Cairo University

Faculty of Engineering

B.Sc. Program-Credit Hours system: CCE-C

CMPN461 - Selected Topics in Computer Engineering

Lab 4 Report

Presented to:

Eng. Mohamed Kotb

By**: Mohamed Gamal Hussein Ali**

Input size, tile size, mask width	Kernel 1	Kernel 2	Kernel 3
20000,4,3	4.4480us	5.8560us	25.248us
30000,4,3	4.7350us	6.1440us	35.488us
30000,4,17	7.1350us	14.495us	40.895us
30000,64,17	7.1350us	544.08us	6.1120us
30000,64,10	5.9190us	337.34us	5.8240us
100000,4,3	8.6390us	10.176us	111.17us
100000,4,10	11.296us	21.183us	121.98us
100000,64,3	8.5760us	140.19us	9.9200us
100000,64,10	11.424us	339.67us	10.624us
300000,4,3	18.432us	23.007us	325.63us
300000,4,10	26.239us	57.023us	360.60us
300000,64,17	37.184us	550.74us	26.207us
500000,4,3	27.936us	34.463us	539.48us
500000,4,10	40.735us	88.062us	599.60us
500000,64,10	40.543us	644.59us	34.143us
500000,64,17	58.879us	1.0478ms	41.023us
10^6,64,10	81.823us	1.2859ms	74.270us
10^6, 64,17	114.37us	2.0883ms	82.590us
10^6,256,10	81.662us	1.3528ms	73.407us
1.50^6,256,10	121.69us	2.5602ms	111.07us
1.50^6,512,10	122.08us	2.7011ms	117.57us

Comment:

Kernel 1 is faster for smaller sizes and smaller number of tiles – which is expected - but as both increase, performance of kernel 3 gets noticeably better, more so as the filter increase (more computations). Kernel 2 is the slowest.

Kernel 1 (No Tiling):
Performs well for smaller inputs because it has a simple structure, but as the input size and filter increase, performance degrades due to inefficient memory access and no use of shared memory. It struggles with larger problem sizes because it doesn't optimize memory access patterns.

Kernel 2 (Output Tiling):
Uses output tiling but is the slowest. It suffers from inefficient memory access because it focuses on organizing the output without optimizing how input data is loaded, leading to cache misses and global memory contention.

Kernel 3 (Input Tiling):
Performs best as input size and filter size increase. By loading chunks of input data into shared memory, it improves memory access patterns, reduces cache misses, and makes efficient use of shared memory, which results in better scalability and performance on larger problems.

In short, **Kernel 3** is the most efficient for larger datasets because it optimizes memory usage and access.