FP.0 Final Report

FP.1 Match 3D Objects

The idea about this point was to relate the bounding boxes obtained by the DNN, and the keypoints matches that the detector/descriptor found. The implementation should be in the `matchBoundingBoxes` function.

To be successful about this point, the implementation follows the tips that Andreas nanodegree's teacher gave to us:

1. Find all the matches between the previous and the current frame, and make a loop of those.
2. Then, find on which bounding boxes keypoints are enclosed on both previous and current frames. Those will be the potential candidates, which should be added to a data structure, like multimap.
3. Finally, once completed the loop, find all the candidates on the multimap which shares the same pair and count them. The highest number of occurrences will be the pair <prev.boxId, curr.boxId> to fill the bbBestMatches.

A `multimap` data structure from the C++ STL has been chosen to find the potential candidates. To make sure a 1:1 relationship, the outer-most loop iterates all the current frame bounding boxes (ex: if to do so, we use `matches` instead, we would iterate more than once current box id, which results to an incorrect correspondences).

The potential candidates are the ones that previous frame and current frames bounding boxes, includes the keypoint match point.

Then, we should find the highest number of occurrences from all potential candidates before to assign the pair previous `boxID` and the current one, to make the relationship on `bbBestMatches`. To find the maxim number of occurrences, all previous values that have been filled to `bbBestMatches` are no longer considered.

FP.2 Compute Lidar-based TTC

This point asks to us to implement the `computeTTCLidar` function, and make sure that the calculation of the TTC is statistically robust.

The calculation of this TTC has been achieved by following the classes that explains about this topic, using the formula $TTC = \frac{d1 * dT}{\mathrm{d2} - \mathrm{d1}}$, where d1 is the minimum distance obtained through the current frame, d2 from the previous frame, and dT the time differential between both frames.

To make sure that this calculation is statistically robust, we decided to find the mean and the standard deviation, and discard all the outliers before obtain the minimum distance necessary for the formula for both frames.

FP.3 Associate Keypoint Correspondences with Bounding Boxes

Before calculate the Camera-based TTC, we has been asked to associate a given bounding box with the keypoints it contains.

To make sure that it is statistically robust, we have calculated, as Andreas pointed, all Euclidean distances between previous and current frames for all matches points that are enclosed inside the ROI of the bounding box.

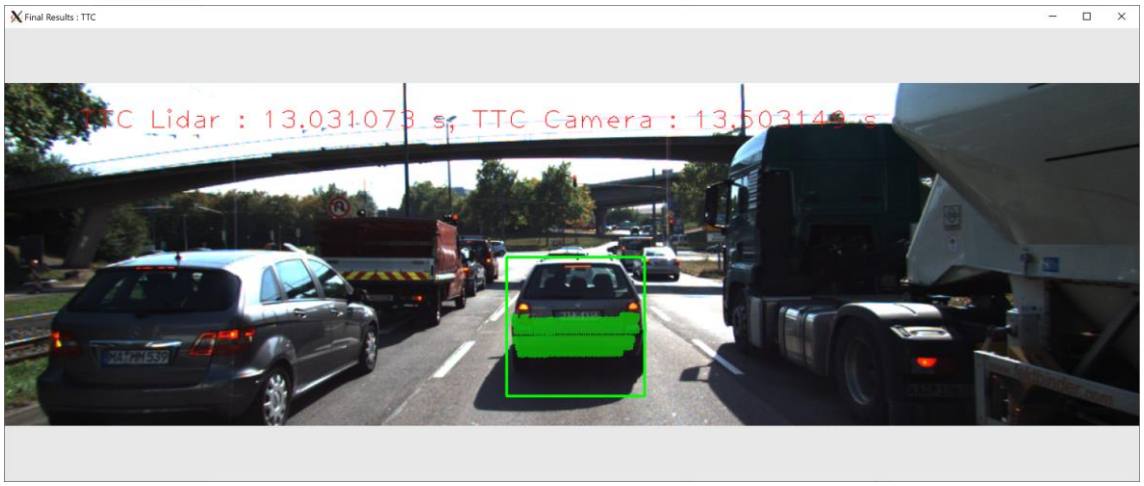
Once calculated all the Euclidean distances, we decided to took the median instead of the mean after check that with the median value, we got more reasonable values for a time value. We keep the calculations of mean in a comment section for future references.

Finally, for all matches, we find all the enclosed points for the current bounding box frame that are not so much far away from the median, and add the keypoint match to kptMatches field.

FP.4 Compute Camera-based TTC

Similarly of the Lidar one, the calculation of this TTC has been achieved by following the classes that explains about this topic, calculating the ratio distance between previous and current frame. To make it statistically robust, we used the median, sorting the all ratio distance that has been calculated and took the middle one. Once obtained the median value, we used the formula $TTC = \frac{-dT}{1-\mathrm{d}}$, where d is the median distance ratio, and dT the time differential between both frames.

Please, find enclosed an image showing the calculations of TTC for Lidar and Camera on the first frame (SHITHOMASI-BRISK):



FP.5 Performance Evaluation 1

Using SHITHOMASI as a detector, and BRISK as a descriptor, the following frames does not seem plausible for my perspective:

The reason that I don't find them descriptive for the behavior is because that TTC value has been increased "so much", without any so much difference in the environment, as it can be seen on the 3D points showing from a top perspective (`xmin` value, get from `show3DObjects` function).

On the following table the sequence between [Frame Number, xmin, TTC Lidar] can be shown:

Frame Number [#]	Xmin [m]	TTC Lidar [s]
1	7.91	13.031073
2	7.85	11.255764
3	**7.79**	**19.597519**
4	**7.68**	**17.319971**
5	7.64	11.890759
6	7.58	10.069708
7	7.55	14.617295
8	7.47	14.517295
9	7.43	14.143403
10	7.39	14.043403
11	7.20	10.532903
12	7.27	11.243067
13	7.19	8.397678
14	7.13	11.363455
15	7.04	8.128746
16	6.83	8.966659
17	6.90	11.180682
18	6.81	7.688745

The previous given examples correspond to the 3 and 4 frames numbering.

In addition, it can be seen that the Xmin has decreased from the previous frames, which we should wait for a less time per collusion value, which is not the case.

FP.6 Performance Evaluation 2

This requirement asks to do all descriptor / descriptor combinations possible, with the help of a spreadsheet and a graph, and make an evaluation with the TTC estimation. Please, see `spreadsheet.xlsx` file about this point for all data collection.

As it can be seen on the spreadsheet, some detector/descriptor have a nan or -inf results on TTC Camera-based. The explanation that I give to that is that, distance between matches of each current frames does not achieve the threshold defined (100) in `computeTTCCamera`.

AUTHOR: MARCEL VILALTA I SOLER