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: Р.С. Лисин

Группа: 8О-406Б

Преподаватели: К.Г. Крашенинников,

А.Ю. Морозов

Москва, 2023

**Условие**

**Цель работы**: Научиться использовать GPU для обработки изображений.

Использование текстурной памяти и двухмерной сетки потоков.

**Вариант 7. Выделение контуров. Метод Собеля.**:

**Входные данные**. На первой строке задается путь к исходному изображению,

на второй, путь к конечному изображению.

**Программное и аппаратное обеспечение**

В качестве графического процессора использую видеокарту NVIDIA Tesla T4.

Compute capability : 7.5

Name : Tesla T4

Total Global Memory : 15835398144

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 : 40

В качестве редактора кода использовался Jupyter Notebook в Google Colab.

**Метод решения**

На GPU реализуем метод Собеля, который заключается в проходе фильтра 3×3 по всему изображению и вычисления нового значения в каждой точке в зависимости от яркости.

**Описание программы**

Создаётся один динамический двумерный массив data. В программе применяется интерфейс работы с данными - текстурная ссылка, которая привязывается к определённой области памяти. Далее настраиваем её с помощью различных политик, и связываем интерфейс с данными. Он копируется на GPU. В функции ядра kernel в функции девайса rgb\_to\_luma выполняется преобразование rgb в параметр яркости (luma) и реализуется метод Собеля выделения контуров. Результат записывается в out\_arr.

**Результаты**

Рассмотрим время работы программы на различных тестах при различных размерах сетки и на CPU. Будем замерять непосредственно время работы алгоритма. В качестве тестов используется одна картинка с видом на море. Для разных тестов меняются её размеры. Результаты приведены в таблице ниже.

|  |  |  |  |
| --- | --- | --- | --- |
| Размер сетки ядра | 500x500 px, мс | 1000x1000 px, мс | 5000x5000 px, мс |
| CPU | 44.245000 | 190.048000 | 5530.973000 |
| <<<(1, 1), (32, 32)>>> | 18.299232 | 73.058434 | 818.964783 |
| <<<(8, 8), (8, 8)>>> | 0.633056 | 2.323936 | 56.674946 |
| <<<(8, 8), (16, 16)>>> | 0.624128 | 2.376096 | 57.311039 |
| <<<(16, 16), (32, 32)>>> | 0.535264 | 1.991584 | 48.960224 |
| <<<(32, 32), (32, 32)>>> | 0.537216 | 1.932384 | 45.457890 |

Алгоритм на CPU справляется гораздо медленнее чем на GPU. Это безусловно связано с тем, что в данном случае распараллеливание в разы ускоряет работу алгоритма.

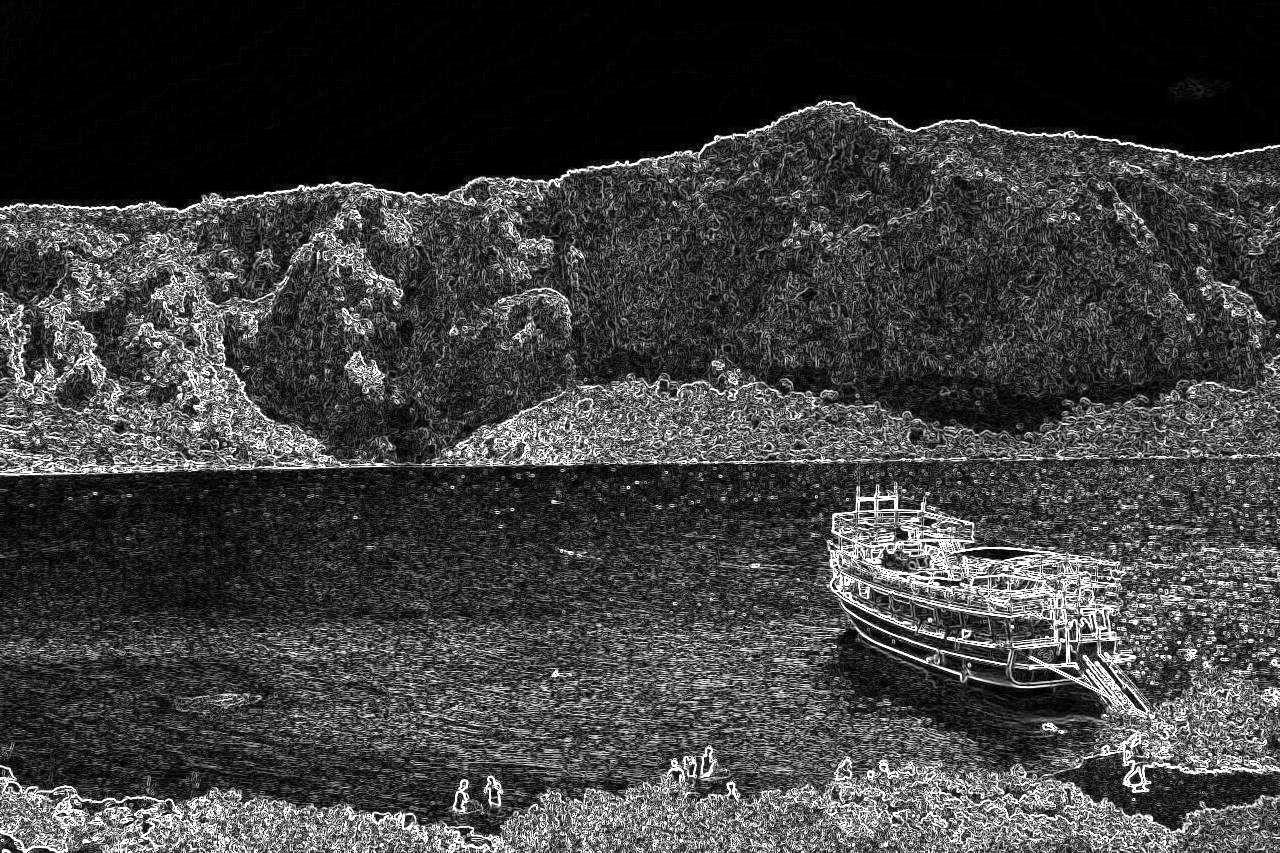
Примеры картинок.











**Выводы**

Во второй лабораторной работе я познакомился с текстурами, методом Собеля выделения контуров изображения. GPU позволяет очень быстро обрабатывать изображения, собственно поэтому он и называется графическим процессором. Но с другой стороны программы на CPU отлаживать проще и удобнее. Получились красивые картинки, которые можно использовать в качестве иллюстраций к какой-нибудь книге или статье.