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: И. П. Моисеенков
Группа: М8О-408Б-19
Преподаватель: А.Ю. Морозов

Москва, 2022

Условие

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

Формат изображений: изображение является бинарным файлом со следующей структурой:

- в первых восьми байтах записывается размер изображений
- далее построчно значения пикселей r, g, b, a.

Вариант 2. Метод расстояния Махалнобиса:

Входные данные. На первой строке задается путь к исходному изображению, на второй - путь к конечному изображению. На следующей строке число - количество классов. Далее идут строки, описывающие каждый класс (известные точки, принадлежащие данному классу)

Выходные данные. Необходимо записать в выходной файл изображение. На месте альфа-канала должен быть записан номер класса (кластера), к которому был отнесен пиксель.

Программное и аппаратное обеспечение

В качестве графического процессора использую видеокарту Nvidia GeForce GT 545, установленную на сервере преподавателя.

```
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
```

В качестве редактора кода использовался Visual Studio Code.

Метод решения

Будем для каждого пикселя вычислять расстояния Махалнобиса для каждого класса. Итоговый класс - класс с наибольшим значением расстояния:

$$jc = arg\,max_j\left[-(p - avg_j)^T * cov_j^{-1} * (p - avg_j)\right]$$

Для использования этой формулы необходимо заранее вычесть средние значения и обратную матрицу ковариации, используя точки с известными классами. На графическом процессоре будем попиксельно проходить и определять классы. Используем одномерную сетку, т.к. нам не важна информация о соседних пикселях - мы рассматриваем всегда один конкретный пиксель.

Описание программы

Предварительные вычисления средних значений и матрицы ковариации будем производить на CPU. Результаты будем хранить в константной памяти. Для этого используем функцию cudaMemcpyToSymbol.

В функции kernel будем проходиться по картинке и вычислять расстояния Махалнобиса для каждого класса. Для вычисления будем использовать предпосчитанные данные из константной памяти.

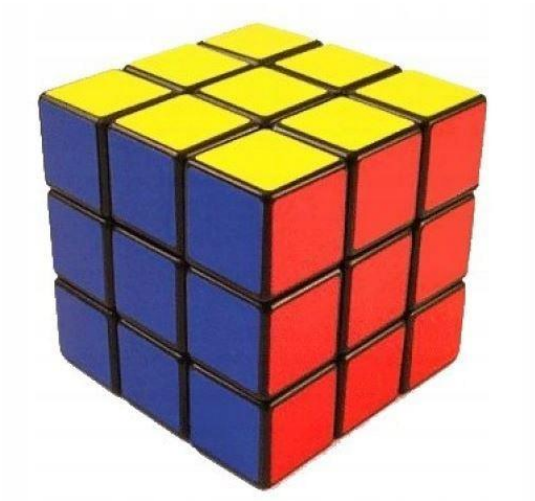
Результаты

Рассмотрим время работы программы на различных тестах при различных размерах сетки (и без использования графического процессора вообще). Будем замерять непосредственно время работы алгоритма классификации (предподсчет замерять не будем). В качестве теста будем решать задачу классификации пикселей изображений различных размеров. Количество классов = 5. Все замеры произведены на сервере преподавателя. Результаты приведены в таблице ниже.

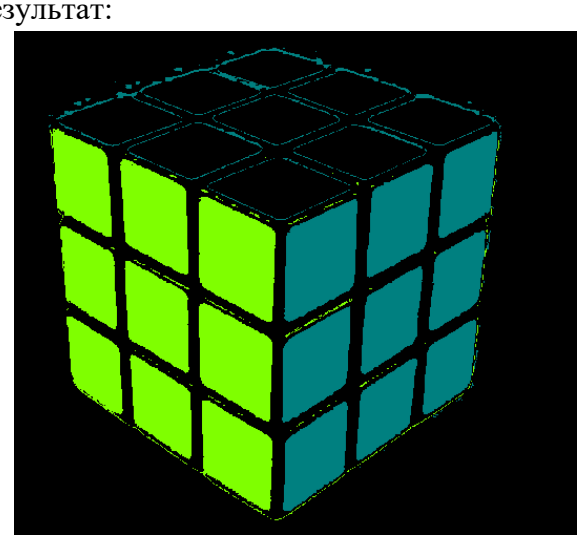
Размер сетки ядра	1к x 1к пикселей, мс	5к x 3к пикселей, мс	6к x 5к пикселей, мс
CPU	2093.6	30957.7	62845.6
<<<1, 32>>>	73.5	1428.8	2467.6
<<<32, 32>>>	6.4	123.3	216.6
<<<128, 128>>>	5.1	98.7	170.5
<<<256, 256>>>	5.1	98.3	169.8
<<<512, 512>>>	5.0	97.7	168.7
<<<1024, 1024>>>	5.5	99.1	171.0

Оптимальный размер сетки - <<<128, 128>>>.

Приведу пример работы программы. Имеется исходное изображение кубика. Попробуем классифицировать его пиксели на 3 класса (по 3 сторонам соответственно). В качестве входных данных выберем по 4 точки из каждого класса.



Полученный результат:



Интересно, что мы смогли отличить черные границы квадратов от самих квадратов!

Выводы

В этой лабораторной работе я потренировался в использовании константной памяти. Эта память характеризуется относительно высокой скоростью доступа, однако с графического процессора в нее нельзя ничего записывать. Но для решения текущей задачи константная память подходит идеально.

Дополнительно я познакомился с математическими алгоритмами классификации и кластеризации изображений и реализовал формулу для одного из них. Возникли небольшие трудности с отладкой - было много громоздких вычислений, в которых было легко запутаться.

Было проведено сравнение времени работы на графическом процессоре и без него. Оказалось, что распараллеливание обработки изображения на порядок ускоряет работу алгоритма.