SFND 3D Object Tracking
Final Project Report

The following describes how I have addressed each of the project rubric specifications.

FP.1 Match 3D Objects

The 'matchBoundingBoxes' fn iterates through the keypoint matches vector and starts by attempting to find a bounding box for the matched point in the previous frame. If found, the function continues by now attempting to find a bounding box for the matched point in the current frame. If found then this pair of bounding boxes has a match, which is recorded by constructing map entry for this box pair, with a key from the two box ids, and the count of the number of matches for this box pair is incremented.

The fn continues though all the keypoint matches, incrementing the counts for previously found box pair associations or creating new ones as appropriate.

Once complete the fn iterates through all the bounding boxes of the previous frame and uses the box pair association map to find the best match for the bounding box in the current frame.

FP.2 Compute Lidar-based TTC

The 'computeTTCLidar ' fn finds the closest valid point of the rear of the preceding vehicle in the previous and current frames and uses the delta distance and the frame frate to compute the time it will take for the distance to the vehicle, measured in the current frame, to be travelled at the computed speed.

I tried two methods to reject outlier lidar points on the rear of the preceding vehicle.

1. Sort the delta distances and then iterate through the vector looking at the delta between adjacent points in the delta vector. Reject all points until the adjacent delta distance is less than 1mm. This method did reject some outliers but suffers from the problem that outliers may be in groups.
2. Sort the delta distances but this time make a histogram of delta distances. This can then be used to exclude distance outside the lower point of defined percentile population. I tried this with the usual 95 percentile population (so find the lower bin where 2.5% of the histogram population is reached). This point still produced large variances in TTC, so I increased this until I settled on 20% percentile population (so the lower 10% point).

FP.3 Associate Keypoint Correspondences with Bounding Boxes

The 'clusterKptMatchesWithROI' fn iterates through the keypoint matches and looks at the Euclidean distance separation of the matched keypoints where the keypoint lies within the ROI of the supplied bounding box. I'm expecting valid keypoint matches to have a relatively small Euclidean distance separation so have simply set a max tolerance of 10. Any keypoint match & point with the separation of less than this is added to the bounding box's keypoint match and keypoint vectors.

FP.4 Compute Camera-based TTC

The ‘computeTTCCamera’ fn iterates through all the keypoint matches in the supplied vector and computes the Euclidean separation distances between each keypoint and all the others in the vector in both the current and preceding frames. A ratio of the separation distance in the current frame divided by the separation distance in the previous frame is then taken.

Essentially this gives a measure of how the separation distances between keypoints on the preceding vehicle has changed between frames. It should be noted that this method does have weaknesses and will become more inaccurate as the size of the vehicle approaches the distance to go before collision.

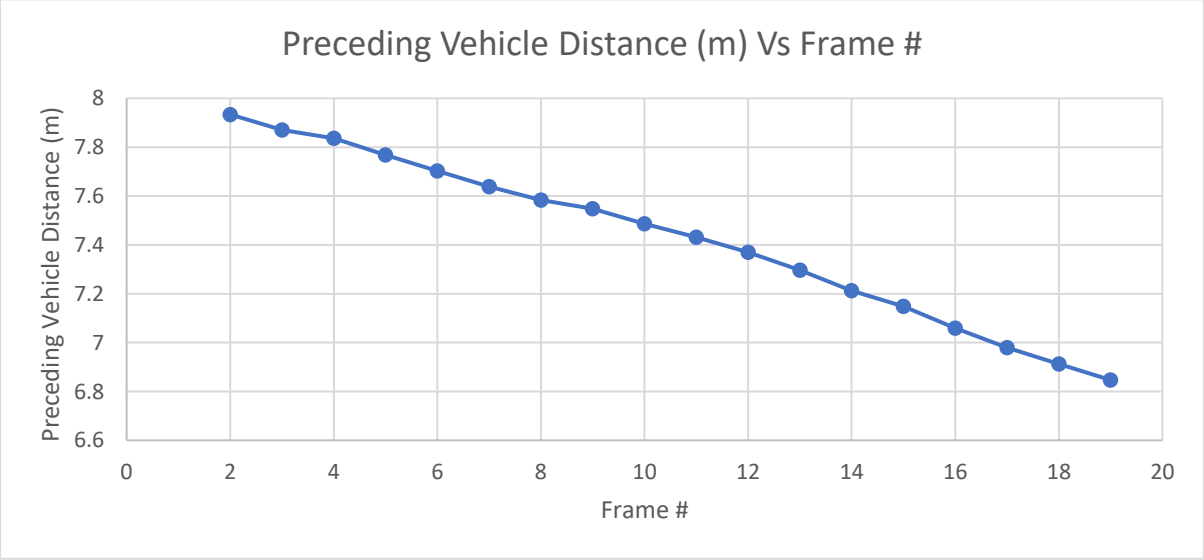
Each separation distance ratio is added to a vector, which is then used to compute a mean for the separation ratio, which is then used to compute the TTC. I tried using a mean (see the results in the last section of the report), but the median appears to give better results. Further filtering could be implemented by using a histogram and taking the peak value (or the mid-point between the lower and upper 95 percentile points).

FP.5 Performance Evaluation 1

By recording the results of the computeTTCLidar fn to a file in csv form its possible to plot the results in excel. The results recorded are given below:

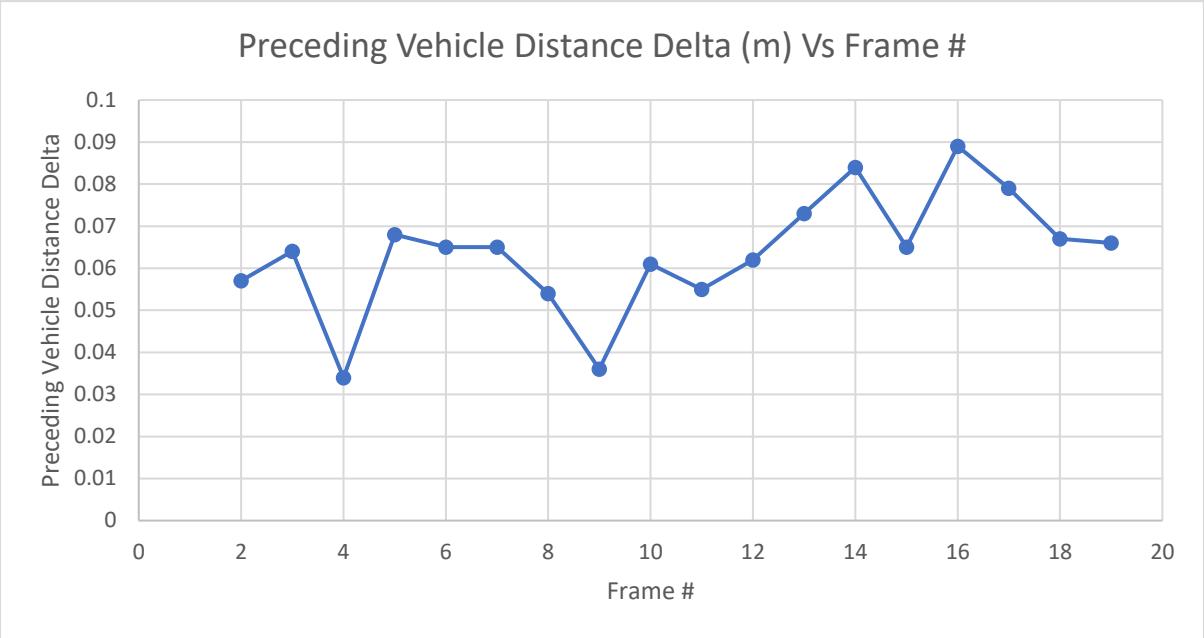
Fame #	Delta Dist (m)	Distance (m)	Speed (m/s)	Avg Speed (m/s)	TTC (s)
2	0.057	7.934	0.569999	0.569999	13.9193
3	0.064	7.87	0.640001	0.605	12.2969
4	0.034	7.836	0.339998	0.516666	23.0472
5	0.068	7.768	0.680003	0.5575	11.4235
6	0.065	7.703	0.649999	0.576	11.8508
7	0.065	7.638	0.649999	0.588333	11.7508
8	0.054	7.584	0.540003	0.581429	14.0444
9	0.036	7.548	0.359999	0.55375	20.9667
10	0.061	7.487	0.609999	0.56	12.2738
11	0.055	7.432	0.549999	0.559	13.5128
12	0.062	7.37	0.620002	0.564546	11.8871
13	0.073	7.297	0.730001	0.578333	9.99588
14	0.084	7.213	0.840001	0.598462	8.5869
15	0.065	7.148	0.649996	0.602143	10.997
16	0.089	7.059	0.890004	0.621333	7.93143
17	0.079	6.98	0.789997	0.631875	8.83548
18	0.067	6.913	0.670002	0.634118	10.3179
19	0.066	6.847	0.659998	0.635555	10.3743

Plotting this data gives:



The chart above shows the distance to the preceding vehicle (calculated from the current frame) vs. frame number. It shows that the test vehicle appears to be approaching the preceding vehicle at approximately a constant velocity.

Looking at the delta distances on each frame:



Shows that frames 4, 9 & 15 look like there are unexpected drops in the delta distance. As speed is derived from this there will be commensurate outliers in the computed TTC.

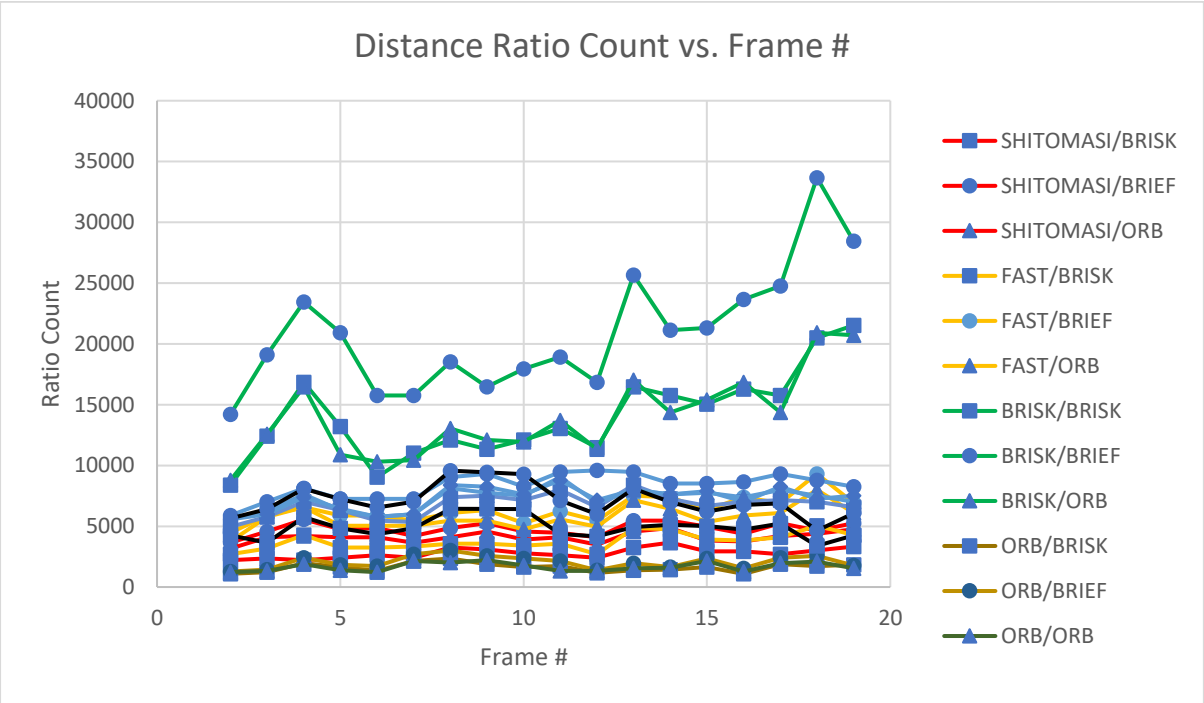
The lidar points have been filtered to exclude outliers so I don't believe the variances in delta distances are caused by using outliers in the calculation.

Assuming the neither vehicle has not almost instantaneously slowed and sped up again, there are 3 things I can think of that may cause this:

1) The geometry between the lidar and the preceding vehicle has changed e.g. the lidar vehicle has hit a pothole/bump in the road that caused lidar to tilt/move for the frames mentioned.
2) Using the frame rate for the time dimension in the calculation is an assumption on the timeliness of the data sets. Providing a timestamp for the lidar data would rule this out/make the calculation more accurate. So, if frames 4, 9, & 15 were taken earlier than 1/frame rate (w.r.t to the preceding frame that would cause the effect observed and the delta distance would be smaller.
3) I have worked with lidars in my current role. See (https://www.guidance.eu.com/cyscan & https://www.guidance.eu.com/scenescan) I suspect the outlier observed in some frames may be caused significant changes in the intensity of the reflected laser beam, particularly from the rear lights on the preceding vehicle. Significant changes in intensity can cause the lidar to measure the distance as slightly less than it really is. These effects are calibrated out (on the sensors I work with) but can still cause a small error in distance measurement.

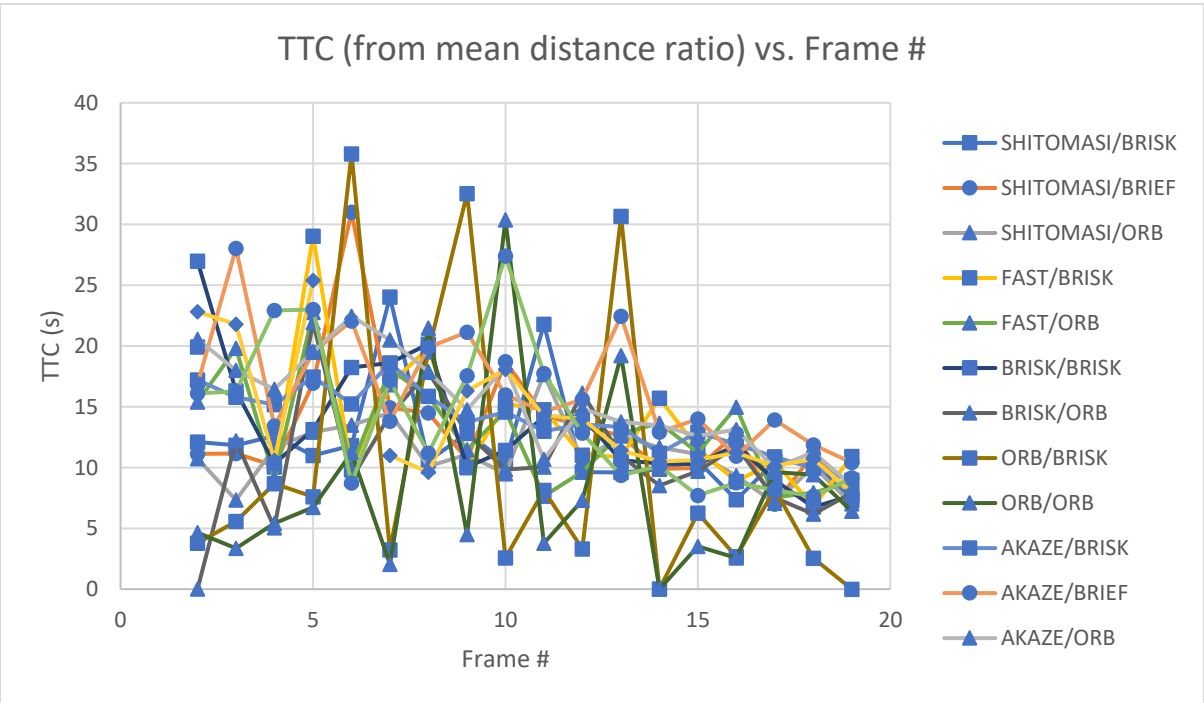
FP.6 Performance Evaluation 2

I have run several detector/descriptor combinations. The results are show the charts below:



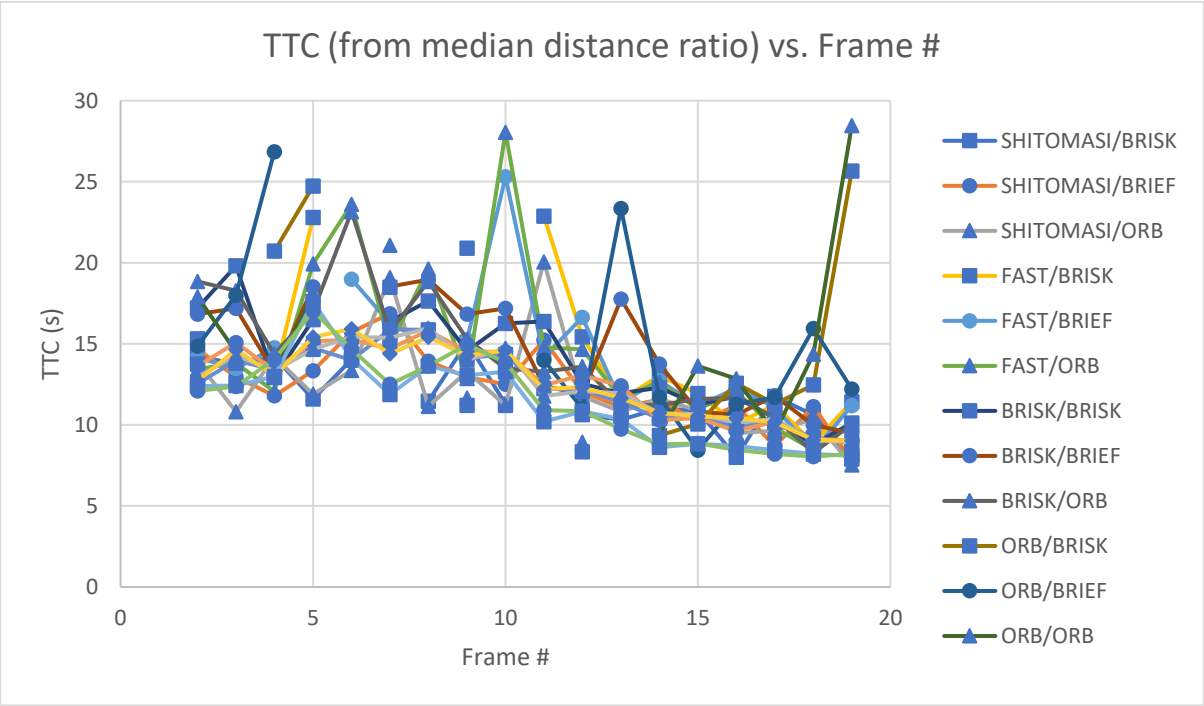
The chart shows the number of keypoint distance ratios computed for each frame. The BRISK detector far exceeds other detectors in the number of points found.

The chart below shows the TTC derived for detector/descriptor combinations:

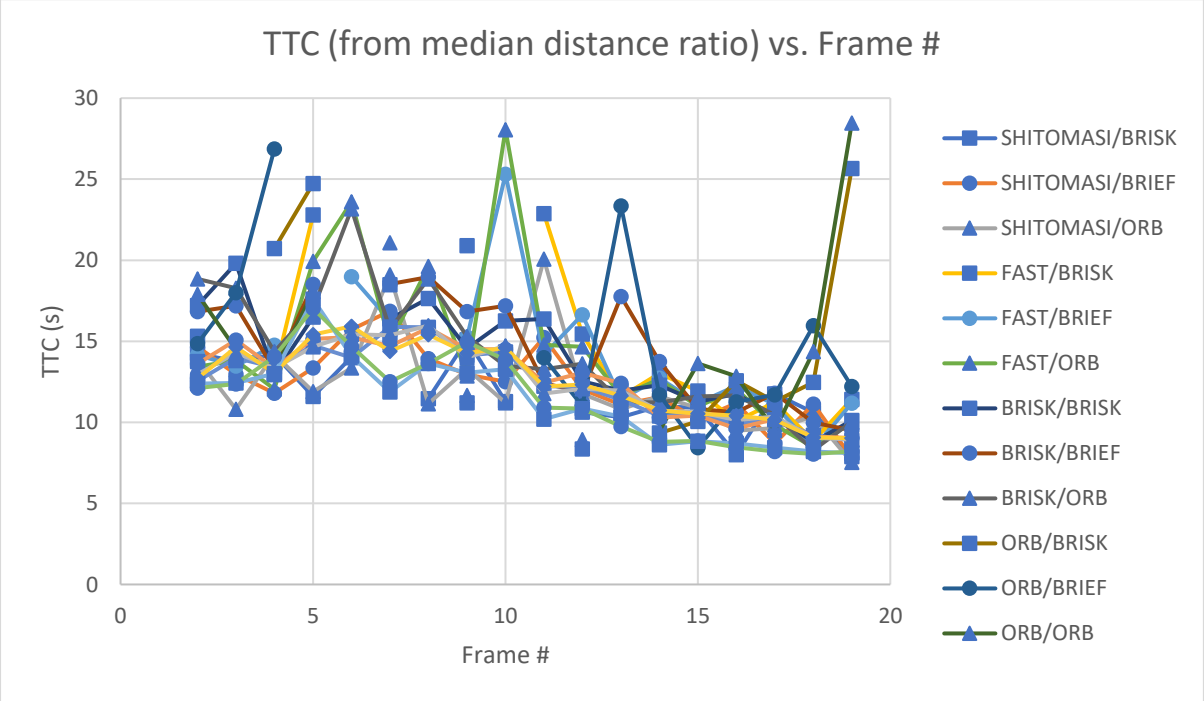


There is significant variation in TTC between the different detector/descriptor combinations. I suspect this may be due to how I'm rejecting outlier ratios from the set found for each frame.

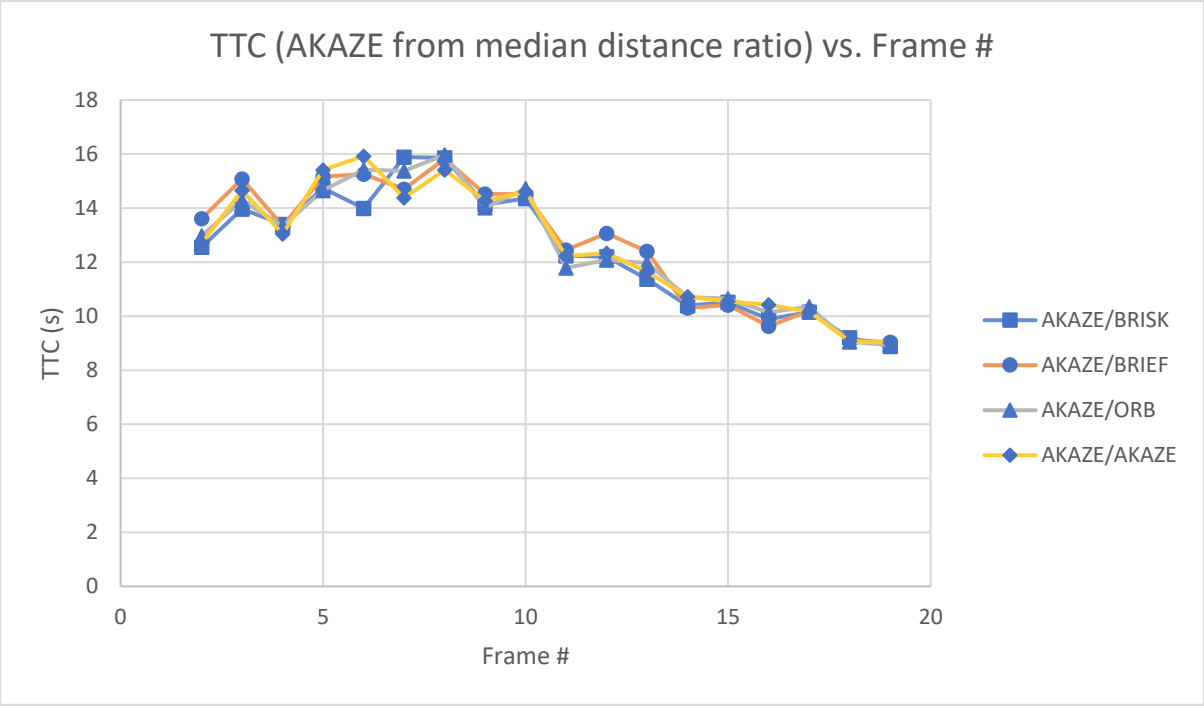
I redid the TTC calculation using the median distance ratio (instead of the mean) and that produced less noisy results:



There are still several outliers, but if these are removed the plot looks like:



Of the detector/descriptor combinations the AKAZE detector seems to produce the best results:



I think the camera result can be in error when the change in Euclidean separation of keypoint get close to or below 1 pixel. This is quite common in the frame set a the vehicle is travelling quite slowly and the frame quite is relatively fast. This could be made better by:

a) Using a camera with the higher resolution
b) Slowing the frame rate down to achieve a larger scale change (of the preceding vehicle) between consecutive images.