FP.1 Match 3D Objects:

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. Matches must be the ones with the highest number of keypoint correspondences.

Approach: To find the best matches bounding boxes between the current frame and the previous frame, first we count the number of matched keypoints for each pair of bounding boxes and store them in vectors. Then, we can start picking up associated bounding boxes starting from the pair with the highest number of keypoints matches.

Implementation is in matchBoundingBoxes function (camFusion_Student.cpp)

FP.2 Compute Lidar-based TTC:

I used 20% of the total number of Lidar points to calculate the averaging closest distance to the preceding vehicle. This would be more robust than simply using the closest Lidar point.

Implementation is in computeTTCLidar function (camFusion_Student.cpp)

FP.3 Associate Keypoint Correspondences with Bounding Boxes:

First we iteratively go through each keypoint match and check whether the corresponding keypoints are in the region of the bounding box. Next, to filter out outliers, we calculate the mean of feature distance among all matched keypoints and only consider those matched keypoints whose feature distance are less than the mean. Finally, we save the qualified keypoint pairs into the bounding box.

Implementation is in clusterKptMatchesWithROI function (camFusion_Student.cpp)

FP.4 Compute Camera-based TTC:

using the knowledge learned in the Lecture 2-2 Collision Detection Systems

we calculate distance ratios between matched keypoints obtained from task 3. Then we pick up the median of all distance ratios for more accurate camera TTC estimation.

Implementation is in computeTTCCamera function (camFusion_Student.cpp)

FP.5 Performance Evaluation 1:

As mentioned in FP.3 part, if we only search the closest point from Lidar point cloud, erroneous TTC estimations may occur. Outliers measurements for Lidar points may be included in the bounding box.

FP.6 Performance Evaluation 2:

For Lidar-based TTC(s):

image Index	TTC LiDAR (in seconds))	Min x coordinate (in m)
1	12.97	7.91
2	12.26	7.85
3	13.91	7.79
4	6.51	7.68
5	7.41	7.64
6	7.50	7.58
7	34.34	7.55
8	9.34	7.47
9	9.18	7.43
10	18.03	7.39
11	3.83	7.20
12	10.12	7.27
13	9.22	7.19
14	10.97	7.13
15	8.09	7.04
16	3.18	6.83
17	10.29	6.90
18	8.31	6.81

For Camera-based TTC(s):

Mean Difference in TTC Estimate (s):

Detector\Descriptor	AKAZE	BRIEF	BRISK	FREAK	ORB	SIFT
AKAZE	1.53	1.23	1.09	1.22	1.27	1.12
BRISK	Nan	3.08	3.33	3.05	3.59	3.02
FAST	Nan	2.03	2.17	2.17	1.87	1.99
SHITOMASI	Nan	1.78	1.49	1.59	1.38	1.43
SIFT	Nan	1.56	1.68	1.38	Nan	1.40

The best three detector/descriptor combinations are:

1 FAST + ORB

2 FAST + BRIEF

3 SHITOMASI + BRIEF

The best three detector/descriptor combinations FAST/BRISK, FAST/ORB, SHITOMASI/ BRIEF, while SIFT/SIFT got the smaller TTC difference comparing to other methods, whereas HARRIS/BRIEF and ORB/ORB got the worst performance. The reason might be these two combinations got very few matched keypoints in the bounding box, resulting in unreliable Camera TTC estimation.