AAIMLIK ABBASI

I212540

H

Running the OpenCL Program in Docker

To run the OpenCL program in Docker, follow these step that I follows:

**Cd pdc1**

**cd OpenCL-and-Docker**

**docker build -t opencl-image** .

**docker run -it opencl-image**

**gcc host.c -o host –lOpenCL**

**./host**

**Implementation Approach Documentation**

**Description of Host Program:**

The host program is designed to utilize OpenCL for parallel execution of a grayscale conversion algorithm on an input image. Here's a breakdown of the configuration steps and the strategy to handle the complete dataset:

**Loading the Image:**

The program starts by loading a colored image using the stbi\_load function from the STB image library. This function reads the image file and extracts its dimensions and color channels**.**

**Setting up OpenCL:**

OpenCL is initialized by acquiring platform and device information using clGetPlatformIDs and clGetDeviceIDs functions, respectively.

A context, command queue, program, and kernel are created to facilitate OpenCL operations.

**Loading Kernel Source and Building the Program:**

The kernel source code, responsible for converting RGB to grayscale, is loaded into memory.

The program is then built using clCreateProgramWithSource and clBuildProgram functions.

Creating Memory Buffers and Setting Kernel Arguments:

Memory buffers are allocated on the device for input and output images using clCreateBuffer.

Kernel arguments are set to specify the input and output buffers, as well as image dimensions.

Executing the Kernel:

The kernel is enqueued for execution using clEnqueueNDRangeKernel.

**Reading Output Buffer and Saving Output Image:**

The output image data is read from the output buffer back to the host using clEnqueueReadBuffer.

A directory is created to store the output images, and the grayscale output image is saved using the stbi\_write\_jpg function.

Cleanup and Memory Management:

Memory objects, kernel, program, command queue, and context are released to free resources.

Image data allocated on the host is freed to prevent memory leaks**.**

**Description of Greyscale Conversion Algorithm:**

The grayscale conversion algorithm used in the OpenCL kernel is a weighted sum method. Each pixel's RGB values are converted to grayscale using the formula:

Gray=0.299×Red+0.587×Green+0.114×Blue

This formula calculates the luminance of each pixel, giving more weight to the green channel, followed by red and blue. The resulting grayscale value is assigned to the corresponding pixel in the output image buffer.

**Thresholds for Greyscale Value Assignment:**

No explicit thresholds for greyscale value assignment are used in this implementation. Instead, the grayscale conversion formula provides a continuous range of grayscale values based on the RGB components' intensities.

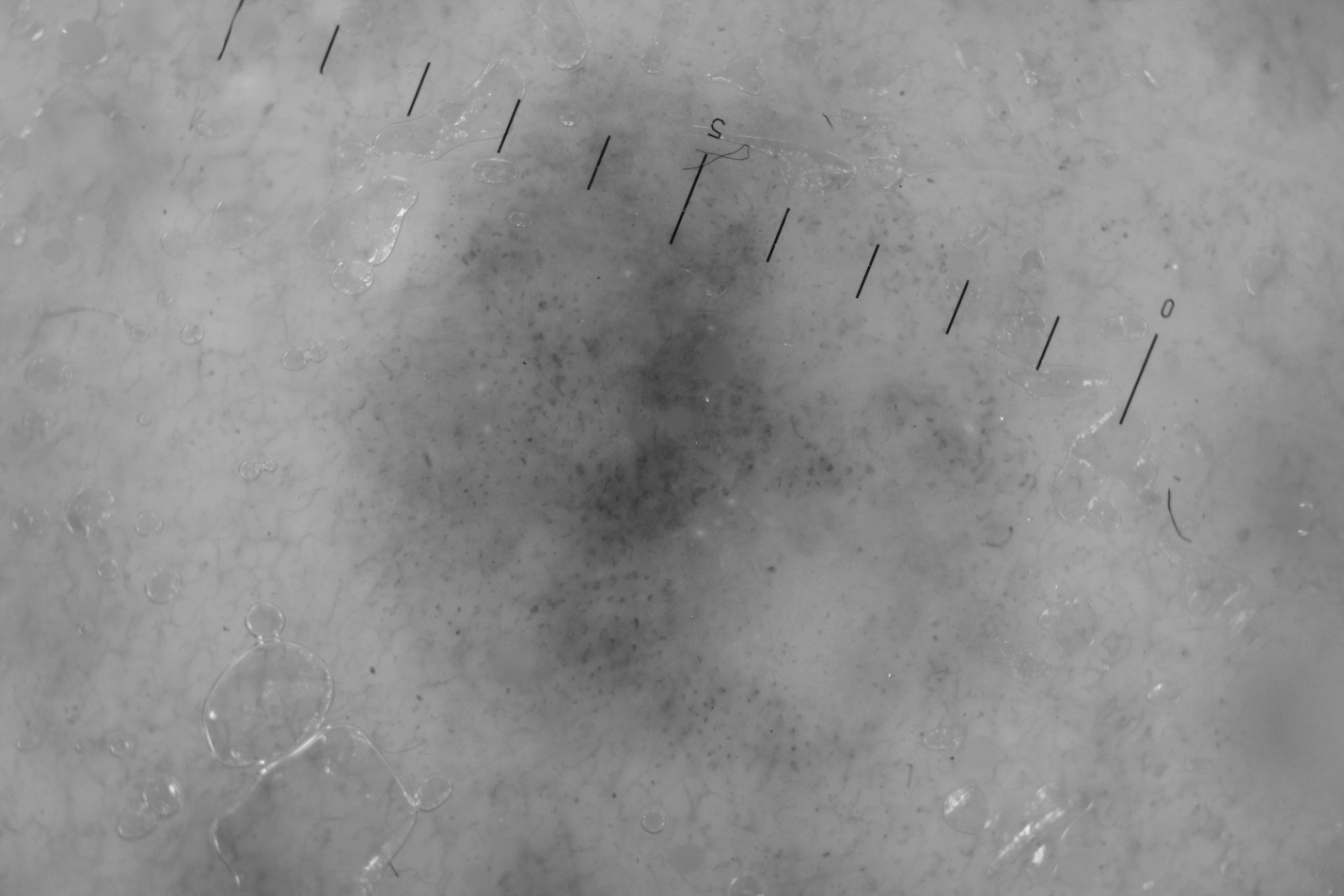
**OpenCL Kernel Design:**

The OpenCL kernel named rgb\_to\_gray is designed to process each pixel in parallel. It takes two global memory buffers as input and output for the colored and grayscale images, respectively. The kernel calculates the grayscale value for each pixel using the aforementioned formula and stores the result in the output buffer.

OUTPUTS:-















**KERNAL CODE:”**

const char \*kernelSource = "\

\_\_kernel void rgb\_to\_gray(\_\_global unsigned char\* inputImage, \_\_global unsigned char\* outputImage, const int width, const int height) {\

int x = get\_global\_id(0);\

int y = get\_global\_id(1);\

int index = y \* width + x;\

outputImage[index] = (unsigned char)(0.299f \* inputImage[3 \* index] + 0.587f \* inputImage[3 \* index + 1] + 0.114f \* inputImage[3 \* index + 2]);\

}";