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.А. Почечура*

Группа: *8О-406Б*

Преподаватели: К.Г. Крашенинников,

А.Ю. Морозов

Москва, 2023

**Условие**

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

**Вариант 2. Метод расстояния Махаланобиса.**

**Формат изображений.** 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.

**В вариантах 1-4, формат входных данных одинаковый**. 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 – количество пикселей в выборке, за ним следуют пар чисел – координаты пикселей выборки.

Оценка вектора средних и ковариационной матрицы:

где - i-ый пиксель из j-ой выборки.

**Входные данные.** Для некоторого пикселя p, номер класса jc определяется следующим образом:

**Программное и аппаратное обеспечение**

**Графический процессор:**

Compute capability: 7.5

Name: Tesla T4

Total Global Memory: 15835398144

Shared memory per block: 49152

Registers per block: 65536

Warp size: 32

Max threads per block: (1024, 1024, 64)

Max block: (2147483647, 65535, 65535)

Total constant memory: 65536

Multiprocessors count: 40

**Процессор:**

vendor\_id : GenuineIntel

cpu family: 6

model: 85

model name: Intel(R) Xeon(R) CPU @ 2.00GHz

stepping: 3

microcode : 0xffffffff

cpu MHz: 2000.184

cache size : 39424 KB

physical id: 0

siblings: 2

core id: 0

cpu cores : 1

apicid: 0

initial apicid: 0

fpu: yes

fpu\_exception: yes

cpuid level: 13

wp : yes

bogomips : 4000.36

clflush size: 64

cache\_alignment : 64

address sizes: 46 bits physical, 48 bits virtual

**Оперативная память и жёсткий диск:**

Memory:

description: System memory

physical id: 0

size: 13GiB

Mem**:** 12982.6 MiB

Storage**:** 55 GiB

**Программное обеспечение:**

*Google Colab*

Операционная система: Ubuntu 22.04.2 LTS

Оболочка: python3

inxi: 3.3.13

**Метод решения**

В данном задании нужно реализовать метод Махаланобиса. Для этого нам для начала потребуется ввести входные данные и для каждого обозначенного класса посчитать следующие значения: мат. ожидание, матрицу ковариаций и обратную матрицу. Данные вычисления являются своего рода “подготовительными”, поэтому их мы производим на ЦПУ. Затем проходимся уже на ГПУ по всем пикселям и считаем для каждого в отдельности степень принадлежности к каждому из классов. Из всех полученных значений нужно выбрать максимальное, а затем классифицировать текущий пиксель кластеру, номер которого соответствует этому самому максимальному значению. Таким образом, мы получим готовую кластеризацию всех пикселей изображения.

**Описание программы**

В начале преобразуем данное на входе изображение в бинарный файл. Затем считаем информацию о классах, заведём переменные *arg*, *cov* и *cov\_rev,* в которых будем хранить значения мат. ожидания, матрицу ковариаций и обратную матрицу ковариаций для каждого кластера. Ввод информации о классах и подсчёт всех необходимых значений будет производиться в функциях *get\_arg\_cov* и *reverse\_cov* (имеют тип *\_\_host\_\_*). Посчитанные значения передадим на ГПУ, используя константную память. Таким образом, копиями *arg* и *cov\_rev* на ГПУ станут *arg1* и *cov\_rev1* (имеют тип *\_\_constant\_\_*)*.* Далее вызываем ядро *kernel,* в котором мы проходимся по каждому пикселю и задаём им номер класса с помощью функции *mahalanobis* (тип *\_\_device\_\_*)*.* В этой функции с помощью формул определяется, принадлежность к какому кластеру текущего пикселя даёт самое максимальное значение. После нахождения максимума, в альфа-канал записывается номер класса и процесс повторяется для всех остальных пикселей. Затем данные передаются с ГПУ обратно на ЦПУ и с помощью скрипта выводится результат (для каждого пикселя значения *r, g, b* определяются из таблички в соответствии с номером его класса).

**Результаты**

Продемонстрируем скорость работы программы в зависимости от количества потоков (количество классов во всех тестах равно 8, количество пикселей в классах равно 32):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Количество блоков и потоков в блоке | Размер входных данных | w \* h = 16 | w \* h = 1000 | w \* h = 1000.00 | w \* h = 1000.00.00 |
| << 1, 32 >> | | 2.377651987e-02 мс | 3.9757619864e-02 мс | 1.8371247895+00 мс | 2.0741821849+02 мс |
| << 4, 64 >> | | 4.194612787e-02 мс | 4.4832871894e-02 мс | 2.3758177981e-01 мс | 2.8414782732e+01 мс |
| << 32, 128 >> | | 3.046172618e-02 мс | 3.3471872417e-02 мс | 3.2748172845e-02 мс | 1.9846126784e+00 мс |
| << 128, 128 >> | | 2.5812939590e-02 мс | 2.9674127741e-02 мс | 3.4718271489e-02 мс | 1.1578137238e+00 мс |
| << 256, 256 >> | | 3.0571289199e-02 мс | 2.4415999651e-02 мс | 4.1748172738e-02 мс | 9.9381237841e-01 мс |
| << 512, 512 >> | | 2.4783758897e-02 мс | 2.2471879131e-02 мс | 3.9741728312e-02 мс | 9.7581239475e-01 мс |
| << 1024, 1024 >> | | 3.5781274879e-02 мс | 4.0047187293e-02 мс | 4.3648172891e-02 мс | 9.3758127859e-01 мс |

Сравним работу алгоритма на ГПУ и ЦПУ:

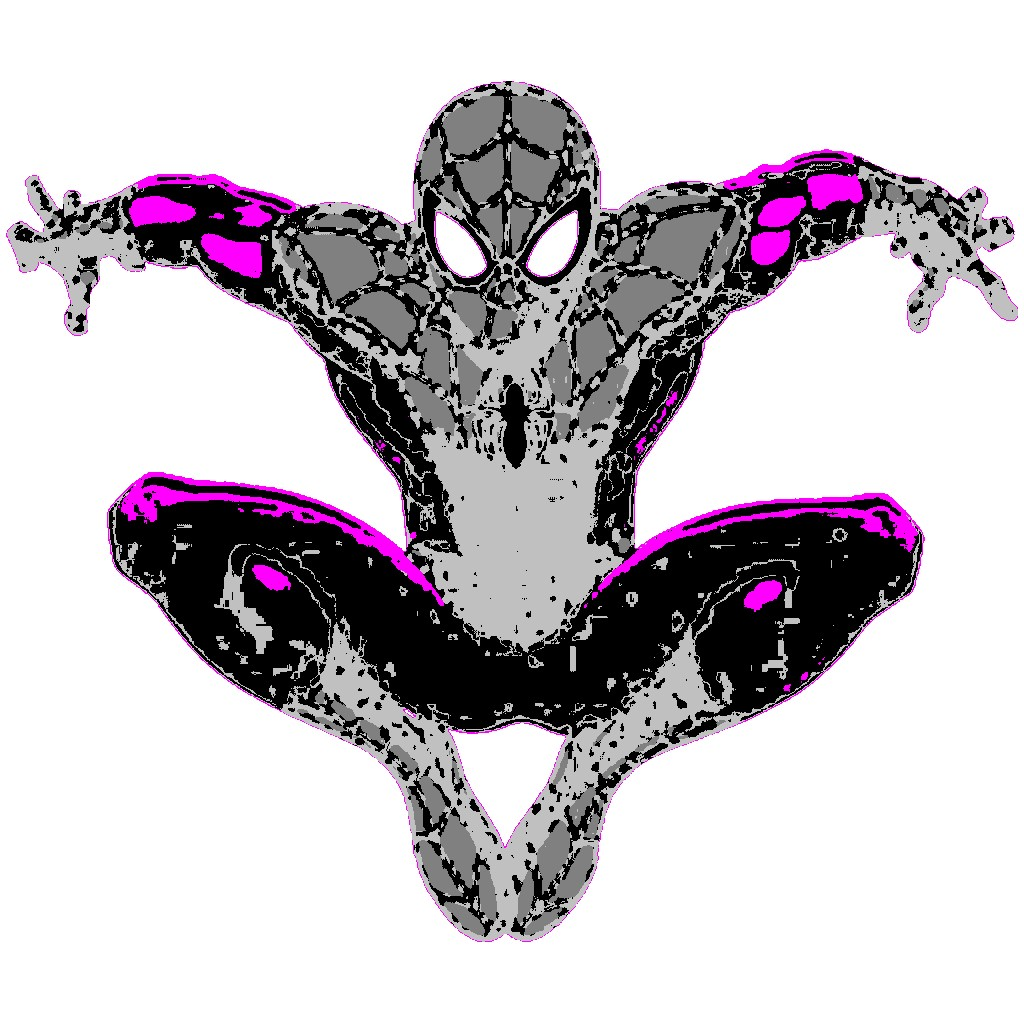
|  |  |  |  |
| --- | --- | --- | --- |
| Размер входных данных | Вид процессора | ГПУ  ( << 512, 512 >> ) | ЦПУ |
| w \* h = 15 | | 2.4783758897e-02 мс | 1.8900000000e-03 мс |
| w \* h = 1000 | | 2.2471879131e-02 мс | 3.5000000000e-03 мс |
| w \* h = 1000.00 | | 3.9741728312e-02 мс | 3.8900000000e-01 мс |
| w \* h = 1000.00.00 | | 9.7581239475e-01 мс | 10.3245000000e+01 мс |

**Демонстрация результата**

Входное изображение:



Выходное изображение:



После обработки удалось классифицировать почти все цвета, кроме тёмно-синего (он очень похож на чёрный, поэтому эти два цвета объединились в один кластер).

**Выводы**

В данной лабораторной работе я узнал, как можно кластеризовать пиксели изображения, используя метод расстояния Махаланобиса. В ходе работы я впервые взаимодействовал с константной памятью на ГПУ и научился записывать в неё информацию с ЦПУ. Для того, чтобы проиллюстрировать работу программы на примере, пришлось вручную задавать данные для классов. В этом деле мне очень помог Paint, ведь в нём легко можно узнать координаты любого пикселя.