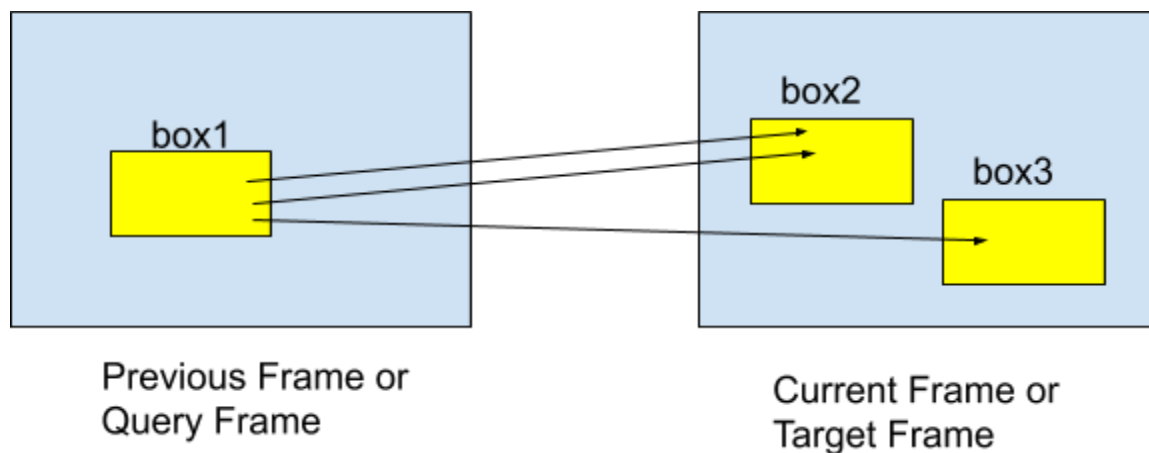


Question: Implement the method "matchBoundingBoxes", which takes as input both the previous and the current data frames and provides as output the ids of the matched regions of interest (i.e. the boxID property). Matches must be the ones with the highest number of keypoint correspondences.

The matchBoundingBoxes is implemented by looking at each of the keypoint matches between the two frames. In the previous frame, for each bounding box, we find the number of keypoint matches that fall within its bounding box and also find the bounding box in the target frame that has the maximum number of keypoint target matches. See illustration below: here the target frame has 3 keypoint matches out of which box2 has 2 matches and box3 has only 1 match. Based on this analysis we can conclude that box1 in the previous frame (query frame) corresponds to the box2 in the current frame (target frame).



This code is implemented by keeping a counter and map. The map key consists of the bounding box ids in the previous frame. The map value consists of the bounding box counter for each of the target keypoints that falls within the target bounding box. After processing all the keypoint matches, we find the bounding box that has the maximum value of the counter (maximum number of target keypoint matches). The result is saved in the map *bbBestMatches* where the key is the bounding box id in the previous query frame and the value is the best matching bounding box id in the target or current frame.

Question: Compute the time-to-collision in second for all matched 3D objects using only Lidar measurements from the matched bounding boxes between current and previous frame.

For time-to-collision using lidar, we compute the x distance to all the obstacles that are in the ego lane of the car. Using a constant velocity model, the distance ratio is computed for the previous frame and the current frame to the nearest obstacle. Lidar sensors are known to generate a few noisy readings that can result in wrong measurements. To remove these outliers, we compute the median of the distance ratio and use the median measurement in the TTC calculation.

$$\text{TTC}_{\text{lidar}} = 1 / (\text{frame_rate} * (\text{mean_distance_previous} / \text{mean_distance_current} - 1))$$

Question: Prepare the TTC computation based on camera measurements by associating keypoint correspondences to the bounding boxes which enclose them. All matches which satisfy this condition must be added to a vector in the respective bounding box.


Before computing the TTC using the camera keypoints, we need to limit the keypoint matches to mainly fall within the bounding box that corresponds to the car that is being tracked in the previous and the current frame. Another important step that is required here is to remove key point matches that are outliers which can corrupt the quality of keypoints used for TTC measurement. For this we compute the “Euclidean” distance for each of the current keypoint and the matching keypoint in the previous frame. For the same object that is being tracked, the median of this euclidean distance is used for outlier rejection. We reject all the keypoints whose euclidean distances are lower than 0.5 or higher than 1.5 of the measured median euclidean distances. This outlier rejection approach gives much better control of the estimates. Another outlier rejection approach that also gives good results is by having a fixed threshold of euclidean distance between matching keypoints should be less than 100.0

Question: Compute the time-to-collision in second for all matched 3D objects using only keypoint correspondences from the matched bounding boxes between current and previous frame


For computing the TTC using keypoint correspondences, we find all the keypoint matches for the given bounding box being tracked. We take two of these key point matches namely (kp1_target, kp1_query), (kp2_target, kp2_query), we compute distance ratio using the formula $\text{Distance_ratio} = \text{norm}(\text{kp1_target} - \text{kp2_target}) / \text{norm}(\text{kp1_query} - \text{kp2_query})$. This distance ratio is measured for all the possible keypoint matches and the median of the distance ratio is used to measure the TTC equation.

Question: Find examples where the TTC estimate of the Lidar sensor does not seem plausible. Describe your observations and provide a sound argumentation why you think this happened.

For TTC measurement using lidar data three different techniques could be used namely mean, min, and median. Among these the mean value of lidar distance estimate can cause under reporting and lead to loose TTC value which is dangerous. On the contrary, using the minimum lidar distance can give the most accurate estimate when no noise is present in the lidar reading. Unfortunately, lidar sensor data is prone to noisy measurement leading to stray pixels that can cause very conservative TTC estimation. For example two instances are shown in the figure below:



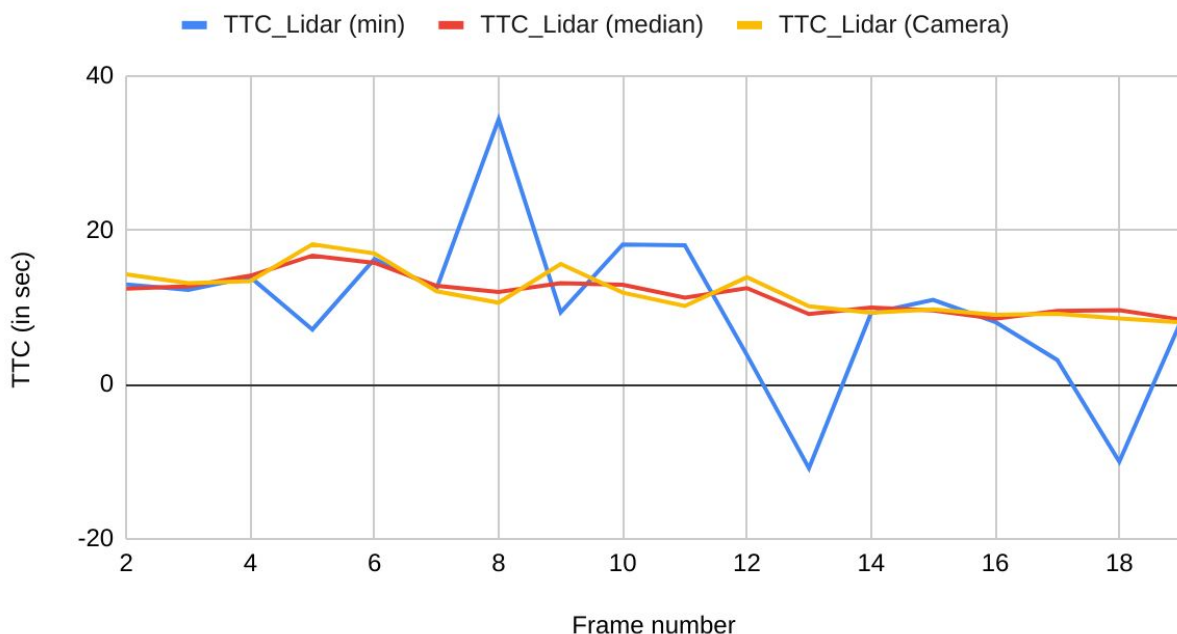
#pts=288
=6.83 m, yw=



#pts=300
=7.20 m, yw=1

In both the above instances the noisy lidar measurement causes TTC_min to have negative values and widely fluctuating calculations. To know the variation in the TTC estimate see figure below where we plot the TTC measured using the minimum lidar distance metric and the median lidar distance metric for frames 2 to frame 19. As we can see in the figure the TTC measurement using the minimum lidar distance estimate can cause large variance in its estimation making the process harder. On the contrary, the median distance approach generates much cleaner and smoother TTC estimates.

TTC using lidar and camera data



Question: Run several detector / descriptor combinations and look at the differences in TTC estimation. Find out which methods perform best and also include several examples where camera-based TTC estimation is way off. As with Lidar, describe your observations again and also look into potential reasons.

For evaluating the TTC performance using camera based technique, we compared the TTC estimate from the camera and the TTC estimate from Lidar (using median approach). At the end we can also

compute the average of percentage TTC estimation difference between camera and lidar. This gives us a good estimation of the quality of the TTC camera based approach for different detectors and descriptors. The smaller the average percentage error the better the TTC camera detector/descriptor for tracking.

Frame number	TTC_Lidar Min approach (in seconds)	TTC_Lidar median approach (In seconds)	TTC_Camera Median approach (in seconds)
			SIFT + SIFT
2	12.9722	12.4178	11.3744
3	12.264	12.715	12.1803
4	13.9161	14.0909	12.6763
5	7.11574	16.6894	15.9213
6	16.251	15.7467	12.8281
7	12.4213	12.7834	11.4816
8	34.3401	11.9844	13.1338
9	9.34372	13.1242	13.6393
10	18.1317	12.9121	12.7795
11	18.0319	11.2587	10.5665
12	3.83244	12.4832	11.1002
13	-10.8538	9.12678	12.472
14	9.22312	9.96437	9.44634
15	10.9678	9.59861	10.4023
16	8.09422	8.52154	9.07095
17	3.17535	9.51553	8.80611
18	-9.99423	9.61236	8.51119
19	8.30975	8.39877	8.76483
Mean Error	5.941896111		1.06925

Below you can find the summary table of the average error between the lidar and camera based TTC measurement. Based on the analysis it was found that the following two combination worked best for TTC camera based estimation approach

1. SIFT keypoint detector and SIFT keypoint descriptor
2. SIFT keypoint detector and FREAK keypoint descriptor
3. FAST keypoint detector and AKAZE keypoint descriptor

Approach 1: No constraints (just the median)					
Average TTC Error between LIDAR and Camera based in seconds					
	BRIEF	ORB	FREAK	AKAZE	SIFT
HARRIS	8.89395	12.0969	6.76012		7.12849
shi-tomashi	4.40779	1.90711	nan	1.90711	2.00309
BRISK	8.62016	nan	2.54232	22.2795	4.69008
FAST	1.70768	11.2213	5.39954	21.2478	36.5877
AKAZE	5.99929	1.9925	4.06373	4.5895	4.75446
ORB	16.346	nan	nan	11.5653	31.9895
SIFT	0.941531	out of memory	3.2162	1.95896	0.772973
Approach 2: TTC should be positive before computing median					
Average TTC Error between LIDAR and Camera based in seconds					
	BRIEF	ORB	FREAK	AKAZE	SIFT
HARRIS	1.76337	1.71958	1.55097	1.93438	1.78883
shi-tomashi	1.57377	1.50999	1.76963	1.8525	1.90701
BRISK	1.81389	1.65372	1.5472	6.28836	3.63124
FAST	2.50102	1.92912	1.80691	2.04468	1.81344
AKAZE	1.50628	1.3382	0.930441	1.09351	1.90267
ORB	2.27469	4.95956	10.7884	6.16492	4.64146
SIFT	1.30055	out of memory	0.955039	1.11614	1.06925

Looking at the debug data from camera based TTC approach several outlier cases were found that caused very poor measurement.

1. Any poor keypoint match will impact the overall measured distance ratio significantly. See the figure below for two keypoint matches using SIFT where the matches are particularly poor. These poor keypoint matches impact the overall distance ratio and the TTC calculation significantly.



2. Second issue is sometimes the bounding box of the vehicle could include the nearby obstacle that is far away but still lies within the out bounding box. Similar to the overlapped bounding box issue in lidar a shrinking factor can be incorporated to limit the points within the bounding box. This conservative approach can be potentially helpful if sufficient keypoints are visible within the box and not just on the outer edges of the bounding box.