WITS UNIVERSITY

School of Computer Science and Applied Mathematics
UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG

COMS4040A & COMS7045A Assignment 3 – Report

Londani Zuma, 1637313, BDA Hons

28-05-2020

1 description

Serial implementation execute the convolution for each pixel using one thread.Design choice I used to implement the kernel using global memory is to have a thread for each output element of the output matrix.The thread will perform all the operations for it corresponding output element.All the threads reads the data from the input matrix from global memory.The global memory kernel outperformed the serial implementation and shared memory kernel.

Design choice I used to implement the kernel using shared memory is to have a thread for each output element of the output matrix.The thread perform all the operations for it corresponding output element.This kernel differs to global memory kernel because it first read all the data from global memory to shared memory.The thread will then read all the data from it corresponding shared memory.If the input element is not in it corresponding shared memory but not a ghost cell,the thread will read that input element from global memory.The shared memory outperformed serial implementation but was outperformed by both global memory kernel and texture memory kernel.The design I used for texture memory kernel is to have a thread for each output element of the output matrix.These threads read all the data required for compute the corresponding output element of the output matrix from the texture memory.The texture memory kernel outperformed all the kernels.

2 Performance comparison among all approaches

In this test all the kernel implementation where using the same convolution mask which is sharpening mask

	shared and const	global and const	serial	texture
lena_bw	0.168000	0.156000		0.078000
image21	0.162000	0.159000		0.078000
man	0.168000	0.166000		0.084000
mandrill	0.179000	0.154000		0.078000

Table 1: Performance comparison among all approaches

3 performance comparison among convolution mask

In this test,the kernel implementation that I used is shared memory kernel.

	Sharpening	edge Detection	Averaging
lena_bw	0.168000	0.170000	0.174000
image21	0.162000	0.161000	0.164000
man	0.168000	0.163000	0.163000
mandrill	0.179000	0.163000	0.167000

Table 2: performance comparison among convolution mask,shared memory kernel

In this test,the kernel implementation that I used is global memory kernel.

	Sharpening	edge Detection	Averaging
lena_bw	0.156000	0.163000	
image21	0.159000	0.157000	
man	0.166000	0.165000	
mandrill	0.154000	0.154000	

Table 3: performance comparison among convolution mask,global memory kernel

In this test,the kernel implementation that I used is texture memory kernel.

	Sharpening	edge Detection	Averaging
lena_bw	0.078000	0.085000	
image21	0.078000	0.078000	
man	0.084000	0.079000	
mandrill	0.078000	0.078000	

Table 4: performance comparison among convolution mask,texture memory kernel

4 Questions and answers

Assuming a $n \times n$ pixel image and $m \times m$ convolution mask

1. Since we have m^2 multiply-add operations for each pixel,the number of floating point operations performed is $2 \times m^2 n^2$
2. For a kernel using global memory the number of global memory reads is $m^2n^2 - 4\lfloor \frac{m}{2} \rfloor n$.For a kernel using shared memory the number of global memory reads is n^2
3. The number of global memory write performed by a kernel using shared memory is n^2
4. The performance will increase because the number of floating point operation per pixel will increase.

5 Discussion

The kernel using texture memory outperformed all the kernels in terms of performance using time as measure of performance .The kernel using shared memory outperformed the serial implementation but it was outperformed by both texture memory and global memory.Shared memory kernel have fewer global memory reads then the global memory kernel but shared memory needs an extra computation to check pixel elements that are outside of it shared memory and fetch those

in the global memory.This extra check is necessary to avoid the threads from attempting the read the shared memory of another block which will cause fatal error.This extra computation might be the one that makes global memory kernel to outperform the shared memory kernel.The global memory kernel outperformed both the serial implementation and shared memory kernel.

6 Images

Figure 1: Texture memory,sharpening mask,lena_bw

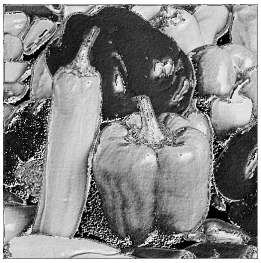

Figure 2: Texture memory,sharpening mask,image21

Figure 3: Texture memory,sharpening mask,man

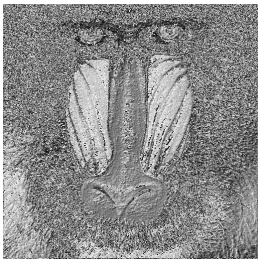

Figure 4: Texture memory,sharpening mask,mandrill

Figure 5: Texture memory,edge detection mask,lena_bw

Figure 6: Texture memory,edge detection mask,image21

Figure 7: Texture memory,edge detection mask,man

Figure 8: Texture memory,edge detection mask,mandrill

Figure 9: global memory,sharpening mask, lena_bw

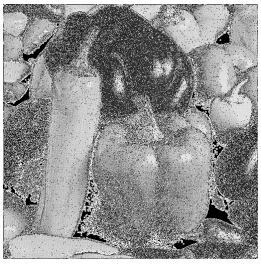

Figure 10: global memory,sharpening mask, image21

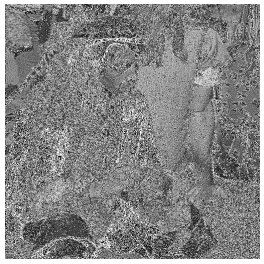

Figure 11: global memory,sharpening mask, man

Figure 12: global memory,sharpening mask, mandrill



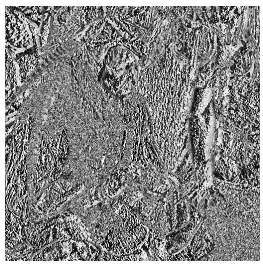
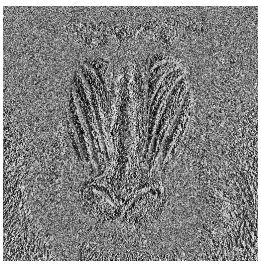

Figure 13: global memory,edge detection mask, lena_bw

Figure 14: global memory,edge detection mask, image21

Figure 15: global memory,edge detection mask, man

Figure 16: global memory,edge detection mask, mandrill


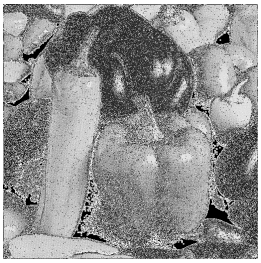
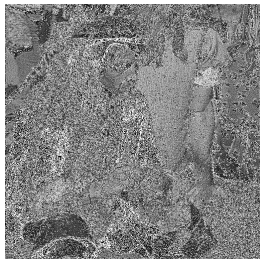


Figure 17: shared memory,sharpening mask, lena_bw

Figure 18: shared memory,sharpening mask, image21

Figure 19: shared memory,sharpening mask, man

Figure 20: shared memory,sharpening mask, mandrill



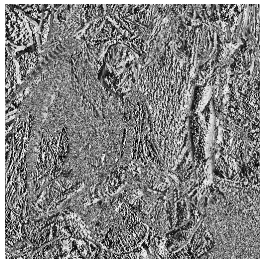
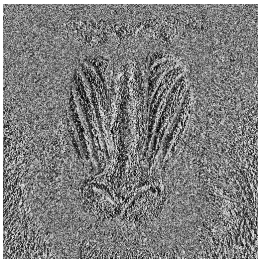

Figure 21: shared memory,edge detection mask, lena_bw

Figure 22: shared memory,edge detection mask, image21

Figure 23: shared memory,edge detection mask, man

Figure 24: shared memory,edge detection mask, mandrill