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: Симонов С.Я.
Группа: 8О-406Б-18
Преподаватели: К.Г. Крашенинников,
А.Ю. Морозов

Москва, 2021

Условие

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 ≤ 5 * 10 , r , w*h*r .

Цель работы: Научиться использовать GPU для обработки изображений. Использование текстурной памяти.

Вариант 2.

Программное и аппаратное обеспечение

GPU:

Compute capability : 6.1
Name : GeForce GTX 1050
Total Global Memory : 2096103424
Shared memory per block : 49152
Registers per block : 65536
Max threads per block : (1024, 1024, 64)
Max block : (2147483647, 65535, 65535)
Total constant memory : 65536
Multiprocessors count : 5

Сведения о системе:

Операционная система: Ubuntu 14.04 LTS 64-x битная
Рабочая среда: nano
Компилятор: nvcc

Метод решения

Изображение подается на вход программе в виде бинарного файла. Сначала считываются размеры изображения, а затем непосредственно само изображение.В данной лабораторной работе использовалась текстурная память память, она имеет существенное преимущество в работе благодаря кэшированию, поэтому выделяем ее на девайсе. Каждый пиксель обрабатывается в отдельном потоке, все потоки обрабатывают пиксель в пределах заданного радиуса и алгоритму задания. После того, как было обработано изображение оно выводится в бинарный файл

Описание программы

Ядро было реализовано исходя из алгоритма данного на лекции. Для каждого пикселя выделяется отдельный поток. В потоке пиксель обрабатывается в пределах заданного радиуса. Поскольку по условию задания вылезать за пределы заданного изображения нельзя, были определены граничные условия, не позволяющие этого сделать. Затем

были созданы массивы заполненные нулями, эти массивы нужны для нахождения медианы. Затем мы высчитываем позицию на которой располагается медиана. Вычисляем для каждого цвета медиану. И присваиваем текущему пикселю конечный результат..

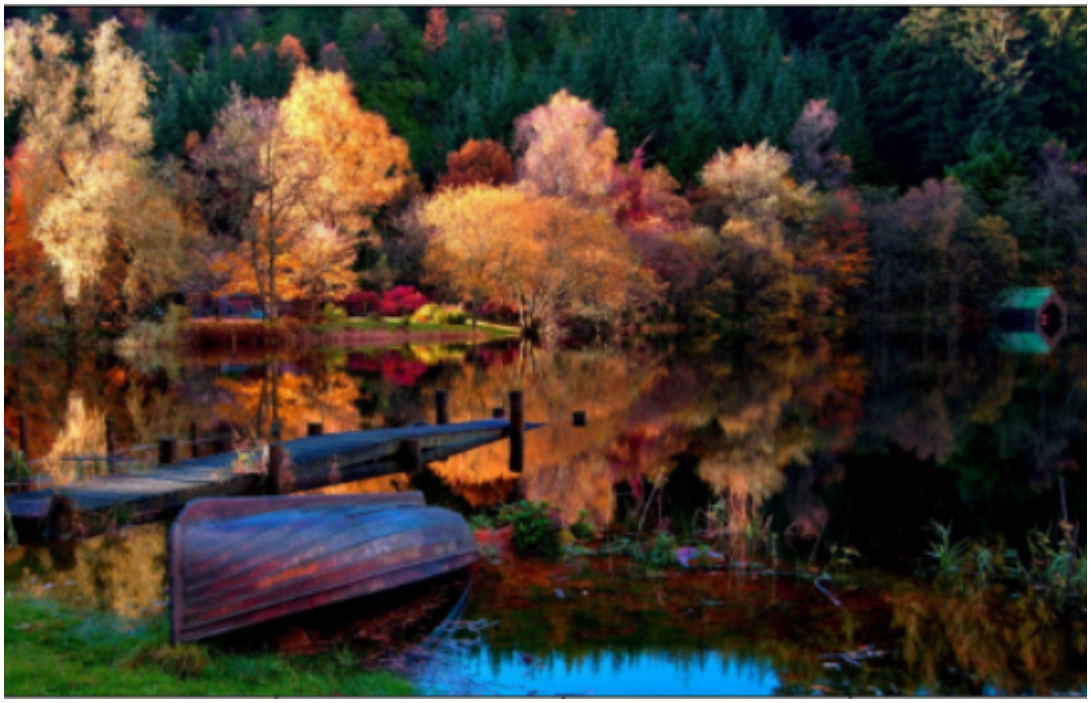
```
__global__ void kernel(uchar4 *out, int w, int h, int r) {
	int idx = blockDim.x * blockIdx.x + threadIdx.x;
	int idy = blockDim.y * blockIdx.y + threadIdx.y;
	int offsetx = blockDim.x * gridDim.x;
	int offsety = blockDim.y * gridDim.y;
	int x, y, i, j, x_0, y_0, x_n, y_n, pos;
	int counts1[257], counts2[257], counts3[257];
	int k;
	uchar4 p_1;

	for(y = idy; y < h; y += offsety) {
		for(x = idx; x < w; x += offsetx) {
			x_0 = max(0, x - r);
			x_n = min(w - 1, x + r);
			y_0 = max(0, y - r);
			y_n = min(h - 1, y + r);
			memset(counts1, 0, sizeof(int) * 257);
			memset(counts2, 0, sizeof(int) * 257);
			memset(counts3, 0, sizeof(int) * 257);
			for (i = x_0; i <= x_n; ++i) {
				for (j = y_0; j <= y_n; ++j) {
					p_1 = tex2D(tex, i, j);
					counts1[p_1.x]++;
					counts2[p_1.y]++;
					counts3[p_1.z]++;
				}
			}
			pos = (x_n - x_0 + 1) * (y_n - y_0 + 1) / 2;
			uchar4 res = make_uchar4(0, 0, 0, 0);
			k = 0;
			for (i = 0; i < 257; ++i) {
				k += counts1[i];
				if (k >= pos + 1) {
					res.x = i;
					break;
				}
			}
			k = 0;
			for (i = 0; i < 257; ++i) {
				k += counts2[i];
				if (k >= pos + 1) {
					res.y = i;
					break;
```

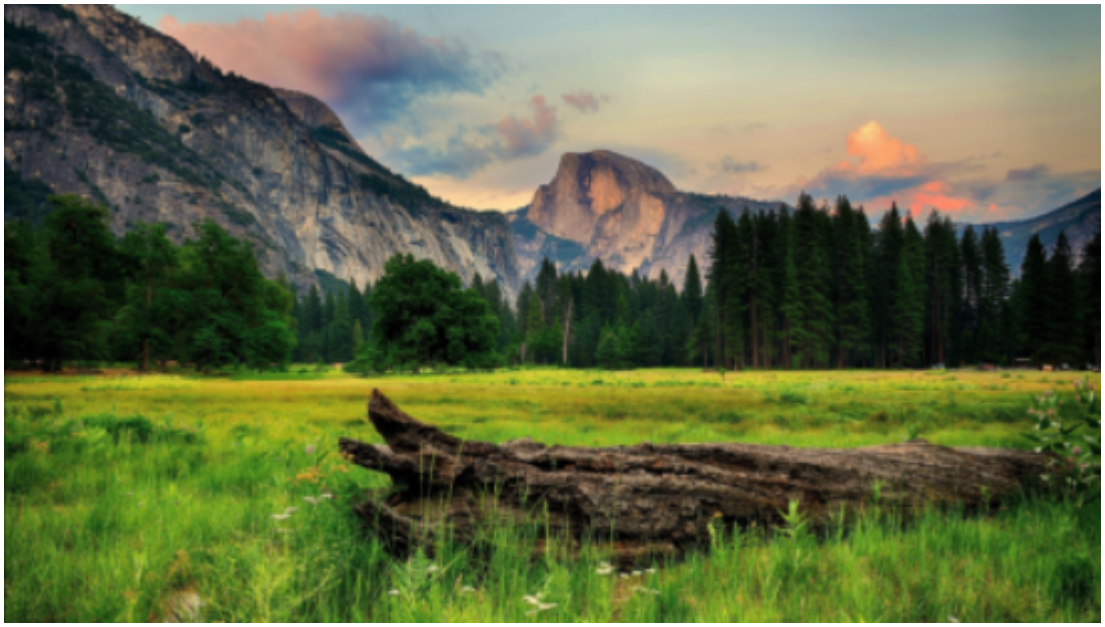
```
                        }
                    }
                    k = 0;
                    for (i = 0; i < 257; ++i) {
                        k += counts3[i];
                        if (k >= pos + 1) {
                            res.z = i;
                            break;
                        }
                    }
                    out[y * w + x] = make_uchar4(res.x, res.y, res.z, 0);

                }
            }
}
```

Результаты

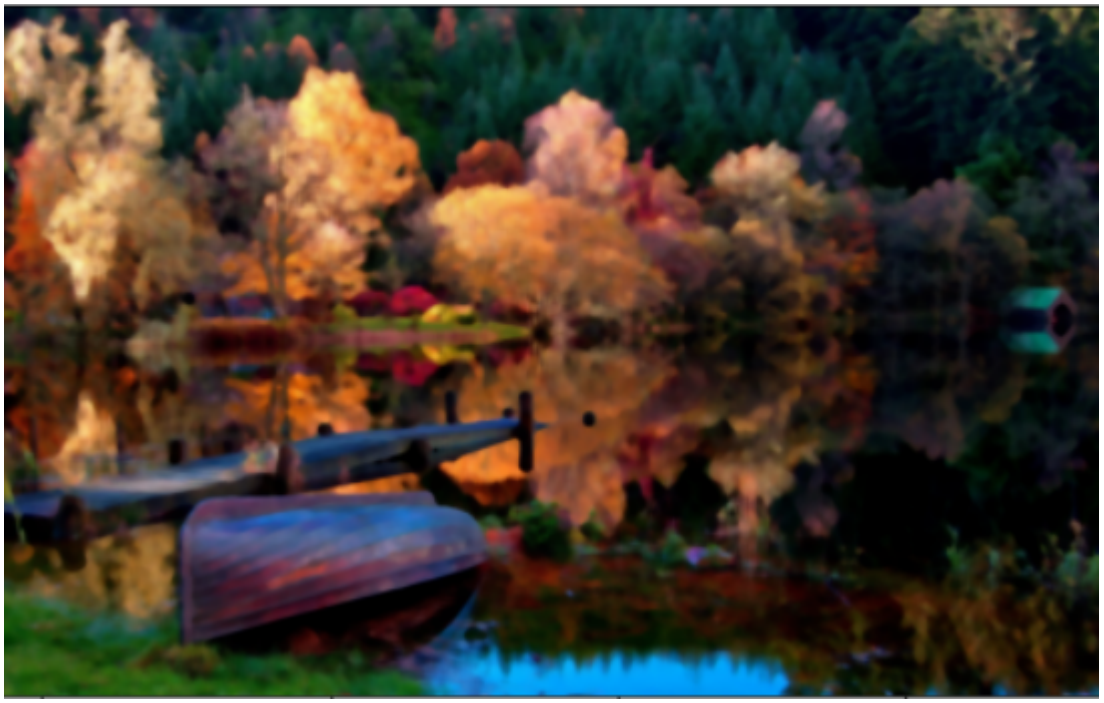
Входные изображения:

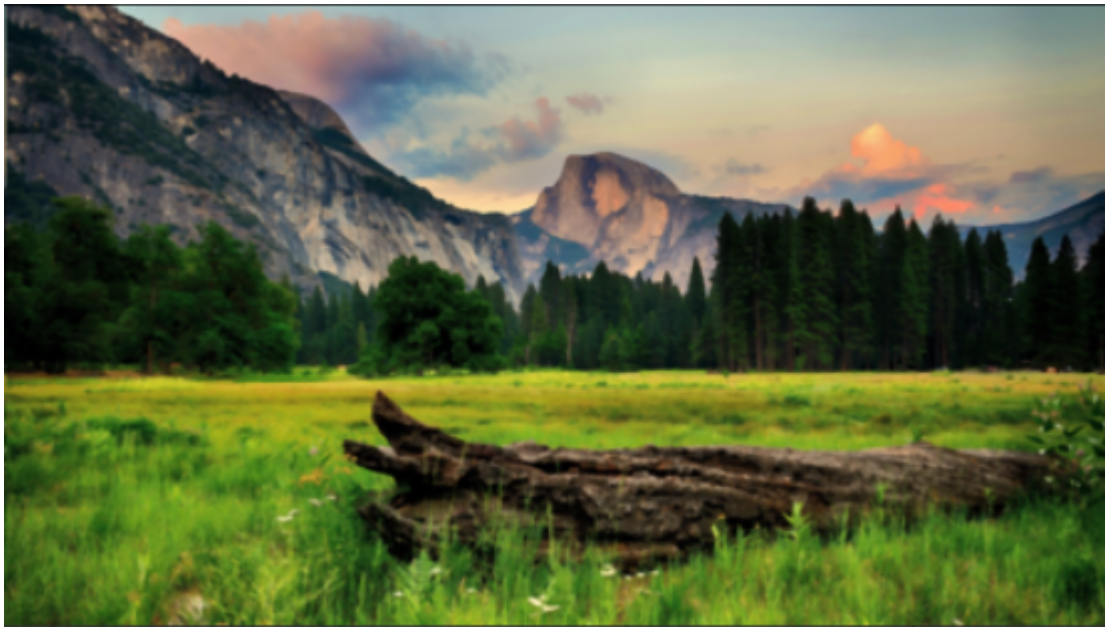

IDEAS





Выходные изображения:





	CPU	GPU	CPU	GPU
	1		5	
2795520	1008.88	449.52	11473.7	1883.03
2304000	836.897	441.18	9519.25	2927.92
8294400	3091.31	1644.53	34832.7	10436.58
33177600	12265	5457.84	144396	22630.93

Выводы

Выполнив данную лабораторную работу, я реализовал алгоритм медианного фильтра, который широко применяется в цифровой обработке сигналов и изображений для уменьшения уровня шума.

Алгоритмы обработки изображений хорошо распараллеливаются, поскольку подаются программе как двумерный массив.

В ходе данной лабораторной работы возникли определенные проблемы с обработкой выхода за изображении, чего не должно быть по установленным условиям задачи.С решением проблемы помог Морозов Александр Юрьевич. Также возникла проблема с нахождением медианного значения.