Computational Vision - Lab 3

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

16th October 2015

Instructions

- Log on to the machine and open an xterm.
- Type '***matlab &***' to start the programming environment.
- Start up a web browser and bring up the following URL, http://www.cs.bham.ac.uk/~dehghanh/vision_files/lab/lab3/
- Download the .m files and the data files (.gif) for Lab 3 and put them in the lab 3 directory.
- In Matlab load up the shakey image

  shakey = read_image('','shakey.150.gif');

  You should also load up some noise and edge filters, type ***load filters***.
- Using the built-in procedure ***conv2*** convolve the image with the 3x3 Gaussian filter, and then the 5x5 filter. Can you see any difference between them? Try applying an edge filter to each and thresholding. Can you see the effect now in comparison with applying the edge filter to the image directly?
- Using the function ***N(m,s,-3:1:3)*** you can create a discrete sample from a Gaussian (Normal) density. You need to specify the mean m (keep it at 0, think about why) and the standard deviation s. The last term simply uses the code to create a vector in Matlab. So you can create larger and smaller vectors by altering the step size (the number between the two colons). So now try creating a 10x10 Gaussian mask. To do this you will need to use matrix multiplication in the right way. Try some initial exploratory experiments with this, what happens as you increase the size of the mask? What happens as you increase the size of s? Make detailed notes as you proceed about what you did and what you observed. Why do you see what you see?
- Now compare the speed of applying two large 1d Gaussian filters in sequence, with applying a single equivalent 2d Gaussian filter that results from their multiplication. To test the CPU

time used you can use a function called ***cputime***. Can you detect differences in the CPU times as the mask sizes increase? You should check that the results are the same by examining areas of the image matrix in detail. Are there any effects due to small floating point errors?

- Now apply gradient operators (e.g. ***difference_gradient_filterX*** or simply the ***gradient*** function in Matlab — this has two output matrices Gx and Gy) or an edge filter such as the Sobel operators to the blurred images. What happens to the edges in the heavily blurred case? Why?
- Apply the operator ***first_order_gaussian_filter_1d_length5***. Apply it to the shakey image, and then convolve the shakey image with its transpose. Why can't you sequence the two operators in the way you did with the Gaussian itself? Now you have two edge images, one estimates Gx and the other Gy. To find the magnitude of the gradient G you can use the magnitude operator you wrote last time. You will need to chop the matrices a little. How should they match up? Does this matter? Compare the edge image you get here to that produced by the Sobel operator. Can you say which one is better?
- If you like you can now try creating first order Gaussian filters of different sizes. You can create an approximation to the gradient of any vector or matrix by using the gradient function in Matlab. If you create edge images with operators derived from larger Gaussian masks what happens? Why?
- Now try applying the Laplacian operator to the Shakey image. You will need to take the absolute value. Think about the result. Why does it produce a poor result compared to the other operators?
- HARD: I mentioned the Laplacian of the Gaussian in the lecture. How could you combine the idea of the Laplacian operator with the idea of Gaussian smoothing? Try out your ideas.

- Remember to submit you short write-up by next Friday 9 am.