Name: Ahmad Farhan

Roll no. 21I-1366

Assignment 3

Section: A

RGB to Grayscale Conversion Program Documentation

Host Program

Configuration Steps:

The host program begins by including required modules and libraries. Following a list of libraries and modules used:

1. OpenCL: Enables GPU-based parallel computation for efficient image processing.
2. OpenCV/imgcodecs4: Facilitates reading and writing various image formats.
3. OpenCV/imgcore4: Provides core image processing functionalities.
4. Filesystem: Handles directory operations for efficient dataset management.

OpenCL execution environment is set up by selecting a platform and GPU device for parallel execution. Then, OpenCL context and command queue are created for the chosen device. Next, the OpenCL kernel code is loaded from the “kernel.cl” file and a program is built for the selected device. Finally, a kernel object is initialized with the “convert” function from OpenCL code. Thus, we end up with a kernel, context, and queue ready for manipulation and parallel execution of computation.

Dataset handling Strategy:

The program provides flexibility in handling either individual images or an entire folder of images for grayscale conversion.

- For individual images: User enters an image filename and the program saves the output grayscale image in current working directory.
- For Image Folders: Following steps are performed:
    - User enters the folder name.
    - Using filesystem, a file is generated that contains all the file names of images in the given folder.
    - The transformation is applied to each image and the results are saved into an output folder.

Grayscale Conversion

Algorithm:

Following are details of the algorithm used:

- The grayscale conversion algorithm used follows the weighted sum method, where each RGB pixel value is converted to a single grayscale value.
- The NTSC formula is used for grayscale conversion:

  $$Gray = 0.299 * Red + 0.587 * Green + 0.114 * Blue.$$

- The decimal results of this formula are casted into integers with 255 cut-off value.
- This algorithm preserves human visual perception by weighing each color channel differently based on its perceived brightness contribution.
- This formula closely represents the average person's relative perception of the brightness of red, green, and blue light.

OpenCL Kernel Design:

Following are details of the approach taken to parallelize the transformation algorithm:

- The OpenCL kernel convert takes in three input buffers representing the red, green, and blue channels of the image, and an output buffer for the grayscale image. Each channel is a 1-dimensional representation of a channel of the RGB image.
- Additionally, the kernel takes a buffer for the width of the image, for assigned limits.
- Work size is defined as the image height (number of rows in image matrix) as each work-item is responsible for transforming an entire row of pixels.
- Each work-item (corresponding to a row in the image) calculates the grayscale value for its corresponding pixels using the provided formula in the kernel function.
- The kernel design ensures parallel processing of image rows, exploiting the parallelism of the GPU.

Justification for Approach:

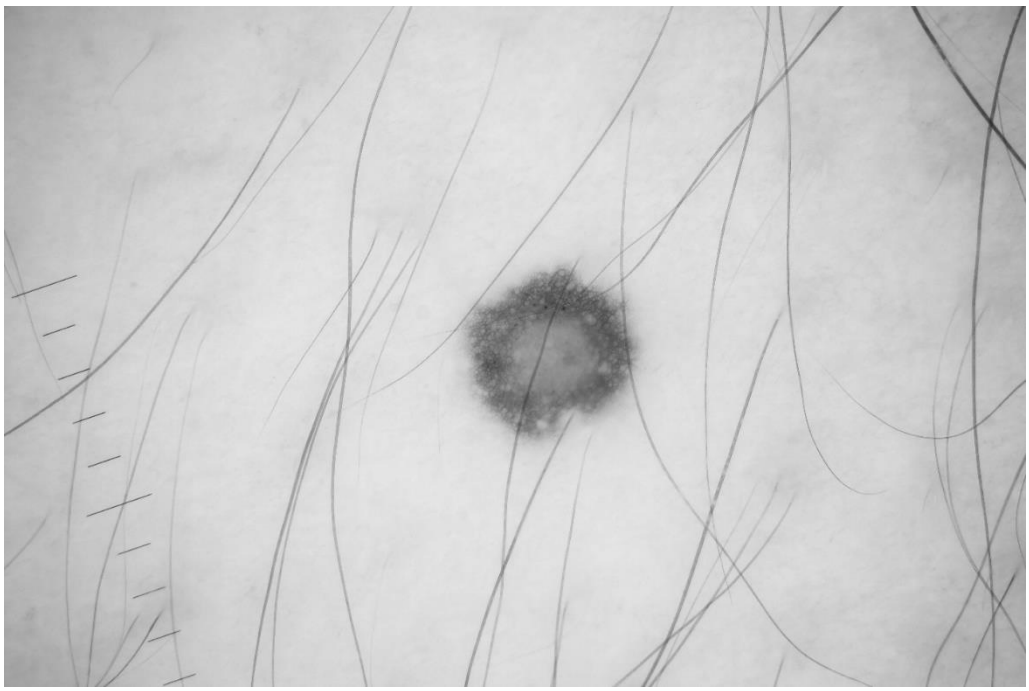
The machine used for this program had an Intel Iris Xe graphics card, which is reported to have around 700 cores. For input images of 4000x6000 pixels, pixel-wise distribution per work-item would result in 24,000,000 work-items, with each item performing minimal work. Whereas entire image per work-item would result in excessive input sizes , thereby leading to higher load imbalance and reduced efficiency of parallelism. Thus, row wise distribution provides just the right balance for individual and collective sets of input images.

Output Results

Example 1: ISIC_0052349

RGB Input Image:

Grayscale Output Image:



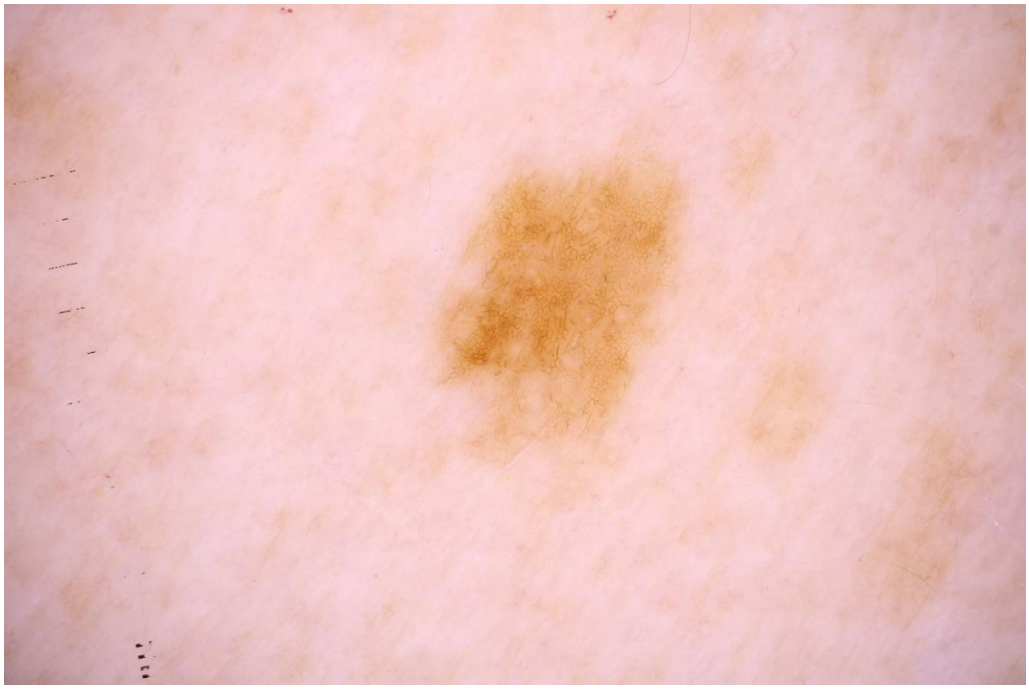
Example 2: ISIC_0073313

RGB Input Image:

Grayscale Output Image:

Example 3: ISIC_0198376

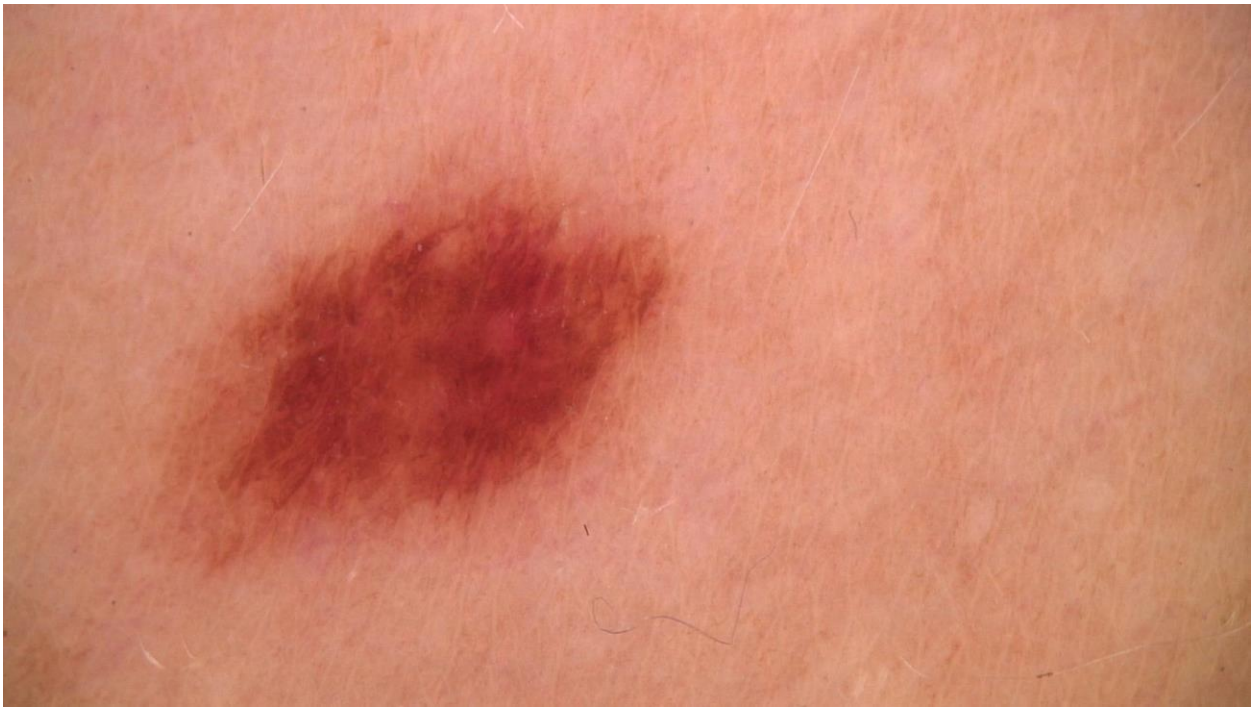
RGB Input Image:



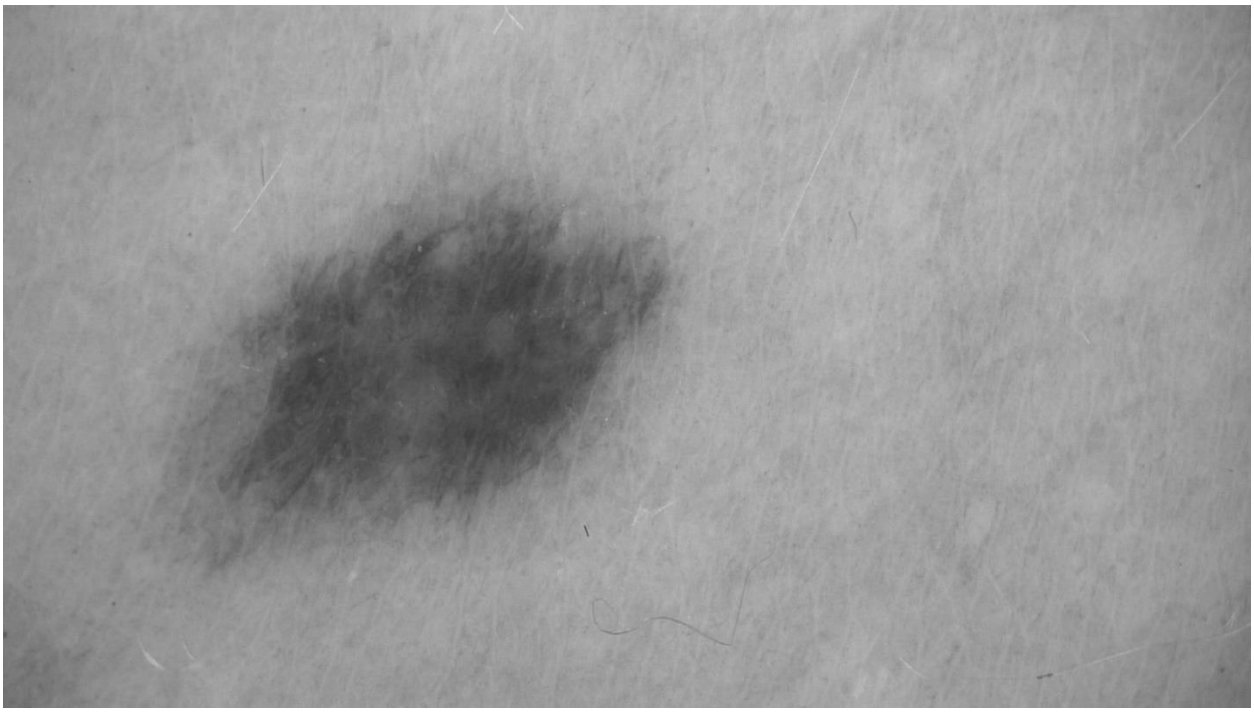
Grayscale Output Image:

Example 4: ISIC_0229266

RGB Input Image:



Grayscale Output Image:



Example 5: ISIC_0228452

RGB Input Image:

Grayscale Output Image