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: Р.Р. Гаптулхаков
Группа: 8О-408Б
Преподаватели: К.Г. Крашенинников,
А.Ю. Морозов

Москва, 2022

Условие

Научиться использовать GPU для классификации и кластеризации изображений. Использование константной памяти и одномерной сетки потоков. Формат изображений соответствует формату описанному в лабораторной работе 2. Во всех вариантах, в результирующем изображении, на месте альфа-канала должен быть записан номер класса(кластера) к которому был отнесен соответствующий пиксель. Если пиксель можно отнести к нескольким классам, то выбирается класс с наименьшим номером.

Вариант 3. Метод минимального расстояния.

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

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

Входные данные. На первой строке задается путь к исходному изображению, на второй, путь к конечному изображению. На следующей строке, число nc – количество классов. Далее идут nc строчек описывающих каждый класс. В начале j-ой строки задается число np_j – количество пикселей в выборке, за ним следуют np_j пар чисел – координаты пикселей выборки. $nc \le 32$, $np_j \le 2^{19}$, $w*h \le 4 * 10^8$.

Оценка вектора средних и ковариационной матрицы:

$$avg_j = \frac{1}{np_j}\sum_{i=1}^{np_j} ps_i^j$$

$$cov_j = \frac{1}{np_j - 1}\sum_{i=1}^{np_j} (ps_i^j - avg_j) * (ps_i^j - avg_j)^T$$

где $ps_i^j = (r_i^j\ g_i^j\ b_i^j)^T$ -- i-ый пиксель из j-ой выборки.

Выходные данные. Бинарный файл

Программное и аппаратное обеспечение

Таблица 1 — Характеристики графического процессора

Compute capability	2.1
Name	GeForce GT 545
Total Global Memory	3150381056

Shared memory per block	49152
Registers per block	32768
Warp size	32
Max threads per block	(1024, 1024, 64)
Max block	(65535, 65535, 65535)
Total constant memory	65536
Multiprocessors count	3

Таблица 2 — Используемое ПО

Operating system	Windows 11 + ssh Ubuntu 16.04.6 LTS
IDE	Visual Studio Code
Compiler	nvcc

Таблица 3 — Характеристики процессора

Name	Intel(R) Core(TM) i7-3770 CPU
Architecture	x86_64
CPU(s)	8
Thread(s) per core	2
Core(s) per socket	4
CPU max MHz	3900
CPU min MHz	1600
CPU MHz	1800
L1d cache	32K
L1i cache	32K
L2 cache	256K
L3 cache	8192K

Таблица 4 — Оперативная память

Size	16 G

Таблица 5 — Постоянная память

Size	1000 G

Метод решения

С начала считаем среднее значение rgb для каждого класса. Затем классифицируем методом минимального расстояния.

Описание программы

Для оптимального выполнения программы вычислим среднее значение на CPU, затем поместим в константную память.

```
__constant__ double AVG[32][3];
    // Инициализация avg
    double avg[32][3];
    for (int i = 0; i < 32; ++i){
        avg[i][0] = 0.0;
        avg[i][1] = 0.0;
        avg[i][2] = 0.0;
    }

    // Вычисление avg
    for(int i = 0; i < num_classes; ++i){
        for(int j = 0; j < classes[i].size(); ++j){
            int x = classes[i][j].x;
            int y = classes[i][j].y;
            uchar4 p = data[y * w + x];
            avg[i][0] += p.x;
            avg[i][1] += p.y;
            avg[i][2] += p.z;
        }
        avg[i][0] /= classes[i].size();
        avg[i][1] /= classes[i].size();
        avg[i][2] /= classes[i].size();
    }
```

Классификация каждого пикселя происходит на GPU в отдельном потоке.

```
__global__ void kernel(uchar4* dev_data, int w, int h, int
num_classes) {
    int idx = blockIdx.x * blockDim.x + threadIdx.x;
    int offset = blockDim.x * gridDim.x;
    while (idx < w * h){
        double4 tmp;
        double res, maxval;
        uchar4 p = dev_data[idx];
        for(int i = 0; i < num_classes; ++i){
            tmp.x = p.x - AVG[i][0];
            tmp.y = p.y - AVG[i][1];
            tmp.z = p.z - AVG[i][2];
            res = tmp.x * tmp.x + tmp.y * tmp.y + tmp.z *
tmp.z;
            res = -res;
            if(i == 0){
                maxval = res;
                dev_data[idx].w = 0;
            }
            if(res > maxval){
                maxval = res;
                dev_data[idx].w =  i;
            }
        }
        idx += offset;
    }
}
```

Результаты

Входная картинка:

Выходная картинка:

Таблица 6 — Результаты

Количество потоков	**710x648**
<<<16, 16>>>	4.828928
<<<32, 32>>>	1.935808
<<<32, 16>>>	3.553344
<<<16, 32>>>	2.596832
<<<8, 32>>>	5.015808
<<<128, 32>>>	1.524896
<<<128, 64>>>	1.380512
CPU	371.123

Выводы

Реализованная программа широко применяется для различных фильтров в обработке изображений. Например, в военных целях, для обнаружения замаскированного противника. Увеличение количества потоков дает значимые результаты ускорения программы. Особенно заметно по сравнению с вычислениями на CPU.