Rubric	How I addressed it
FP.1 Match 3D Objects	Implemented in matchBoundingBoxes within camFusion_Student.cpp
FP.2 Compute Lidar-based TTC	Implemented in computeTTCLidar within camFusion_Student.cpp. Used the Median Point of the point cloud cluster of the preceding vehicle (for both current and previous frames), rather than the closest point, to remove errors introduced by outliers located in between egocar and point cloud cluster.
FP.3 Associate Keypoint Correspondences with Bounding Boxes	Implemented in clusterKptMatchesWithROI within camFusion_Student.cpp. I took the mean of all the euclidean distances between keypoint matches and then removed those that are too far away from the mean.
FP.4 Compute Camera-based TTC	Implemented in clusterKptMatchesWithROI ans computeTTCCamera within camFusion_Student.cpp. I took the median distance ratio to compute TTC, to remove errors introduced by outliers.
FP.5 Performance Evaluation 1	**Errors observed in Lidar TTC computation:** 1) From Image 1 below, I noticed an increase in TTC computed from around 14.1s to 16.7s. This is because I used the median point (in x axis) in the point cloud cluster. I noticed an increase in the size of the bounding box, which implies that the points are more spreaded out within the box. Therefore, extracting the median point would lead to greater TTC computation error as the median point deviates from the true exterior of the preceding vehicle. 2) From Image 2 below, I noticed an increase in TTC computed from around 12.6s to 14s. I also noticed that there are more points that are spreaded out at the extreme ends of the point cloud cluster. These ends represent the curve edges of the body of the preceding vehicle. These points led to a median point that has a greater x value, causing the TTC calculation to be overestimated.
FP.6 Performance Evaluation 2	Kindly refer to the spreadsheet “Final Project Spreadsheet” **Best Detector + Descriptor Combo - Camera TTC (Green):** a. In general, the Shi-Tomasi Detector fared the best because they have the lowest standard deviation across the TTC calculated for these 19 images. This implies that there are less erratic jumps in values as the camera approaches closer to the proceeding vehicle. This trend can also be seen from the values in the spreadsheet for each of these images. b. In particular, the Shi-Tomasi Detector with the SIFT Descriptor yielded the lowest standard deviation. c. Another Detector that fared quite well on average is AKAZE.

	Errors / Abnormal results - Camera TTC (Yellow highlight): a. **Harris Detector** yielded large negative TTCs, sometimes large positive results. This is expected because the Harris Detector is only capable of identifying much fewer keypoints compared to the other detectors, leading to limited distance ratio calculations and hence inaccurate calculations. **(Img 3)** b. **Orb Detector** yielded several large positive TTCs, as well as negative infinity values. Likewise, the Orb Detector yields few keypoints. In fact, few keypoints actually land on the back of the egocar. **(Image 4)** **LiDAR TTC:** a. The LiDAR TTC values are observed to be constant regardless of the detector-descriptor combination. In the Lidar TTC computation process, the only part where detectors and descriptors were used were when we were trying to match bounding boxes ID between different images. For that purpose, all detector-descriptor combinations were accurate in matching the right bounding boxes IDs. Therefore, the results were uniform throughout.

Image 1

Image 2

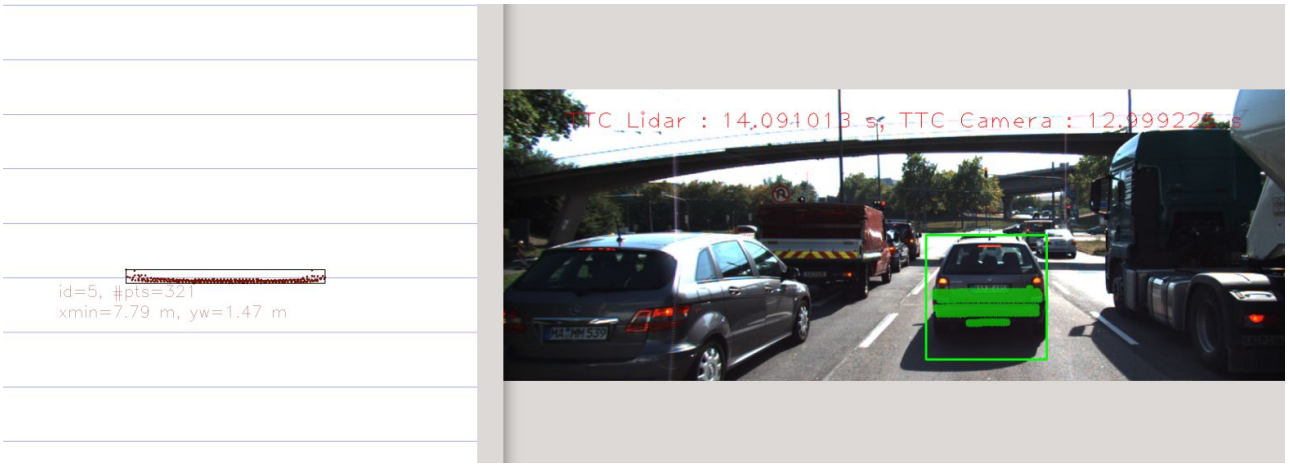

Image 3: Harris Keypoint Detector

Image 4: ORB Keypoint Detector