HPC Project Report

Accelerating KLT Algorithm

Version 2.0: Naive GPU Implementation

M. Abdullah Siddiqui | 23I-0617

M. Daniyal | 23I-0579

Moiz Ansari | 23I-0523

Introduction

This version focuses on the basic GPU implementation of the functions identified as suitable for acceleration. The primary objectives are establishing functional correctness, validating results against CPU, and measuring initial performance improvements without optimization.

GPU Function Implementation Analysis

Based on profiling data from the previous version, three functions were initially identified for GPU acceleration: _convolveImageVert(), _convolveImageHoriz(), and _interpolate(). However, after analyzing their code structures, only the first two were parallelized on the GPU, as they dominated runtime and are inherently parallel, i.e. each pixel can be processed independently.

The _interpolate() function, on the other hand, performs only single-point bilinear interpolation with low computational intensity but high memory overhead, making it unsuitable for GPU acceleration in this phase.

GPU Implementation Overview

The convolution functions were implemented in CUDA, assigning one GPU thread per output pixel to achieve full parallelism. Separate kernels were developed for horizontal and vertical convolutions, mirroring the CPU design to simplify debugging.
For each kernel, memory is allocated on the GPU, and data is transferred from host to device using cudaMemcpy. A 2D grid of 32×32 threads ensure efficient mapping between threads and pixels. Each thread reads neighboring pixels (horizontally or vertically) to compute weighted sums using the convolution kernel, after which results are copied back to host memory and GPU resources are freed.

Performance Results

The GPU performance was measured with Cuda events, whereas CPU performance was measured with standard clock functions. Each convolution function was executed for multiple image resolutions. The GPU demonstrates clear performance benefits on medium and large images.

Overall Metrics:

Total Convolution Calls: 126

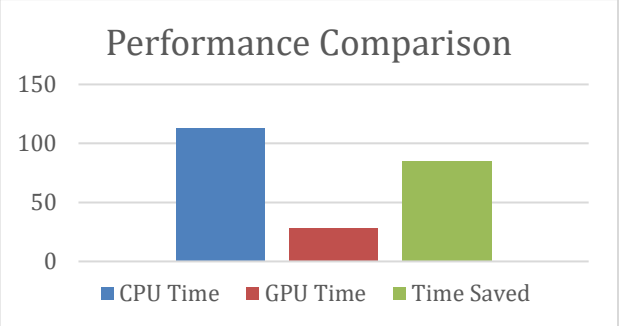
Total GPU Compute Time: 28.25 ms

Total CPU Compute Time: 113.24 ms

Overall Speedup (GPU vs CPU): 4.01x

Time Saved: 84.99 ms

Percentage GPU is Faster: 75.1%



Detailed Metrics:

Operation	Image Size	CPU Time (ms)	GPU Time (ms)	Speedup
Horizontal	320 × 240	~0.90	~0.26	3.4x
Vertical	320 × 240	~1.05	~0.27	3.9x

Both horizontal and vertical convolutions achieved over 3× speedup on average for medium-sized images.

Correctness Verification

Operation	Mean Error	Max Error	Accuracy Remarks
Horizontal	1.9×10^{-6}	4.6×10^{-5}	Excellent agreement
Vertical	1.8×10^{-6}	4.6×10^{-5}	Excellent agreement

Both GPU convolution kernels closely matched the CPU implementation, with mean pixel-wise errors around 1×10^{-6} and maximum absolute differences under 5×10^{-5}. These values are within single-precision floating-point rounding error, confirming that the GPU computations are numerically stable and match exactly the CPU results.

Conclusion

This version successfully achieved GPU-based acceleration for the convolution functions with verified correctness, showing mean errors within 10^{-6} and up to 4× speedup on medium-sized images, confirming CUDA reliability and efficiency. These outcomes establish a solid foundation for further optimization and memory-level enhancements in the next version.

GitHub Private Repository Link

https://github.com/MuhammaDaniyal/Complex_Computing_Problem