Exercise #1 – Familiarity with Linux OpenCV

DUE: 1/26/2013 on Blackboard (before Midnight)

Please thoroughly read Computer Vision: Algorithms and Applications chapter 1 and Learning OpenCV chapter 1. Also, for this lab, please read Explore video analytics in the cloud and download example code for this lab.

Goals for this lab include basic familiarity with OpenCV, familiarity with building OpenCV applications in Linux, reading OpenCV documentation and experience with the API compared to ground up implementation of convolution-based image transformation.

Exercise #1 Requirements:

1) [10 points] Read and summarize the main points of Explore video analytics in the cloud in a paragraph or two.

2) [20 points] Test your VB-Linux and OpenCV installation by downloading the example code and example images from the Explore video analytics in the cloud paper and build the code in capture-transformer. If you have any errors with the build, double check your OpenCV installation. Run the Sobel transform (sobel) on Trees.jpg found in example images and provide the image in your report. Read the OpenCV online tutorial for Sobel and provide your best simple English description of how the Sobel tranform works and what it does.

3) [30 points] To better understand the concept of convolution and the application of a pixel level transform, now download the sharpen-psf example code. Build it and run it and make sure your read and understand it. Now, make modifications if needed to run sharpen on Trees.jpg and use GIMP to save Trees as a PPM (Portable Pixmap) so the code can load the image file (like the Cactus-120kpixel.ppm). Provide the sharpened Trees.ppm image in your report and describe any code modifications you made (if needed). Why does the PSF (Point Spread Function) provide edge sharpening? (You may want to refer to the Engineer's DSP Handbook, Chapter 24). Read the sharpen_grid.c code and run it and describe how it is different from the simpler sharpen.c and why it might be a better implementation for real-time frame transformation.

4) [20 points] Build the code in faceDetect and run it on the lena.jpg image (the demo from Laz's installation test). Now run it on a download of my UAA picture and provide the face detection image in your report. Read the OpenCV tutorial page on Haar Cascades for face detection and describe in your own words how this works.

5) [20 points] Author your own simple OpenCV code to open a window with an image in it and draw a 4 pixel width border around the image and a single pixel YELLOW cross-hairs down the middle column of the image (as close to center as possible) and through the middle row of the image (as close to center as possible). Provide you graphically annotated image in your report at 180x320 resolution (you may want to refer to the simple-cv example code to help get you started). Note that the example code provides an example of the use of the ***Mat*** object and an ***IplImage*** object along with a simple method to directly access the multi-channel array associated with both the ***IplImage*** and the ***Mat***.

Overall, provide a well-documented professional report of your findings, output, and tests so that it is easy for a colleague (or instructor) to understand what you've done. Include any C/C++ source code you write (or modify) and Makefiles needed to build your code. I will look at your report first, so it must be well written and clearly address each problem providing clear and concise responses to receive credit.

In this class, you'll be expected to consult the Linux and OpenCV manual pages and to do some reading and research on your own, so practice this in this first lab and try to answer as many of your own questions as possible, but do come to office hours and ask for help if you get stuck.

Upload all code and your report completed using MS Word or as a PDF to Blackboard and include all source code (ideally example output should be integrated into the report directly, but if not, clearly label in the report and by filename if test and example output is not pasted directly into the report). ***Your code must include a Makefile so I can build your solution on Ubuntu VB-Linux. Please zip or tar.gz your solution with your first and last name embedded in the directory name.***