# ECEN5623, Real-Time Systems:

## Exercise #4 – Real-Time Continuous Media

DUE: As Indicated on Canvas

Please thoroughly read Chapters 7, 8 & 16 in the text.

Please see example code provided - http://mercury.pr.erau.edu/~siewerts/cec450/code/ Note that the computer_vision_cv3_tested versions of the code have specifically been updated and tested on the Jetson upgrade for OpenCV 3.1. See installation instructions for OpenCV on the DE1-SoC on Canvas if using the DE1-SoC, and similar for the Raspberry Pi.

This lab introduces the use of cameras for computer vision applications for emergent real-time systems such as intelligent transportation, but also for traditional real-time instrumentation applications including machine vision, digital imaging for science and defense, and avionics instrumentation.

**Exercise #4 Requirements**:

1) [10 points] Obtain a Logitech C200 camera or equivalent and verify that is detected by the DE1-SoC, Raspberry Pi or Jetson Board USB driver. You can check the camera out from the TA's or purchase one of your own, or use another camera that has a compliant UVC driver. Use ***lsusb***, ***lsmod*** and ***dmesg*** kernel driver configuration tool to make sure your Logitech C200 USB camera is plugged in and recognized by your DE1-SoC, Raspberry Pi or Jetson (note that on the Jetson, it does not use driver modules, but rather a monolithic kernel image, so lsmod will not look the same as other Linux systems – see what you can find by exploring /cat/proc on you Jetson to find the camera USB device). For the Jetson, do ***lsusb | grep C200*** and prove to the TA (and more importantly yourself) with that output (screenshot) that your camera is recognized. For systems other than a Jetson, do ***lsmod | grep video*** and verify that the UVC driver is loaded as well (http://www.ideasonboard.org/uvc/ ). To further verify, or debug if you don't see the UVC driver loaded in response to plugging in the USB camera, do ***dmesg | grep video*** or just ***dmesg*** and scroll through the log messages to see if your USB device was found. Capture all output and annotate what you see with descriptions to the best of your understanding.

2) [10 points]

   Option 1: Camorama

   If you do not have ***camorama***, do ***apt-get install camorama*** on your DE1-SoC, Raspberry Pi, or Jetson board [you may need to first do ***sudo add-apt-repository universe; sudo apt-get update***]. This should not only install nice camera capture GUI tools, but also the V4L2 API (described well in this series of Linux articles - http://lwn.net/Articles/203924/ ). Running

camorama should provide an interactive camera control session for your Logitech C2xx camera – if you have issues connecting to your camera do a "man camorama" and specify your camera device file entry point (e.g. /dev/video0). Run camorama and play with Hue, Color, Brightness, White Balance and Contrast, take an example image and take a screen shot of the tool and provide both in your report.

Option 2: Cheese

If you do not have ***cheese***, do ***sudo apt-get install cheese*** on your Jetson, Raspberry Pi or DE1-SoC board or other native Linux system. This should not only install nice camera capture GUI tools, but also the V4L2 API (described well in this series of Linux articles - http://lwn.net/Articles/203924/ . Running cheese should provide an interactive camera control session for your Logitech C2xx camera – if you have issues connecting to your camera do a "man cheese" and specify your camera device file entry point (e.g. /dev/video0). Show that you tested your camera with a cheese screen dump and test photo.

3) [10 points] Using your verified Logitech C200 camera on a DE1-SoC, Raspberry Pi or Jetson, verify that it can stream continuously using to a raw image buffer for transformation and processing using example code from the computer-vision or computer_vision_cv3_tested folder such as simple-capture, simpler-capture, or simpler-capture-2. Read the code and modify the device that is opened if necessary to get this to work. Provide a screen shot to prove that you got continuous capture to work. Note that simpler capture requires installation of OpenCV on your DE1-SoC, Raspberry Pi, Jetson, or native Linux system. For the Jetson this will likely already be available on your board, but if not, please follow simple instructions found here to install openCV [the "Option 2, Building the public OpenCV library from source" is the recommended approach with –DWITH_CUDA=OFF. Don't install CUDA and please leave it off when you build OpenCV.

   For the DE1-SoC please find the files soc_system.rbf and SettingUp.pdf on Canvas. They contain instructions for setting up board and installing OpenCV. The TAs have set up using these and are able to complete exercise 4 requirements. The cmake command for opencv installation has been changed so that it works on the board too. Alternatively, you can use OpenCV port found here: http://rocketboards.org/foswiki/view/Projects/OpenCVPort. If you have trouble getting OpenCV to work on your board, try running it on your laptop under Linux first. You can use OpenCV install, OpenCV with Python bindings for 2.x and 3.x for this.

4) [20 points] Choose a continuous transformation OpenCV example from computer-vision such as the canny-interactive, hough-interactive, hough-eliptical-interactive, or stereo-transform-impoved or the from the same 4 transforms in computer vision cv3 tested. Show a screen shot to prove you built and ran the code. Provide a detailed explanation of the code and research uses for the continuous transformation by looking up API functions in the

OpenCV manual (http://docs.opencv.org ) and for stereo-transform-improved either implement or explain how you could make this work continuously rather than snapshot only.

5) [50 points] Using a Logitech C200, choose 3 real-time interactive transformations to compare in terms of average frame rate at a given resolution for a range of at least 3 resolutions in a common aspect ratio (e.g. 4:3 for 1280x960, 640x480, 320x240, 160x120, 80x60) by adding time-stamps (there should be a logging thread) to analyze the potential throughput. You should get at least 9 separate datasets running 1 transformation at time. Based on average analysis, pick a reasonable soft real-time deadline (e.g. if average frame rate is 12 Hz, choose a deadline of 100 milliseconds to provide some margin) and convert the processing to SCHED_FIFO and determine if you can meet deadlines with predictability and measure statistical jitter in the frame rate relative to your deadline for both the default scheduler and SCHED_FIFO in each case.

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. Any real-time control code must be in C/C++. 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. Note that it is up to you to pick an OOP for implementation and tools of your choice to develop and test your application – this is a large part of the challenge of this assignment.

**Grading Rubric**

[10 points] Camera and device driver verification:

[5 points] USB hot-plug _____

[5 points] UVC driver verify _____

[10 points] Camera streaming checkout:

[5 points] Use of tools such as camorama_____

[5 points] Verification_____

[10 points] Camera continuous streaming tests and applications:

[5 points] Build and test _____

[5 points] Streaming verification_____ _____

[20 points – **choose 2**] Continuous transformation tests and applications:

[10 points] Canny Demonstration, CPU loading by core and Description_____

[10 points] Hough Lines Demonstration, CPU loading by core and Description_____

[10 points] Hough Elliptical Demonstration, CPU loading by core and Description_____

[10 points] Example Stereo Transform Improved Demonstration, conversion to continuous transformation, CPU loading by core and Description _____

[50 points] Soft real-time analysis and conversion to SCHED_FIFO with predictability analysis:

[10 pts] Design concepts_____

[15 pts] Algorithm analysis_____

[15 pts] Prototype analysis_____

[10 pts] Final predictable response jitter analysis_____