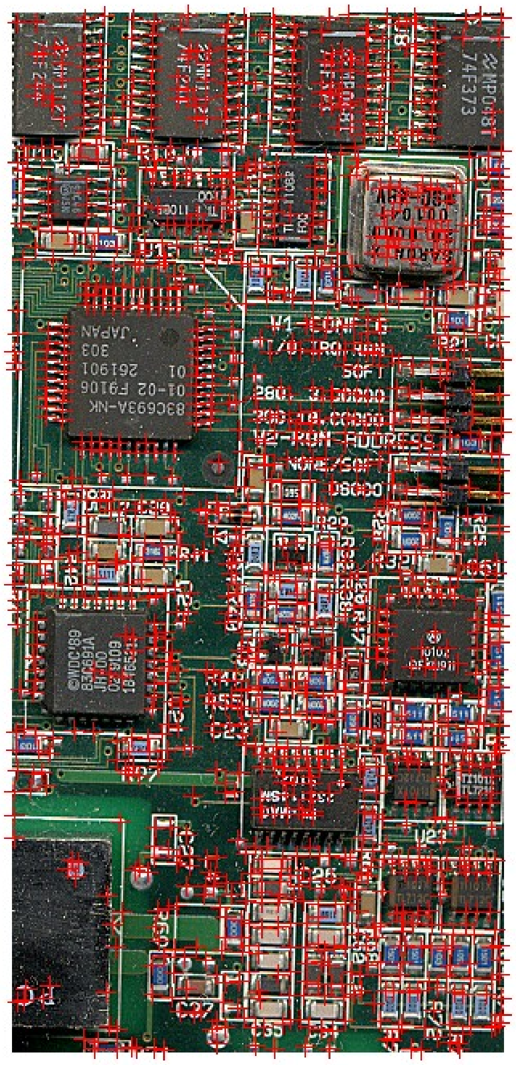
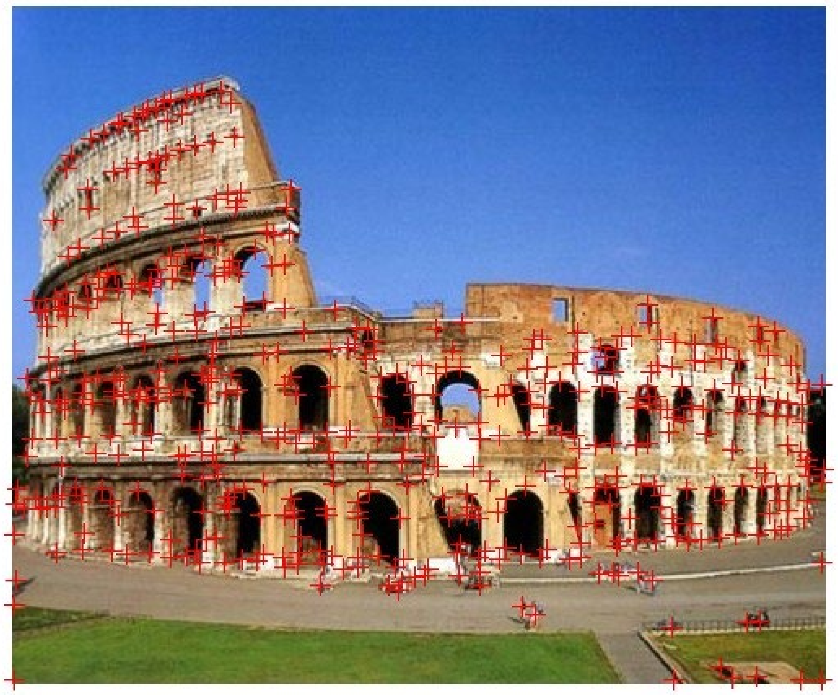
Lab04: Feature Detection

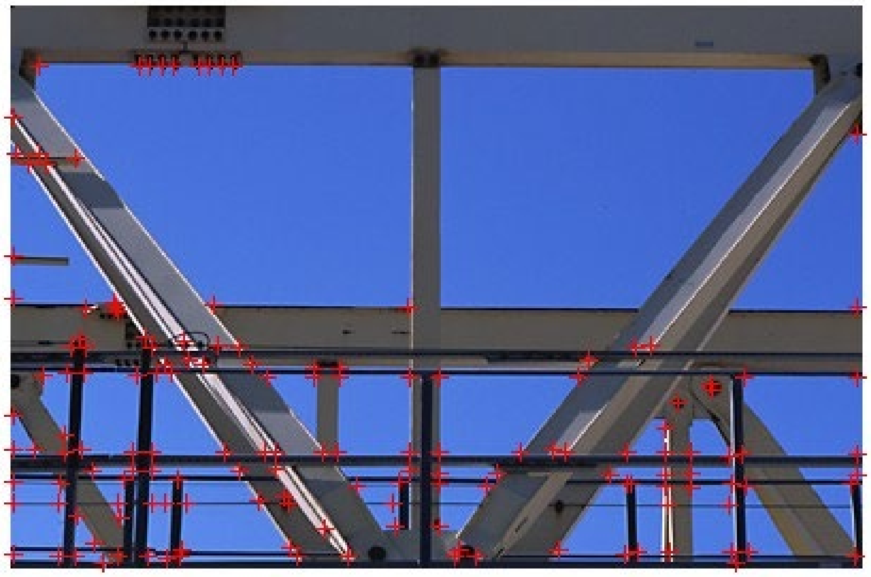
Oct 2022

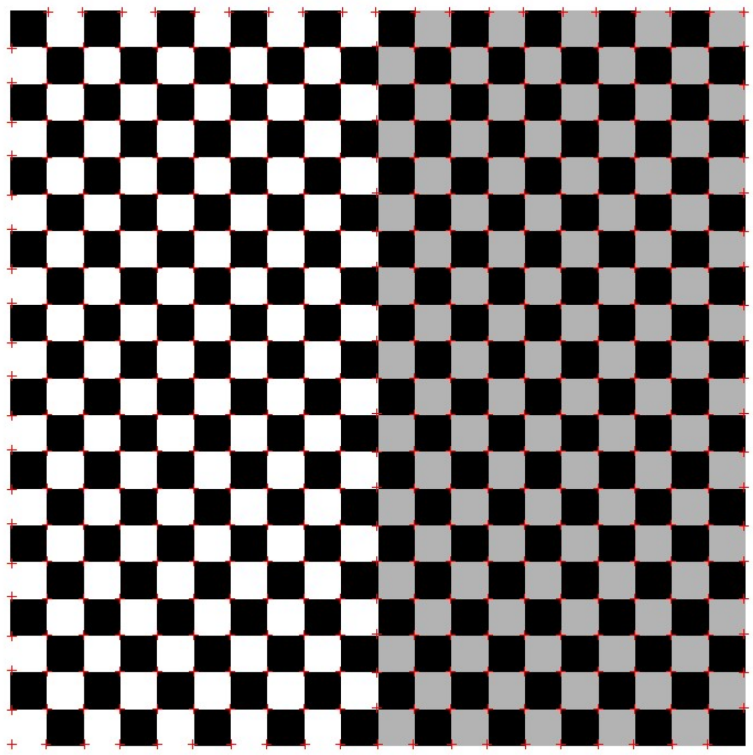
Problem 1

The corners output of three images and checkboard are as follows:







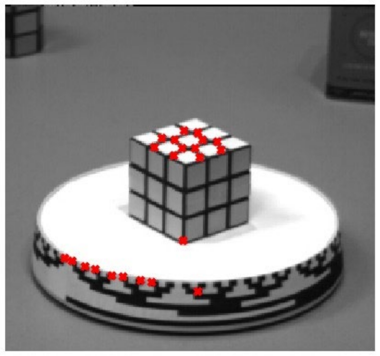
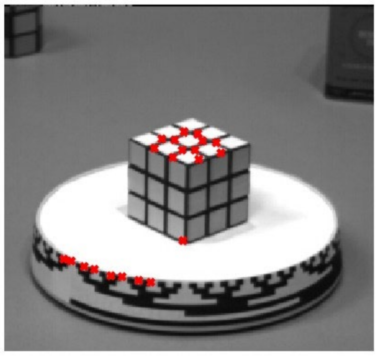


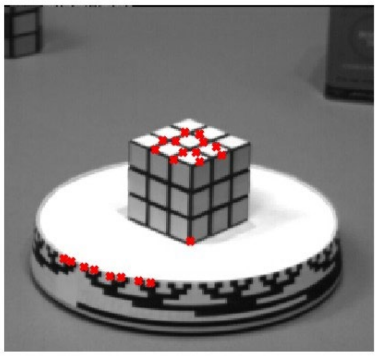
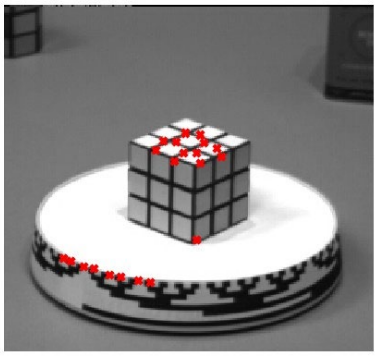
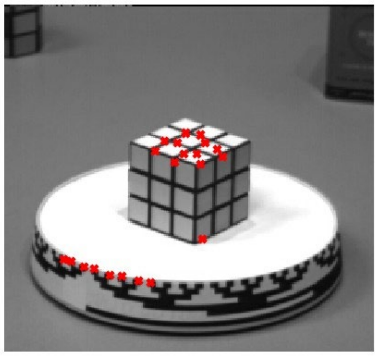
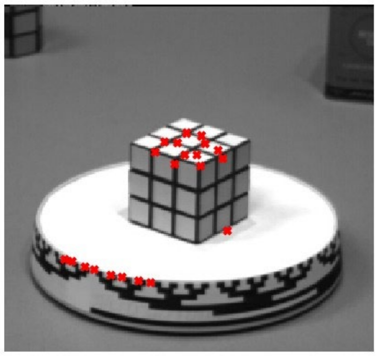
The corners are marked with red '+'.

Problem 2

The KLT uses corners_dectetor.m in P1, and its parameter is sigma1=0.7; sigma2=2.0. And I chose M=25.

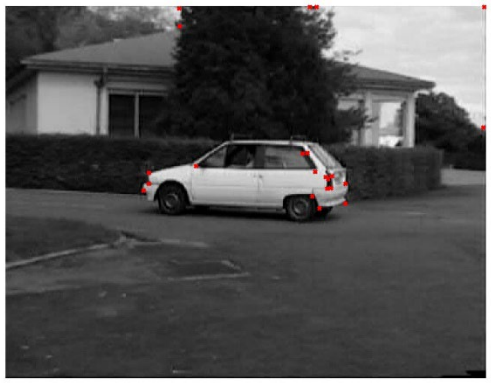
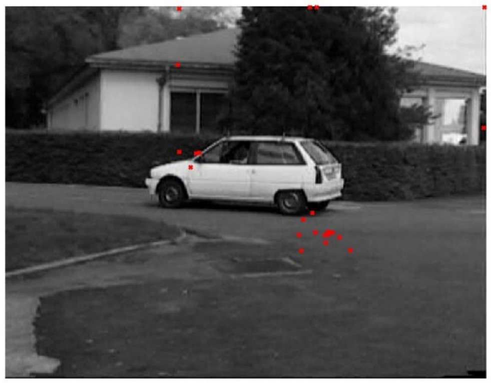
The simple KLT tracker performs fine with **Rubix Cube** input, and the tracked corners with m=10 are as follows:

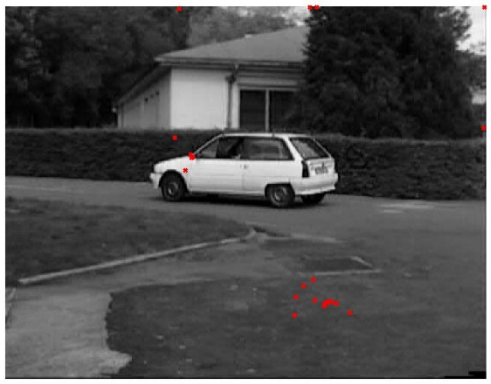
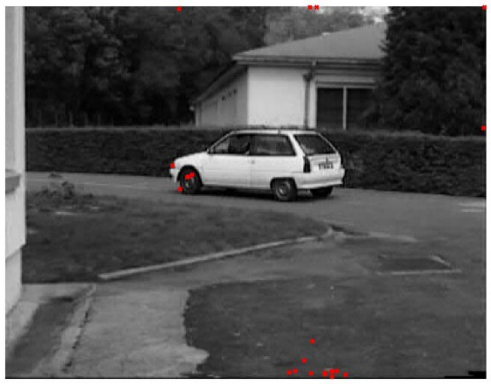
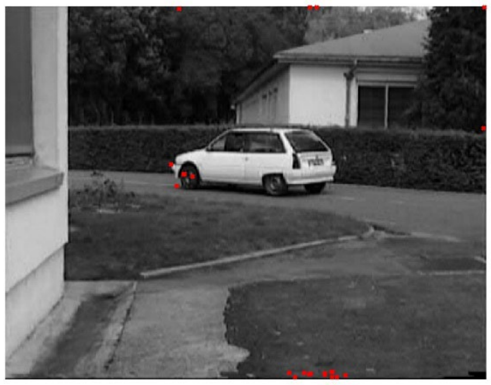
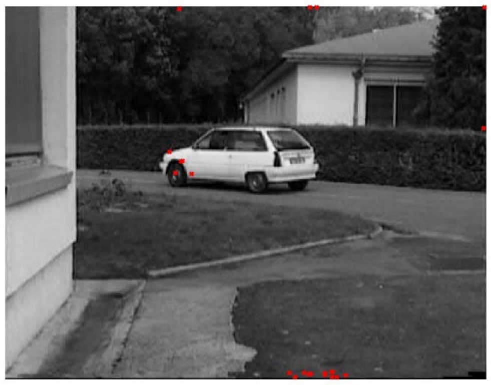



It works fine with Rubix dataset, while the corners still drift in the process, and m=10 works better than smaller m.

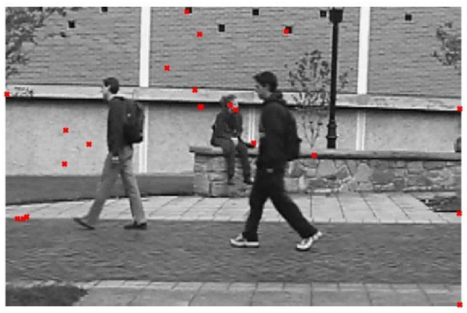
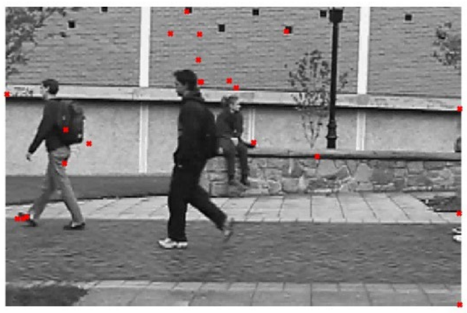
The tracked corners of **car sequence** with m=10 are as follows:

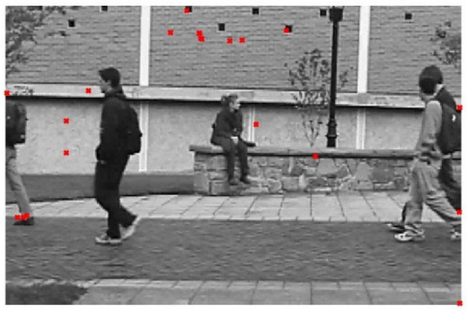
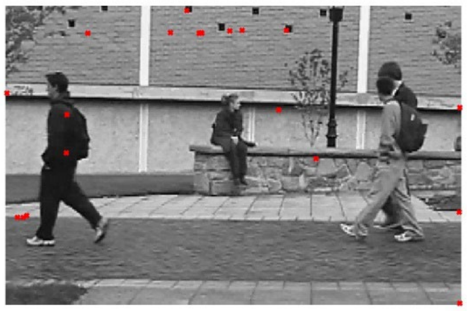



The performance is not good, the eventual corners that are valid are those on the tyers, and the others just go lost during the process.

The tracked corners of **pedestrian walking** with m=30 are as follows:

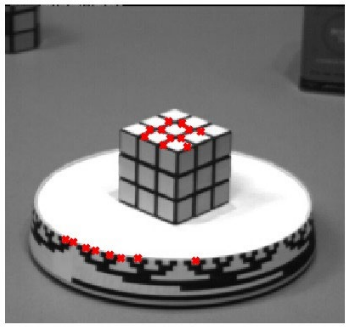
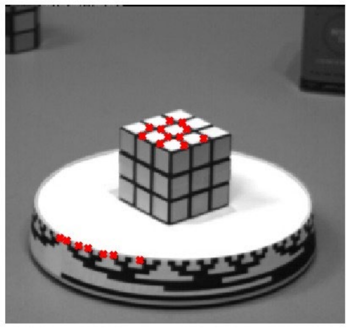
The performance is awful, and I tried different m, and it improves merely. I think the dataset is complicated for simple KLT algorithm, or my code has flaw.

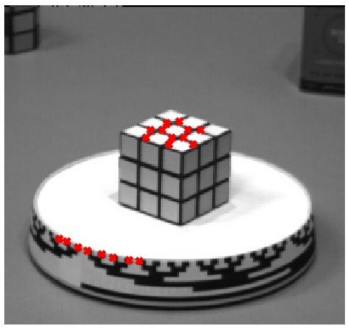
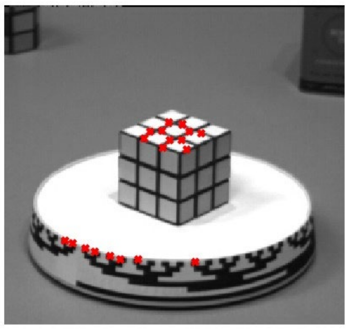
The videos that record the tracking process is stored in the corresponding folders, please check the video if needed.

I tried to solve the **Challenge 1** by updating the corners_tmp as input when the number of valid tracked_corners is less than a threshold (18-20 for example), it turns out that it works several times within a process period, and it helps the tracked corners to grab the corners back.

It's obviously takes more time to process since corners detector is applied more, while it improves the performance in some ways, because when the dataset isn't changing very rapidly, then strong corners in different images are highly corresponding, then with the update by direct corner detection, the tracked corners are more accurate. The result are as follows: I selected pic 0,4,9,13,16 as example.

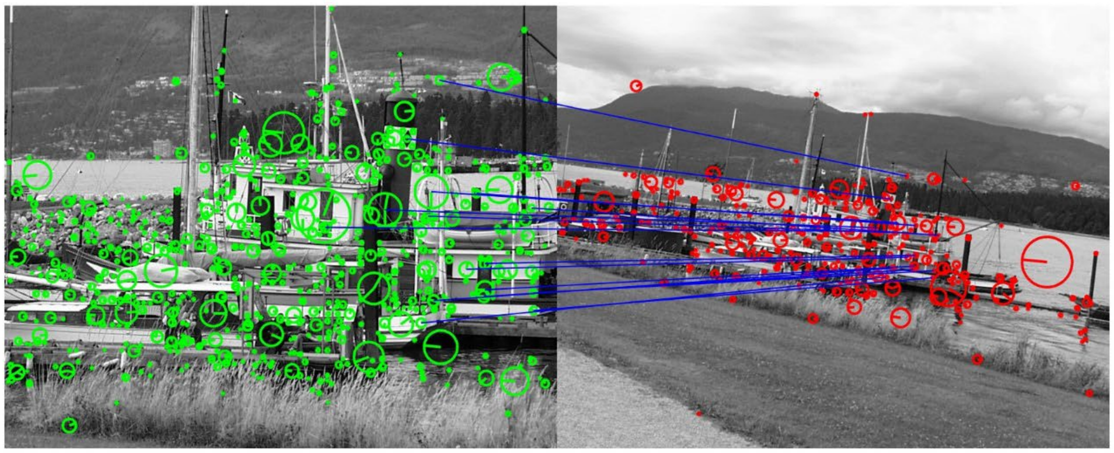
The video is more direct to show the process.

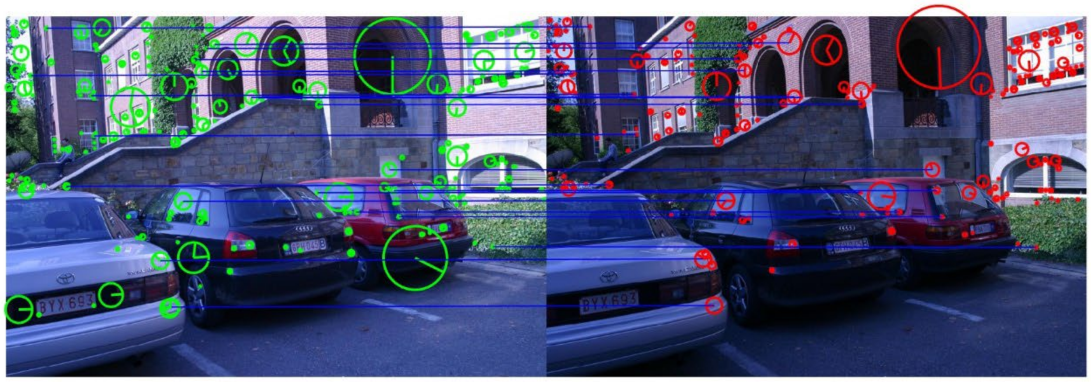



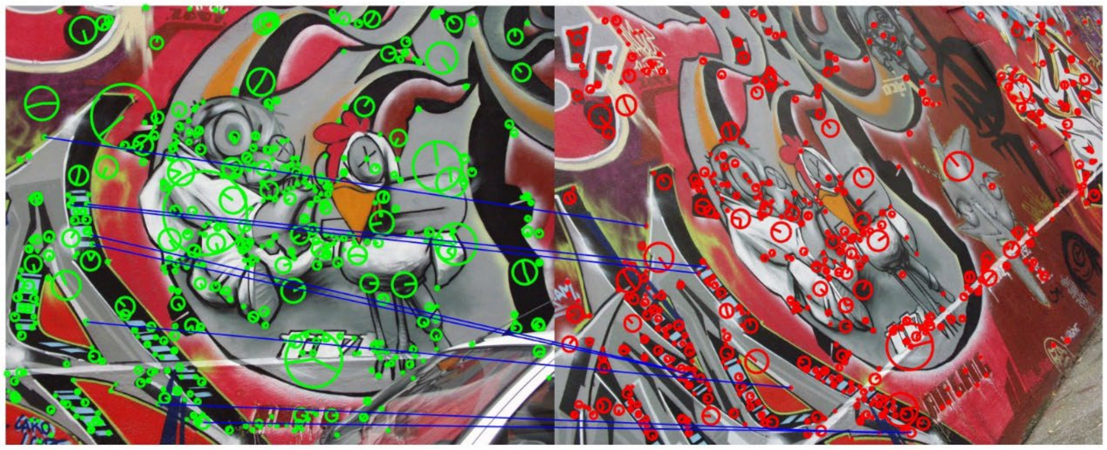



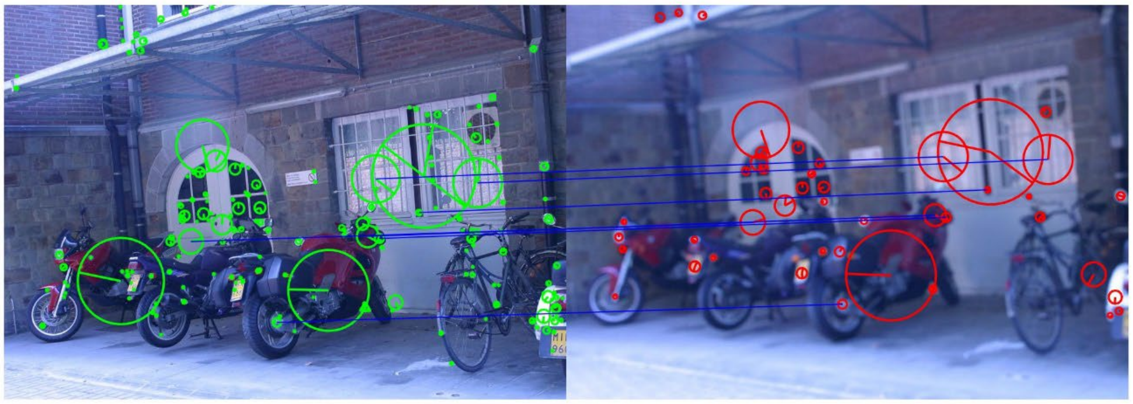
Problem 3

The output images with SIFT features are as follows:









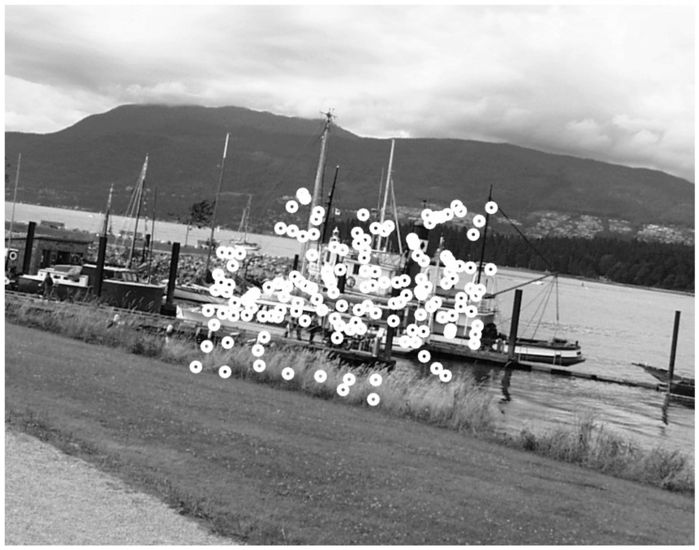
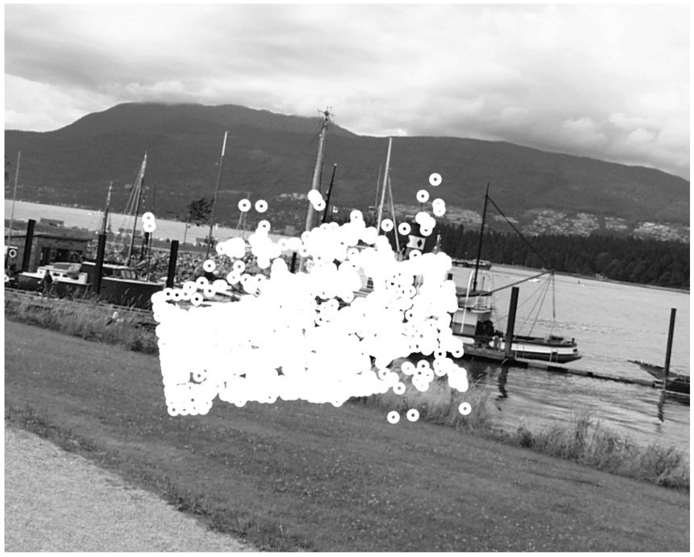
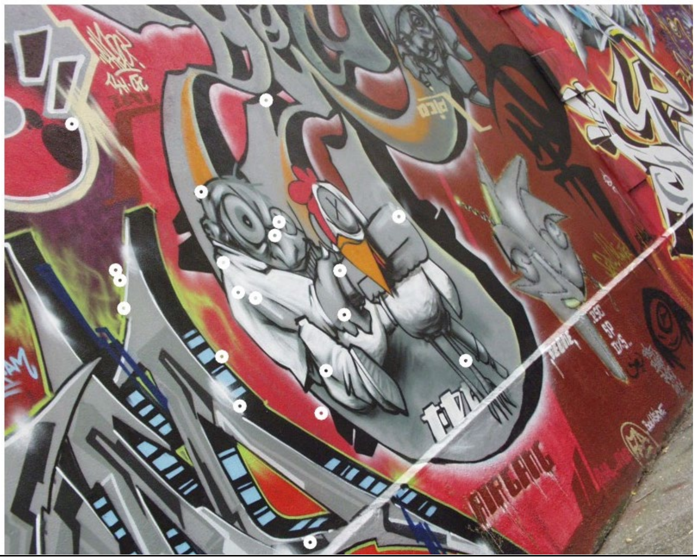
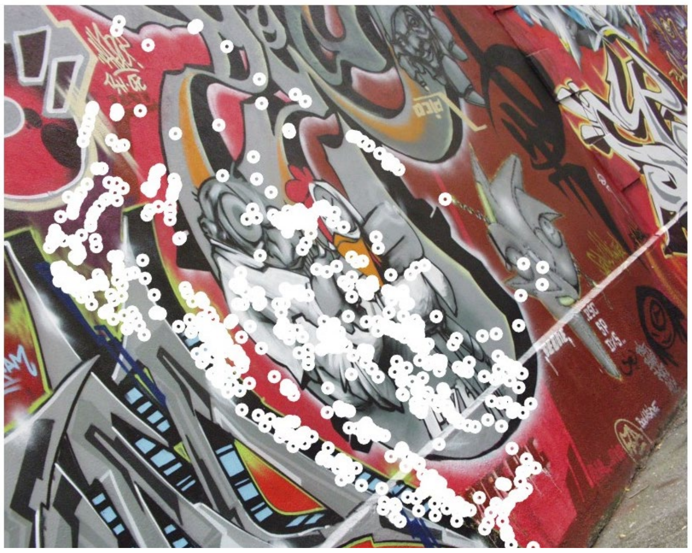
The above images are all processed under peakthresh=8 edge_thresh=5.

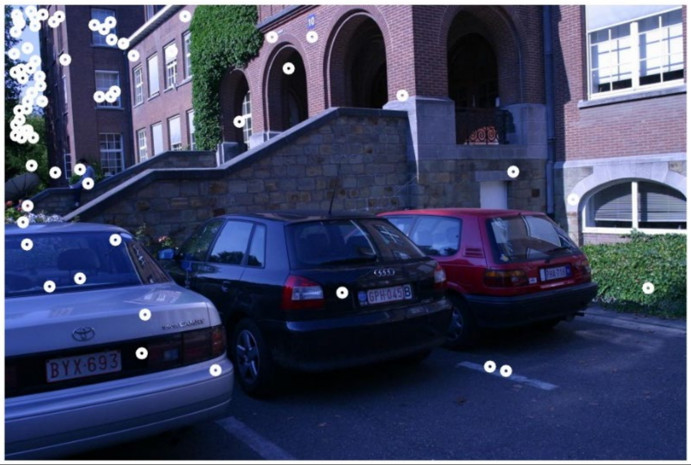
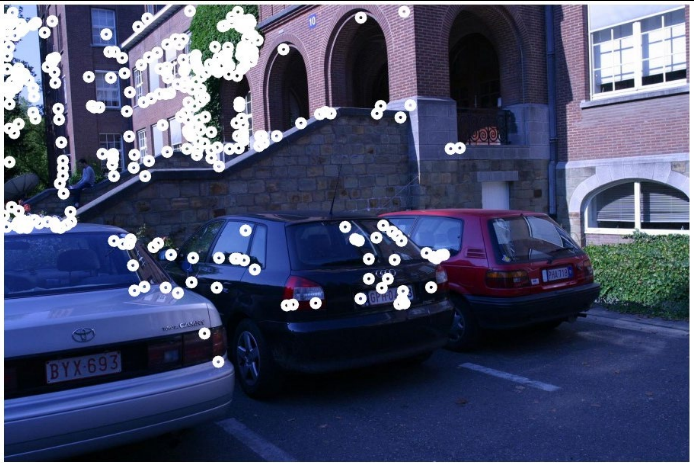
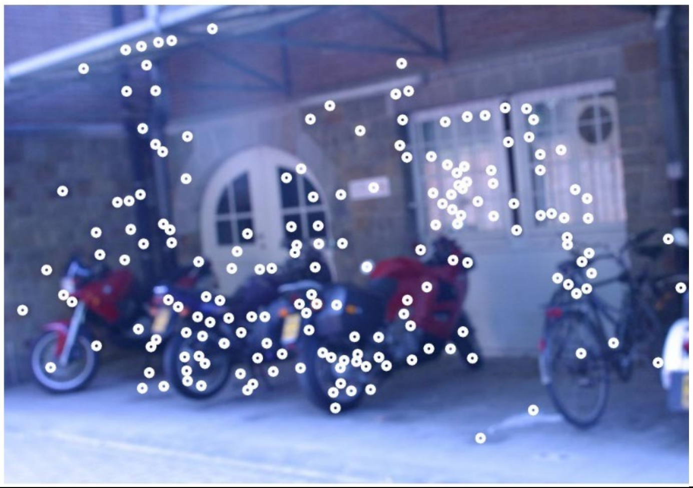
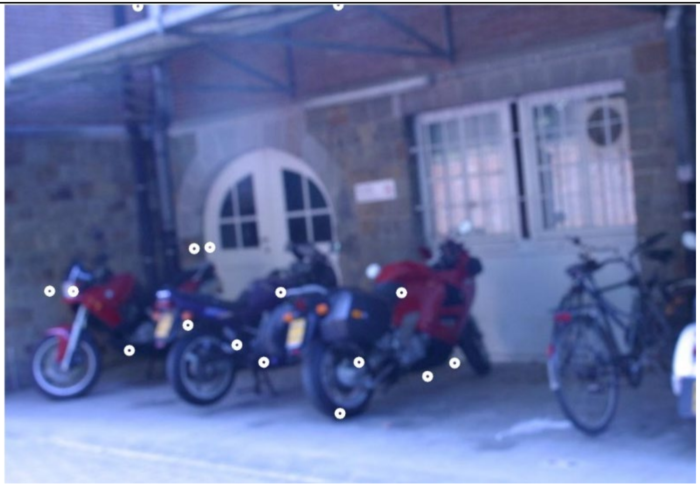
These two parameters decide how many and what features will be recognized.

I matched the feature in two images with vl_ubcmatch() function, and plot top 15% of the corresponding features through line().

Problem 4

The result and corresponding repeatability rate is shown in the form:

	Left	Right
1	0.7020	0.9721
		
2	0.1875	0.8493
		
3	0.1091	0.5045

		
4	**0.4941**	**0.0343**
		

Base on this lab, I think the corner is better due to its higher repeatability in three pictures, but it is not absolute. The corner detection works fine in sharp images, because it can easily detect the corner, but in the set4, the second image is blurred, then it will be very hard for the algorithm to find correspondence features. The SIFT has rather more stable performance because it doesn't only depend on the gradient of image, thus the blur images can be processed as well.

I read a paper which compares corners and SIFT in feature matching, and it says that SIFT features take less time to achieve better performance in correctness and robustness, though this lab isn't about matching, but the corners isn't necessarily better than SIFT.

Problem 5

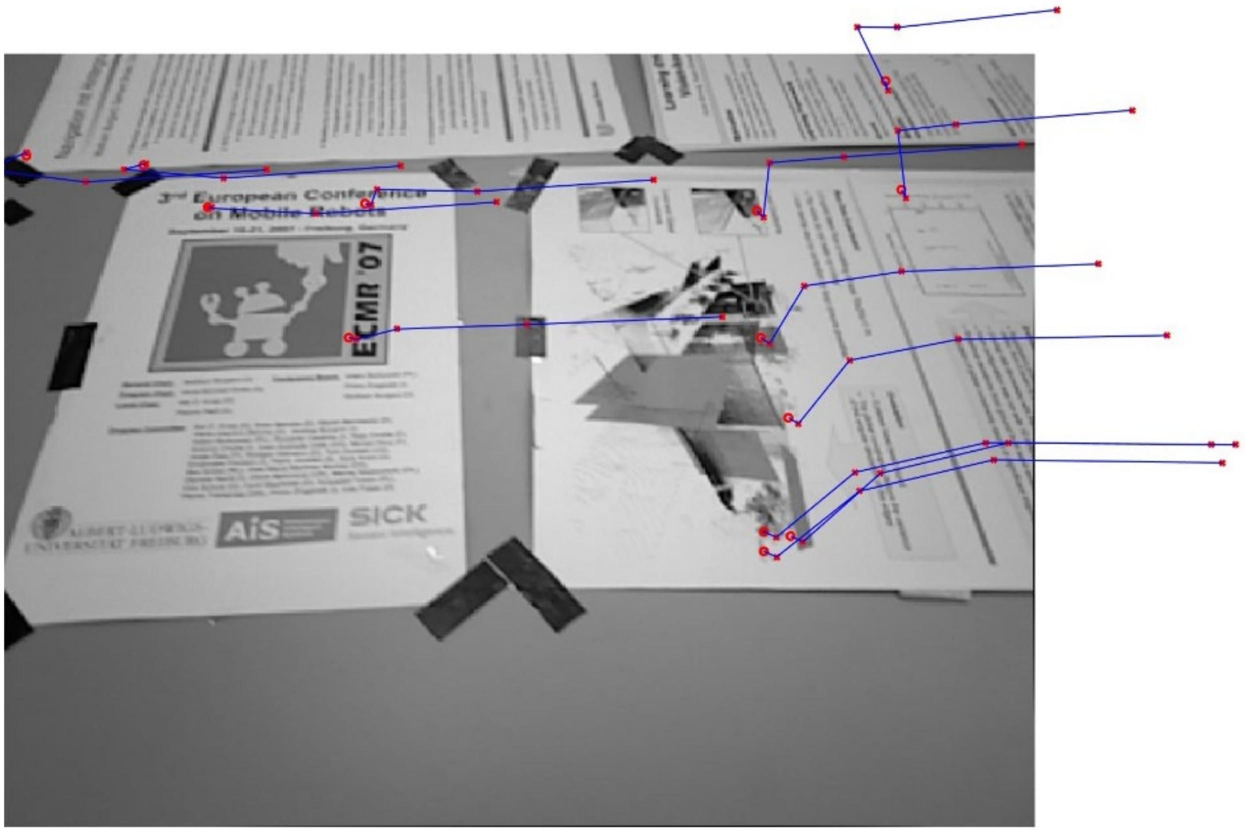
I wrote find_correspondences.m and `find_correspondences_corners.m to execute each update and find correspondence features.`

As shown in the images, I labelled circle as the features selected in the first frame, and 'x' show their position afterwards, and the trajectories are shown by blue lines. Basically the trace shows that the camera is moving left with a bit swirl towards the papers.

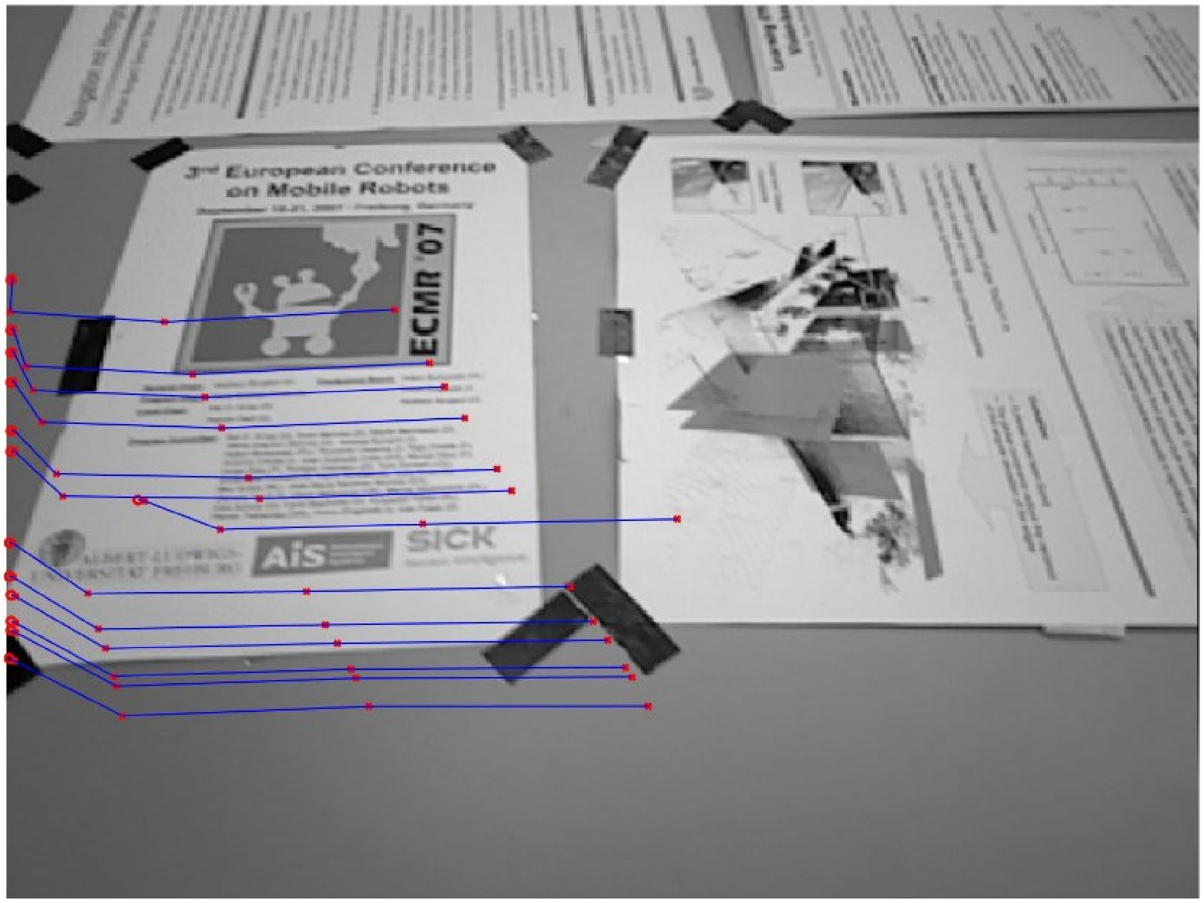
The SIFT features are easier to survive from all five images, when threshold of both methods are set to 10, SIFT survival rate=37/335=11.04%; and corners survival rate=3/315=0.95%. When the number of traces is all between 10 to 20, then SIFT using smaller threshold also proves that the SIFT features are easier to survive.

The question about what features generate more feature tracks is ambiguous, since the parameters of two methods are different. They depend on `vl_sift() and corners detector(), in this case, the SIFT finds a little bit more features than corner.`

`The tracks of two methods are shown as below:`



SIFT



Corners