Exercise #1 – Familiarity with Linux OpenCV

DUE: AS INDICATED on Canvas (before Midnight)

Please thoroughly read Computer and Machine Vision by E.R. Davies, 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. You will want to refer to example code found here - http://ecee.colorado.edu/~siewerts/extra/ecen5763/ecen5763_code/.

For this course, all work should be done on a native Linux system, preferably an NVIDIA Jetson Nano or R-Pi3b+, but can be done on any native installation of Linux with a USB port (as a backup to an embedded system configuration).

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. Note that OpenCV 2.x and 3.x have differences that can require modification to code, so you will find examples for OpenCV 2.x and examples tested for OpenCV3.x for this class, but on the Jetson Nano and R-Pi3b+, you should follow OpenCV 3.x examples and adapt other code as necessary. Also, be sure consult the correct version of the OpenCV documentation (https://docs.opencv.org/, e.g. for 3.4 use https://docs.opencv.org/3.4.5/ and you can write a simple program to determine version you have installed - https://www.learnopencv.com/how-to-find-opencv-version-python-cpp/).

It is preferred that you complete all work an embedded NVIDIA Jetson Nano or TK1 or equivalent system that runs embedded Linux and OpenCV, but while you're getting this setup, you can use a native Linux laptop or desktop in the meantime with cameras and VB-Linux without a camera, instead using stored MPEG video and PNG or JPEG images.

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) [15 points] Test your Jetson embedded 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 Jetson Nano Getting Started, OpenCV installation for R-Pi3 or the JetPack 3.x install for NVIDIA Jetson TK1, TX1, TX2 or generic install instructions for OpenCV Linux or Windows. 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) [15 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 CU faculty picture and provide the face detection image in your report. Read the OpenCV tutorial page on Haar Cascades for face detection for OpenCV 3.x, or OpenCV 2.x 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 with180x320 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***.

[20 points] 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 and make sure your code is well commented, documented and follows coding style guidelines. 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 D2L 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 an embedded***

Linux system (R-Pi 3b+ or Jetson). Please zip or tar.gz your solution with your first and last name embedded in the directory name and/or provide a GitHub public or private repository link. Note that I may ask you or SA graders may ask you to walk-through and explain your code. Any code that you present as your own that is "re-used" and not cited with the original source is plagiarism. So, be sure to cite code you did not author and be sure you can explain it in good detail if you do re-use, you must provide a proper citation and prove that you understand the code you are using.

Grading Rubric

[10 points] Read and summarize the main points…

[2 pts] main point #1 ____

[2 pts] main point #2 ____

[2 pts] main point #3 ____

[4 pts] summary overall ____

[15 points] Test your VB-Linux and OpenCV installation…

[3 pts] example #1 ____

[3 pts] example #2 ____

[3 pts] example #3 ____

[6 pts] Sobel description ____

[15 points] Work with example sharpen PSF code…

[3 pts] build, run, test ____

[3 pts] trees or similar ____

[4 pts] explanation of PSF ____

[6 pts] sharpen_grid vs. sharpen ____________________________________

[20 points] Build the code in faceDetect…

[5 pts] build, run, test ____

[5 pts] picture example ____

[10 pts] explanation of Haar ____

[20 points] Author your own simple OpenCV code to open a window with an image in it…

[5 pts] build, run, test ____

[5 pts] 180x320 graphics ____

[10 pts] cross hairs and annotation ___________________________________

[20 points] Quality of reporting and code quality and originality:

[10 pts] Professional quality of reporting, testing and analysis (0…6 is below average, 7 is average, 8 is good, 9 excellent, and 10 is best overall.)____

[10 pts] Code quality including style, commenting, originality, proper citation for re-used code, modified code, etc.____