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: Днепров И. С.
Группа: 8О-407Б-17
Преподаватели: К.Г. Крашенинников,
А.Ю. Морозов

Москва, 2021

Условие

Цель работы:

Научиться использовать *GPU* для обработки изображений. Использование текстурной памяти.

Формат изображений. Изображение является бинарным файлом со следующей структурой:

width(w)	*height(h)*	*r* *g* *b* *a*	*r* *g* *b* *a*	*r* *g* *b* *a*	...	*r* *g* *b* *a*	*r* *g* *b* *a*
4 байта, *int*	4 байта, *int*	4 байта, значение пикселя [1, 1]	4 байта, значение пикселя [2, 1]	4 байта, значение пикселя [3, 1]	…	4 байта, значение пикселя [*w* - 1, *h*]	4 байта, значение пикселя [*w*, *h*]

В первых восьми байтах записывается размер изображения, далее, построчно все значения пикселей, где

r – красная составляющая цвета пикселя

g – зелёная составляющая цвета пикселя

b – синяя составляющая цвета пикселя

a – значение альфа-канала пикселя

Пример картинки размером 2 на 2, синего цвета, в шестнадцатеричной записи:

```
02000000 02000000 0000FF00 0000FF00 0000FF00 0000FF00
```

Студентам предлагается самостоятельно написать конвертер на любом языке программирования для работы с вышеописанным форматом.

В данной лабораторной работе используются только цветовые составляющие изображения (*r g b*), альфа-канал не учитывается. При расчетах значений допускается ошибка в ±1. Ограничение: $w < 2^{16}$ и $h < 2^{16}$. В пограничном случае, необходимо «расширять» изображение за его границы, при этом значения соответствующих пикселей дублируют граничные. То есть, для любых индексов *i* и *j*, координаты пикселя $\left[i_p, j_p\right]$ будут определятся следующим образом:

$$i_p := \max\left(\min\left(i, h\right), 1\right), \quad j_p := \max\left(\min\left(j, w\right), 1\right)$$

Вариант 1. Гауссово размытие

Необходимо реализовать размытие по Гауссу с линейной сложностью от радиуса размытия (двухпроходный алгоритм).

Входные данные. На первой строке задается путь к исходному изображению, на второй, путь к конечному изображению. На следующей строке, целое число r – радиус размытия, $wh \leq 5*10^7, 0 \leq r < 1024$.

Примеры работы алгоритма:

Входной файл	hex: in.data	hex: out.data
in.data out.data 1	03000000 03000000 00010200 03040500 06070800 08070600 05040300 02010000 00000000 00140000 00000000	02000000 02000000 03030400 05060600 05050500 04040400
in.data out.data 1	01000000 05000000 00000000 00000000 64646400 00000000 00000000	01000000 05000000 00000000 1B1B1B00 2D2D2D00 1B1B1B00 00000000
in.data out.data 5	03000000 03000000 00000000 00000000 00000000 00000000 FFFFFF00 00000000 00000000 00000000 00000000	03000000 03000000 02020200 02020200 02020200 02020200 03030300 02020200 02020200 02020200 02020200

Программное и аппаратное обеспечение

GPU

Название: GeForce GT 545
Размер глобальной памяти: 3150381056
Размер константной памяти : 65536
Размер разделяемой памяти: 49152
Регистров на блок: 32768
Максимум потоков на блок: 1024
Размер варпа: 32
Максимальные размеры блока: 1024 x 1024 x 64
Максимальные размеры сетки: 65535 x 65535 x 65535
Количество мультипроцессоров : 3

CPU

Название: Intel Core i7-3770

Частота: 3.40GHz

Размер кеша: 8192 KB

Количество ядер: 4

Количество потоков: 8

MEM

Размер: 15 GB

Тип: ddr3

Прочее
OS: Linux Ubuntu 16.04.6
Редактор: Atom

Метод решения

В данной версии алгоритма необходимо осуществить два прохода по изображению по следующим формулам:

$$f(x,y) = \sum_{u=-r}^{r} \frac{e^{-\frac{u^2}{2r^2}}}{l} image(x+u,y),$$

$$y(x,y) = \sum_{v=-r}^{r} \frac{e^{-\frac{v^2}{2r^2}}}{l} f(x,y+v),$$

$$l = \sum_{v=-r}^{r} e^{-\frac{v^2}{2r^2}}.$$

$y(x,y)$ – выходное отфильтрованное изображение. Входное изображение $image$ нужно поместить в текстурную память для увеличения скорости чтения пикселей. После вычисления f также можно туда переместить из глобальной памяти, но я не стал. Коэффициенты $\frac{e^{-\frac{u^2}{2r^2}}}{l}$ и $\frac{e^{-\frac{v^2}{2r^2}}}{l}$ я высчитал до обработки изображения и поместил в константную память для ускоренного к ним обращения, т.к. они не зависят от обрабатываемого потоком пикселя (x,y), а $0 \le r < 1024$ – следовательно они туда поместятся.

Описание программы

Работа программы начинается с функции main. В ней мы получаем входные данные, открываем файл, читаем из него, выделяем память под изображение и запускаем ядра calculate_f и calculate_y. Ядро calculate_f вычисляет промежуточные коэффициенты типа double (f в формуле). А calculate_y вычисляет рузультирующие значения пикселя. Макросы xp, yp предотвращают выход за границы, но если использовать текстурную память, то они бесполезны. R_MAX — максимальный радиус (нужен для предварительного вычисления коэффициентов на CPU). CSC — макрос инициализации ошибки и вызова call.

Сравнение времени исполнения

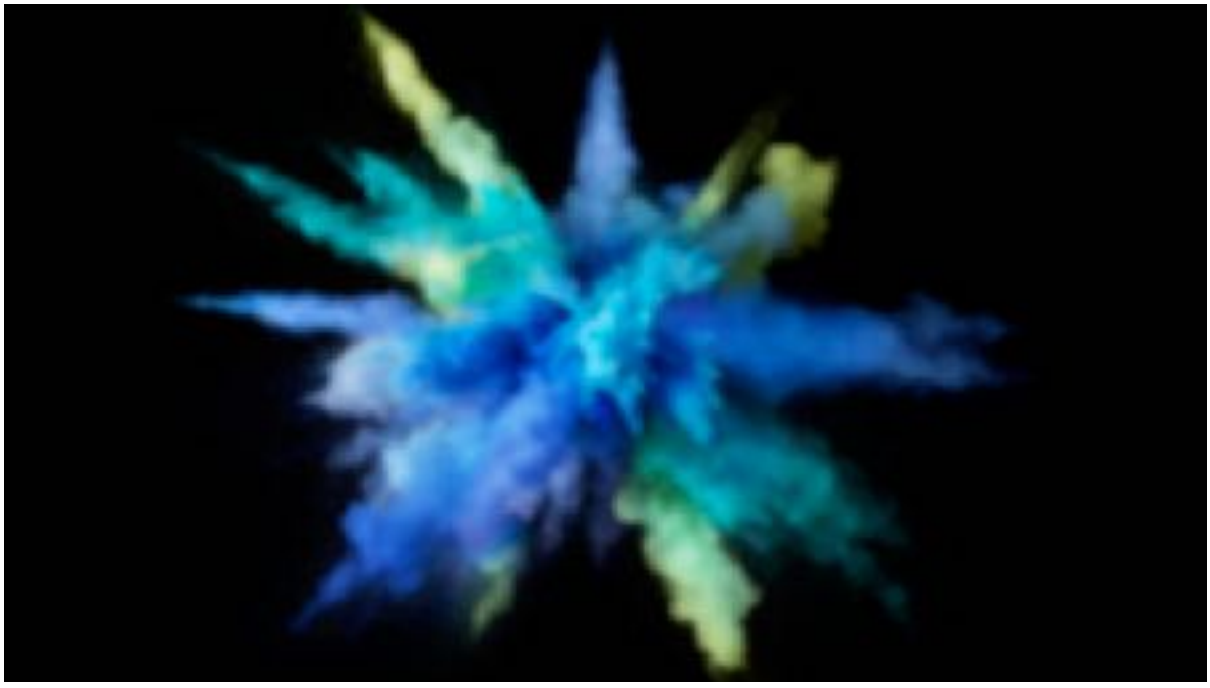
№ теста	size		*r*	Число блоков в сетке		Число потоков в блоке		Время, ms	
	x	y		x	y	x	y	GPU	CPU
1	2	2	1	1	1	8	8	1.23e-02	1.00e-10
2						16	16	8.38e-03	
3						24	24	1.00e-02	
4						32	32	1.20e-02	
5				16	16	8	8	7.97e-03	
6						16	16	9.63e-03	
7						24	24	1.00e-02	
8						32	32	1.11e-02	
9				24	24	8	8	8.00e-03	
10						16	16	8.86e-03	
11						24	24	1.02e-02	
12						32	32	1.17e-02	
13				32	32	8	8	8.00e-03	
14						16	16	9.25e-03	
15						24	24	9.63e-03	
16						32	32	1.12e-02	
17	5	5	2	1	1	8	8	6.48e-01	1.00e-09
18						16	16	5.63e-01	
19						24	24	6.11e-01	
20						32	32	6.52e-01	
21				16	16	8	8	2.72e-02	
22						16	16	5.74e-02	
23						24	24	4.05e-02	
24						32	32	6.82e-02	
25						8	8	2.93e-02	

26				24	24	16	16	5.13e-02	
27						24	24	4.10e-02	
28						32	32	6.91e-02	
29				32	32	8	8	3.19e-02	
30						16	16	6.45e-02	
31						24	24	4.24e-02	
32						32	32	7.01e-02	
33	100	100	5	1	1	8	8	3.11e+03	5.66e+03
34						16	16	2.74e+03	
35						24	24	2.74e+03	
36						32	32	2.74e+03	
37				16	16	8	8	8.60e+01	
38						16	16	8.64e+01	
39						24	24	8.62e+01	
40						32	32	8.61e+01	
41				24	24	8	8	8.57e+01	
42						16	16	8.15e+01	
43						24	24	8.19e+01	
44						32	32	8.26e+01	
45				32	32	8	8	8.62e+01	
46						16	16	8.24e+01	
47						24	24	8.42e+01	
48						32	32	8.28e+01	

Примеры работы алгоритма

$$r = 10$$

$r = 25$



$r = 50$

Выводы

Распараллеленный на видеокарте алгоритм Гауссова размытия изображений может пригодится в редакторах изображений или видео. Его можно применять для размытия всего изображения или части, например для аналога эффекта боке.

Сложности у меня возникли в том, что в материалах к данной ЛР не написано про нормировку коэффициентов, поэтому первая версия программы работала не верно.

В ходе лабораторной выяснилось, что на *GPU* лучше всего выполнять однотипные операции для большой размерности матриц. Т.к. они выполняются параллельно, то и суммарное время работы алгоритма заметно снижается.