Rehan Tariq
22i-0965
CS-6A

Report for Question 1

OpenCL Case Study: SIMD-Optimized Image Convolution (Edge Detection Focus)

1. Overview and Approach

This project compares two implementations of image convolution for edge detection on grayscale images. The first is a scalar, CPU-based approach implemented in standard C/C++; the second is an accelerated version using OpenCL that leverages SIMD optimizations on GPU hardware. Both methods apply a vertical edge detection kernel to highlight edges in the image, but while the scalar version processes one pixel at a time, the OpenCL version processes many pixels concurrently using vectorized data types and optimized memory access patterns.

2. Introduction to Convolution

Image convolution is a fundamental operation in image processing. In this project, the convolution process involves sliding a small kernel (filter) over the image, multiplying overlapping values, and summing the results to form a new image that highlights edges. For edge detection, the kernel is designed to emphasize differences between adjacent pixel intensities, making vertical edges prominent.

3. Convolution Type Clarification

Two different kernels are used in the implementations:

- **Scalar Implementation:** Uses a simple vertical edge detection kernel with weights arranged as follows:

```
1, 0, -1
1, 0, -1
1, 0, -1
```

- **OpenCL Implementation:** Uses a Sobel operator for vertical edge detection. This kernel, with weights such as

`1, 0, -1`
`2, 0, -2`
`1, 0, -1`,
provides improved edge detection with smoothing benefits.

4. SIMD Optimization Details

4.1 Vectorization in OpenCL

The OpenCL version takes advantage of SIMD capabilities through:

- **Vectorized Data Types:**
  Using data types like `float4` allows the processing of multiple floating-point values simultaneously. For example, four adjacent pixels can be loaded and processed in parallel using functions like `vload4`, which improves computational throughput.

- **Work-Item Distribution:**
  The image is processed with a 2D global work size matching the image dimensions. Each work-item handles one pixel (or a block of pixels), thus harnessing the massive parallelism available on GPUs.

4.2 Memory Access Optimization

Key strategies include:

- **Memory Type Selection:**
  Input images and output buffers reside in global memory, while the kernel itself is stored in constant memory. This arrangement takes advantage of hardware optimizations for read-only data.

- **Coalescing Techniques:**
  Functions such as `vload4` ensure that adjacent pixels are loaded together, resulting in coalesced memory accesses and better bandwidth utilization.

- **Boundary Handling:**
  The kernel includes explicit checks to prevent out-of-bounds memory access, which ensures both correctness and optimized memory operations.

5. Performance Analysis

5.1 Execution Time Analysis

Both implementations measure execution time using high-resolution timers:

- The **scalar implementation** runs on the CPU and serves as a baseline.
- The **OpenCL implementation** launches a kernel with a 2D global size and waits for completion before measuring the elapsed time.

5.2 Expected Speedup Factors

Due to the inherent parallelism and SIMD processing, the OpenCL version is expected to deliver significant speedups compared to the scalar version. Key contributors to performance gains include:

- **Massive Parallelism:** Thousands of pixels are processed concurrently.
- **SIMD Execution:** Multiple data elements are handled per instruction.
- **Optimized Memory Access:** Coalesced reads/writes and the use of constant memory for kernels enhance performance.

The actual speedup depends on factors such as image size, GPU hardware, kernel complexity, and memory transfer overhead. In many cases, speedup factors of 10x to 100x or more can be observed.

6. Challenges and Solutions

6.1 Boundary Handling

Challenge:
Convolution near the image edges can access invalid memory regions.
Solution:
The scalar implementation pads the input image using functions like copyMakeBorder, while the OpenCL kernel incorporates boundary checking to ensure safe memory access.

6.2 Memory Transfer Overhead

Challenge:
Transferring data between the CPU and GPU can slow down performance.
Solution:

The OpenCL implementation minimizes transfers by sending the image data once, performing all computations on the GPU, and retrieving only the final result.

6.3 Normalization and Output Processing

Challenge:
The convolution output must be normalized to display properly.
Solution:
Both implementations include post-processing steps to normalize the output to a displayable range. In the OpenCL version, functions such as fabs() are used to enhance edge visibility.

6.4 Vectorization Limitations

Challenge:
Using vectorized data types like `float4` requires careful handling when image dimensions are not multiples of 4.
Solution:
A reduction step combines vector elements correctly, and additional optimization can be achieved by adjusting work-group sizes and handling edge cases separately.

7. Conclusion

The project successfully demonstrates two approaches to image convolution for edge detection:

- The **scalar implementation** provides a straightforward, baseline method.
- The **OpenCL implementation** exploits SIMD and parallelism to achieve substantial performance improvements.

Key optimizations include:

- Use of vectorized data types (e.g., `float4`).
- Optimized memory access with coalescing and constant memory.
- Effective distribution of computation across numerous work-items.

The OpenCL approach not only produces correct and visually appealing edge detection results but also significantly outperforms the scalar method, especially on larger images. Future improvements could involve further optimizing local memory usage, fine-tuning work-group sizes, and exploring additional kernel optimizations.

Screenshots:

PartA:

```
rehan@rehan:~/Assignments/PDC/i220965_A_A3/Scalar$ g++ -o c q1_multiFiles.cpp `pkg-config --cflags --libs opencv4`
rehan@rehan:~/Assignments/PDC/i220965_A_A3/Scalar$ ./c
Processing: flickr_wild_000006.jpg (512x512)
Execution time: 0.010661 seconds
Saved: /home/rehan/Assignments/PDC/i220965_A_A3/OutputQ1/processed_flickr_wild_000006.jpg
Processing: flickr_wild_000008.jpg (512x512)
Execution time: 0.009946 seconds
Saved: /home/rehan/Assignments/PDC/i220965_A_A3/OutputQ1/processed_flickr_wild_000008.jpg
Processing: c2.jpg (512x512)
Execution time: 0.010116 seconds
Saved: /home/rehan/Assignments/PDC/i220965_A_A3/OutputQ1/processed_c2.jpg
Processing: flickr_wild_000009.jpg (512x512)
Execution time: 0.010459 seconds
Saved: /home/rehan/Assignments/PDC/i220965_A_A3/OutputQ1/processed_flickr_wild_000009.jpg
Processing: flickr_wild_000002.jpg (512x512)
Execution time: 0.010168 seconds
Saved: /home/rehan/Assignments/PDC/i220965_A_A3/OutputQ1/processed_flickr_wild_000002.jpg
Processing: c1.jpg (512x512)
Execution time: 0.010108 seconds
Saved: /home/rehan/Assignments/PDC/i220965_A_A3/OutputQ1/processed_c1.jpg
Processing: flickr_wild_000011.jpg (512x512)
Execution time: 0.011107 seconds
Saved: /home/rehan/Assignments/PDC/i220965_A_A3/OutputQ1/processed_flickr_wild_000011.jpg
Processing: flickr_wild_000010.jpg (512x512)
Execution time: 0.010425 seconds
Saved: /home/rehan/Assignments/PDC/i220965_A_A3/OutputQ1/processed_flickr_wild_000010.jpg
Processing: flickr_wild_000003.jpg (512x512)
Execution time: 0.010199 seconds
Saved: /home/rehan/Assignments/PDC/i220965_A_A3/OutputQ1/processed_flickr_wild_000003.jpg
Processing: flickr_wild_000005.jpg (512x512)
Execution time: 0.010047 seconds
Saved: /home/rehan/Assignments/PDC/i220965_A_A3/OutputQ1/processed_flickr_wild_000005.jpg
Processing: flickr_wild_000007.jpg (512x512)
Execution time: 0.010334 seconds
Saved: /home/rehan/Assignments/PDC/i220965_A_A3/OutputQ1/processed_flickr_wild_000007.jpg

Processing complete! Total images processed: 11
rehan@rehan:~/Assignments/PDC/i220965_A_A3/Scalar$
```

PartB:

```
rehan@rehan:~/Assignments/PDC/i220965_A_A3/OpenCL$ ./convolution
Processed /home/rehan/Assignments/PDC/i220965_A_A3/DataSet/flickr_wild_000006.jpg in 0.00100552 seconds.
Saved: ./processed/flickr_wild_000006_processed.jpg
Processed /home/rehan/Assignments/PDC/i220965_A_A3/DataSet/flickr_wild_000008.jpg in 0.000635786 seconds.
Saved: ./processed/flickr_wild_000008_processed.jpg
Processed /home/rehan/Assignments/PDC/i220965_A_A3/DataSet/c2.jpg in 0.000823197 seconds.
Saved: ./processed/c2_processed.jpg
Processed /home/rehan/Assignments/PDC/i220965_A_A3/DataSet/flickr_wild_000009.jpg in 0.000805429 seconds.
Saved: ./processed/flickr_wild_000009_processed.jpg
Processed /home/rehan/Assignments/PDC/i220965_A_A3/DataSet/flickr_wild_000002.jpg in 0.00106131 seconds.
Saved: ./processed/flickr_wild_000002_processed.jpg
Processed /home/rehan/Assignments/PDC/i220965_A_A3/DataSet/c1.jpg in 0.000769538 seconds.
Saved: ./processed/c1_processed.jpg
Processed /home/rehan/Assignments/PDC/i220965_A_A3/DataSet/flickr_wild_000011.jpg in 0.000741986 seconds.
Saved: ./processed/flickr_wild_000011_processed.jpg
Processed /home/rehan/Assignments/PDC/i220965_A_A3/DataSet/flickr_wild_000010.jpg in 0.00104977 seconds.
Saved: ./processed/flickr_wild_000010_processed.jpg
Processed /home/rehan/Assignments/PDC/i220965_A_A3/DataSet/flickr_wild_000003.jpg in 0.001058 seconds.
Saved: ./processed/flickr_wild_000003_processed.jpg
Processed /home/rehan/Assignments/PDC/i220965_A_A3/DataSet/flickr_wild_000005.jpg in 0.000701638 seconds.
Saved: ./processed/flickr_wild_000005_processed.jpg
Processed /home/rehan/Assignments/PDC/i220965_A_A3/DataSet/flickr_wild_000007.jpg in 0.000725403 seconds.
Saved: ./processed/flickr_wild_000007_processed.jpg
rehan@rehan:~/Assignments/PDC/i220965_A_A3/OpenCL$
```