Performance evaluation of different OpenCV detectors and descriptors

This performance evaluation is done as a part of Sensor Fusion Nano Degree course. 6 different descriptors and 5 different detectors were implemented and the different combinations have been evaluated to assess the best pair of detector and descriptor for the purpose of detecting the car in front of the ego car.

Detectors implemented:

1. Harris
2. BRISK
3. FAST
4. ORB
5. SIFT
6. AKAZE

Descriptors implemented:

1. BRIEF
2. ORB
3. FREAK
4. SIFT
5. AKAZE

You can see results of the tasks in the following pages.

Task 1: Number of keypoints detected and neighborhood size.

The first task is to evaluate the number of keypoints a detector will detect. On the whole there were a lot of key points detected on the image frame. But in the project our region of interest in the car in front of the ego vehicle. So the number of points and average keypoint size on the car was counted for each of the 10 frames and the average is tabulated in Table 1.

Detector Type	No of Keypoints on car	Average keypoint size
Harris	14	2
BRISK	272	23
FAST	51	7
ORB	115	57
AKAZE	138	8
SIFT	57	6

Table 1: Detector performance

Conclusion of Task 1:

The average keypoint size is the size of the neighborhood considered by the detector to uniquely defined a keypoint. The bigger the value is the more likely it is to find the same keypoint on the consecutive image. So here we can clearly see that the **ORB algorithm is the bes**t in performing a good keypoint detection.

The following are the top 3 detectors that will be recommended

1. **BRISK**
2. **ORB**
3. **AKAZE**

Task 2: Number of matches for the detectors – descriptor pair:

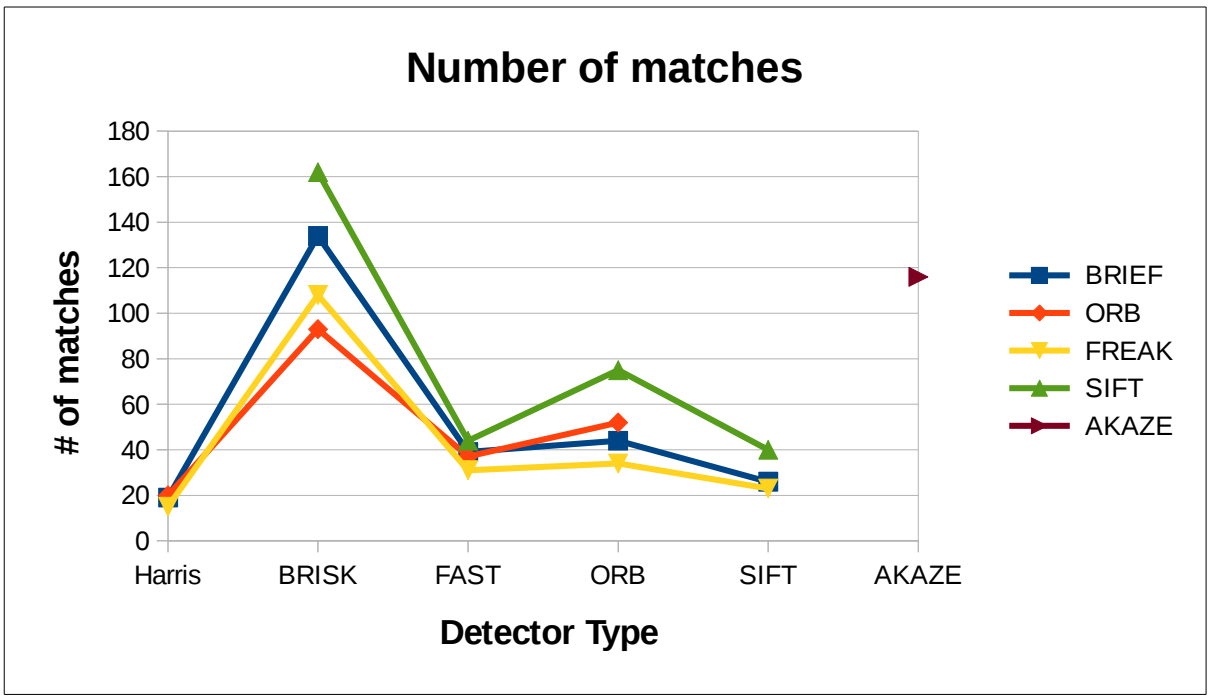
The second task was to evaluate the different keypoint detector-description pair and find out the best performing pair to be used in our particular use case, which is, detect the vehicle straight in front of the ego car. In the following table (Table 2), the number of matches between the different detector and descriptor is presented.

Note: A red cell means that the keypoint is not supported by the descriptor and hence cannot be evaluated. For example, the AKAZE detector only supports AKAZE keypoint and so it cannot be evaluated against other descriptors.

Number of Matched detect with FLANN matcher and KNN selector with k=2					
Descriptor Type → / Detector Type ↓	BRIEF	ORB	FREAK	SIFT	AKAZE
Harris	19	20	15		
BRISK	134	93	108	162	
FAST	39	37	31	44	
ORB	44	52	34	75	
SIFT	26		23	40	
AKAZE					116

Table 2: Number of matches for detector - descriptor pair

Table 2 is presented a line graph in Graph 1 to ease our comparison process.



Graph 1: Number of matches

From the above graph we can easily conclude that the descriptor paired with BRISK detector consistently provides high number of matches. This is also consistent with the

previous result since we can see that BRISK provided higher number of keypoints compared to other detectors.

The following are the best pair of detector-descriptor from high to low

1. **BRISK – SIFT**
2. **BRISK – BRIEF**
3. **BRISK - FREAK**

Task 3: Total time taken for detection and description:

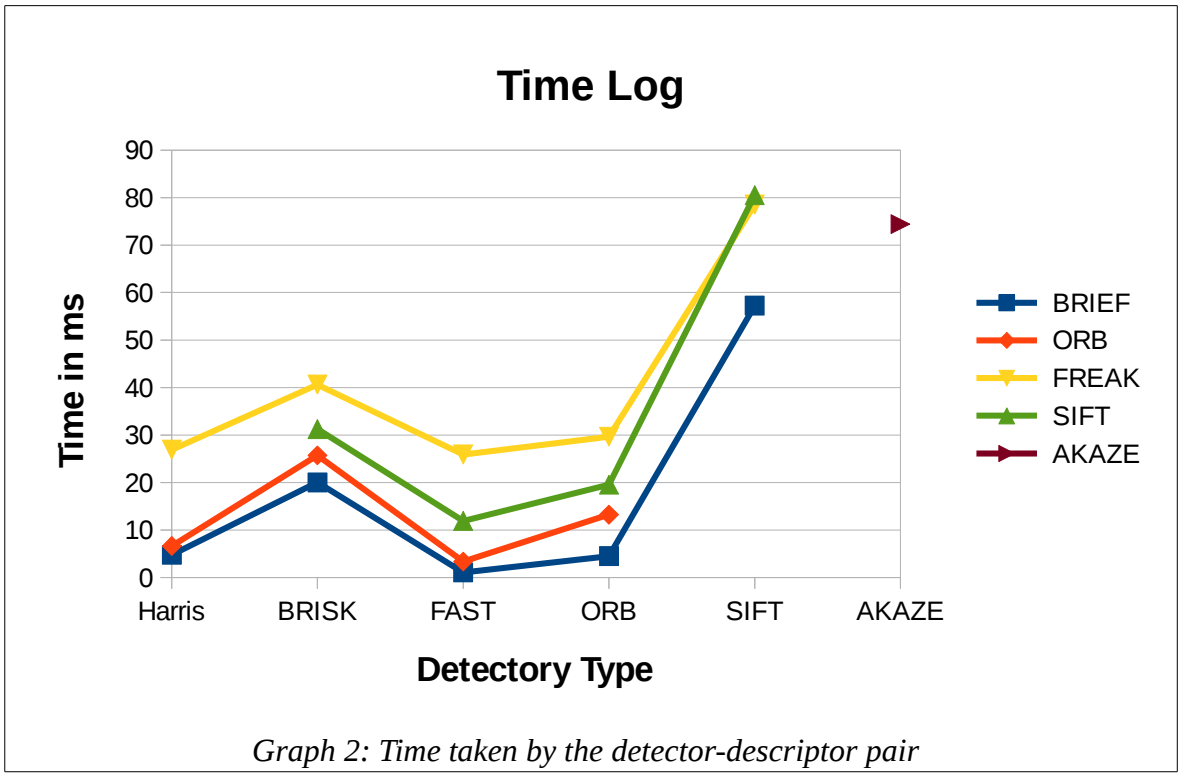
In the final task we were asked to evaluate the overall best in terms of real time application. For application in real time, we need a detector-descriptor pair that will take the least amount of time to perform both detection and description of the keypoints in the image.

Table 3 present the time taken by the detector-descriptor pair

Time log for both detection and descriptor extraction (in ms)					
Descriptor Type → / Detector Type ↓	BRIEF	ORB	FREAK	SIFT	AKAZE
Harris	4.75	6.63	26.9		
BRISK	20.02	25.71	40.62	31.25	
FAST	1.04	3.35	25.9	11.91	
ORB	4.49	13.22	29.64	19.53	
SIFT	57.25		78.55	80.51	
AKAZE					74.42

Table 3: Time taken by the detector-descriptor pair

Table 3 is presented a line graph in Graph 2 to ease our comparison process.



Graph 2: Time taken by the detector-descriptor pair

In OpenCV application there is always a take between the detection time and accuracy of detection. So the decision has to made based on the application and use case in hand. **Here, the application is Time To Collision (TTC) which requires that the computation time be as low as possible.**

Keeping this in mind, referring to Graph 2, we can see that the FAST & ORB detector algorithms provide faster detection of keypoints.

So the top 3 detector-descriptor combination to use for Time To Collision detection, which is the use case in this project are

1. **FAST – BRIEF**
2. **FAST – ORB**
3. **ORB - BRIEF**