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: Семин А. В.
Группа: 8О-406Б
Преподаватели: К.Г. Крашенинников,
А.Ю. Морозов

Москва, 2023

Условие

1. **Цель работы.** Научиться использовать GPU для классификации и кластеризации изображений. Использование константной памяти и одномерной сетки потоков.
2. **Вариант 1. Метод максимального правдоподобия.**
   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.
   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, $np_j \leq 32 \leq 2$, $w*h \leq 4 * 10^8$.

Программное и аппаратное обеспечение

Графический процессор (GeForce GTX 1650 Ti)

1. Количество потоковых процессоров: 1024
2. Частота ядра: 1350 МГц
3. Частота в режиме Boost: 1485 МГц
4. Количество транзисторов: 6,600 млн
5. Тип памяти: DDR6
6. Видеопамять: 4096 МБ
7. Частота памяти: 12000 МГц

Процессор AMD Ryzen 7 4800H

1. ядра: 8
2. потоки: 16
3. частота: 2.9 ГГц
4. максимальная частота: 4.2 ГГц
5. кэш 1 уровня: 64 КБ (на ядро)
6. кэш 2 уровня 512 КБ (на ядро)
7. кэш 3 уровня: 8 МБ (общий)

16 ГБ ОЗУ и 512 ГБ SSD.

OS – Windows 11Домашняя, WSL, IDE – VS Code, Compiler - nvcc, g++.

Метод решения

Считываем из стандартного потока ввода название входного и выходного файлов, количество классов для классификации и пары позиций пикселей для создания начальной выборки каждого класса. Считываем из файла его размеры и бинарные данные. Далее вычисляем значения для вектора средних avg, ковариационной матрицы cov и обратной ковариационной матрицы cov^{-1}, определителя ковариационной матрицы detCov. Данные предвычисления проводим на CPU. После копируем все вычисленные значения в соответствующие объекты в памяти GPU. Затем вызываем ядро (с одномерной сеткой потоков), которое для каждого пикселя вычисляет соответствующий ему класс. Вычисление класса производится в отдельной функции на GPU. Результирующий номер класса определяется как номер j класса, для которого получается максимальное значение функции дискриминанта D_f.

После вычислений на ядре копируем данные на CPU, записываем в файл и очищаем память.

Описание программы

Программа состоит из одного файла, в котором находятся функция ядра и main. Помимо них есть функции __host__, которые на CPU выполняют все необходимые предвычисления, и функции __device__, которые совершают вычисления на GPU для классификации каждого пикселя. В главной функции осуществляется считывание и запись данных, вызов функций для предвычислений, вызов ядра, копирование данных. Вычисления на GPU производятся с использованием константной памяти: необходимые объекты создаются как __constant__ вне метода main.

Сигнатура функции ядра:

```
__global__ void kernel(uchar4 *data, int img_size, int classCount)
```

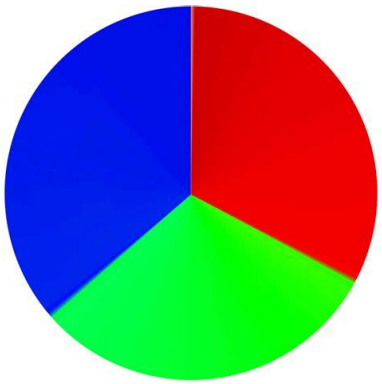
Результаты

Время выведено в ***миллисекундах***.

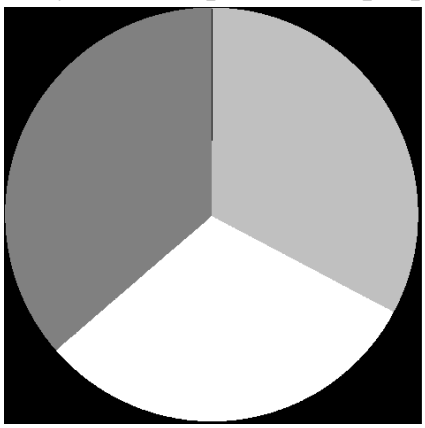
Конфигурация	Размер тестового файла				
	$80 * 80$	$250 * 250$	$900 * 900$	$2200 * 2200$	$9000 * 9000$
CPU	461689	935900	94160	$1.45683 * 10^6$	$1.40116 * 10^6$
<<<(16, 16) >>>	12.11865	19.80620	5.125728	29.573120	29.447104
<<<(64, 64) >>>	0.928672	1.679584	0.446048	2.478368	2.407104
<<<(128, 128) >>>	**0.826656**	**1.607680**	0.212800	2.500608	2.496512
<<<(256, 256) >>>	0.901120	1.751776	0.311200	**2.406400**	**2.347968**
<<<(512, 512) >>>	0.985536	1.657024	**0.188416**	2.464352	2.393824
<<<(1024,1024) >>>	1.003200	1.716224	0.305152	2.502464	2.478144

Примеры работы

Исходная картинка:



Результат обработки при разделении на 4 класса с ручной разметкой:



Размер исходной картинки: *474 x 474* пикселей.

Размер обработанной картинки: *474 x 474* пикселей.

Выводы

Метод максимального правдоподобия – очень распространенный алгоритм, который много применяется в машинном обучении, а также в разных научных сферах.

Как видно из замеров вычислений в таблице результатов по данной лабораторной работе, этот метод работает на CPU довольно медленно, однако при запуске этого метода на GPU с одномерной сеткой потоков время работы ускоряется вплоть до нескольких миллисекунд

В данной работе основной сложностью было разобраться и корректно выполнить все необходимые предвычисления для матриц и векторов, необходимых для корректной работы метода максимального правдоподобия.