FAST UNIVERSITY
CS DEPARTMENT

Grayscale Image Using OpenCL

Introduction:

The purpose of this assignment is to get used to the framework of OpenCL ran on a Linux Operating System (Ubuntu in our case) and use the framework's advantage to use our CPUs and GPUs to run C/C++ tasks (a simple gray scaling task in our assignment).

Technology Used:

Hardware:

- Integrated GPU (Intel HD 620)

Software:

- OpenCL Framework
- Docker
- WSL (Windows Subsystem for Linux)
- C Programming Language
- STB Library Header Files for C/C++

System Information:

Platform Name: Intel(R) OpenCL HD Graphics

Device Name: Intel(R) Graphics [0x5917]

Maximum Group Size (According to Device): 256

Maximum Compute Units: 24

Local Dimensions per Work Group: 16 x 16

Methodology:

The general format of the program is as follows

```
//include libraries

int main() {
```

// OpenCL variables

// Specify the input and output image paths

// Load the image

// Prepare a buffer for the grayscale output

// Get the specified platform and device

// Determine device type and print the device name

// Create a context

// Create a command queue

// Create the program

// Build the program

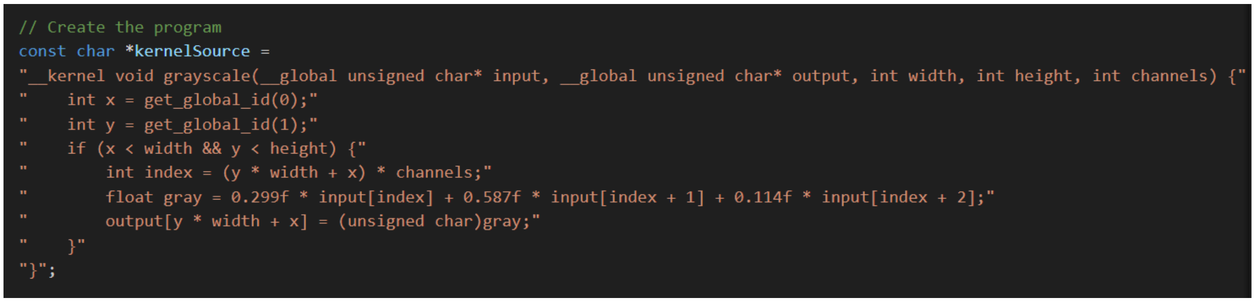
// Create the kernel

// Create memory buffers

// Set kernel arguments

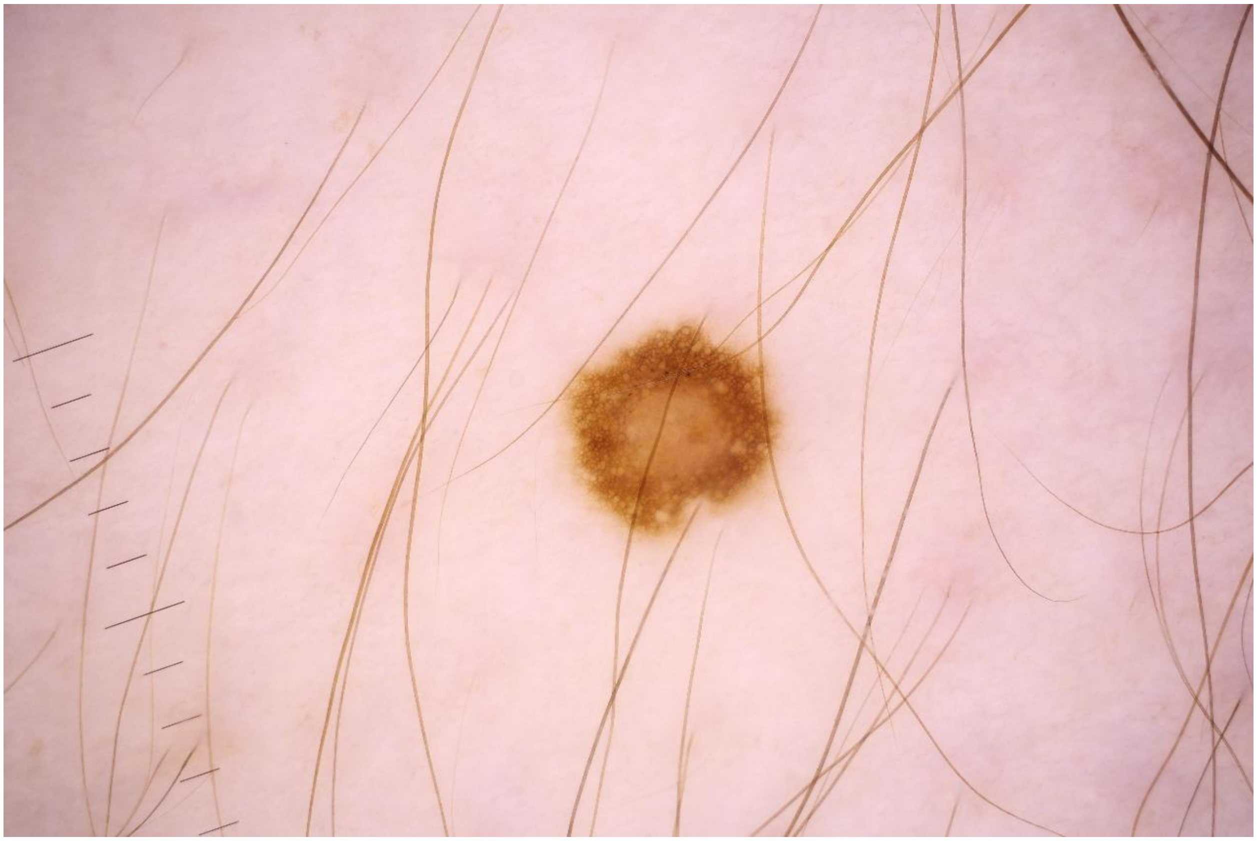
```
    // Set global and local sizes


    // Determine local size based on device type


    // Start timer


    // Execute the kernel


    // End timer


    // Read the result


    // Clean up


    // Save the Converted GrayScale Image


    return 0;
}
```

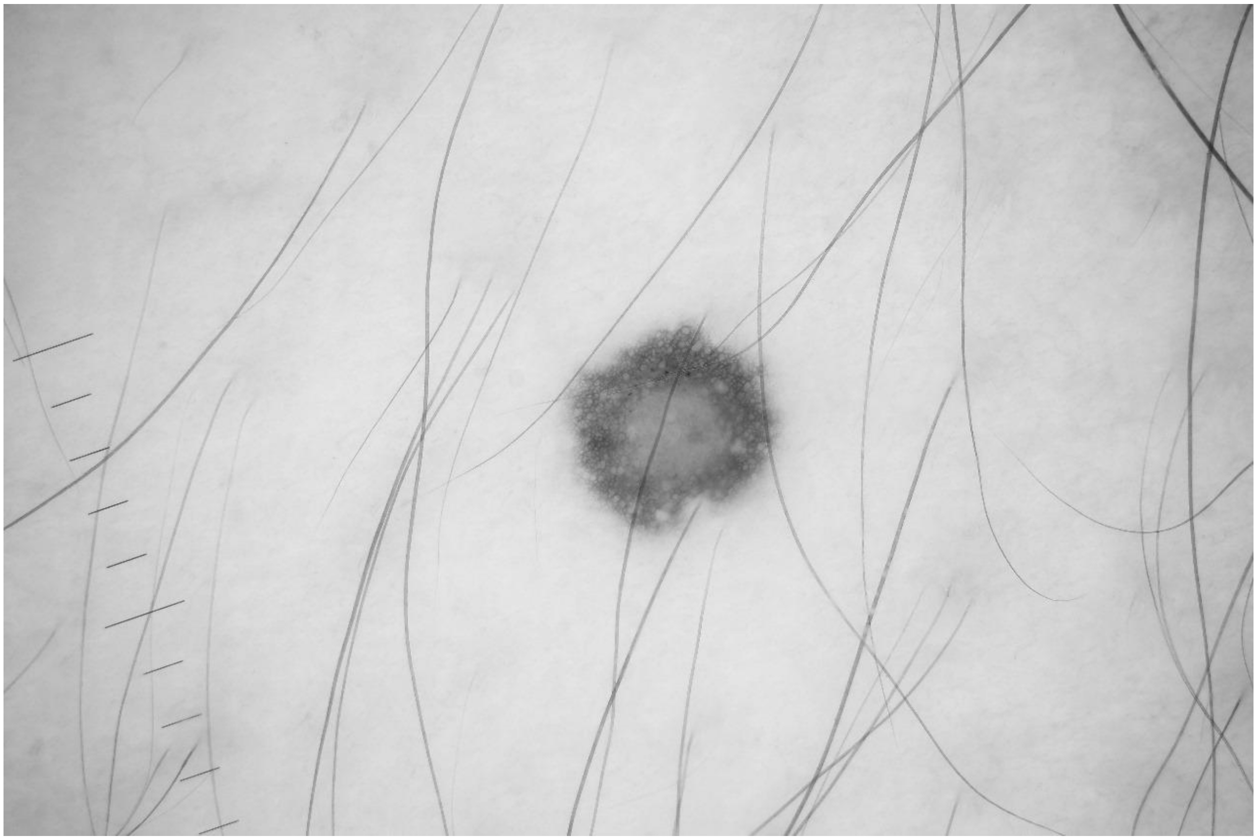
Kernel Code Snippet:



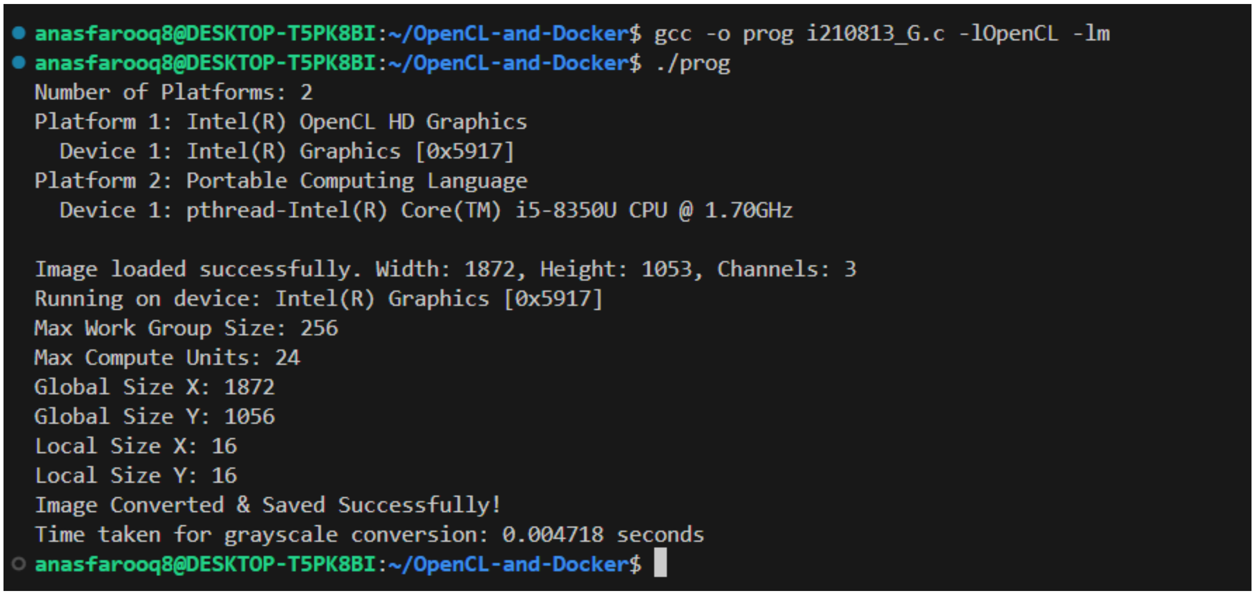
```
// Create the program
const char *kernelSource =
"__kernel void grayscale(__global unsigned char* input, __global unsigned char* output, int width, int height, int channels) {"
"    int x = get_global_id(0);"
"    int y = get_global_id(1);"
"    if (x < width && y < height) {"
"        int index = (y * width + x) * channels;"
"        float gray = 0.299f * input[index] + 0.587f * input[index + 1] + 0.114f * input[index + 2];"
"        output[y * width + x] = (unsigned char)gray;"
"    }"
"}";
```

Results:





Output Log:



```
● anasfarooq8@DESKTOP-T5PK8BI:~/OpenCL-and-Docker$ gcc -o prog i210813_G.c -lOpenCL -lm
● anasfarooq8@DESKTOP-T5PK8BI:~/OpenCL-and-Docker$ ./prog
  Number of Platforms: 2
  Platform 1: Intel(R) OpenCL HD Graphics
    Device 1: Intel(R) Graphics [0x5917]
  Platform 2: Portable Computing Language
    Device 1: pthread-Intel(R) Core(TM) i5-8350U CPU @ 1.70GHz

  Image loaded successfully. Width: 1872, Height: 1053, Channels: 3
  Running on device: Intel(R) Graphics [0x5917]
  Max Work Group Size: 256
  Max Compute Units: 24
  Global Size X: 1872
  Global Size Y: 1056
  Local Size X: 16
  Local Size Y: 16
  Image Converted & Saved Successfully!
  Time taken for grayscale conversion: 0.004718 seconds
○ anasfarooq8@DESKTOP-T5PK8BI:~/OpenCL-and-Docker$ 
```

Time Comparison:

Image 1:

- Resolution: 1920x1080
- Time Taken: 0.000785 seconds

Image 2:

- Resolution: 640x480
- Time Taken: 0.000514 seconds

Image 3:

- Resolution: 3264x2448
- Time Taken: 0.000559 seconds

Image 4:

- Resolution: 1872x1053
- Time Taken: 0.000533 seconds

Image 5:

- Resolution: 6000x4000
- Time Taken: 0.000560 seconds

Note: Time difference is only taken after and before executing of kernel, other tasks (like reading result, saving output file etc. are not catered for).

Conclusion:

In conclusion, using OpenCL to leverage the GPU's power (even integrated) makes tasks like gray scaling images much faster. It shows that dividing the work on a powerful processor like a graphics card can reduce the time taken as compared to just doing the process serially (or even in parallel) on a CPU.

References:

- STB Library Header Files: nothings/stb: stb single-file public domain libraries for C/C++ (github.com)
- OpenCL and Docker Setup: Umar-Waseem/OpenCL-and-Docker: Docker Image for Open CL (github.com)
- Docker for Desktop: Docker Desktop: The #1 Containerization Tool for Developers | Docker
- Windows Subsystem for Linux: Install WSL | Microsoft Learn