МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»

Лабораторная работа № 1
по курсу «Технологии параллельного программирования»

Обработка изображений на GPU. Фильтры

Выполнил: И.В. Сектименко
Группа: М8О-410Б-22
Преподаватели: А.Ю. Морозов,
Е.Е. Заяц

Москва, 2025

Условие

Кратко описывается задача:

1. Цель работы. Научиться использовать GPU для обработки изображений. Использование текстурной памяти и двухмерной сетки потоков.
2. Вариант 4. SSAA.
   Необходимо реализовать избыточную выборку сглаживания. Исходное изображение представляет собой «экранный буфер», на выходе должно быть сглаженное изображение, полученное уменьшением исходного.
   Входные данные. На первой строке задается путь к исходному изображению, на второй – путь к конечному изображению. На следующей строке – два числа *wn* и *hn* – размеры нового изображения. Гарантируется, что размеры исходного изображения соответственно кратны им ($w * h <= 4 * 10^8$, где w и h – размеры исходного изображения).
   Выходные данные. Необходимо в выходной файл на сначала записать размеры полученного изображения: w и h, потом байтовые значения каждого пикселя изображения.

Программное и аппаратное обеспечение

Характеристики:

— графического процессора:

compute capability:	7.5
графическая память:	15828320256
разделяемая память на блок:	49152
константная память:	65536
количество регистров на блок:	65536
максимальное количество блоков:	(2147483647, 65535, 65535)
максимальное количество нитей на блок:	(1024, 1024, 64)
количество мультипроцессоров:	40

— процессора: Intel(R) Xeon(R) CPU @ 2.00GHz;

— оперативной памяти: 12Гб общей памяти;

— жесткого диска: 256 Гб общей памяти.

Во время выполнения работы использовалась IDE Google Colab. В нее встроена ОС Ubuntu 22.04.4 LTS с видеокартой NVIDIA-SMI 550.54.15.

Метод решения

Сначала необходимо определить размер блока путем деления изначальной длины и ширины изображения на конечную. Далее следует пройтись по блокам и вычислить усредненное значение для каждого блока. Полученный результат записать в новый массив.

Описание программы

Программа состоит из 1 файла.

Сначала считывается вся необходимая информация из стандартного потока ввода. Потом открывается файл и считывается информация об изображении (его размеры и байтовые значения пикселей).
Создается текстура и настраивается ее интерфейс. Часть переменных, значение которых необходимо знать при решении задачи на GPU, перекопируется в регистровую память GPU.
Внутри функции kernel осуществляется проход по изображению окном, внутри которого будут усредняться значения пикселей, со смещением на суммарное количество выделенных потоков. Так получается, что каждый поток обрабатывает пиксель исходного изображения.
Вычисленный результат записывается в массив пикселей выходного изображения.
После того, как все потоки отработали, результат перекопируется в память на CPU. Полученные значения записывается в файл согласно формату вывода.
Чистится память как на CPU, так и на GPU.

Результаты

Ниже приведена сравнительная таблица. В строках – программа выполняется ядрами с разными конфигурациями (двумерная сетка потоков) или на центральном процессоре, в столбцах – разные объемы данных (при этом размер окна, по которому будем усреднять изображение, – 2x2).

	2560 x 1440	1280 x 720	640 x 360
dim3(1,1), dim3(4, 4)	117,39 мс	28,01 мс	6,77 мс
dim3(4, 4), dim3(8, 8)	2,04 мс	0,63 мс	0,6 мс
dim3(8, 8), dim3(16, 16)	0,32 мс	0,16 мс	0,16 мс
dim3(16, 16), dim3(32, 32)	0,26 мс	0,17 мс	0,19 мс
dim3(32, 32), dim3(32, 32)	0,3 мс	0,21 мс	0,2 мс
CPU	28,9 мс	8,14 мс	1,77 мс

Далее показан пример работы алгоритма. Изображение размером 640x360 (рисунок 1) было уменьшено в 2 раза (рисунок 2). Если увеличить полученное после применения алгоритма изображение, то можно увидеть, что оно стало размытым и менее четким (рисунок 3). Особенно видно на буквах (вотермарка фотграфии) справа внизу.

Рисунок 1 – Исходное изображение

Рисунок 2 – Изображение после применения алгоритма

Рисунок 3 – Увеличенная версия результирующего изображения

Выводы

Данный алгоритм можно применять непосредственно для уменьшения изображения и подгонку его под определенный размер (можно растянуть изображение, сделать его вертикальным и так далее). Кроме того, SSAA позволяет сделать изображение более размытым. Поэтому его можно применять, например, для размытия границ.

Запрограммировать данный алгоритм было достаточно легко. Самым сложным было найти ошибки, связанные с опечатками. Например, вместо переменной x случайно использовать переменную y.

Распараллеливание алгоритма привело к хорошим результатам и значительно ускорило программу. Однако важно помнить, что на выделение потоков и ресурсов требуется время, поэтому конфигурации с малым количеством потов не дают значительного ускорения.

В зависимости от размера данных для максимального ускорения требуются разные конфигурации – разное количество потоков. Чем меньше объем данных, тем меньше потоков надо выделить для максимального ускорения.