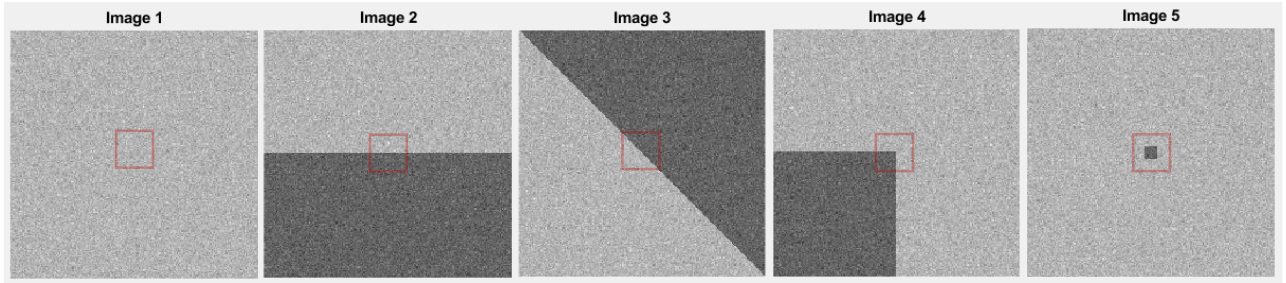


CS4243 Quiz 3

Solutions given in red boxes. Dashed red boxes are auto-graded by LumiNUS. Solid red boxes graded manually by TA.

Q1: Eigenvalues of Second Moment Matrix (8 pts)

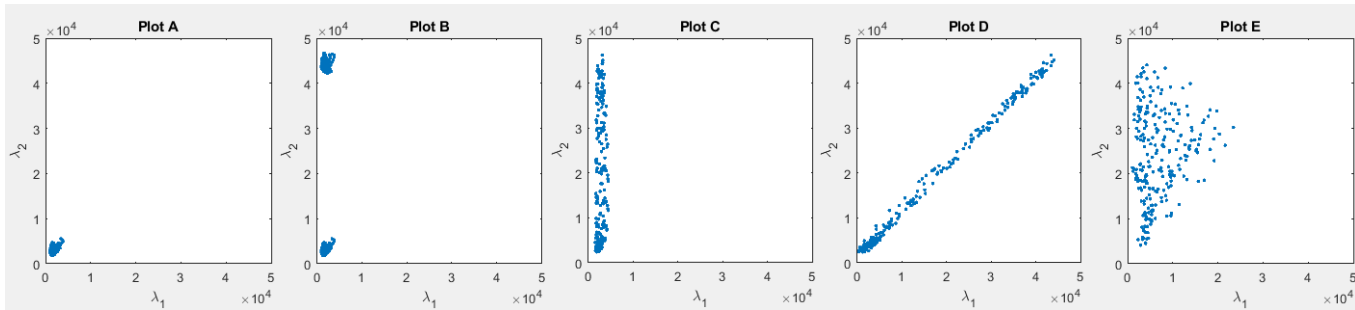
Consider the following 5 greyscale images. Each image is two-toned with values of 100 and 180, but is corrupted by zero-mean Gaussian noise with a standard deviation of 15. Note the red squares are not part of the image, but are placed only to denote a region of interest.



You implement a Harris corner detector and estimate the second moment matrix H using a fixed 5×5 window for the pixels in the red square. You opt to compute the eigenvalues associated with each H matrix explicitly. The built-in eigenvalue function that you use returns the two eigenvalues λ_1 and λ_2 sorted so that λ_2 is always the greater eigenvalue.

5 pts

(a) For each image, select the scatter plot which will be representative of the resulting eigenvalues.



Img 1 – Plot A; Img 2 – Plot C, Img 3 – Plot C, Img 4 – Plot E, Img 5, Plot E; 1 pt each

For each statement below, consider whether the spread of points in the representative scatter plots for each image change? If so, describe the change in spread. If not, explain why. Note: spread of points refers to the region or areas covered in the plot, but not the density of the points.

- 1 pt (b) The region of interest i.e. the red square is increased to cover the entire image.
- 1 pt (c) The sigma of the Gaussian noise is reduced to a very small value, e.g. a standard deviation of 1
- 1 pt (d) The sigma of the Gaussian noise is increased to a very large value, e.g. a standard deviation of 50.

(b) There are no significant changes to the representative plots since the small square already contains all the representative image statistics.

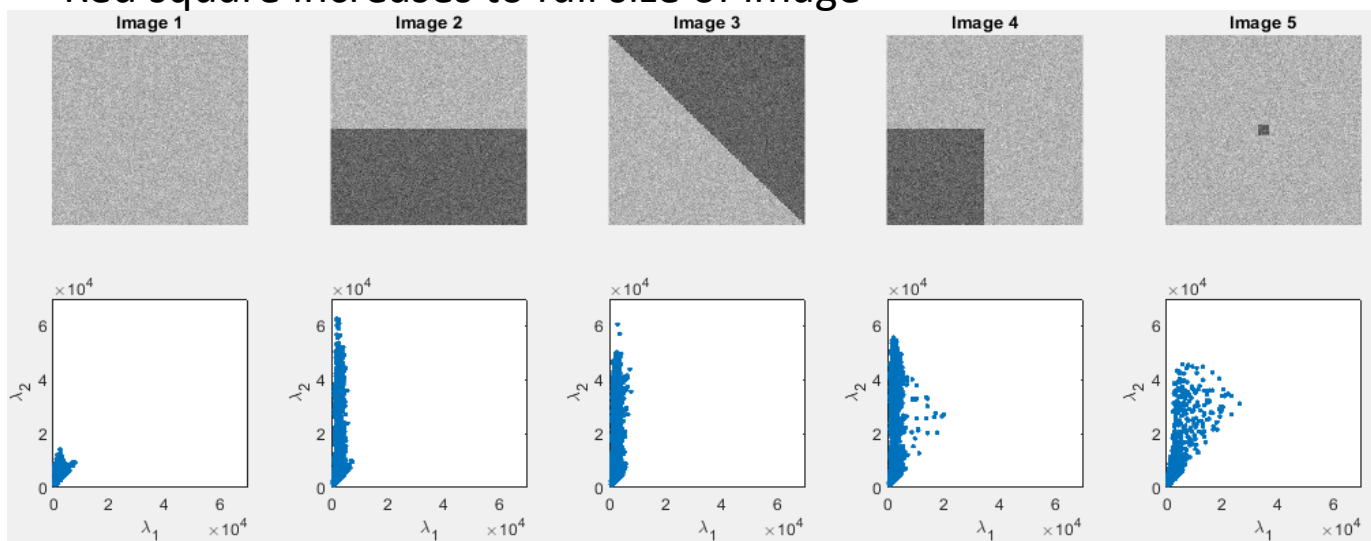
1 pt for each explanation.

Note: sigma affects the magnitude of the image gradients which in turn affects the range of eigenvalues.

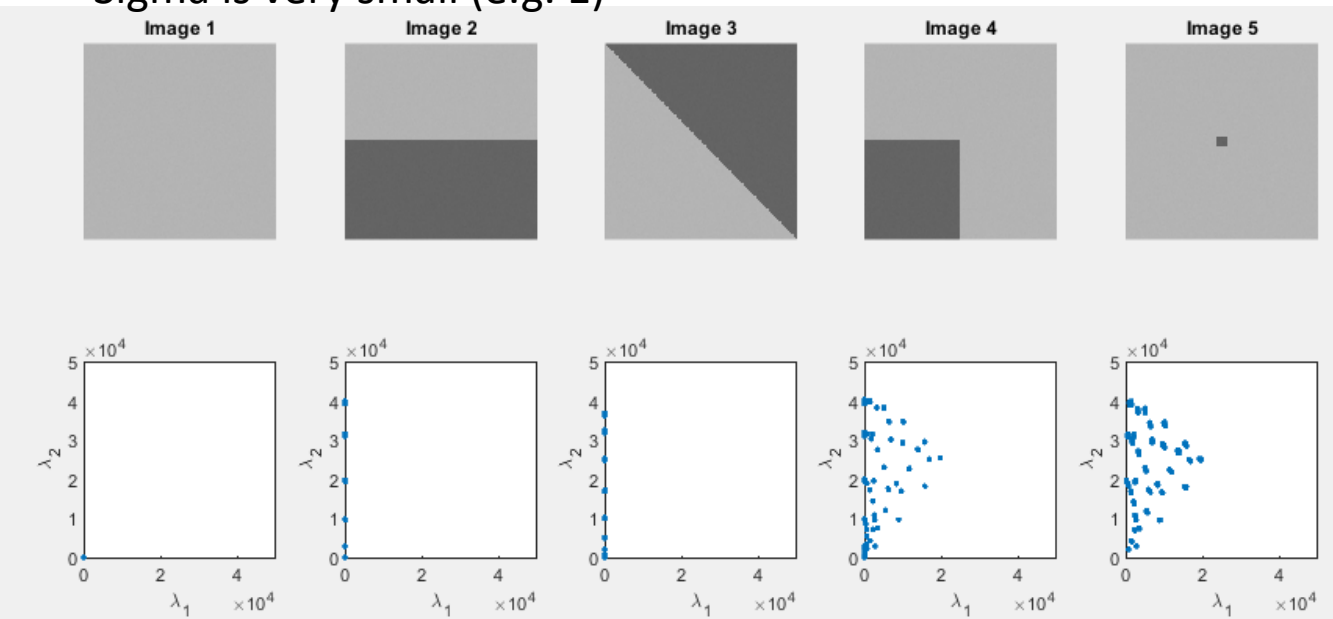
(c) As sigma reduces the cluster for image (1), plot A gets tighter and closer to the origin since the range of λ_1 and λ_2 both decreases and moves towards 0. For Img 2 and 3, plot (c), the range of λ_1 decreases but λ_2 does not so the scatter plot aligns closer to the y axis. For img 4 and 5, plot (e), the spread will not change.

(d) As sigma increases to be very large, the noise starts to dominate all the image gradients so the spread of the eigenvalues all begin to resemble that of Plot E.

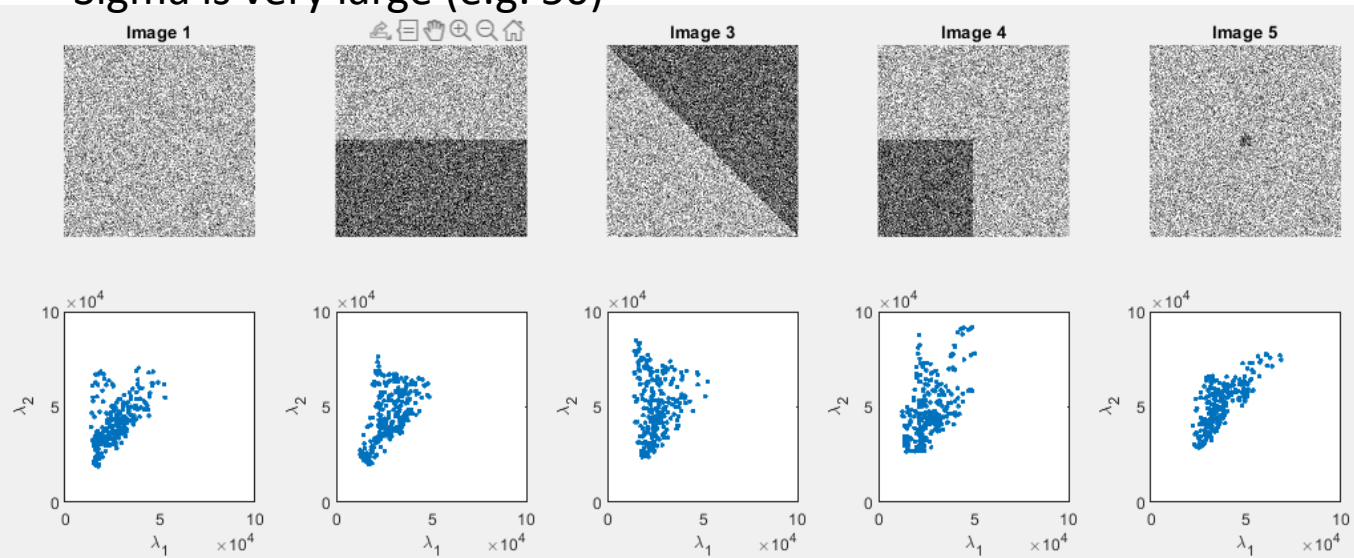
- Red square increases to full size of image



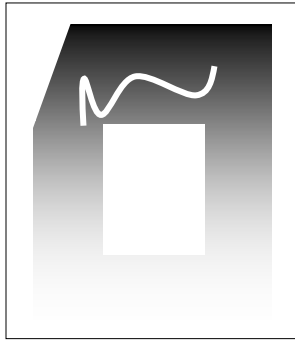
- Sigma is very small (e.g. 1)



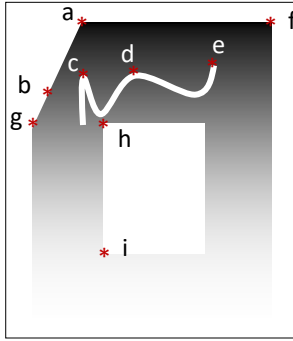
- Sigma is very large (e.g. 50)



Q2: Keypoint Detection & Descriptors (8 pts)



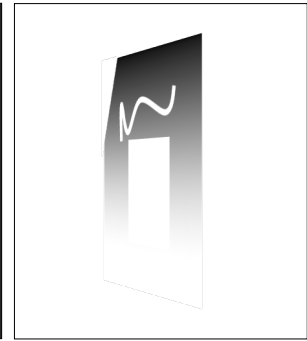
(a) original image



(b) annotated image



(c) transformed image 1



(d) transformed image 2

- i. Consider the image in Fig. (a) to which you apply a Harris corner detector. For the corresponding points a, f, b, g , and h , as marked on the annotated image in Fig. (b), fill in the blanks below so that the points are ordered according to their cornerness response from strongest to weakest, e.g. $x > y > z$. In the case of ambiguities, the same rank can be assigned to multiple points using " \approx ". In this case, both solutions will be accepted e.g. $x \approx y > z$ and $y \approx x > z$.

2 pts

$f > a \approx h > g \approx b$

- ii. Currently, points c and e are detected but d is not. Why is this the case? What image transformations could you apply so that point d is also detected as a corner?

1 pt

At the current image scale, point d is likely perceived as an edge. Scaling the image so that it is smaller will allow it to be detected. Increasing the contrast will also strengthen the Harris response.

- iii. Point h is currently detected as a corner but point i is not. Why is this the case? What image transformations could you apply so that point i is also detected as a corner?

1 pt

The contrast difference at h is higher than i . To detect i as a corner, one needs to increase the contrast of the image.

- iv. To match keypoints detected in the original image in Fig. (a) with keypoints found in the transformed image 1 in Fig. (c), you opt to use the MOPS descriptor. Is this a good choice? Why or why not?

2 pts

MOPS will not work. The transformed image undergoes 3 types of transformations: scaling, rotation and inversion of the intensities. The MOPS descriptor is scale and rotation invariant but the descriptor values come from sampling the gray-scale values so once the greyscale values are inverted, it will no longer be able to match the keypoints.

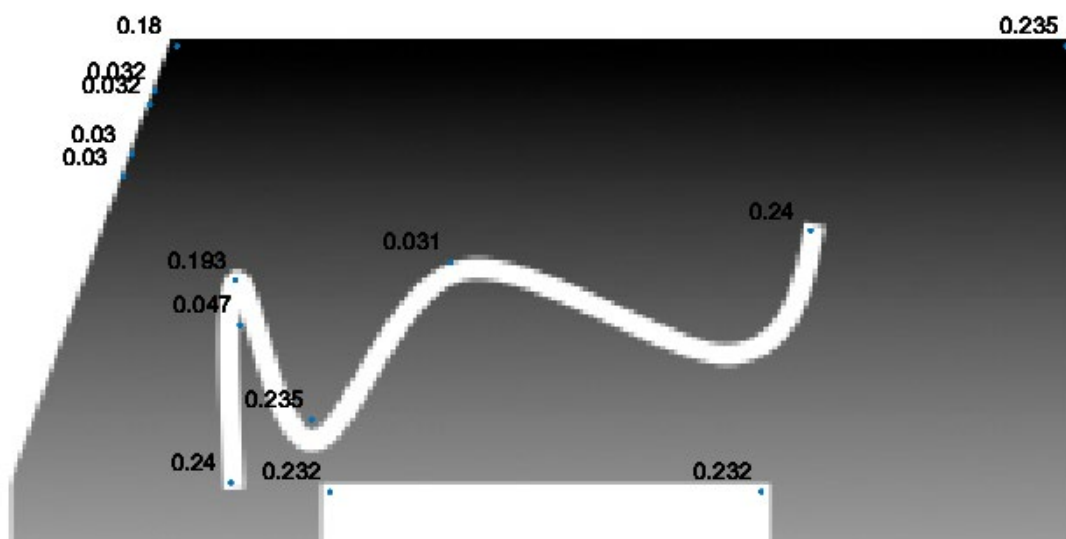
0.5 pts for no, 1.5 points for rationale

- iv. Consider transformed image 2 in Fig. (e) which has undergone a projective transformation. Are SIFT descriptors invariant to such a transformation? Why or why not?

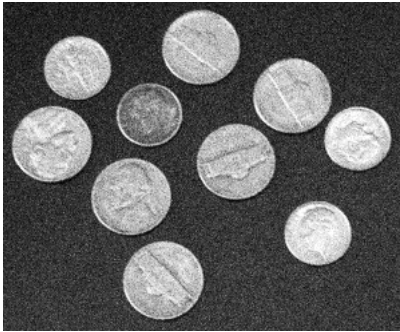
2 pts

No. While SIFT descriptors are robust to such transformations, they are not invariant. A simple intuition is that projective transformations (and affine), with the non-uniform scaling in different directions will change local structures and shapes. This in turn affects the distribution of gradient orientations locally.

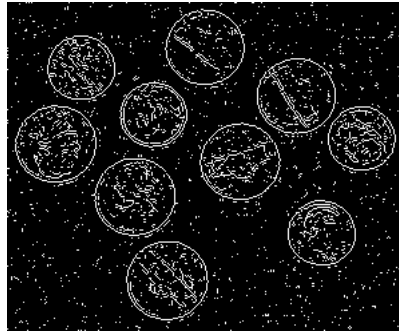
0.5 points for no, 1.5 points for rationale



Q3: RANSAC (8 pts)



(a) original image



(b) Canny edge result



(c) Cropped coin, Alg A



(d) Cropped coin, Alg. B

Consider the image in Fig. (a). You are tasked with counting the number of coins in the image and determining the radius of each coin (in pixel units). You decide the best option is to fit a circle to each of the coins. You first run a Canny edge detector and obtain the result in Fig. (b). You chose a poor threshold so the result is noisy, and around 50% of resulting edge pixels do not belong to a coin edge. To discard these noisy pixels, you opt to use RANSAC.

Note the equation of a circle can be given as $(x - x_o)^2 + (y - y_o)^2 = r^2$ where x_o, y_o are the circle center coordinates and r is the circle radius.

Derive the linear least squares solution for solving the circle parameters. In this solution, you form a system of equations which can be expressed as $\mathbf{A}\mathbf{p} = \mathbf{b}$. Like in the Direct Linear Transform (DLT), \mathbf{A} and \mathbf{b} are matrices stacked together from \mathbf{A}_i and b_i where i is the data index, while the matrix \mathbf{p} contains the parameters we wish to solve, i.e. x_o, y_o , and r , and is common to all i . \mathbf{p} can then be solved using the pseudo-inverse $\mathbf{p} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b}$.

- (i) Derive the expression for \mathbf{A}_i and \mathbf{p} for a given sample i . *Hint: \mathbf{A}_i is a $[1 \times 3]$ vector, \mathbf{p} is a $[3 \times 1]$ vector and b_i is a scalar, where $b_i = x_i^2 + y_i^2$.*

4 pt

$$\mathbf{A}_i = [x \quad y \quad 1], \quad \mathbf{p} = \begin{bmatrix} 2x_o \\ 2y_o \\ r^2 - x_o^2 - y_o^2 \end{bmatrix}$$

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} 2x_o \\ 2y_o \\ r^2 - x_o^2 - y_o^2 \end{bmatrix} = [x^2 + y^2]$$

$$(x - x_o)^2 + (y - y_o)^2 = r^2$$

$$x^2 - 2xx_o + x_o^2 + y^2 - 2yy_o + y_o^2 = r^2$$

$$2xx_o + 2yy_o + r^2 - x_o^2 - y_o^2 = x^2 + y^2$$

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} = [x^2 + y^2]$$

Which of the following is a valid expression for the distance threshold δ for points to be considered an inlier?

☐ $r - \sqrt{x^2 + y^2} < \delta$

☐ $\sqrt{r^2 + (x - x_o)^2 + (y - y_o)^2} < \delta$

☐ $r - \sqrt{(x - x_o)^2 + (y - y_o)^2} < \delta$

☐ $(r - \sqrt{(x - x_o)^2 + (y - y_o)^2})^2 < \delta$

Only the last option is valid. Third option without the squaring may result in negative distances. First two options are random.

1 pt

You take a brute-force approach to count the coins and estimate their radii by applying RANSAC with the linear least squares solution from Q12. You use as input the coordinates of all the edge pixels found by Canny as shown in Fig.

(b). For counting the coins, you accept all fitted circles with inliers greater than some threshold and discard overlapping circles in a greedy fashion.

How many iterations of RANSAC will you need to run, if you use the minimum number of points necessary for fitting the model and you set $p=0.99$, i.e. probability 0.99 that at least one set of points sampled does not contain any outliers. Recall that the number of iterations N is determined by $N=\log(1-p)\log(1-(1-e)s)$, where e is the probability that a point is an inlier and s is the number of points drawn per iteration for fitting.

1 pt

50% of the pixels are estimated to be non-edge pixels, and as an oracle, we see there are 10 coins, this means ~95% of the points are non-edge pixels i.e. outliers for a given circle or coin. $e = 0.05$; If we use the minimum 3 points per RANSAC draw, this would result in 36840 iterations of RANSAC. Will also accepted rounded instead of ceiling answer of 36839 iterations.

Propose a method to speed up the brute-force solution in Question 14. Outline the inputs, outputs, and parameters of your method and clearly state any assumptions that you make. Comment on the accuracy of your approach compared to the brute force solution. Comment on the efficiency of your approach by estimating the number of iterations you will need to run RANSAC for the same settings (s , p stay the same) as given in Question 14.

In the brute force version we sample points from across the entire image to count the coins. There is no need to do this since the coins are relatively small and uniform compared to the entire image. We can simply sample three points which are close to each other e.g. within some region of interest 2 or 3 times the size of the coin. This would allow us to reduce the outlier rate from 95% to e.g. 50%.

Input: edge pixel coordinates from overlapping cells of the image (instead of entire image)

Method: Partition the image into overlapping cells. Search for one coin within each cell. Afterwards, we can further refine the inlier list with the found center and radius candidates. After accepting all fitted circles with inliers greater than some threshold, again, discard overlapping circles in a greedy fashion.

Output: circle parameters relative to the cell location; shift back to global coordinates with offset of the cell, radius of circle will stay the same.

Parameters: size of each cell and extent of overlap

Accuracy: Since we are not random sampling across the entire scene, and actually search across the entire image within the cells, we expect that this method may have better recall than the brute force search

Efficiency: If the number of cells in the image is reasonable, e.g. ~100, then you will only need to run about 3500 iterations ($N = 35$ iterations per cell for $p=0.99$, $s=0.3$, assuming 50% outliers).

1 pt for method with inputs, outputs, params & rationale

1 pt for accuracy, 1 pt for efficiency