PROJECT SUBMISSION NOTES

Track an Object in 3D Space

FP.1 Match 3D Objects

CRITERIA	IMPLEMENTATION
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.	see below

For(each boundinbox in prevFrame,(say **bBpF**)) //***loop it starts***

{ **For**(each boundingbox in currFrame,(say **bBcF**))//***loop it1 starts***

{For(each element in cv::Dmatch structure ,**matches** with its **trainIdx** & **queryIdx**)

{
if(**currFrame.keypoints**[**trainIdx**] within **bBcF** & **prevFrame.keypoints**[**queryIdx**] within **bBpF)**
++ **temp1//** an int variable
}

// temp1 to be used with another int temp(initially 0) and bool crossChk(initially false}
// to update a 2D vector of int **tmps** & 1D int vector tmp as below

```
     if(temp1>temp)//steps to store prev and curr bounding box with max matches

      {
        temp = temp1;
        tmp = {boxIDprev,boxIDcurr1,temp};//boxID for prev,boxID curr,no: of shared
matches
        for(auto i=tmps.begin();i!=tmps.end();++i)
        {//check if curr bounding box already paired,
          // if so keep the pair with max matches and eliminate the other.
          if((*i).at(1)==boxIDcurr1&&(*i).at(2)>temp)
          {
              crossChk = true;
              continue;
          }
          if((*i).at(1)==boxIDcurr1)
          {
            tmps.erase(i);
            --i;
          }

        }//eof loop i
        if(crossChk == false)
            tmps.push_back(tmp);
        crossChk = false;
```

}

}***eof loop it1***

} ***eof loop it2***

```
for(auto i=tmps.begin();i!=tmps.end();++i)
{
  bbBestMatches.insert(std::make_pair((*i).at(0),(*i).at(1)));
}
```

FP.2 Compute Lidar-based TTC

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.

TTC for Lidar is calculated as

TTC = (minimum distance in current frame)*(1/frame-rate)/(difference between minimum distances between previous and current frames) .

To deal with outlier Lidar points
Points in bounding box are clusterd by
the function-**euclideanCluster** (similarly implemented in course1:Lidar).
Node, KdTree, Clust structs added to **camFusion.hpp.**

FP.3 Associate Keypoint Correspondences with Bounding Boxes

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.

clusterKptMatchesWithROI(*currBB, (dataBuffer.end() - 2)->keypoints, (dataBuffer.end() - 1)->keypoints, (dataBuffer.end() - 1)->kptMatches);
implemented by

a) Using a vector of cv::Dmatch variables, **tempMatches** to Store all **- kptMatches** with **trainIdx** so that **(dataBuffer.end() - 1)->keypoints[trainIdx]** is contained in the roi of **currBB**
b) Find only those elements of **tempMatches** which are neither outliers in current frame nor outliers in previous frame.
c) Push the matches found in (b) to **currBB->kptMatches** and corresponding **(dataBuffer.end() - 1)->keypoints[trainIdx]** to **currBB->keypoints**

step (b) implementation details

1) For loops are used to find cv::norm of difference of each keypoint with every other keypoint for both previous and current frames.
2) The average of above distances for each keypoint is stored in a vector and average of all the elements is stored in a double variable .(This is done separately for previous and current frames)
3) For looping is done over both the above vectors and tempMatches for checking indice positions against outlier condition.

FP.4 Compute Camera-based TTC

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.

Current frame's boundingBox->kptMatches is used to find keypoint matches in previous and current frames.

The Euclidean distance of each keypoint with every other point is calculated in both previous(dprev) and current (dcurr) frames and dcurr/dprev ratios are calculated for all dcurr>100 and dprev>0.

The median of all the dcurr/dprev ratios i.e medianDistRatio is calculated and TTC is calculated as below

TTC = -dT/(1- medianDistRatio) , where dT = 1/frameRate

FP.5 Performance Evaluation 1

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.

Lidar TTC estimates for Frames 4,5,7,12,17 does not seem plausible, even though clustering algorithm eliminates too far outliers but some stray points near to the rear of the vehicle are occasionally considered by algorithm as part of the vehicle, hence causing erroneous measurement.

FP.6 Performance Evaluation 2

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.

The details of TTC estimates of all detector / descriptor combinations implemented in previous chapters on a frame-by-frame basis is represented by the **Table FP.6** and **Graph FP.6** attached below.

All combinations Harris detector and ORB detectors are way off,(may be due to higher sensitivity localized intensity transition of these detectors)

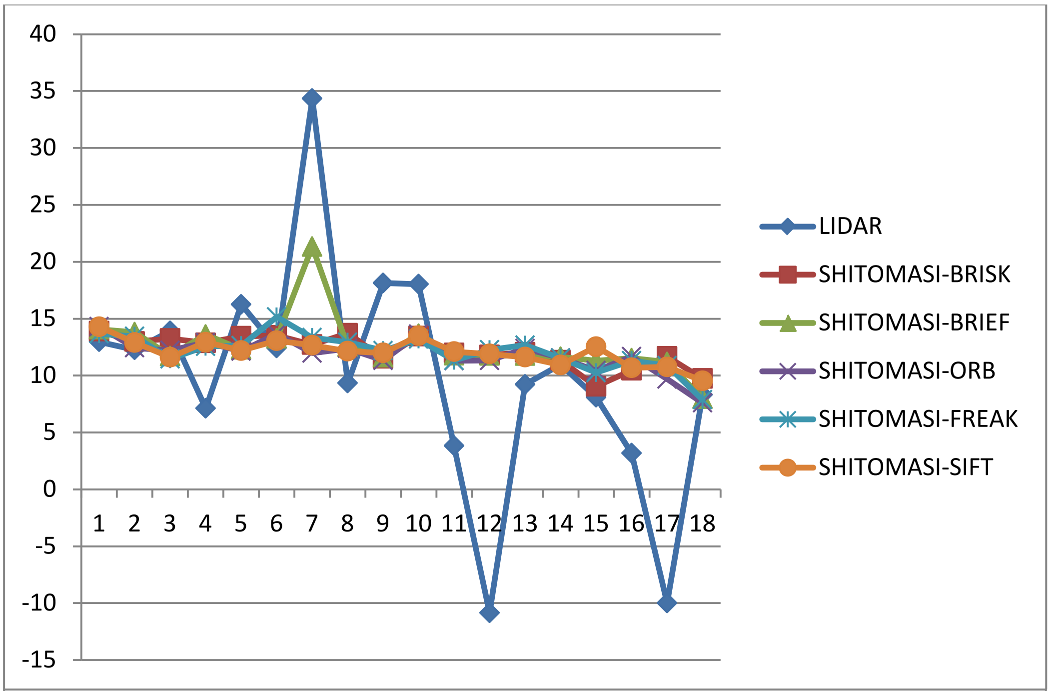
Recommendable combinations: "Shitomasi-BRIEF","FAST-SIFT", all combinations with AKAZE detector, these conclusions are based on absence of overshoots in the **Graphs FP.6** attached below **Table FP.6**.

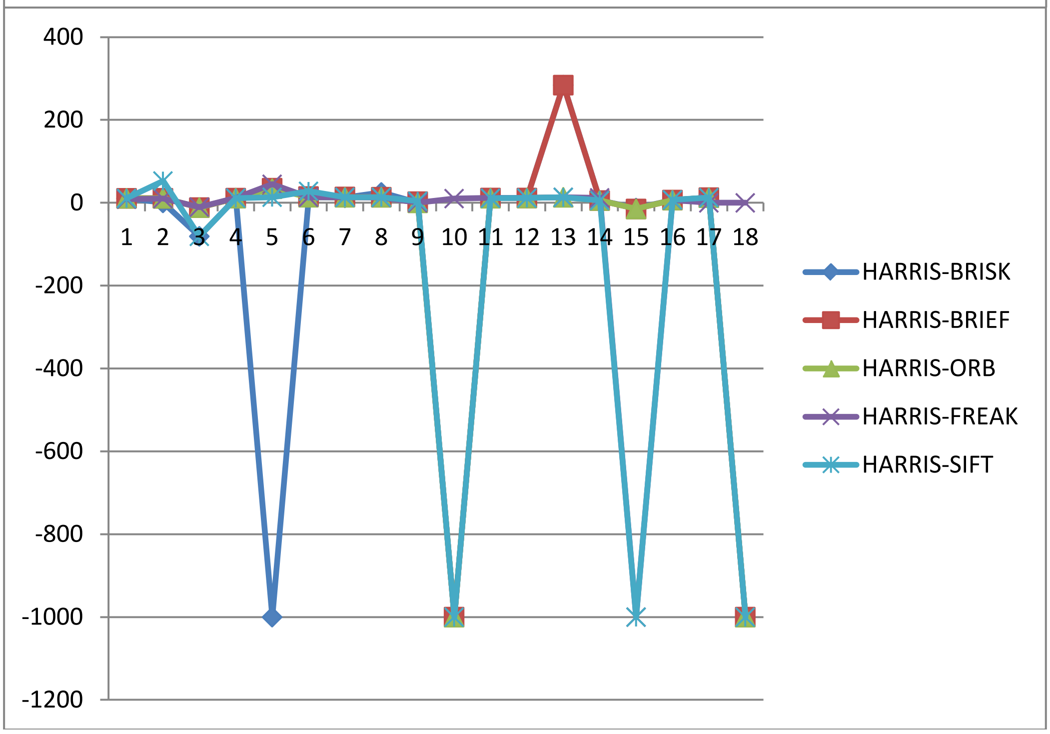
Table FP.6

LIDAR/ DETECTOR-DESCRIPTOR	TTC ESTIMATE FOR FRAMES NUMBERED IN THE ORDER DISPLAYED ON THE WINDOW																	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
LIDAR	13.0	12.3	13.9	7.1	16.3	12.4	34.3	9.3	18.1	18.0	3.8	-10.9	9.2	11.0	8.1	3.2	-10.0	8.3
SHITOMASI-BRISK	13.9	13.0	13.3	12.9	13.5	13.5	12.7	13.7	11.5	13.5	12.0	11.8	12.0	11.3	9.1	10.5	11.7	9.7
SHITOMASI-BRIEF	14.1	13.8	11.7	13.6	12.3	13.3	21.3	12.5	11.6	13.7	11.8	11.8	11.8	11.7	11.3	11.5	11.2	8.0
SHITOMASI-ORB	14.3	12.4	12.2	12.9	12.1	13.7	12.0	12.4	11.3	13.6	11.3	11.3	12.4	11.6	10.4	11.7	9.7	7.6
SHITOMASI-FREAK	13.7	13.5	11.5	12.6	12.6	15.2	13.3	12.9	12.2	13.2	11.3	12.3	12.7	11.5	10.3	11.3	10.9	7.9
SHITOMASI-AKAZE	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
SHITOMASI-SIFT	14.3	12.9	11.6	12.9	12.2	13.1	12.7	12.1	12.0	13.5	12.1	11.9	11.6	10.9	12.5	10.7	10.8	9.5
HARRIS-BRISK	10.9	10.2.	-80.9	11.4	(-)Inf	13.0	12.2	25.1	nan	(-)Inf	11.7	11.7	284.0	5.7	-12.6	6.3	12.6	(-)Inf
HARRIS-BRIEF	10.9	11.0	-11.5	11.4	35.4	15.3	14.3	13.6	2.9	(-)Inf	11.7	11.7	284.0	5.6	-14.8	6.8	12.6	(-)Inf
HARRIS-ORB	10.9	11.0	-11.5	11.4	34.2	13.6	13.5	13.6	nan	(-)Inf	11.2	11.7	13.4	5.7	-14.8	6.7	12.6	(-)Inf
HARRIS-FREAK	8.8	10.2	-10.3	11.8	44.9	13.6	13.5	12.9	nan	10.3	11.7	11.7	13.4	12.1	(-)Inf	6.6	nan	nan
HARRIS-AKAZE	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
HARRIS-SIFT	10.9	52.8	-80.9	11.4	13.6	27.9	13.5	13.6	3.9	(-)Inf	11.2	11.7	13.4	5.7	(-)Inf	7.3	12.6	(-)Inf
FAST-BRISK	12.3	12.4	13.0	13.0	(-)Inf	13.1	12.1	11.4	12.0	13.6	13.2	12.2	12.0	11.6	11.8	12.7	8.7	11.9
FAST-BRIEF	11.3	10.6	12.8	13.6	18.8	13.5	12.1	12.0	12.3	12.5	14.2	11.1	12.1	11.7	11.6	11.7	7.6	11.1
FAST-ORB	12.1	10.6	12.6	13.9	32.2	12.7	12.7	12.0	11.8	13.6	13.7	11.5	12.5	11.3	10.5	11.8	9.8	11.8
FAST-FREAK	11.9	24.2	13.9	14.4	80.2	12.4	12.5	11.6	12.3	13.3	13.0	11.6	11.8	11.4	11.1	12.3	8.5	12.1
FAST-AKAZE	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
FAST-SIFT	11.4	11.7	14.1	14.7	28.5	13.2	12.3	12.6	12.1	13.8	14.3	11.8	12.5	11.5	11.4	11.3	7.5	10.8
BRISK-BRISK	13.4	21.5	12.6	15.2	27.7	18.3	17.1	16.1	14.8	13.9	13.1	11.3	11.9	12.4	12.7	11.5	9.3	10.8
BRISK-BRIEF	12.9	16.9	11.7	20.6	19.7	17.6	15.6	18.8	15.5	11.5	13.4	14.3	12.3	11.0	11.6	12.2	11.4	10.7
BRISK-ORB	14.8	18.8	13.3	16.4	20.2	19.9	18.2	17.0	14.6	11.3	13.3	11.4	11.8	12.8	11.2	11.6	10.3	11.3
BRISK-FREAK	12.8	24.1	14.0	13.9	23.1	16.9	15.5	18.8	15.0	13.7	12.9	11.5	11.4	11.3	13.3	10.4	9.3	10.9
BRISK-AKAZE	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
BRISK-SIFT	13.0	16.1	16.7	12.6	26.3	14.5	14.0	17.0	17.9	14.7	16.0	11.7	13.3	10.6	14.2	11.0	10.2	12.3
ORB-BRISK	15.0	13.1	12.4	29.9	156	13.7	10.9	13.1	19.5	8.4	(-)Inf	8.0	7.5	9.5	8.7	13.3	12.4	26.0
ORB-BRIEF	20.7	47.9	103	14.1	22.9	15.1	55.2	76.8	70.7	16.3	17.5	14.1	13.3	9.7	8.8	11.0	13.4	18.4
ORB-ORB	19.5	10.8	16.7	21.7	50.6	32.0	19503	10.7	141	16.9	9.2	26.0	9.5	9.6	12.9	10.5	16.7	30.3
ORB-FREAK	12.2	20.1	17.5	10.9	(-)Inf	(-)Inf	(-)Inf	9.0	13.3	(-)Inf	8.0	35.9	6.5	57.4	8.6	9.3	11.3	16.9
ORB-AKAZE	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
ORB-SIFT	12.8	10.5	11.7	209	502	(-)Inf	10.5	15.1	11.1	10.1	(-)Inf	(-)Inf	7.9	18.7	13.7	9.6	18.3	10.8
AKAZE-BRISK	11.9	15.6	13.5	14.4	14.8	14.8	16.2	14.0	13.9	12.0	12.5	10.4	10.2	10.0	10.2	10.4	9.6	9.0
AKAZE-BRIEF	13.3	15.4	13.7	14.0	15.1	13.9	15.9	14.5	13.5	11.7	12.4	11.7	10.1	9.9	9.5	9.7	9.5	9.1
AKAZE-ORB	12.5	14.6	13.3	14.4	15.6	13.8	15.6	14.2	13.4	11.8	12.4	11.5	10.0	10.4	10.5	9.7	9.0	9.1
AKAZE-FREAK	12.1	14.1	14.0	14.3	16.1	14.5	15.5	13.7	13.1	12.0	12.2	11.0	11.0	10.0	9.9	10.4	9.1	8.7
AKAZE-AKAZE	12.3	14.1	12.9	14.5	16.8	13.9	15.3	14.1	13.8	11.6	12.1	11.1	11.3	10.6	10.2	9.8	9.1	9.0
AKAZE-SIFT	12.4	14.6	13.2	14.1	16.6	14.0	15.8	13.8	14.1	11.8	12.2	11.2	10.9	10.5	9.9	10.1	9.0	9.0
SIFT-BRISK	11.7	13.3	14.5	18.9	14.9	11.6	14.4	15.6	13.1	10.8	13.2	10.3	10.0	9.6	9.5	8.8	9.5	9.3
SIFT-BRIEF	12.1	14.2	15.1	21.3	13.6	12.6	14.6	15.3	13.0	10.3	13.1	9.9	10.1	9.3	10.0	9.1	9.0	9.2

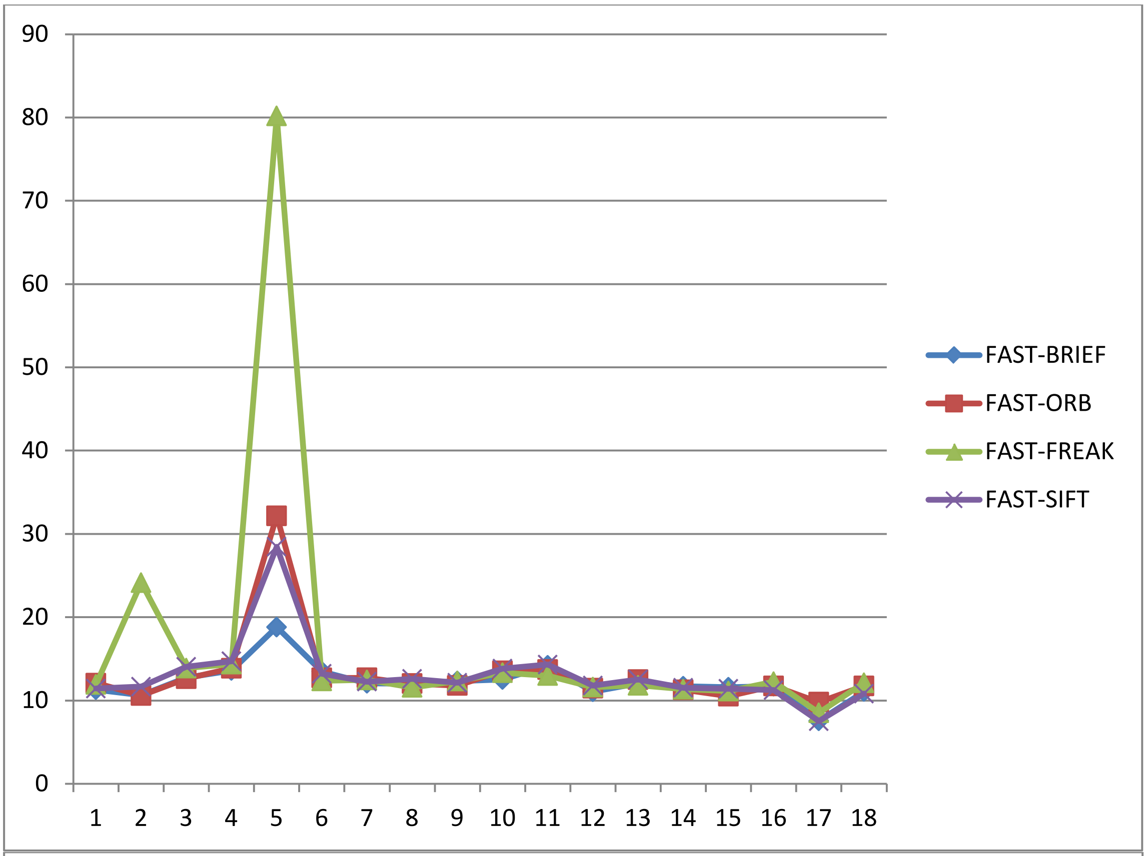
SIFT-ORB	**SHOWS INSUFFICIENT MEMORY DURING RUNTIME**																	
SIFT-FREAK	11.3	13.8	13.6	21.2	13.6	12.2	14.8	15.2	14.4	11.6	12.1	10.1	10.3	9.9	9.2	9.5	9.2	9.9
SIFT-AKAZE	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
SIFT-SIFT	12.0	12.6	13.2	19.6	15.3	11.1	13.9	15.6	15.0	10.6	11.7	11.6	9.3	10.6	10.4	9.6	8.9	9.0

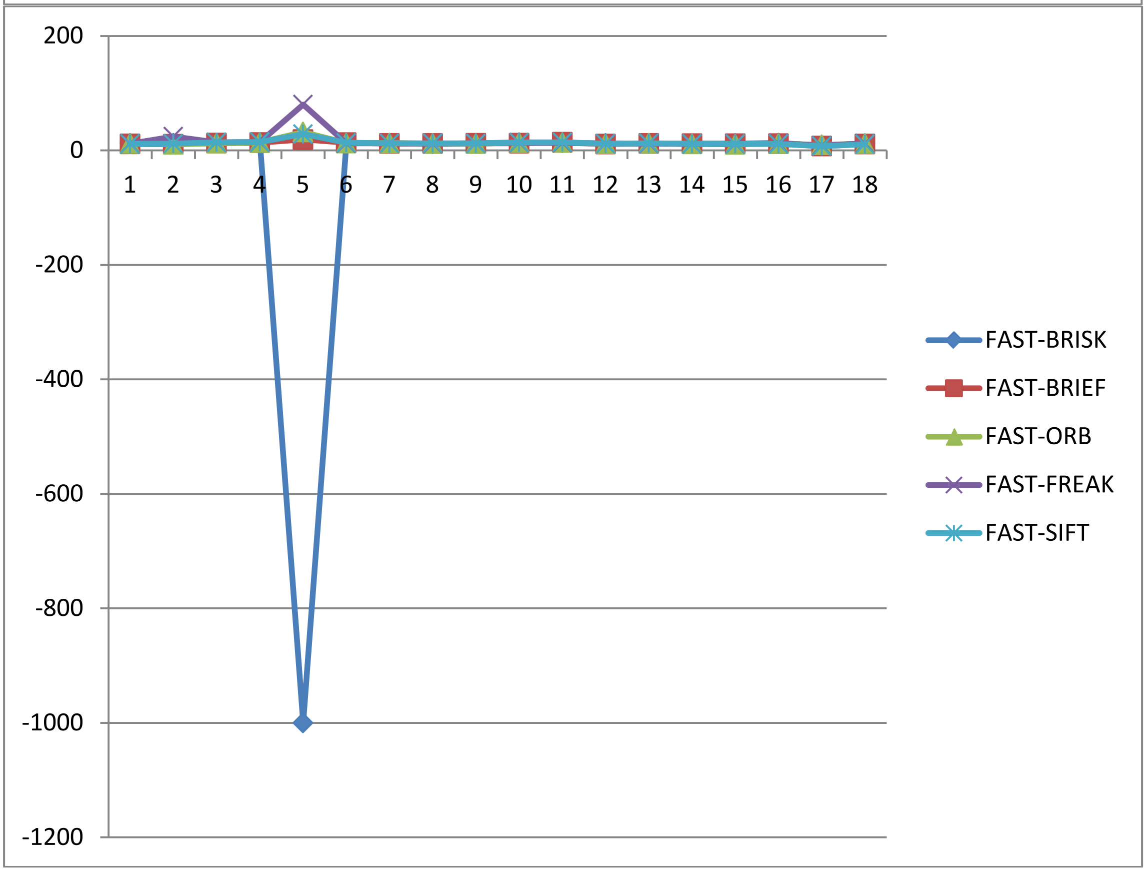
Graphs FP.6 (“frame number” on x axis and “TTC” on y axis)



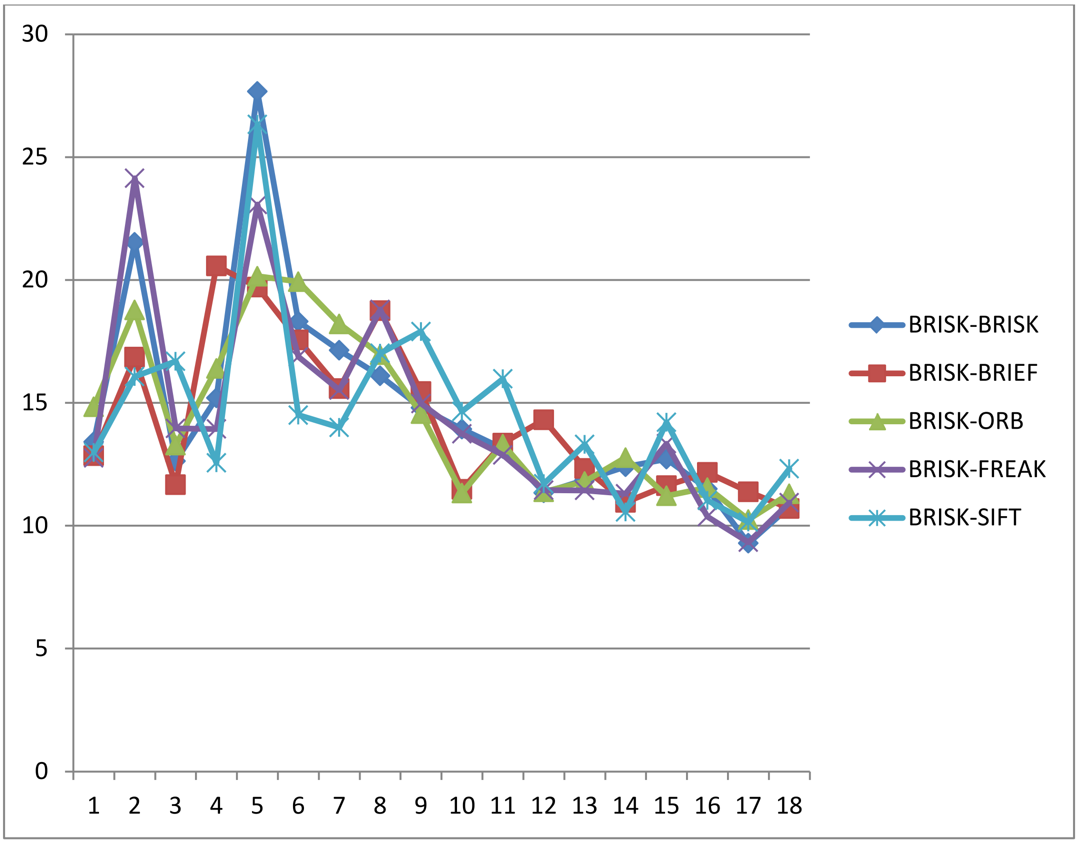


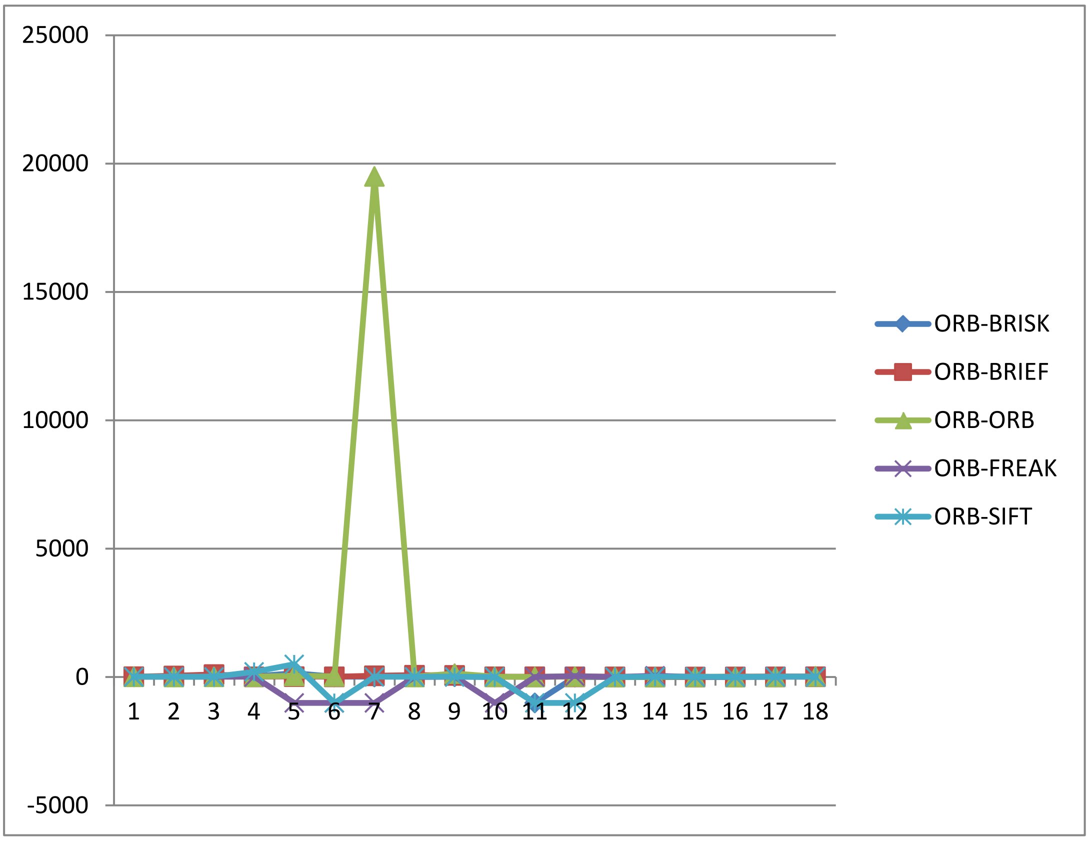
All descriptor combinations with Harris detector seem unsuitable.



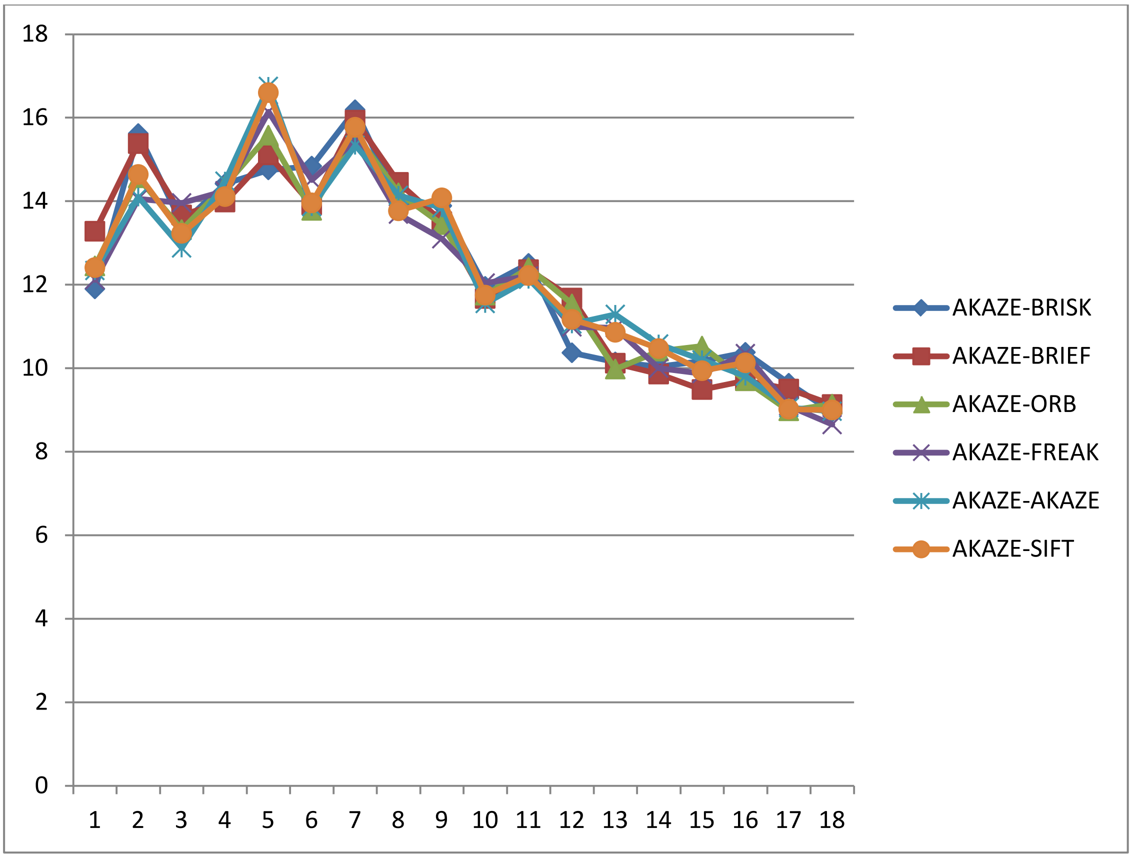


FAST-BRISK & FAST-BRIEF combinations seem unsuitable

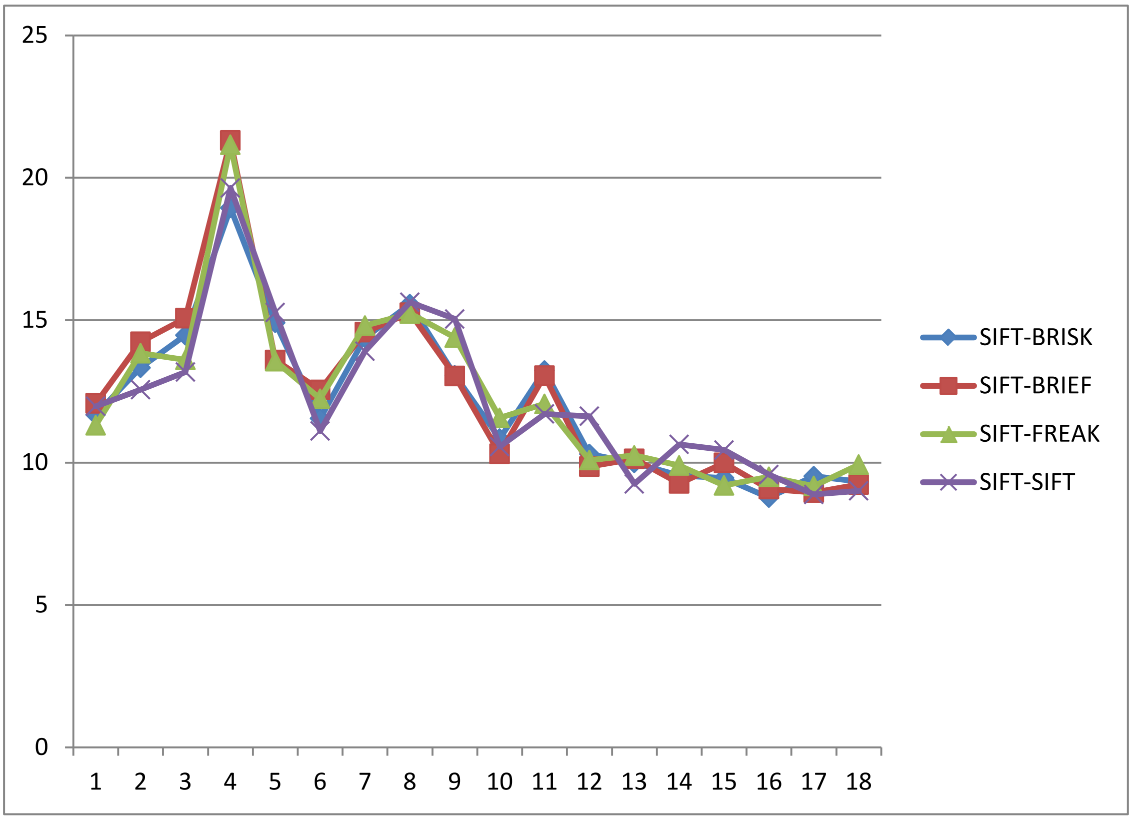




All descriptor combinations with ORB detector are prone to either too high or too low estimates.



18
16
14
12
10
8
6
4
2
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
AKAZE-BRISK
AKAZE-BRIEF
AKAZE-ORB
AKAZE-FREAK
AKAZE-AKAZE
AKAZE-SIFT



25
20
15
10
5
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
SIFT-BRISK
SIFT-BRIEF
SIFT-FREAK
SIFT-SIFT