FINAL PROJECT CAMERA WRITE UP

Task FP1 : Matching 3D Objects

The function 'matchBoundingBoxes' receives the arguments of data from previous frame and current frame and code is written to identify the matching bounding boxes' IDs (i.e. the boxID property). The IDs of the bounding boxes' (region of interest) containing the keypoints are identified and stored in the lists separately, by iterating through the keypoint matches. Each of the bounding boxes is assigned the match candidate with the maximum number of occurrences, achieved through keypoint correspondences, and the IDs are collected.

Task FP2 : Compute Lidar-based TTC

The time to collision is calculated based on the constant velocity model. Therefore, it is calculate by the formula TTC = (d1 * dT) / (d0 - d1) as provided in the lesson under "Camera" topic of this course. The function 'computeTTCLidar' receives the lidar data from the previous frame and the current frame, and the framerate.

The inverse of the framerate is the time between two measurements i.e. dT. The lidar data are chosen by taking the median x-distance for eliminating random outliers and only the lidar points present in the matched bounding boxes from the previous and current frames are used. The initial distance measurement d0 is from the previous frame and the nearing distance measurement d1 is from the current frame.

Task FP3 : Associate Keypoint Correspondences with Bounding Boxes

The function 'clusterKptMatchesWithROI' receives the keypoints from the previous and current frames and the specific keypoint matches property of the bounding boxes ("kptMatches"). The function adds the keypoints correspondences to the "kptMatches" and outliers are removed based on the Euclidean distance between them in relation to all the matches in the bounding box.

Task FP4 : Compute Camera-based TTC

The function 'computeTTCCamera' receives the keypoints from the previous and current frames and also the "kptMatches" parameter. The computation of the TTC is based on the median distance ratio as explained in the lesson "Estimating TTC with a Camera".

TTC = -dT/ (1-medDistRatio)

Task FP5 : Performance Evaluation 1

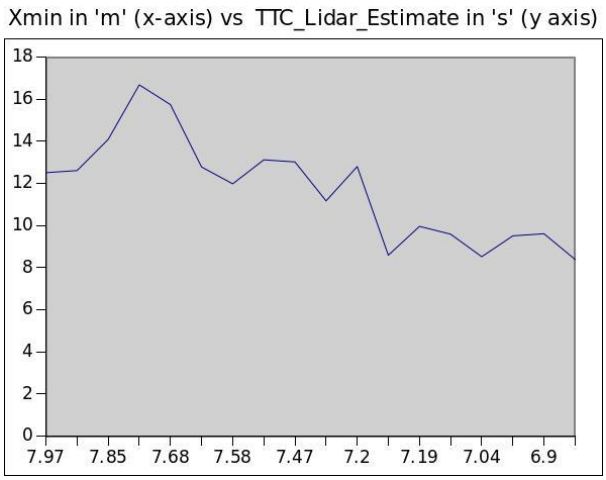


Image	Xmin distance (m)	TTC_Lidar (s)
1	7.97	12.51
2	7.91	12.61
3	7.85	14.09
4	7.79	16.68
5	7.68	15.74
6	7.64	12.78
7	7.58	11.98
8	7.55	13.12
9	7.47	13.02
10	7.39	11.17
11	7.2	12.8
12	7.27	8.59
13	7.19	9.96
14	7.13	9.59
15	7.04	8.52
16	6.83	9.51
17	6.9	9.61
18	6.81	8.39

The data from the TTC estimate from the LIDAR seems not to be completely in tandem with the decreasing distance between the driven vehicle and the preceding vehicle (i.e. it not showing a completely decreasing TTC trend with respect to decreasing distance X min trend).

We can also observe that, at image frames 5,6,7 the values seems to be plausible as the decreasing distance trend shows a considerable decreasing trend of the TTC estimate which is required for our case. However, in the 8^{th} frame the TTC estimate is greater than the previous estimate though the distance is reduced. This is not a plausible situation.

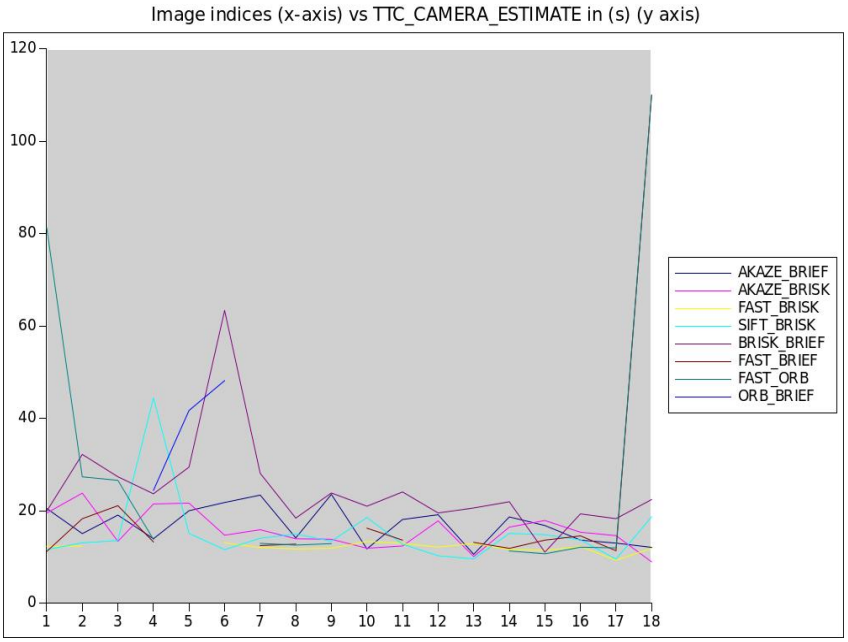
This can be made better if we incorporate the acceleration/deceleration characteristics and thereby shifting towards the acceleration based TTC estimate instead of the constant velocity model, as in reality the vehicles in these situations rarely move with constant velocities.

Task FP6 : Performance Evaluation 2

A set of top 8 detector-descriptor combos are chosen based on the total average time taken for keypoint detection and descriptor extraction as per the information obtained through the Mid-term project of the Camera module of this course.

These are AKAZE_BRIEF (2.55 ms), AKAZE_BRISK (2.35 ms), FAST_BRISK (2.26 ms), SIFT_BRISK (1.66 ms), BRISK_BRIEF (1.12 ms), FAST_BRIEF (1.26 ms) , FAST_ORB (1.11 ms), ORB_BRIEF (0.615 ms). /* Note : the time mentioned within the brackets are total average time for detection and description of keypoints */

TTC _CAMERA_ESTIMATE in (s) For Different Detector and Descriptor Combo								
Image/ Combo	AKAZE_BRIEF	AKAZE_BRISK	FAST_BRISK	SIFT_BRISK	BRISK_BRIEF	FAST_BRIEF	FAST_ORB	ORB_BRIEF
1	20.48	19.49	12.3	11.59	19.89	11.17	81.26	INF
2	15.04	23.81	12.34	13.05	32.18	18.27	27.33	INF
3	19.05	13.36	INF	13.54	27.35	21.09	26.57	INF
4	13.93	21.46	12.88	44.42	23.66	13.24	13.78	24.4
5	20	21.64	INF	15.07	29.44	INF	INF	41.7
6	21.78	14.7	13.03	11.57	63.34	INF	INF	48.15
7	23.38	15.88	12.04	14.05	28.09	12.46	12.93	INF
8	14.11	13.92	11.72	14.95	18.4	12.76	12.56	34.62
9	23.53	13.82	11.86	13.4	23.84		12.87	INF
10	11.73	11.87	13.34	18.52	20.97	16.21	INF	
11	18.1	12.36	12.94	12.75	24.06	13.63	INF	21.39
12	19.13	17.8	12.11	10.25	19.51	INF	INF	
13	10.56	10.08	12.91	9.57	20.59	13.14	INF	INF
14	18.65	16.45	11.6	15.08	21.94	11.85	11.29	25.88
15	16.79	17.87	11.4	14.81	11.03	13.63	10.66	INF
16	13.63	15.34	12.25	13.68	19.31	14.55	12.1	14.67
17	12.98	14.6	9.29	9.53	18.28	11.34	11.95	INF
18	12.06	8.92	11.86	18.68	22.42	109.96	109.96	



The combos AKAZE_BRIEF, AKAZE_BRISK, SIFT_BRISK, BRISK_BRIEF seems to give a somewhat reasonable TTC_Camera_Estimate. However, even SIFT_BRISK has some unexpected large estimate : 44.42s at image frame 4 and similarly BRISK_BRIEF has a large valueof 63.34 s at image frame 6. **Hence, AKAZE_BRIEF, AKAZE_BRISK are somewhat reliable and recommended among these combinations**.

The detectors descriptor combos (FAST_BRIEF, FAST_ORB, ORB_BRIEF) are not recommended because of certain unwanted/unreliable estimates because the based on the keypoints detected and described by these combo the median distance ratios will be nearly 1 thus giving “inf” value of TTC.