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

DUE: AS INDICATED on D2L (before Midnight)

Please thoroughly read Computer and Machine Vision by E.R. Davies, chapter 2 and Learning OpenCV (on D2L) 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.

Exercise #2 Requirements:

1) [10 points] Verify that your ffmpeg installation on Ubuntu Linux (e.g. 12.04 LTS on Virtual Box) is working by taking a video from the Open Source HD or Big Buck Bunny (http://ecee.colorado.edu/~siewerts/extra/ecen5763/ecen5763_media/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) [20 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 (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) [40 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.

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.***