The solution with an explanation for each task are described bellow.

FP.1 : Match 3D Objects

The first step was to get all possible bounding box pairs (one from current image and the other from the previous image) and the respective number of matched keypoints inside both bounding boxes for each pair. Then, sort all the pairs by the counted number of keypoints matches. Finally, insert the pairs with highest number of keypoint matches to bbBestMatches without repeating already used bounding boxes along the choosed pairs. An extra step was done, were keypoint from matches that are out of the current or previous image bounding boxes was deleted from "matches" vector.

FP.2 : Compute Lidar-based TTC

Before compute the Lidar-based TTC, the Lidar points are filtered in filterLidarOutliers function. This function filters Lidar points with distance in x axis near to the median value (median-threshold <= filtered <= median+threshold). Then, the closest distance in the filtered Lidar points is found both in previous frame and the current frame, so the TCC is calculated.

FP.3 : Associate Keypoint Correspondences with Bounding Boxes

First, for each keypoint match inside the current bounding box (current bounding box is slightly shrinked to avoid having too many outlier points around the edges), insert the keypoint into empty vector and the match into a second empty vector, also compute the distance between matched keypoints position and insert this distance into third empty vector (named distance vector). In distance vector, the mean and standart deviation values are found for Z score computation. Zscore is found for each distance in the distance vector, and keypoints and matches with the respective distance Zscore less than a threshold value (1.0) are filtered. Finally, the keypoints inside the vector are associated to the respective Bouding Box.

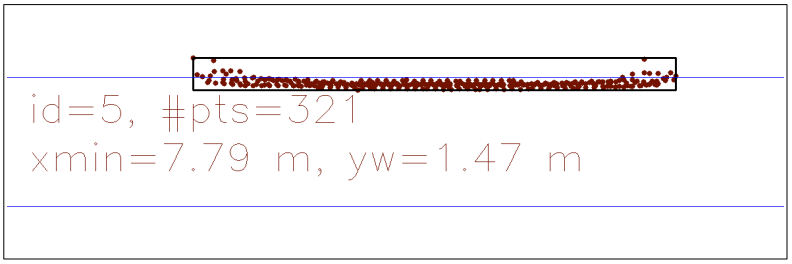
FP.4 : Compute Camera-based TTC

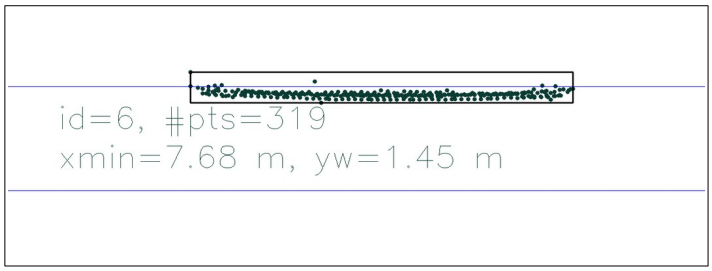
Distance ratios between all matched keypoints are computed and stored in a vector. If this vector is not empty, the median value is found and used for TTC camera-based calculation. The median value is used to avoid outliers.

FP.5 : Performance Evaluation 1

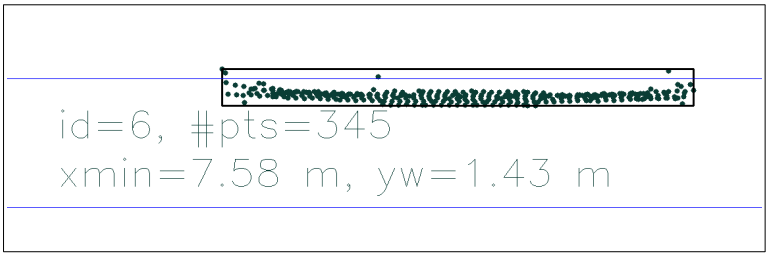
Comparing the estimated Lidar-based TTC values with the manually calculated ones based on the top view perpective, it gives an impression that some Lidar-based TTC estimations are way off (Frame 4, 7 and 10 from TCC LIDAR table). These wrong TTC values estimation are probably due to the filtering step, where some Lidar points are erroneously deleted from the point cloud, leading to a TTC miscalculation (distances are not the true minimum values in one or both current and previous frames). The top view image of each given example are shown in the figures bellow, where the current frame is in the right side and the previous frame in the left.

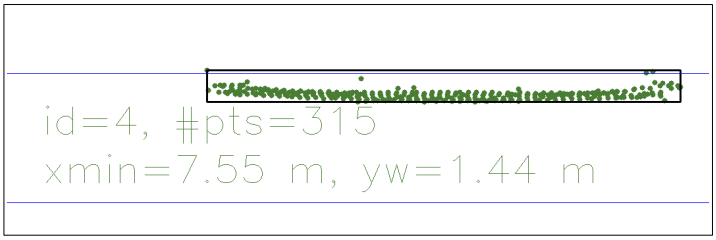
Frame 3 (left figure) and 4 (right figure):



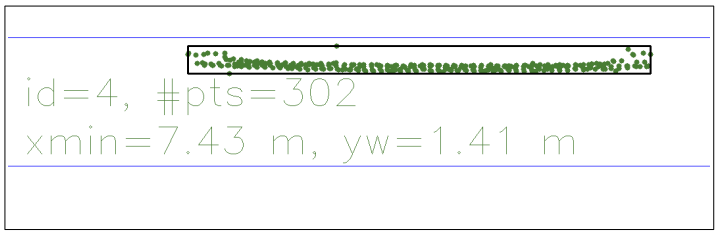


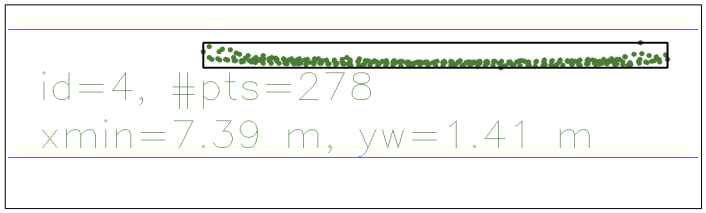
Frame 6 (left figure) and 7 (right figure):





Frame 9 (left figure) and 10 (right figure):





FP.6 : Performance Evaluation 2

Looking at the camera-based TTC estimations, many nan values are found, where the major use HARRIS or ORB detector. This happend lijely because of small number of keypoint detected along the frames. High or too low TTC camera based estimations appear among the results, such values are exemplified in frame 7 from ORB/FREAK set and frame 6 from HARRIS descriptor sets. It is probably because of mismatching keypoints, leading to wrong h ratio (h_1 / h_0) calculation and then wrong TTC estimation. According to MAE (Mean of Absolut Error) metric, where manually TTC estimated values were considered as the reference TTC values for each frame, FAST/FREAK set showed the best performance.