Lab 10

Rayane Guerou

November 11, 2024

1 Introduction

The Kuwahara filter is an image processing technique aimed at smoothing an image while preserving edges. It divides a region into four quadrants, selects the one with the lowest variance, and uses the mean color of that quadrant for the target pixel. We implemented the filter on the CPU, GPU, and GPU with shared memory to compare their performance.

2 CPU Implementation of the Kuwahara Filter

The CPU-based Kuwahara filter is applied using a nested loop for each pixel. The following function, `kuwahara_region_filter`, processes a subset of pixels in the local region.

The `apply_kuwahara_filter` function applies this filter to each pixel in the image. Below is the code for applying the filter.

3 GPU Implementation of the Kuwahara Filter

The GPU implementation uses CUDA to parallelize the processing of pixels. Each pixel is processed independently, allowing for faster execution.

The filter application is performed by calling the GPU function with specified CUDA grids and blocks.

4 GPU Implementation with Shared Memory

Using shared memory reduces global memory access by temporarily storing pixel values in fast local memory. This optimization improves performance for larger images.

5 Performance Comparison

To evaluate performance, we tested the three implementations with different `omega` values. Below are the results for each implementation, based on the neighborhood size.

5.1 Performance Results

- **CPU:** Computation time increases proportionally with region size due to nested loops.
- **GPU:** Significant improvement with faster processing times, even for larger regions.
- **GPU with Shared Memory:** This is the fastest implementation, utilizing shared memory to minimize global access.

6 CPU Result

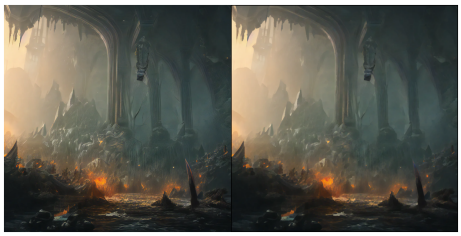

Figure 1: Output image using CPU implementation of the Kuwahara filter.

7 GPU Result

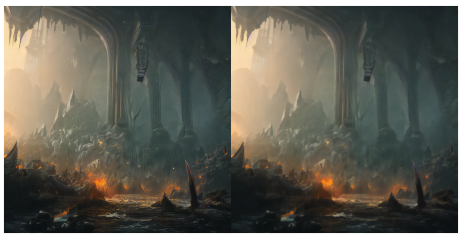

Figure 2: Output image using GPU implementation of the Kuwahara filter.

8 GPU with Shared Memory Result

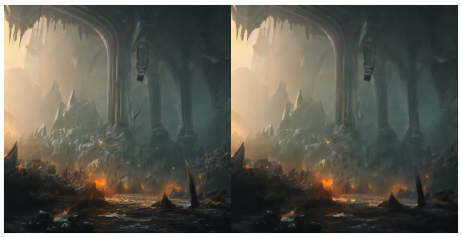

Figure 3: Output image using GPU with shared memory implementation of the Kuwahara filter.

9 Conclusion

We demonstrated how the Kuwahara filter can be optimized by leveraging GPU parallelism and shared memory. The results confirm that the GPU implementation with shared memory is the most efficient, particularly for large region sizes.