MP.1 Data Buffer Optimization

- Inside the while loop of reading image files, after pushing back the new frame to the buffer (vector), I check the new size of the buffer.
    - If it exceeds the maximum size, I remove the 1st element.
    - Since we check from the start, so we always need to remove one element only which is at “begin()”.

MP.2 Keypoint Detection

- I used the function “detKeypointsModern” to implement all the needed types of detectors.
- Sequence:
    - Record the current time
    - Check the needed detector type
    - Call the corresponding function to detect the keypoints
    - Calculate the spent time.
    - Do visualization if necessary.
- Detectors:
    - SHITOMASI: Already implemented. I adapted the function by removing the time measurement as well as visualization.
    - HARRIS: Similar to the previous exercise with NMS.
    - Rest: Create detector object from OpenCV, then call the detect function.

MP.3 Keypoint Removal

- I used the contain function from the cv::Rect.
- By looping on all keypoints and removing all keypoints that are not in the rectangle.
- Since removing from the vector changes the iterator, I used “while” instead of “for”.

MP.4 Keypoint Descriptors

- Inside “descKeypoints”, I created the corresponding object from OpenCV and assigned it to the pointer “extractor”.

MP.5 Descriptor Matching

- Brute Force
    - Set the value of descriptor type to either binary or HOG based on the type.
    - Used “Hamming Distance” for binary descriptors while L2 for HOG.
- FLANN
    - Converted the type of Source and Ref to CV_32F
- KNN
    - I called the function “knnMatch” instead of “match”.

MP.6 Descriptor Distance Ratio

- Loop on all matches and calculate the distance ratio, push the match only if less than the threshold of 0.8

MP.7 Performance Evaluation 1

- For MP.7, 8, 9 , I renamed main into “doMain” passing all the types of detector, descriptor, matcher, selector as input arguments.
- In the new main, I created vectors and loops to go through all the needed combinations.
- In the “doMain” function, I print the number of points in csv format to the standard error. From the terminal, I run the exe and redirect the standard error to a CSV file.
- Please see the output data in attached file: MP.7-8-9.xls” in tab “MP.7”
- Please see the observation about the neighborhood of the detected keypoints in the attached file “MP.7-9.pdf”.

MP.8 Performance Evaluation 2

- In the “doMain” function, I print the number of matched points in csv format to the standard error. From the terminal, I run the exe and redirect the standard error to a CSV file.
- Please see the output data in attached file: MP.7-8-9.xls” in tab “MP.8-9”

MP.9 Performance Evaluation 3

- To measure the time, I modified the detector function and descriptor functions to return the spent time as “float”.
- In the “doMain” function, I print the time values in csv format to the standard error. From the terminal, I run the exe and redirect the standard error to a CSV file.
- Please see the output data in attached file: MP.7-8-9.xls” in tab “MP.8-9”.
- Please see the selection of the best 3 detector/descriptor in the attached file “MP.7-9.pdf”.