CS4110 - High Performance Computing with GPUs

Deliverable 03:

KLT Feature Tracking

(Optimized GPU Implementation)

Presented By:

Ali Aan Khowaja [23I-0708]
Wajih-Ur-Raza Asif [23I-0819]
Ismail Sheikh [23I-2524]

Submitted to:

Dr. Imran Ashraf

National University of Computing & Emerging Sciences

FAST, Islamabad

30th October, 2025

Abstract

This report details the performance results of our optimized GPU implementation (V3) for the Kanade-Lucas-Tomasi (KLT) feature tracker. Building upon the naive GPU version (V2), we focused optimizations on the convolution operations (`_convolveImageHoriz` and `_convolveImageVert`), which were identified as the dominant hotspots in larger datasets during D2 profiling. Key enhancements included tuning launch configurations for improved occupancy, communication optimizations to reduce host-device data transfers, and memory hierarchy improvements using shared memory for stencil computing to minimize global memory accesses.

Testing Results

Testing was conducted on datasets d1 (250 frames, 396 KB per frame) and d2 (250 frames, 8 MB per frame). For accurate benchmarking, timings were measured specifically for the offloaded GPU convolution segments during full application runs. These timings exclude high-intensive write operations performed by the CPU, which were not part of the parallelized workload. The results compare V3 (optimized GPU) against V1 (CPU) and V2 (naive GPU), focusing on example 3 as the most representative workload. All timings represent averages over multiple executions where applicable, to isolate the impact of optimizations.

Performance results on Dataset d2

On d2, the optimized GPU (V3) achieved an average timing of approximately **17.185 seconds** for the offloaded convolutions, compared to the CPU's (V1) total execution time of **396.4 seconds** for the same dataset. This demonstrates a substantial speedup of about **23x** for the parallelized components, primarily due to the use of shared memory.

This represents a significant improvement over the naive GPU implementation (V2), which took approximately **35.986 seconds** on the same d2 dataset, yielding roughly a **2x speedup** from V2 to V3 through targeted optimizations. These results highlight the GPU's efficiency for large-scale data, where overheads are minimized and parallelism is most effective.

Performance on Dataset d1: Multiple Kernel Launch Configurations

For dataset d1, we evaluated different launch configurations to optimize thread utilization and occupancy:

Kernel Launch Config	Total Execution Time
16x16	6.923 seconds
32x32	7.118 seconds
32x8	6.955 seconds
32x16	6.937 seconds
16x32	6.957 seconds

Table 1: Kernel launch configuration performance on Dataset d1 (V3)

The **16x16** configuration performed best overall, with minimal variation across options, indicating a good balance between occupancy and thread efficiency. Compared to V2's **35.986 seconds** on d2, V3 showed a

modest improvement of about **2.09x** for the offloaded part, thanks to tuned launches and communication overlaps.

Conclusion

This compares to the CPU's total execution time of 396 seconds for the dataset. The **23x speedup** for the convoluted portions stems largely from shared memory stencils, which efficiently handled the large frame sizes by reducing memory traffic.

The V3 optimizations markedly improved performance on larger datasets, where convolutions dominate. The combination of tuned launch configurations, reduced communication, and shared memory for stencils scales well with data size, making GPU ideal for high-resolution workloads. For smaller datasets, overheads may still favor CPU, but d1 (medium sized data) and d2 (jumbo sized data) results affirm the value of these techniques. Future efforts could integrate more advanced features like multi-stream processing.

GitHub Repo Link:

`https://github.com/aliaankhowaja/KLT-Using-CUDA`