3D Tracking - Collision Detection/Avoidance

UDACITY

Contents

1 FP.1 Matching 3D Objects

Figure 1: Matching of bounding boxes. Same colors indicate bounding box match.

The matching of the bounding boxes detected by the Yolo neural network is keypoint based.

1. In a first step keypoints are detected, described, and matched between the previous and current frame. Then these matchings are associated to the bounding boxes to get a bounding box similiariy metric.

   This metric is calculated by iterating thorugh all keypoints of box i and the corresponding matches in box j. Then the distance of the corresponding points is summed:

   $$d_{i,j} = \frac{1}{N} \sum_{u \in \text{keyPtsBox}_i} \sum_{n \in \text{kptMatchBox}_j} \text{dist}(p_u - p_n), \tag{1}$$

   whereby N is the number of keypoint pairs considered. Moreover if there are not enough keypoint matches between two bounding boxes - no similiarity metric is calculated.

		Bounding-Box ID Prev		
		1	2	3
Bounding-Box ID Current	1	Similariy metric based on keypoints similarities		
	2			
	3			
	4			

Figure 2: Matrix of bounding box similarities.

2. Then from the bounding box similiary metric table, the most proimising correspondences are extraced. As a match the one with the highest similiary metric $d_{i,j}$ is choosen. The goal is a 1:1 correspondence, thus the matching of multiple boxes is not allowed.

The result of this procedure can be seen in figure 2. Mached bounding boxes are associated the same color.

2 FP.2 LidarBased TTC

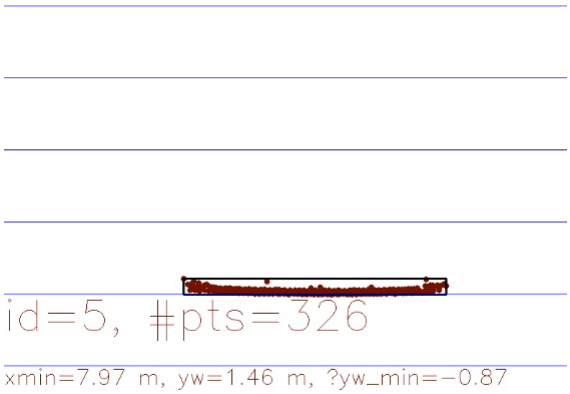


Figure 3: Lidar points of the front vehicle in the go lane - in top view.

To calculate the TTC from lidar based measurements, the keypoints associated with the ego lane bounding box are evaluate - see figure 3. The formula for calculating the lidar based time to collision is

$$TTC_{lidar} = \bar{x}_k \frac{dt}{\bar{x}_{k-1} - \bar{x}_k} \tag{2}$$

To gain some robustness against outliers, the center of the vehicle is calculated and only points near the vehicle center are taken into account. Furthermore the lidar points are sorted in an ascending order (closest first) and the 0.6 percentile is taken.

$$\bar{x}_k = \text{percentile}(x, 0.6) \tag{3}$$

Maybe Ransac would also be an option here.

3 FP.3 Associate Keypoint Correspondences with Bounding Boxes

To associate keypoints to bounding boxes all keypoint matches are iterated. Then the keypoints that are within the bounding box ROI are kept. This criteria is checked using

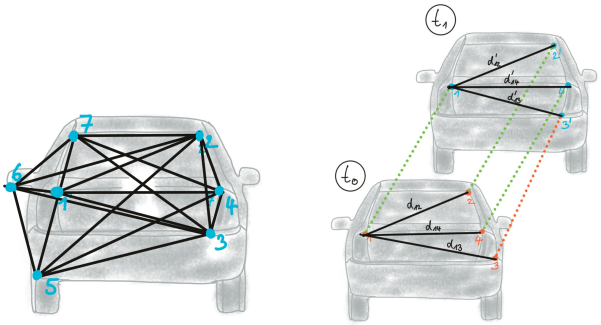
the contain method of OpenCV. Furthermore, outliers are filtered using 90% of the distance values as threshold.

```
    size_t na = kptDist.size() * 0.9; // 10% outliers
    std::nth_element(kptDist.begin(), kptDist.begin() + na, kptDist.end
                                        ());
    double medianKptDist = kptDist[na];
    double distThresh = medianKptDist * 1.1;

    for (auto match : kptMatches)
    { // loop through kptMatches, filter for bounding box
        cv::KeyPoint kptCurrent = kptsCurr[match.trainIdx];
        cv::KeyPoint kptPrev = kptsPrev[match.queryIdx];

        if(boundingBox.roi.contains(kptCurrent.pt) &&
           cv::norm(kptPrev.pt - kptCurrent.pt) < distThresh)
        {
            boundingBox.keypoints.push_back(kptCurrent);
            boundingBox.kptMatches.push_back(match);
        }
    }
```

4 FP.4 CameraBased TTC



(a) relative distances between keypoints in an image

(b) selected keypoint distances between successive frames

Figure 4: Keypoint count and size distribution - plotted for first image

To compute the camera TTC - the keypoint matches from one bounding box are considered. Then the distance ratios between all keypoints of the matches associated with the bounding box are calculated

$$r = \frac{d_{i,j,curr}}{d_{i,j,prev}} = \frac{|p_{i,curr} - p_{j,curr}|}{|p_{i,prev} - p_{j,prev}|} \tag{4}$$

With the distance ratios the TTC can be calculateds as.

$$TTC_{camera} = \frac{dt}{1 - \bar{r}}, \tag{5}$$

where by $\bar{r}$ is the median of all keypoint matches distance ratios.

5 Performance evaluation

6 FP.5 Performance Evaluation 1

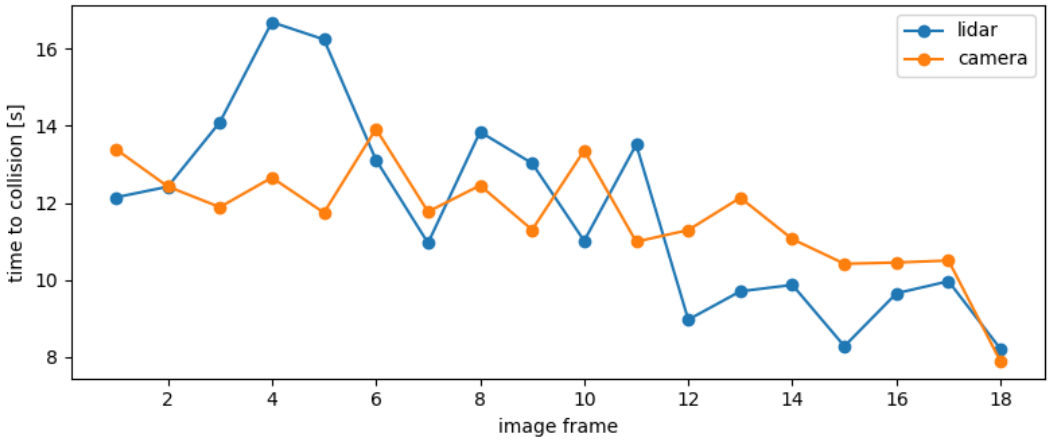


Figure 5: Estimation of TTC from lidar and camera over image frame for different detector / descriptor combinations.

It can be seen in figure 6 that the Lidar TTC estimates seem pretty plausible. Errors are generally due to:

- constant velocity model,
- surface shape / properties of reflecting object / ground not flat,
- lidar and framerate accuracy,
- partial occlusion,
- outlier handling / angle of previous car (currently the height information / z-component from the lidar measurement is neglected).

Figure 6: Estimation of TTC from lidar and camera over image frame

7 FP.5 Performance Evaluation 2

All different detector / descriptor combinations where tested - see figures 8, 9, 10.
The following can be observed

- in general the detector seems to be more important than the descriptor.
- the ORB and HARRIS detectosr seems very unreliable.
- AKAZE, SIFT and SHITOMASI detectors seem most consistent.
- FAST seems a performant and also pretty reliable solution - but has problems with the transition from frame 4 t o5

In general the camera estimates seem more consistent than the lidar estimate. But for some frames the TTC is largely off. The following problems occure for camera based TTC estimation:

- erroneous matching
- changes in illumnation / viewpoint
- keypoints that are not part of the obstacle (shadow or other cars)
- partial occlusion

In figure 7 one can see, that keypoints in the shadow and other vehicles are matched .

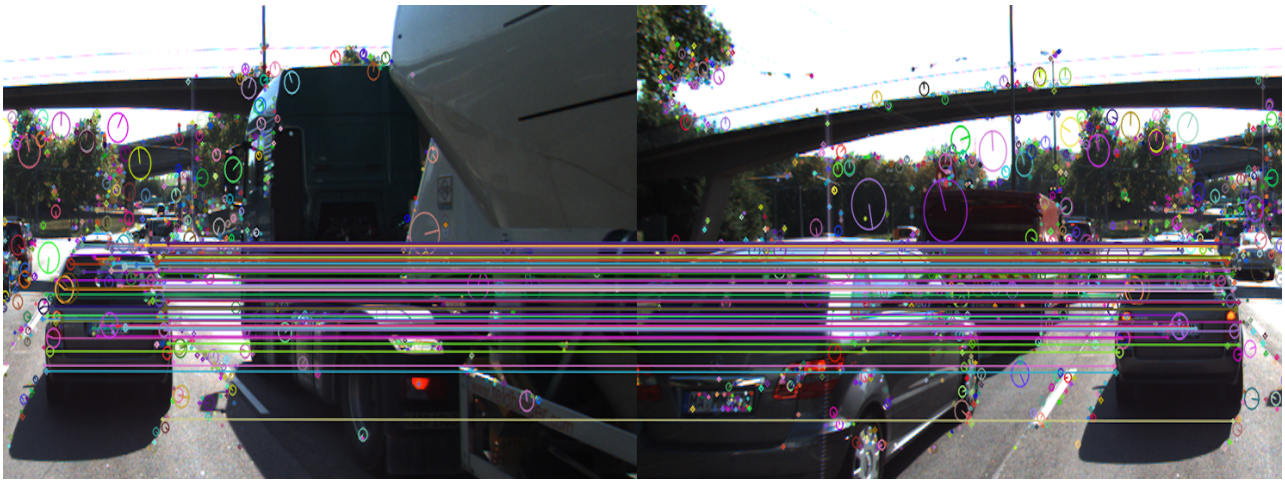


Figure 7: Example of a keypoint in the shadow.

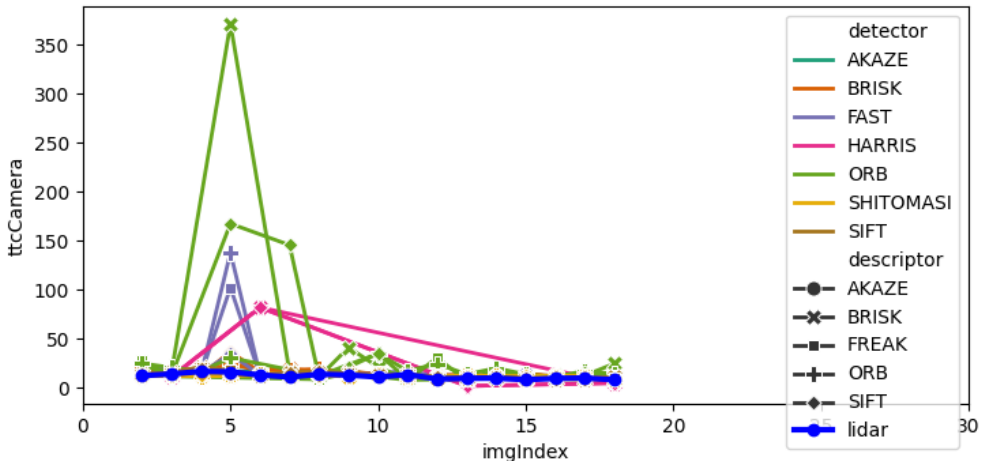


Figure 8: Estimation of TTC from camera over image frame.

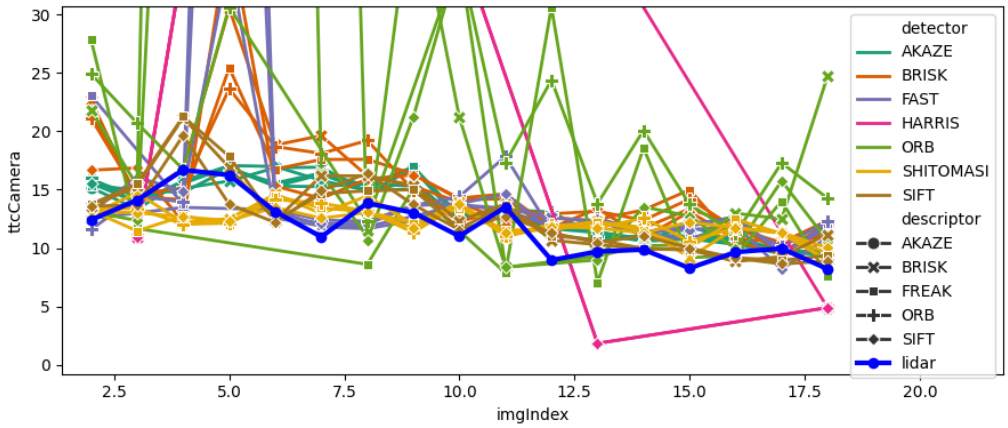


Figure 9: Estimation of TTC from lidar and camera over image frame.

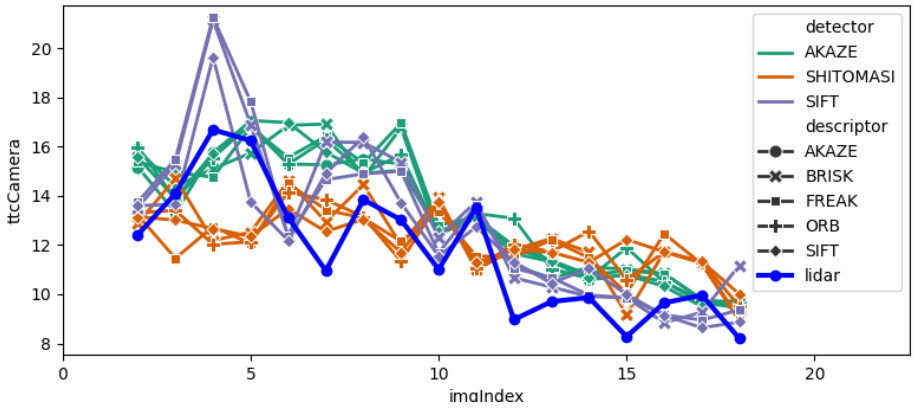


Figure 10: Estimation of TTC from lidar and camera over image frame.