Exercise #2 – Edge Detection using Cameras and Digital Video

DUE: AS INDICATED on Canvas

Please thoroughly read notes through Week-4 while working on this lab and OpenCV.org documentation for 4.1.1. Also, for this lab, please refer to this Example Code and Lab2-Examples 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 2g running JetPack 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. Recall that **CSCI 612 starter code** has generally been tested for OpenCV 3.x, but latest (https://developer.nvidia.com/embedded/jetpack) installs OpenCV 4.x, so please update and adapt starter code as necessary. The starter code that has been specifically tested with OpenCV 4.1.1 is - https://www.ecst.csuchico.edu/~sbsiewert/csci612/code/computer_vision_cv4_tested/ and similarly, you can find more example on opencv.org for 4.1.1 here - https://docs.opencv.org/4.1.1/examples.html.

Exercise #2 Goals and Objectives:

1) [10 points] Verify that your ffmpeg installation on JetPack Linux is working (***already installed on Jetson Nano 2g***) by taking a video from the Open Source HD or Big Buck Bunny (Lab2-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, and class notes on FFMPEG).

2) [15 points] Using ffmpeg on your Jetson, take the **100th frame from Big Buck Bunny** or the **400th frame from Open Source HD** (your choice) and apply Sobel Edge Detection to it using with OpenCV – put your transformed frame into your report and describe any options you set or used. Explain the concept behind the Sobel operator as best you can.

3) [15 points] Using ffmpeg on your Jetson, take the **200th frame from Big Buck Bunny** or the **800th frame from Open Source HD** and apply Canny Edge Detection to it using with OpenCV – put your transformed frame into your report and describe any options you set or used. Explain the concept behind the Canny edge detector as best you can.

4) [30 points] Compare the OpenCV camera capture code from previous work and your USB webcam by downloading video capture examples (OpenCV 4.x simpler-capture-4/capture.cpp, and simple-capture/ C code built bottom-up with V4L2). Try building, and running each camera capture application and note how each works in good detail. Show a screen dump from each and describe which version you like best and why. Compute your frame rate using time-stamp information (using posix_clock_gettime is best, but older code may still use https://man7.org/linux/man-pages/man2/settimeofday.2.html) and provide the average frame rate for a time period of 1 minute or more of run time.

5) [30 points] Using the OpenCV camera capture code from previous work and your USB webcam, create a viewer where you can turn on/off edge detection for Canny and/or Sobel by keystroke (e.g., "c" for Canny, "s" for Sobel, "n" for None). You should be able to build this by combining work on Sobel, Canny, and OpenCV camera capture code you have already completed. This example may however help - capture-transformer which was written for OpenCV 3.x (so update as needed or write your own!). Again, compute the frame average rate for Canny and Sobel test runs using ***posix_clock_gettime*** time stamping. Add code for this analysis as needed. Consider making a slider or other interactive features for thresholds used for Canny and/or Sobel, but this is not required.

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 and documented. 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 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). ***Your code must include a Makefile so I can build your solution on my Jetson Nano 2g. 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 to extract frames from MPEG encoded video captured.

Problem	Points Possible	Score	Comments
ffmpeg command used to extract frames from open source video	5		
correct 100th frame and command indicating proper frame	5		
TOTAL	10		

[15 points] Work with FFMPEG to analyze Sobel edges in a particular frame.

Problem	Points Possible	Score	Comments
Sobel transform of extracted frame	10		
Description of Sobel operator	5		
TOTAL	15		

[15 points] Work with FFMPEG to analyze Canny edges in a particular frame.

Problem	Points Possible	Score	Comments
Canny transform of extracted frame	10		
Description of Canny edge detector	5		
TOTAL	15		

[30 points] Comparison of camera capture written bottom-up with V4L2 to OpenCV.

Problem	Points Possible	Score	Comments
OpenCV vs. V4L2 camera capture code build, run, test	5		
Code for capture performance analysis by instrumenting with posix_clock_gettime and logging timestamps to syslog	10		
Analysis of logged data to compute average rate over period of 1 minute or more (typically 1800 frames or less for 30 Hz camera)	15		
TOTAL	30		

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

Problem	Points Possible	Score	Comments
capture-transformer build, run, test	5		
Canny and Sobel switch with example of each provided	10		
Canny/Sobel transformation based upon user command with computation of average frame rate	15		
TOTAL	40		

Report file MUST be separate from the ZIP file with code and other supporting materials.

<u>Rubric for Scoring for scale 0…10</u>

Score	**Description of reporting and code quality**

0	No answer, no work done
1	Attempted and some work provided, incomplete, does not build, no Makefile
2	Attempted and partial work provided, but unclear, Makefile, but builds and runs with errors
3	Attempted and some work provided, but unclear, build warnings, runs with no apparent error, but not correct or does not terminate
4	Attempted and more work provided, but unclear, build warnings, runs with no apparent error, but not correct or does not terminate
5	Attempted and most work provided, but unclear, build warnings, runs with no apparent error, but not correct or does not terminate
6	Complete answer, but does not answer question well and code build and run has warnings and does not provide expected results
7	Complete, mostly correct, average answer to questions, with code that builds and runs with average code quality and overall answer clarity
8	Good, easy to understand and clear answer to questions, with easy to read code that builds and runs with no warnings (or errors), completes without error and provides a credible result
9	Great, easy to understand and insightful answer to questions, with easy-to-read code that builds and runs cleanly, completes without error, and provides an excellent result
10	Most complete and correct - best answer and code given in the current class