Camera Based 2D Feature Tracking

Aim:

Execute feature detection and extraction algorithms on the set of images, Analyse and compare the methods based on its performance like robustness, time efficiency.

Methodology:

The feature tracking method involves two steps

1. Key point detection: Key point is a point in the image which is unique in the local area of the image, and can be matched with the corresponding point in other image
   Various key point detection algorithms are SHI-TOMASI, HARRIS, FAST, BRISK, ORB, AKAZE, SIFT
2. Descriptor extraction :

   Examples of Feature descriptor algorithms are BRISK, BRIEF, ORB, FREAK, AKAZE, SIFT.

   Descriptor extraction can be further categorized into
   1. Gradient based descriptor : Basic idea behind the gradient descriptor is based on distribution of intensity gradient in the local neighbourhood.
      EX: SIFT, SURF
   2. Binary descriptor : Binary descriptor is based solely on the intensity of the image. It encodes the information around the key points in the string of binary numbers.
      EX: BRISK, FAST, BRIEF, AKAZE, ORB

Bugs in OPENCV:

While working on the OPEN CV 4.1.1 version, I came across some bugs in opencv lib

1. AKAZE Library does not work with any other key points detector other than AKAZE itself
2. SIFT key point detection algorithm does not work with ORB descriptor lib.

Performance Analysis:

TASK 1 : Using STL library function, Implemented a queue to add new image, and delete last element if the vector(queue) size is greater than 2

TASK 2 : Implemented different key point detection methods like HARRIS, FAST, BRISK, ORB, AKAZE, and SIFT using open CV libraries and decision to choose the method is decided by the string "detectorType".

TASK 3 : Removed all the key point outside bounding box of the vehicle using open cv::rect class, and used "contains" functionality of cv::rect to find if the point is inside the bounding box.

TASK 4 : Implemented different key point descriptor methods like BRIEF, ORB, FREAK, AKAZE and SIFT using open CV libraries and decision to choose the method is decided by the string "descriptorType".

TASK 5 : Added Key point matching algorithms to Like FLANN and K-Nearest-Neighbor approach using open CV library function.

TASK 6 : Implemented the descriptor distance ratio test as a filtering method to remove bad keypoint matches.

TASK 7 : BRISK has detected the highest number of key points, followed by AKAZE, FAST, SHIFT and SHI-TOMASI

TASK 8 and TASK 9: comparing the number of matched and time taken by the algorithm, my top three combination would be

1. FAST- ORB
2. FAST - BRIEF
3. AKAZE - ORB

This recommendation is because, from the result I observed that it generated a good number of key points in considerably less time.