Explanation about how I addressed each task:

MP.1 Data Buffer Optimization

- As we don't want to keep pushing all the new images into 'dataBuffer', because this can slow down the computer after some time. The idea was to hold a limited number of images in the ring buffer and process them. In the case of constant velocity model, we want to hold 2 images only. Hence my approach was to check the size of the dataBuffer and push the image into it if the size is less than 2. But if the size of the dataBuffer is 2, then erase the image present at index 0 in the vector dataBuffer using 'erase()' function and then push the latest image into dataBuffer using 'pushback()' function.

MP.2 Keypoint Detection

- ShiTomasi HARRIS, FAST, BRISK, ORB, AKAZE, SIFT – all these mentioned keypoint detectors were implemented and one of them can be selected by changing the string 'detectorType'. Selecting one of them invokes the concerned functions which were defined in 'macthing2D_Student.cpp'. ShiTomasi and HARRIS were implemented with the code taught in earlier lessons while FAST, BRISK, ORB, AKAZE, SIFT were implemented using the keypoint detector common interface provided by OpenCV. → 'cv::Ptr<cv::FeatureDetector> detector;'

MP.3 Keypoint Removal

- My first thought was to run a for loop through all keypoints and check if both x and y coordinates of the keypoints are inside pre-defined rectangle. But going through documentation of cv::Rect class and cv::Point class helped me find 2 functions that does this job.
- The functions are: contains() and inside().
- Both these functions were implemented and both of them are working.

MP.4 Keypoint Descriptors

- BRISK, BRIEF, ORB, FREAK, AKAZE and SIFT - all these mentioned descriptors were implemented and one of them can be selected by changing the string 'descriptorType'.
- They were implemented using the Descriptor Extractors common interface provided by OpenCV. → cv::Ptr<cv::DescriptorExtractor> extractor;

MP.5 Descriptor Matching

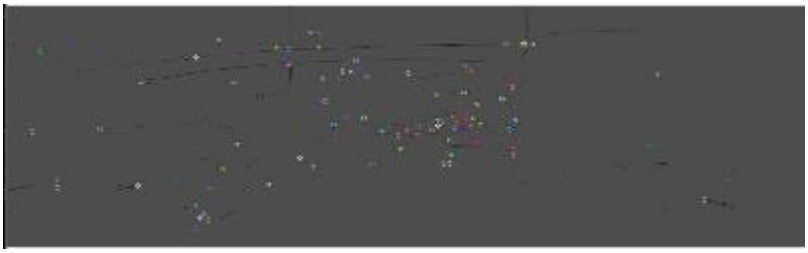
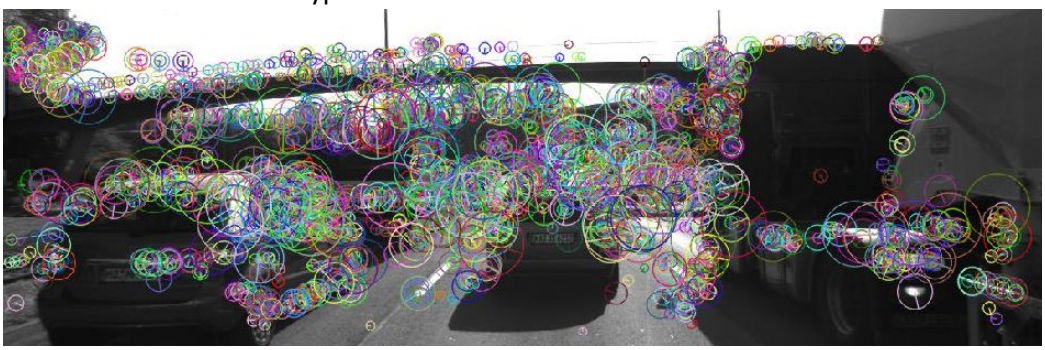
- FLANN matching as well as k-nearest neighbor selection were both implemented as per the code taught in previous lessons. Descriptor matcher common interface provided by OpenCV (cv::Ptr<cv::DescriptorMatcher> matcher;) was used to implement Brute Force and FLANN matching. They could be chosen by changing the strings 'matcherType' and 'selectorType'.
- Also we must change the string 'descriptorType' as per descriptors we are using to match.

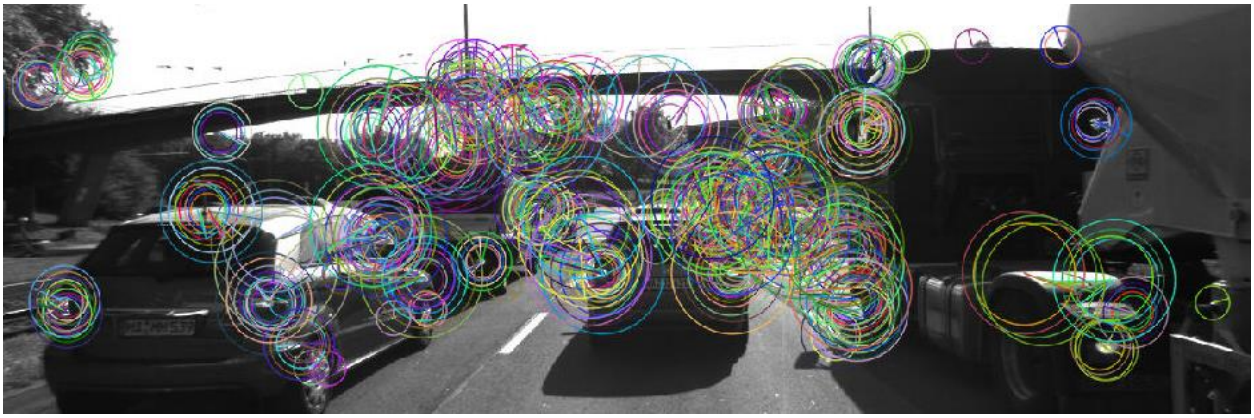
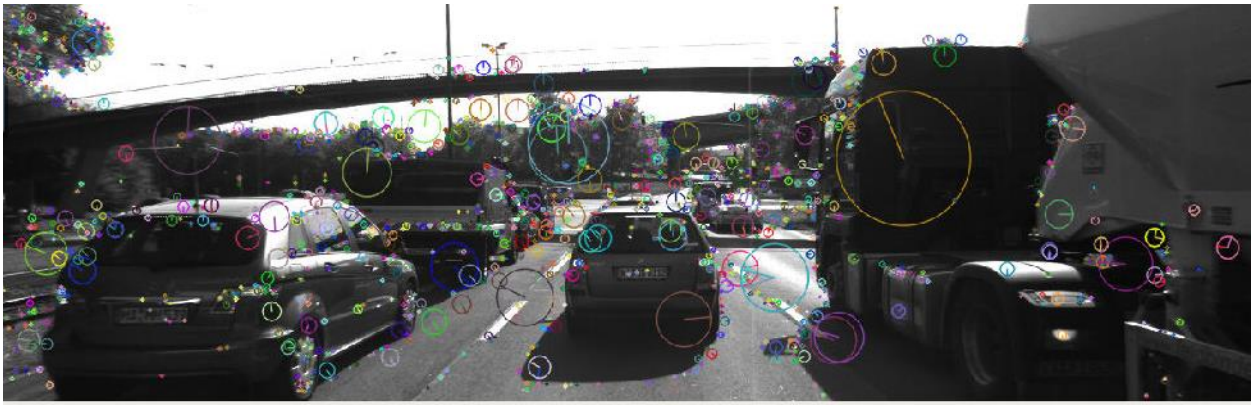
MP.6 Descriptor Distance Ratio

- In the case of K Nearest Neighbor searches, the matches are returned in the distance increasing order. Hence the first match in the vector is the best match. 2 best matches will be taken as K is set to 2.
- A new variable 'vector<vector<cv::DMatch>> knnMatches;' was created for this purpose.

- As the descriptor distance ratio threshold was set to 0.8, we would consider the best match to be not a false positive, if → the distance of best match is less than 0.8 x distance of second best match.

MP.7 Performance Evaluation 1

- Number of keypoints on the preceding vehicle were counted in the code in line 140 in MidTermProject_Camera_Student.cpp file. The number of keypoints would be the size of the 'keypoints vector' and hence is equal to 'keypoints.size()'
- ShiTomasi detector: All the Keypoints have same neightborhood size
- 
- Similarly for HARRIS detector, the keypoints all have same size.
- 
- Similarly for FAST, keypoints all have same size.
- 
- BRISK detector detects keypoints with different sizes
- 

- ORB also detects keypoints with varying size of neighborhoods.
- 
- AKAZE also detects keypoints with varying sizes but the neighborhoods of keypoints here appear to be smaller.
- 
- SIFT also generates keypoints with varying sizes
- 

MP.8 Performance Evaluation 2

- Number of keypoint matches that passed Lowe's descriptor distance ratio test (with threshold = 0.8) were counted in code line 60 in 'matching2D_Student.cpp' file. Keypoint matches for all images will be counted as the iteration happens through all the images.
- In below table, number of keypoint matches for different detector / descriptor combinations for first 2 images in the sequence is logged, this gives us an idea (relative estimation) to compare various keypoint detector / descriptor combinations in terms of the number of keypoint matches.

Keypoint Detectors	Keypoint Descriptors					
	BRISK	BRIEF	ORB	FREAK	AKAZE	SIFT
ShiTomasi	95	115	106	86	--	112
HARRIS	12	14	12	13	--	14
FAST	97	119	118	98	--	118
BRISK	171	178	162	160	--	182
ORB	73	49	67	42	--	67
AKAZE	137	141	131	126	138	134
SIFT	64	86	--	65	--	82

MP.9 Performance Evaluation 3

- Below is time taken by different keypoint detectors and descriptors in milli seconds (ms)
- As the time taken by these algorithms might slightly differ for different pictures, I have mentioned a range of time the algorithm takes for different pictures in the sequence.

Keypoint Detector	Keypoint Descriptor	Detection in ms	Descriptor in ms
ShiTomasi	BRISK	19 to 22	2 to 4
	BRIEF	18 to 23	1 to 4
	ORB	18 to 22	1
	FREAK	20 to 24	50 to 60
	SIFT	18 to 22	25 to 35

Keypoint Detector	Keypoint Descriptor	Detection in ms	Descriptor in ms
HARRIS	BRISK	17 to 22	1
	BRIEF	17 to 21	1
	ORB	17 to 21	<1
	FREAK	17 to 20	50 to 59
	SIFT	17 to 21	16 to 26

Keypoint Detector	Keypoint Descriptor	Detection in ms	Descriptor in ms
FAST	BRISK	1	2 to 3
	BRIEF	1	1 to 2
	ORB	1	1
	FREAK	1	50 to 60
	SIFT	1	30 to 40

Keypoint Detector	Keypoint Descriptor	Detection in ms	Descriptor in ms
BRISK	BRISK	45 to 50	3
	BRIEF	45 to 50	1 to 2
	ORB	45 to 50	5 to 6
	FREAK	45 to 50	50 to 60
	SIFT	45 to 50	65 to 85

Keypoint Detector	Keypoint Descriptor	**Detection in ms**	**Descriptor in ms**
ORB	BRISK	7 to 12	1 to 2
	BRIEF	7 to 12	1
	ORB	7 to 12	5
	FREAK	7 to 12	50 to 60
	SIFT	7 to 12	65 to 80

Keypoint Detector	Keypoint Descriptor	**Detection in ms**	**Descriptor in ms**
AKAZE	BRISK	108 to 120	2
	BRIEF	108 to 120	1
	ORB	108 to 120	3
	FREAK	108 to 120	53
	AKAZE	108 to 120	88 to 103
	SIFT	108 to 120	30 to 40

Keypoint Detector	Keypoint Descriptor	**Detection in ms**	**Descriptor in ms**
SIFT	BRISK	160 to 170	1 to 2
	BRIEF	160 to 170	1 to 3
	ORB	--	--
	FREAK	160 to 170	50
	SIFT	160 to 170	90 to 130

- Based on the criteria of speed and number of matched keypoints, the TOP 3 detector / descriptor combinations I suggest are:
    1. FAST (detector) / BRISK (descriptor)
    2. FAST (detector) / BRIEF (descriptor)
    3. FAST (detector) / ORB (descriptor)
- Because these combinations take least amount of time (high speed) for the whole process and also result in relatively good number of keypoint matches after Lowe's descriptor distance ratio test when compared with other detector / descriptor combinations.