**Machine Intelligence 2 Assignment 2**

**Question 1**

**B-** Before scaling the image the Harris Corner detection it detected a dense amount of corners, which did include a big amount false positives. I also noticed that it missed some corners. After scaling the image by a factor of 1.5. The amount of false positives did decrease, and some corners that it has missed at the normal scale were detected. But others not so much. I also noticed that it missed some previously detected corners

**C-** I assumed since I have the parameter to filter by circularity as false it would have a better chance of detecting square areas, but it was very inconsistent in comparison to the Harris Corner Detection. Although after scaling the image the Blob detection (using DoG) seemed to have gotten better at detecting the smaller square areas of the image. A lot of the detections seem out of my place, to my eyes at least, so I would fine tune some of the parameter values in order to make it the algorithm detects core consistently and accurately.

Between the two detection systems, the blob system seemed to have gotten better once the image was scaled, while the Harris Corner detection got better at removing some of the false positives, while worse at detecting the details it got before the image was scaled.

**Question 2**

|  |  |
| --- | --- |
| A group of buildings with cars parked on the side of the road  Description automatically generated | A building with cars parked in front of it  Description automatically generated |
|  |  |

A group of buildings with a blue sky

Description automatically generated



A car parked in front of a building

Description automatically generated

A car parked on the side of a road

Description automatically generated

When using the two codes, the one with blending worked much better than the one without. As an example here are two of the pictures that were stitched together once using the code with blending and once without.

After trying several different approaches the fixing the stitching of the first code (the one without blending) the first was trying to decrease the value of ratio test threshold, after that had 0 effect. I started messing with homography, due to thinking that maybe the estimation of the code was wrong, but that also had no effect. I tried applying a filter to find outliers by trying to once again match points to it’s correspondences after the homography operation took place and then perform a second homography but that also didn’t work. (Median Output below)



It consistently gave the same output, nothing changed in all cases. The only conclusion I can think of is that the blending played a huge role in making the stitching more seamless.   
**(The pdf I submitted only included the median, since everything became much more cluttered as I was testing stuff out and I kept making new notebooks. I just thought this code was one of the more interesting things I have tried out that had 0 effect on the output)**

**Question 3**

A problem I consistently ran into when trying this code was finding the right region of interest values. After struggling for a while, I thought perhaps rather than triangle, I needed a rhombus shape. Except that messed it up more if anything. After a rather long time of being finicky with the values I found the right numbers, that detected the lane, while avoiding most outliers.

The video I used is in Bahrain, where a lot of white cars are used and the shadows are particularly strong. The threshold as well as the min and max length of a line needed to be changed as well in order to properly detect the lines while avoiding detection of any cars. One issue I noticed when passing by a lamp post the shadow of the lamp post would only sometimes be detected as a lane while other times not.

A purple chart with numbers

Description automatically generated

A road with cars on it

Description automatically generated



The first two images is after the region of interest was done using a frame that I extracted from the initial video. The third image is the algorithm in action.

Before finding the right values, it would operate correctly until the car on the left would show up. It would assume that due to its white colour and straight white edge that it’s another lane that would then be detected. Now it ignores it and operates properly.



The threshold, min and max line values that needed to be changed, at first it wouldn’t detect the white dividers, I thought it might’ve been because the spaces in between each dot had the algorithm determine that they were not lanes. So I had to decrease the minimum distance to ensure their detection which ultimately helped and led to their detection.

**(The last page of the pdf for this question includes the code I used to extract the frames)**