Computational Vision - Lab 3

Hamid Dehghani,
School of Computer Science,
University of Birmingham, U.K.
B15 2TT
2nd November 2018

Instructions

- Start up a web browser and bring up the following URL, http://www.cs.bham.ac.uk/~dehghanh/vision_files/lab/lab4/

- Download the .m files and the data files (.jpg) for Lab 4 and put them in a directory. You may want to group all the Matlab files you have collected in a single directory and use the ***File..., Setpath***... menu tabs to add that directory to your Matlab path. Alternatively, use right click "Add to path" on the Matlab folder's window.

- You can load in a .jpg file to Matlab using

  cluttera1 = imread('cluttera1.jpg');

- If you type ***size(cluttera1)*** you will see that the file is a three dimensional array. To convert it to a grey level image you can average over the pixel values in the third dimension. To do this type

  clutter_grey=mean(cluttera1,3);

  Or more appropriately, if the option is available

  clutter_grey=rgb2gray(cluttera1);

  What is the difference between these approaches?

  Also, you need to change the array to double precision numbers, e.g : **clutter_grey=double(clutter_grey);**

- Display clutter_grey to see what it looks like. You can use "imshow(clutter_grey,[])". The empty brackets in the second term ensure that the intensity of the image will be normalised, for good visualisation. If you want to save a figure, you can work with the following lines of code: **f1 = figure;** (this opens a new figure and names its handle "f1")
  imshow(clutter_grey); (display something in the figure, in this case the greyscale image)
  saveas(f1,'grey_img','png'); (we save the "f1" figure, in a file named "grey_img", as a png file)
  If the figure you want to save is already open by a third-party function, you can use: **f1 = gcf;** which stores the current figure handle as "f1".
- Now apply your favorite smoothing and edge detection filter(s) to the image. You will get an array (e.g."img"), which is an un-thresholded edge image.

- Now use the Hough transform implementation given to you. You can type

  A = myhough(img(1:500,1:500)>45,400,'fast'); to start with.

  Here **img*(1:500,1:500)>45*** crops and thresholds your edge image, the ***250*** input is the threshold for Hough image, i.e. how many votes you want to consider

  Note that even this 'fast' version of the transform is inefficient and takes tens of seconds to run. The in-built Matlab version is much faster. So there is a much more efficient implementation than mine. Try varying the threshold for Hough votes, and/or edge image. How does changing the threshold affect the results?

- Download some other images from the data set for the assignment, and repeat your experiments.

- The issue of suppressing local non-maxima in the Hough space was raised in the lecture. Sketch a routine for doing this. The Hough transform didn't localise the beginning and end of each line in the image space. How could you achieve this?

- Remember to submit you short write-up by next Friday in Lecture.