Pixel Operations

Dr. François Pitié
Electronic and Electrical Engineering Dept.
pitief@tcd.ie www.sigmedia.tv

This lab accounts for 5% of the final 4C8 mark.

Report Instructions: You are required to compile a report that answers specific instructions included in the report. Include relevant code and output figures in your report.

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.

You must have read and understood the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at: https://www.tcd.ie/calendar.

Images and Matlab

You can read images in Matlab using the `imread` function. See the help file for details:

```
>> help imread
```

You can write data back out using the `imwrite` function. Remember that images are just 2-d arrays of numbers. For 8bit grey scale, the image pixels take on values from 0 to 255.

```
>> I = imread('kodim24bw.png');
```

You can display images in matlab easily. Say that the image is in the matrix `I` and is of size 256×256. Then

```
>> figure(1); image(I); colormap(gray(256)); 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 `I(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 I. Try it.

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. Most native Matlab functions can take either uint8 or double as image input type. However, note that when working with doubles, Matlab assumes that the data range is [0,1] and not [0, 255]. you must therefore rescale the values into that range. For instance:

```
>> I = double(imread('kodim24bw.png'))/255; figure(2);
   image(I); axis image;
```

Also, note that you don't need to specify the colormap when using doubles.

1 Basic image loading and display

Q 1.1 *Write a Matlab script that loads* `kodim07.png`*, and generate the grayscale histogram of the image. Use the picture* `kodim07.png`*, 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 colour 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.*

Q 1.2 *Add a line in your .m file which adds 128 to all the pixels' brightness value in* `pic` *(we assume that you are working in uint8, with a 0-255 range) After displaying the resulting picture in Figure window 3, explain what you see compared to* `pic`*. Why has much of the picture turned white?*

Q 1.3 *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 4. Explain what you see compared to* `pic`*. Why has much of the picture turned black?*

2 Histograms

Histograms summarise the pixel values distribution of an image. It is a useful tool for informing subsequent analysis. The histogram plots on the x-axis values

of greysacales or colour channels, 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 `histogram(pic,0:256);` 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 (use `help histogram`).

Note that the `histogram` command can operate on RGB images, but will implicitly flatten any input multi-dimensional array into a single vector.

Q 2.1 *Load the* `tennis.png` *image and 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, etc.*

Q 2.2 *Convert the tennis court image to the YCbYr colour space. YCbCr is an equivalent colour space to the YUV colour space. Use the* `rgb2ycbcr` *command and plot the histograms of the Y, Cb, Cr components.*

Q 2.3 *Put images of the 6 histograms into your report. Explain how the tennis court region of the image manifests in these histogram spaces.*

3 Segmentation

In Matlab, thresholding can be simply done with basic binary comparisons. For instance `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 `t`), and it is zero otherwise.

Q 3.1 *Choose suitable thresholding values for each of the colour channels of the tennis court image and combine them to create a binary mask (*`mask`*) of the tennis court. Make sure that pixels that are 1 correspond to the court itself.*

Include in your report the code and an image showing the segmentation that you achieve. Describe how well your segmentation mask *matches the actual limits of the court. 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.*

4 Colour Balancing

In photography, the colour balance is the global adjustment of the intensities of the red, green, and blue primary colours so as to render neutral colours like white

or grey correctly. Colour balance can change with the scene illumination.

In the example below, we will be looking at balancing the colours between two images A (`kodim23a.png`) and B (`kodim23b.png`). The colour correction can be modelled as a simple rescaling of the three R, G, B values as follows:

$$\begin{bmatrix} R_A \\ G_A \\ B_A \end{bmatrix} = \begin{bmatrix} 255/R_w & 0 & 0 \\ 0 & 255/G_w & 0 \\ 0 & 0 & 255/B_w \end{bmatrix} \begin{bmatrix} R_B \\ G_B \\ B_B \end{bmatrix}, \tag{1}$$

where R_w, G_w, B_w define the values of a white point in B.

Q 4.1 *Load* `kodim23a.png` *(A) and* `kodim23b.png` *(B). For each colour channel R, G, B, compute the histograms of the corresponding channel values for both pictures (using* `histcounts`*). Then compute the cumulative histograms using* `cumsum`*. For each channel, plot both resulting cumulative histograms for A and B into one single plot (eg.* `plot(0:255, CDFRedA, 0:255, CDFRedB)`*). Include the three figures (one for each channel) in your report.*

Q 4.2 *For each colour channel, use the cumulative histogram graphs to read where the channel value 150 in A gets mapped to in B. You can use the data tip tool in the Matlab figure to identify the matching values.*

Q 4.3 *Using these correspondences estimate the overall values for R_w, G_w, B_w. You can apply Eq (1) to check if the transform can indeed map B back to something resembling A.*