1. The circular databuffer is implemented by erasing the elements at the beginning of the buffer and inserting new elements to the end of the buffer in a FIFO format. The STL API, *buffer.begin()* returns an iterator to the beginning of the buffer, *buffer.erase(iterator)* allows to erase or remove an element of the buffer pointed by the iterator, and all the other elements of the buffer are shifted to account for the deleted element, *buffer.push_back()* allows to push the element to the end of the buffer. Every time the size of the buffer becomes greater than the required size (using the *buffer.size()* API) we erase elements at the beginning of the buffer to ensure that the size always stays within the required bound.
2. Three different functions were implemented for the keypoint detector. One function for shi-tomashi detector, one function for the Harris keypoint detector, and finally one last function to select a modern keypoint detector. The Harris Keypoint detector includes a **non-maximum suppression** (NMS) criteria on top of the keypoint detection to ensure points are spread out and not clustered around regions of the image with high-contrast or edges. For modern keypoint detectors we have FAST, BRISK, ORB, AKAZE and SIFT to select. All these types requires the same interface namely the *create()* and *detect()* interface provided by *Feature2D* class.
3. For removing keypoints that are outside the range, I used the *cv::Rect()* class that allows to specify a region (here the vehicle that we want to track) and also the method of this object called *rect.contains(Point)* that can be used to discard keypoints that are outside the rectangle. We loop over each keypoints detected and check whether the point lies within or outside the Rectangular region specified earlier.
4. For the descriptor, we have a single function that takes in the keypoints and the descriptor type and uses the opencv method *compute()* to find the descriptor for each of the keypoints. The descriptors supported are BRIEF, ORB, FREAK, AKAZE and SIFT and returned in the opencv Mat datatype where each row corresponds to a descriptor. SIFT descriptor consists of a 128 dimensional vector where each vector represents the bin value for histogram of orientation.All other descriptors namely BRIEF, ORB, FREAK and AKAZE are binary descriptors.
5. For matching between the source and the destination descriptor two options are allowed namely brute force (BF) search, or fast library for approximate nearest neighbour search (FLANN). If the descriptor is binary (such as BRIEF, ORB, FREAK, AKAZE) we use hamming distance to find the distance between two descriptors or L2-norm for non-binary descriptors (SIFT).
6. For each of the matching methods we can either just return 1 best match or 2 best match (K=2). With two matches, we can reject descriptors where the distance for the first and second is very close. This helps to avoid poor descriptors and keypoints that can be wrongly matched.

7. After implementing all the descriptors and detectors we analyzed the number of keypoints generated by various methods. Each descriptor has various parameters that can be used to tweak the number of keypoints that can be generated. We would like to have almost the same number of keypoints for all the methods to make the computation time manageable. If the parameters are set such that the threshold is high, we may only generate very few descriptors. In contrast if we have the parameter set too weak we may generate lots of descriptors that can make the matching process very time consuming. The goal is to measure the following properties of these descriptors:
    a. Size - small or large radius
    b. Distribution - Is the descriptor uniform or clustered around few regions
    c. Overlap - Are the keypoints spread out so there is not too much overlap between them or it has lots of overlap.

Keypoint method	Properties of the features
Harris	small, localized but keypoint generated various drastically
ShiTomashi	nicely distributed around the car including the license plate. Less overlap.
BRISK	Large aperture and large number of keypoints with lots of overlap. All keypoints have similar scale.
FAST	Uniformly distributed, similar sized keypoint, Does not detect license plate. Less overlap
AKAZE	Does not detect license plate. Small aperture size and clusters on the edge and not uniformly covered
ORB	Large aperture and large scale and not evenly spread on the target. Most of the scale is huge with lots of overlap between other keypoints.
SIFT	Normal aperture and nicely spread on the target and uniformly distributed. Detects keypoints of various scales.

8. All combinations of the keypoints and descriptors were evaluated to find the number of matches and quantity of keypoints generated. It was difficult to identify and measure the quality of match in different descriptors. So it is not entirely clear which descriptor has the largest discrimination capability. It was possible only to visually identify few of the matches and ensuring all matches are accurate is a hard task. Most of the descriptors can be successfully matched from one frame to the next frame according to the table.
9. For measurement to be consistent, I divided the total time taken for keypoint generation by the number of keypoints to get the time required per key point generation. This metric is standard for all different keypoints. Similarly, for descriptor extraction we divide the total time taken for

the descriptor extraction stage by the number of descriptors to derive time required for generating 1 descriptor.

Keypoint name	Avg. time taken (ms / keypoint)
FAST	**0.000695419**
Shi-Tomashi	**0.008198086**
ORB	**0.01849034**
HARRIS	**0.04323768**
SIFT	**0.06136165**
BRISK	**0.1289466**
KAZE	**0.1636648**

Descriptor name	Avg. time taken (ms / keypoint)
BRIEF	**0.00916010142**
ORBi	**0.0676237**
SIFT	**0.210542185**
FREAK	**0.4597175714**
AKAZE	**0.53891071**