Harry Grenier

Assignment 6 Report

12/4/2023

Image Stack Focusing

The development of this program has had a relatively steep learning curve though im glad to be complete with each version and it has helped me better understand to use of multithreading and using third party classes. Starting off with my design process my first step was to break down what needs to happen we need to take each of the needed images and store them in a vector so that they are all loaded and ready to examine. Next it was important to get the threads created we did this two different ways with the P_thread class and the C++ thread class. Though they are different types of thread classes the logic doesnt change on how the program is created. Inside the main function we break down the arguments passed through main and store these in variables in order to make calling those values easier and in some cases callable on a global scale. We then have to initalize the application and get the image stack array created and create a output image that is all black pixels Next is setting up the threads this happens in the main function with the only difference being that pthread requires a struct in order to pass the needed parameters. Once this is compleated we just have to implement the logic for each thread. This is where we have our focusStackingThread used to do the logic for the program and is different for each version.

In version 1 we have to implement this program without any synchronization to do this we assign each thread a start and end row based on how many threads are created. Once this is decided we push these threads to the processing function and begin the logic. Because this version doesnt have any synchronization all we need to do is have each pixel grab its neighbors convert it to gray scale by suming up all the RGB values and finding the average. Once this is done we can find the max and min values in this area and find the difference. We do this for every image in the stack in order to find the best fit image. Once it is decided we copy that specific pixel to the output image. In version 2 we start the same general way except we break up the into windows which we seach randomly in order to write that pixel value to the output. Though one more significant change we make is that we allow for a single lock here we lock the output image so that only one thread can write to it at a time in order to prevent data corruption. Version 3 acts as a combination of both of these versions with us dividing each of the threads to a small area where we then implement the window once again and filling in the values randomly. Though in this version we implement multiple locks in order to septerate the data into regions that will allow for us to write pixels to that particular region.

When it come to limitations there are a few that I have noticed and should consider first being proformance bottle necks in general you would assume that using 12 thread would be more efficient then using 2 but thats actually not the case from what ive seen with just playing around with the program in every version lower thread count tends to result in a faster completed output. To add to this this program could become

pretty resource intensive with multithreading in general it consumes a large amount of resources. Though at this level its not too big of a deal being on a smaller scale. I could see how if we added even more pictures to get a more precise image this could be intensive on our hardware. Lastly I want to touch on the idea of quality though the image is significantly better then it started out to be there is definitely room for improvement when it come to image quality there are some areas of the output image that were a little rough around the edges but this could be a factor of the original image not being too precise. We have to also consider that this isnt the best way in detecting the best focus for that pixel though it does work for what we need to get done.

When it come to the extra credit I worked on creating script06.sh this is one of the first things I created on this project inorder to make testing alot easier. Overall this was a very interesting project to work on and I look forward to continuing with implementing multithreading into future projects.