ECPS CS244 - Assignment 3 Guidelines

In this assignment, you are required to finish the implementation of a real-time canny edge detection using a Raspberry Pi and a webcam. The sample code of processing one frame captured from camera is provided on the class website, you will need to extend the code to process multiple frames in real time.

Step 1 – Setting up the Raspberry Pi webcam API

1. To download and install the raspicam api, you need to create an empty directory (e.g. raspicam), then go through the following commands:
```
$ git clone https://github.com/cedricve/raspicam ./raspicam
$ cd raspicam
$ mkdir build
$ cd build
$ cmake ..
$ make
$ sudo make install
$ sudo ldconfig
```
2. You can find the detailed installation instructions and the source code on github: https://github.com/cedricve/raspicam
3. After you done with the installation, you can execute raspicam_test to test the success of your installation with:
```
$ raspicam_test
```

Step 2 – Downloading, testing and understanding the source code

1. Download the C and C++ source codes from the course website. The codes are canny_local_raspicam.c and raspicam_canny.cpp.
2. To compile the source code use the following command:
```
$ g++ raspicam_canny.cpp -o raspicam_canny -I/usr/local/include -lraspicam –lm
```
3. To test it, type:
```
$ ./raspicam_canny 1.0 1.0 0.9
```
4. Compared to the previous assignment, we are feeding the captured camera image to the C code instead of test.pgm by using the C++ script.
5. Now, go over the source codes and understand how they work and how to capture image data from webcam APIs .
6. Describe the C++ script in your report.

Step 3 – Modifying the code for multiple images

1. Extend the code for processing multiple images. Notice that the provided code only processes one image, you will need to add a loop in the code to make processing any number of captured images. You are required to implement this part yourself.
2. The sample code already provides you the method that calculates the execution time of processing one frame and gives you the calculated frame per second. Your

implementation should also include the extension of this timing calculation for multiple images in your code. Your fps calculation should be an averaged value. (Total execution time divided by the number of frames.)

3. You are required to include your timing results in your report.
4. Show your modifications during the evaluation session.

Step 4 – Applying parallelization to the system

1. The canny_local_raspicam.c doesn't include the parallelization.
2. Recall the implementation of pthreads and OpenMP from Assignment 2.
3. Implement the multi-threading to the canny_local_raspicam.c script for blur in the -x and -y directions to increase the speed of the computation similar to the Assignment 2.
4. Compare the improvement of the camera capturing and edge detection system in terms of time.
5. Don't forget to add the name of the library while compiling the code (-lpthreads, -fopenmp)
6. Show your results in the report and during evaluation session.

Step 5 – Encoding the processed images into a video

1. For you to deeper understand how your real-time application works, you can encode your processed images into an .h264 video and play it.
2. First you need to install the ffmpeg codec. The installation can use the following command, but first don't forget update your Raspberry Pi as explained in Assignment 1
   ```
   $ sudo apt-get install ffmpeg
   ```
3. After you install the codec, create around 1000 processed image .pgm files in a certain directory with the file names of: Frame0001.pgm, Frame0002.pgm, Frame0003.pgm, … Frame1000.pgm.
4. Don't forget to modify the code to change the name of the output image files (i.e. Frame%04d.pgm.)
5. To encode the video, type the following command:
   ```
   $ ffmpeg -i video.h264 frame%04d.pgm
   ```
6. You can play your videos by using omxplayer.
7. Show your videos during the evaluation session.