МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

Лабораторная работа №3
по курсу «Программирование графических процессоров»

Классификация и кластеризация изображений на GPU.

Выполнил: А.С. Федоров
Группа: 8О-407Б
Преподаватели: К.Г. Крашенинников,
А.Ю. Морозов

Москва, 2022

Условие

Цель работы. Научиться использовать GPU для классификации и кластеризации изображений. Использование константной памяти и одномерной сетки потоков.

Вариант задания. 3. Метод минимального расстояния.

Для некоторого пикселя p, номер класса jc определяется следующим образом:

$$jc = \arg max_j \left[-(p - avg_j)^T * (p - avg_j) \right]$$

Программное и аппаратное обеспечение

Графический процессор

Название: NVIDIA GeForce GTX 1050
Compute capability: 6.1
Объем графической памяти: 2147352576 байтов
Объем разделяемой памяти на блок: 49152 байтов
Объем регистров на блок: 65536
Размер варпа: 32
Максимальное количество потоков на блок: (1024, 1024, 64)
Максимальное число блоков: (2147483647, 65535, 65535)
Объем постоянной памяти: 65536 байтов
Число мультипроцессоров: 5

Процессор

Название: i5-8250U
Базовая тактовая частота процессора: 1,60 ГГц
Количество ядер: 4
Количество потоков: 8
Кеш L1: 256 Кб
Кеш L2: 1 Мб
Кеш L3: 6 Мб

Оперативная память

Тип: DDR4
Объем: 11.9 Гб
Частота: 2400 МГц

Програмное обеспечение

ОС: WSL2 (Windows 11)
IDE: Microsoft Visual Studio 2022 (аддон NVIDIA Nsight)
Компилятор: nvcc

Метод решения

Перед обработкой данных на GPU для каждого класса подчитываются значения выборочных средних для всех трех цветовых каналов. Затем эти значения копируются в константную память. Так как по условию задачи классов всего может быть 32, то массив в константной памяти тоже длины 32. Если классов оказалось меньше, то в ненужных ячейках лежит мусор и туда обращений не производится.

На GPU для каждого пикселя происходит последовательный подсчет дистанций для каждого выборочного среднего классов. Из всех значений выбирается минимальное расстояние и соответствующий ему класс. Значение этого класса записывается в альфа-канал исходного изображения. Дополнительная память для результата не выделяется.

Описание программы

Программа считывает пути для входного и выходного файлов. Затем считывает выборки по классам одновременно подсчитывая значения выборочных средних. Так как нигде, кроме этого этапа, выборки классов не нужны, они не сохраняются в оперативной памяти. После этого данные для классификации и значения выборочных средних копируются в память GPU (данные изображения в глобальную, а выборочные средние в константную). Далее в каждом потоке ядра, согласно формуле задания, высчитывается наиболее подходящий класс для классификации пикселя. Делается это в цикле, для каждого класса высчитывается расстояние до его выборочного среднего. После завершения вычислений на GPU все данные сохраняются в выходной файл.

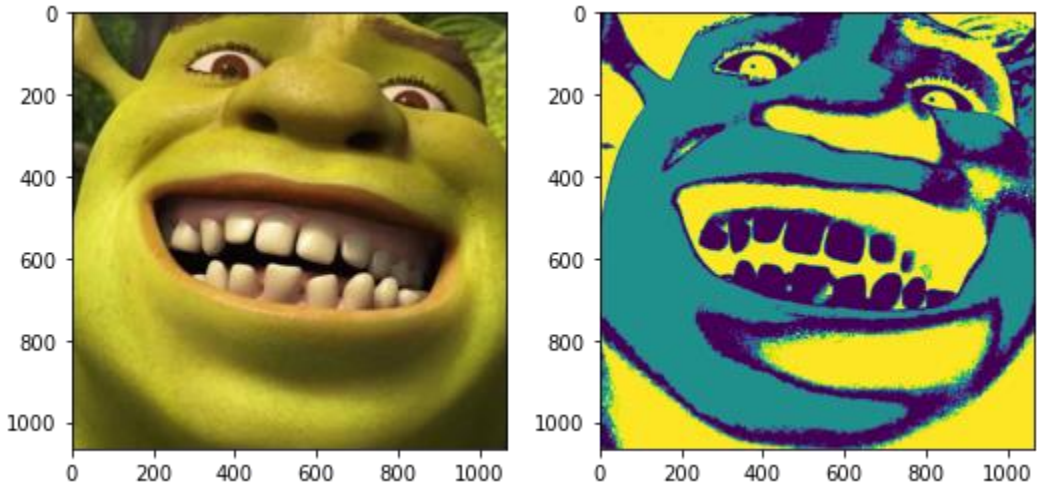
Код ядра:

```
__constant__ float dev_avg[32][3];

__global__ void kernel(uchar4* data, int nc, int w, int h) {
    int idx = blockIdx.x * blockDim.x + threadIdx.x;

    while (idx < w*h) {
        char prediction;
        float best_distance = -INFINITY;
        for (char i = 0; i < nc; ++i){
            float current_distance = -(data[idx].x-
dev_avg[i][0])*(data[idx].x-dev_avg[i][0])
                                     -(data[idx].y-
dev_avg[i][1])*(data[idx].y-dev_avg[i][1])
                                     -(data[idx].z-
dev_avg[i][2])*(data[idx].z-dev_avg[i][2]);
            if (current_distance > best_distance){
                prediction = i;
                best_distance = current_distance;
            }
        }
        data[idx].w = prediction;
        idx += gridDim.x * blockDim.x;
    }
}
```

Результаты (в миллисекундах)

	Размер теста (по вертикали и горизонтали)						
	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}
GPU(CUDA) < 1, 32>	3.902	12.570	46.047	181.347	634.109	2299.780	9013.668
GPU(CUDA) < 32, 32>	1.636	1.630	2.793	8.233	25.940	98.072	308.638
GPU(CUDA) < 64, 64 >	1.420	1.711	2.155	4.428	11.070	41.199	129.679
GPU(CUDA) < 128, 128 >	1.425	1.607	2.545	3.742	11.275	38.736	139.311
GPU(CUDA) < 256, 256>	1.456	**1.393**	**1.984**	4.198	10.818	37.569	118.073
GPU(CUDA) < 512, 512>	1.521	1.670	2.192	3.734	**10.480**	37.737	**112.175**
GPU(CUDA) < 1024, 1024>	**1.417**	1.641	1.996	**3.717**	10.726	**36.829**	116.402
CPU	18.018	69.057	266.350	1060.142	4220.603	16888.577	69671.771

Визуальное представление результата работы фильтра:
Размер изображения: 1065 на 1065
Число классов: 3
Оригинал: Результат:



На изображении видно, что классификация произошла, в какой-то степени, по яркости пикселей. Желтым выделены темные, синим средней и фиолетовым самой сильной яркости.

Выводы

Данный алгоритм хорошо подойдет для выделения пикселей схожего цвета в редакторах изображений. Возможно это можно использовать при реализации инструмента «волшебная палочка». Также этот алгоритм хорошо подойдет для отрезания монотонного фона на видео, то есть использовать для обработки съемок на хромакее.
Алгоритм программируется несложно. Однако, следует отметить, что часть расчетов все-таки выполняется на CPU. Перенос этих вычислений сильно бы усложнил программу, что не было задумано в рамках лабораторной работы. Из результатов тестирования, в принципе, понятно, что наибольшее число блоков и потоков выигрывает на больших объемах данных. Но если данные достаточно малы, чтобы каждому потоку досталось для обработки только по одному пикселю, то дальнейшее наращивание блоков и потоков не дает никакого прироста к эффективности.