

**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. Л. Ядров
Группа: 8О-408Б
Преподаватель: А. Ю. Морозов,
К. Г. Крашенинников

Москва, 2023

Условие

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

Вариант: 4. Метод спектрального угла.

Программное и аппаратное обеспечение

Характеристики графического процессора:

- Наименование: NVIDIA GeForce GTX 1050
- Compute capability: 6.1
- Графическая память: 2047 Mb
- Разделяемая память на блок: 48 Kb
- Количество регистров на блок: 65536
- Максимальное количество потоков на блок: 1024
- Максимальное количество блоков: [1024, 1024, 64]
- Константная память: 64 Kb
- Количество мультипроцессоров: 5

Характеристики системы:

- Процессор: Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz
- Память: 8192 Mb

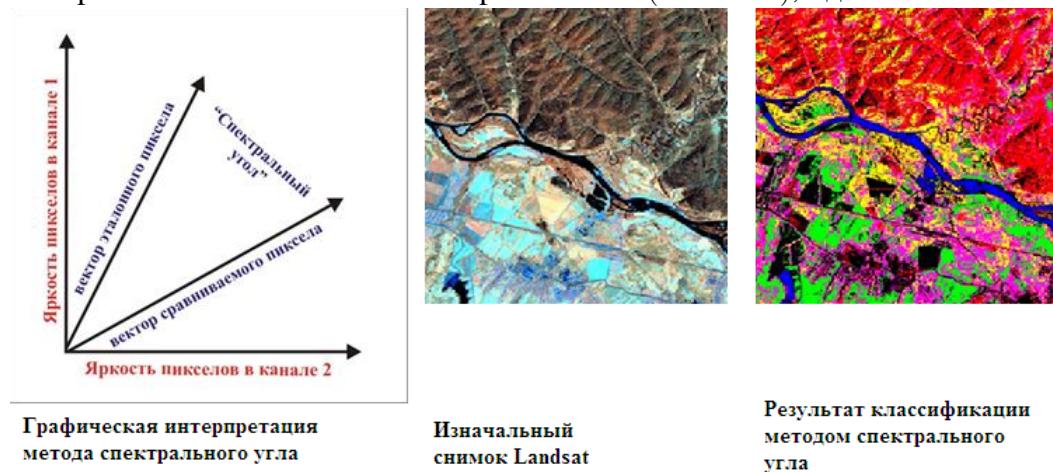
Программное обеспечение:

- ОС: Windows 11 (22H2)
- IDE: Microsoft Visual Studio 2022 (Community Edition)
- Компилятор: nvcc V12.2.140, MSVC V19.37.32824

Метод решения

Метод спектрального угла относится к методам классификации с обучением. Данное семейство методов предполагает наличие эталонов, с которыми сравнивается каждый пиксель. В результате, имея несколько эталонов, заранее заданных, мы имеем множество объектов, разделенных на классы. Для метода спектрального угла нам понадобится представить каждый пиксель в виде вектора и вычислить угол между текущим пикселям и эталонным пикселям. В качестве результирующего класса для пикселя выбирается класс, имеющий наименьший угол между текущим и эталонным пикселям.

Алгоритмическая сложность алгоритма — $O(n \cdot w \cdot h)$, где n — количество классов.



Описание программы

Проект я разделил на следующие модули:

1. *common defines* - содержит множество полезных дефайнов для более приятной работы с *CUDA* и *C++*.
2. *common structures* - содержит класс-обертку над *CUDA* событиями и класс-обертку над *std :: stringstream*, позволяющим конструировать строки (*sprintf* с возможностями *C++*). Также сюда были помещены классы-обертки над текстурными объектами и ресурсами.
3. *operation system* - содержит операции, связанные с системными вызовами (например, получение информации о количестве оперативной памяти). Реализован только для ОС *Windows*.
4. *claster* - содержит класс кластера, имеющий методы для чтения и вычисления среднего значения.
5. *kernel* - непосредственно программа, выполняющая все операции, связанные с вводом, вычислениями и выводом.

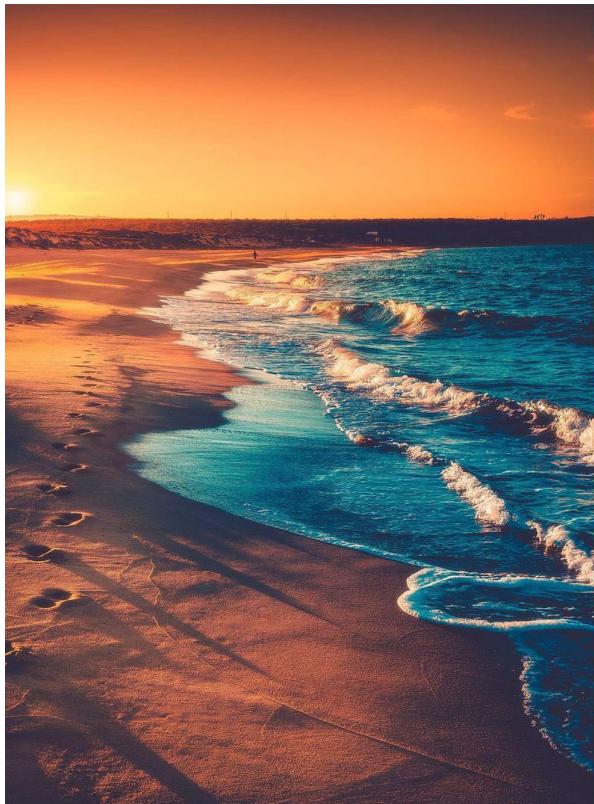
Также для замера времени была написана программа на языке программирования *Python*, состоящая из следующих модулей:

1. *build* используется для компиляции *Microsoft Visual Studio* проекта (файла с расширением *.sln*).
2. *tests generator* используется для генерации тестов и ответов на них (ответы генерируются с помощью программы, написанной под *CPU*).
3. *cheker* используется для проверки работоспособности программы. Для этого запускается раннее сгенерированный *.exe* файл с входными данными из теста, далее вывод программы проверяется с ответом.
4. *benchmark* используется для замера скорости работы программы. Для этого программа запускается 5 раз с входными тестовыми данными. Для каждого теста временем работы будет минимальное время работы из всех запусков данного теста.
5. *converter* используется для конвертации изображения в бинарный вид и для получения изображения из бинарных данных.
6. *main* используется в качестве обертке над описанными выше модулями для более комфортного запуска модулей.

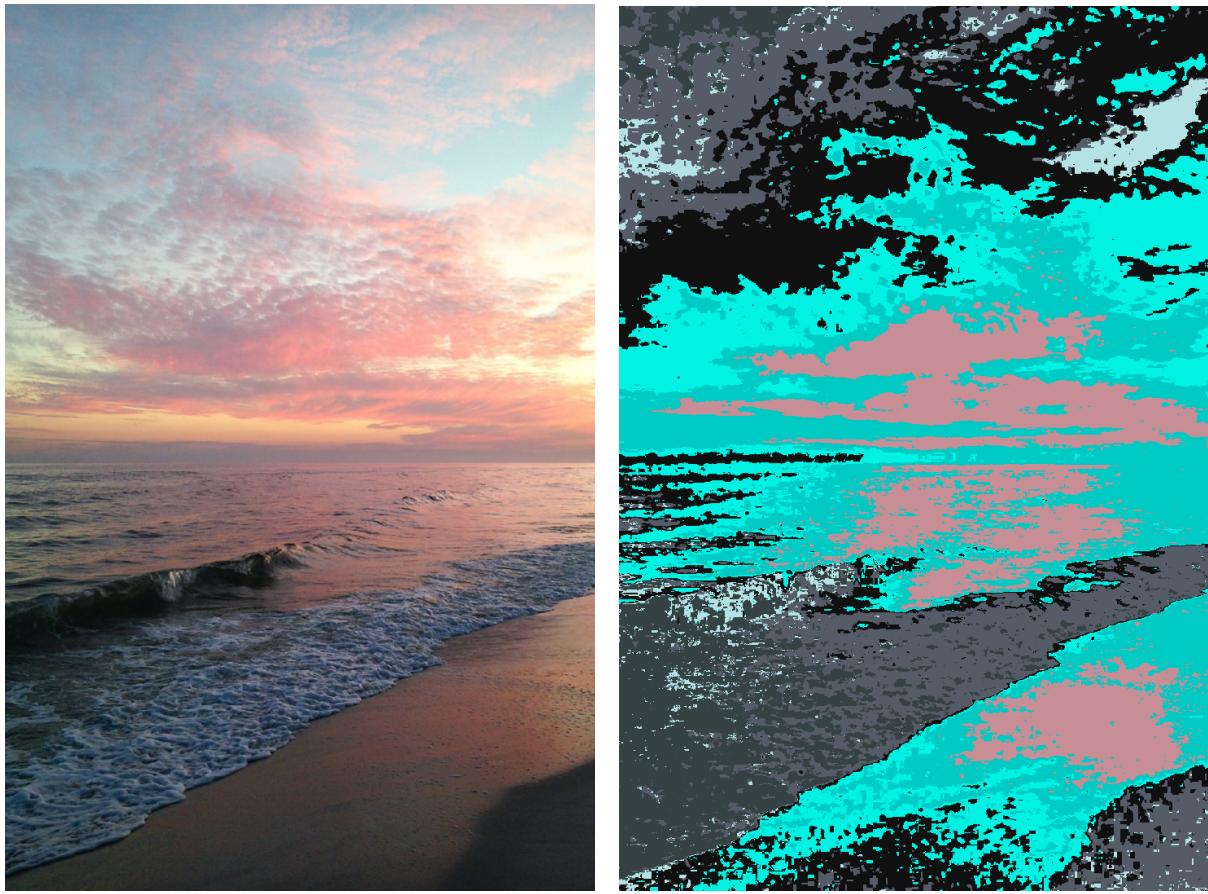
Результаты

Для удобной работы с реальными изображениями мною был переделан формат кластеров. По условию кластер представляет собой набор пикселей. Для удобного ввода я представил кластер в виде прямоугольника и подавал на вход лишь координаты левого верхнего и правого нижнего углов прямоугольника.

Примеры работы программы:







В качестве тестовых данных было взято квадратное изображение заданного размера. Пиксели генерировались с помощью генератора псевдослучайных чисел.

Сравнение времени работы

| Размер теста | 100×100 | 500×500 | 1000×1000 | 1000×1000 | 2000×2000 | 2000×2000 | 5000×5000 | 5000×5000 |
|---------------|----------------------|------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|
| Число классов | 10 | 10 | 10 | 32 | 10 | 32 | 10 | 32 |
| Конфигурация | Время выполнения, мс | | | | | | | |
| CPU | 9.1379 | 189.2646 | 529.117 | 1545.433 | 1937.266 | 5923.956 | 11940.333 | 36821.500 |
| 1, 32 | 1.4703 | 38.0822 | 152.122 | 442.399 | 608.409 | 1769.516 | 3802.423 | 11058.966 |
| 32, 32 | 0.1618 | 3.4147 | 13.548 | 40.965 | 54.131 | 164.102 | 338.453 | 1028.226 |
| 64, 64 | 0.1383 | 2.7711 | 11.035 | 34.528 | 44.074 | 138.093 | 275.482 | 862.861 |
| 128, 128 | 0.1385 | 2.6698 | 10.624 | 33.326 | 42.452 | 133.250 | 265.203 | 832.737 |
| 256, 256 | 0.1378 | 2.6618 | 10.618 | 33.315 | 42.408 | 133.211 | 264.936 | 832.471 |
| 512, 512 | 0.1647 | 2.7680 | 10.757 | 33.754 | 42.873 | 134.950 | 267.780 | 843.309 |
| 1024, 1024 | 0.2754 | 2.9208 | 11.234 | 34.477 | 43.730 | 136.688 | 271.044 | 852.250 |

Выводы

Данный алгоритм может быть использован для обработки спутниковых фотографий поверхности земли, обработки снимков биологических объектов под микроскопом. Полученные области на изображении могут быть использованы для дальнейшего анализа изображения (например, для распознавания образов).

Также стоит увеличить количество классов, используемых для классификации. В идеале можно создать приложение, в котором пользователь выделяет нужные ему пиксели, задающие кластер. После выделения по нажатию кнопки происходит классификация. В рамках одного открытого изображения можно занести вектор пикселей в константную память, а кластера загружать динамически.

Список литературы

- [1] Бъярне Страуструп. *Программирование. Принципы и практика с использованием C++, 2-е издание.* — Издательский дом «Вильямс», 2016. Перевод с английского: И. В. Красиков. — 1328 с. (ISBN 978-5-8459-1949-6 (рус.))
- [2] *CUDA Runtime API.*
URL: <https://docs.nvidia.com/cuda/cuda-runtime-api/index.html> (дата обращения: 16.12.2013).