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.С. Федоров
Группа: 8О-407Б
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 задания. 2 Медианный фильтр.
Необходимо реализовать медианный фильтр для изображения. Медианным элементом считается элемент с номером n / 2 в отсортированном массиве, для его нахождения использовать гистограмму и неполную префиксную сумму по ней.

Программное и аппаратное обеспечение

Графический процессор
Название: NVIDIA GeForce GTX 1050
Compute capability: 6.1
Объем графической памяти: 2147352576 байтов
Объем разделяемой памяти на блок: 49152 байтов
Объем регистров на блок: 65536
Размер варпа: 32
Максимальное количество потоков на блок: (1024, 1024, 64)
Максимальное число блоков: (2147483647, 65535, 65535)
Объем постоянной памяти: 65536 байтов
Число мультипроцессоров: 5

Процессор
Название: i5-8250U
Базовая тактовая частота процессора: 1,60 ГГц
Количество ядер: 4
Количество потоков: 8
Кеш L1: 256 Кб
Кеш L2: 1 Мб
Кеш L3: 6 Мб

Оперативная память
Тип: DDR4
Объем: 11.9 Гб
Частота: 2400 МГц

Програмное обеспечение
ОС: WSL2 (Windows 11)
IDE: Microsoft Visual Studio 2022 (аддон NVIDIA Nsight)
Компилятор: nvcc

Метод решения

Каждый поток работает с одним пикселем. Если пикселей изображения слишком много, чтобы распределить на каждый поток по одному, то потоки обрабатывают несколько пикселей последовательно беря их сдвигая координаты первого пикселя на целое число размера общей сетки потоков по горизонтали горизонтали и вертикали.
Индивидуально для пикселя осуществляется сортировка подсчётом значений всех соседних пикселей по трем каналам в радиусе r, который задается. Так как максимальное количество пикселей в таком секторе не точно может быть более, чем 2^{16}, то буду использовать для сортировки тип ushort4. После подсчета числа всех возможных значений выбираю элемент с номером n/2 по всем трем каналам. Делаю это в цикле приходясь по всем значениям и проверяя является сумма предыдущего числа элементов и числа текущих значения больше, чем n/2.

Описание программы

Разделение по файлам, описание основных типов данных и функций. Обязательно описать реализованные ядра.

Программа считывает путь и данные из входного файла. Затем выделяет массив на GPU и копирует туда все данные из файла. Также выделяется второй массив такого же размера для сохранения обработанных данных. Далее этот массив привязывается к текстурному объекту, который передается в качестве одного из параметров в функцию ядра. В ядре происходит вычисление обработанного изображения по алгоритму, описанному в разделе «Метод решения».
Код ядра:

```
__global__ void kernel(uchar4 * out, cudaTextureObject_t texObj, int w, int
h, int r) {
    int idx = blockDim.x * blockIdx.x + threadIdx.x;
    int idy = blockDim.y * blockIdx.y + threadIdx.y;
    int offsetx = blockDim.x * gridDim.x;
    int offsety = blockDim.y * gridDim.y;
    int x, y;
    uchar4 p;
```

```
    for (y = idy; y < h; y += offsety)
        for (x = idx; x < w; x += offsetx) {
            int left_boundry = max(0, x-r);
            int right_boundry = min(w-1, x+r);
            int top_boundry = max(0, y-r);
            int bottom_boundry = min(h-1, y+r);

            ushort counts[3][256];
            ushort* counts_x = counts[0];
            ushort* counts_y = counts[1];
            ushort* counts_z = counts[2];
            ushort n = (right_boundry-left_boundry+1)*(bottom_boundry-
top_boundry+1);

            for (int i = 0; i < 256; ++i) {
                counts_x[i] = 0;
                counts_y[i] = 0;
                counts_z[i] = 0;
            }

            for (int i = top_boundry; i <= bottom_boundry; ++i) {
                for (int j = left_boundry; j <= right_boundry; ++j) {
                    p = tex2D<uchar4>(texObj, j, i);
                    ++counts_x[p.x];
                    ++counts_y[p.y];
                    ++counts_z[p.z];
                }
            }

      uchar4 median_p;
            for (int k = 0; k < 3; ++k){
                    for (int i = 0; i < 256; ++i) {
                    if (i > 0)
                        counts[k][i] += counts[k][i-1];
                    if (counts[k][i] > n/2) {
                        if (k == 0)
                            median_p.x = i;
                        if (k == 1)
                            median_p.y = i;
                        if (k == 2)
                            median_p.z = i;
                        break;
                    }
                }
            }
```

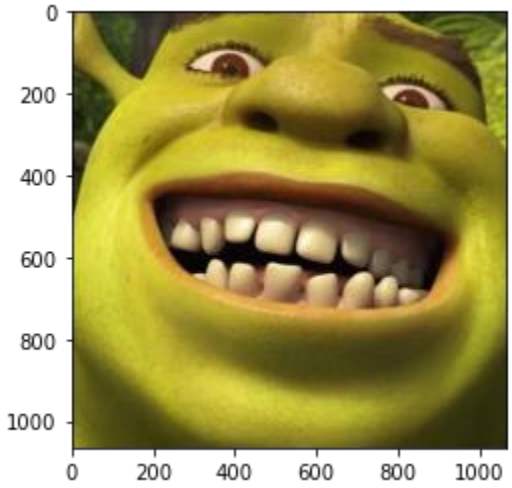
```
            out[y * w + x] = make_uchar4(median_p.x, median_p.y, median_p.z,
p.w);
        }
}
```

После отработки ядра обработанные данные копируются в оперативную память и сохраняются в файл.

Результаты (в миллисекундах)

	Размер теста (по вертикали и горизонтали)					
	2^8, r=3	2^9, r=4	2^{10}, r=4	2^{11}, r=4	2^{12}, r=4	2^{13}, r=4
GPU(CUDA) < (1, 1), (32, 1) >	230.962	1028.344	3796.368	15123.947	60653.101	243772.437
GPU(CUDA) < (1, 1), (32, 32) >	29.684	158.545	564.928	2019.496	7838.383	31367.386
GPU(CUDA) < (16, 16), (32, 32) >	**20.986**	116.011	**450.470**	**1721.783**	**6732.092**	**26796.488**
GPU(CUDA) < (32, 32), (32, 32) >	21.032	115.602	456.668	1806.620	6926.469	26925.703
GPU(CUDA) < (64, 64), (32, 32) >	21.304	**115.484**	455.919	1832.754	7234.110	27685.505
CPU	153.465	663.881	2771.734	11410.231	52573.986	195099.441

Визуальное представление результата работы фильтра:
Размер изображения: 1065 на 1065
Радиус фильтра: 16
Оригинал: Результат:



Выводы

Медианный фильтр может быть широко применен во всех областях, где широко распространена работа с растровыми изображениями. В частности, он может быть использован, как фильтр при обработке изображений в программах редактирования фото или как визуальный эффект в обработке видео. Реализация алгоритма относительно

простая, однако, следует постоянно держать в уме систему координат. Из-за того, что я неаккуратно просмотрел этот момент, порядок координат для вызова был поменян местами, что приводило к транспонированному результату.
С ростом числа блоков время работы программы уменьшается. Однако, сетка блоков более, чем 16 на 16 не дает ожидаемого прироста. Полагаю, что это связанно с тем, что потоков становится достаточно, чтобы на каждый пришлось не более одного пикселя, поэтому скорость работы начинает в больше мере зависеть от двойного вложенного цикла сортировки подсчетом.