Camera Based 2D Feature Tracking

1. Data Buffer

MP.1 Data Buffer Optimization

- Implemented a check to vector dataBuffer size does not exceed dataBufferSize (e.g. 2 elements).
- This can be achieved by pushing new elements on end and removing front element
- Solution: Lines 67 to 71 in MidTermProject_Camera_Student.cpp

2. Keypoints

MP.2 Keypoint Detection

- Implement detectors HARRIS, FAST, BRISK, ORB, AKAZE, and SIFT and make them selectable by setting a string accordingly.
- Solution code: Lines 93 to 109 in MidTermProject_Camera_Student.cpp
- Solution code: Lines 150 to 271 in matching2D_Student.cpp

MP.3 Keypoint Removal

- Remove all keypoints outside of a pre-defined rectangle and only use the keypoints within the rectangle for further processing. By using OpenCV Rect and contains method.
- Solution code: Lines 118 to 137 at MidTermProject_Camera_Student.cpp

3. Descriptors

MP.4 Keypoint Descriptors

- Implement descriptors BRIEF, ORB, FREAK, AKAZE and SIFT and make them selectable by setting a string accordingly.
- Solution code: Lines 167 to 201 in MidTermProject_Camera_Student.cpp
- Solution code: In function descKeypoints, Lines 77 to100 in matching2D_Student.cpp

MP.5 Descriptor Matching

- Implement FLANN matching as well as k-nearest neighbor selection.
- Both methods must be selectable using the respective strings in the main function.
- Solution code: In function matchDescriptors, Lines 28 to 61in matching2D_Student.cpp

MP.6 Descriptor Distance Ratio

- Use the K-Nearest-Neighbor matching to implement the descriptor distance ratio test, which looks at the ratio of best vs. second-best match to decide whether to keep an associated pair of keypoints.
- Solution code: Lines 190 in MidTermProject_Camera_Student.cpp
- Solution code: In function matchDescriptors, Lines 45 to 61 in matching2D_Student.cpp

4. Performance

MP.7 Performance Evaluation 1

- Count the number of keypoints on the preceding vehicle for all 10 images and take note of the distribution of their neighborhood size.
- Do this for all the detectors you have implemented.
- Solution Result: Task 7 – Results.csv

HARRIS detector has the lowest number of keypoints. FAST detector has more number of keypoints.

MP.8 Performance Evaluation 2

- Count the number of matched keypoints for all 10 images using all possible combinations of detectors and descriptors.
- In the matching step, the BF approach is used with the descriptor distance ratio set to 0.8.
- Solution Result: Task 8 – Results.csv

MP.9 Performance Evaluation 3

- Log the time it takes for keypoint detection and descriptor extraction.
- Summed up all detector times and descriptor time for all images into Total time taken for considering the recommendation of combination for detector and descriptor.
- Solution Result: Task 9 – Results.csv

Below are my observations from the results spreadsheet,

SHITOMASI and SIFT have more matching keypoints but the total taken for all the images was more.

HARRIS detector with all other Descriptor types had very few keypoints from the preceding car and also taking more time.

FAST detector in combination of ORB, BRIEF & BRISK descriptor has more (around 400) keypoints from preceding car and almost matching 50% with processing times of 37. 102 ms for ORB, 42. 98 ms for BRIEF & 68.26 ms for BRISK for all images.

ORB detector in combination with BRISK, BRIEF, ORB descriptor has around 120 keypoints from preceding car and matching around 45% of keypoints with processing times 92 ms for BRIEF, 120 ms for BRISK, 147 ms for ORB.

All other combination are consuming more amount of time for processing.

AKAZE descriptor is compatible to be used with other detector types.

Considering all my observations, I would rank TOP 3 as below

1. FAST Detector & ORB Descriptor
2. FAST Detector & BRIEF Descriptor
3. FAST Detector & BRISK Descriptor