ECE 2420 Programming Exercise #5

(Threading in C++)

# Overview

In this programming exercise the coder will construct a pipeline of threads which process a video file. Each of these threads will accept an image matrix (one frame of the video), perform some transform on the image, pass it to the next pipeline stage, and then display the image at that stage of the pipeline, (one graphical window at each pipeline stage). The library package this PEX will utilize is called OpenCV (Open Computer Vision). It is a highly capable open source package, and this PEX merely scratches the surface of its functionality. Should this area of study interest you, further exploration will be rewarding.

 At the conclusion of this exercise, the coder will have completed a tool that will display a video in its original form in one window, a second window will show the video in black and white, and a third window will show the video after an edge detection algorithm has been executed. All windows will display simultaneously.

# Video Processing Architecture

In this exercise, the coder will first construct a thread safe FIFO template class. This will allow the coder to pass arbitrary objects to other threads safely. Students should base their object FIFO template class on the well documented partial header provided in this package (i.e. Fifo.hpp). Remember that template method implementations must be included in the header because templates are **always** a compile time type of polymorphism. Therefore, students will have to add implementations to the header file, but they should not change the template class definition.

The second phase of the exercise is to create the Transformer base class. The right most three threads in the figure 1 diagram inherit from this base class due to the common functionality. Each of them read a frame from the inbound FIFO. Then they execute a transform on the image and write it to the output FIFO. Finally, they display the image to a graphical window. The only difference between each of the threads is the action accomplished by the transform function. A well-documented header for this class has been provided as part of this PEX package. Read the in-line documentation closely; there are many hints contained there. Students should implement the Transformer base class with no modifications to the provided header.

Next, the coder needs to implement three derived classes which inherit functionality from the Transformer class. The first does nothing, the second converts the image to black and white, and the last does a Canny image detection algorithm. For this piece, the code will have to look at documentation and examples for the OpenCV package. OpenCV provides function for both color conversion and edge detection. Students are encouraged to use these functions for their first attempt at this exercise. Once the PEX functions well with the provided functions, students may choose to write their own color conversion and edge detection algorithms (google Sobel edge detection). Extra credit will be awarded for self-implemented transform algorithms.

Finally, students must create a main function to instantiate the correct objects, (Fifos and Transformers), and connect them together correctly. The coder must open a video file, read a number of frames, pass them through the pipeline, wait for all processing to complete, and then cleanly end the process/threads. It is this piece of code that must create an executable callback object that the Tranformer classes will use to display the frames. The imshow and waitKey OpenCV function do not appear to be thread safe. Make sure that you ensure only one thread is executing these particular API calls at any given time.



Figure 1. Video Processing Architecture

# Programming Concepts

This exercise covers many programming concepts including threads, reference counting pointers, templates, STL library, mutexes, condition variables, abstract classes and inheritance, dynamic linking, lambdas, closures, functors, and introductory computer vision.

# System Requirements

The design must use the provided headers verbatim. This will allow automated testing of the design you produce. See grading rubric for specific system requirements and associated grade values.

# Turn-in Procedures

Turn in all source code and required analysis via canvas by 11:59p.m. on <DUE DATE>

Grading Rubric

(ECE 2420 PEX5)

|  |  |  |
| --- | --- | --- |
| Requirement / Criteria | Available Points | Student’s Score |
| Correctly implemented thread safe FIFO template | 20 |  |
| Transformer correctly implements two constructors (The last pipeline stage has no output) | 10 |  |
| Transformer constructor starts the process thread | 10 |  |
| Transformer correctly reads/tranforms/writes/displays | 10 |  |
| Transformer wait functionality is correct | 10 |  |
| NOP transform function/class is correct | 10 |  |
| Black and white transform function/class is correct | 10 |  |
| Canny edge detect is correct | 20 |  |
| Main correctly instantiates and connects objects | 20 |  |
| Main correctly creates and passes display callback to Transform objects | 10 |  |
| Main correct displays frames and protects OpenCV API calls | 10 |  |
| Student implemented black/white transform | 20 extra |  |
| Student implemented edge detect algorithm | 40 extra |  |
| Total | **140** |  |