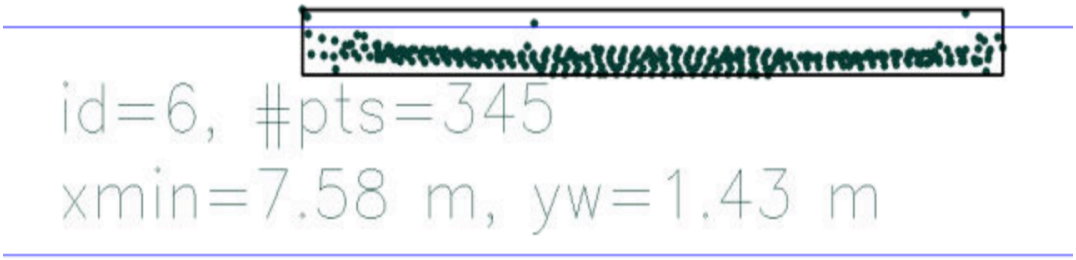
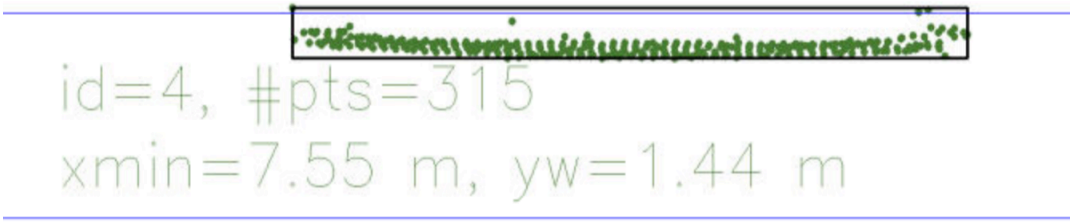
Performance evaluation:

Part 1: LIDAR-based TTC computation

a. From the plots of Lidar-based TTC computation using different detector-descriptor combinations in ttc_plots.png, there is an obvious jump around the 5th datapoint in almost all of the plots. Consider the case where detector=brisk, descriptor=brisk.
Previous frame top view:



Current frame top view:

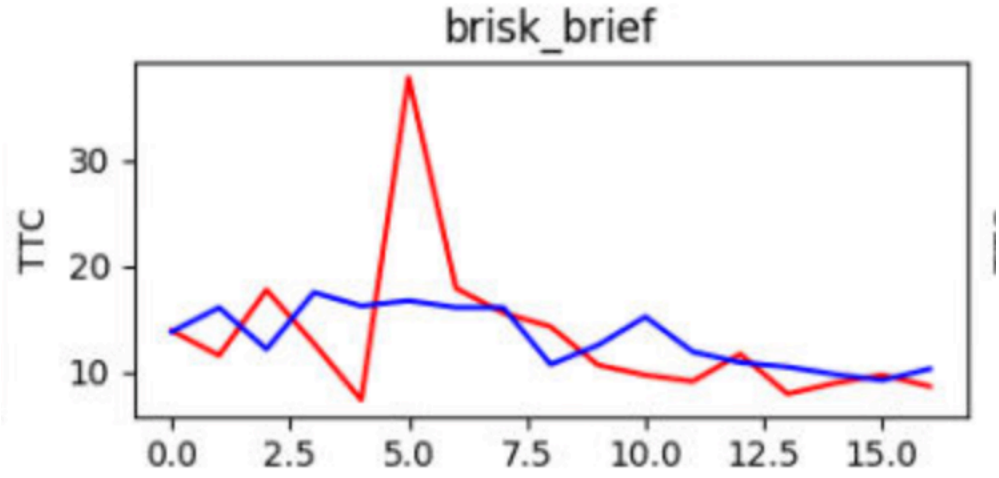


The TTC computed by the program in this case is uses an xPrev = 7.581 and xCurr = 7.561 giving a TTC = 37.8 s
If we compute the TTC using the closest point to the ego car:
TTC = 7.55 * 0.1 / (7.58 – 7.55) = 25.16 s

The TTC computed by the program is higher because the program does not use the closest lidar point. The TTC formula is based on assuming a constant velocity model and so we assume a constant frame rate, which in this case implies a smaller change in

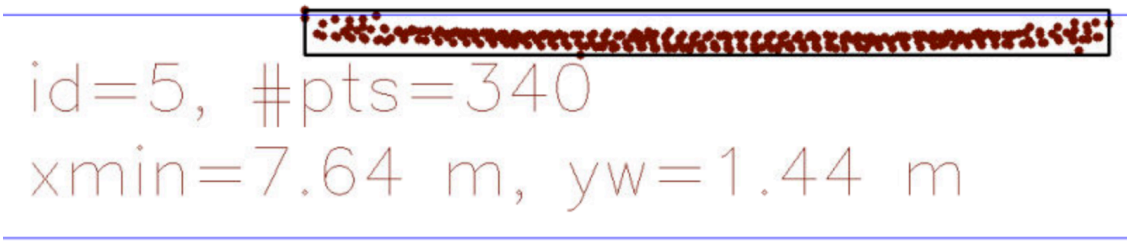
distance from the car in front over a 0.1 s duration. And hence the TTC computation is way off.

b. There is another datapoint where the TTC computed from the lidar data is much smaller than that computed using the camera data.

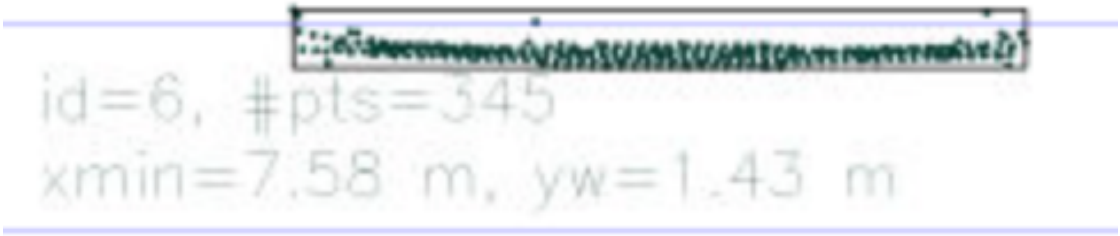


Around the third data point, the TTC computed by the program is 7.38 s. Since we don't use the closest lidar point to the ego car in an effort to avoid outliers, the values used are:
minCurr = 7.581, minPrev = 7.684
TTC = 7.581 * 0.1 / (7.684 – 7.581) = 7.38 s

However, if we use the minimum x-valued lidar point to perform this computation:
minPrev = 7.64



minCurr = 7.58



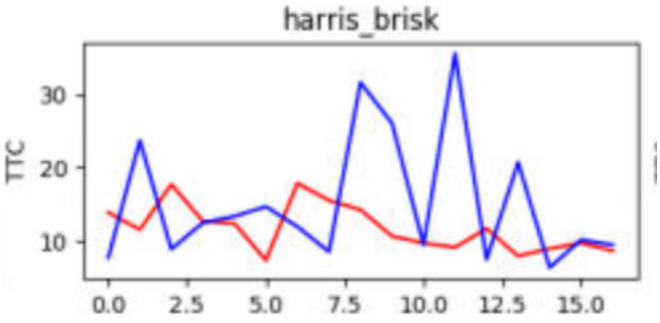
TTC = 7.58 * 0.1 / (7.64-7.58) = 12.63 s

In this case, we would need to use a better technique to filter outliers to get a better estimate of the TTC. Using an averaging technique over multiple frames, instead of computing the TTC based on only the previous and current frame would give a smoother TTC change over time.

Part 2: Performance Evaluation 2

The plots of LIDAR based and camera based TTC computation using different detector and descriptor combinations are shown in **ttc_plots.png** under the project's main directory. The readings used to generate these plots are under the results/ directory.

Computations using **AKAZE, BRISK and SHI-TOMASI** as the detector perform much better than the others. Using HARRIS corner detection as the detector type performs particularly bad. This seems to follow form the mid-term project where I noticed that using the HARRIS detector resulted in generating only 1/10th the keypoints of the other detectors. With a reduced sample size, the matches between keypoints may also not be very effective.



For this project, I am using the median distance ratio between keypoint matches between current and previous frame to compute the TTC. However, in some cases the difference

between the mean and median of these ratios is significant. It may be worth exploring using a weighted technique where the ratio used to compute the TTC is a weighted value of the median and mean with higher weight being assigned to the median (especially in cases where the detector-descriptor performance to identify matches is poor).

Another possible reason for the TTC outliers could be that the some of the keypoint distances may have decreased even though the ego car moved closer to the car in front, either because the road has a slight curve or the car in front drifted slightly. The model used to compute TTC using camera data does not fully correct for these outliers.