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

Москва, 2024

Условие

1. **Цель работы:** научиться использовать GPU для обработки изображений. Использование текстурной памяти и двухмерной сетки потоков.
2. **Вариант 7:** выделение контуров. Метод Собеля.

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 применения метода Собеля на изображение сводится к применению определенного алгоритма к каждому из пикселей изображения. Задача такого алгоритма состоит в расчете значений цветов пикселя, которые получатся после применения фильтра.

Опишем этот алгоритм в виде функции $f(x, y)$, где (x, y) – позиция текущего исходного пикселя. Тогда применение метода Собеля можно описать как вызов функции $f(x, y)$ для $x \in \overline{1, m}$, $y \in \overline{1, n}$, где (m, n) – размер изображения.

Отметим, что подзадачи $f(x, y)$ не зависят друг от друга и используют неизменяемую текстурную память. А следовательно, есть основание рассмотреть параллельную версию этого алгоритма.

Пусть $p_{x,y} = (r_{x,y}, g_{x,y}, b_{x,y})$ – информация о цветах пикселя, находящегося на позиции (x, y). Так как метод Собеля представляет собой применение сверток

$$G_X = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}, G_Y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

То функцию $f(x, y)$ можно описать как

$$f(x, y) = \sqrt{\left(G_X \circ W_{x,y}\right)^2 + \left(G_Y \circ W_{x,y}\right)^2}$$

Где $A \circ B = \sum_{i=1,j=1}^{n,m} A_{i,j} \cdot B_{i,j}$, и $W_{x,y} = \begin{pmatrix} Y_{x-1,y-1} & Y_{x,y-1} & Y_{x+1,y-1} \\ Y_{x-1,y} & Y_{x,y} & Y_{x+1,y} \\ Y_{x-1,y+1} & Y_{x,y+1} & Y_{x+1,y+1} \end{pmatrix}$, $Y_{x,y} = 0.299 * r_{x,y} + 0.587 * g_{x,y} + 0.114 * b_{x,y}$.

Описание программы

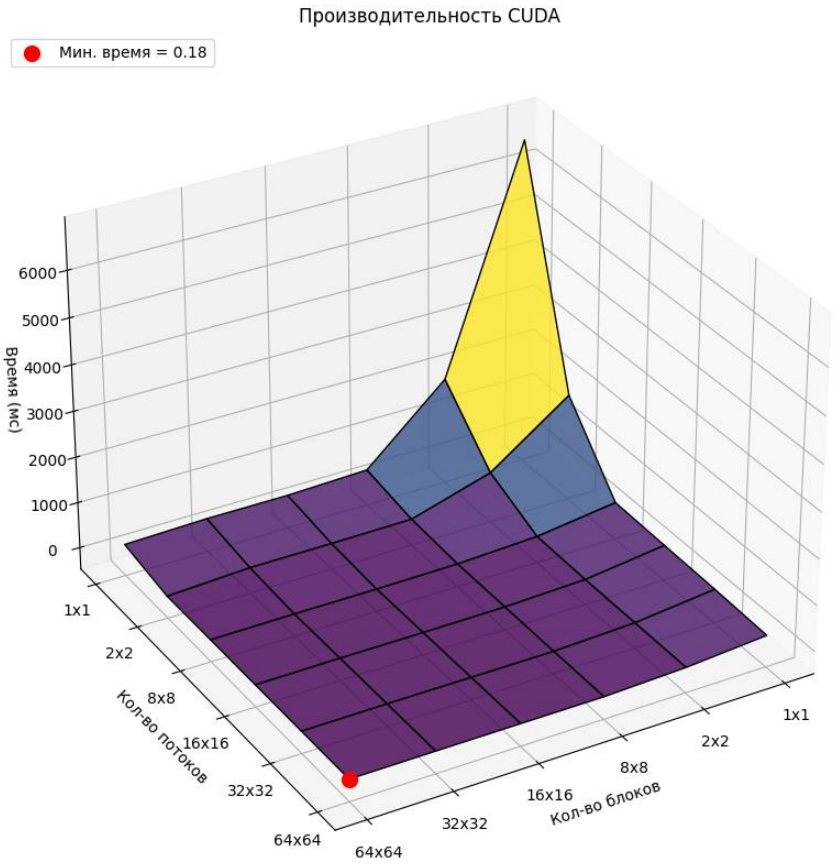
Программа состоит из одного файла, содержащего код, написанный на языке CUDA. Код программы состоит из:

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. Функции `kernel`, которая содержит в себе реализацию метода Собеля;
4. Типа `Image`, представляющего абстракцию изображения, и методов `loadImage` и `saveImage` для чтения и сохранения изображения в формате, описанном в условии лабораторной работы;
5. Типа `Texture`, представляющего абстракцию текстуры, а также метода `create_texture_from_image` для создания текстуры на основе изображения и метода `free_texture`, выполняющего роль деструктора для типа `Texture`;
6. Функции `main`, которая отвечает за инициализацию ресурсов, с которыми в дальнейшем будет работать функция `kernel`. А также за их освобождение и вывод результата на экран.

Результаты

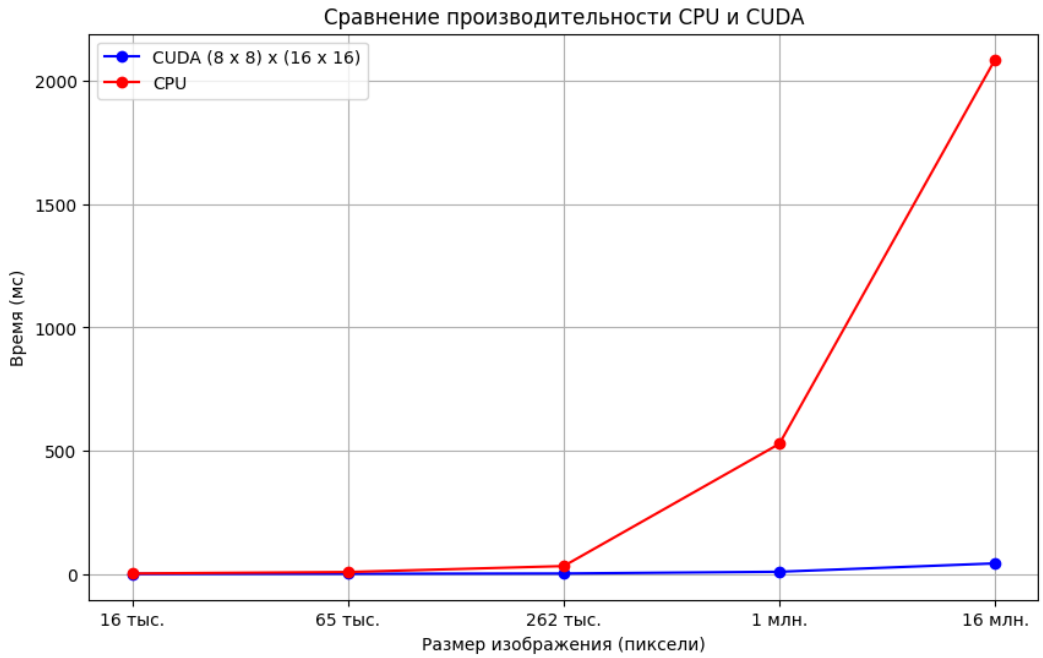
1. Зависимость времени выполнения программы от количества используемых потоков. В качестве единиц измерения времени были выбраны миллисекунды (мс). Тестирование производилось на изображении с размерами 1024×1024

Количество блоков / потоков	1 × 1	2 × 2	8 × 8	16 × 16	32 × 32	64 × 64
1 × 1	6627	1851	309	254	256	210
2 × 2	1853	625	102	89	88	0,2
8 × 8	334	103	26	11	11	0,2
16 × 16	295	77	19	9	10	0,2
32 × 32	260	75	9	9	9	0,2
64 × 64	207	72	9	8	5	0,14



2. Сравнение программы на CUDA с 8×8 блоками и 16×16 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)
128×128	0,301	1,989
256×256	0,992	8
512×512	1,941	32
1024×1024	9	528
4048×4048	43	2087



Выводы

В данной лабораторной работе был реализован алгоритм выделения контуров с использованием метода Собеля на языке CUDA. Данный метод широко применяется в графических редакторах, в компьютерном зрении, в частности, при препроцессинге для выделения границ объектов на изображениях.

При реализации алгоритма основной задачей была работа с текстурной памятью и организации двумерной сетки потоков. Текстурная память примечательна тем, что изначально она была разработна для решения задач, связанной с обработкой графики, поэтому обработка изображений с ее использованием получается оптимизированным и удобным, из-за наличия пространственного кеша, нормализации координат и значений.

Также, в ходе выполнения лабораторной работы была использована двумерная сетка потоков. Ее преимущество заключается в том, что теперь потоки идентифицируются парой чисел, в следствие чего их расположение естественным образом соответствует структуре изображения. Данный подход позволяет без осложнений распараллеливать процесс обработки изображения.

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 свертки к каждому из пикселей, однако из-за использования текстурной памяти и распараллеливания, вычисление на GPU оказалось значительно оптимальнее.