SFND_3D_Object_Tracking

FP.1 Match 3D Objects

Problem statement:

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.

Logic behind the solution:

Code can be found in matchBoundingBoxes function in camFusion_Student.cpp

1. Looping over the keypoint matches from current frame

2. Every keypoint match has IDs of keypoints from previous frame and current frame, with that we can fetch actual keypoint from respective frames.

3. Once keypoint is fetched, checking if both keypoints lie in any of the bounding box. This is done with the function findBoundingBox. This function checks both set of bounding boxes i.e. from previous frame and current frame, individually and if keypoint lies in any of the bounding box in both frames then it sets the bothframes flag and returns the Box IDs for both.

4. Afterwards, a map of <int, vector<int>> is created, where for every box in previous frame a matching candidate of current frame is noted. Here map.first i.e. int denotes Box ID in previous frame and vector<int> stores Box ID of boxes in current frame.

5. Once we are done looping all keypoint matches, for each box in previous frame the vector storing potential match candidates is fetched and

sorted. Occurances of each elements are then count and element with maximum occurance is found.

6. In last step, bbBestMatches map is filled with box ID from previous frame and the best matched Box ID from current frame.

FP.2 Compute Lidar-based TTC

Problem statement:

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.

Logic Behind the solution:

Code can be found in computeTTCLidarfunction in camFusion_Student.cpp

1. Frame rate is converted to time between two sensor measurements by taking reciprocal of the frame rate.

2. The x values from lidarPointsPrev and lidarPointsCurr are stored in seperate vector named prev_x_pts and curr_x_pts. Both the vectors are then sorted.

3. To increase reusability of code, both the x_val vectors are included in a vector<vector<double>> and median of both vectors was calculated. While calculating median, the even-odd size of vector was taken care and the median values were out in selected_x_val_vec.

4. TTC is calculated with formula

$$\text{TTC} = \text{x\_val\_curr} * \text{del\_t\_measurement} / (\text{x\_val\_prev} - \text{x\_val\_curr})$$

FP.3 Associate Keypoint Correspondences with Bounding Boxes

Problem statement:

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.

Logic behind solution:

Code can be found in clusterKptMatchesWithROI in camFusion_Student.cpp

1. By looping over the keypoint matches, euclidean distances for all matches are stored in euclidean_distances vector. This vector is then sorted after the end of the loop.

2. Median of the all euclidean distances is calculated, the even-odd size of vector was taken care.

3. Every keypoint match is checked if it lies in threshold around the median of euclidean distance. If yes, then the keypoints from previous frame and current frame are fetched.

4. These keypoints are further checked if they lie in ROI of the bounding box no not? If both the keypoints are found to be lying in the ROI of the bounding box then the respective keypoint match is included in kptMatches vector of the boundingBox.

FP.4 Compute Camera-based TTC

Problem statement:

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.

Logic behind the solution:

Code can be found in computeTTCCamera in camFusion_Student.cpp

The whole logic for this function is based on the „Compute camera based TTC“ concept from „Engineering Collision detection system“ lesson.

The code was found to work as it is and hence logic is not explained here.

FP.5 Performance Evaluation 1

Problem statement:

Find examples where the TTC estimate of the Lidar sensor does not seem plausible. Describe your observations and provide a sound argumentation why you think this happened.

Solution:

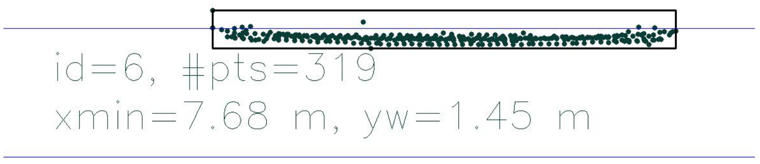
When visualizing the top view image of LiDAR points and final output image, it can be found that sometimes inspite of decreasing distance to next car the TTC increases. The tabel shown below displays the values of TTC and distance to next car.

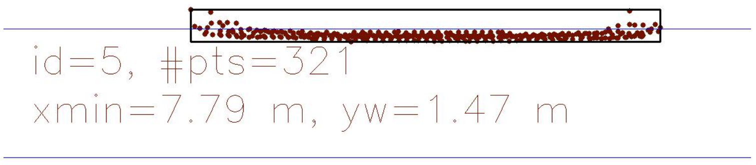
It can be observed from the sample numbers 3-4, 7-9 and 15-16, that distance to next car decreases but the TTC increases. Inversely, for sample number 11-12 the distance to next car increases but TTC sharply decreases.

Image ID	Distance to next car in m	TTC in sec
1	7.91	12.51
2	7.85	12.61
3	7.79	14.09
4	7.68	16.68
5	7.64	15.90
6	7.58	12.67
7	7.55	12.17
8	7.47	13.12
9	7.43	12.80
10	7.39	11.17
11	7.20	12.80
12	7.27	9.01
13	7.19	9.96
14	7.13	9.53
15	7.04	8.57
16	6.83	9.51
17	6.90	9.68
18	6.81	8.40

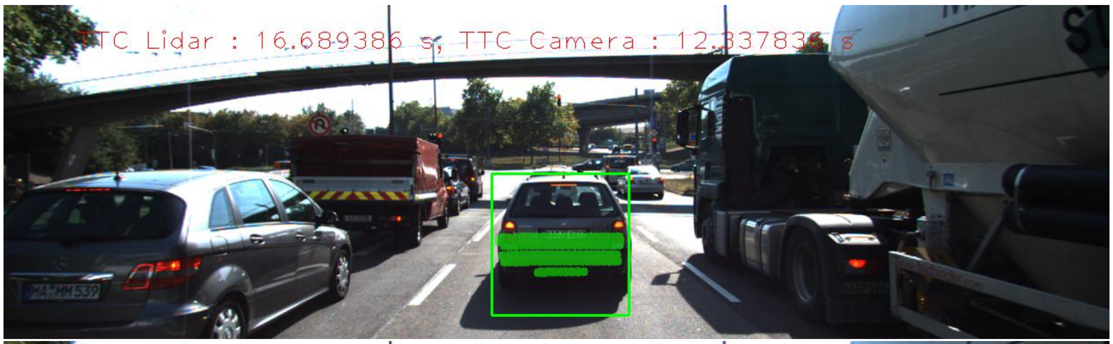
Below are the some images which supports the above mentioned claim and the figures in the table.

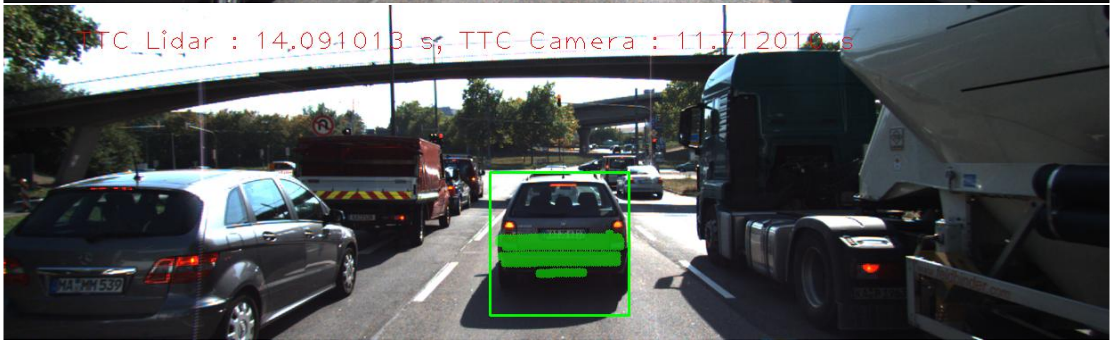
Top view of image id 3-4



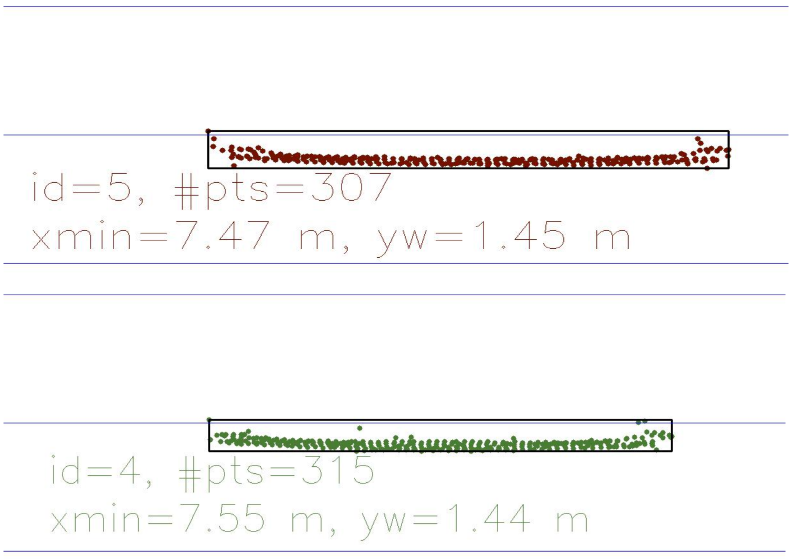


Front view of image id 3-4

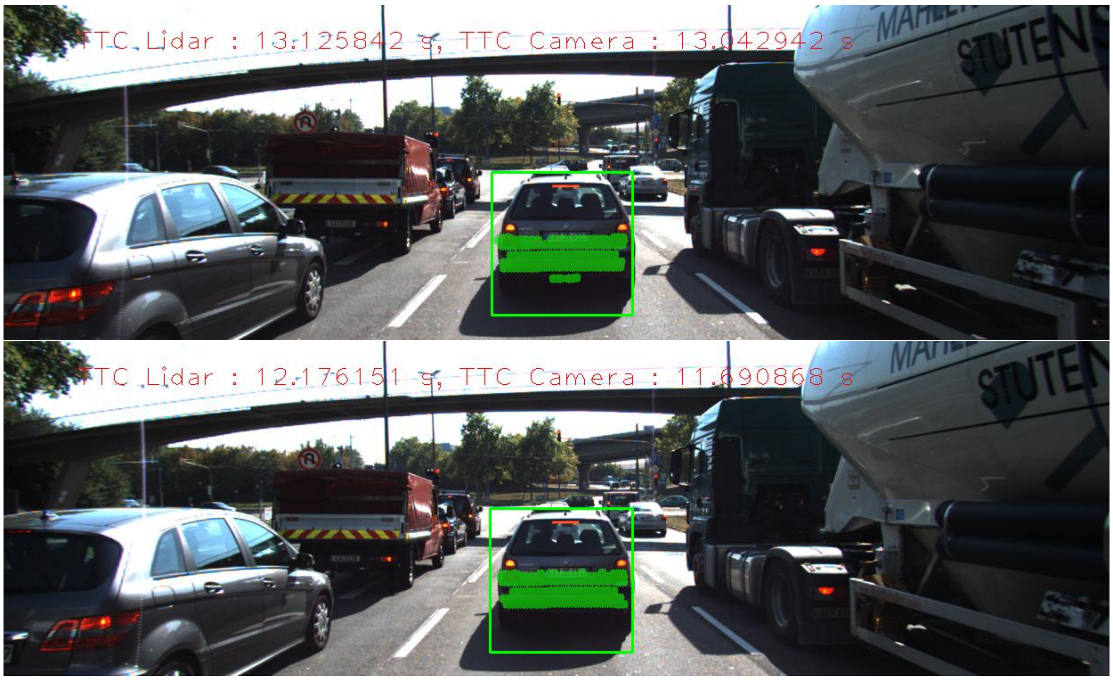




Top view of image id 7-8



Front view of image id 7-8


TTC Lidar : 13.125842 s, TTC Camera : 13.042942 s
TTC Lidar : 12.176151 s, TTC Camera : 11.690868 s

Reasoning behind this behaviour:

1. Assumption of constant velocity:

Even though the next car brakes, the ego car had some velocity. Here when the 2 lidar measurements were taken velocities of both cars were not constant, which impacts the distance between cars and hence has the impact on x_dist in lidar points.

2. Error in selecting x_value from lidar points:

I have used median technique to find suitable lidar point on next car to find distance. However mean or median are affected by distribution of data set hence some additional technique like percentile of points when combined with median filtering techniques can improve the results.

FP.6 Performance Evaluation 2

Problem Statement:

Run several detector / descriptor combinations and look at the differences in TTC estimation. Find out which methods perform best and also include several examples where camera-based TTC estimation is way off. As with Lidar, describe your observations again and also look into potential reasons.

Solution:

All the detector-descriptors were tested for TTC calculation. The detail stats of this can be found in cam_ttc_analysis.xlsx. Below are some observations made from the TTC values and plot of the TTC values.

The TTC values with camera are relatively unstable as compared to TTC values with LIDAR.

The error in TTC calculation with camera can be attributed to the keypoints matches on the road, or (quite literally) in the surrounding of the car. The image shown at the end supports this claim. So robust match filtering or even ROI filtering like mid-term project can
improve result.

Looking from the (just a maskeshaft) graph, almost all TTC values with SIFT are way off, many times Inf values are also produced.