МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Информационные технологии 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.В. Куликов
Группа: 8О-408Б
Преподаватели: К.Г. Крашенинников,
А.Ю. Морозов

Москва, 2020

Условие

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

Вариант 5. Метод k-средних.

На первой строке задается путь к исходному изображению, на второй, путь к конечному изображению. На следующей строке, число nc -- кол-во кластеров. Далее идут nc строчек описывающих начальные центры кластеров. Каждая i-ая строчка содержит пару чисел -- координаты пикселя который является центром. $nc \leq 32$.

Программное и аппаратное обеспечение

Видеокарта	GeForce GT 545
Compute capability	2.1
Графическая память	3004 Мб
Разделяемая память	48 Кб
Константная память	64 Кб
Количество регистров на блок	32768
Максимальное кол-во блоков	65535*65535*65535
Максимальное кол-во нитей в блоке	1024
Кол-во мультипроцессоров	3
Ядер CUDA	144

Процессор	Intel Core i7-3770
ОЗУ	16 Гб
ЖД	

Операционная система	Ubuntu 16.04.6 LTS
IDE	VS Code
Компилятор	nvcc V7.5.17

Метод решения

Инициализируем центроиды значениями пикселов, указанных как центры класстеров во входных данных. Далее каждый из пикселов относится к ближайшему по евклидовой метрике (в качестве координат вектора используются каналы цвета) кластеру. После отнесения к классам пересчитываются центроиды классов. Новая центроида получается как среднее по кластеру значение пикселя. Если при пересчете центроиды не изменились (или изменились незначительно), значит кластеризация завершена.

Описание программы

В файле main.cu находится основная логика программы, включая ввод/вывод, работа с памятью, а также сама функция, реализующая алгоритм k-means, и вспомогательное ядро belong_to_cluster для вычисления на ГП.

В файле error.h находится функционал для обработки ошибок CUDA API и хендлеры для стандартных сигналов SIGSEGV, SIGABRT.

В файле benchmark.h находится функционал для измерения времени исполнения кода.

Поставленную задачу решает функция k_means. Она реализует описанный выше алгоритм. На ГП происходит только отнесение пикселя к кластеру с помощью ядра belong_to_cluster. Перерасчет центроид кластера происходит на ЦП.

При этом рассчитанные центроиды хранятся в константной памяти, с целью ускорения доступа к ним в ядре. Это оправданно т. к. центроид немного, но число обращений к ним равно числу пикселей изображения, что достаточно много.

Результаты

Тестирование ядер с различными конфигурациями

Маленький файл, полученный из bmp-файла 800x600 (1,8 мб)

---	32 блок	64 блок	128 блок	256 блок	512 блок	1024 блок
1 грид	126.897346	63.670654	32.549789	16.791424	9.324576	5.347456
32 грид	4.718976	2.432288	1.594688	1.007904	0.875936	0.838976
128 грид	1.495520	1.042624	0.896352	0.857344	0.892832	0.866976
256 грид	1.033216	0.858016	0.886080	0.820352	0.875680	0.939552
512 грид	1.011648	0.851296	0.877856	0.904128	0.892544	0.897600
1024 грид	1.034208	0.849952	0.853440	0.836512	0.919456	1.010432

Лучший результат: 0.820352 мс при параметрах <256, 256>

Средний файл, полученный из bmp-файла 1920x1080 (7,9 мб)

---	32 блок	64 блок	128 блок	256 блок	512 блок	1024 блок
1 грид	354.50238	198.64749	98.113503	51.021763	27.832542	15.078688
32 грид	13.344417	7.082497	4.167264	2.710592	2.329440	2.336640
128 грид	4.096224	2.678560	2.281504	2.305632	2.288352	2.286144
256 грид	2.679616	2.290016	2.287744	2.276384	2.254848	2.260448
512 грид	2.719392	2.281760	2.280032	2.266464	2.264736	2.251104
1024 грид	2.704160	2.268512	2.273344	2.231200	2.239104	2.247840

Лучший результат: 2.23120 мс при параметрах <1024, 256>

Большой файл, полученный из bmp-файла 6000x4000 (91,6 мб)

---	32 блок	64 блок	128 блок	256 блок	512 блок	1024 блок
1 грид	4495.00879	2234.76807	1168.61108	633.930237	374.721497	203.886032
32 грид	175.368408	92.237595	52.721821	33.874207	28.988607	28.858784
128 грид	52.921410	34.363262	30.339586	28.502207	27.715681	27.404896
256 грид	34.638783	31.039392	29.400671	27.841120	27.168417	27.271776
512 грид	34.224129	30.012352	28.149502	27.221504	27.152189	27.010015
1024 грид	33.589439	28.496672	27.585438	27.112736	27.008484	26.717022

Лучший результат: 26.717022 мс при параметрах <1024, 1024>.

Сравнение с CPU

Маленький файл, полученный из bmp-файла 800x600 (1,8 мб)
Результат: 30.977298 мс

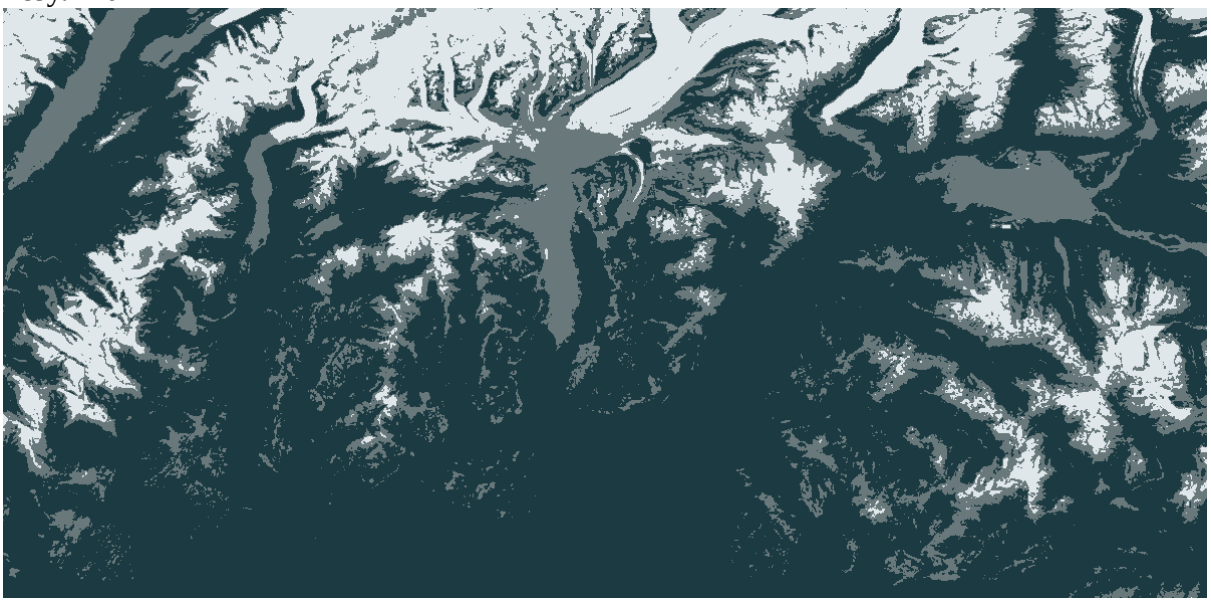
Средний файл, полученный из bmp-файла 1920x1080 (7,9 мб)
Результат: 84.469921 мс

Большой файл, полученный из bmp-файла 6000x4000 (91,6 мб)
Результат: 884.664719 мс

Результаты работы программы

Исходное изображение

Результат



Выводы

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

Особых проблем, кроме глупых ошибок, в решении задачи не возникало. В решении данной задачи программа с использованием GPU значительно превзошла реализацию, работающую только на CPU.