Word Count (excluding code and headers) [ ]

Module Code: **COMP10065**

Unsharp Mask Report

University of the west of scotland

GPGPU

B00308927

2018

Table of Contents

[Introduction 2](#_Toc531389461)

[Design approach 2](#_Toc531389462)

[Benchmarking 2](#_Toc531389463)

# Introduction

Introduce problem and the goal of the code -> To parallelise a serial image processor that applies an unsharp mask.

# Design approach

How was the parallelisation achieved? What level did I achieve? Naïve parallelisation -> + constants + local memory + shared memory. Why did you use and not use all of these? Why use MSVC++ Optimisation flags?

Why is the box blur applied three times? Would a gaussian blur improve the quality of the sharpened image? Did I use Gaussian blur? If not why not?

What code API did you use and why? What coding approach did you take?

Explain why you use two kernels and three buffers i.e. to separate the steps and be able to pass the two buffers into the first step and the three buffers into the second step.

Explain why you have used C or C++ OpenCL API commands. – Figure out which one I’ve used and justify it :P

An RGB datum can be packed in 32-bits or 24-bits. In terms of performance, does it matter?

Which high-level parameters of the GPGPU language/API can affect performance?

Big 8k image size is a power of 2 as opposed to the small one which is not, this should make the parallelised code make larger gains on the big image than the small one as a ratio. Perhaps make different resolution .ppm files to illustrate this effect?

Pre-compiled headers for faster build times? Is this something that could get marks?

# Benchmarking

Show a table of benchmarking, compare the serial against each iteration of parallelisation on a graph.

**Serial vs Naïve Parallelisation**

**Serial vs Naïve Parallelisation vs Naïve Parallelisation/w Constants**

**Serial vs Naïve Parallelisation vs Naïve Parallelisation/w Constants vs Naïve Parallelisation/w constants + local memory**

Show these with and without **MSVC++ Optimisation flags** and a variety of **Blur Radii** and a variety of **PPM File Resolutions.**

## Graphs