**YODA BLOG DRAFT**

**Problem Description:**

The YODA project involves the design and implementation of a digital accelerator. This is a piece of hardware used to offload high-speed processing from the main CPU. Our elders have decided to create a median filter and edge detection algorithm to help with image processing, built on OpenCL, a framework for writing programs in C++ that execute across heterogeneous platforms consisting of central processing units (CPUs) and graphics processing units (GPUs). OpenCL achieves a speed-up through the use of parallelized execution by allowing data and work decomposition across work groups.

The Median Filter:

The median filter is a non-linear digital filtering technique used to remove noise from an image or signal. This noise reduction is a common preprocessing step in digital image processing.

A picture containing text

Description automatically generated

Figure 1 - Median filtering example. https://www.wikiwand.com/en/Median\_filter

The algorithm involves taking a PNG or JPEG and processing the image until it is in a numerical representation. Here, the pixels can be stored in a 2D array, where each pixel has an x and y coordinate and is represented by 3 numbers, representing its red, green and blue components encoded as a number in the range 0 to 255 with 0 meaning zero light and 255 meaning maximum light. Next, the algorithm takes a fixed frame (typically 3x3 but could also be 5x5 or any other odd numbered square) centered around the current pixel and reads in the RGB values of all the pixels located within that frame. For boundary pixels, the algorithm would need to zero pad the missing pixels to ensure proper functionality. To be classed as a median filter, The algorithm sums the RGB values of each individual pixel in order to get 1 numerical value for each pixel. These numbers within the frame are ordered in ascending order, and the middle value (median) is then written out to the output 2D array which is where the median filtered image is recreated. The frame is then shifted to the next adjacent pixel until the entire image has been iterated through and the median filtered output 2D array has been created. Since each output pixel is now only represented by 1 number, it would have to be represented graphically as a gray scale image. In order for them to be displayed visually, the output can be written as a PGM (Portable gray map file), a simple format that’s easy to parse and requires no libraries.

Edge Detection:

Edge detection is a technique used in image processing to identify points with discontinuities or sharp changes in image brightness. These points where the image brightness changes sharply are called the boundaries or edges.

A person wearing a garment

Description automatically generated with medium confidence

Figure 2 - Edge detection example

A possible solution to this is the Sobel algorithm, which works by running a 3 by 3 matrix across the pixels of our grayscale image and at every iteration, measures the change in gradient of the pixels that fall within the 3 by 3 matrix, the greater the change in pixel intensity, the more significant the edge is. This algorithm will make use of matrix convolution to compute and can produce a new grayscale image whose pixel intensity reflects how close they are to an edge of the original

**Prototype Specification:**

A digital accelerator can be a piece of hardware, often in the form of an expansion board, and has the purpose of “offloading” highspeed or specialized programming/computing from the main general purpose Central Processing Unit (CPU). In other words, a digital accelerator is a device that accelerates the processing beyond the CPU’s capabilities.

Most accelerator boards implement heterogeneous computing, a system that contains different types of computing units such as multicore CPUs, Graphics Processing Units (GPUs), Digital Signal Processors (DSPs) and Field Programmable Gate Arrays (FPGAs). Hence, OpenCL is a suitable framework to solve such a problem, mainly due to the fact that the Median Filtering and Edge Detection processes can easily be implemented on the heterogeneous systems OpenCL enables it to. Without the use of OpenCL and heterogeneous computing, the host computer’s CPU relies on software to implement tasks such as Median Filtering and Edge Detection, and subsequently does the processing of these tasks sequentially, whereas the OpenCL framework will enable these tasks to be processed on dedicated GPU hardware, increasing the efficiency of the program drastically.

OpenCL provides a standard interface for parallel computing using task-based and data-based parallelism. The OpenCL program will enable parallel execution of the Median Filter and Edge Detection processes on dedicated hardware located separately from the host computer. In order to achieve such a task, the appropriate Software Development Kits (SDKs) for the heterogenous device the program will run on needs to be installed on the host computer. Depending on the host computer being used and the device/platform chosen to implement such a task, the SDKs for NIVIDIA, Intel and AMD can be used.

The basic operation of the OpenCL program will run as follows: the CPU copies data to the GPU, and subsequently interacts with a kernel to process the data. Once the kernel has finished processing all the data, the CPU of the host computer can read back the result. The kernel is essentially a function written in OpenCL that enables it to be compiled for execution on any device supported by OpenCL.

**Project Goals:**

As young Jedis nearing the end of our training, it is our goal to deliver fully functional and optimized median filtering and edge detection algorithms which could serve as tools in the fight against the dark forces of the Sith. We hope to utilize our training in OpenCL and parallel programming under Master Winberg to do so.

Our first objective is to deliver an algorithm for each application that can serve as golden measures. These won’t necessarily be the most efficient algorithms, but they will work. Before optimizing each algorithm, we want to ensure that these algorithms are robust and perform well on a wide variety of images. For the median filter algorithm, this can be done by inspecting the resolution of a range of filtered images against their originals. For the edge detection algorithm, this can be done by analysing the output image’s resemblance of the original.

Once the performance of our golden measure algorithms have been validated, we will optimize the algorithms. This will be done predominantly by exploiting the memory hierarchy and finding an optimal division of the computation into work-groups and work-items. Various metrics (such as speed and memory consumption) of the optimized algorithms will be benchmarked against the golden measures.

Chart

Description automatically generated

Figure 3 - speed and sizes of memory at different memory levels  
[Slide 1 (anu.edu.au)](https://comp.anu.edu.au/courses/acceleratorsHPC/slides/OpenCLMemory.pdf)

**References:**

* *Introduction to Digital Images* (2022). Available at: <https://web.stanford.edu/class/cs101/image-1-introduction.html#:~:text=In%20RGB%2C%20a%20color%20is,and%20255%20meaning%20maximum%20light>. (Accessed: 29 April 2022).
* *Image Edge Detection Operators in Digital Image Processing - GeeksforGeeks* (2020). Available at: <https://www.geeksforgeeks.org/image-edge-detection-operators-in-digital-image-processing/> (Accessed: 29 April 2022).
* *What is an Accelerator Board? - Definition from Techopedia* (2022). Available at: <https://www.techopedia.com/definition/568/accelerator-board> (Accessed: 29 April 2022).
* *OpenCL - Wikipedia* (2021). Available at: <https://en.wikipedia.org/wiki/OpenCL> (Accessed: 29 April 2022).
* Kirk, D. and Hwu, W. (2013) "Introduction to Data Parallelism and CUDA C", *Programming Massively Parallel Processors*, pp. 41-62. doi: 10.1016/b978-0-12-415992-1.00003-1 Available at: <https://www.sciencedirect.com/topics/computer-science/heterogeneous-computing-system> (Accessed: 29 April 2022).
* *Introduction to OpenCL - Parallel Programming* (2022). Available at: <https://leonardoaraujosantos.gitbook.io/opencl/chapter1#:~:text=OpenCL%20specifies%20a%20programming%20language,based%20and%20data%2Dbased%20parallelism> . (Accessed: 29 April 2022).
* *How the Kernel Interacts With Data in OS X OpenCL* (2022). Available at: <https://developer.apple.com/library/archive/documentation/Performance/Conceptual/OpenCL_MacProgGuide/InteractingWiththeKernelinOSXOpenCL/InteractingWiththeKernelinOSXOpenCL.html#:~:text=A%20kernel%20is%20essentially%20a,start%20running%20on%20the%20device> . (Accessed: 29 April 2022).
* *Aryaman Sharma. How Image Edge Detection Works* (2018). Available at: <https://aryamansharda.medium.com/how-image-edge-detection-works-b759baac01e2> (Accessed: 29 April 2022).
* *The OpenCL Memory Hierarchy.* Available at: <https://comp.anu.edu.au/courses/acceleratorsHPC/slides/OpenCLMemory.pdf>. (Accessed 29 April 2022)