

**МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)**

**Институт №8 «Компьютерные науки и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»**

**Лабораторная работа №2
по курсу «Программирование графических процессоров»**

Обработка изображений на GPU. Фильтры.

**Выполнил: В. В. Бирюков
Группа: 8О-407Б
Преподаватель: А. Ю. Морозов**

Москва, 2022

Условие

Цель работы: Научиться использовать GPU для обработки изображений. Использование текстурной памяти.

Вариант: 7. Выделение контуров. Метод Собеля.

Программное и аппаратное обеспечение

Характеристики графического процессора:

- Наименование: NVIDIA GeForce GTX 1050
- Compute capability: 6.1
- Графическая память: 4236378112 Б
- Разделяемая память на блок: 49152 Б
- Количество регистров на блок: 65536
- Максимальное количество потоков на блок: (1024, 1024, 64)
- Максимальное количество блоков: (2147483647, 65535, 65535)
- Константная память: 65536 Б
- Количество мультипроцессоров: 5

Характеристики системы:

- Процессор: Intel Core i5-8300H 2.30GHz
- Память: 32 ГБ
- SSD: 1 ТБ
- HDD: 1 ТБ

Программное обеспечение:

- ОС: Kubuntu 20.04
- IDE: Sublime text 3
- Компилятор: nvcc 10.1.123

Метод решения

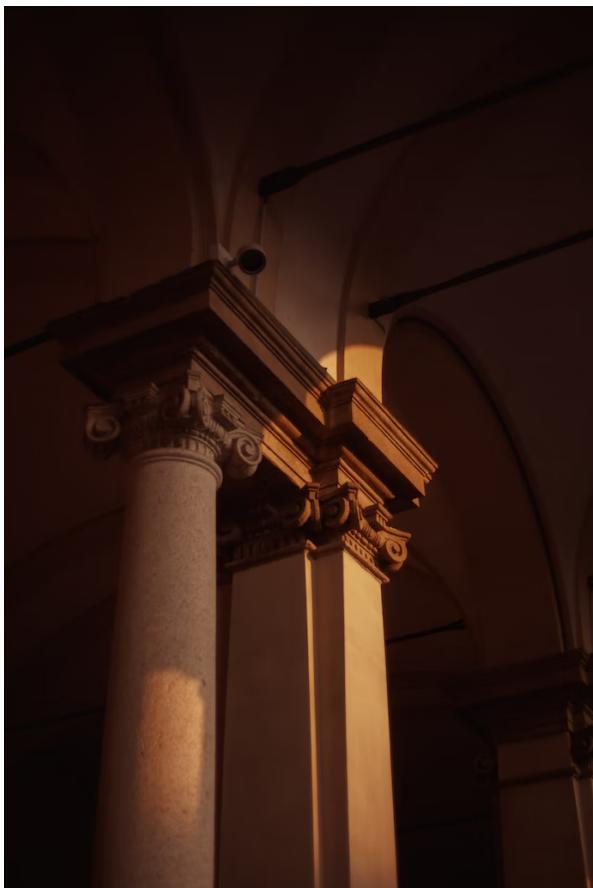
Для выделения границ изображения необходимо осуществить на каждом его пикселе две операции свертки, по одной на компоненту вектора градиента. При этом свертка осуществляется не по исходным значениям цвета, а по величине яркости, вычисляемой по формуле $u = 0.299r + 0.587g + 0.114b$. По полученным компонентам находится модуль градиента, его значение необходимо ограничить максимальной величиной компоненты цвета – 255. В пиксель выходного изображения записывается модуль градиента во все цветовые каналы, альфа канал не изменяется. Сложность алгоритма — $O(w \cdot h)$.

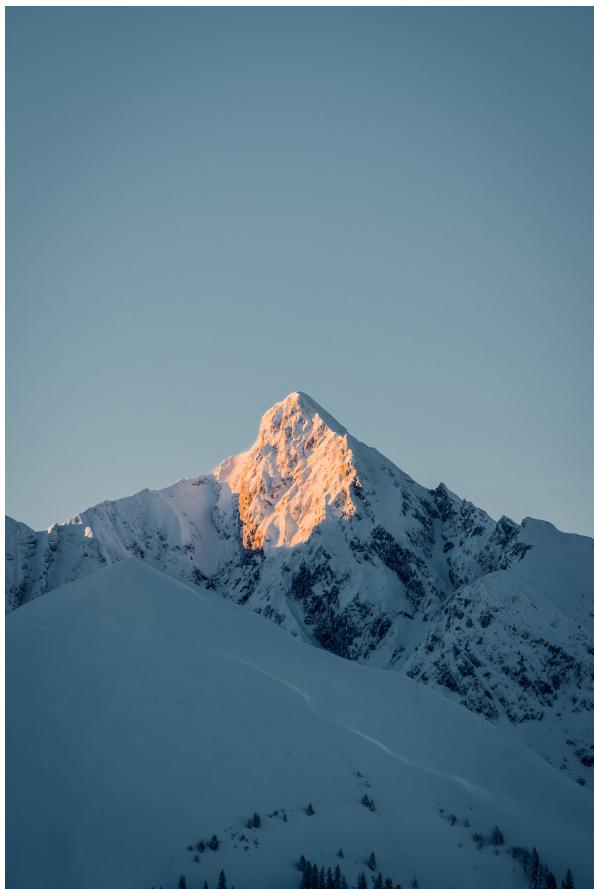
Описание программы

В файле `1ab2.cu` расположен основной код программы. Функция `sobel_filter` — вычислительное ядро — осуществляет описанный выше алгоритм и принимает четыре аргумента: текстурный объект, указатель на массив данных выходного изображения и размеры изображения. Ядра свертки записаны в константную память, так как к ним осуществляется много обращений.

Результаты

Примеры работы программы:





Сравнение времени работы:

Конфигурация	Время выполнения, мс				
CPU	1.902230	48.822700	202.029000	819.271000	5135.030000
1x1, 32x1	0.627040	12.253200	65.633900	215.664000	1238.230000
1x1, 32x32	0.183968	3.098140	15.405900	47.854300	302.103000
32x32, 32x8	0.114176	0.774368	2.823550	9.764260	60.597000
32x32, 32x32	0.155936	0.971296	3.378780	10.265200	61.083200
64x64, 32x8	0.200288	1.184830	3.285180	10.160500	60.955600
64x64, 32x32	0.356704	1.397250	4.145730	12.191700	63.058200
128x128, 32x8	0.425472	1.819140	5.060800	11.851400	62.656300
128x128, 32x32	1.035650	2.492030	5.967460	15.019600	71.128200
Размер теста	100x100	500x500	1000x1000	2000x2000	5000x5000

Выводы

В ходе выполнения лабораторной работы я познакомился с обработкой изображений при помощи механизма свертки. Сверточные фильтры используются как непосредственно для обработки, так и, например, в компьютерном зрении.

В процессе выполнения работы я столкнулся с тем, что текстурные объекты доступны только на GPU с compute capability ≥ 3.0 , в связи с чем мне пришлось тестировать программу на другом устройстве. Результаты тестирования в целом не изменились: время выполнения падает при увеличении числа потоков, но затем снова начинает возрастать. Единственное отличие: даже использование только одного варпа значительно сокращает время работы, по сравнению с исполнением на CPU. Оптимальная конфигурация ядра: сетка блоков — 32x32, сетка потоков — 32x8.