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: М.С.Гаврилов

Группа: 8О-406Б

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 2. Метод расстояния Махаланобиса.

**Программное и аппаратное обеспечение**

|  |  |
| --- | --- |
| Характеристики графического процессора | |
| Compute capability: | 7.5 |
| Name: | NVIDIA GeForce GTX 1650 |
| Total Global Memory: | 4102029312 |
| 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: | 14 |

Процессор: Intel(R) Core(TM) i5-11260H @ 2.60GHz

Оперативная память: 7875 Мб

Накопитель: kimtigo SSD 256GB

OS: Linux Mint 21

IDE: Visual Studio Code 1.72.0

compiler: Cuda compilation tools, release 11.8, V11.8.89

**Метод решения**

Для хранения и обработки изображений используется класс image. Для работы с пикселями используется класс pixel. Этот класс — обертка над классом uchar4, реализующая методы записи и чтения из файла, а также вывода на экран. Пиксели изображений хранятся в одномерном массиве. При выполнении алгоритма классификации для уменьшения времени доступа используется текстурная память в двумерном массиве. Чтобы большие изображения умещались в текстурную память, использует двумерная текстура, а перед копированием изображение балансируется так, чтобы длины его сторон не сильно отличались. При работе ядра результат записывается в глобальную память и, по завершении, помещается на место изначальных пикселей изображения. В ядре каждый поток выполняет расчет расстояния соответствующих пикселей до каждого из классов методом Махаланобиса и определяет класс, к которому они принадлежат.

**Описание программы**

Программа состоит из одного файла, в котором реализованы классы pixel и image, написано ядро и функция main. Класс image, будучи унаследованным из л.р.2, помимо классификации, по-прежнему может выполнять сглаживание SSAA.

Основные методы двух перечисленных выше классов:

**Класс pixel**

Имеет один член — значение (uchar4)

|  |  |
| --- | --- |
| Метод | Описание |
| pixel(char red,char green, char blue, char alpha) | Конструктор, создает пиксель с заданными значениями каналов. |
| pixel read\_from\_file(std::ifstream& in) | Считывает пиксель в формате, описанном в задании к л.р. из файла и записывает его в себя. |
| pixel print\_to\_file(std::ofstream& out) | Записывает себя в файл в формате, описанном в задании к л.р. |
| void print() | Выводит в stderr значение каждого канала |
| void print\_avg() | Выводит stderr среднее значение цветовых каналов |

**Класс image**

Имеет 3 члена — Ширина, Высота и Указатель на массив, в котором лежат пиксели.

|  |  |
| --- | --- |
| Метод | Описание |
| image(int w\_,int h\_, pixel\* array) | Создает изображение размером w\*h с пикселями, хранящимися в arrray |
| image(std::string in\_adress) | Считывает изображение из файла в формате, описанном в задании |
| int SSAA(int new\_w, int new\_h, int block\_dim, int thread\_dim) | Выполняет сжатие SSAA на GPU. |
| void Classify(std::vector<std::vector<int>>> class\_coords) | Выполняет классификацию методом расстояний Махаланобиса на GPU. |
| void print\_to\_file(std::string filename) | Записывает изображение в файл в формате, описанном в задании |
| void print\_visual( | Выводит на экран усредненные значения цветовых каналов всех пикселей в сетке со сторонами w и h |
| void print() | Выводит в stderr значения всех каналов всех пикселей |

**Ядро**

|  |
| --- |
| \_\_global\_\_ void kernel(cudaTextureObject\_t pix, int size\_w, int size\_h , int coef\_w, int coef\_h,  pixel\* res) |

Перед запуском ядра выполняется расчет средних значений и обратных матриц ковариации для каждого класса и помещение оных в константную память. Так как не все изображения из поддерживаемого диапазона размеров могут уместиться в текстурной памяти, ядро последовательно применяется к частям изображения в 100 000 000 пикселей.

**Результаты**

1. Сравнение времени работы ядра с различными конфигурациями.

Количество примеров на класс в каждом тесте одинаково и равно 1000, так как обработка примеров выполняется вне ядра.

Размер файла 1 000 x 1 000 pix

|  |  |
| --- | --- |
| Размерность ядра | Время работы (мс) |
| <<<1,32>>> | 243.279 |
| <<<32,32>>> | 14.4843 |
| <<<32,1024>>> | 10.5372 |
| <<<64,64>>> | 10.655 |
| <<<512,512>>> | 9.5369 |
| <<<1024,1024>>> | 9.6189 |

Размер файла 10 000 x 10 000 pix

|  |  |
| --- | --- |
| Размерность ядра | Время работы (мс) |
| <<<32,1024>>> | 1499.98 |
| <<<32,64>>> | 1522.25 |
| <<<64,64>>> | 1476.44 |
| <<<512,512>>> | 1369.29 |
| <<<1024,1024>>> | 1340.36 |

Размер файла 100 000 x 100 000 pix

|  |  |
| --- | --- |
| Размерность ядра | Время работы (мс) |
| <<<64,64>>> |  |
| <<<512,512>>> |  |
| <<<1024,1024>>> |  |

1. Сравнение времени работы CPU и ядра (конфигурация <<64 64>>)

|  |  |  |
| --- | --- | --- |
| Размер теста | Время на CPU (мс) | Время на GPU (мс) |
| 1 000 x 1 000 pix | 4179 | 10.5969 |

1. Примеры работы программы

|  |
| --- |
|  |
|  |

|  |  |
| --- | --- |
|  | |
|  | |
|  | |
|  | |
|  | |
|  | |
|  |
|  |

**Вывод**

В ходе выполнения этой лабораторной работы, я ознакомился с методами классификации пикселей изображения, допускающими распараллеливание, и реализовал метод расстояний Махаланобиса, который имеет применения в области машинного обучения. Я получил опыт работы с константной памятью в среде CUDA.

Также я понял, что иногда проще и эффективнее использовать глобальную память, даже несмотря на то, что обработка текстур уже реализована и, казалось бы, совсем немного осталось изменить. Помимо этого, оказывается, программа, хранящая расстояния в интах, а не даблах может дожить до 20-го теста на чекере, что весьма вводит в заблуждение относительно причин ее поломки на оном.

Версия на мультипроцессоре, разумеется, оказалась быстрее, чем на CPU, уменьшение затраченного времени при увеличении размерности сетки также наблюдалось, хотя вообще время затрачивалось больше, чем в л.р.2, что, вероятно, связано с большим количеством операций в ядре.