Training Diary
Abdurrehman Syed
2308546

Feb - 9th - 2024
At 3pm, I started studying about the algorithm, couldn't find any specific videos on youtube, found some articles on google and am currently understanding those articles with chatGPT.
""" Zero-mean Normalized Cross Correlation (ZNCC) is an algorithm used for comparing two images to determine how similar they are, basic idea is that we slide a template image over the target image in a sliding window manner, checking for resemblances/corr- -relations. """

Feb - 8th - 2024
Learning how C++ works (compiling code and running executables) for mac - vscode.

Feb - 10th - 2024
Installing required software (Xcode) for openCL on mac m1

Feb - 22nd - 2024
Understanding the algorithm.

- B -> Window-size I_l = left-image
- I_R = right-image
- I_L_Bar, I_R_Bar = averages
- d = disparity (0 to d)

We have 2 different images: Template and target We initially slide the window over the template along the target pixel to check for comparisons, we have 'd' which is used for creating displacement in the pixels of the target image: eg, think of two images from 2 different cameras slightly different angle like 1st cam is at 0,0 and 2nd is at 0,2 if we take 2 pics from these we will have a displacement of 2 in between the cameras and therefore the angles would be slightly different now when we check for similarities in the images we would settle the difference using the 'd' element. The window size is similar to how filters work in convolutions. Averaging is used so we can detect important features in the images such that the whiter portions will have a higher value so rather than considering the entire image as bright we would only consider the bright features to be bright. Which will help in finding key features. Also we do not take the average of the entire image but only the region around the pixel. Numerator: Calculates the cross correlation between pixel values within each window. Denominator: Normalizes
Completed the implementation of ZNCC in python today, will transfer to C++ later on, followed the pseudo code and formula given in exercise notes. (Tested it with a custom matrix representation of image and is giving me a matrix of all ones since i have both template and target as same):

```
template_image =   [[1, 2, 33, 41],
                    [10, 2, 25, 1],
                    [9, 15, 5, 20],
                    [22, 1, 1, 24]]
target_image =     [[1, 2, 33, 41],
```

[10, 2, 25, 1],
[9, 15, 5, 20],
[22, 1, 1, 24]]
computed_zncc = [[1. 1. 1. 1.], [1. 1. 1. 1.], [1. 1. 1. 1.], [1. 1. 1. 1.],]

Feb - 28 - 2024

Finally installed openCL on the mac after trying for days to get the xcode and finding header file, wrote a hello world script

Feb - 29 - 2024

checked out some openCL tutorials but turns out chatGPT is way faster at teaching with appropriate code examples.

March - 1 - 2024

Converting my python zncc algorithm to C++, took a long time. Then did the matrix addition task. However, for the image part I couldn't find the header file in my PC and instead got stb-image headerfile but it is giving errors so they still need to get resolved while I did write the code for kernel command to turn the rgb image to grayscale but due to errors havent checked it yet. Finally, implemented profiling in the C++ algorithm using chrono headerfile. Quite easy.

March - 10 - 2024

The issue I couldn't resolve during this time was how to incorporate header files that I have installed using package managers such as brew, since my Operating System is Mac it is not identifying the headers. After continuously trying for 5-6 hours and researching on stackOverflow/youtube/other online resources I realized that the for mac the C++ compilers use a different directory from where they access the header files which is via XCode Application, I copied the header file folder (openCV2) in my case and pasted it in the xcode folders and finally I could see the headers working correctly!

March - 11 - 2024

My images were now finally loading, getting read, showing, and being written.

March - 15 - 2024

During this time I was now testing out the grayscale kernel and openCL conversion, to my surprise it was acting really weird, I would create a Context -> Command Queue -> Load the kernel -> Create and Build Program -> Create Kernel -> Allocate buffers and pass the arguments, Enqueue and run it but the output i was getting was either a complete black image, no image at all just png background or sometimes I would get an image but it wouldn't be Grayscale instead it will be RGB with 90% pixels missing, spent like 2 days figuring this out because i would run the cl file and sometimes it will give me an output and other times it wouldn't I finally fixed it by rechecking and fixing my kernel code - helped by LLMs. Note: after openCV was not working I also tried using LodePNG which was mentioned in the guide but it was pretty ineffective too with my results.

March - 19 - 2024

I wrote the code for accessing images in my ZNCC files.

March - 20 - 2024
It would take so long to run the file and my output was stuck at
1 1 1 1 || -1 -1 -1 -1 or 0 0 0 0 even when the input images were different, I realized that even though it seemed correct my implementation was wrong.

March - 21 - 2024
I again tried to debug the issue for hours but couldn't, one of the reasons mainly being that I am really bad at C++ and other one being that I implemented the algorithm correctly based on the pseudo code from the guide - little did I know this was not correct... (future me)

March - 28 - 2024
I gave up C++ and came back to Python implementation, I re-accessed the algorithm and how it works, created a new example problem in my mind, how I should slide the window checking out the borders the corner cases, horizontal shifts and how I should manage each pixel value based on the window mean.. I then loaded up images and tested them out to my surprise this time the algorithm seemed to work, I was no longer getting [1 1 1 1] as outputs for the disparity calculations. This took around a week.

April - 10 - 2024
I wrote the whole zncc algorithm in python and now started converting it to C++ (again) took me a couple days to do that.

April - 13 - 2024
Ran in some C++ issues again my outputs seem different from Python and C++, Python took me a lot more time to calculate disparity maps, crossCheck and occlusion filling. C++ was much faster but my results were different. I tried to debug the issue, I have using opencv's RECT function to get the window patch which seems like it was causing issues so instead of doing that I tried a different approach, this took me a week as well different trial and errors / trying out different fixes, understanding the problem.

April - 18 - 2024
My code finally runs now and calculates different maps, the disparity map looks exactly like the template given in the guide. Sadly there were no other template images given so that I could know whether my implementation beyond the disparity calculation is correct or not (like a final product - reference image).

April - 19 - 2024
Used chrono header file to implement profiling in this new code

April - 20 - 2024
Moved on to threading using the standard thread header to run the code (calculating left and right disparity concurrently) by creating two threads.

April - 21 - 2024
After threads I tried using the openMP framework using the <omp.h> header but I was once again getting the same "linker cannot find the command" issue and so I searched a lot turns out just like everything else it requires some special command to run it on mac I tried that and it worked! This took me a couple days to figure out - quite frustrating.

April - 25 - 2024
Now I moved onto the OpenCL implementation understanding how my structure should look like (decided to keep 2 kernels, 1 for resizing images and 2 for calculatingZncc)

May - 10 - 2024
Started working back on the OpenCL implementation - understanding the algorithm again after taking the break

May - 11 - 2024
Checked the configurations (as was mentioned in the task guide) to check for platform, devices, memory etc..

May - 12 - 2024
Getting kernel errors (Prints the Build log with failed execution and error codes) tried debugging but still stuck

May - 14 - 2024
Finally fixed the build log now copying my other functions in here such that I only have 1 single ZNCC kernel and from within that I call the kernel

May - 15 - 2024
Started writing the final report

May - 21 - 2024
Finally implemented the complete openCL implementation for GPU. My output images seem a little off from the CPU implementation but I'm just happy that it works. Also learned about Barriers in OpenCL which pause the execution and wait just like in threads.

May - 22 - 2024
Completed this Journal. Total Hours of Work Done: around 250-300 hours mainly because of non familiarity with C++, MacOS, and python to C++ conversion along with all the errors due to it. A good amount also went into experimentation.

May - 23 - 2024
Completed the final report and optimizations in GPU.