МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Информационные технологии и прикладная математика»
Кафедра 810 «Информационные технологии в моделировании и управлении»

Лабораторная работа №2
по курсу «Основы Python, Java и Scala, платформы CUDA для анализа данных»

Обработка изображений на GPU. Фильтры.

Выполнил: А.С.Бобряков
Группа: М8О-103М-19
Преподаватель: А.Ю. Морозов

Москва, 2020

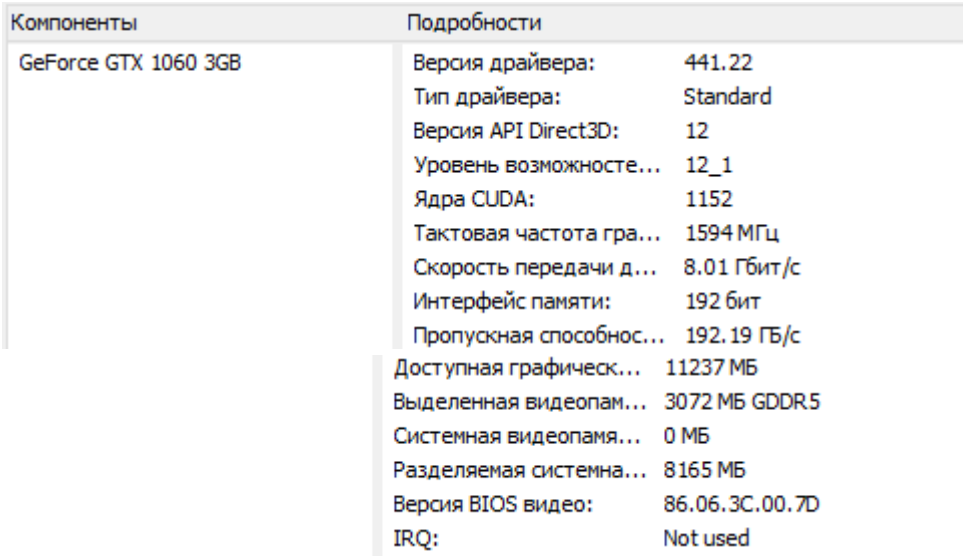
Условие

Необходимо реализовать избыточную выборку сглаживания. Исходное изображение представляет собой “экранный буфер”, на выходе должно быть сглаженное изображение, полученное уменьшением исходного.

Вариант 4. SSAA.

Программное и аппаратное обеспечение

Видеокарта: NVIDIA GeForce GTX 1060 3Gb



Компоненты	Подробности	
GeForce GTX 1060 3GB	Версия драйвера:	441.22
	Тип драйвера:	Standard
	Версия API Direct3D:	12
	Уровень возможносте...	12_1
	Ядра CUDA:	1152
	Тактовая частота гра...	1594 МГц
	Скорость передачи д...	8.01 Гбит/с
	Интерфейс памяти:	192 бит
	Пропускная способнос...	192.19 ГБ/с
	Доступная графическ...	11237 МБ
	Выделенная видеопам...	3072 МБ GDDR5
	Системная видеопамя...	0 МБ
	Разделяемая системна...	8165 МБ
	Версия BIOS видео:	86.06.3C.00.7D
	IRQ:	Not used

Процессор: Intel® Core™ i7-8700K CPU @ 3.70GHz

Другое: ОС Windows, IDE – Clion EAP,

Метод решения

Решение выполнено путем “свертки” всех пикселей внутри скользящего окна по матрице изображения. Размеры окна выбирались исходя начальных и конечных размеров изображения. Свертка реализовывалась среднеарифметическим значением исходных пикселей окна.

Описание программы

В программе использовано ядро для реализации основной логики приложения. Код ядра описан на листинге 1.

```
__global__ void kernel(uchar4 *out, int w, int h, int delta_w, int delta_h) {
    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;
    for (y = idy; y < (h/delta_h); y += offsety) {
        for (x = idx; x < (w/delta_w); x += offsetx) {
            int xx = 0;
            int yy = 0;
            int zz = 0;
            int ww = 0;
            for (int inner_x = x*delta_w; inner_x < x*delta_w + delta_w; inner_x++) {
                for (int inner_y = y*delta_h; inner_y < y*delta_h + delta_h; inner_y++) {
                    xx += (tex2D(tex, inner_x, inner_y)).x;
                    yy += (tex2D(tex, inner_x, inner_y)).y;
                    zz += (tex2D(tex, inner_x, inner_y)).z;
                    ww += (tex2D(tex, inner_x, inner_y)).w;
                }
            }
            out[y*(w/delta_w) + x] = make_uchar4(
                    x: xx/(delta_h*delta_w),  y: yy/(delta_h*delta_w),  z: zz/(delta_h*delta_w),  w: ww/(delta_h*delta_w)
            );
        }
    }
}
```

Листинг 1 – Код ядра программы.

Результаты

Пример исходной картинки размером 1005x558 изображен на рисунке 1.

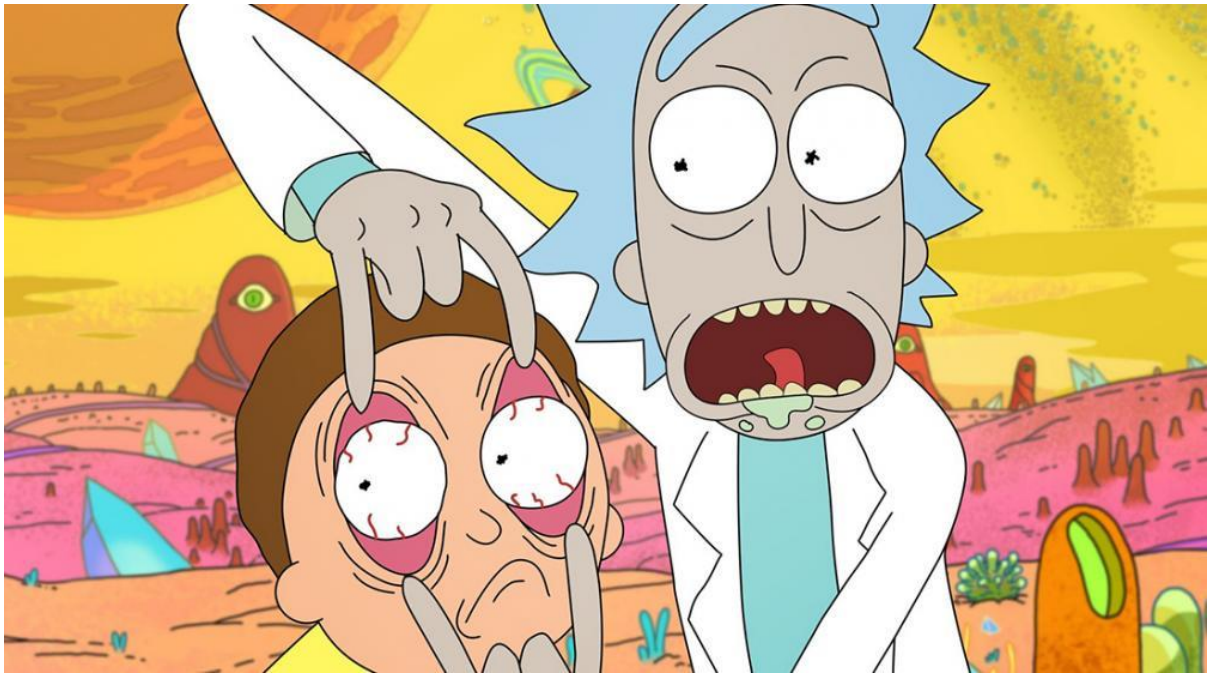


Рисунок 1. Исходное изображение.

Результат сглаживания SSAA под конечный размер 201x93 изображен на рисунке 2.

Рисунок 2 – Результат сглаживания SSAA.

Сравнение исходного и результирующего изображения показано на рисунке 3.

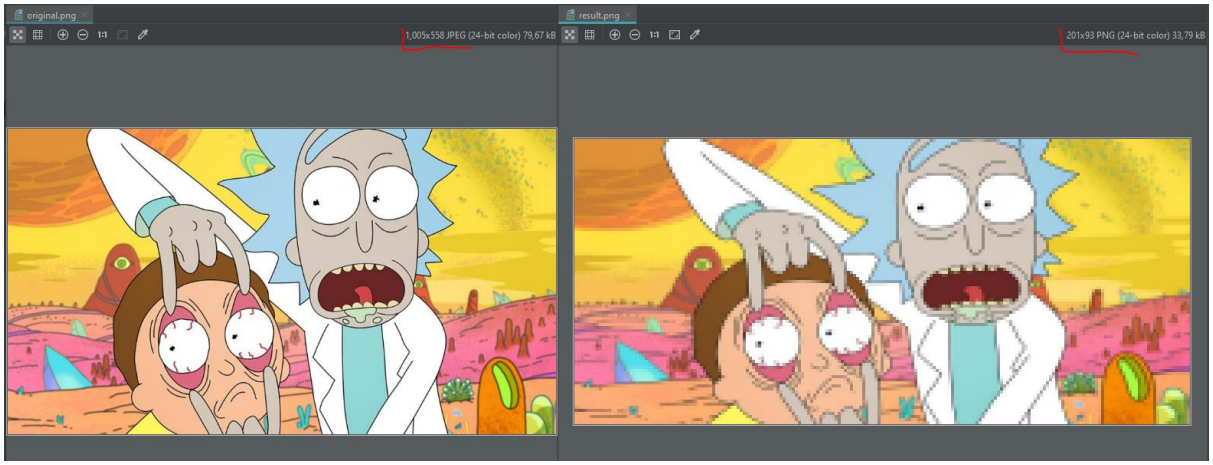


Рисунок 3 – Сравнение исходного и сглаженного изображений.

Время работы ядра в зависимости от конфигурации представлены в Таблице 1.

Таблица 1. Время выполнения ядра программы в зависимости от конфигурации.

Число потоков / Число блоков	32	128	512
32	0.077728	0.068608	0.144384
128	0.063488	0.141152	0.453632
512	0.153600	0.463680	1.702528

На CPU время выполнения 0.12.

Выводы

В ходе выполнения лабораторной работы был исследован метод сглаживания SSAA до уровня его программной реализации на CUDA.