FP.1 Match 3D Objects:

- Loop through all matches. For each match, find corresponding boxes pair. Then, update occurrence table.
- For each box in previous frame, check occurrence table to find the corresponding box in current frame which has the maximum occurrence

FP.2 Compute Lidar-based TTC:

- Check total number of lidar points. If total number is smaller than threshold, directly calculate average dis from all points and use current and previous average distance to calculate TTC
- If total number of points is larger than threshold, calculate TTC by focusing on those lidar points which have (y, z) location close to med(y)+/-tolY and med(z)+/-tolZ. Then, discard the points which have x position larger than top 10% of the whole population or smaller than bottom 10% of whole population. The rest of lidar point will be used to calculate distance.
- Calculate TTC by applying constant velocity model.

FP.3 Associate Keypoint Correspondences with Bounding Boxes:

- Loop through matches. Then, discard those matches which have current keypoints outside this ROI region.
- Then, loop though all the remaining matches and calculate distance shift between current keypoint and its corresponding previous keypoint.
- Accumulate all distance shifts and calculate mean shift.
- Discard matches that have shift larger than ((mean shift) + 2*(STD shift))

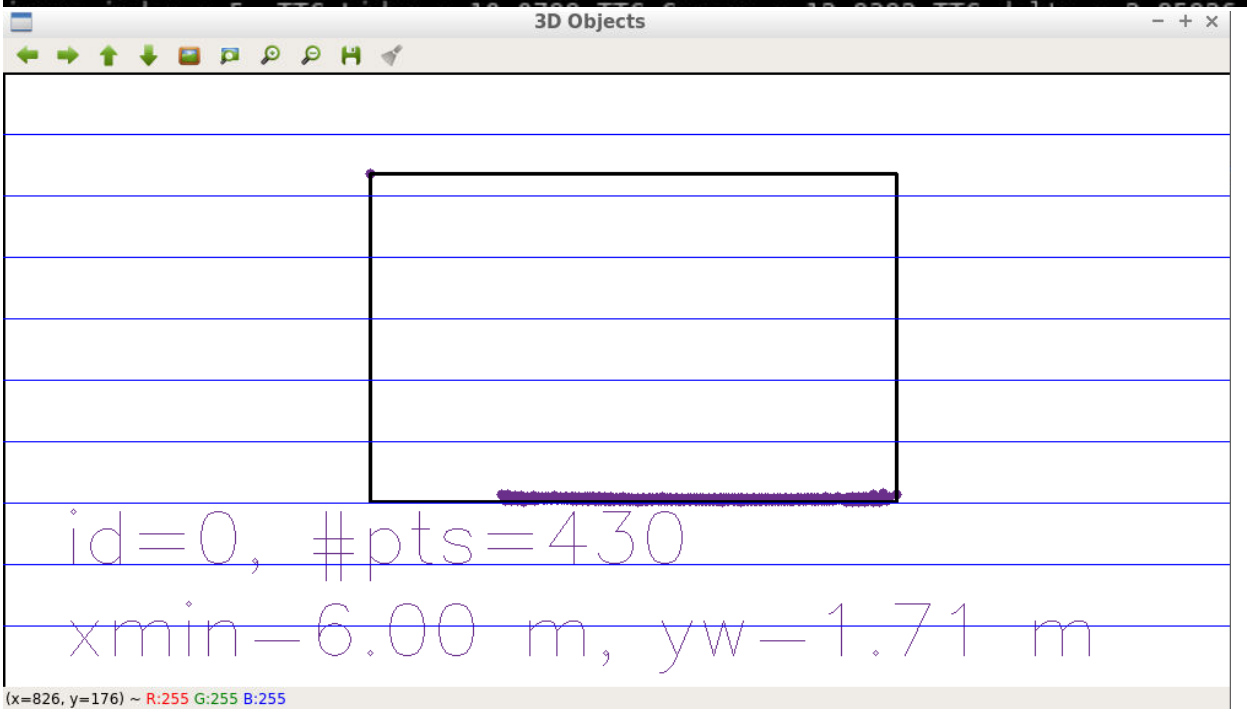
FP.4 Compute Camera-based TTC:

- Loop through all keypoint pair
- For each keyoint pair, calculate distance(dcurr) of the pair in current frame and distance(dprev) of the same pair in previous frame. Then, only use keypoint pair which has dcurr > 100 and dprev > numeric accuracy to calculate ratio = dcurr/ dprev.
- Store those ratios in a vector. Then, calculate median ratio and use median ratio to calculate TTC.

FP.5 Performance Evaluation 1:

- **The outlier of lidar point causes irregular jump of TTC estimation**

```
image index = 2, TTC Lidar = 12.7915 TTC Camera = 10.0906 TTC delta = 2.7009
image index = 3, TTC Lidar = 5.44837 TTC Camera = 9.77019 TTC delta = 4.32182
image index = 4, TTC Lidar = 8.80528 TTC Camera = 11.2285 TTC delta = 2.42323
```



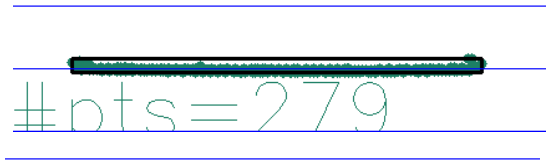
- **After discarding outlier by cutting off points which have x position within the largest 10 % of the whole population. Irregular jump of estimation is gone.**

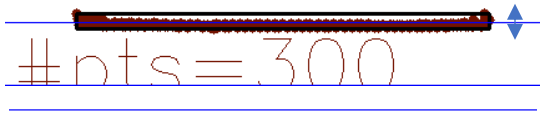
```
image index = 0, TTC Lidar = 7.47943 TTC Camera = 8.77742 TTC delta = 1.29799
image index = 1, TTC Lidar = 7.27077 TTC Camera = 9.5061 TTC delta = 2.23533
image index = 2, TTC Lidar = 8.47796 TTC Camera = 10.0906 TTC delta = 1.61262
image index = 3, TTC Lidar = 6.98098 TTC Camera = 9.77019 TTC delta = 2.78921
image index = 4, TTC Lidar = 9.07511 TTC Camera = 11.2285 TTC delta = 2.1534
```

- **However, there is another jump couldn't be solved by simply cutting of the data.**

```
image index = 0, TTC Lidar = 13.6663 TTC Camera = 11.711
image index = 1, TTC Lidar = 11.9185 TTC Camera = 11.7466
image index = 2, TTC Lidar = 19.1259 TTC Camera = 14.1553
image index = 3, TTC Lidar = 14.4276 TTC Camera = 13.0949
image index = 4, TTC Lidar = 13.1744 TTC Camera = 14.9186
```

- **Observed the x distance distribution differences between three different frames. Suspect that the distribution difference is the major issue for the irregular TTC jump.**





X position distribution is more spread-out

X position distribution is more concentrated

- **Therefore, I change the code that calculates average distance to the code that calculates median distance. It solves the problem.**

```
image index = 0, TTC Lidar = 12.3201 TTC Camera = 11.2412 TTC
image index = 1, TTC Lidar = 12.2199 TTC Camera = 11.2876 TTC
image index = 2, TTC Lidar = 14.7467 TTC Camera = 13.742 TTC c
image index = 3, TTC Lidar = 16.5095 TTC Camera = 13.7783 TTC
image index = 4, TTC Lidar = 15.5839 TTC Camera = 13.2401 TTC
image index = 5, TTC Lidar = 12.8867 TTC Camera = 12.1955 TTC
```

FP.6 Performance Evaluation 2:

- **ORB-BRISK (detector-descriptor) produces unstable TTC estimation**

```
image index = 0, TTC Lidar = 12.3201 TTC Camera = 12.1101 TTC delta = 0.209995
image index = 1, TTC Lidar = 12.2199 TTC Camera = 15.9384 TTC delta = 3.71853
image index = 2, TTC Lidar = 14.7467 TTC Camera = 12.0117 TTC delta = 2.73504
image index = 3, TTC Lidar = 16.5095 TTC Camera = 18.9249 TTC delta = 2.41539
image index = 4, TTC Lidar = 15.5839 TTC Camera = -inf TTC delta = inf
image index = 5, TTC Lidar = 12.8867 TTC Camera = 10.8366 TTC delta = 2.05006
image index = 6, TTC Lidar = 11.9813 TTC Camera = -inf TTC delta = inf
image index = 7, TTC Lidar = 13.1207 TTC Camera = 11.919 TTC delta = 1.20164
image index = 8, TTC Lidar = 13.0207 TTC Camera = -inf TTC delta = inf
image index = 9, TTC Lidar = 11.3424 TTC Camera = -inf TTC delta = inf
```

- **ORB-BRIEF produces unstable TTC estimation**

```
image index = 0, TTC Lidar = 12.3201 TTC Camera = 22.9508 TTC delta = 10.6307
image index = 1, TTC Lidar = 12.2199 TTC Camera = 34.2381 TTC delta = 22.0182
image index = 2, TTC Lidar = 14.7467 TTC Camera = 22.4481 TTC delta = 7.70132
image index = 3, TTC Lidar = 16.5095 TTC Camera = 21.4016 TTC delta = 4.89207
image index = 4, TTC Lidar = 15.5839 TTC Camera = 19.7893 TTC delta = 4.20531
image index = 5, TTC Lidar = 12.8867 TTC Camera = 10.2234 TTC delta = 2.66332
image index = 6, TTC Lidar = 11.9813 TTC Camera = 102.418 TTC delta = 90.4367
image index = 7, TTC Lidar = 13.1207 TTC Camera = -inf TTC delta = inf
image index = 8, TTC Lidar = 13.0207 TTC Camera = 4.5545e+06 TTC delta = 4.55449e+06
image index = 9, TTC Lidar = 11.3424 TTC Camera = 15.1252 TTC delta = 3.78277
```

- **ORB-ORB also produces unstable TTC estimation**

```
image index = 0, TTC Lidar = 12.3201 TTC Camera = 37.9479 TTC delta = 25.6278
image index = 1, TTC Lidar = 12.2199 TTC Camera = -inf TTC delta = inf
image index = 2, TTC Lidar = 14.7467 TTC Camera = 19.3282 TTC delta = 4.58144
image index = 3, TTC Lidar = 16.5095 TTC Camera = 14.2496 TTC delta = 2.25993
image index = 4, TTC Lidar = 15.5839 TTC Camera = 154.646 TTC delta = 139.062
image index = 5, TTC Lidar = 12.8867 TTC Camera = -inf TTC delta = inf
image index = 6, TTC Lidar = 11.9813 TTC Camera = -inf TTC delta = inf
image index = 7, TTC Lidar = 13.1207 TTC Camera = -inf TTC delta = inf
image index = 8, TTC Lidar = 13.0207 TTC Camera = -inf TTC delta = inf
image index = 9, TTC Lidar = 11.3424 TTC Camera = -inf TTC delta = inf
```

- **Conclude that ORB detector can't produce good results. Also, observed less match counts when using ORB detector. This problem happens to Harris detector too. Harris detector also produces unstable TTC prediction due to less match counts which is even less than ORB.**

- **Back to test detector-descriptor combination selected during midterm project. Here is the performance table**

```
// Compute time-to-collision (TTC) based on keypoint correspondences in successive images
void computeTTCCamera(std::vector<cv::KeyPoint> &kptsPrev, std::vector<cv::KeyPoint> &kptsCurr,
                      std::vector<cv::DMatch> kptMatches, double frameRate, double &TTC, cv::Mat *visImg)
{
    // extract first match point (p1 current image, p1 previous image)
    // loop through all the other match to get (p2 current image, p2 previous image)
    // calculate ratio = norm(p1 current image - p2 current image) / norm(p1 previous image-p2 previous image)
    // store ratio > thresh into vector distRatios
    vector<double> distRatios;
    for (int i=0; i<kptMatches.size()-1; i++)
    {
        cv::KeyPoint pt1Prev, pt1Curr;
        pt1Prev = kptsPrev[kptMatches[i].queryIdx];
        pt1Curr = kptsCurr[kptMatches[i].trainIdx];

        for (int j = i+1; j<kptMatches.size(); j++)
        {
            double thresh = 35.0;
            cv::KeyPoint pt2Prev, pt2Curr;
            pt2Prev = kptsPrev[kptMatches[j].queryIdx];
            pt2Curr = kptsCurr[kptMatches[j].trainIdx];

            double disPrev, disCurr, disRatio;
            disPrev = cv::norm(pt1Prev.pt - pt2Prev.pt);
            disCurr = cv::norm(pt1Curr.pt - pt2Curr.pt);

            // avoiding division by zero and discard small distance
            if (disPrev > std::numeric_limits<double>::epsilon() && disCurr>thresh)
```

For first 30 Frames, mean (TTC Lidar – TTC Camera), unit: ms		
Det-Dec	**disCurr>90.0**	**disCurr>35.0**
FAST-BRISK	**2.0618**	**1.4881**
FAST-ORB	**1.6041**	**1.1518**
FAST-BRIEF	**1.6085**	**1.2322**

- **From the performance table we have following conclusion.**
    1. **Decrease threshold improves all three det-dec pairs performance. Here is the potential explanation. By reducing threshold, TTC calculation will include those distance from keypoints at center of the car to the keypoints at edge of the car. Those distances are crucial because they represent the change of the car size in image. However, with large threshold, the majority of distances are from keypoints at one side of the car to the other side of the car. However, many edge keypoints don't represent edge of the car, so larger threshold eventually resulted in degraded performance.**
    2. **The table also shows that FAST-ORB performance is the best.**