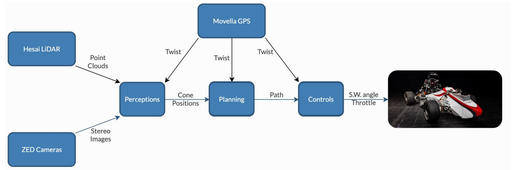
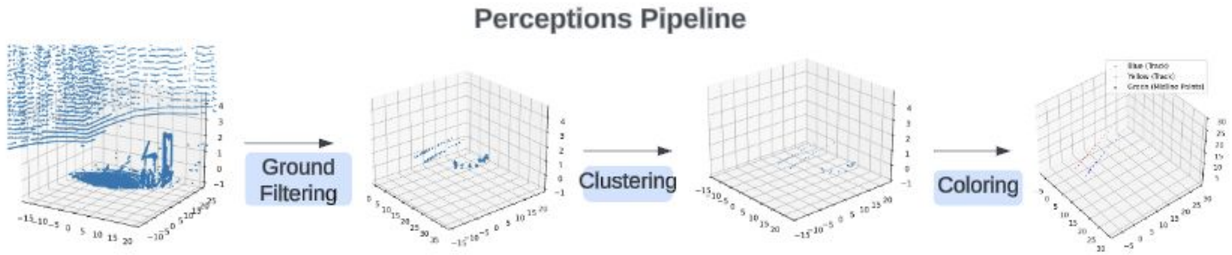
Alexander Blasberg, Alice Tran

CUDA LiDAR Perceptions Algorithms

Abstract

The **perceptions pipeline** converts a LiDAR point cloud frame into identifiable, colored cones. It is one part of CMR's larger autonomous driving pipeline. We used **CUDA** and tested on **GHC clusters** to parallelize this process.





Conclusion

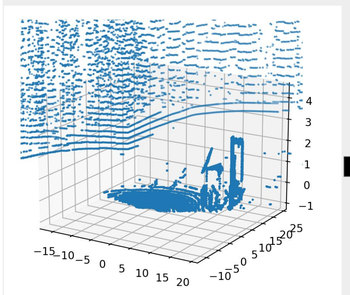
We experimented with various decomposition strategies to parallelize computation across the pipeline. With our test cases we often had to balance speed and scalability to more general cases that could be encountered on the racetrack.

Ground Filtering

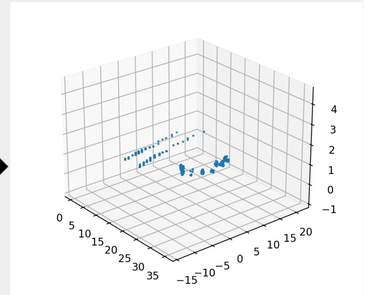
Decomposition: Allocate points into respective bins by assigning one thread to one point. Each bin is now a thread where the lowest point is found in each bin. Segments (groups of bins) are then each a thread for line fitting.

Synchronization: Information is shared after kernel end. Each cell/segment thread must access information about point allocation into bins. As grouping grows larger so does information shared.

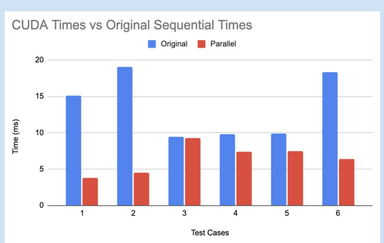
Communication Reduction: Each thread individually does work without communicating with other threads and without modifying shared information that is used elsewhere.

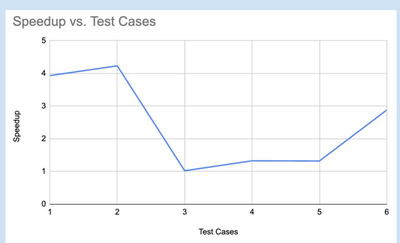


Before Ground Filtering



After Ground Filtering





Computationally Difficult Test Cases - 1,2, 6

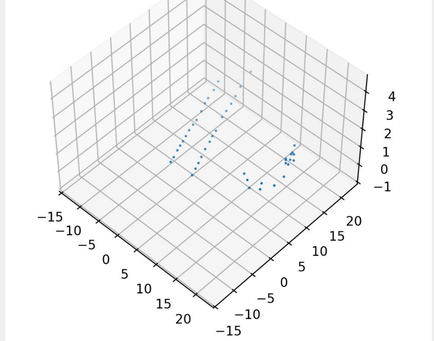
- **Test Cases with more points for each angle segment results in slow work distribution for sequential**
- **Parallelization resulted in better distribution of difficult work resulting in higher speedup times**

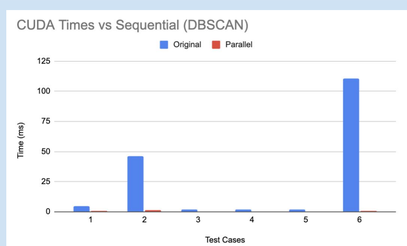
Clustering

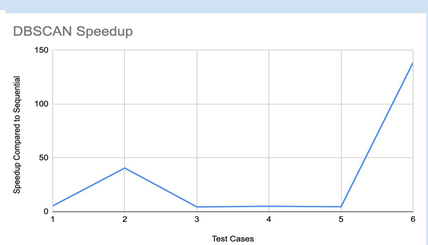
Decomposition: Each thread represents a non-ground point. We compare single point to all other points. If the distance is small enough we combine the two parent roots of the point. (path compression)

Synchronization: Threads access parent array to make decision on which cluster to union to. We synchronize when comparing root sizes of points. At the end, point joins the largest and closest parent group.

Communication Reduction: Each point only modifies its individual parent belonging reducing conflicting access.







Computationally Difficult Test Cases - 2, 6

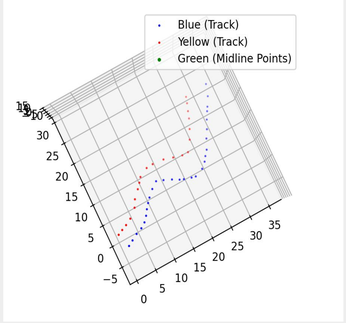
- **Test Cases with larger clusters of points results in more further and slower depth finding in sequential**
- **Parallel Path compression results in reduced neighbor finding and better work distribution**

Coloring

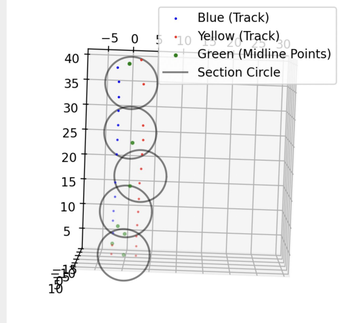
Decomposition: Allocate a thread per section of the track and find the midline point by averaging all the points in the segment. After, launch a thread for each point and classify it based off the closest section of midline.

Synchronization: After we compute the midline points and after cone coloring we must synchronize to ensure that all other threads have the same information.

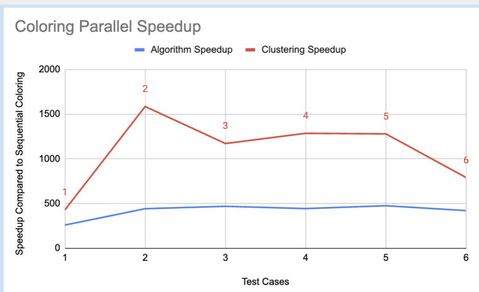
Communication Reduction: Each segment's midline is calculated separately and once midline points have been calculated each point can calculate its color independently.



Clustered Coloring Algorithm



Midline Algorithm



Computationally Difficult Test Cases - 2, 5

- **Clustering outperforms the midline algorithm which has higher communication and synchronization costs**
- **Clustering performs better in high density culsters**