SFND_3D_Object_Tracking writeup

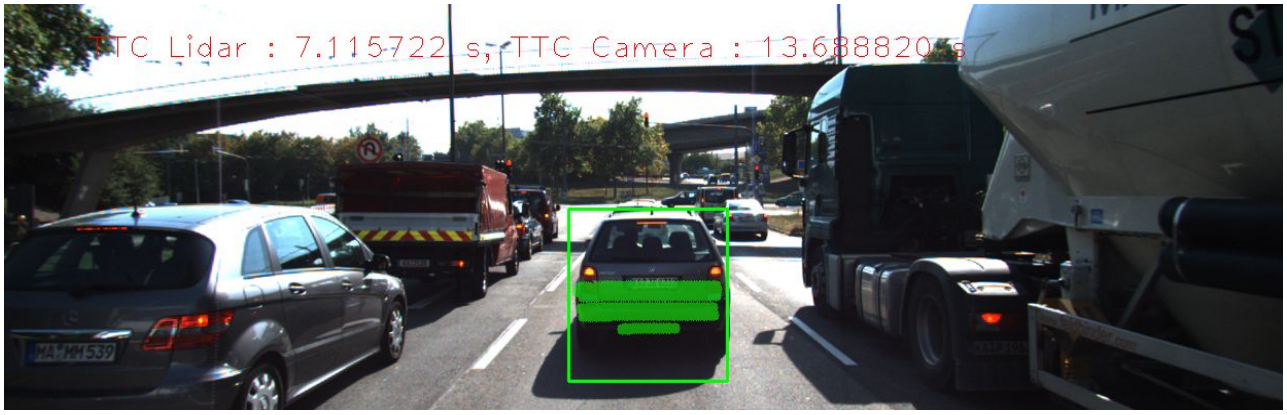
FP.5 Performance Evaluation 1

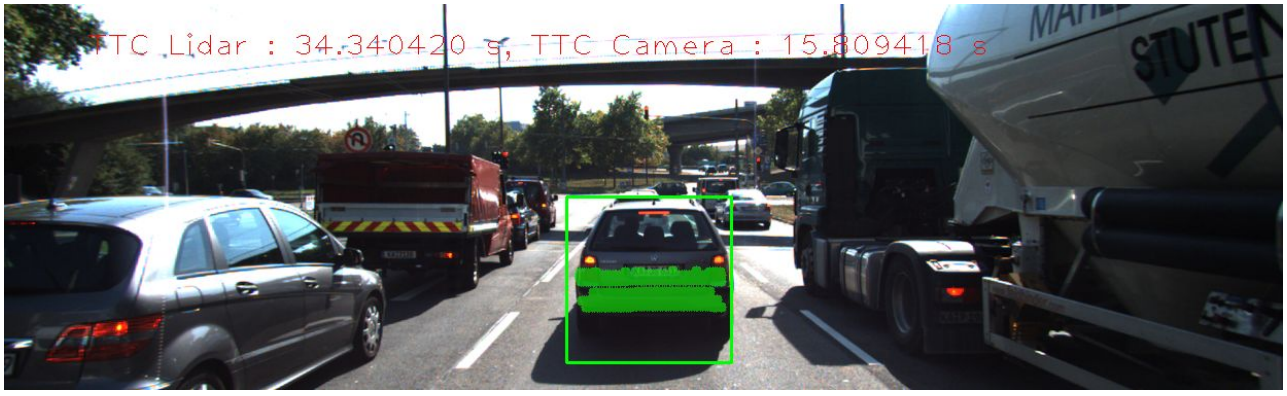
Examples where the TTC estimate of the Lidar sensor does not seem plausible (Observations and argumentation why this happened):

Lidar TTC estimation over the 18 images:

- With TTC calculation using the closest point :

  Some frames are not plausible such as the frame 4 (only 7 secs) or frame 7 (34 secs).



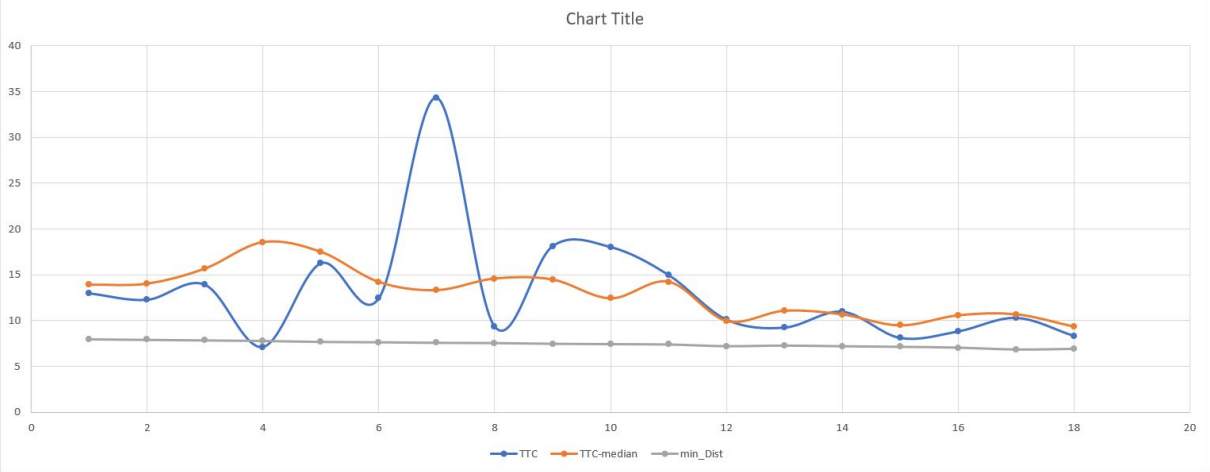


- With TTC calculation using the median point :

As the graph shows , Calculating TTC using median of points is much better than calculating using the closest point . The values are more stable and plausible .

A graph shows TTC using closest point vs TTC using median point vs closest point distance over the 18 images.

Frame	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
TTC	12.9722	12.264	13.9161	7.11572	16.2511	12.4213	34.3404	9.34376	18.1318	18.0318	14.9877	10.1	9.22307	10.9678	8.09422	8.81392	10.2926	8.30978
TTC-median	13.9062	14.0158	15.6567	18.5438	17.4962	14.2038	13.3159	14.5824	14.4712	12.4163	14.2318	9.95531	11.0715	10.6651	9.46841	10.5728	10.6804	9.332
min_Dist	7.97	7.91	7.85	7.79	7.68	7.64	7.58	7.55	7.47	7.43	7.39	7.2	7.27	7.19	7.13	7.04	6.83	6.9



FP.6 Performance Evaluation 2

All detector / descriptor combinations implemented in previous chapters have been compared with regard to the TTC estimate on a frame-by-frame basis.

Examples where camera-based TTC estimation is way off:

1-Camera estimation of TTC is -inf: when the median of camera krypoint distance ratios = 0 . According to the software in the next image , TTC is -inf happens when medDistRation is 1 .

```
// STUDENT TASK (replacement for meanDistRatio)
std::sort(distRatios.begin(), distRatios.end());
long medIndex = floor(distRatios.size() / 2.0);
double medDistRatio = distRatios.size() % 2 == 0 ? (distRatios[medIndex - 1] + distRatios[medIndex]) / 2.0 : distRatios[medIndex];
utlier influence

double dT = 1 / frameRate;
TTC = -dT / (1 - medDistRatio);
```

medDistRation is 1 when the detector sees that many points getting closer (distCurr/distPrev<1) and in the same frame many points getting far (distCurr/distPrev>1) .

```
if (distPrev > std::numeric_limits<double>::epsilon() && distCurr >= minDist)
{ // avoid division by zero

    double distRatio = distCurr / distPrev;
    distRatios.push_back(distRatio);
}
} // eof inner loop over all matched kpts
```

I think this issue may be for 2 reasons:

- The yolo bounding box isn't very accurate or consistent all the time . It sometimes include other vehicles or road .
- The keypoints extractor- descriptor extractor - matcher combination quality isn't very good.

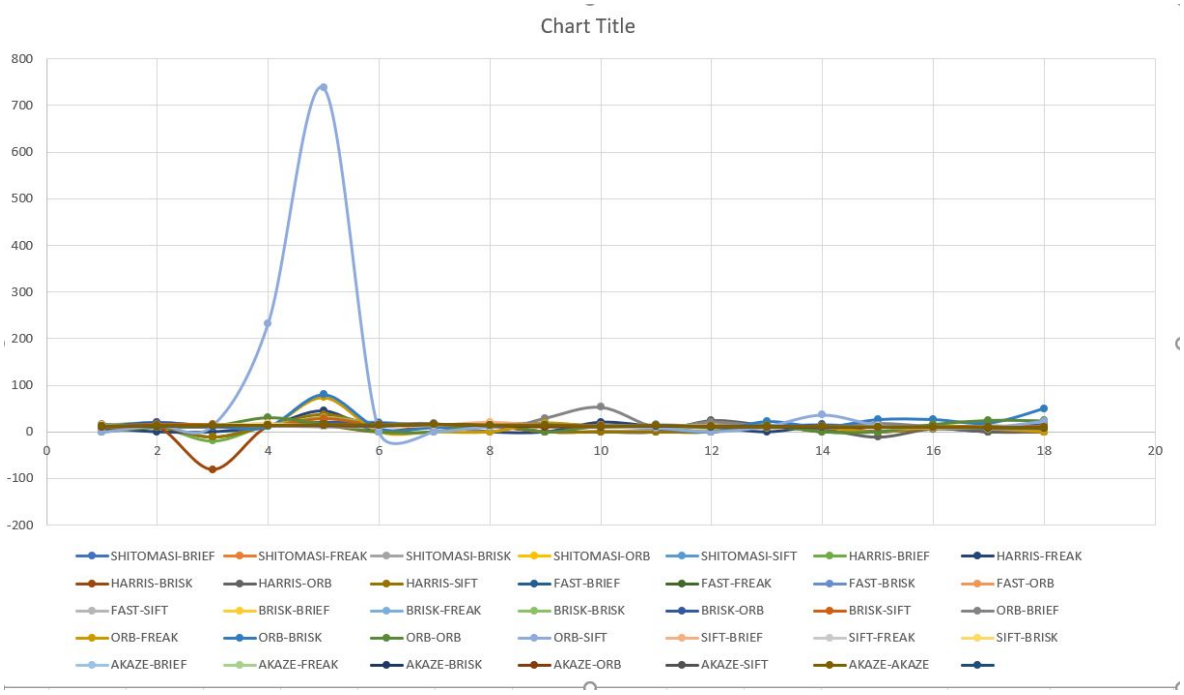
2-The number of matches is 0 in some combinations , that produce runtime error output as there are no distances-ratios to calculate .I output TTC in this case as a NAN . One example is image #3 in HARRIS-FREAK combination.

These are the 3 frames : 2 , 3 ,4 at HARRIS-FREAK combination :

No. of matches inside the box is low in general and is zero in the 3rd frame.

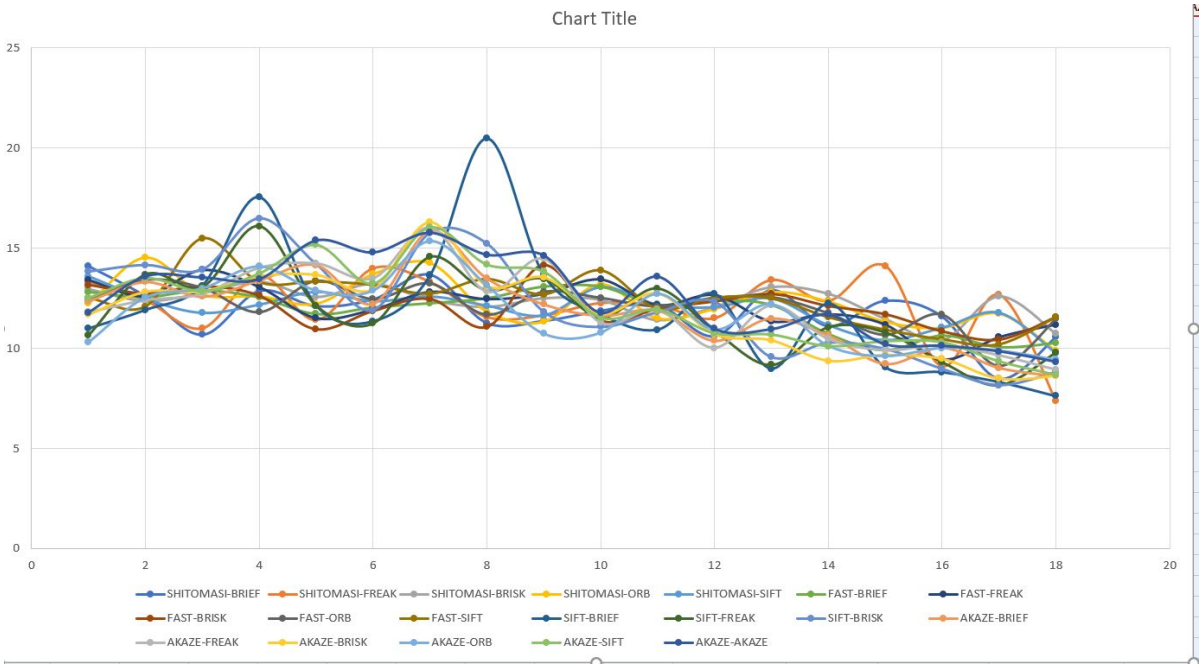
Comparison of combinations

Using the spreadsheet of all TTC estimation of all combinations of detector/ descriptor types: This is the graph of all TTCs over the 18 images.

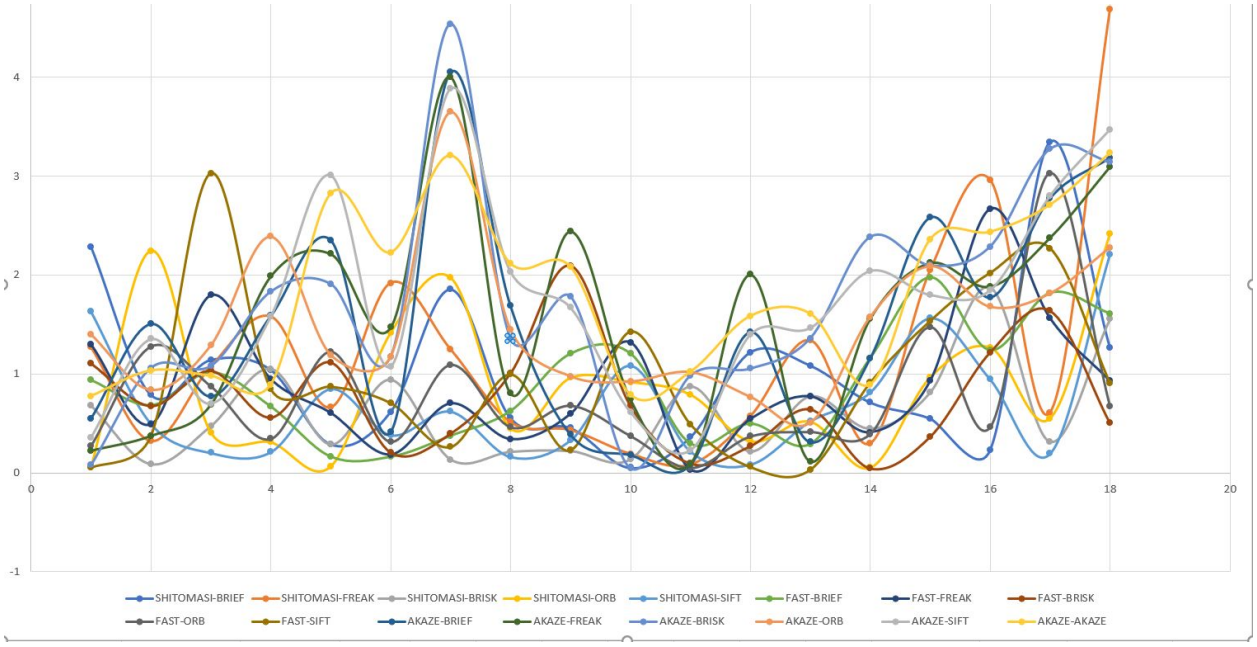


Found that some combinations have infinite values or NAN values in more than 1 image . Also some combinations have very large TTC that isn't consistent with LIDAR readings or other combinations readings.

After neglecting them :



Then I made a chart of the difference between TTC of every frame and average of all frames in every combination .



From the average / min / max data , I concluded that the best 4 combinations(detector-descriptor) are:

- **FAST-BRISK , FAST-BRIEF , SHI TOMASI-BRISK ,SHI TOMASI-SIFT**