7060 Image Sensors & Processing

Tutorial 2: More MATLAB (optional, informal lab session)

D.Gibbins

Location: CAT Suite 15 (IW.B15)

Summary

The following is an informal list of things to look at during the week 2 informal lab session. The intention here is to give you some additional experience with manipulating images in MATLAB and to reinforce some of the concepts presented in lectures.

The tutorial is optional and informal. Please only attend if you feel you would like to explore MATLAB further before for first assignment.

This is not a formal tutorial session so feel free to ask questions and explore the MATLAB processing functions which interest you over the next 1-2 hours.

The following is a rough guide of things I'd like you to try out during the session. It is up to you how much of this you do. You can also finish looking at material from tutorial 1 if you wish.

Background – A quick summary of the functions we will be using

MATLAB contains a Toolbox specifically designed for working with imagery and provides numerous image-processing functions that we can experiment with today. This toolbox includes the following functions:

imread() - read an image from file

rgb2hsv(),rgb2ycbcr(),rgb2ind() – convert RGB image to HSV,YCbCr(YUV) and an indexed image

hsv2rgb(),ycbcr2rgb() – convert image back to RGB from HSV or YCbCr

demosaic() – convert a Bayer image back to a normal RGB image

imhist() – create a histogram of an image's intensity values

histeq() – histogram equalisation

adapthiseq() – local adaptive histogram equalisation

imadjust() – adjust and image to improve contrast (roughly equivalent to the percentile normalisation method shown in class)

im2double() – convert a uint8 image to an array of doubles in the range 0..1.

A more complete list can be found by typing 'help images' into MATLAB.

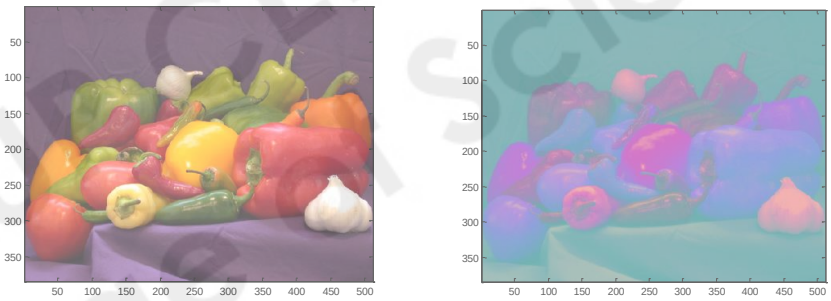
Use the help command in MATLAB to examine some of the documentation for the functions shown above (eg. 'help imhist' or 'doc imhist').

Exercise 1 – Colourspaces

The concept of colourspaces was introduced in week 1. MATLAB supports several of the colorspaces described in class including HSV (hue,saturation,value) and YCbCr (luminance, blue-chroma, red-chroma).

- Load an example colour test image into MATLAB eg. I=imread('peppers.png');
- Convert the image to and from YCbCr and HSV colour spaces and examine the individual channels. eg. Ihsv=rgb2hsv(I); I2=hsv2rgb(Ihsv);

  Note: 1. The YCbCr channels are all defined in the range 0..1 in MATLAB despite that fact that strictly speaking Cb and Cr should contain both positive and negative values. HereMATLAB adds an offset of around 0.5 to the values tokeep them in the range 0..1. If you try and display the converted imagery(eg. using image() or imagesc()) you will find that the colours will not display correctly. This is because imshow(), imagesc() and image() always assume any NxMx3 arrays contain RGB values.



The peppers image show before and after conversion to YCbCr. As you can see matlab does not display the colors correctly (because it doesn't know how to display non RGB imagery).

- Examine the differences between the channels in the original image and those that have been converted to and then from YCbCr and HSV. For example if I is the original, Ihsv is the HSV conversion and Ireconv is the image Ihsv images converted back to RGB then `imagesc(double(I(:,:,1))/255 - Irecon(:,:,1))` would display the differences in the red channel caused during the conversion process *(note: the "double(…)/255" used here is because the original image I is 8-bit0..255 and the MATLAB converted image Irecon is type double in the range 0..1)*. Q: What do you think would happen if MATLAB represented the HSV and YCbCr values as uint8 and not doubles?
- Take your HSV and YCbCr versions of the imagery and rescale one or more of the channels slightly (say by a factor of 0.7 or 1.2). Convert the images back to RGB and display them. Q: What effect did this have on the reconstructed RGB image? (recall the colour manipulation examples shown in the second half of the week 2 lectures)
  - What happens if you scale the H,S, and V channels in HSV space by 50%?
  - What happens if you scale the Y channel in (YCbCr) space by 50%?

Exercise 2 – Demosaicing

As noted in week 1, a typical CCD sensor in a colour camera sees the world through a repeating pattern of Bayer pattern filters. This requires us to process the image inside the camera to reconstruct the 'full' colour image.

- MATLAB includes its own demosaicing function for Bayer images. Run the following MATLAB example:
    - I = imread('mandi.tif'); J = demosaic(I,'bggr');
- Display both of the above images and examine them closely using the zoom/magnifiying glass on the plot window). Compare the images I and J.
- ***OPTIONAL*** *- Try creating your own Bayer image from a full colour image and passing it through the demosaic() function.*

Exercise 3 – Contrast Enhancement

Contrast enhancement was the focus of the image processing work to be covered in week 2. In those notes we will at several techniques. Normalisation, Gamma correction, histogram equalisation, etc. MATLAB supports several of these operations, which we will examine here. Note also that the function imhist() can be used to display the histogram of an image which will be useful here.

- Read in a monochrome image such as 'pout.tif'. Use im2double() to convert it to an array of doubles so we can work with it Find the max and min values (using max() and min()) and see if you can figure out how to rescale the image data such that min=0 and max=1 after rescaling. Display the result. Q:Did it work?
- Try using histeq() to contrast enhance the same image. Use the function imhist() to display the histogram of the image before and after normalisation. Q: What did the results look like? Did it improve the contrast of the image?
- The function imadjust() can be used to perform a form of simple percentile normalisation. Again examine the result. Q: Did it improve contrast, and if so was this better, worse or just different to the result from histeq()
- Read in a color image (eg. 'peppers.png')
- Try and equalise each channel separately using histeq() separately and display the results (remember the (:,:,1) etc indexing used earlier to reference individual channels). Q: Did histeq() improve the contrast? Did anything else change about the image?
- Convert the original image to HSV and histogram equalise the V channel and then convert back. Compare the result(s) to what you had earlier when you balanced R,G and B separately. What changed? Was it better/different than using histeq() on each channel?

  (time permitting: try doing much the same in YCbCr space on the Y channel)
- Experiment with adapthhisteq() in a greyscale image such as I=imread('pout.tif');