# Final report: Camera Based 2D Feature Tracking

Gia Minh Hoang

## F.P1 Match 3D Objects

Lines 317 – 344 in `camFusion_Student.cpp` implements method "matchBoundingBoxes". It iterates through all the keypoint matches as outer loop. Each keypoint match is checked whether it is enclosed in a bounding box for current and previous frames. Helper function `isEnclosedInBoundingBox` (lines 296 – 315) uses `cv::Rect.contains(cv::Point)` to find the bounding box. Moreover, if there is more than 1 bounding boxes, the helper function returns false to avoid mismatching.

A variable `match2DMatrix` whose each row represents the index of bounding box in current frame and each column shows that in previous frame counts the number of number of keypoint correspondences in order to find the bounding matches.

## FP.2 Compute Lidar-based TTC

Lines 248 – 294 in `camFusion_Student.cpp` computes lidar-based TTC. As lidar cannot measure the velocity directly, it can only be done by differencing 2 successive measurements. For each measurement i.e., point cloud, the closest point in x-axis (forward direction) is found. It is compared with the mean value in x-axis of all the points scaled by a factor in order to make it robust to possible outliers. Then the TTC is calculated as

$$TTC_{lidar} = \frac{1}{frameRate}\frac{minXCurr}{minXPrev - minXCurr}, \tag{1}$$

where $minXCurr$ and $minXPrev$ are the closest points in x-axis in current and previous frame respectively. Note that the preceding vehicle is running at higher speed than the ego vehicle, $minXPrev < minXCurr$ causes negative TTC. Thus, the function returns `TCC = NAN` and signals "No potential collision, preceding vehicle is running at higher speed!".

## FP.3 Associate Keypoint Correspondences with Bounding Boxes

Lines 133 – 194 in `camFusion_Student.cpp` adds the keypoint correspondings to the bounding box. It simply loops over the keypoint matches and uses `cv::Rect.contains(cv::Point)` similar to task FP.1. To make it robust to the outliers, I have tested 3 methods as follows:

1. Compute a mean of all the euclidean distances between keypoint matches and then remove those that are too far away from the mean.
2. Compute a median of all the euclidean distances between keypoint matches and then remove those that are too far away from the median.
3. Compute a mean and a standard deviation ($1\sigma$) of all the euclidean distances between keypoint matches and then remove those that are far away more than $3\sigma$ from the mean.

I found that method 3 did not work well as some outliers made the standard deviation so large. Methods 1 and 2 worked better. Though median is slightly better than mean, I found median method did not work for the case HARRIS/BRISK due to the fact that HARRIS detects only the few points and the median are rejected all the points.

## FP.4 Compute Camera-based TTC

Lines 197 – 245 in `camFusion_Student.cpp` computes camera-based TTC. As monocular cameras are not able to measure metric distances, TTC is computed by observing relative height change (also called scale change) directly in the image. Identifiable keypoints are detected and tracked from one frame to the next, the distance between all keypoints on the vehicle relative to each other to compute a robust estimate of the height ratio in out TTC equation as

$$TTC_{camera} = \frac{-1}{frameRate} \frac{1}{1 - medDistRatio}, \tag{2}$$

where $medDistRatio$ is the median of all the relative distance ratios computed from keypoint matches in 2 successive images to remove the outliers. The $medDistRatio$ is calculated by iterating through all points in the `vector<cv::DMatch> kptMatches`, and each of these points to all other points in the same vector using an inner for loop.

## FP.5 Performance Evaluation 1

The lidar-based TTC is logged using all possible combinations (of detectors and descriptors). Since AKAZE descriptor only works with AKAZE detector and descriptor ORB cannot function with detector SIFT, we have 35 combinations.

The results is collected in the results.csv in directory `/SFND_3D_Feature_Tracking/report/`

The keypoint matches provided by all combinations (of detectors and descriptors) are able to find the bounding box matches. Thus, the lidar-based TTC estimates are the same regardless of the combinations. Let take a look at an example with SHITOMASI/BRISK as follows:

*Table 1: Results syntheses for SHITOMASI/BRISK.*

| Detector | Descriptor | Image id | No matched keypoints in ROI | TTC lidar | TTC camera |
|---|---|---|---|---|---|
| SHITOMASI | BRISK | 0 | 0 | nan | nan |
| SHITOMASI | BRISK | 1 | 59 | 12.9722 | 14.1119 |
| SHITOMASI | BRISK | 2 | 59 | 12.264 | 12.9876 |
| SHITOMASI | BRISK | 3 | 58 | 13.9161 | 13.4904 |
| SHITOMASI | BRISK | 4 | 63 | 7.11572 | 12.398 |
| SHITOMASI | BRISK | 5 | 63 | 16.2511 | 12.6754 |
| SHITOMASI | BRISK | 6 | 61 | 12.4213 | 14.5029 |
| SHITOMASI | BRISK | 7 | 64 | 34.3404 | 12.9117 |
| SHITOMASI | BRISK | 8 | 58 | 9.34376 | 15.5997 |
| SHITOMASI | BRISK | 9 | 66 | 18.1318 | 11.5126 |
| SHITOMASI | BRISK | 10 | 57 | 18.0318 | 14.685 |
| SHITOMASI | BRISK | 11 | 60 | 3.83244 | 11.3024 |
| SHITOMASI | BRISK | 12 | 78 | nan | 11.6524 |
| SHITOMASI | BRISK | 13 | 75 | 9.22307 | 11.7243 |
| SHITOMASI | BRISK | 14 | 75 | 10.9678 | 11.5069 |

| | | | | | |
|---|---|---|---|---|---|
| SHITOMASI | BRISK | 15 | 64 | 8.09422 | 9.33785 |
| SHITOMASI | BRISK | 16 | 63 | 3.17535 | 11.3079 |
| SHITOMASI | BRISK | 17 | 66 | nan | 11.4487 |
| SHITOMASI | BRISK | 18 | 71 | 8.30978 | 9.12792 |

`TTC = NAN` for the first image is normal because 2 lidar measurements are needed to compute TTC.

Let consider the second NAN for image 12, the reason is that the preceding vehicle has higher speed than the ego vehicle leading to increased relative distance between them as explained in task FP.2. The top view perspective of the Lidar points in the following figure confirms this observation.

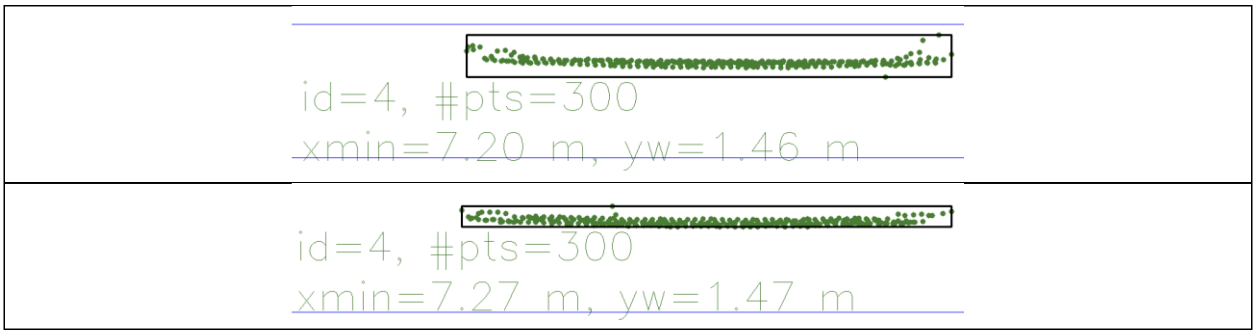



*Figure 1: Top view perspective of the Lidar points for frame 11 (top) and frame 12 (bottom). Note that the relative distance increases.*

Similarly, TTC = NAN for image 17 has the same reason as illustrated by the figure below.

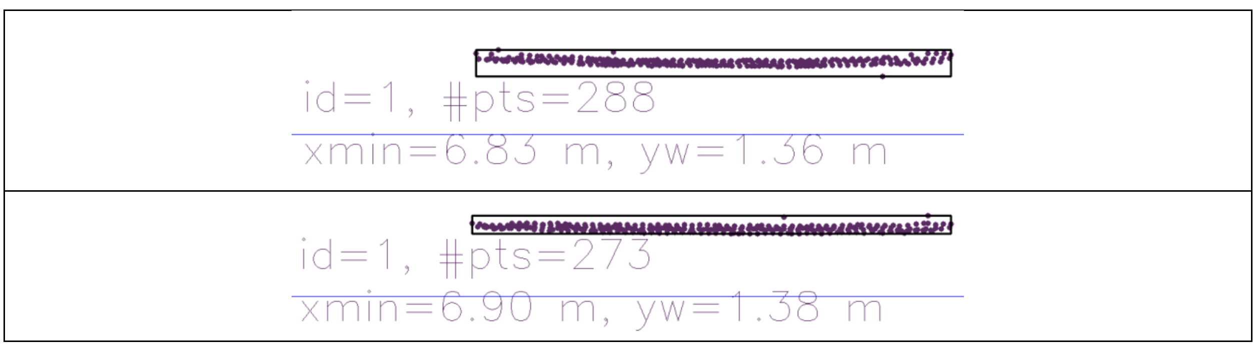
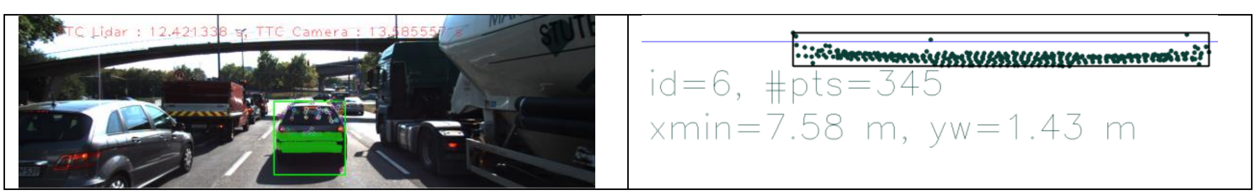



*Figure 2: Top view perspective of the Lidar points for frame 16 (top) and frame 17 (bottom). Note that the relative distance increases.*

Finally, let try to understand why the TCC jumped to rather high value of 34.34 s in image 7. The front views with displayed bounding box, lidar points, and keypoints as well as top view perspective of the lidar points are depicted in the following figure.



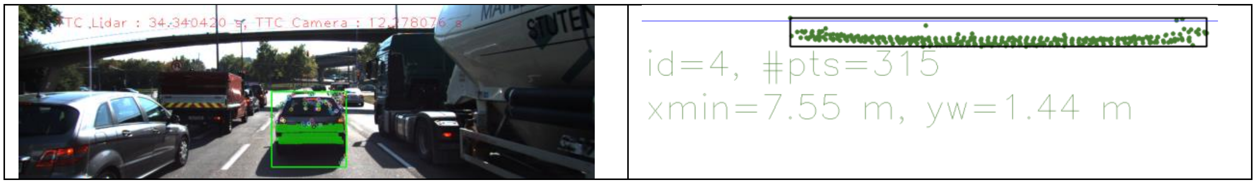



*Figure 3: Front views displayed bounding box, lidar points, and keypoints for image 6 (top left) and 7 (bottom left) while top view perspective of the lidar points for image 6 (top right) and 7 (bottom right).*

Based on the top view of lidar point, using equation (1) lidar-based TTC = 25.17 s leading to an error of 36%. The possible reason could be some lidar points are unstable to use in TTC computation especially low speed scenario when differing position to get velocity is prone to error.

## FP.6 Performance Evaluation 2

The results is stored in the results.csv in directory `/SFND_3D_Feature_Tracking/report/`

Based on the results.csv, there are a lot of combinations (of detectors and descriptors) that yields rather stable lidar-based TTC.

However, in midterm project, the top 3 detector/descriptor combinations are FAST/BRIEF, FAST/BRISK, and FAST/ORB which are the fastest in execution time while maintaining rather good portion of keypoints on the preceding vehicle, and matching rather good portion of keypoints between successive images. Let consider their camera-based TTC estimates in the following table. Overall, TTC estimates are rather stable except FAST/BRISK has 1 irrelevant estimate.

*Table 2: Results syntheses for FAST/BRIEF, FAST/BRISK, and FAST/ORB.*

| Detector | Descriptor | Image id | No matched keypoints in ROI | TTC lidar | TTC camera |
|---|---|---|---|---|---|
| FAST | BRISK | 0 | 0 | nan | nan |
| FAST | BRISK | 1 | 58 | 12.9722 | 12.3 |
| FAST | BRISK | 2 | 67 | 12.264 | 12.3453 |
| FAST | BRISK | 3 | 75 | 13.9161 | 14.1877 |
| FAST | BRISK | 4 | 68 | 7.11572 | 12.8857 |
| FAST | BRISK | 5 | 65 | 16.2511 | -Inf |
| FAST | BRISK | 6 | 69 | 12.4213 | 13.0386 |
| FAST | BRISK | 7 | 77 | 34.3404 | 12.041 |
| FAST | BRISK | 8 | 72 | 9.34376 | 11.4066 |
| FAST | BRISK | 9 | 75 | 18.1318 | 11.8684 |
| FAST | BRISK | 10 | 74 | 18.0318 | 13.3473 |
| FAST | BRISK | 11 | 67 | 3.83244 | 12.9492 |
| FAST | BRISK | 12 | 86 | nan | 12.1174 |
| FAST | BRISK | 13 | 87 | 9.22307 | 12.1074 |
| FAST | BRISK | 14 | 82 | 10.9678 | 11.6077 |
| FAST | BRISK | 15 | 79 | 8.09422 | 11.4079 |
| FAST | BRISK | 16 | 86 | 3.17535 | 12.2566 |
| FAST | BRISK | 17 | 89 | nan | 9.2933 |
| FAST | BRISK | 18 | 84 | 8.30978 | 11.8606 |
| FAST | BRIEF | 0 | 0 | nan | nan |

| | | | | | |
|---|---|---|---|---|---|
| FAST | BRIEF | 1 | 83 | 12.9722 | 11.1897 |
| FAST | BRIEF | 2 | 85 | 12.264 | 12.5319 |
| FAST | BRIEF | 3 | 92 | 13.9161 | 17.3793 |
| FAST | BRIEF | 4 | 88 | 7.11572 | 13.6612 |
| FAST | BRIEF | 5 | 84 | 16.2511 | 30.2446 |
| FAST | BRIEF | 6 | 81 | 12.4213 | 13.5856 |
| FAST | BRIEF | 7 | 97 | 34.3404 | 12.2781 |
| FAST | BRIEF | 8 | 82 | 9.34376 | 12.5357 |
| FAST | BRIEF | 9 | 87 | 18.1318 | 13.4042 |
| FAST | BRIEF | 10 | 88 | 18.0318 | 13.4879 |
| FAST | BRIEF | 11 | 83 | 3.83244 | 14.132 |
| FAST | BRIEF | 12 | 93 | nan | 12.6994 |
| FAST | BRIEF | 13 | 94 | 9.22307 | 12.3705 |
| FAST | BRIEF | 14 | 104 | 10.9678 | 11.7034 |
| FAST | BRIEF | 15 | 97 | 8.09422 | 12.1403 |
| FAST | BRIEF | 16 | 102 | 3.17535 | 12.5062 |
| FAST | BRIEF | 17 | 104 | nan | 8.48654 |
| FAST | BRIEF | 18 | 96 | 8.30978 | 12.6126 |
| FAST | ORB | 0 | 0 | nan | nan |
| FAST | ORB | 1 | 75 | 12.9722 | 12.0985 |
| FAST | ORB | 2 | 84 | 12.264 | 12.8193 |
| FAST | ORB | 3 | 89 | 13.9161 | 16.3163 |
| FAST | ORB | 4 | 84 | 7.11572 | 13.9934 |
| FAST | ORB | 5 | 82 | 16.2511 | 37.8535 |
| FAST | ORB | 6 | 78 | 12.4213 | 14.0835 |
| FAST | ORB | 7 | 86 | 34.3404 | 12.559 |
| FAST | ORB | 8 | 90 | 9.34376 | 11.9141 |
| FAST | ORB | 9 | 93 | 18.1318 | 12.8711 |
| FAST | ORB | 10 | 89 | 18.0318 | 13.8572 |
| FAST | ORB | 11 | 78 | 3.83244 | 18.3152 |
| FAST | ORB | 12 | 87 | nan | 12.7731 |
| FAST | ORB | 13 | 94 | 9.22307 | 12.9215 |
| FAST | ORB | 14 | 96 | 10.9678 | 11.1115 |
| FAST | ORB | 15 | 97 | 8.09422 | 10.8439 |
| FAST | ORB | 16 | 100 | 3.17535 | 11.3939 |
| FAST | ORB | 17 | 102 | nan | 10.6085 |
| FAST | ORB | 18 | 95 | 8.30978 | 12.6746 |

The camera-based TTC estimates are worse if using HARRIS detector as depicted in the following table as an example of HARRIS/BRISK. The reason is that HARRIS detector provides less keypoints than other detectors making the TTC computation is not reliable.

*Table 3: Results syntheses for HARRIS/BRISK.*

| Detector | Descriptor | Image id | No matched keypoints in ROI | TTC lidar | TTC camera |
|---|---|---|---|---|---|
| HARRIS | BRISK | 0 | 0 | nan | nan |
| HARRIS | BRISK | 1 | 7 | 12.9722 | 10.9082 |
| HARRIS | BRISK | 2 | 8 | 12.264 | 10.586 |
| HARRIS | BRISK | 3 | 7 | 13.9161 | -80.8525 |
| HARRIS | BRISK | 4 | 11 | 7.11572 | 11.5792 |
| HARRIS | BRISK | 5 | 12 | 16.2511 | -Inf |
| HARRIS | BRISK | 6 | 9 | 12.4213 | 12.9945 |
| HARRIS | BRISK | 7 | 15 | 34.3404 | 11.6947 |
| HARRIS | BRISK | 8 | 14 | 9.34376 | 17.6204 |
| HARRIS | BRISK | 9 | 7 | 18.1318 | nan |
| HARRIS | BRISK | 10 | 3 | 18.0318 | -Inf |
| HARRIS | BRISK | 11 | 6 | 3.83244 | -Inf |
| HARRIS | BRISK | 12 | 8 | nan | 12.245 |
| HARRIS | BRISK | 13 | 12 | 9.22307 | 568.322 |
| HARRIS | BRISK | 14 | 9 | 10.9678 | 7.72144 |
| HARRIS | BRISK | 15 | 9 | 8.09422 | -13.6263 |
| HARRIS | BRISK | 16 | 17 | 3.17535 | 6.65726 |
| HARRIS | BRISK | 17 | 7 | nan | 12.5848 |
| HARRIS | BRISK | 18 | 4 | 8.30978 | -Inf |

For example, the following figure shows that HARRIS produced only 12 matched keypoints but some of them are located on the farther vehicle on the left lane.

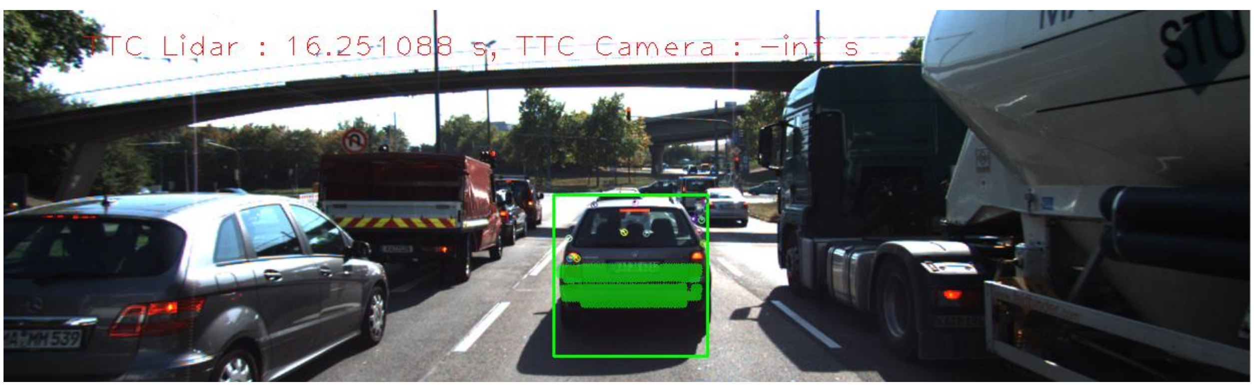



*Figure 4: Front views displayed bounding box, lidar points, and keypoints using HARRIS/BRISK for image 5.*