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: И.А. Мариничев
Группа: 8О-408Б
Преподаватели: К.Г. Крашенинников
А.Ю. Морозов

Москва, 2022

Условие

Цель работы: научиться использовать GPU для обработки изображений.
Использование текстурной памяти и двухмерной сетки потоков.

Вариант 1: гауссово размытие.

Программное и аппаратное обеспечение

Compute capability	:	2.1
Name	:	GeForce GT 545
Total Global Memory	:	3150381056
Shared memory per block	:	49152
Registers per block	:	32768
Warp size	:	32
Max threads per block	:	(1024, 1024, 64)
Max block	:	(65535, 65535, 65535)
Total constant memory	:	65536
Multiprocessors count	:	3
Processor	:	Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz
RAM	:	16 GB
Drive	:	349G
OS	:	Ubuntu 16.04.6 LTS
IDE	:	Visual Studio Code
Compiler	:	NVIDIA (R) Cuda compiler driver V7.5.17

Метод решения

Считываем входные данные, в том числе из файла. Затем создаем текстурный объект и на CPU рассчитываем массив ядра свертки, который копируем в константную память. После этого вызываем ядро размытия по гауссу по оси OX, где в каждый цветовой канал всех пикселей записывается результат свертки. Копируем результат обратно на CPU, обновляем данные в текстурной памяти и вызываем ядро размытия по гауссу по оси OY. В конце записываем результат в выходной файл и освобождаем память.

Описание программы

```
// ядро свертки в константной памяти
__constant__ float kernel[1024]

// обработка пикселей за границей (приводим к граничным)
__device__ float modeClamp(int p, int b)
```

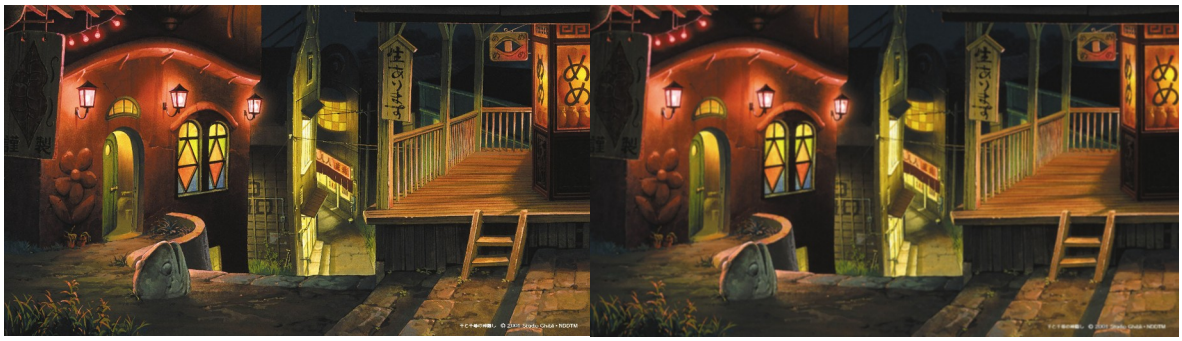
```
// размытие по гауссу по оси OX
__global__ void gaussianBlurX(uchar4 *out, cudaTextureObject_t texObj, int w, int h, int r)

// размытие по гауссу по оси OY
__global__ void gaussianBlurY(uchar4 *out, cudaTextureObject_t texObj, int w, int h, int r)
```

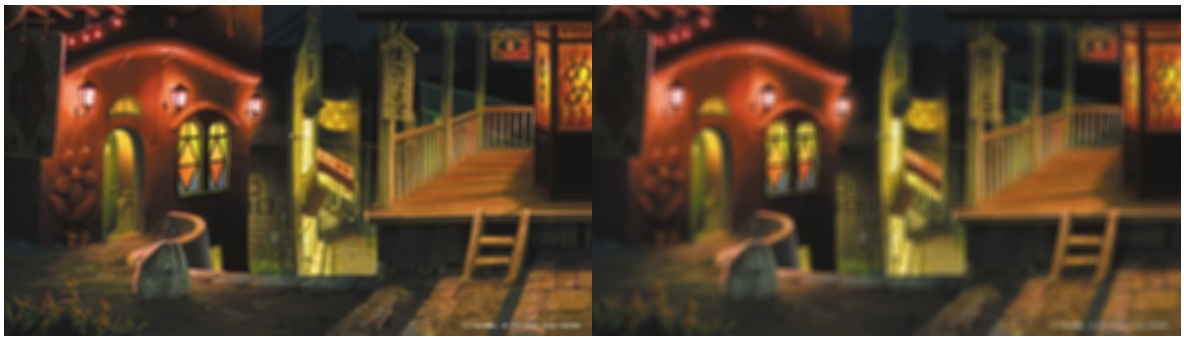
Результаты

Конфигурация	Размер теста (время указано в ms)				
	O(10^2)	O(10^4)	O(10^5)	O(10^6)	O(10^7)
<<<(32, 32), (32, 32)>>>	0.034464	14.546017	75.905632	1519.959473	3142.233154
<<<(64, 64), (32, 32)>>>	0.031200	15.001823	76.867264	1469.816406	3107.387939
<<<(128, 128), (32, 32)>>>	0.034176	18.386560	80.483261	1449.951538	3022.137207
<<<(256, 256), (32, 32)>>>	0.038144	31.649632	94.315842	1467.479370	2991.355713
<<<(512, 512), (32, 32)>>>	0.041120	84.128609	147.850754	1524.941895	3061.609131
<<<(1024, 1024), (32, 32)>>>	277.791901	293.115112	358.368591	1743.271729	3306.678223
CPU	4654.0	448429.0	2.12557e+06	5.68283e+07	8.77892e+07

Сравнение изображений



r = 0 r = 3



r = 10 r = 15

Выводы

Данный алгоритм применяется в сверточных сетях и в разных программах для работы с изображениями. Как видно из результатов сравнения, GPU дает значительный прирост по времени, особенно на больших изображениях. Так как в данной лабораторной работе использовалась двухпроходная версия алгоритма размытия по Гауссу, то сложность его O(w*h*r), стандартная версия имеет сложность O(w*h*r^2).