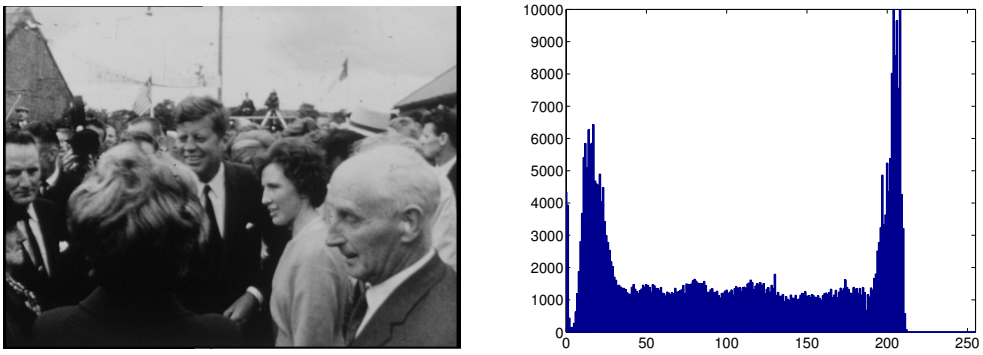

Figure 1: Left: Original jfk image, Right Histogram of the image on the left using 256 bins from 0 to 255

5.1 Histograms of colour planes

☐ Change your script above to load the `pool.01.bmp` image. Use `imread` and your file should also work out the size of the image with `[vres,hres]=size(picture)`.

☐ Plot the histograms of the R, G, B components of the image in three different figure windows. Recall that `pic(:,:,1)` is the first colour plane, `pic(:,:,2)` is the second.

1. Put images of the 3 histograms into your report. Explain how the table region of the image manifests in the three histogram spaces.

6 Thresholding

A basic point wise operation is thresholding. Open the greyscale image in the file `nuclei.02.png`. Suppose as part of a recognition problem you require to identify the nuclei in the image. As you can see, the nuclei is brighter than the background. The operation that tests this proposition is called *thresholding*. It is possible then to generate a *segmentation* of the image into two regions (nuclei and not nuclei) with the following operation.

$$l(h,k) = \begin{cases} 0 & \text{For } I(h,k) \leq t_1 \\ 1 & \text{For } t_1 < I(h,k) < t_2 \\ 0 & \text{Otherwise} \end{cases}$$

where the input image is $I(h,k)$ and the output image is $l(h,k)$. In this example, when the pixel falls inside t_1 and t_2, $l(\cdot) = 1$, and it is zero otherwise. In Matlab, thresholding

like this is simple `mask = (pic>=t);` returns a binary image `mask` which is 1 everywhere the condition is satisfied (i.e. in this case that a pixel has a value greater than or equal to `t`), and it is zero otherwise.

☐ Write a script to read in the image and using your own judgement choose a suitable value of t_1 and t_2 for thresholding the intensity of the image so that `mask` contains pixels that are 1 where the nuclei exist. Satisfy yourself that you have achieved an acceptable segmentation of the image.

☐ By examining the histograms of colour planes that you estimated in Section 5.1, write a script that segments the table in the image `pool.02.bmp` by applying thresholds to each of the 3 colour channels.

2. Include in your report, an image showing the segmentation that you achieve. Describe how well your *segmentation mask* matches the actual limits of the table. Specifically mention where the algorithm works and where it does not. Write down the values of the thresholds that you use for each channel. Explain why applying thresholds on all 3 colour channels improves segmentation compared to performing the segmentation on the green channel alone.