FP.1 Match 3D Objects

this task is implemented in the *matchBoundingBoxes* function. The function algorithm is as following:

1. get a bounding box from the previous frame (outer loop)
2. get a bounding box from the current frame (inner loop)
3. iterate over all matches and check it is within the both bounding boxes of prev and curr.
4. then the bounding box with the highest kpNum from the current frame is matched with the bouding box from the previous frame.
5. apply this to all bounding boxes from previous frame

FP.2 Compute Lidar-based TTC

This task in implemented in the function *computeTTCLidar*. Before calculating the TTC a filteration function named *filterLidarPoints* is called where it implement a euclidean clustering algorithm with kd tree. The tree structure is in the *kdtree.h* file.

The distance thresh hold chosen is based upon several experiment to exclude outliers.

FP.3 Associate Keypoint Correspondences with Bounding Boxes

This task is implemented in the function *clusterkptMatchesWithROI*.

1. Iterating over all keypoint matches and it in temporary bounding box
2. applying euclidean distance mean to exclude outliers

FP.4 Compute Camera-based TTC

This task is implemented in the function computeTTCCamera.

FP.5 Performance Evaluation 1

Based on the markers set to visualize the lidar data from a top perspective, the TTC can be estimated to be around 7.5 to 8.9 by visual inspection. Given that we will consider any TTC value that is far from this range as an off value.

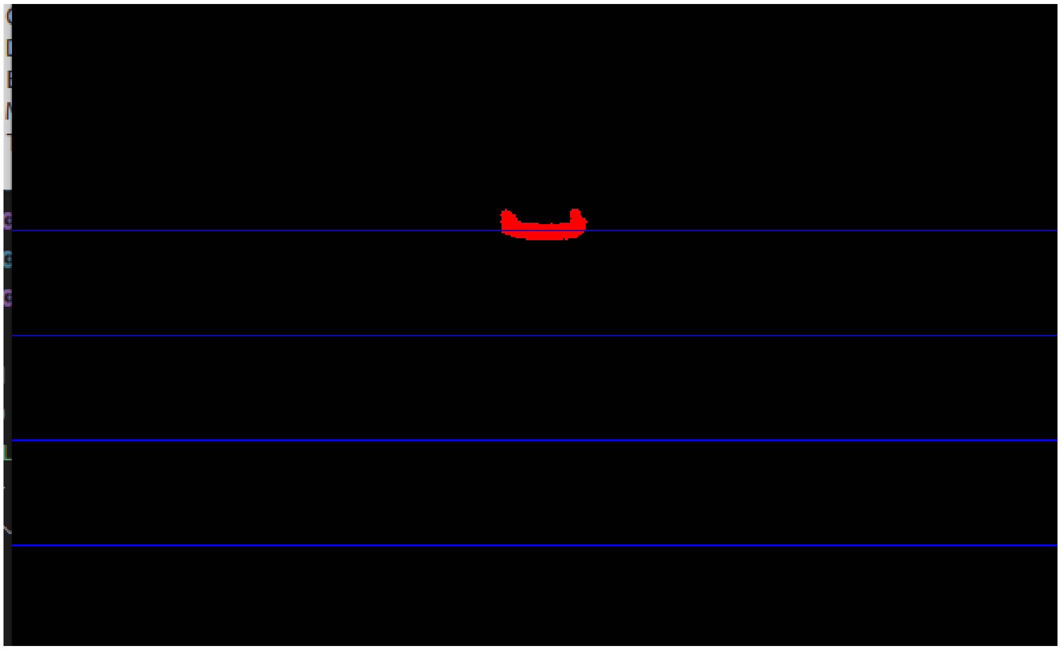

Figure 1unfiltered lidar data with visual markers

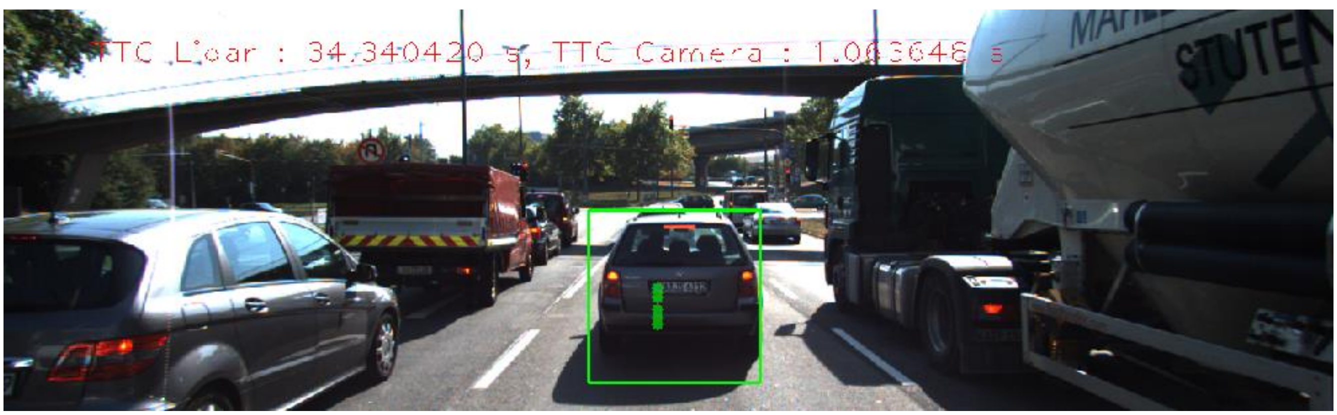

Figure 2 way off Lidar TTC estimation of 34.34 s

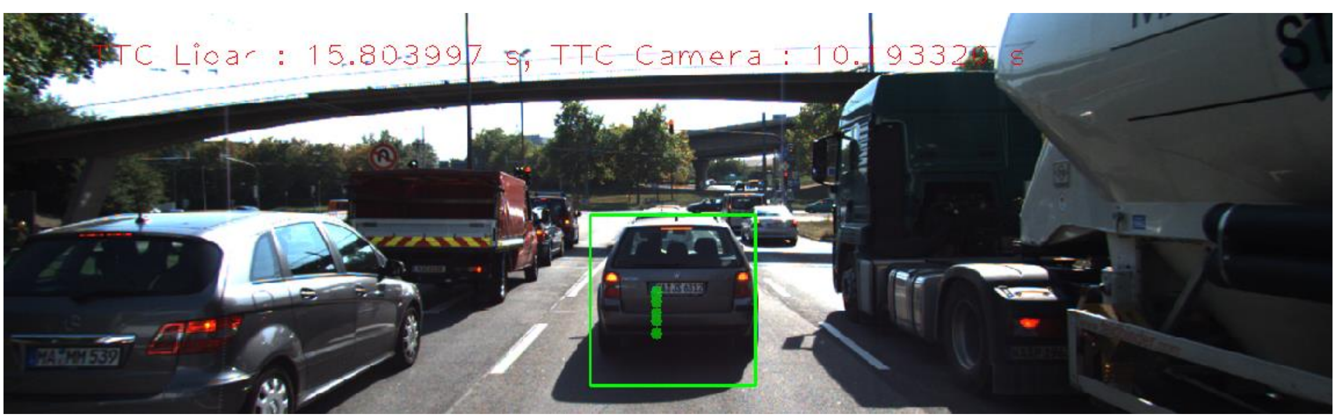

Figure 3 way off Lidar TTC estimation of 15.80 s before filtering the lidar points the values were way off the expected range. However, after implementing euclidean clustering algorithm it seems that there is still some of values. I can conclude that the filter implemented is not robust enough, even though it provided better results.

FP.6 Performance Evaluation 2

For this task I will be using the detector/descriptor combination evaluated from the midterm project and using the standard deviation as a comparison metric.

The tested combinations as follows (these are some of the best combinations that performed best in past performance evaluation).

Detector	Descriptor
FAST	BRISK
FAST	BRIEF
BRISK	BRISK
BRISK	ORB

1. FAST/BRISK

TTC Est.	4.61	6.95	0.83	0.22	2.68	Nan	0.77	0.07	2.81	Nan	2.41	0.14	16.13	0.17	5.36	5.21	5.4	0.58

Mean = 3.39625
Std. Deviation = 3.97507527

1. FAST/BRIEF

TTC Est.	10.46	4.06	13.34	10.19	8.18	4.23	14.13	-158.31	4.67	5.31	5.31	4.32	Nan	51.57	8.79	4.46	5.26	7.26

Mean = 0.19
Std. Deviation = 41.0791566

1. BRISK/BRISK

TTC Est.	4.61	6.95	0.83	0.22	2.68	Nan	0.77	0.07	2.81	Nan	2.41	0.14	16.13	0.17	5.36	5.21	5.4	0.58

Mean = 3.39625
Std. Deviation = 3.97507527

1. BRISK/ORB

TTC Est.	10.73	5.6	9.25	7.1	7.11	5.87	8.56	5.12	3.08	6.39	35.73	3.91	13.89	4.07	12.79	2.55	3.7	4.34

Mean = 8.32166667
Std. Deviation = 7.35907848

As shown in the previous detector/descriptor combinations. The choice may affect the resulted estimation. I believe there is post processing techniques that can validate these values. The appearance of way off values can be because a lot of reasons. One of which is the accuracy of the key points matches. Where an incorrect matches can lead to wrong estimation.

Based on the combinations tested. The BRISK/ORB seems to have the most plausible values with a mean within the range of the actual TTC computed manually. In addition the standard deviation shows a plausible dispersion of the estimations.