МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

Лабораторная работа № 5

Работа с текстурной памятью

Студентка: Старшинова С.Н.
Группа: 08-408
Преподаватель: Семенов С. А.

МОСКВА, 2012

Содержание

Постановка задачи

Написать программу, которая:
– должна выполняться параллельно
– должна распределять память
– должна использовать векторные операции
Вариант 12: Идентифицировать движение в кадре

Алгоритм

• Пусть есть два изображения, обозначим их *I*, *J*.
Зададим объявление текстур как глобальных переменных и свяжем их с требуемой областью глобальной памяти.

• Пусть *u* — точка на *I*. Найдем точку на J, соответствующую u. Для этого необходимо найти вектор оптического потока (Vx, Vy). Он должен быть решением системы, состоящей из следующих уравнений:
Ix(q1)*Vx + Iy(q1)*Vy = - It(q1)
...
Ix(qn)*Vx + Iy(qn)*Vy = - It(qn), где Ix, Iy, It – частные производные по x, y, t, а qi – точки изображения, производные в точках считаются в отдельных нитях

• Решением этой системы является

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_i I_x(q_i)^2 & \sum_i I_x(q_i) I_y(q_i) \\ \sum_i I_x(q_i) I_y(q_i) & \sum_i I_y(q_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x(q_i) I_t(q_i) \\ -\sum_i I_y(q_i) I_t(q_i) \end{bmatrix}$$

• Результатом является вектор (Vx, Vy)

Реализация

```
__global__ void convertToGrey(unsigned char *d_in, float *d_out, int N)
{
    int idx = blockIdx.x*blockDim.x + threadIdx.x;
    if(idx < N)
        d_out[idx] = d_in[idx*3]*0.1144f + d_in[idx*3+1]*0.5867f + d_in[idx*3+2]*0.2989f;
}

__global__ void dX(float *in, int w, int h, float *out)
{
    int x = blockIdx.x*blockDim.x + threadIdx.x;
    int y = blockIdx.y*blockDim.y + threadIdx.y;

    if(x >= w || y >= h)
        return;

    int idx = y*w;

    int a = x-2;
    int b = x-1;
    int c = x;
    int d = x+1;
    int e = x+2;
```

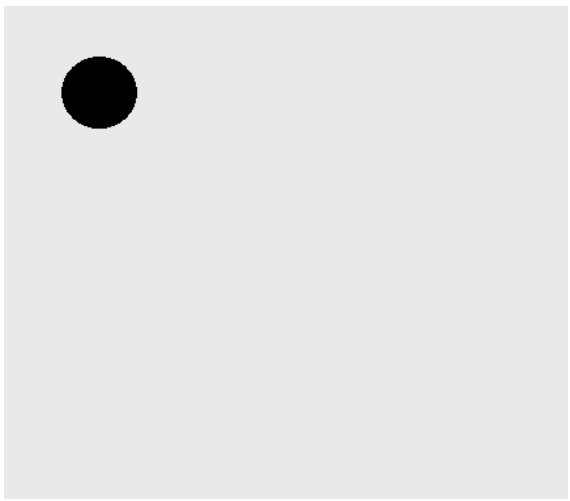
```
    if(a < 0) a = 0;
    if(b < 0) b = 0;
    if(c >= w) c = w-1;
    if(d >= w) d = w-1;

    out[y*w+x] = 0.0625f*in[idx+a] + 0.25f*in[idx+b] + 0.375f*in[idx+c] + 0.25f*in[idx+d]
+ 0.0625f*in[idx+e];
}

__global__ void dY(float *in, int w, int h, float *out)
{
    int x = blockIdx.x*blockDim.x + threadIdx.x;
    int y = blockIdx.y*blockDim.y + threadIdx.y;

    if(x >= w || y >= h)
        return;

    int a = y-2;
    int b = y-1;
    int c = y;
    int d = y+1;
    int e = y+2;

    if(a < 0) a = 0;
    if(b < 0) b = 0;
    if(c >= h) c = h-1;
    if(d >= h) d = h-1;

    out[y*w+x] = 0.0625f*in[a*w+x] + 0.25f*in[b*w+x] + 0.375f*in[c*w+x] + 0.25f*in[d*w+x]
+ 0.0625f*in[e*w+x];
}
```

Результаты

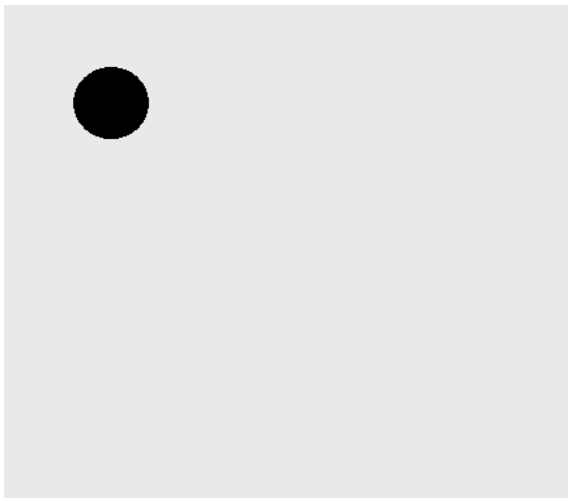
Входные данные: 2 изображения, представляющие собой последовательные кадры. Смещение предполагается небольшим.
Выходные данные: изображение, на котором обозначено движение и текстовый файл со значениями времени выполнение программы на CPU и GPU

Пример:

Входные данные

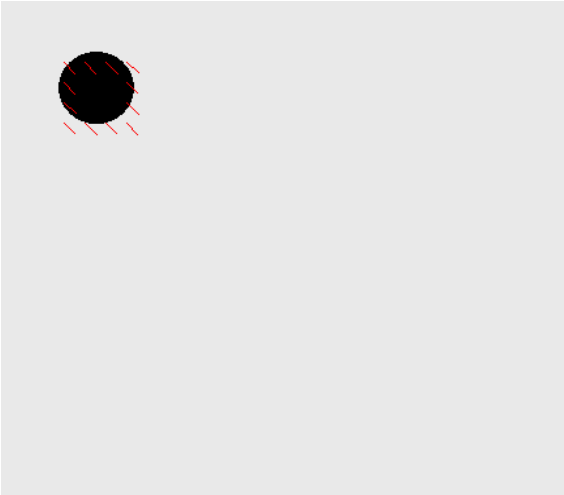


1



2

Выходные данные:



time spent executing by the GPU: 0.04500 milliseconds
time spent executing by the CPU: 0.27900 milliseconds

Сравнение CPU, GPU

Данная лабораторная работа выполнена с использованием: CUDA, Visual Studio 2008, OpenCV
Операционная система: Windows 7
Видеокарта: NVIDIA GeForce GT 320
Процессор: Intel Core i3-540

За счет параллельного выполнения программы достигается значительное преимущество по времени.

Выводы

В данной лабораторной работе используется алгоритм Лукаса-Канаде. Он основан на поиске вектора оптического потока. Оптический поток – это изображение движения наблюдателя относительно сцены. Методы, основанные на оптическом потоке, вычисляют движение между двумя кадрами, взятыми в момент времени t и (t + delta(t)) в каждом пикселе. Эти методы называются дифференциальными, они используют частные производные по времени и пространственным координатам.
Также возможно определить движение в кадре, используя блочные методы, основанные нахождении соответствующего блока текущего изображения на последующем.

Список литературы

1. http://ru.wikipedia.org/wiki/Алгоритм_Лукаса_—_Канаде
2. http://robocraft.ru/page/opencv/
3. Боресков А.В., Харламов А.А. - основы работы с технологией CUDA