CS4110 - High Performance Computing

Deliverable 03:

KLT Feature Tracking

Profiling

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

04 October, 2025

Contents

1. Executive Summary

This report points out the most resource-intensive functions in the Kanade–Lucas–Tomasi (KLT) feature tracker implementation that are good candidates for offloading to a GPU with CUDA. The analysis relies on profiling data from the `example3` application, which follows features across a series of images.

The main goal is to speed up the KLT algorithm by splitting tasks that can run in parallel and offloading them to a GPU for performance improvement. According to our profiling output, the following functions are the main contributors to the program's runtime and are very suitable for parallelization (Table 1):

Table 1: Functions contributing to runtime (profiling summary).

Function	**Percentage of Runtime**
Image Convolution (`_convolveImageHoriz` & `_convolveImageVert`)	∼42.50% (total)
`_computeGradientSum()`	∼22.15%
`_computeIntensityDifference()`	∼27.85%

Offloading these functions could parallelize ∼87% of the application's computational workload.

Side Note: We are thinking about implementing the sorting functionality using radix sort instead of quicksort as done in this algorithm; that would reduce sorting to $\mathcal{O}(n)$ time (for fixed key width).

2. Functions Recommended for GPU Offloading

2.1 Image Convolutions

Runtime Cost: `_convolveImageHoriz()` at 25.00%; `_convolveImageVert()` at 12.50%.

This portion of code is the biggest bottleneck in the entire program. Convolution is a classic, embarrassingly parallel problem: the value of each output pixel is calculated based on a small neighborhood of input pixels, and each pixel calculation is independent of the others. These two portions can be very easily offloaded to the GPU for a substantial speedup.

2.2 Feature Tracking (`_trackFeature` and sub-functions)

The running cost of feature tracking is around 49% combined from its most expensive sub-functions: `_computeIntensityDifference()` at 27% and `_computeGradientSum()` at 22%.

These two functions are perfect candidates for parallelization because they are highly parallelizable and account for a large portion of total runtime.

2.3 Conclusion

By offloading the four functions mentioned above we can achieve ∼89% boost in performance. This calculation is data dependent and may vary based on a different dataset, but for a rough ballpark we can assume that roughly one-third of the program is highly parallelizable.