Tracking Image Features

Project report

MP.1

In order to optimize the memory usage, a ring buffer is implemented to store only the recent images. A threshold for the size of data-buffer is defined. Until the data-buffer reaches its threshold, images are pushed into it. Then, the oldest image is removed using .erase() method before pushing a new one into it.

MP.2

Implemented HARRIS corner detection to detect keypoints. The keypoints are filtered using the gradient threshold and non-maxima suppression. Additionally, FAST, BRISK, ORB, AKAZE and SIFT detectors are also implemented as the objects of key point detector classes from OpenCV library. The detector is made selectable using a string variable "detectorType".

MP.3

In this project, we intend to build a collision detection system considering only the preceding vehicle for simplicity. Hence only the keypoints of the preceding vehicle region are retained and the rest are removed. This is achieved using the Rect datatype in OpenCV and the presence of a keypoint in the rectangle is verified using the .contains() method of Rect variable.

MP.4

BRIEF, ORB, FREAK, BRISK, AKAZE, SIFT descriptors are implemented in the "descKeypoints" function. The descriptors are made selectable using the string variable descriptorType. The SIFT descriptor is a HOG type descriptor and returns descriptors of datatype CV_32F that are only compliant for FLANN matching. While the AKAZE descriptor works only with AKAZE keypoints.

MP.5

The Brute Force matching that calculated the L2-distance between all the keypoints is already implemented. In addition, the FLANN matching method which is based on the KD-Tree data-structure is implemented. FLANN is a faster method and provides satisfactory results.

MP.6

K-Nearest neighbor algorithm is implemented in order to discard the keypoint matches whose first best and the second best matches have greater than 0.8 distance ratio. Since in the cases where the second match is very close to the first, there is a good chance for the second match to be the right match. Hence in order to reduce the number of false matches, this algorithm is implemented.

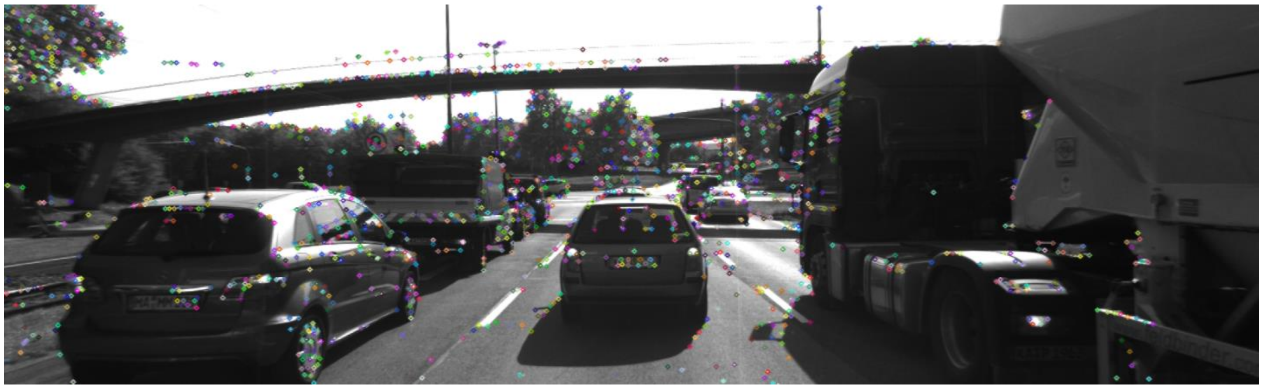
Performance evaluation:

MP. 7

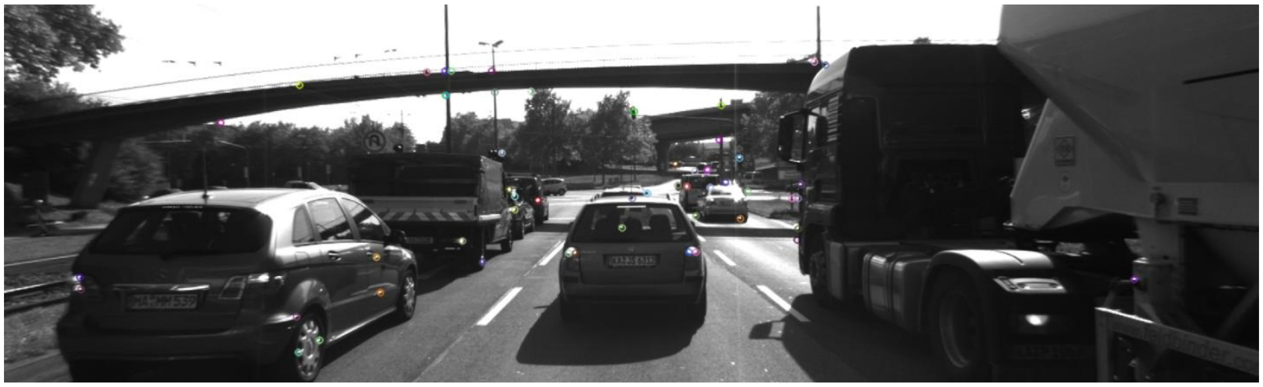
The number of keypoints detected by each detector is counted and logged in **keypoints_count** file. BRISK detects highest number of keypoints while Harris detects least number of keypoints as its keypoints undergo threshold filtering and non-maxima suppression.

Keypoint neighborhoods:

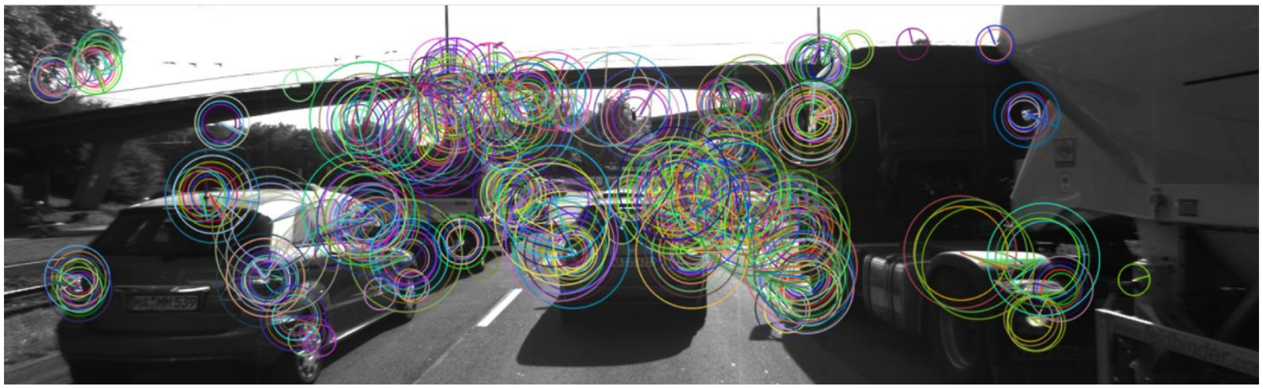
Shitomasi: Keypoints are quite evenly distributed. Many keypoints are detected on the road surface, bridge and trees that are not very useful. The keypoints on the car are majorly found at the corners.



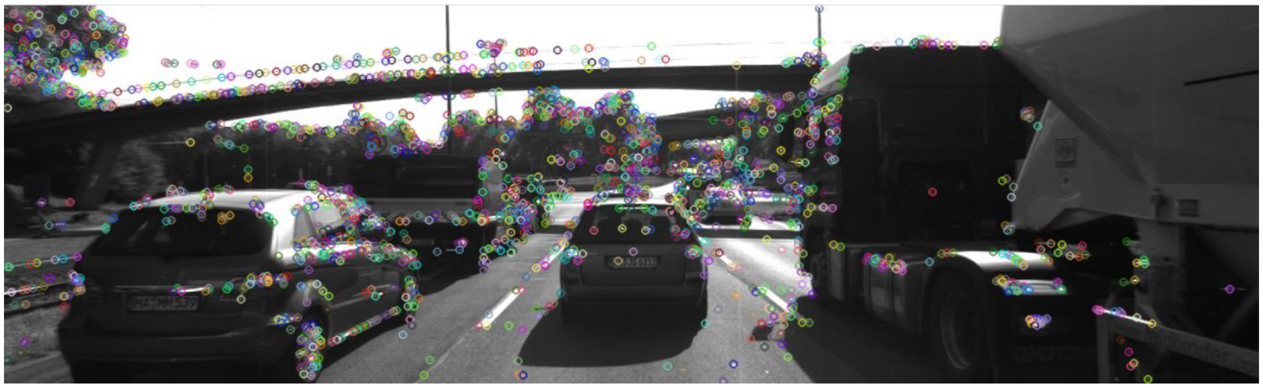
Harris: Though the keypoints are evenly distributed, the density is very sparse. The amount of keypoints are very less to rely on. The detector did not detect any keypoints on the road surface and trees which is a good thing for this project as we are only concerned on the vehicle.



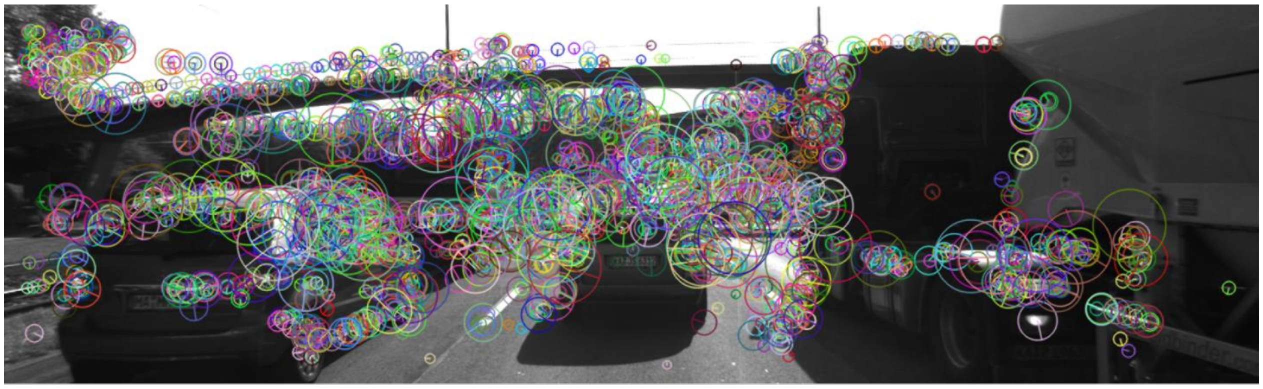
ORB: Large number of keypoints are detected on the vehicles and none on the road surface. The keypoints also indicate the scale and orientation which is a valuable information that helps obtain matches efficiently. The keypoints exist both in the corners and also in the blobs. The keypoints are densely located.



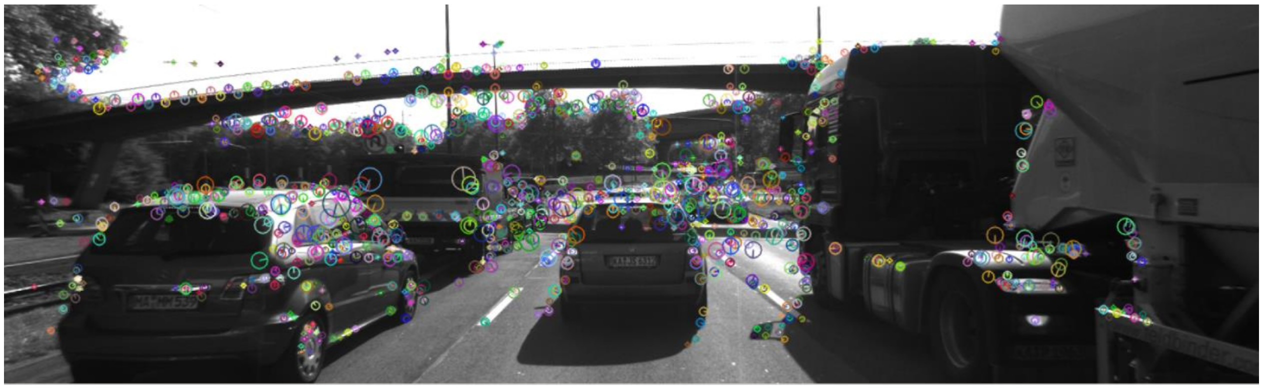
FAST: There are significant number of keypoints on the vehicles and are majorly found on the edges. There are a large number of keypoints on the bridge and the road surface.



BRISK: Largest number of keypoints are detected using BRISK. The keypoints are not evenly distributed. Provides scale and orientation information.



AKAZE: Large number of keypoints are detected on the cars and some also on the road surface, trees and the bridge. Although the keypoints are detected at multiple scales, the scale range is narrow. However, this can be adjusted by tuning the parameters of the detector.



SIFT: Keypoints are not evenly distributed. Decent number of keypoints exist on road surface and trees. The scale range is large making the keypoints highly scale invariant and reliable.

All in all, ORB, BRISK, and SIFT are the best detectors.

MP. 8

All possible combinations of seven detectors and six descriptors are evaluated and the number of matches are logged in the file **matched_keypoints**. BIRSK + BRIEF (det + disc) and AKAZE + BRIEF yielded highest number of matches. BRIEF and ORB descriptors yield high number of matches in general compared to other descriptors.

MP. 9

The processing times to detect keypoints and compute the descriptors for corresponding keypoints are evaluated for all possible combinations. The results are logged in the file **processing_times**. (FAST det, BRIEF desc), (FAST det, ORB desc), (FAST det, BRISK desc) combinations have least processing times. Additionally, ORB and BRISK descriptors and ORB detector has competitive processing times.

Considering the performance evaluation metrics MP.7, MP.8, MP.9 the following are found to be the top-3 combinations:

1. FAST + BRIEF
2. ORB + BRIEF
3. FAST + ORB