Complex Computing Problem

OPTIMIZING A KLT FEATURE TRACKER USING GPU KERNELS

Menahil Ahmad 23I-0546
Hiyam Rehan 23i-0039

Experimental Setup

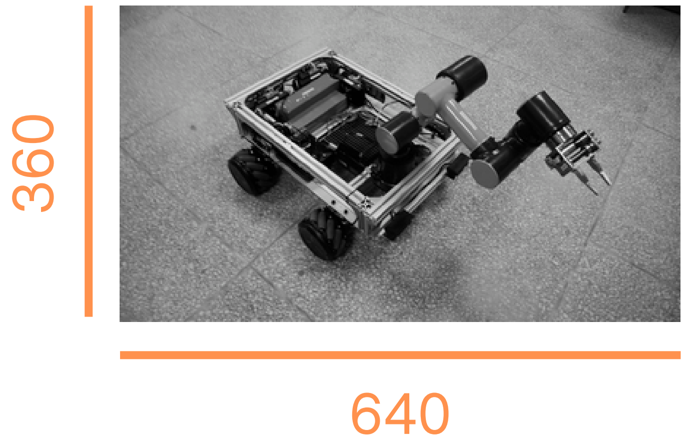
3 workloads:

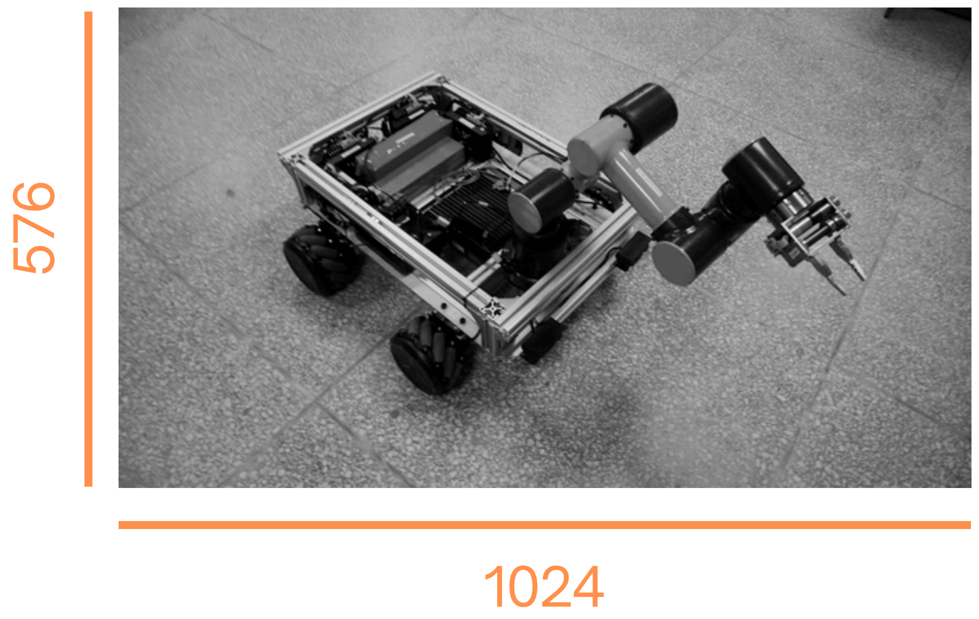
- **Small:** 640x360 px
- **Medium:** 1024x576 px
- **Large:** 1920x1080 px

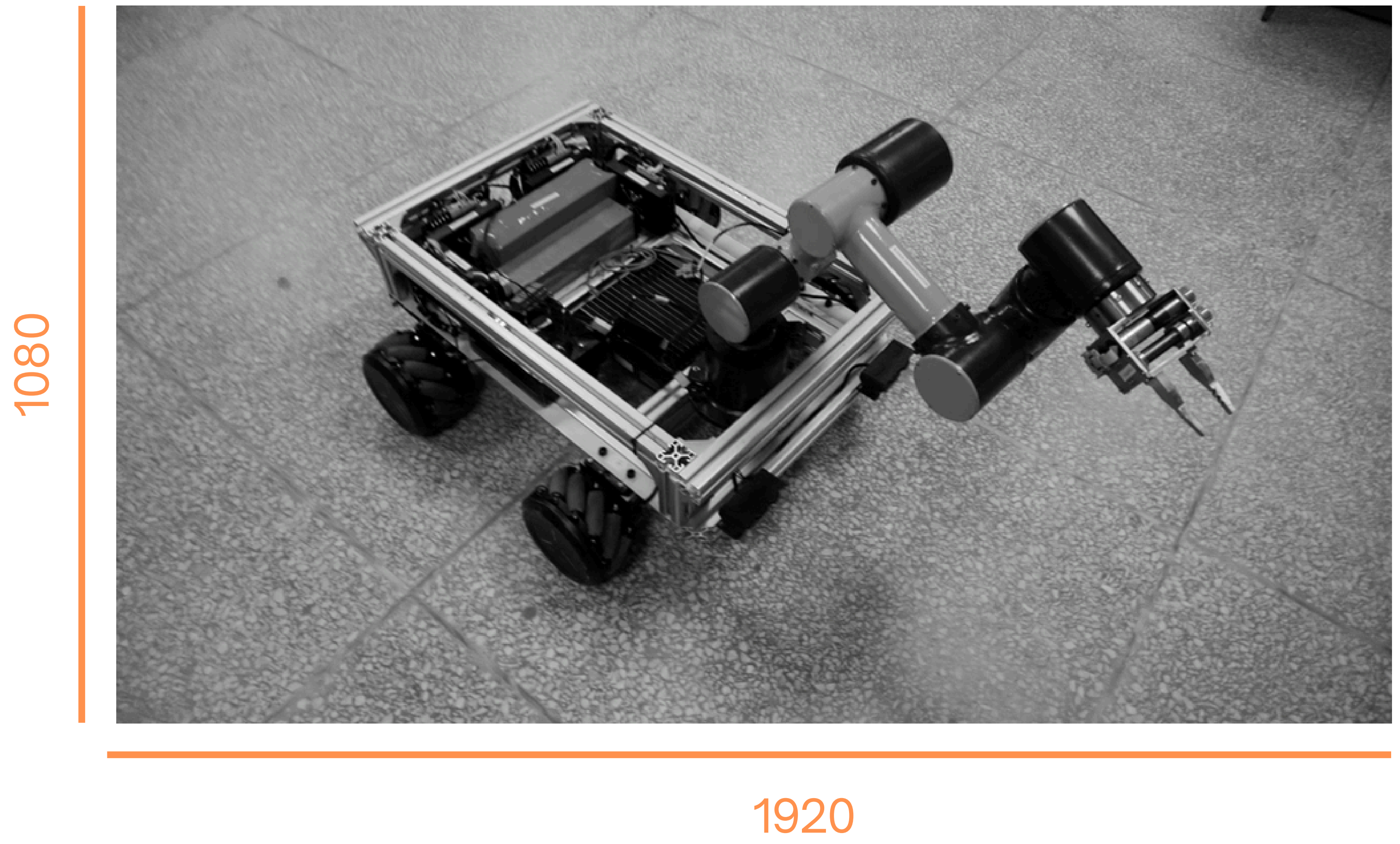
30 frames each

Running on:

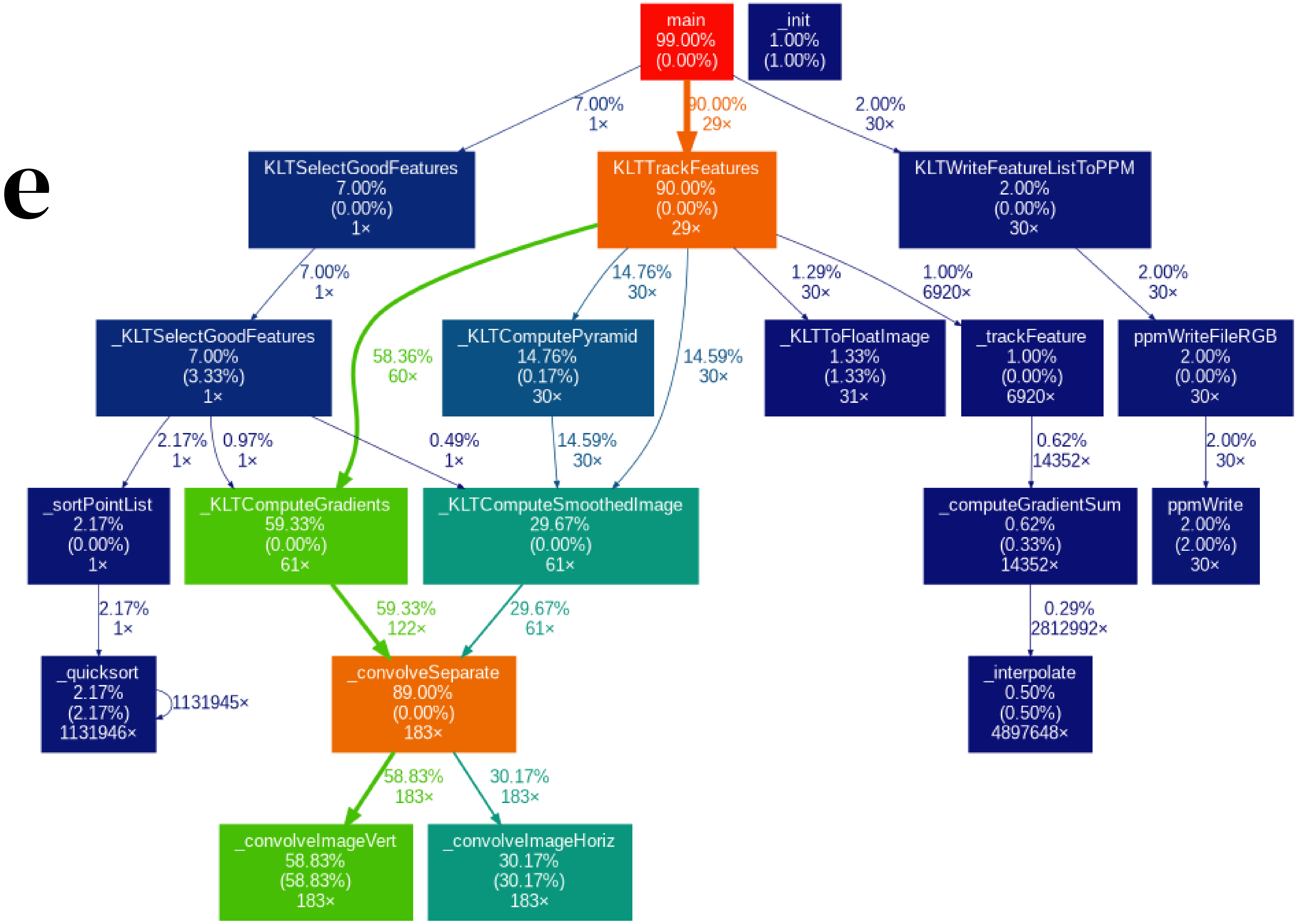
NVIDIA GeForce RTX 3080 (Compute Capability 8.6)







Motivation
V1: CPU baseline



gprof2dot results on V1

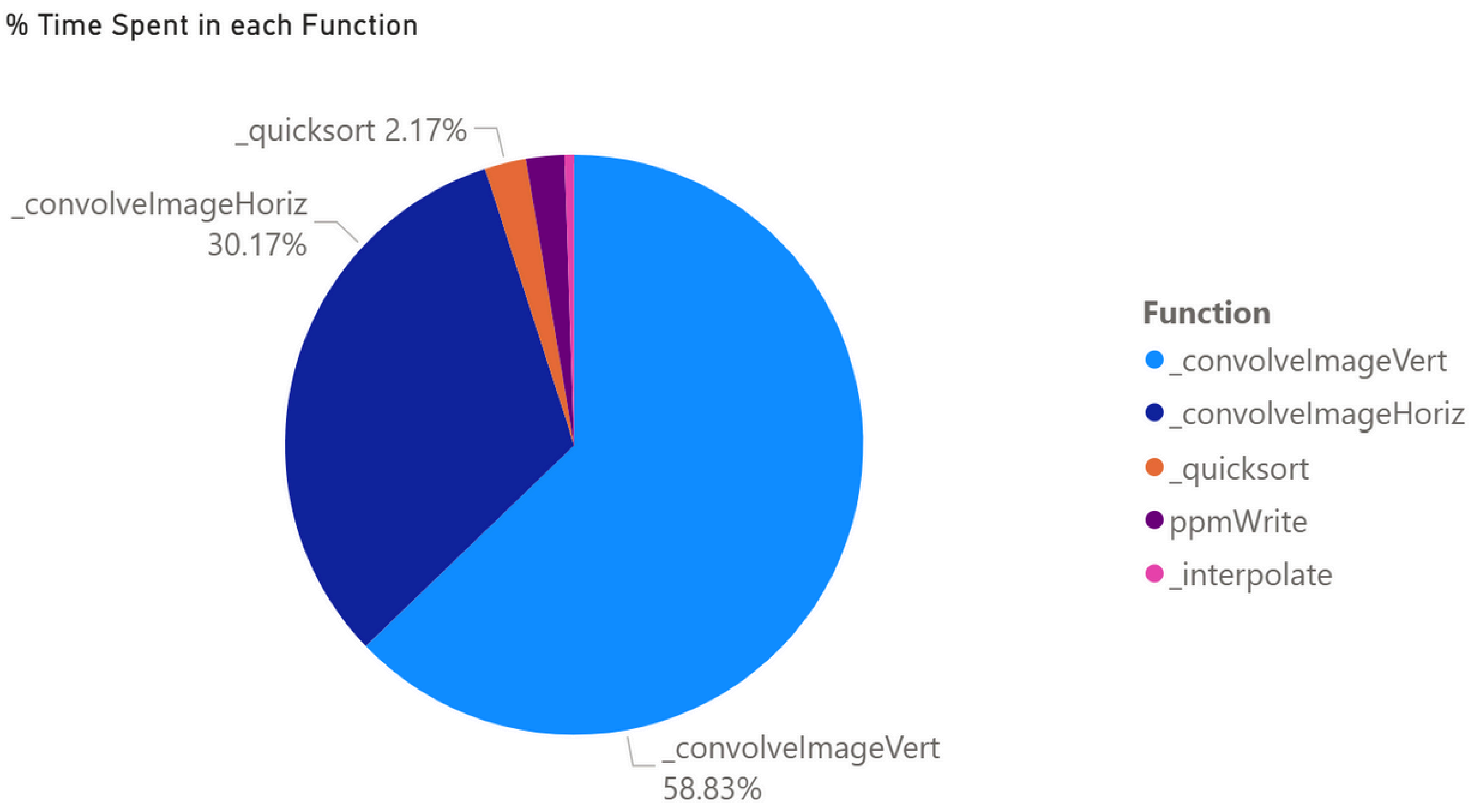
Our bottlenecks…

We ran our program several times over different datasets to find our main bottlenecks:

- _convolveImageHoriz
- _convolveImageVert
- _interpolate *

* _interpolate is included becasue it gets called many times

Identified **Convolutions** and **Feature Tracking** as main opportunities for parallel computation



V2: Naive GPU Port

Introduction of kernels:

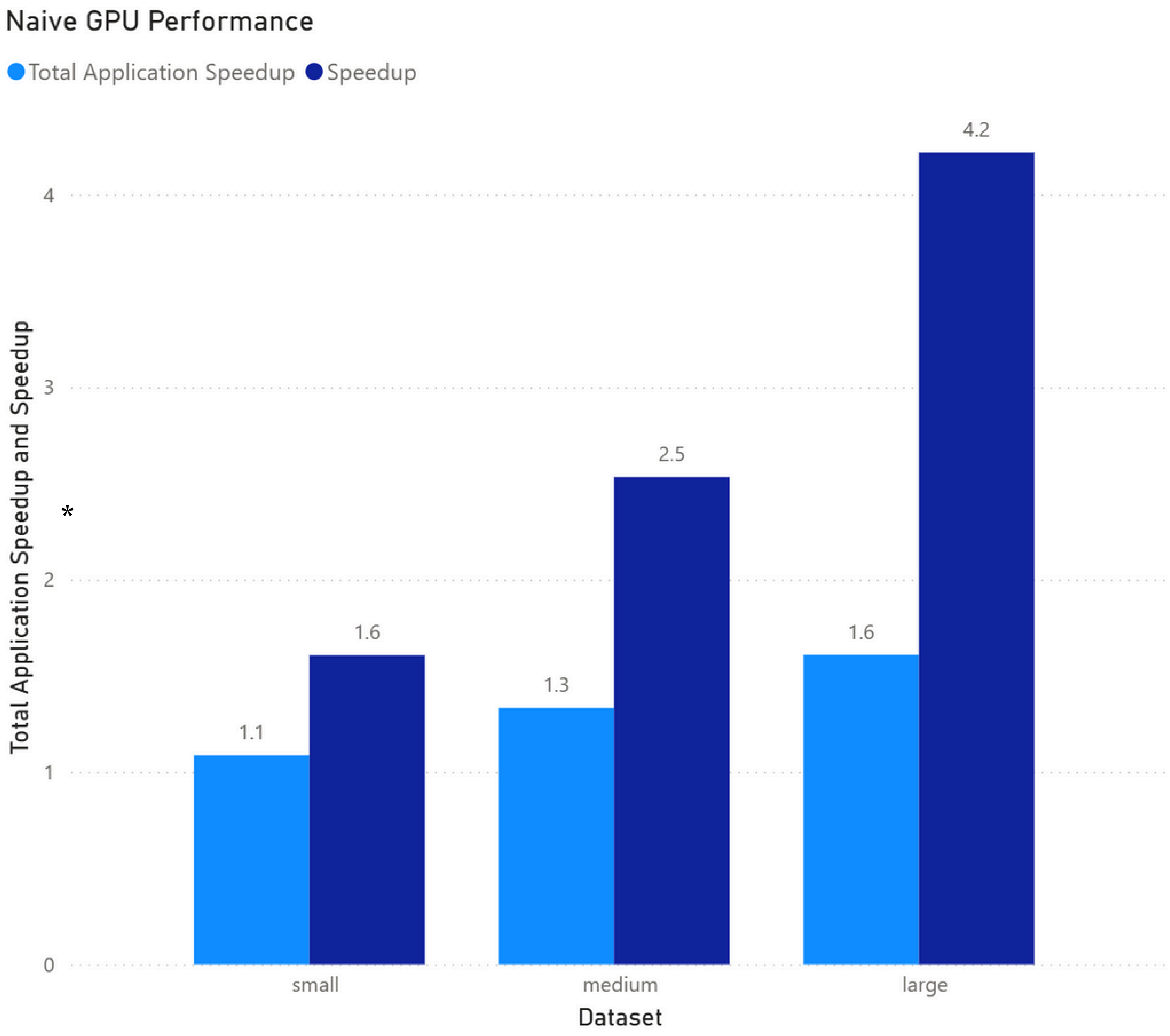
- convolveHorizKernel
- convolveVertKernel
- trackFeatureKernel

Bottlenecks:

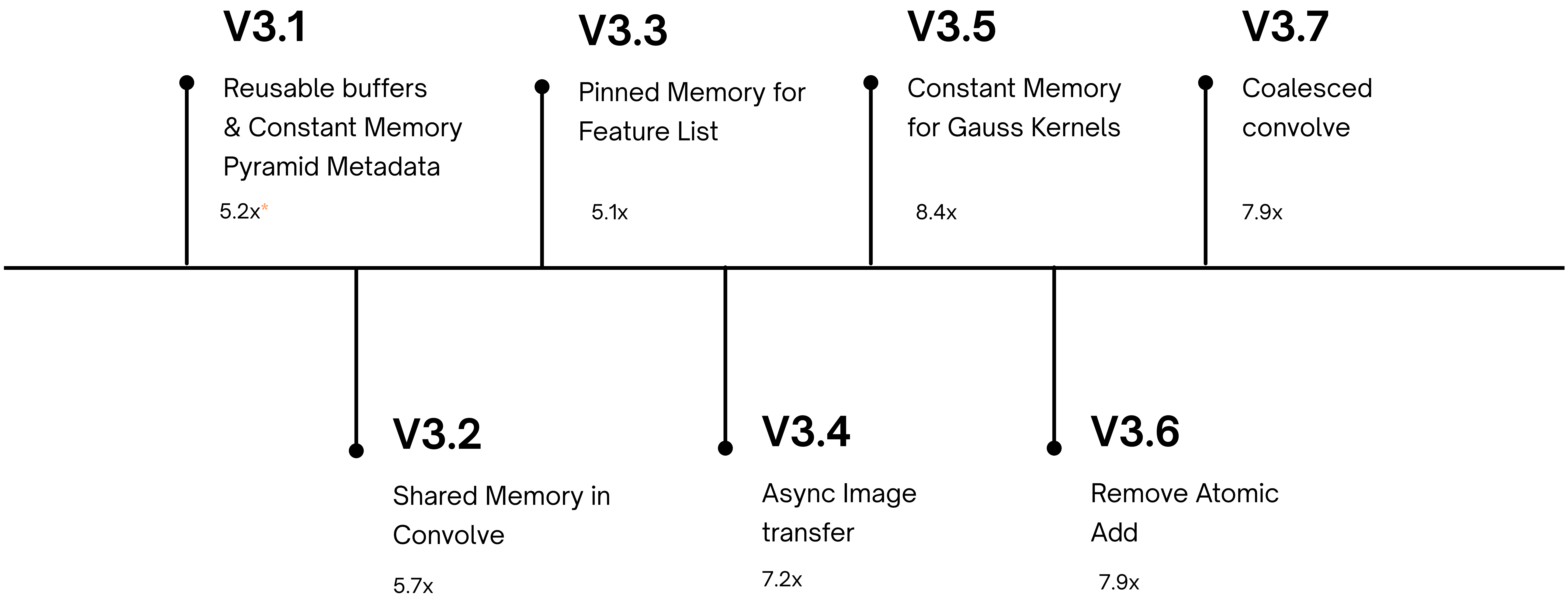
- Memory transfers (H2D, D2H) took upto **51.2%** of tracking time (large dataset)

Small Gains:

- **1.6**x speedup for total application time
- **4.2**x speedup if only tracking time is considered



V3: Optimising V2...



V3.1

Reusable buffers
& Constant Memory
Pyramid Metadata

5.2x*

V3.2

Shared Memory in
Convolve

5.7x

V3.3

Pinned Memory for
Feature List

5.1x

V3.4

Async Image
transfer

7.2x

V3.5

Constant Memory
for Gauss Kernels

8.4x

V3.6

Remove Atomic
Add

7.9x

V3.7

Coalesced
convolve

7.9x

*on our medium workload

V3: Optimised!

Obtained up to **18.6x** speedup!

Small:

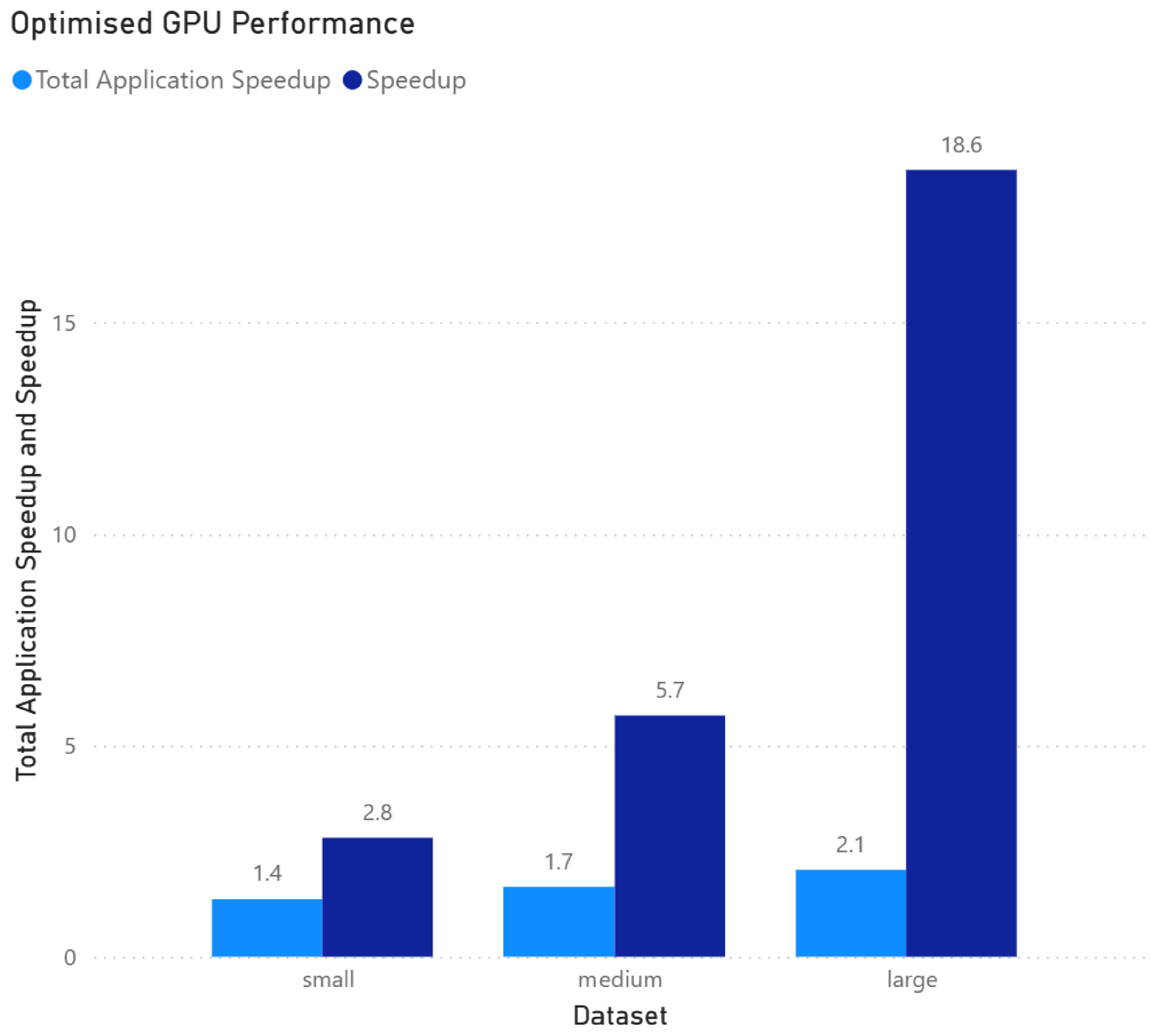
- **2.5x** tracking time
- **1.4x** total application time

Medium:

- **5.7x** tracking time
- **1.7x** total application time

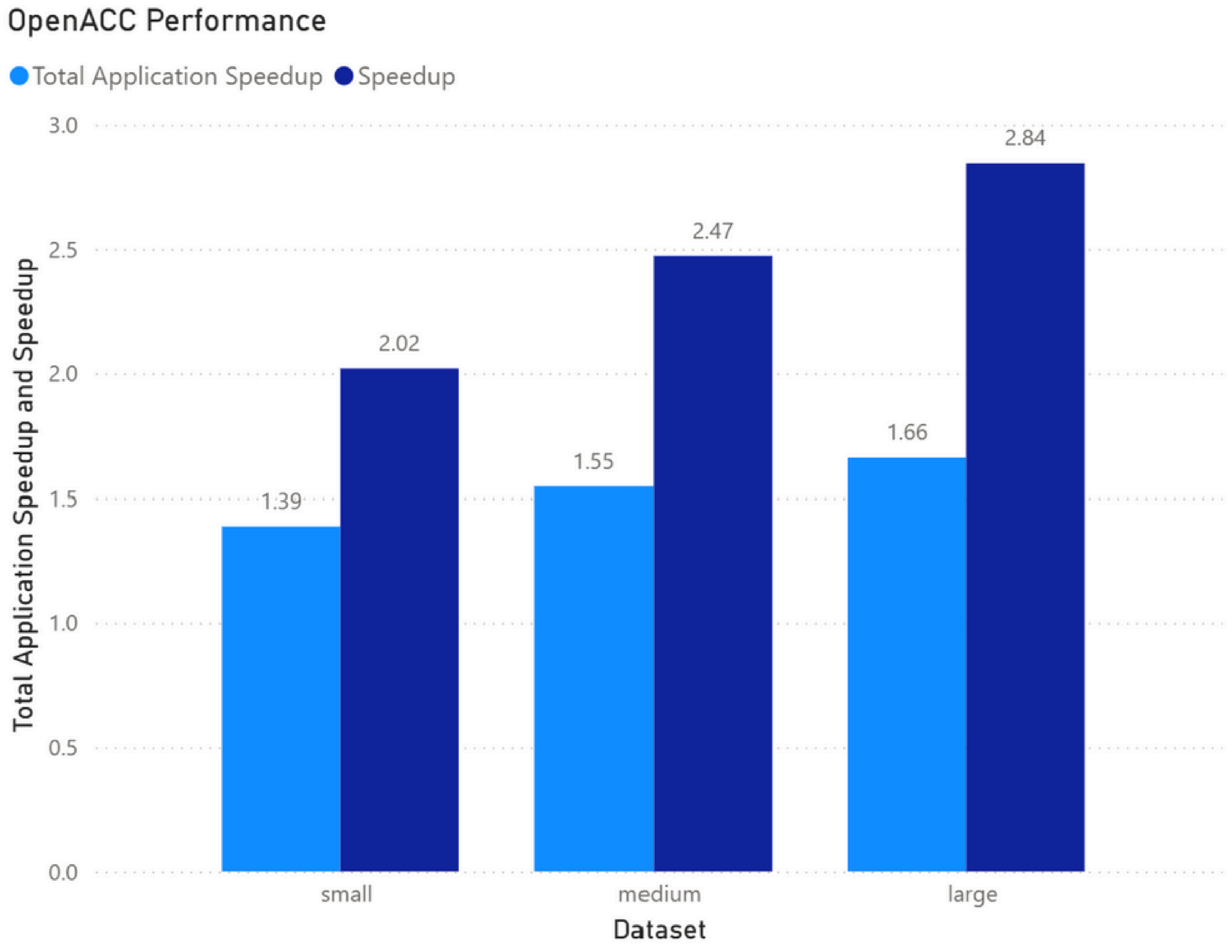
Large

- **18.6x** tracking time
- **2.1x** total application time



V4: OpenACC

Parallalesing our main bottlenecks: **Horizontal and Vertical Convolution** using pragma ACC kernels



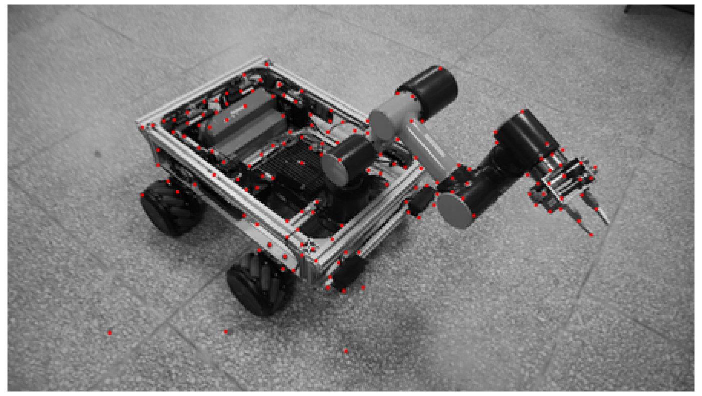
Quick, but Small Gains: 2.84x on large dataset (1.66x application speedup)

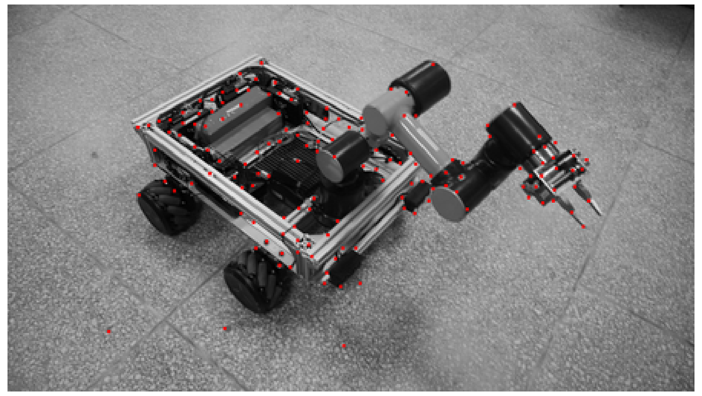
Correctness

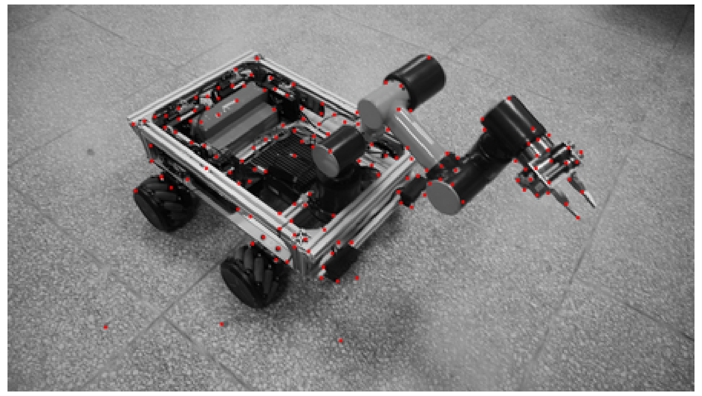
CPU tracks:

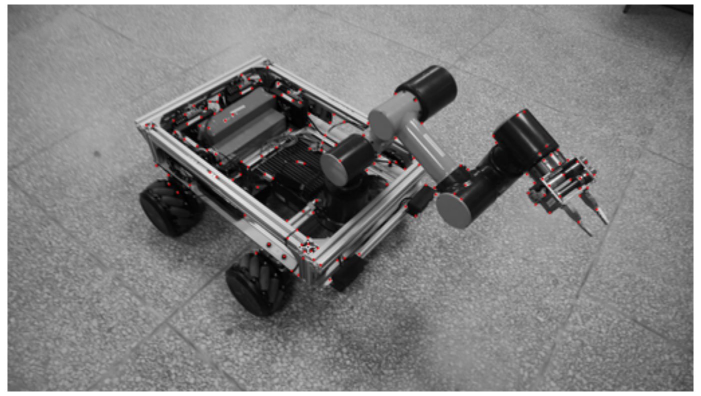
- small : 150 features
- medium : 145 features
- large: 113 features

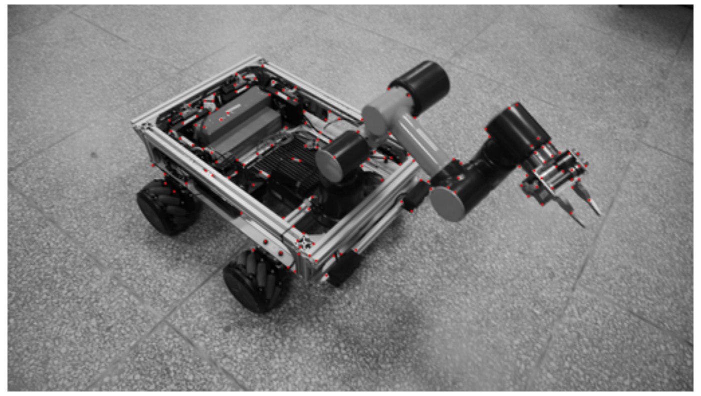
While V2 shows slight divergence, V3 and V4 track features correctly when compared with CPU

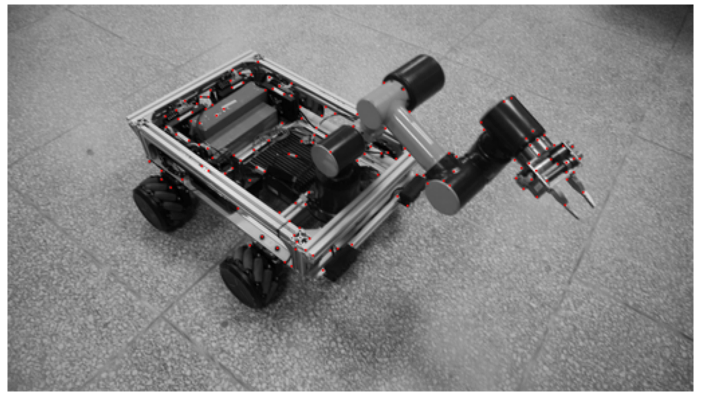

frame 0

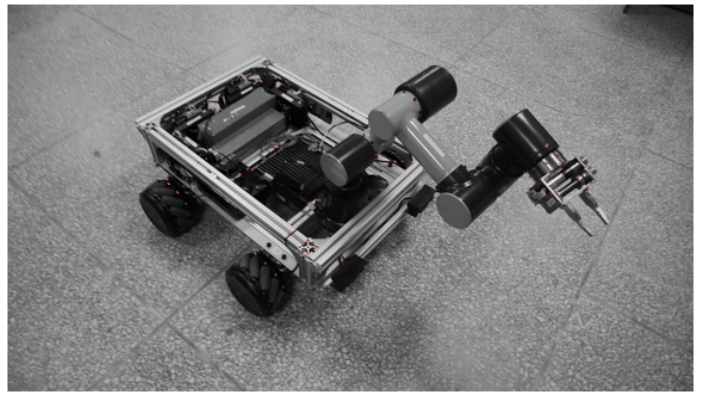

frame 14

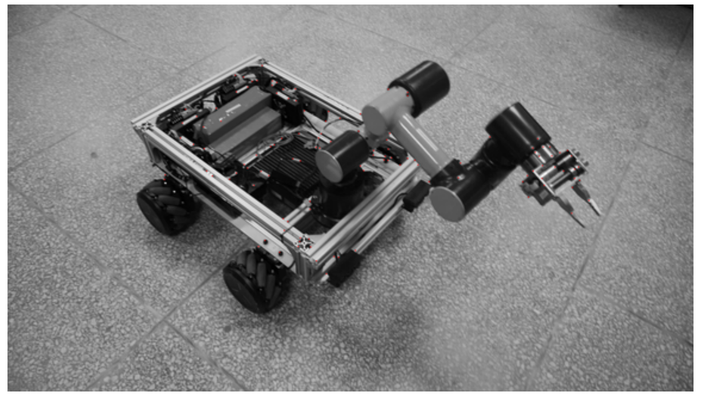

frame 29

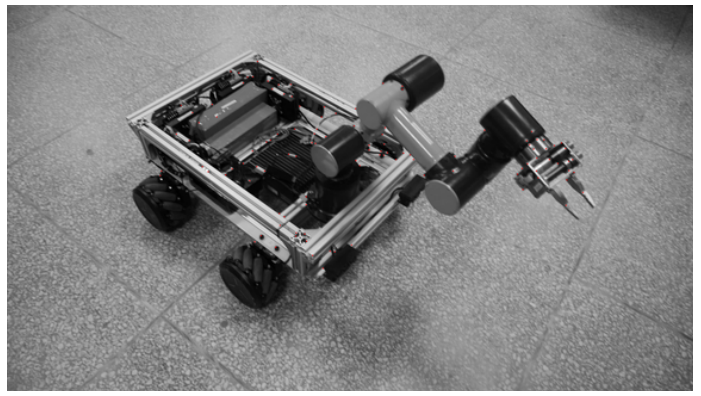

frame 0


frame 14


frame 29

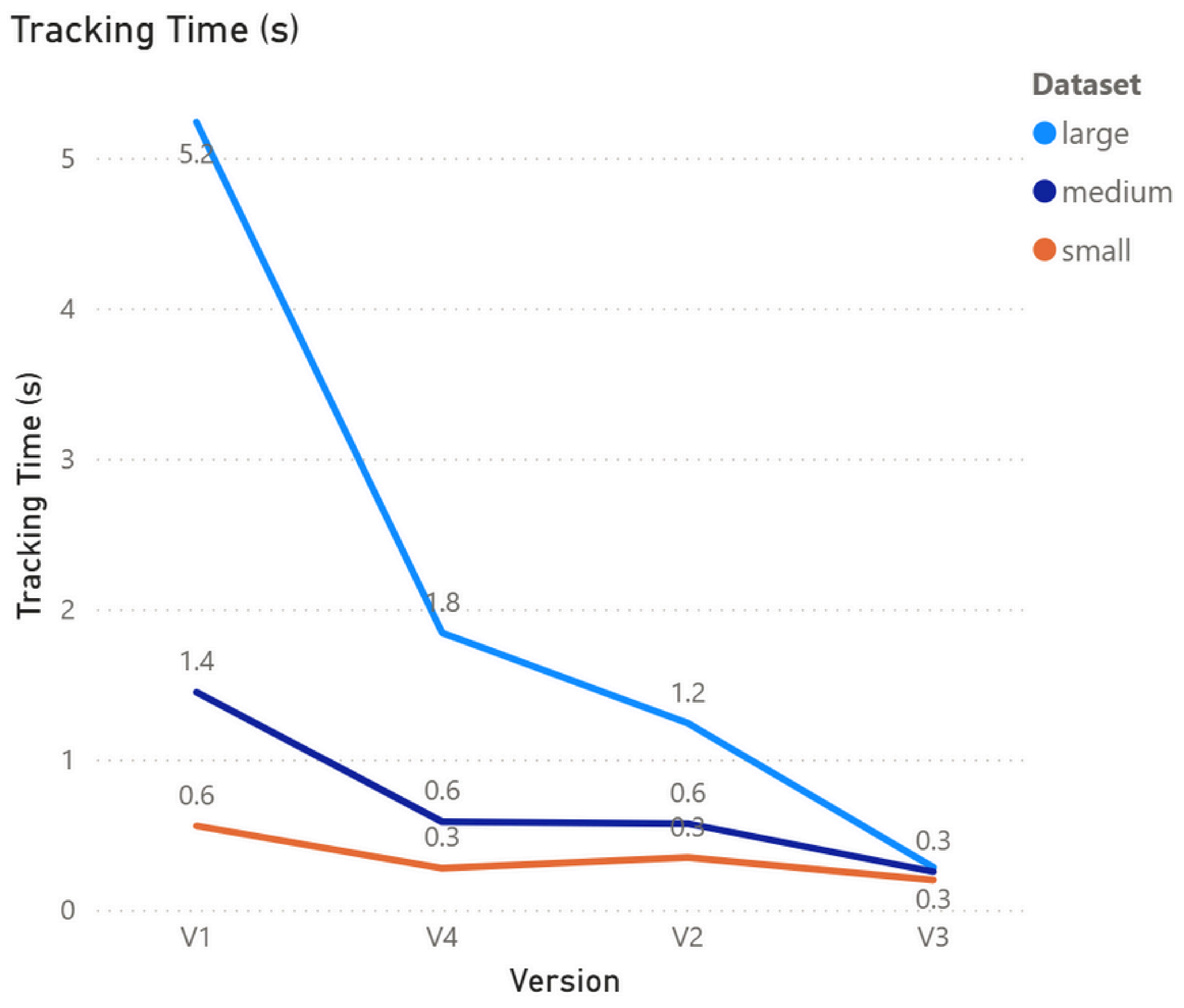

frame 0

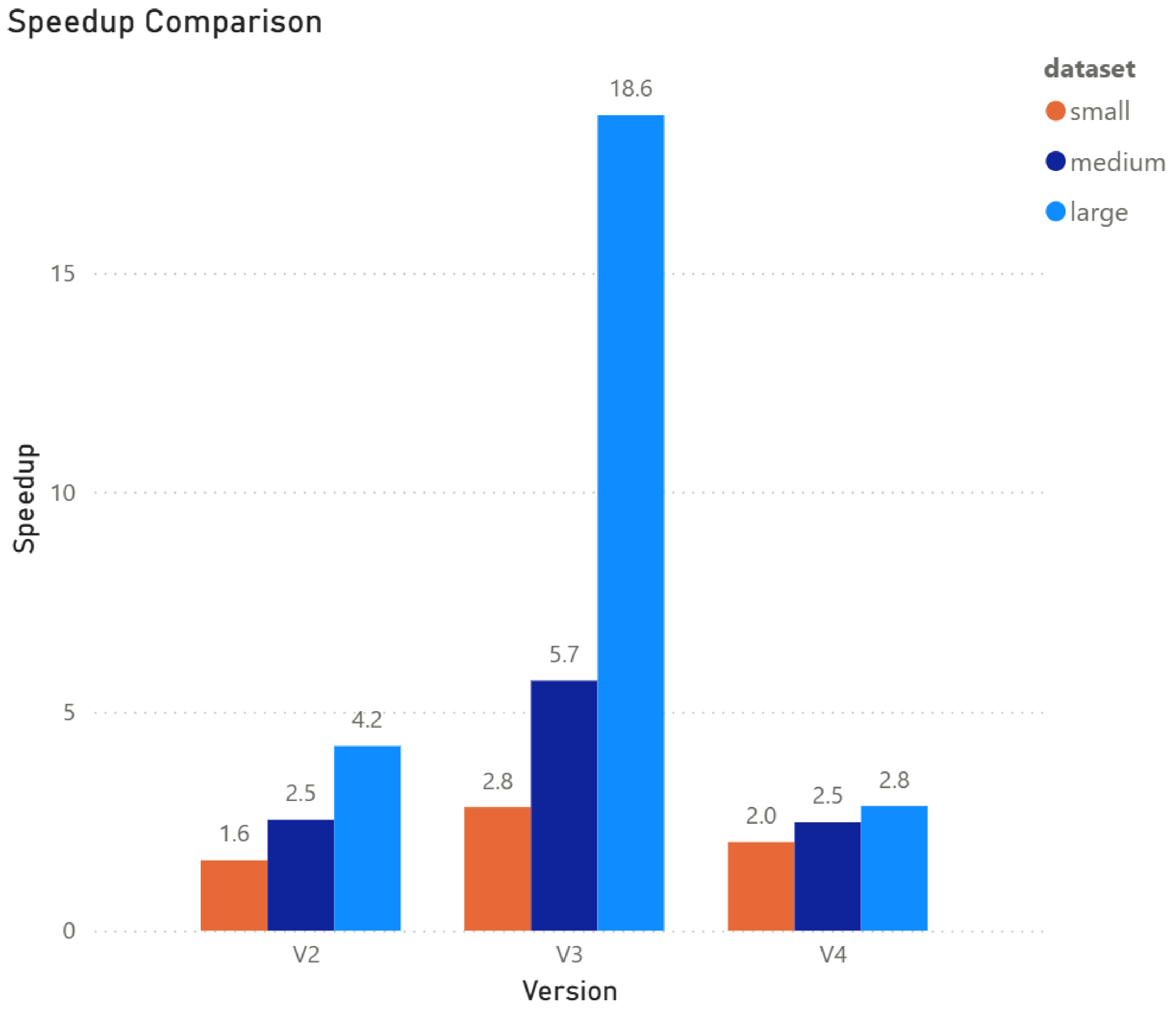

frame 14


frame 29

Comparison of Results

Speedup





CUDA Brought Tracking Time down from **5.2s to 0.28s** while **OpenACC** brought it down to **1.84s.** *

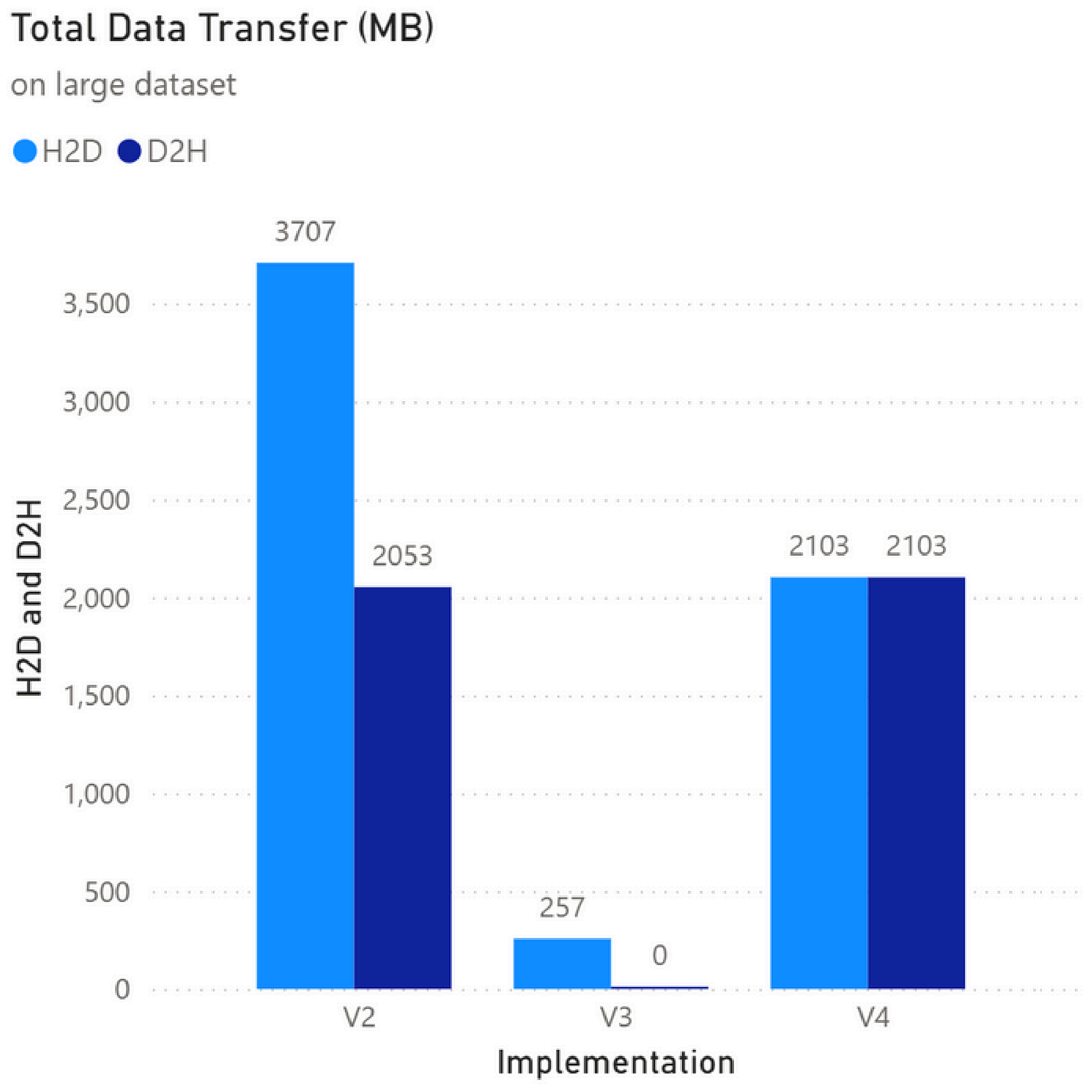
OpenACC shows poor scaling

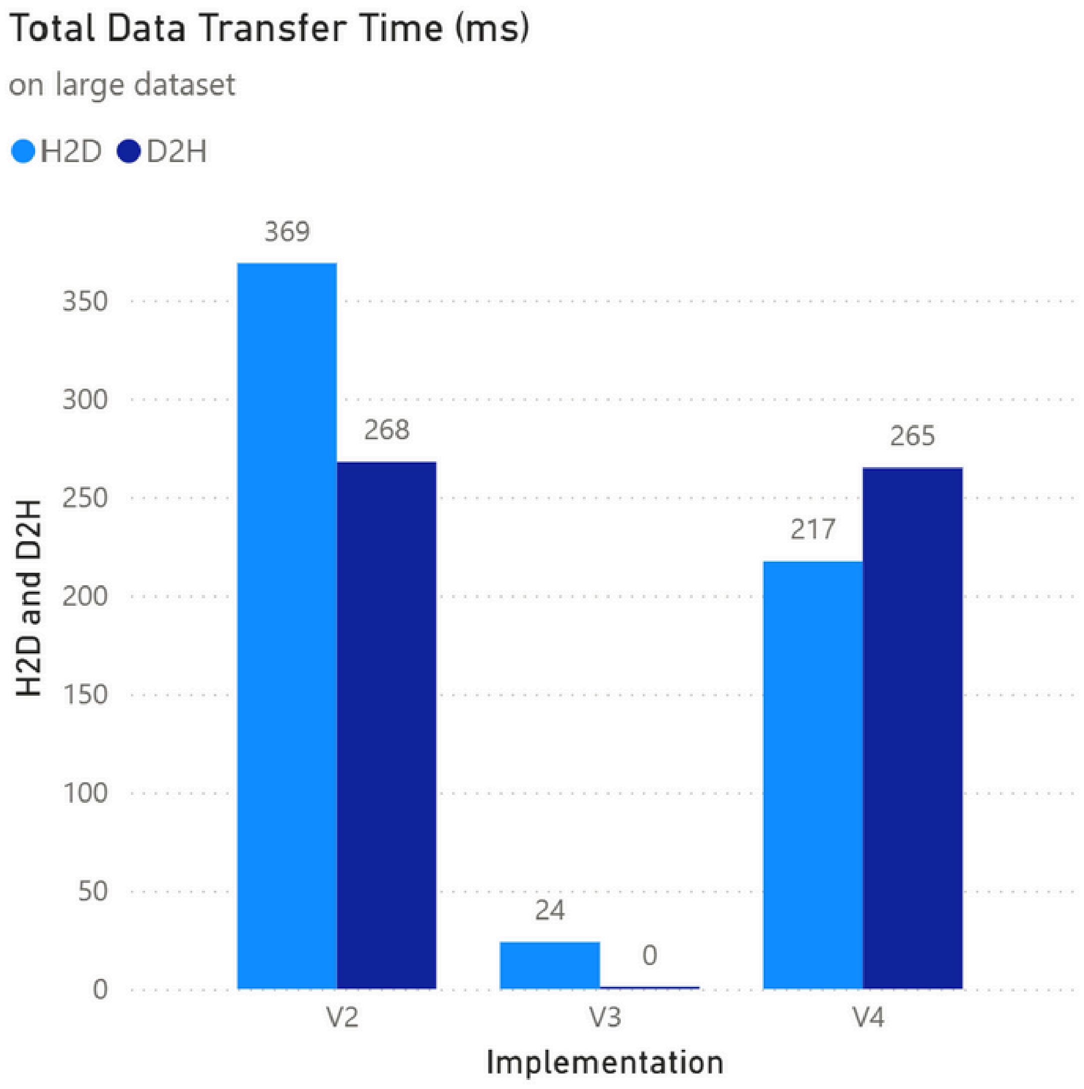
* on large dataset

Comparison of Results

Memory Transfer

The amount of data (MBs), and the time taken transferring that data (ms) from Host to Device and Device to Host ***decrease*** significantly from our CUDA optimization





Comparison of Results

Kernel Breakdown

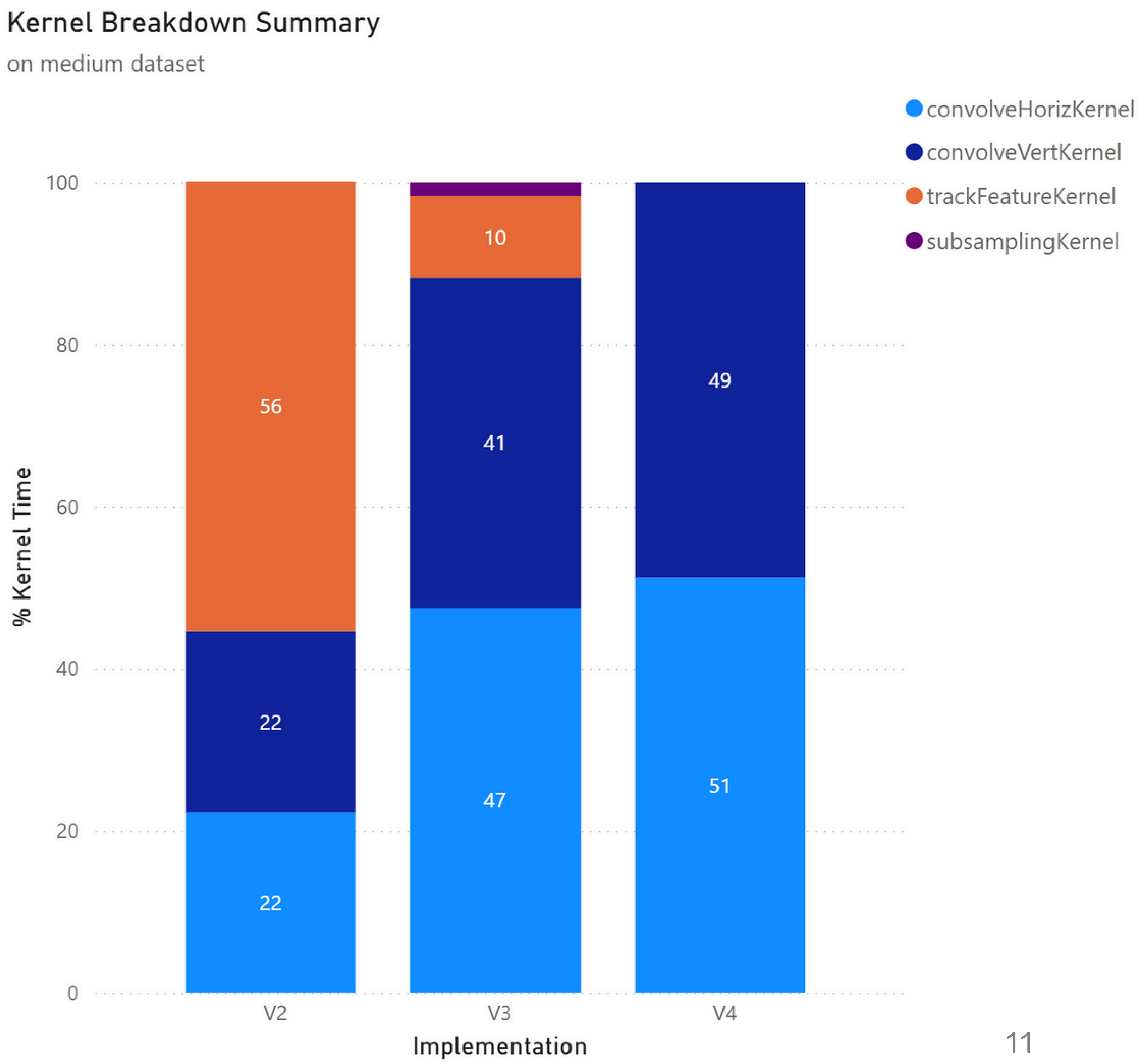
Before Optimisation:

Tracking Dominates

After Optimisation:

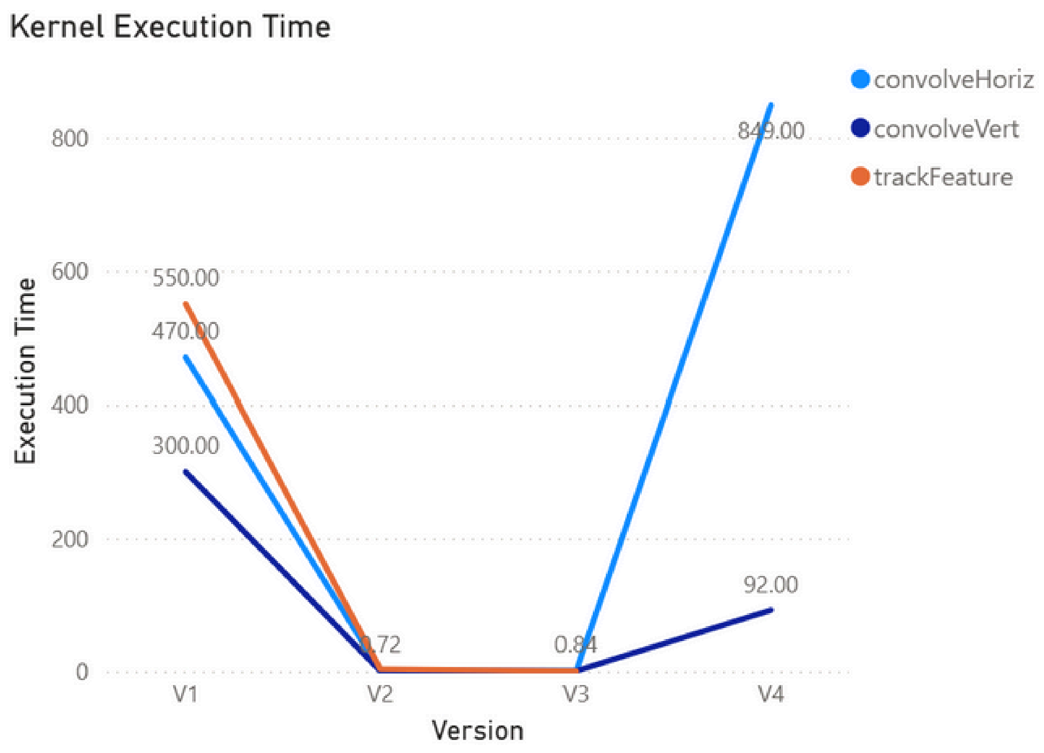
Convolutions Dominate

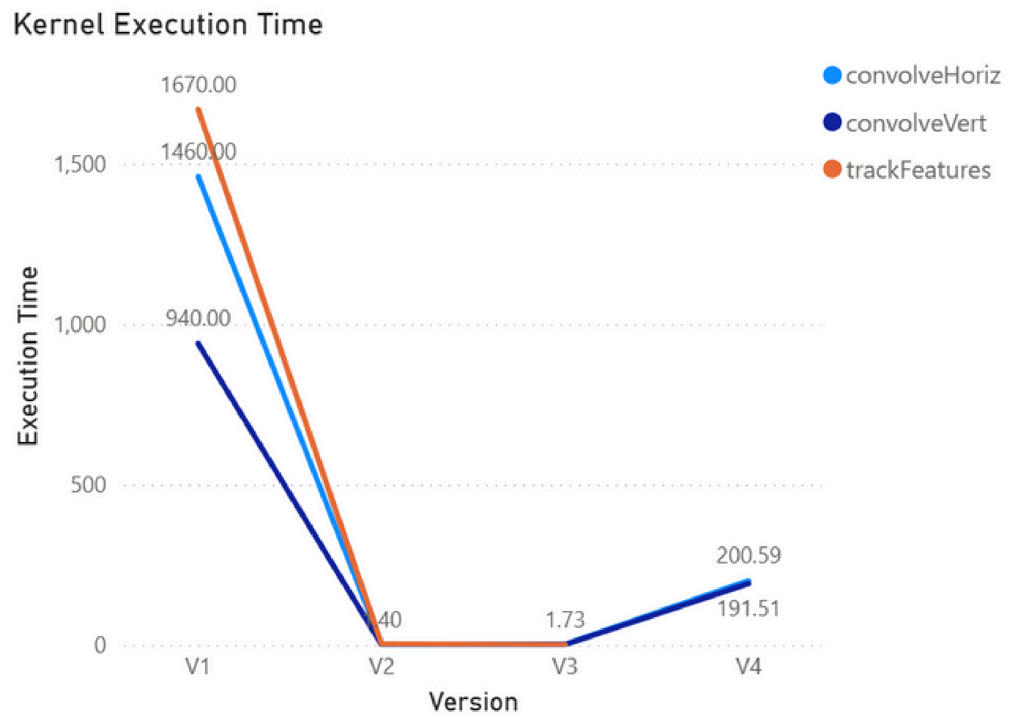
This aligns with the CPU baseline!

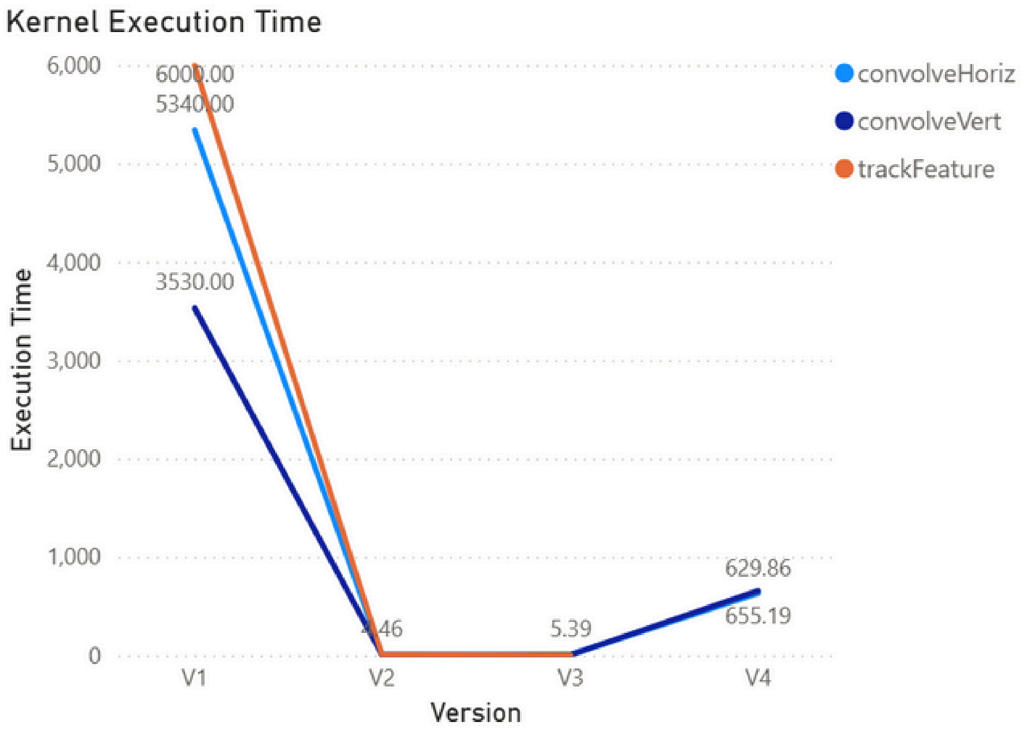

Kernel Breakdown Summary

on medium dataset

Comparison of Results


Kernel Execution Time


Kernel Execution Time


Kernel Execution Time

Successfully parallelised our bottlenecks:

Horizontal Convolution:	**5340 ms**		**5.39 ms**
Vertical Convolution:	**3530 ms**	⟶	**4.93 ms**
Feature Tracking:	**6000 ms**		**0.39 ms**

Key Takeaways

- Highly Suitable for Parallelisation
- Scales well for large workloads
- CUDA > OpenACC?
- Sequential Dependencies Constraints