UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HANOI

Advanced Programming for HPC
Part A: Final Project Report

Pham Phan Bach
M21.ICT.002

January, 2024

I. Project Summary & Overview

In this final project, the main objective is to implement and compare 2 versions of the Kuwahara filter. One will be using the power of the CPU, others will utilize the advantageous numbers of cores in the GPU.
In the image processing field, to smoothen an image most other filters implement linear low-pass filters, this makes them efficient in removing noise and blurring the edges. However if the edges information are required to be preserved during the smoothen process, the Kuwahara filter is preferable options.

Suppose we have a pixel Φ(x,y), and we take a square window size of 2ω + 1 centered around a point (x,y) in the image. Then the square can be divided into four smaller square regions, A, B, C, and D. The pixels at borders between two regions belong to both regions so there is a slight overlap.

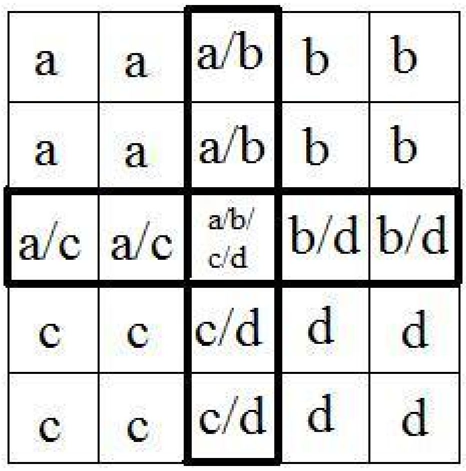

Figure 1: Kuwahara filters operator

Both versions of the implementation will follow the same structure, which contain 5 main functions:

1. Calculate image's HSV color values
2. Calculate the values of 4 windows for each pixel using the extracted V values
3. Calculate 4 standard deviations from each windows of each pixel

4. Calculate the new image pixel values
5. It is important to add proper padding to the border before generate the output image for a clamp to edge behavior

While having the same structure, there are still some differences in implementation within both version:

- In CPU version, to generate the output image, the programs have to iterate each pixel to calculate the new pixel value
- In GPU version, the calculation is done in parallel using the main GPU memory

II. Result

The project is ran on my trusty old laptop:

- CPU: Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz
- GPU: GeForce GTX 1050 Mobile

Below the result of both implementation compare the original image:

Figure 2: Original Image (600x388)

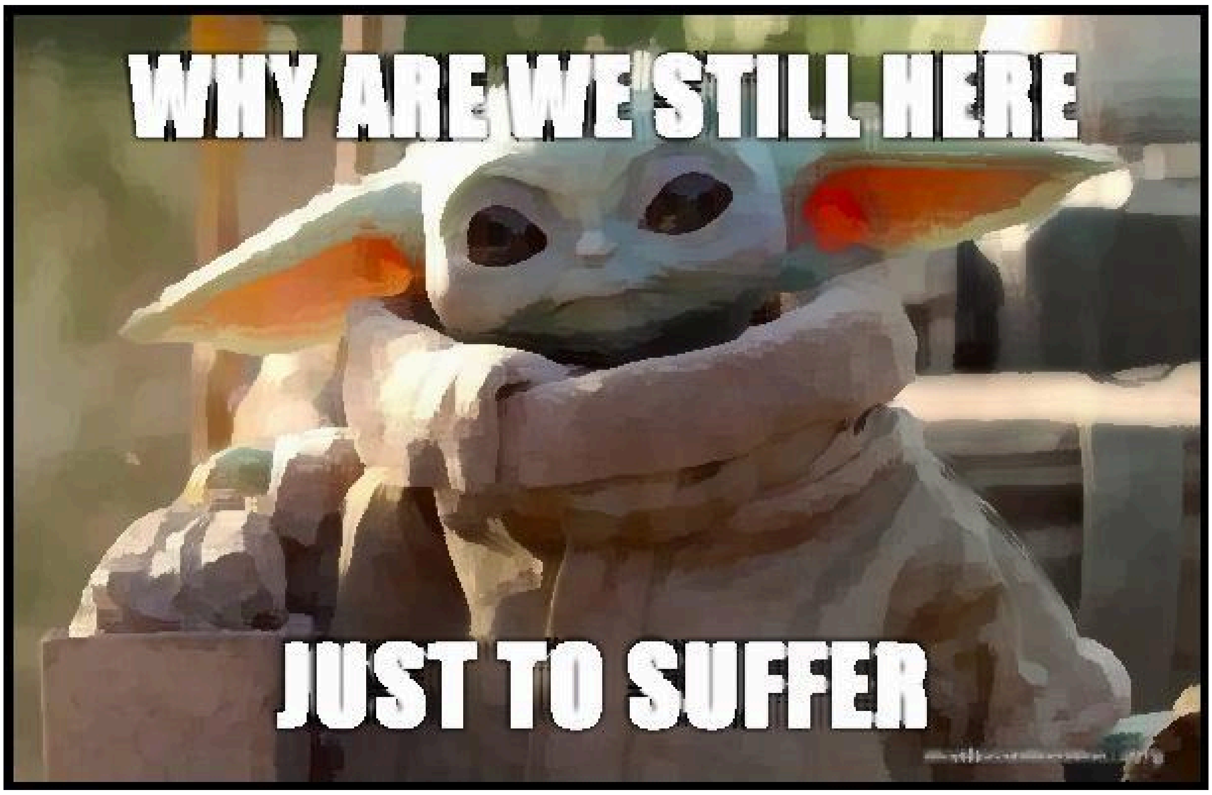

Figure 3: Result image generated by CPU version

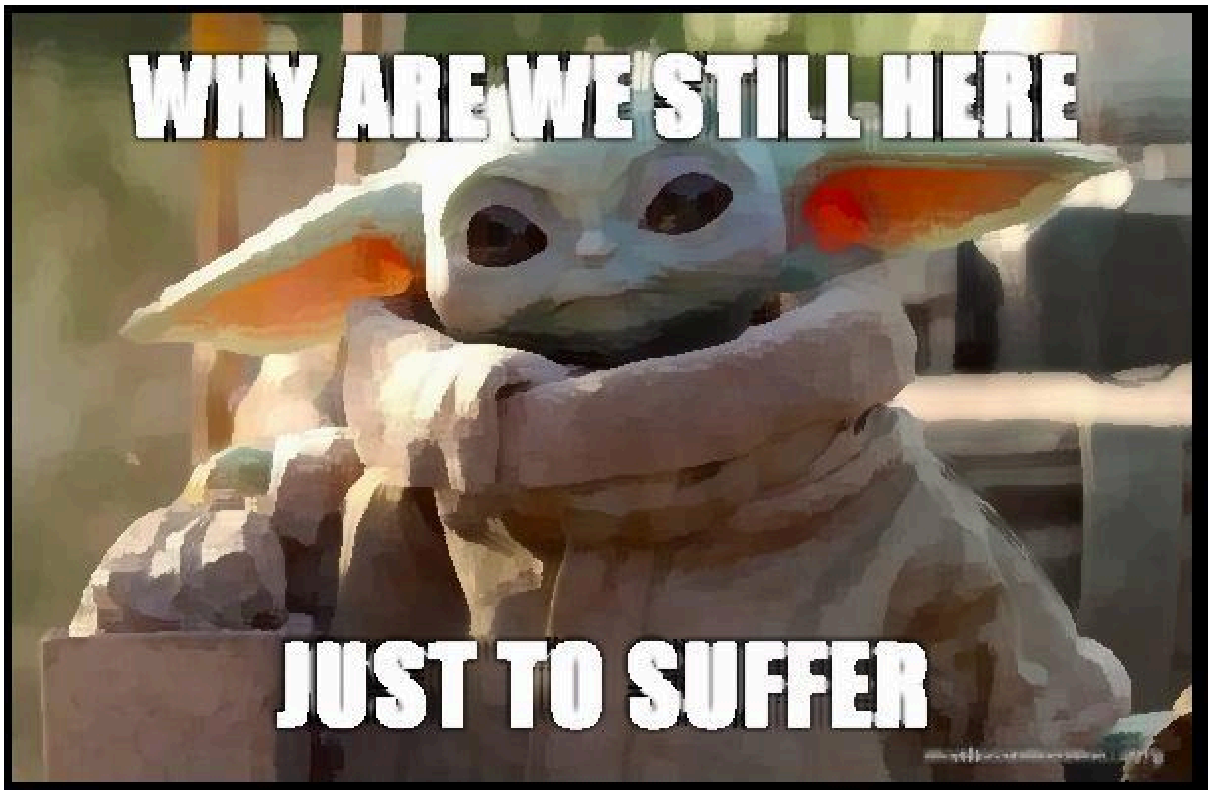

Figure 4: Result image generated by GPU version

Implementation	Time ran (second)
CPU	~32.56
GPU	~1.33

Figure 5: Performance comparison between CPU and GPU version

From result, the GPU version is much faster than the CPU version which is within expectation.