МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Факультет «Информационные технологии и прикладная математика»
Кафедра «Вычислительная математика и программирование»

Лабораторная работа №2
по курсу «Программирование графических процессоров»

Обработка изображений на GPU. Фильтры.

Студент: Лысенко Д.А.
Группа: 8О-408Б
Преподаватели: К.Г. Крашенинников,
А.Ю. Морозов

Москва, 2019

Условие

1. *Цель работы:* научиться использовать GPU для обработки изображений. Использование текстурной памяти
2. *Вариант 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.

Входные данные. На первой строке задается путь к исходному изображению, на второй, путь к конечному изображению. На следующей строке, целое число r -- радиус размытия, $w*h \leq 5 * 10$, r , $w*h*r$.

Программное и аппаратное обеспечение

Compute capability: 5.0
Name: GeForce GTX 960M
Total Global Memory: -2147483648
Shared memory per block: 49152
Registers per block: 65536
Warp size: 32
Max threads per block: (1024, 1024, 64)
Max block : (2147483647, 65535, 65535)
Total constant memory: 65536
Multiprocessors count: 5
Процессор (CPU) – Intel Core i5-6300HQ 2.30GHz; 8 ГБ RAM;
OS Microsoft Windows 10 (x64)
CUDA V10.1

Метод решения

Принцип работы медианной фильтрации заключается в нахождении медианного элемента в окне заданного размера. Для нахождения медианного элемента применяется гистограмма и префиксная сумма по ней. В процессе сдвигов окна происходит обновление гистограммы. Основная сложность заключается в обработке пикселей, которые находятся на краях изображения т.к. возле них размер окна меняется.

Описание программы

Ядро программы, параллельные вычисления происходят по вертикали изображения и для каждой его цветовой составляющей:

```
__global__ void MedianFilter(uchar4 *res, int h, int w, int r) {
    int idy = threadIdx.y + blockDim.y * blockIdx.y;
    int offsety = blockDim.y * gridDim.y;
    for (int i = idy; i < h; i += offsety) {
        int histogram[256];
        int mid;
        initHistogram(histogram, mid, i, h, w, r);
        setColorComponentByIDx(res, i * w, findMedian(histogram, mid));
        for (int j = 1; j < w; ++j) {
            updateHistogram(histogram, mid, i, j, h, w, r);
```

```
            setColorComponentByIDx(res, i * w + j, findMedian(histogram, mid));
        }
    }
    return;
}
```

Функции инициализации и обновления гистограммы и поиска медианного элемента по префиксной сумме:

```
__device__   void initHistogram(int histogram[], int &mid, int idy, int h, int w, int r) {
    int hTop = MAX(idy - r, 0);
    int hBot = MIN(idy + r, h - 1);
    int wRight = MIN(r, w - 1);
    for (int i = 0; i < 256; ++i) {
        histogram[i] = 0;
    }
    mid = (hBot - hTop + 1) * (wRight + 1) / 2;
    for (int i = hTop; i <= hBot; ++i) {
        for (int j = 0; j <= wRight; ++j) {
            histogram[getColorComponentByIDx(j, i)] += 1;
        }
    }
    return;
}

__device__   void updateHistogram(int histogram[], int &mid, int idy, int idx, int h, int w, int r) {
    int hTop = MAX(idy - r, 0);
    int hBot = MIN(idy + r, h - 1);
    int wRight = MIN(idx + r, w - 1);
    int wLeft = MAX(idx - r, 0);
    mid = (hBot - hTop + 1) * (wRight - wLeft + 1) / 2;
    for (int i = hTop; i <= hBot; ++i) {
        if (idx - r - 1 >= 0) {
            histogram[getColorComponentByIDx(wLeft - 1, i)] -= 1;
        }
        if (idx + r <= w - 1) {
            histogram[getColorComponentByIDx(wRight, i)] += 1;
        }
    }
    return;
}

__device__  inline int findMedian(int histogram[], int mid) {
    int i = 0, count = 0;
    for (; count <= mid; ++i) {
        count += histogram[i];
    }
    return i - 1;
}
```

Применение фильтра:

Оценка производительности

R	CUDA <(3,64)> TIME	CUDA <(3,64),(1,64)> TIME	CUDA <(3,256),(1,512)> TIME	CUDA <(3,256),(1,1024)> TIME	CPU TIME
4	1225.60	121.51	121.63	121.54	1185
10	2433.13	218.81	217.97	227.28	2321

Вывод

В процессе выполнения лабораторной работы я изучил принцип работы медианной фильтрации. Медианный фильтр позволяет эффективно избавляться от шума и битых пикселей в изображении, однако при плохом выборе радиуса окна картинка становится размытой и “мультяшной”.

Для поиска медианного значения использовалась гистограмма, которая позволяет ускорить поиск медианного значения т.к. при каждом сдвиге окна у гистограммы обновляются лишь значения крайних элементов окна. Найти средний элемент по гистограмме не составляет труда, достаточно лишь знать текущий размер окна, по которому составлена гистограмма.

Программа выполняется параллельно только по вертикали изображения и для каждой цветовой составляющей. По горизонтали процесс парализации изображения оказался трудно реализуем т.к. тогда бы или терялся весь смысл применения гистограммы или пришлось бы разбивать изображение на блоки.