PERFORMANCE EVALUATION OF EDGE DETECTION TECHNIQUES ON GPU USING OPENCL

Aruna Dore
4th sem M.Tech, VLSI Design and Embedded System
NMAMIT, Nitte
arunadhore@gmail.com

Sunitha Lasrado
Assistant professor, Electronics and communication
NMAMIT, Nitte
dsouzasuni@gmail.com

ABSTRACT
GPU (Graphic processing system) enhance the performance of the performance of the computing field due to its hundreds of cores in parallel. CUDA (Compute Unified Device Architecture) and OpenCL (Open Computing Language) programming models are included in GPU. The advantage of these two programming models in GPU is that developers don’t have to understand any graphics language like OpenGL (Open Graphics Language) which reduce the development time and it can be used with simple programming language. In this paper OpenCL is used as parallel programming platform, which has the advantage of cross-platform which is vendor independent. This paper also gives insight of different edge detection filters and comparison is done between parallel and sequential implementation.

General Terms
High Performance Computing, Heterogeneous computing, Parallel programming.

Keywords
OpenCL, Convolution filter, CUDA, GPU

1. INTRODUCTION
Graphics architecture provides wider memory band width and uses hundreds of simple cores in parallel processing to enhance the performance. Developers are becoming interested in harnessing parallel computing architecture of the GPU to enhance the performance for general purpose computing known as GPGPU[General-Purpose Computing on Graphic Processing units][10][1][9] was developed. GPU can be programmed using OpenCL and CUDA programming models. The advantage of these two programming models in GPU is that developers don’t have to understand any graphics language like OpenGL or other programming design which reduce the development time and parallel computing performance of GPU can be used in much easier way using simple programming language. GPUs are also on the cutting edge of processor technology; for example, the most recently announced GPU at this writing contains over 300 million transistors [9] and is built on a 110-nanometer fabrication processor.

OpenCL is a parallel framework for heterogeneous computations. It is a standard which is royalty free which is platform independent and is used to work on any kind of computational device. It is a programming language that allows the programmer to write one version of the code that can be executed virtually on any device that has OpenCL drivers. OpenCL consists of an API for coordinating parallel computations in heterogeneous computational environment and a cross–platform programming language based on C99[9].

CUDA [4] is a popular development tool for scientific GPU computing provided by the NVIDIA manufacturer for its GPU products. OpenCL [5] programming model supports parallel programming of heterogeneous multi-core architectures utilizing different CPU, DSP, FPGA and GPU Types and configuration.

CUDA is a parallel computing framework designed only for NVIDIA’s GPUs [7], and OpenCL is a standard designed for diverse platforms including CUDA-enabled GPUs, some ATI-GPUs, multi-core CPUs from Intel and AMD, and other processors such as the Cell Broadband Engine. In this paper OpenCL is used to detect edges for images of different sizes and we see that the performance of a GPU is greater than the performance got through CPU for which the explanation will be given in the next section.

2. OPENCL STANDARD
2.1 OpenCL Architecture
Open Computing Language is a framework for writing programs that execute across heterogeneous platform consisting of Central Processing Units (CPUs), Graphics Processing Units (GPUs), Digital Signal Processor (DSPs), Field Programmable Gate arrays and other computing devices. The OpenCL architecture consists of following programming models:
- Platform Model
- Execution Model
- Memory Model
- Programming Model

2.1.1 Platform Model
OpenCL platform model consists of host and ‘n’ number of compute devices. The compute device consists of compute units which in turn consist of processing elements. The host processor is usually CPU; For example, if GPU is used as device then GPU is compute unit, and each GPU contains processing cores which are processing elements. All computation is done on processing elements. These processing elements execute code as Single Instruction Multiple Data (SIMD) or Single Program Multiple Data (SPMD). Figure 1 shows OpenCL platform model[9].
2.1.2 Execution Model
The OpenCL application is split into two parts: kernel and compute device. The part that runs on the host is named host program and the part that runs on the compute devices is called a kernel. To facilitate parallel execution, an index space is created when the kernel is submitted to the device for computation. An instance of the kernel is called work-item. The work-items are organized in an N-Dimensional Range (NDRange) where N can be 1, 2, or 3. Work-items can be grouped into work-group shown in Figure 2. The host program defines the device context which includes program objects, memory objects, and kernel function. The interaction between host and OpenCL device is controlled by the host program using command queues. These commands can be memory commands which control all memory transfers, kernel execution commands which submit the kernel code for execution on the devices, and synchronization commands which control the order of command execution.

2.1.3 Memory Model
The memory model of OpenCL includes Private memory, Local memory, Constant memory, and Global memory. Host processor has host memory which is on CPU and accessible by the CPU. Compute device has global memory as well as constant memory. The compute device used in this work is GPU. The GPU architecture is discussed in the next section. OpenCL Memory model [2] can be seen in Figure 3.

Global Memory: In this region, all the work-item and work-group has both read and write access on both the compute device and the host.

Constant Memory: This is Read only—Global memory accessible to all work-items.

Local Memory: Each work-group has Local memory. Memory is shared within the work-group, i.e., work-items within the same work-group can access this memory region.

Private Memory: Memory visible to only work-item.

2.1.4 Programming Model
Under the OpenCL programming model, computation can be done in data parallel, task parallel, or a hybrid of these two models.

3. GRAPHIC PROCESSOR ARCHITECTURE
GPUs comprise of set of parallel multi-processors which are called as Streaming multi-processors (SM). Each SM contains an instruction cache, an instruction queue, a warp-scheduler, registers, 32 CUDA cores, and special-function units (SFUs) as shown in Figure 5[3]. Fermi contains up to 448 general purpose arithmetic units known as Streaming Processors (SP) ,64 Special Function Units (SFU) for computing special transcendental and algebraic functions not provided by the SPs. Groups of 32 SPs, 16 LDSTs, 4 SFUs, and 4 TEXs compose a Streaming Multiprocessor (SM). Each CUDA cores consists of single Arithmetic Logic Unit (ALU) and Floating Point Unit (FPU).
International Conference on Information and Communication Technologies

Department of Computer Science & Engineering, PESITM, Shivamogga.

64K of configurable RAM, and thread control logic. Each core has both floating-point and integer execution units.

3.1 Comparison between GPU and CPU
Modern GPUs are highly optimized and highly parallel computational units can outperform the traditional CPU both in terms of arithmetic operations and memory bandwidth. GPU uses hundreds of simple cores in parallel to enhance performance. The performance of the CPU could hardly be increased anymore by increasing the clock frequency so they developed a CPU into which they installed several cores in order to increase performance. The comparison of the number of cores on CPU and GPU is shown in figure 4.

![CPU Multiple cores](image1)

![GPU (hundreds of cores)](image2)

Fig 5: Comparison between CPU and GPU

3.1.1 Sobel Operator
The Sobel edge detector is a one of the Gradient based method. It works with the first order derivatives. It calculates the first order derivatives of the image separately for the ‘X’ and ‘Y’ axis.

Sobel operator consists of a pair of 3x3 convolution kernels as shown in Figure 6[12]. One kernel is simply the other rotated by 90°.

![Fig 6: Kernel/Filter used by sobel operator](image3)

4. IMAGE CONVOLUTION
Digital Image processing is used in wide range of applications such as medical areas, metrology fields and computer vision. Convolution[8] is a commonly used algorithm in image processing and it used to apply different kinds of effect such edge detection, image smoothing, blurring, edge enhancing. The kernel/filter is applied to input image which gives filtered output image. Different edge detection filters are used in our convolution is discussed in next section.

4.1 Edge Detection Techniques
Edge detection is a common image processing technique used in feature detection and extraction. Applying edge detection on an image can significantly reduce the amount of data needed to be processed at a later phase while maintaining the important structure of the image. The idea is to remove everything from the image except the pixels that are part of an edge.

There are many ways to perform edge detection. Two commonly used edge detection are Gradient and Laplacian based edge detector both them work with convolution.

Gradient: The gradient method (Sobel, Prewit, Robert) detects the edges by looking for the maximum and minimum in the first derivative of the image.

Laplacian: The Laplacian method searches for zero crossings in the second derivative of the image to find edges.

4.1.1 Sobel Operator
The Sobel edge detector is a one of the Gradient based method. It works with the first order derivatives. It calculates the first order derivatives of the image separately for the ‘X’ and ‘Y’ axis.

Sobel operator consists of a pair of 3x3 convolution kernels as shown in Figure 6[12]. One kernel is simply the other rotated by 90°.

![Fig 6: Kernel/Filter used by sobel operator](image3)

4.1.2 Laplacian Operator
Unlike the sobel edge detector, the Laplacian edge detector uses only one kernel. It calculates second order derivatives in single a pass hence faster to calculate but extremely sensitive to noise. The Laplacian kernel is shown in figure 7.

![Fig 7: Two commonly used Kernel/filter by Laplacian operator](image4)

4.2 Implementation
Image convolution can be efficiently implemented on massively parallel hardware, since the same operator gets executed independently for each image pixel. Figure 8 shows the convolution using a small 3 x 3 kernel. A convolution kernel works by iterating over each pixel in the input image. For each source pixel, the filter is centered over the pixel and the values of the filter multiply the pixel values that they overlay. A sum of the products is then taken to produce a new pixel value.

Steps of edge detection
- Create Image and Buffer Objects
5. EXPERIMENTAL RESULTS
The main objective is to develop the OpenCL application that runs on any GPUs like AMD, Intel, and NVIDIA etc with less computation time and implementations of OpenCL are available for the operating systems like Mac OS X 10.6 and higher, Microsoft Windows and Linux.

5.1 Experimental setup
For this experimental set up Intel i7 processor and NVIDIA GTX 470 GPU devices used. Detailed Technical specification is given in the table 1. The code is compiled using OpenCL 1.1 NVIDIA CUDA version.

Table 1. Device Technical Specification

<table>
<thead>
<tr>
<th>Specifications</th>
<th>GPU</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vendor</td>
<td>NVIDIA GeForce GTX 470</td>
<td>INTEL Core i7</td>
</tr>
<tr>
<td>No. Of cores</td>
<td>448</td>
<td>8</td>
</tr>
<tr>
<td>Clock(MHz)</td>
<td>1215</td>
<td>3400</td>
</tr>
<tr>
<td>Global Memory(MB)</td>
<td>1280</td>
<td>2047</td>
</tr>
<tr>
<td>Local Memory(KB)</td>
<td>48</td>
<td>32</td>
</tr>
<tr>
<td>Max.Work-Item size</td>
<td>1024 x 1024 x64</td>
<td>1024 x 1024 x1024</td>
</tr>
<tr>
<td>Max.Work-Group size</td>
<td>1024</td>
<td>1024</td>
</tr>
</tbody>
</table>

5.2 Results
The size of image is varied from 512 pixels to 2048 pixels. When image size is increased the serial program execution time considerably increases. From Table 1 & 2 we can see that, according to applications- Sobel filter and Laplacian filter, the executive time of GPU was faster than that of CPU.

<table>
<thead>
<tr>
<th>Input image</th>
<th>Filter Size</th>
<th>CPU(time in sec)</th>
<th>GPU (time in sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>512x512</td>
<td>3x3</td>
<td>0.053</td>
<td>0.003</td>
</tr>
<tr>
<td></td>
<td>7x7</td>
<td>0.111</td>
<td>0.003</td>
</tr>
<tr>
<td></td>
<td>9x9</td>
<td>0.149</td>
<td>0.003</td>
</tr>
<tr>
<td>1024x1024</td>
<td>3x3</td>
<td>0.364</td>
<td>0.010</td>
</tr>
<tr>
<td></td>
<td>7x7</td>
<td>0.390</td>
<td>0.010</td>
</tr>
<tr>
<td></td>
<td>9x9</td>
<td>0.555</td>
<td>0.011</td>
</tr>
<tr>
<td>2048x2048</td>
<td>3x3</td>
<td>1.187</td>
<td>0.084</td>
</tr>
<tr>
<td></td>
<td>7x7</td>
<td>2.941</td>
<td>0.084</td>
</tr>
<tr>
<td></td>
<td>9x9</td>
<td>4.377</td>
<td>0.084</td>
</tr>
</tbody>
</table>

Fig. 9 compares the performance of two applications on NVIDIA GTX 470 GPU.

Table 2. Sobel edge performance analysis for different filter size between CPU and GPU

6. Conclusion
In this paper OpenCL is used as parallel programming platform, which has the advantage of cross-platform. Sobel and Laplacian edge detection is implemented on GPU using OpenCL in C for different sizes of images. The comparison and the analysis results show that the execution time of GPU was faster than CPU. As a future expansion the adoption of OpenCL as an FPGA programming environment and comparison of speed ups on different accelerator such as DSPs, FPGAs and GPUs is considered.
7. ACKNOWLEDGMENTS
Our thanks to the Department of Electronics and Communication and Department of Computer Science Engineering, NMAMIT, Nitte for their help and guidance. Our gratitude to Institution and the Management of NMAMIT, Nitte for providing with all the facilities that helped us to complete the project.

8. REFERENCES