Track an Object in 3D Space

Arjun Agrawal
15th July 2021

FP.1 Match 3D Objects
Implement the method "matchBoundingBoxes", which takes as input both the previous and the current data frames and provides as output the ids of the matched regions of interest (i.e. the boxID property). Matches must be the ones with the highest number of keypoint correspondences.

A matrix is created of the size number of bounding boxes in the current frame by number of bounding boxes in the previous frame. Each time a key point is found in both a bounding box in the current frame and in the previous frame, the corresponding box in the matrix is given a point. At the end of the matchBoundingBoxes function, the largest value in the matrix is found. The corresponding bounding boxes in the current and previous frame are matched and the specific row and column is zeroed before the process is repeated.

FP.2 Compute LiDAR-based TTC
Compute the time-to-collision in second for all matched 3D objects using only Lidar measurements from the matched bounding boxes between current and previous frame.

The time to collision formula from lesson 2 is used in this part of the project.

$$t_{ttc} = \frac{d_1}{v_0} = \frac{d_1 \Delta t}{d_0 - d_1}$$

In order to use this formula, I needed to find the distance to the car ahead of the ego in the previous frame (d_0) and the current frame (d_1) with the help of the LiDAR sensor. RANSAC is used on the LiDAR points of the car to create a plane which can be used to eliminate erroneous points and estimate the tail of the car.

The RANSAC algorithm is adapted from my midterm project and hence the point cloud library is not used.

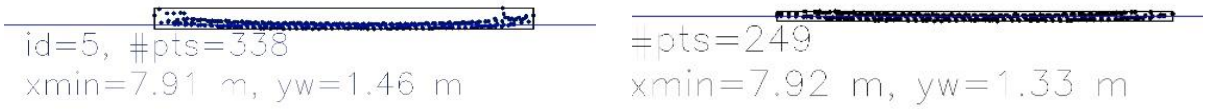

Figure 1.0
Left: unfiltered LiDAR points from of the car ahead of the ego car. Right: LiDAR points filtered with the help of RANSAC.

FP.3 Associate Keypoint Correspondences with Bounding Boxes
Prepare the TTC computation based on camera measurements by associating keypoint correspondences to the bounding boxes which enclose them. All matches which satisfy this condition must be added to a vector in the respective bounding box.
First all of the key points are looped over and checked if they are in the region of interest. Once this requirement is met, in order to remove erroneous points and outliers, the Euclidean

distance of each point is calculated and only point within two standard deviations from the mean are included in the correspondence.

FP.4 Compute Camera-based TTC
Compute the time-to-collision in second for all matched 3D objects using only keypoint correspondences from the matched bounding boxes between current and previous frame.

The TTC formula from lesson 3 was used to compute the camera-based TTC. Instead of height ratio, the average distance ratio between the keypoints of the current frame and the keypoints of the previous frame are used.

FP.5 Performance Evaluation 1
Several examples (2-3) have been identified and described in detail. The assertion that the TTC is off has been based on manually estimating the distance to the rear of the preceding vehicle from a top view perspective of the LiDAR points.

In order to evaluate the LiDAR time to collision function the distance, velocity and TTC values calculated by the function are plotted as seen in the figures below. Based on the TTC formula used in the function,

$$t_{ttc} = \frac{d_1}{v_0} = \frac{d_1 \Delta t}{d_0 - d_1}$$

the key information which we need to extract from the lidar is the distance to the car directly in front of the ego car. Figure 3.0 shows that there is a gradual decrease in the distance between the ego car and the car directly ahead. Hence at first glance there are no outliers which can be identified.

Frame	TTC Manual (s)	TTC lidar (s)	Velocity (m/s)	Distance (m)
1	12.80	12.97	0.61	7.933
2	13.13	11.21	0.70	7.865
3	13.69	19.05	0.41	7.803
4	14.05	11.91	0.65	7.785
5	14.12	16.04	0.48	7.721
6	14.21	11.23	0.68	7.671
7	14.02	15.79	0.48	7.611
8	13.56	15.37	0.49	7.550
9	12.78	12.66	0.59	7.476
10	11.83	12.16	0.61	7.393
11	10.82	9.30	0.79	7.379
12	9.95	10.71	0.68	7.293
13	9.25	8.77	0.82	7.217
14	8.93	10.80	0.66	7.131
15	8.98	8.09	0.87	7.042
16	9.31	8.81	0.79	6.976
17	9.63	10.29	0.67	6.940
18	8.66	8.31	0.82	6.826

Table 1.0: Distance, velocity and time to collision data from LiDAR TTC system

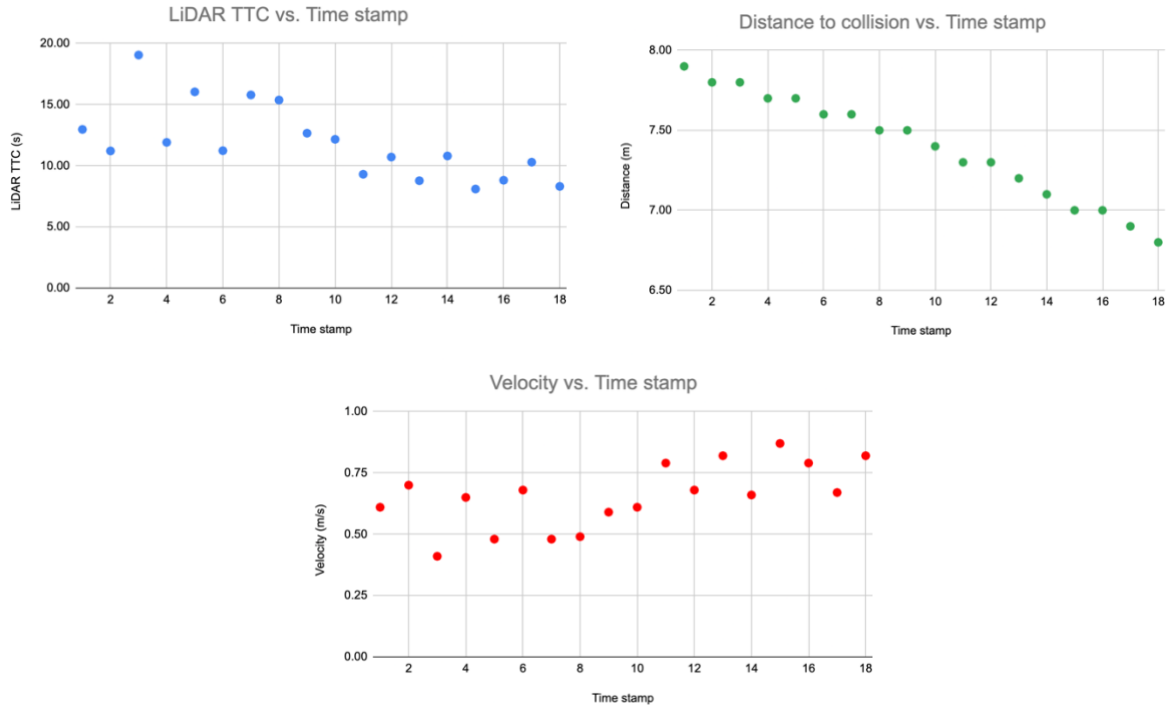

Figure 2.0: Distance, velocity and time to collision graphs from LiDAR TTC system

Next we look at the top down view of the filtered LiDAR points in Figure 3.0. The figure shows the LiDAR points found on the bumper of the car ahead. RANSAC is used to remove outliers after which the point closest to the ego car is used to provide the distance measurement. The RANSAC algorithm is given a distance tolerance of 0.03m which allows for tight clustering of points on the car's bumper. Due to which, I do not see any major issues with the distance estimation carried out by the LiDAR.

A key assumption made by the TTC LiDAR function is that velocity between each time stamp is linear. This could be a problematic assumption as when there is little relative movement in the car ahead (ie. there is only a small change between the current and next distance reading made by the RANSAC algorithm), there is an drastic increase in the TTC. If we take a look at frames 3 and 4, we can see that there is a small change in relative distance of 2cm which causes a drastic spike in the LiDAR TTC estimate. The TTC estimate at point 3 peaks by 7.8s and then dips back down at point 4 by 7.1s. The denominator in the formula tends to 0 and amplifies the noise in the system. For comparison, at point 3 the LiDAR TTC estimate is 19.05s while the manual estimate is manual TTC estimate is 13.69s. That's a difference of more than 5s.

Another instance of this problem can be seen in points 10 and 11 where there is an incremental change in relative distance leading to a sudden change in LiDAR TTC of almost 3s (30% change in TTC estimate).

Figure 3.0: Top down representation of LiDAR data after RANSAC filtration

FP.6 Performance Evaluation 2

All detector / descriptor combinations implemented in previous chapters have been compared with regard to the TTC estimate on a frame-by-frame basis. To facilitate comparison, a spreadsheet and graph should be used to represent the different TTCs.

All of the detector and descriptor combinations were applied to the camera TTC estimation system. A spread sheet has been attached assessing all of them. In order to systematically asses the viability of each of the combinations the RMSE is calculated with reference to both the manual TTC system and the LiDAR TTC system.

It is found that the Shi-Tomasi detector serves us well in this scenario. My main criteria for selecting a suitable detector and descriptor pair is stability and accuracy. This is because we cannot have the system jerking due to sudden irregular changes in the TTC estimate. Along with there is also importance given to computational speed as described and discussed in the midterm project. Table 2.0 summaries my findings from the midterm project. Hence my top three contenders for the camera TTC system are:

1) Shi-Tomasi detector with Brisk keypoints
2) Shi-Tomasi detector with Orb keypoints
3) Shi-Tomasi detector with Brief keypoints (though Shi-Tomasi with Sift keypoints has a marginally smaller RMSE, Brief is more than twice as fast)

Total time taken for keypoint detection and descriptor computation across all 10 images							
		DESCRIPTOR					
		BRISK	BRIEF	ORB	FREAK	AKAZE	SIFT
DETECTOR	SHITOMASI	545.015	453.158	535.375	1606.09	N/A	1000.74
	HARRIS	591.84	576.049	632.368	1543.24	N/A	1150.68
	FAST	122.621	71.5719	145.319	1171.03	N/A	893.697
	BRISK	2048.82	2129.76	2397.91	3128.72	N/A	3568.59
	ORB	702.517	548.24	887.921	1535.8	N/A	2081.97
	AKAZE	3857.72	3608.57	3821.64	4854.34	6370.31	4766.42
	SIFT	5686.5	4616.62	N/A	5693.62	N/A	7246.53

Table 2.0: Table summarising computational speed of every detector-descriptor combination discussed

Summary of camera detectors and descriptor combinations		Descriptors					
		Akaze	Brief	Brisk	Freak	Orb	Sift
Detectors	Shi-Tomasi	NIL	5.202	2.402	3.983	3.817	3.836
	Harris	NIL	44.772	105.059	11.879	13.710	11.886
	Akaze	11.931	11.953	11.963	11.957	11.960	11.926
	Sift	NIL	12.576	20.163	13.245	NIL	NIL
	Brief	NIL	12.499	11.789	11.879	13.710	11.886
	Orb	NIL	13.889	11.601	18.801	13.745	15.084
	Fast	NIL	12.307	11.927	11.923	11.958	13.223

Table 3.0:
Table summarising effectiveness of every detector-descriptor combination discussed in the camera TTC system

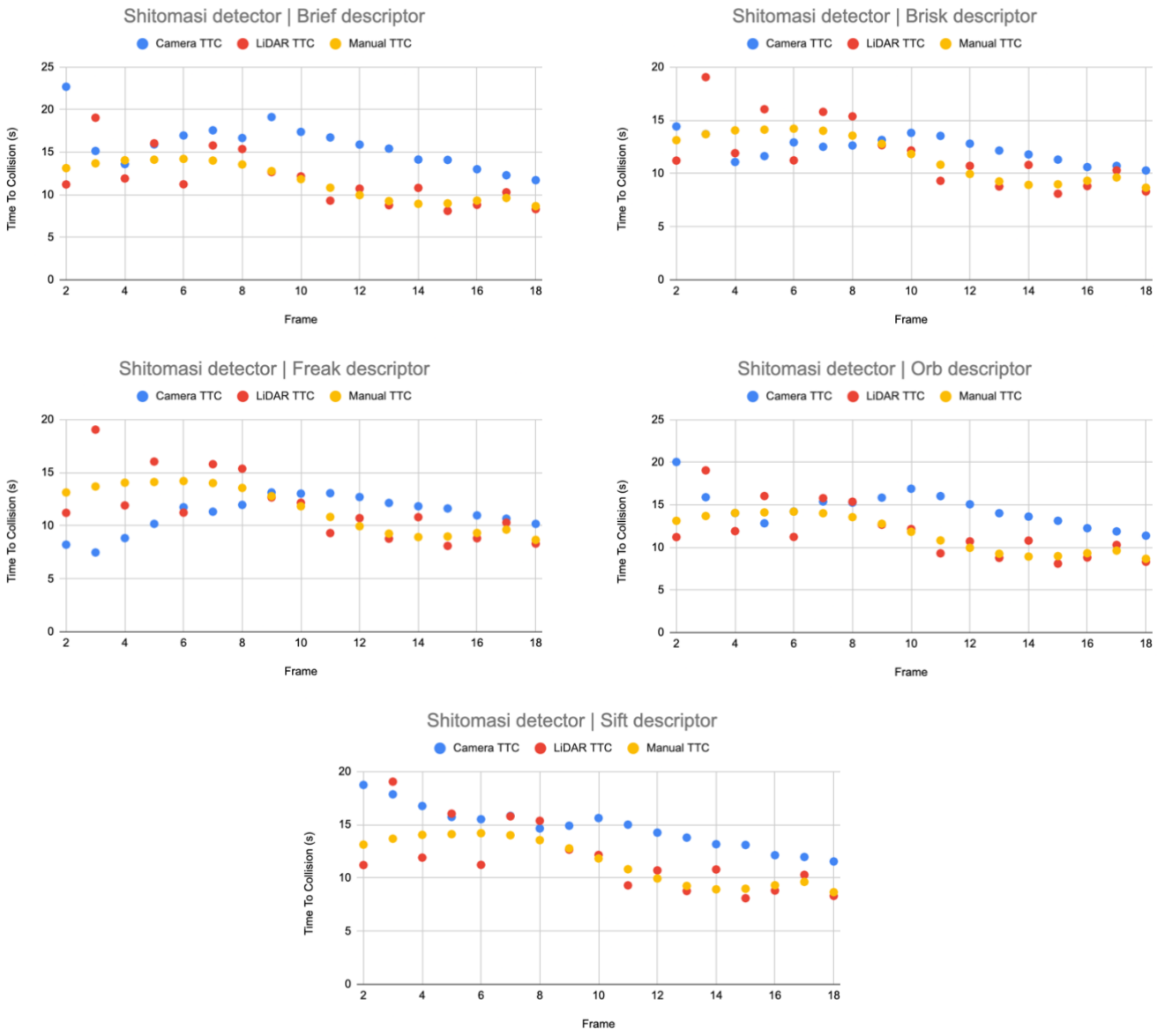

Figure 4.0: TTC estimates of camera based system with Shi-Tomashi detector

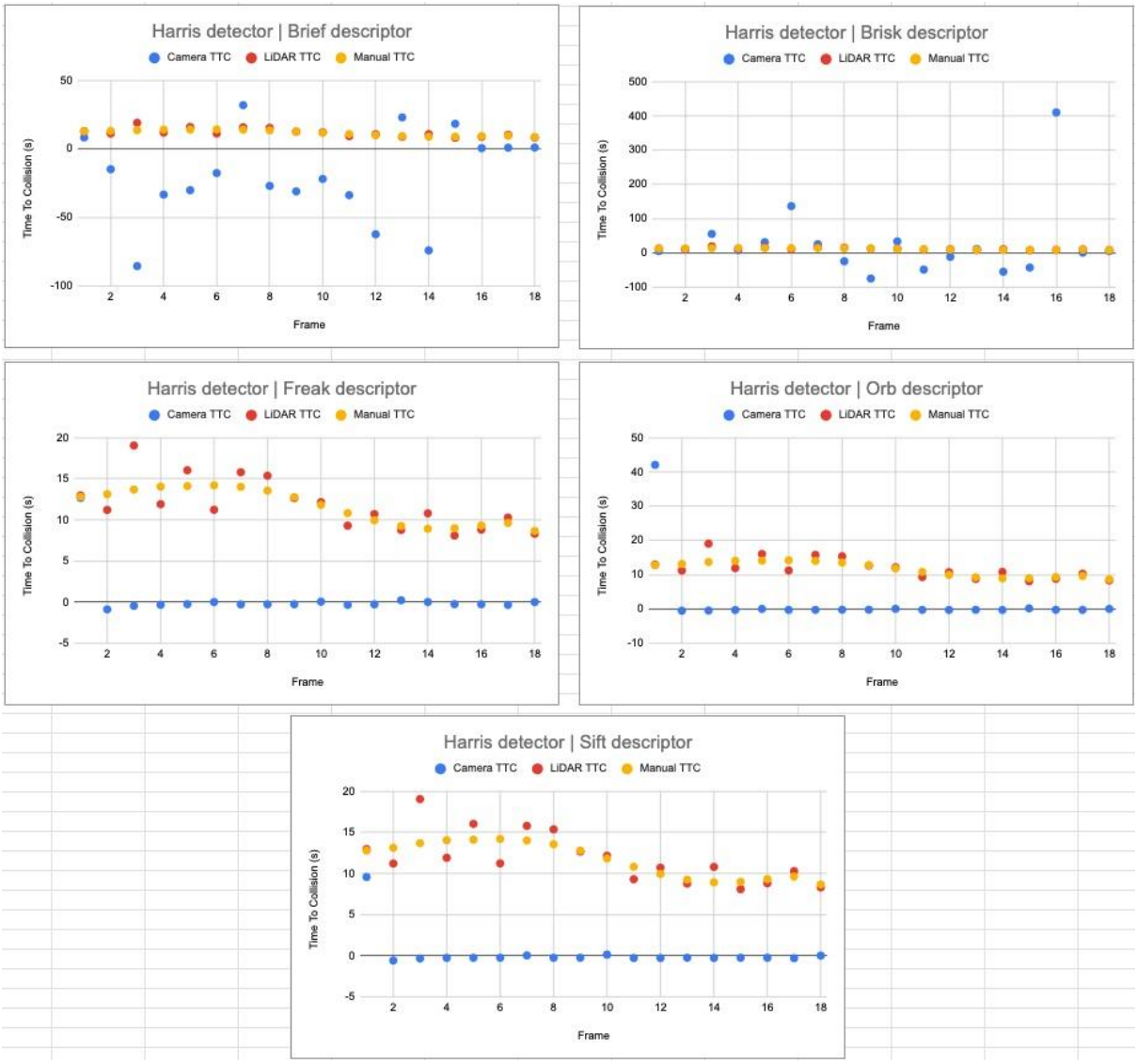

Figure 4.0: TTC estimates of camera based system with Shi-Tomashi detector

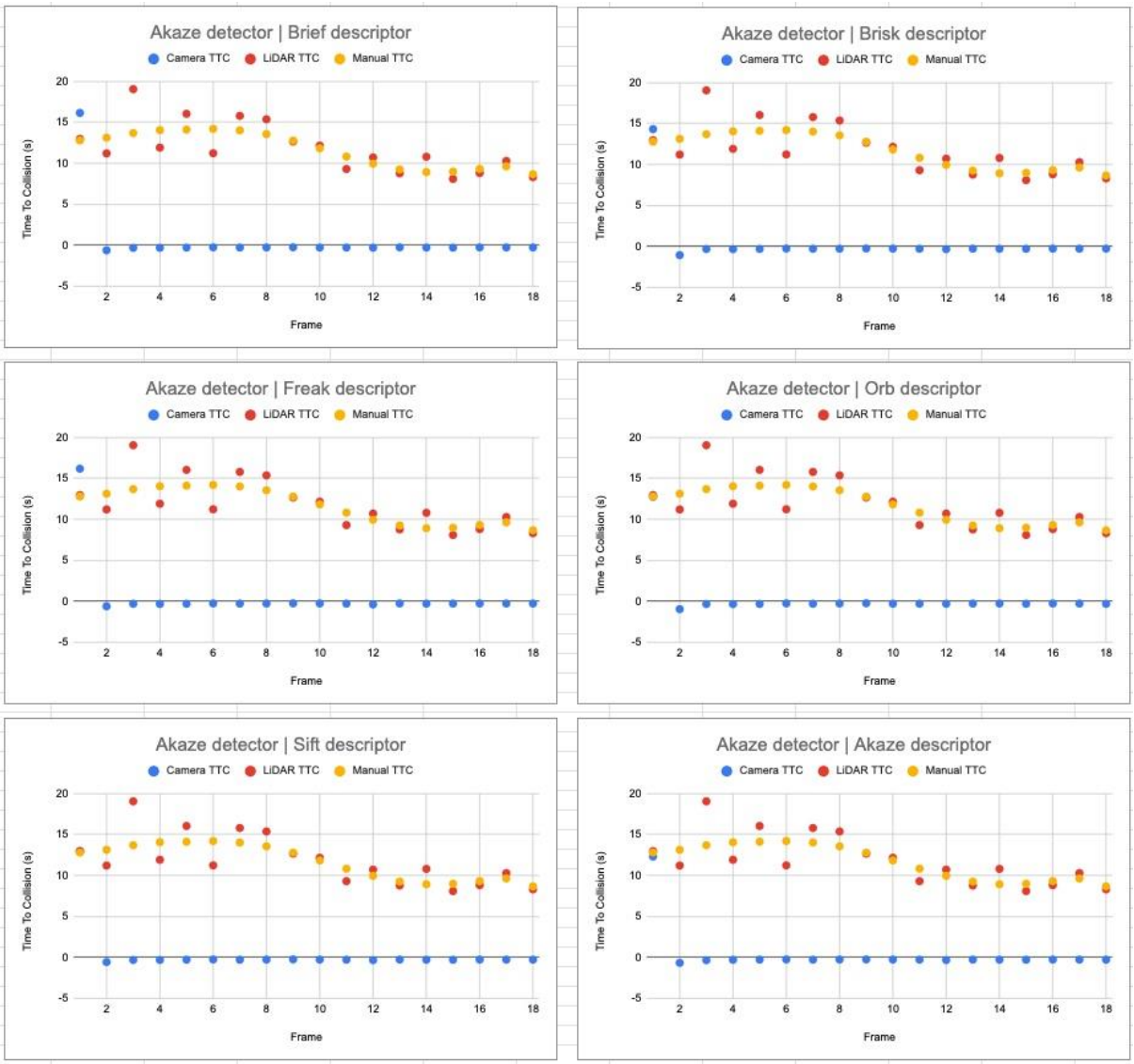

Figure 5.0: TTC estimates of camera based system with Akaze detector

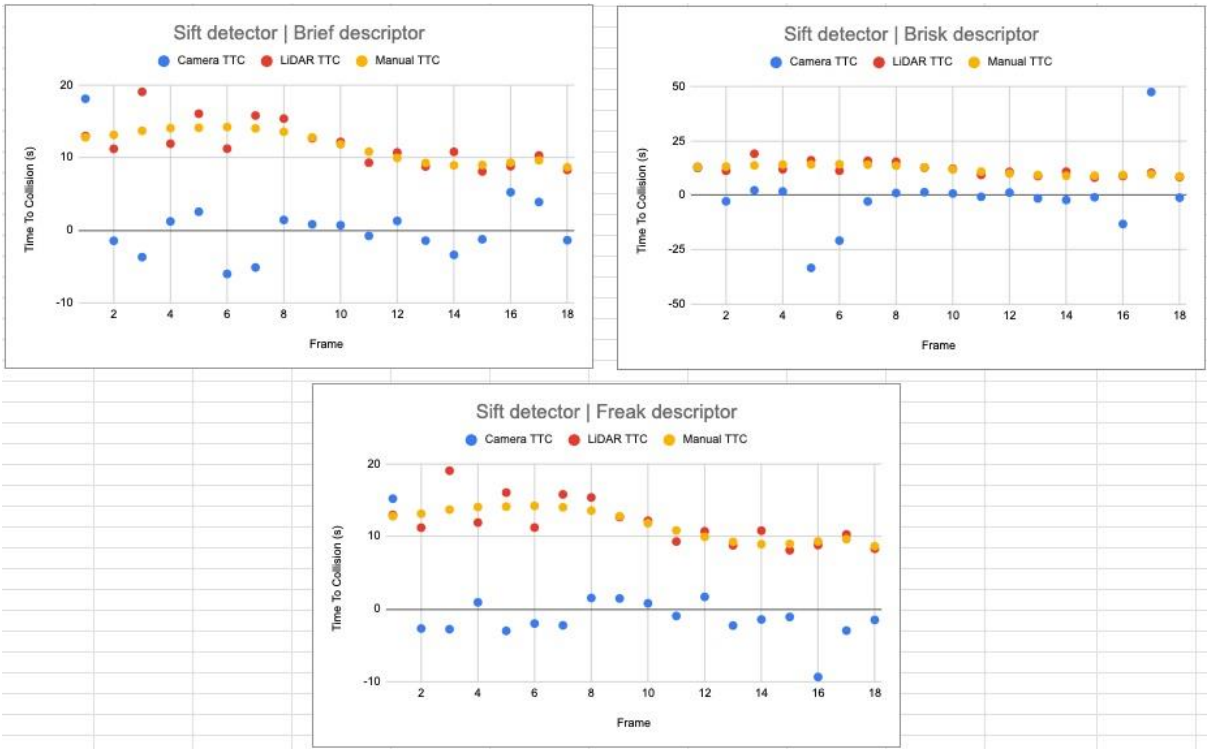

Figure 6.0: TTC estimates of camera based system with Sift detector

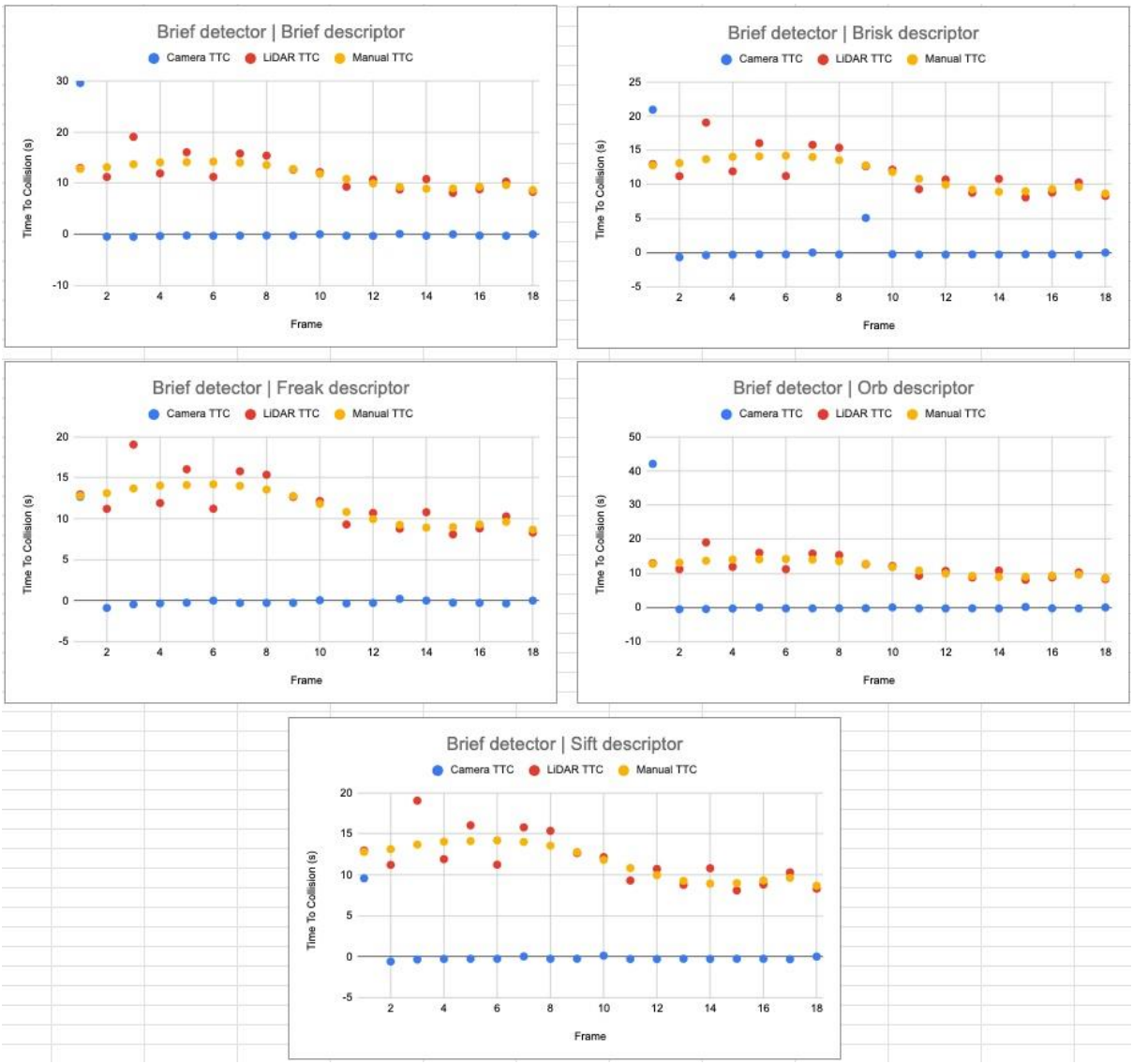

Figure 7.0: TTC estimates of camera based system with Brief detector

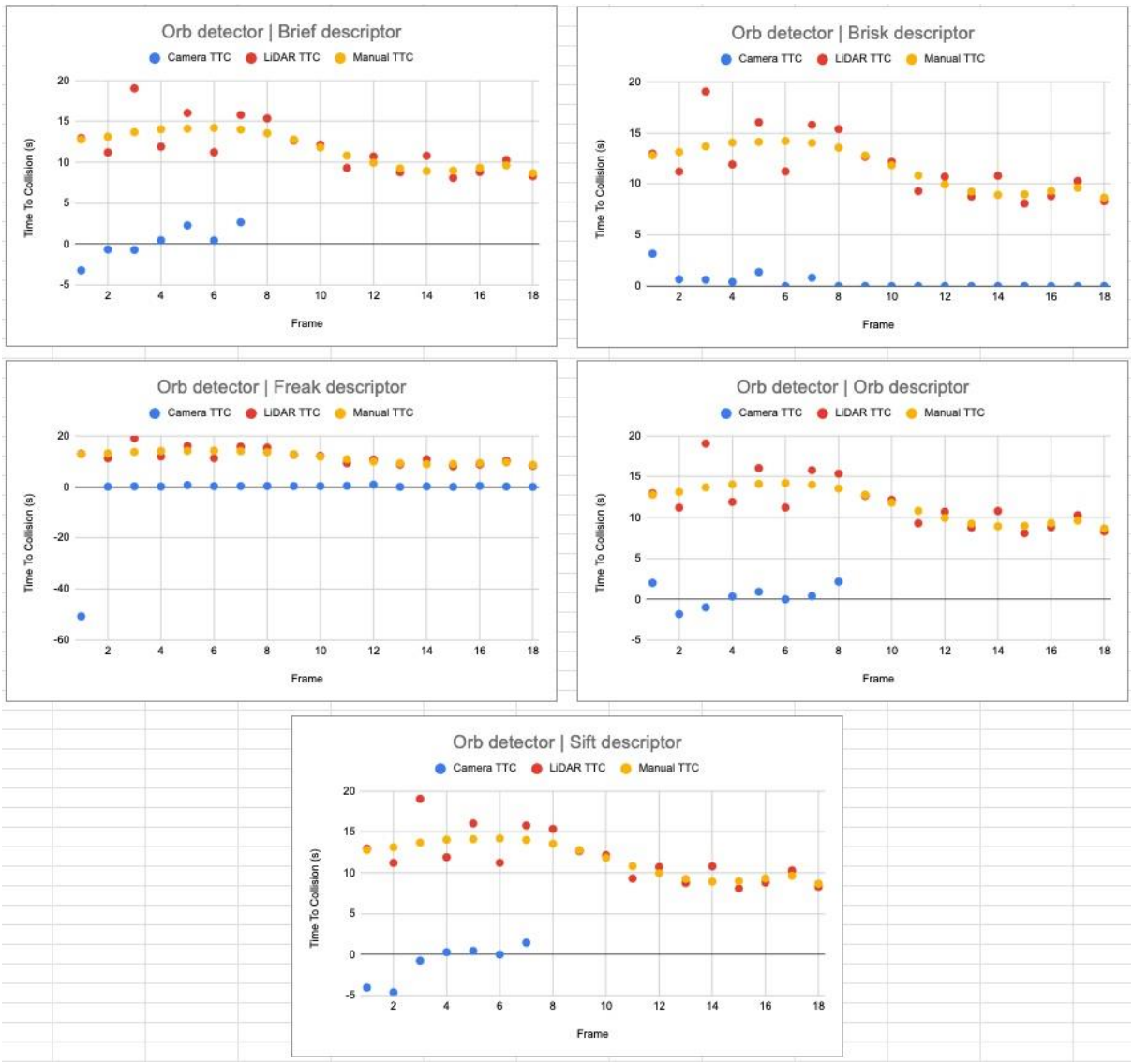

Figure 7.0: TTC estimates of camera based system with Orb detector

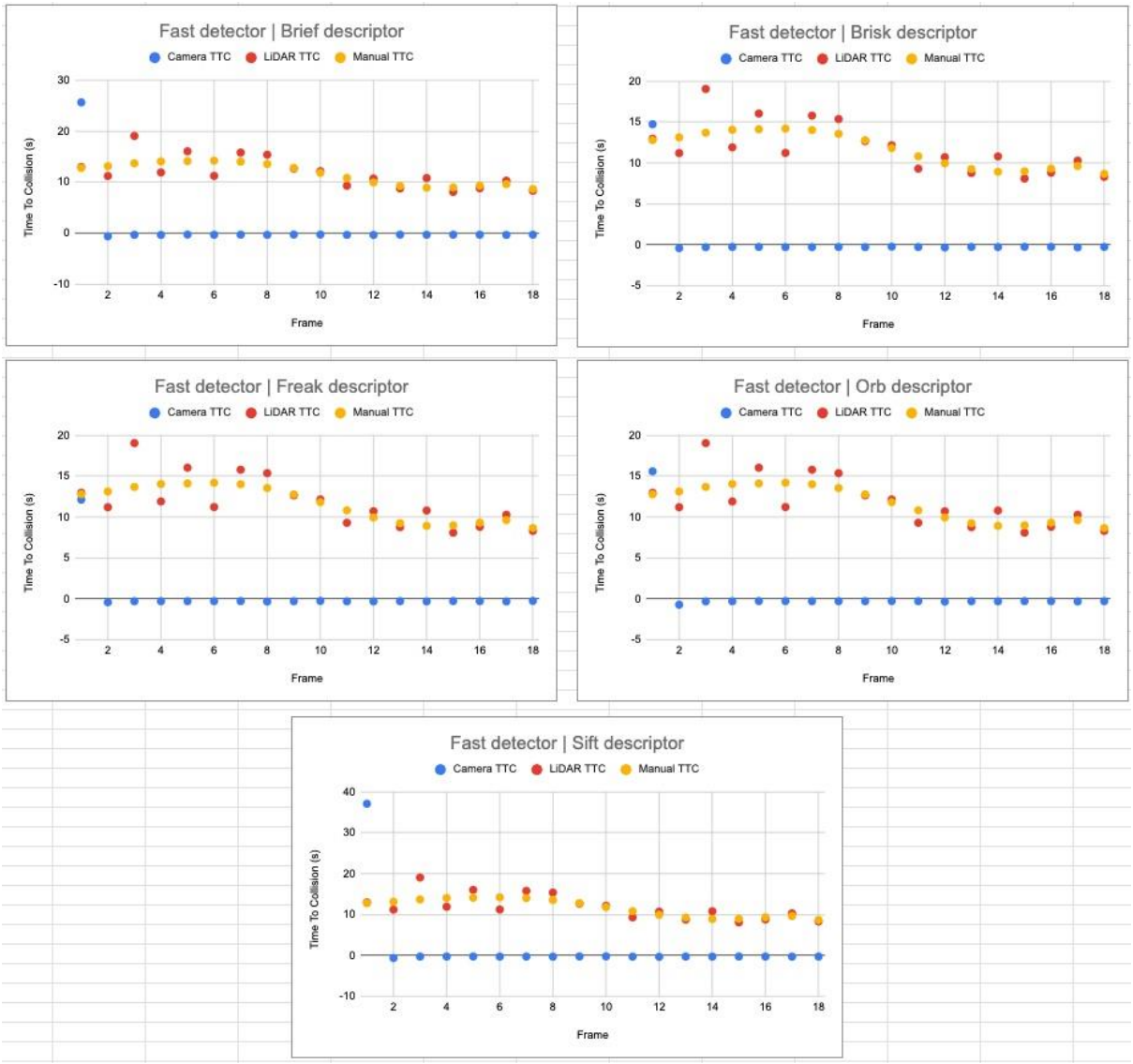

Figure 8.0: TTC estimates of camera based system with Fast detector