Exercise #2 – Familiarity with Linux OpenCV, Camera Interface and Digital Video

DUE: AS INDICATED on Canvas

Please thoroughly read Computer and Machine Vision by E.R. Davies, chapter 2 and Learning OpenCV (on Canvas) and chapters 2, 3 and 4. Also, for this lab, please refer to this Example Code for OpenCV camera interfacing and frame transformation as well as FFMPEG frequently asked questions.

Goals for this lab include familiarity with OpenCV for interactive camera application development and comparison to batch video analytics using FFMPEG and OpenCV.

For this lab you MUST have Native Linux and a USB webcam (e.g. Logitech C200, C270 or one supported by the Linux UVC driver - http://www.ideasonboard.org/uvc/). If you don't have this, you can come see me and I can set you up with a loaner camera and/or an on-campus Native Linux option. You must have a camera for all labs going forward from this point.

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 #2 Requirements:

1) [10 points] Verify that your ffmpeg or avconv installation on Jetson Linux or equivalent (e.g. Ubuntu LTS native install on a laptop) is working by taking a video from the Open Source HD or Big Buck Bunny (Lab-1-and-2-Examples/) and converting the first 100 frames into PPM or JPEG individual frames. Paste the 100th frame from either into your report (read FFMPEG FAQ).

2) [10 points] Using GIMP installed on your VB-Linux or Native Linux system, take the 100th frame from Big Buck Bunny or the Open Source HD and apply Sobel to it using the GIMP tools – put your transformed frame into your report and describe any options you set or used.

3) [30 points] Test your Native Linux and OpenCV installation with a USB webcam by downloading video capture examples (capture-viewer, simple-capture, or simpler-capture) and compare the code for each, try building, and running each and note how each works in good detail. Show me a screen dump of a scene from your home lab using the version you like best. Compute your frame rate using time-stamp information (clock_gettime is best, but also gettimeofday for example) and provide the average rate, worst-case and jitter for a time period of 1 minute or more of run time.

4) [30 points] To start exploring OpenCV in more detail, modify the capture-transformer code so that it displays either Sobel or Canny edge transformations based on a key press of "C" or "c" for Canny and "S" or "s" for Sobel. Again, compute the frame average rate, worst-case rate and jitter (+/- frame rate) for both Canny and Sobel test runs. Add code for this analysis as needed.

[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 Canvas 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] Using ffmpeg (avconv) to extract frames from MPEG encoded video captured.

[5 pts] ffmpeg command used to extract frames from open source video _____

[5 pts] correct 100th frame and command indicating proper frame _____

[10 points] Work with GIMP to analyze an individual image

[5 pts] Sobel transform of extracted frame _____

[5 pts] Description of options used to create transform and why they were selected

_____

[30 points] Work with capture and transform examples with EMVIA system and camera.

[10 pts] capture-viewer, simple-capture or simpler-capture build, run, test

_____

[10 pts] Code for capture performance analysis by instrumenting with clock_gettime or gettimeofday and logging time-stamps to syslog

_____

[10 pts] Analysis of logged data to compute average rate, worst-case and jitter over period of 1 minute or more (typically 1800 frames or less for 30 Hz camera)_____

[30 points] Work with capture and transformation examples and code in OpenCV.

[5 pts] capture-transformer build, run, test _____

[10 pts] Canny and Sobel switch with example of each provided

_____

[15 pts] Code additions well documented to provide Canny/Sobel transformation based upon user command input _____

[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._____