Final Report

FP.1 Match 3D objects

The match is implemented in matchBoundingBoxes in camFusion_Student.cpp. The algorithm counts the number of matches between each pair of bounding boxes in the previous and current frame, and considers previous box A and current box B a match if B has the most matches from A out of all the current matches, and also A has the most matches from B out of all the previous boxes. We use a stable two-way maximum to avoid situations like "a fragment of a larger image is detected as a separate object, e.g. an entire vehicle is matched to just its mirror"; in this dataset, nothing like that came up.

Note, this function also assigns enclosed keypoints and matches to the bounding boxes; consequently, clusterKptMatchesWithROI doesn't have to find the keypoints within each bounding box, it just has to filter out the outliers among them.

FP.2 Compute Lidar-based TTC

Implemented computeTTCLidar in camFusion_student.cpp. The algorithm uses the 10^{th} quantile of the distance to all the points as a robust estimate of the distance to the previous object. A lower quantile, or a different order statistic – e.g. the 3^{rd} closest point – may have been more appropriate, since there are typically only 1 or 2 outlier points per image, and over a hundred points are detected.

FP.3 Associate keypoint correspondences with bounding boxes

Note, a first pass at assigning keypoints and matches to bounding boxes is implemented in FP.1, when we have to go through all the keypoint matches and their bounding boxes anyway! Here we filter out correspondences that are likely to be spurious. The implementation can be found in clusterKptMatchesWithROI in camFusion_Student.cpp.

The median of the distances by which matched keypoints within each box move from one image to the next is computed, and any match where the computed distance exceeds 1.5x of the median is considered invalid. A more statistically robust approach might pay attention to the vector displacement (not just the scalar distance), and would also use the standard deviation of the displacements rather than a constant 1.5 * median to figure out which pairs are outliers.

FP.4 Compute camera-based TTC

Implemented computeTTCCamera in camFusion_student.cpp. The median distance between keypoints in each pair is used to estimate the scale factor between the image from one frame to the next, which is then plugged into the formula from Lesson 3.3 for the constant velocity model (with one exception, instead of negative times when the image apparently shrank, we report large positive times).

In this part of the final project as well as in FP.3, the median is used because it is more robust to outliers than the mean; however, it is more time-consuming to compute, since it requires sorting the array of distances. In FP.2, the entire array is also sorted; again, there are slightly faster ways of running the computation that do not require sorting the entire array.

FP.5 Performance evaluation 1

See attached document FP5.doc and the data (Sheet 1) and plot (Sheet 2) in FP5.xlsx.

PF.6 Performance evaluation 2

See attached document FP6.doc and the data (Sheet 1) and plots (Sheet 2) in FP6.xlsx.