Assignment #3 Part 1

CSD2170 Programming Massively Parallel Processors, Fall 2022

Due Date:	As specified on the moodle
Topics covered:	Shader programming
Deliverables:	The submitted project file is the source code of your shader program (kirsch.comp). The file should be put in a folder and subsequently zipped according to the stipulations set out in the course syllabus.
Objectives:	Learn how to use shader language to write efficient code to achieve parallelism.

Programming Statement

This is a shader programming assignment. Students are expected to finish the programming for a given computation problem. This assignment is the implementation of image processing routines in GLSL. For your reference, a C implementation are provided.

Implementing the Kirsch Edge Detection Filter

The aim of this assignment is to implement a GLSL version of the Kirsch Filter, which is a kind of filter used in image processing for edge detection. The purpose of edge detection is to identify points in the image where there are discontinuities or major shifts in color change (hence detecting the "edges" of the image). This is useful for applications such as feature extraction or face detection etc. For example, the following pictures show the effect of edge detection.

Before Edge Detection

Kirsch Edge Detection

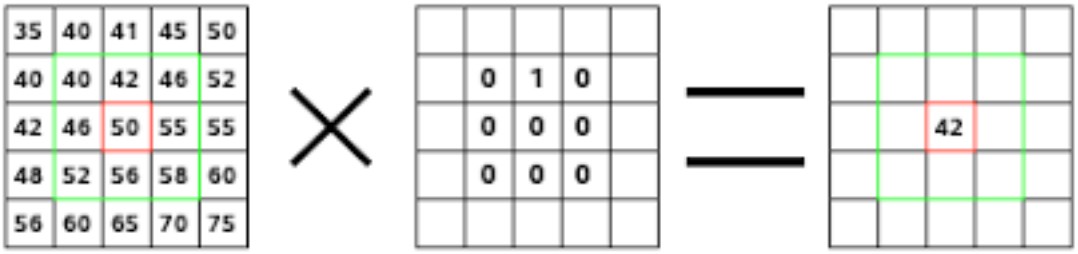
Kirsch Edge Detection Algorithm

At the heart of the Kirsch edge detection algorithm is one mask that is rotated 8 times and applied. The following are 8 masks that are used in Kirsch edge detection:

$$g^{(1)} = \begin{pmatrix} +5 & +5 & +5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{pmatrix}, g^{(2)} = \begin{pmatrix} +5 & +5 & -3 \\ +5 & 0 & -3 \\ -3 & -3 & -3 \end{pmatrix}, g^{(3)} = \begin{pmatrix} +5 & -3 & -3 \\ +5 & 0 & -3 \\ +5 & -3 & -3 \end{pmatrix}, g^{(4)} = \begin{pmatrix} -3 & -3 & -3 \\ +5 & 0 & -3 \\ +5 & 5 & -3 \end{pmatrix},$$

$$g^{(5)} = \begin{pmatrix} -3 & -3 & -3 \\ -3 & 0 & -3 \\ +5 & +5 & +5 \end{pmatrix}, g^{(6)} = \begin{pmatrix} -3 & -3 & -3 \\ -3 & 0 & +5 \\ -3 & 5 & 5 \end{pmatrix}, g^{(7)} = \begin{pmatrix} -3 & -3 & +5 \\ -3 & 0 & +5 \\ -3 & -3 & +5 \end{pmatrix}, g^{(8)} = \begin{pmatrix} -3 & +5 & +5 \\ -3 & 0 & +5 \\ -3 & -3 & -3 \end{pmatrix}.$$

The filter works by applying the masks as convolution matrices on every 3×3 pixels in the image. The following image shows how the convolution works in general:



We apply the mask to each 3×3 squares of the whole image in a similar fashion to how we apply the mask for the above. For all the convolution results after applying the eight Kirsch edge detection masks, we take the maximum convolution value to be the final value of the center pixel. The value of the ghost cell/pixel is regarded as 0 when the boundary pixels are computed. The student is invited to consult `kirsch_cpu.cpp` to find out more details.

The Assignment

In the given file `kirsch_cpu.cpp`, we have a basic implementation of the kirsch edge detection in a function with the following prototype:

```
void kirschEdgeDetectorCPU(
    const unsigned char *data_in,
    const int *mask,
    unsigned char *data_out,
    const unsigned channels,
    const unsigned width,
    const unsigned height
)
```

Please read the function and understand the code in the light of what was described in the previous section.

You are supposed to use GLSL shader language to optimize the basic implementation. The function, `main`, is supposed to be filled in by you in the file (kirsch.comp).

Please consider the following hints:

- There are two design options as shown in our lecture notes. You are required to implement the option of Design 1.

    1. Design 1: The size of each thread block matches the size of an output tile. All threads participate in calculating output elements. You should define a shared memory area that is (TILE_WIDTH + Mask_width - 1) × (TILE_WIDTH + Mask_width - 1) and Work Group (namely thread block size in CUDA) is TILE_WIDTH × TILE_WIDTH. Assume the shared memory area size is smaller than 2× TILE_WIDTH × TILE_WIDTH. Mask_width is the width of convolution mask, which is 3.
    2. Design 2: The size of each thread block matches the size of an input tile. Each thread loads one input element into the shared memory with the size of BLOCK_WIDTH × BLOCK_WIDTH. BLOCK_WIDTH should be O_TILE_WIDTH + MASK_width - 1, where O_TILE_WIDTH is the output tile width.

Rubrics

This assignment will be graded over 100 points. Here is the breakdown:

- If your code fails to compile, zero is awarded immediately.
- If you did not use shared memory and Design 1 to implement the optimized kirsch edge detection, zero is awarded immediately.
- Comments/Code Readability. Comments are important for the code to enhance readability. Up to 10 points will be deducted for insufficient comments.
- Correctness. You must ensure that the code works for images of all sizes. At least pass the test for 512×512 picture. Pictures with different sizes could be used in the evaluation.
- Latency. Frame rate will be tested for the given samples. 30 points are allocated for performance evaluation.