МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Информационные технологии и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

Лабораторная работа №2
по курсу «Программирование графических процессоров»

Обработка изображений на GPU. Фильтры.

Выполнил: Д.В. Коростелев
Группа: 8О-408Б
Преподаватели: К.Г. Крашенинников,
А.Ю. Морозов

Москва, 2021

Условие

Цель работы:

Научиться использовать GPU для обработки изображений.
Использование текстурной памяти.

Вариант 6. Выделение контуров. Метод Превитта.

Входные данные. На первой строке задается путь к исходному изображению, на второй, путь к конечному изображению. $w*h \leq 10^6$

Программное и аппаратное обеспечение

- Графический процессор NVIDA GeForce GTX 1050

Графическая память	2 Gb
Разделяемая память на блок	48 Mb
Константная память	64 Mb
Количество регистров на блок	65536
Максимальное количество блоков на процессор	32
Максимальное количество потоков на блок	1024
Количество мультипроцессоров	5

- Процессор Intel Core i7-7700HQ 4x 2.808ГГц

Количество ядер	4
Количество потоков	8
Базовая тактовая частота	2.80 GHz
Максимальная тактовая частота	3.80 GHz
Кеш-память	6 Mb

- Оперативная память DDR4-SODIMM

Объем памяти	8 Gb
Частота	2400 MHz
Форм-фактор	SODIMM
Количество плашек	2

- SSD и HDD накопители

Объем SSD накопителя	128 Gb
Объем HDD накопителя	1 Tb

- Программное обеспечение

Операционная система	Windows 10 Pro
Средство разработки на CUDA (IDE)	Microsoft Visual Studio 2019
Компиляторы	MSVC 2019
Версия CUDA Toolkit	11.4.2
Дополнительный текстовый редактор	Notepad++

Метод решения

Данные изображения помещаем в текстурную память CUDA, так как CUDA в контексте данной задачи имеет удобный функционал работы с индексацией по текстуре как внутри

границ, так и за ее пределами. Создаем двумерную сетку из блоков и потоков. Запускаем поток и в каждом потоке фильтруем (находим новое значения пикселя) и записываем результат пикселя.

Описание программы

В функции main производим считывание данных из стандартного потока данных, аллоцируем текстурную память на видеокарте, задаем текстурную ссылку, предварительно задав каким образом обрабатывать взятие точек за границей текстуры, а также дополнительные параметры, передаем считанные данные на видеокарту и производим вычисления, ожидаем, когда все потоки закончат свою работу, производим копирование из памяти видеокарты в оперативную и выводим в выходной файл. Обрабатываем все возможные ошибки в процессе работы с CUDA-функциями.

Результаты

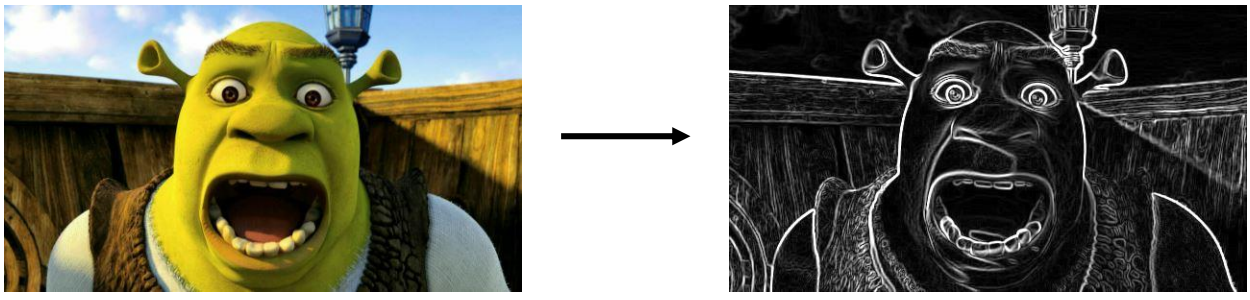
Замеры времени работы ядер с различными конфигурациями (размерность теста показывает по сколько элементов содержится во входных массивах, значения в ячейках показывает затрачиваемое время на обработку данных).

CUDA конфигурация / Размерность теста	180x146	539x299	1200x675
<<<4x4, 4x4>>>	1036нс	971нс	1150нс
<<<8x8, 8x8>>>	1047нс	1115нс	1093нс
<<<16x16, 16x16>>>	1062нс	1198нс	1113нс
<<<32x32, 32x32>>>	924нс	1270нс	1209нс

Сравнение вычислений на CPU с вычислениями на GPU

Размерность входных данных	CPU	<<<16x16, 16x16>>>	<<<32x32, 32x32>>>
180x146	8495нс	1062нс	924нс
539x299	37086нс	1198нс	1270нс
1200x675	179700нс	1113нс	1209нс

Пример работы программы



Выводы

Полученная программа позволяет крайне быстро обрабатывать картинки больших размерностей и выделять на них контуры. Фильтры подобного рода используются во многих нейронных сетях, в сверочных слоях. Кроме того, на примере данной программы можно легко понять, как взаимодействовать с текстурной памятью CUDA.