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.Ю. Гришин
Группа: 8О-408Б
Преподаватель: А.Ю. Морозов

Москва, 2024

Условие

1. **Цель работы:** научиться использовать GPU для классификации и кластеризации изображений. Использование константной памяти и одномерной сетки потоков.
2. **Вариант 4:** метод спектрального угла.

Программное и аппаратное обеспечение

- Графический процессор
  - Compute capability: 7.5
  - Объем графической памяти: 15,83 ГБ
  - Объем постоянной памяти: 65536 байт
  - Разделяемая память на блок: 49152 байт
  - Количество регистров на блок: 65536
  - Максимальное количество потоков на блок: 1024
  - Количество мультипроцессоров: 40
- Процессор
  - Количество физических ядер: 2
  - Количество логических ядер: 4
  - Частота: 2000 МГц
- Оперативная память
  - Тип оперативной памяти: DDR4
  - Объем: 32 ГБ
- Жесткий диск
  - Объем: 1 ТБ
- Программное обеспечение
  - ОС: Ubuntu 22.04.4 LTS
  - IDE: Visual Studio Code
  - Компилятор: nvcc

Метод решения

Решение данной задачи я начал с того, что задача классификации пикселей с использованием метод спектрального угла заключается в определении номера класса для каждого из пикселей.

Каждый класс описывается выборкой пикселей $(p_1, p_2, \dots, p_n)$, где $p_i = (r_i, g_i, b_i)$ – i-й пиксель, представленный в выборке класса. На основе этих трехмерных векторов высчитывается средний вектор

$$avg_i = \frac{1}{n} \cdot \sum_{j=1}^{n} p_j$$

Итого, номер класса, которому принадлежит пиксель p, определяется следующим образом

$$jc = argmax_j \langle p, \frac{avg_j}{|avg_j|} \rangle$$

Из этого определения следует, что для определения номера класса используется один и тот же набор средних векторов классов $(avg_1, avg_2, \dots, avg_n)$. Более того, эти векторы не изменяются в ходе работы алгоритма. Поэтому есть основания расположить их в константной памяти.

Как можно увидеть, при вычислении номера класса, к которому принадлежит текущий пиксель, не была использована информации о положении пикселя, а следовательно порядок вычисления не имеет значения, что дает основание для распараллеливания алгоритма, а также представлении исходного изображения как одномерного массива пикселей.

Описание программы

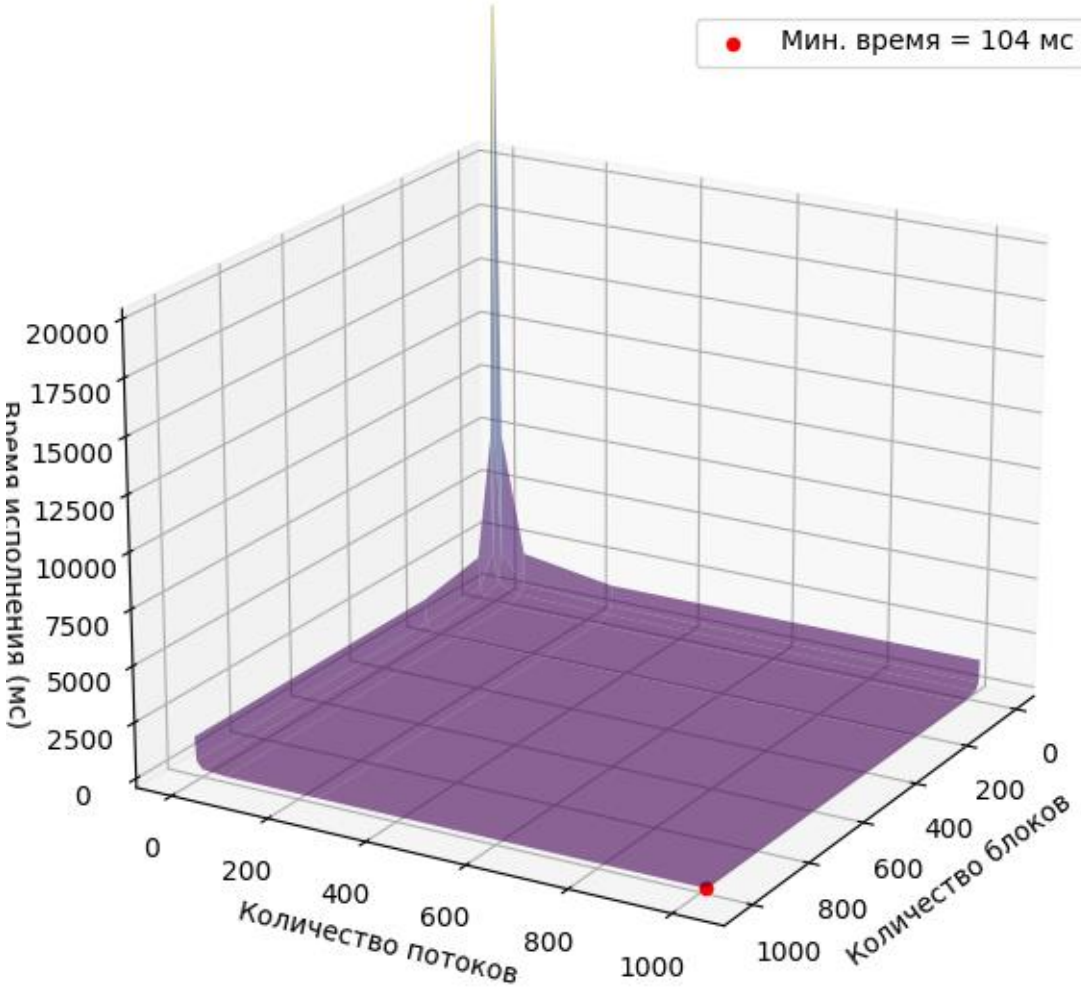
Программа состоит из одного файла, содержащего код, написанный на языке CUDA. Код программы состоит из:

1. Макроса `EXIT_WITH_ERROR`, который отвечает за вывод сообщения об ошибке в `stderr` и экстренный выход из программы с кодом возврата 0;
2. Макроса `SAVE_CUDA`, который проверяет, что CUDA функция сработала успешно. В противном случае происходит экстренный выход через `EXIT_WITH_ERROR`;
3. Типа Vector, который представляет трехмерный вектор, а также связанные с ним функции `vectorAdd`, `pixelToVector`, `vectorMult`, `vectorLength` для сложения векторов, получения вектора из пикселя, умножения вектора на скаляр и получения длины вектора соответственно.
4. Типа `Image`, который представляет абстракцию изображения, а также связанные с ним функции `loadImage` и `saveImage` для чтения и сохранения изображения в формате, описанном в условии лабораторной работы.
5. Функций `devicePixelToVector`, `deviceVectorDor` для выполнения конвертации пикселя в вектор и скалярного произведения векторов на GPU.
6. Функции `kernel`, которая содержит в себе реализацию алгоритма классификации пикселей методом спектрального угла
7. Переменной `constVectors`, которая представляет массив из 255 векторов, расположенных в константной памяти GPU. 255 описывает максимальное количество классов. Действительное количество классов описывается в переменной `actualSize`, которая также расположена в константной памяти.

Результаты

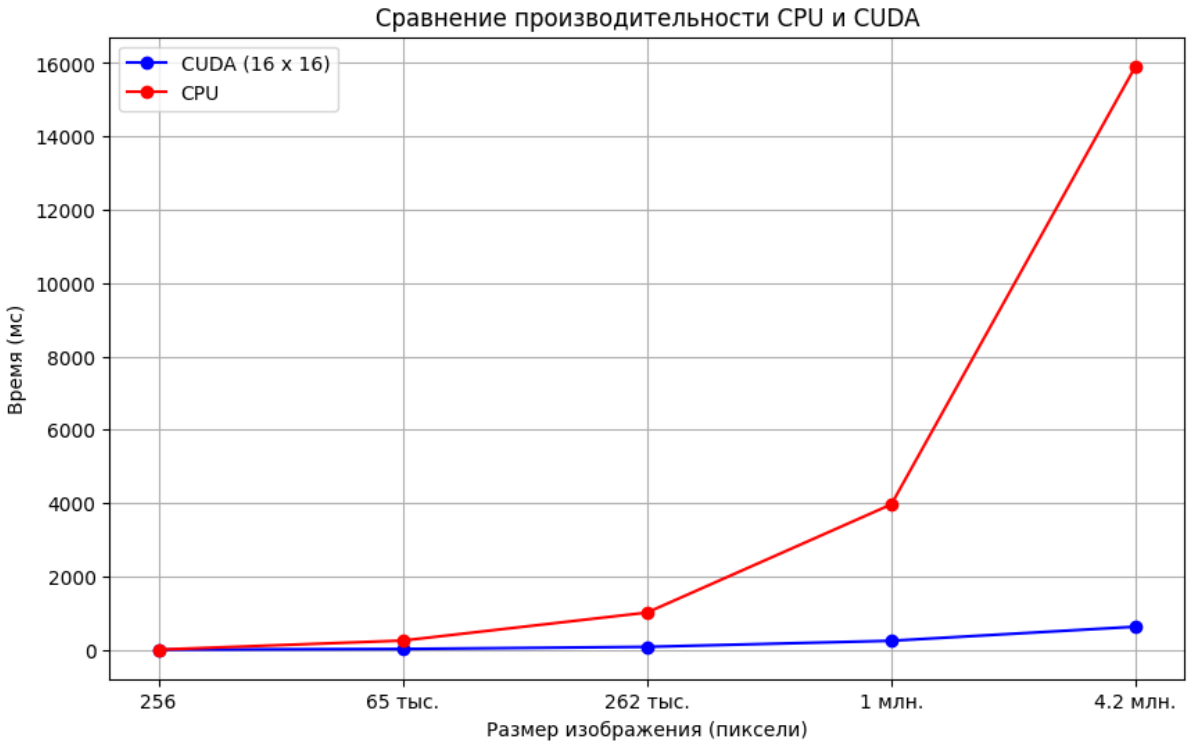
1. Зависимость времени выполнения программы от количества используемых потоков. В качестве единиц измерения времени были выбраны миллисекунды (мс). Тестирование производилось на изображении с размерами 4096×4096 и количестве классов, равным 255, каждый из которых содержит по 20 векторов.

Количество блоков / потоков	1	4	16	64	256	1024
1	116351	29408	7521	2156	1651	1606
4	29406	7503	2045	682	554	532
16	7519	1988	610	313	225	215
64	2141	688	275	129	128	128
256	1464	508	211	112	112	112
1024	1508	489	192	104	104	104

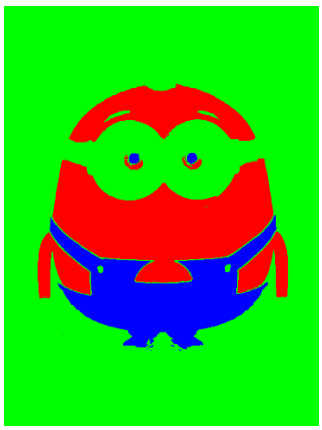


2. Сравнение программы на CUDA с 16 блоками и 16 потоками и программы на CPU с одним потоком. В качестве единиц измерения времени были выбраны миллисекунды (мс).

Размер изображения	Время на CUDA (мс)	Время на CPU (мс)
16×16	0,13	0,9
256×256	19	248
512×512	77	1012
1024×1024	243	3963
2048×2048	627	15915



3. Примеры обработанных изображений изображений



Выводы

В данной лабораторной работе был реализован алгоритм классификации пикселей с использованием метода спектрального угла на языке CUDA. Данный алгоритм широко применяется при анализе изображений и дистанционном зондировании.

При реализации алгоритма основной задачей была работа с константной памятью и организации одномерной сетки потоков. Текстурная память примечательна тем, что имеет низкую латентность для операции чтения из-за используемого независимого кеша. Однако, она имеет ограничения в виде доступа только на чтение при выполнении кода ядра и ограниченного объема памяти.

Также, в ходе выполнения лабораторной работы был использован модификатор `__device__` для некоторых функций. Такой модификатор позволяет создавать функции, которые в будущем могут быть вызваны со стороны ядра GPU. Это позволяет декомпозировать логику, исполняемую на ядре, на несколько функций.

При анализе работы я заметил значительное ускорение выполнения по сравнению с аналогичной реализацией на CPU. Временная сложность задачи при выполнении на GPU или CPU была квадратичной, из-за того, что алгоритм представляет собой применение алгоритма классификации к каждому из пикселей, однако из-за использования константной памяти и распараллеливания, вычисление на GPU оказалось значительно оптимальнее.