Kajol Taresh Shah
A20496724
CS512 Spring 2022

Assignment 2.

Q1. a) SNR in an image

$$SNR = \frac{E_s}{E_n} = \frac{\sigma_s^2}{\sigma_n^2} = \frac{\frac{1}{n}\sum_{i,j}\left(I(i,j) - \bar{I}\right)^2}{\sigma_n^2}$$

where

σ_n^2 = variance for multiple frames of a static scene or variance in a uniform image region.

E_s = Energy of signal

E_n = Energy of noise

SNR is the ratio of the desired signal to background noise.

Q1. b)

Gaussian noise	Impulsive noise
1. We see Gaussian noise is the fluctuations that we can see in the image as we move to the next levels.	1. Impulsive noise can be caused due to the defect in the sensors and also called as salt and pepper noise.
2. It occurs due to sensor limitations while capturing image under low-light conditions.	2. It corrupts the images and replaces some of the pixels in the original image.
3. Gaussian filter handles gaussian noise better.	3. Median filter handles impulsive noise better.

Averaging filter is sensitive to outliers whereas median filter is not. So, median filter handles impulsive noise better.

Q1.c Let the image be I =>

2	2	2	2	2
2	2	2	2	2
2	2	2	2	2
2	2	2	2	2
2	2	2	2	2

Convolution filter =>

1	1	1
1	1	1
1	1	1

If we compute the filter with one window of the image, we will get

$1*2 + 1*2 + 1*2 + 1*2 + 1*2 + 1*2 + 1*2 + 1*2 + 1*2 = 18$.

In convolution, we replace the computed value at the center of 3×3 window. So, after applying convolution to the image I, we will get:

18	18	18	18	18
18	18	18	18	18
18	18	18	18	18
18	18	18	18	18
18	18	18	18	18

Q1.d) Convolution is a linear filter and according to convolution properties

$$\frac{d}{dx}(f * g) = \frac{d}{dx} f * g = f * \frac{d}{dx} g$$

where f is the filter & g is the image. In this way, the operation can be applied more efficiently.

Q1.c. The three different ways to handle boundaries during convolution are

1) zero padding
add zeros around the image. when we use the filter and number of zeros depends on size of the filter

eg.

0	0	0	← zeros padded
0			← image
0			
0			

2) Mirror / replicate - better than zero padding but more complicated

replicated values

5	5	6	7 ...
5	5	6	7
8	8	9	10
11	11	12	13
			...

3) Ignore the outside values of the filter or boundaries (if image is large and filter is small)

	0	0	1
	1	1	1
	1	1	1

Q1. f) Smoothing filter = $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

∴ Smoothing filter =

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Sum of all the entries $= \frac{1}{9} + \frac{1}{9} + \frac{1}{9} + \frac{1}{9} + \frac{1}{9} + \frac{1}{9} + \frac{1}{9} + \frac{1}{9} + \frac{1}{9} = \frac{9}{9} = 1$.

The sum should be equal to 1 so as to keep the mean value of the filtered image or the image

g) If $I_G = I * G = \sum_i \sum_j I(i,j) \cdot e^{-\frac{i^2+j^2}{2\sigma^2}}$

$= \sum_i e^{-\frac{i^2}{2\sigma^2}} \sum_j I(i,j)\, e^{-j^2/2\sigma^2}$

$= (I * G_y) * G_x = I * G_x * G_y$.

1D gaussian cols; 1D gaussian rows

One 2D pass will perform $M \times N \times m^2$ operations

2 1D passes will perform $2 \times M \times N \times m$ operations

So, implementing 2D Gaussian using two 1D convolution filters is more efficient.
It is possible to implement ~~any~~ 2D filters in this way only if the filter is linear and separable. Only then, the 2D convolution filter can be de-composed into 1D convolutions.

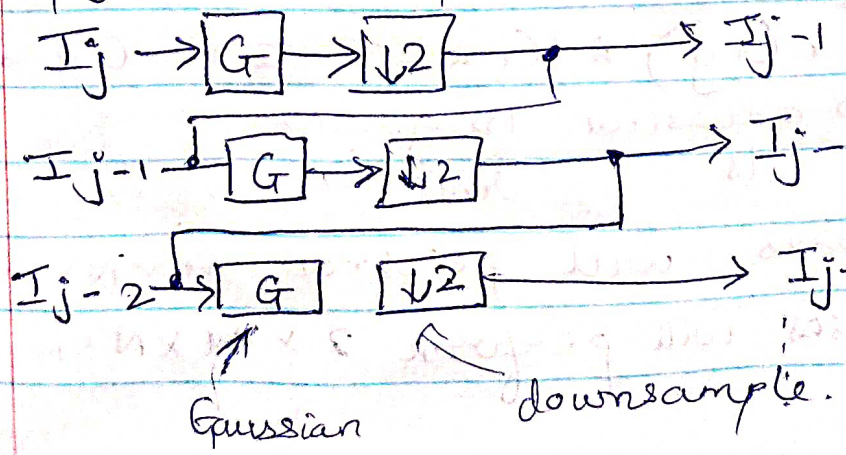
(1.h) $\sigma = 2$.

$m \geq 5\sigma$

$\therefore m \geq 5 \times 2 = 10$.

$\therefore m \geq 10$

(1.i) We use image pyramids for multiple scale analysis. The purpose of producing such image pyramids is to reduce the size of the image.
Below is the way in which Gaussian pyramid is produced in the following way



At each level, half of the size of the image is considered.

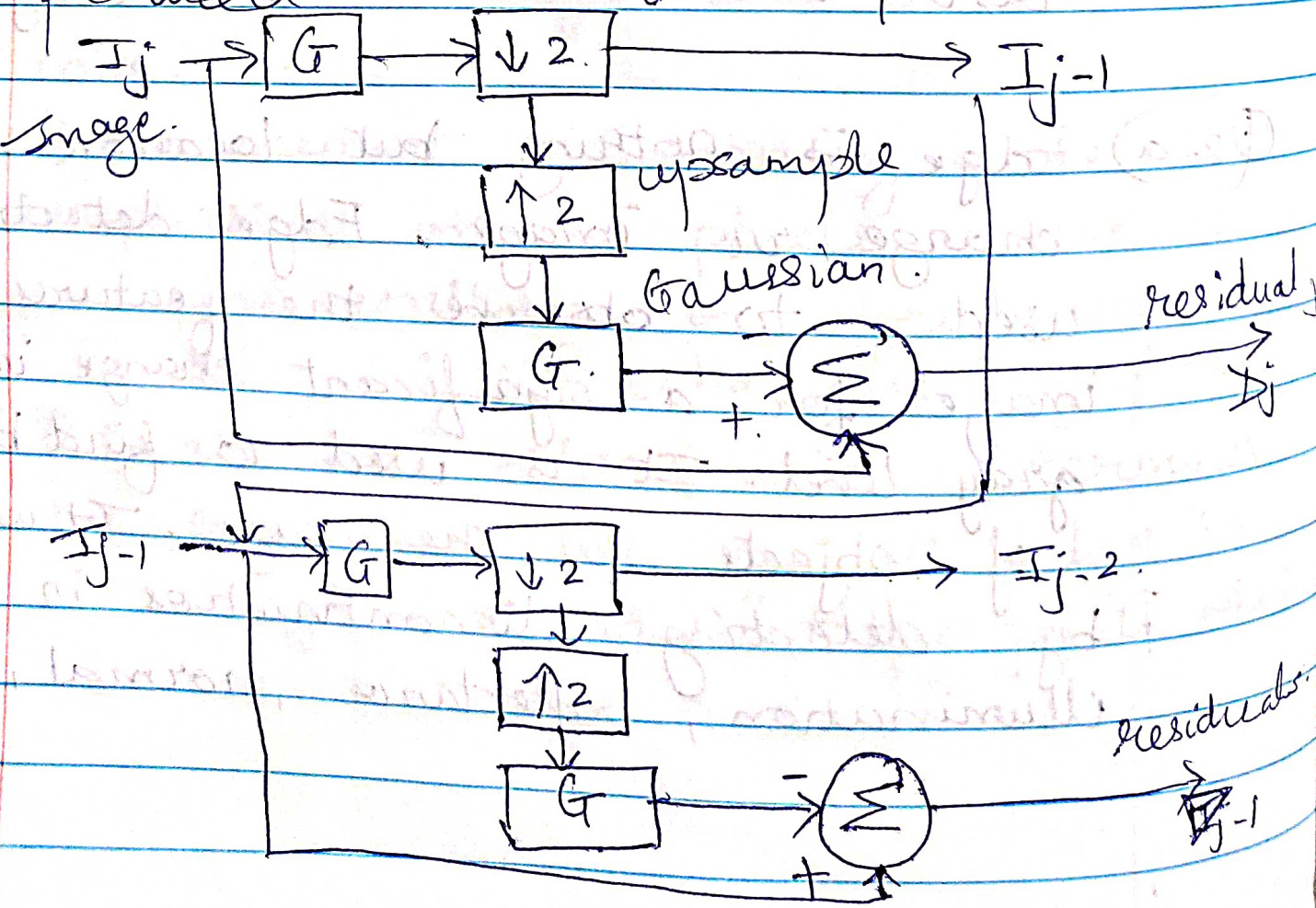
$m = 2^J \quad \therefore J = \log_2 m$

Total number of pixels in a pyramid will be (Additional processing).

$m^2 + \frac{1}{4} m^2 + \frac{1}{16} m^2 + \ldots \leq \frac{4}{3} m^2$

for single image

Q1.j) Below is how Laplacian pyramid is produced



The Laplacian pyramid is useful for compression of image (small residuals)
In Laplacian, first we convolve the image using Gaussian (smoothing), then downsample, upsample (back to original image size) and again convolve (Gaussian). After that, we compute the difference between the original image and the new reduced image (which is a little destroyed due to the operations)
D_j is the difference in the images.
We continue this till we reach the last level which is the image I_0.

Q2. a) Edge is nothing but location with change in image. Edge detection is used to observe the features of an image for a significant change in the gray level. It is used to find boundaries of objects in the image. It works by detecting discontinuities in depth, illumination, reflectance, normal, etc.

The desired properties of edge detection are:-

1) corresponding to scene elements
2) invariant
3) reliable detection

Q2.b) Basic steps of edge detection

1) Smoothing - Smoothing is done to reduce noise but without affecting the edges or blurring them. This ~~co~~

~~2) Enhance the edges - This~~

improves the performance of an edge detector as they are sensitive to noise.

2) Enhancement - Enhancing the edges emphasizes on the pixels where there is edge contrast or significant change in the image. To enhance quality of edges

3) Localization - Localization determines the exact location of an edge in the image. We can detect the points which lie on the edge

Q2.c) To compute change in image intensity, we need image gradient. Gradient vector points in the direction of maximum change. The magnitude of the gradient provides information about the strength of the ~~image~~ edge. We use gradients for detecting edges in image and to find contours & outlines of objects in images. The gradient of an image measures how it is ~~to~~ changing; the magnitude tells how quickly the image is changing while the direction of the gradient tells us the direction in which the image is changing.

① Sobel filter (for computing image gradient)
In this filter, we first do smoothing and then take derivative

$$\Delta x = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} * \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\Delta y = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

(2) Laplace filter or LOG

In this, we first do smoothing using Gaussian and then apply Laplacian

$$\Delta I = \nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} = Ixx + Iyy$$

$$Ixx = \begin{bmatrix} 1 & -2 & 1 \end{bmatrix}$$

$$Iyy = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

$$Ixx + Iyy = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Q2.d) In sobel filter, we first do smoothing an then take derivatine

$$Ix = I * \left(G * \frac{\partial}{\partial x}\right) \quad Iy = I * \left(G * \frac{\partial}{\partial y}\right)$$

$$\therefore \Delta x = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} * \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\Delta y = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

(2.e) $I_x = I * G_x$ if $[x] * G[x]$

$I_y = I * G_y$

By using separable property of Gaussian

$I_x = I * G'[x] * G[y]$ ← 1D convolution with horizontal Gaussian derivative

1D convolution with vertical Gaussian

$I_y = I * G'[y] * G[x]$ ← 1D convolution with horizontal Gaussian

1D convolution with vertical Gaussian derivative

I_y = [Image] * [ ] * [Smooth]

derivative

I_y = [Image] * [ ] I_x = [Im] * [ ] * [ ]

derivative Smoo

$$G(x) = e^{-\frac{x^2}{2\sigma^2}}$$

$$\therefore G'(x) = -\frac{2x}{2\sigma^2} e^{-\frac{x^2}{2\sigma^2}}$$

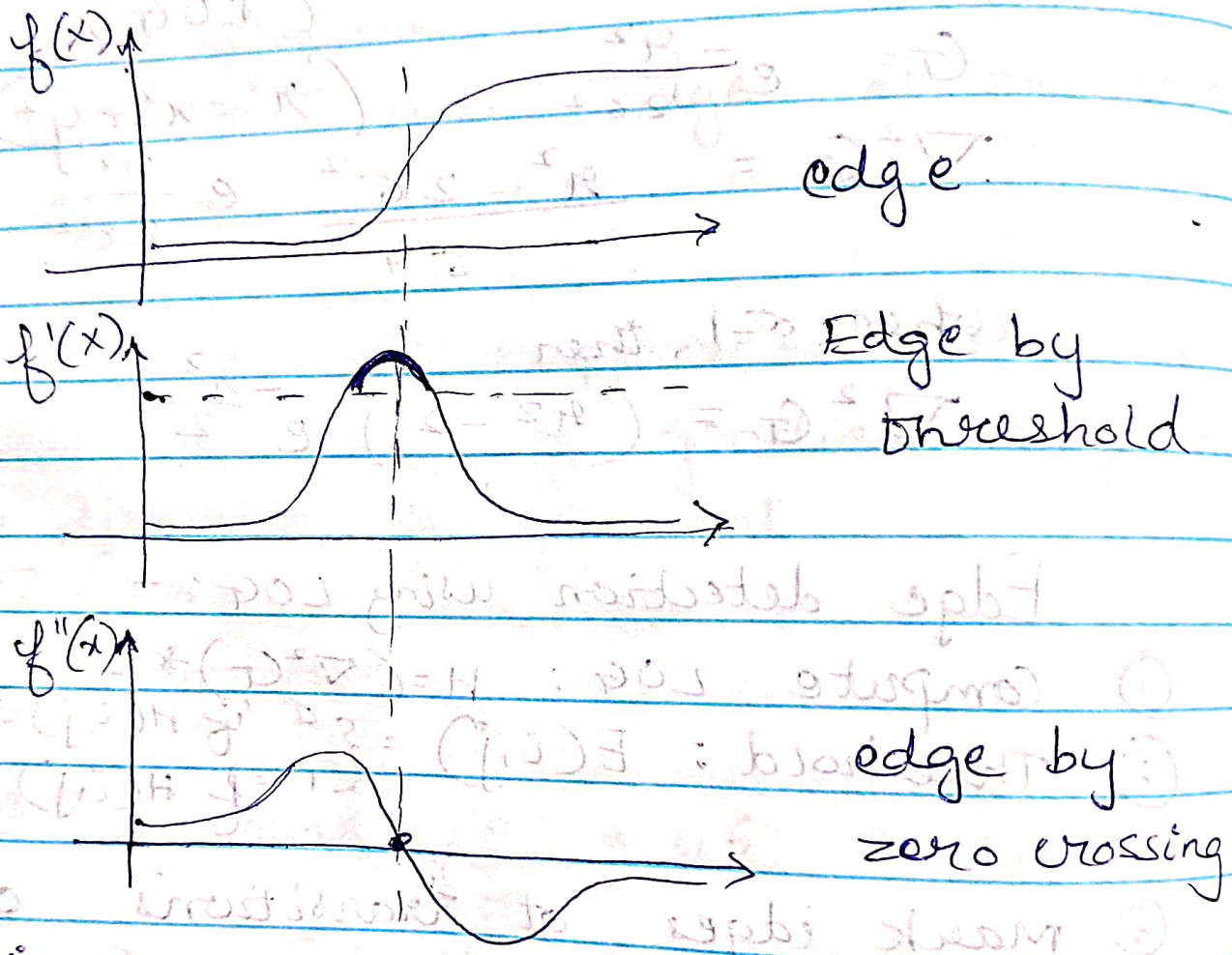
$$G'(y) = -\frac{2y}{2\sigma^2} e^{-\frac{y^2}{2\sigma^2}}$$

$\therefore$ If $\sigma = 2$; the I_x & I_y

$$I_x = I * \left(-\frac{x}{2^2} e^{-\frac{x^2}{2\times 2^2}}\right) = I * \left(-\frac{x}{4} e^{-\frac{x^2}{8}}\right)$$

$$I_y = I * \left(-\frac{y}{4} e^{-\frac{y^2}{8}}\right)$$

Q2. f) Using Laplacian of Gaussian, we can detect edges using the first and second order derivative of the image.



Using first order derivative, we can localize the edge by threshold.
Using second order derivative, we can localize the edge by zero crossing.

(2.g) $\sigma = 1$

LOG ?

$$H = \nabla^2 (I * G) = \nabla^2 G * I$$

(LOG)

$$G = e^{-\frac{r^2}{2\sigma^2}} \qquad (r^2 = x^2 + y^2)$$

$$\nabla^2 G = \frac{r^2 - 2\sigma^2}{\sigma^4} e^{-\frac{r^2}{2\sigma^2}}$$

when $\sigma = 1$, then

$$\nabla^2 G = (r^2 - 2) e^{-\frac{r^2}{2}}.$$

Edge detection using LOG :-

① Compute LOG : $H = (\nabla^2 G) * I$.

② Threshold : $E(i,j) = \begin{cases} 0 & \text{if } H(i,j) < 0 \\ 1 & \text{if } H(i,j) \geq 0 \end{cases}$

③ Mark edges at transitions $0 \to 1$, $1 \to 0$ (scan left to right and top to bottom

(2.h) Canny edge detection algorithm uses directional derivative to detect edges whereas standard edge detection algorithm ~~does~~ does not. Canny edge detection detects edges at zero crossings of second order directional derivative taken ~~ordg~~ along the gradient. Also, it will attempt

detection only if gradient magnitude is large enough ($|n| > \tau$). This is the condition for detecting edges.

n = gradient

if $|n| > \tau$ detect edges at zero crossing of $\frac{\partial^2}{\partial n^2}(I * G)$.

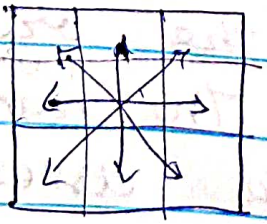
Q2. i) Non maximum suppression is to get local maximum of gradient magnitude in direction of gradient

$\nabla (I * G) = (I_x, I_y)$

$\theta = \tan^{-1}\left(\frac{I_y}{I_x}\right)$

$\theta^* = \text{round}\left(\frac{\theta}{45}\right) * 45$

$E(i,j) = \begin{cases} 1 & \text{if } \nabla(I*G) \text{ is a local maximum} \\ 0 & \text{otherwise} \end{cases}$



compare neighbours at their discretization.

Hysteris thresholding

Use τ_H to start tracking and τ_L to continue $(\tau_H > \tau_L)$

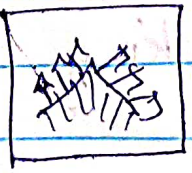
① Initialize array of visited ~~places~~ pixels $v(i,j) = 0$

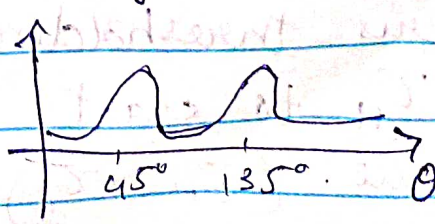
② Scan image T-B, L-R:
if $!v(i,j)$ && $|\nabla I| > \tau_H$ start tracking an edge

③ Search for additional neighbours in direction orthogonal to ∇I such that $|\nabla I| > \tau_L$.

Q3 & a) Corner detection is used to find points of change in an image. It is used to track or characterize objects and characterize local windows. A corner is detected if there is more than one direction in orientation histogram of an image. So, we find two distinct directions in the orientation histogram for corner. The ~~number of~~ number of ~~of principal~~ direction is assessed using orientation histograms.

eg. for corner





Two angles.
~~two~~

We get a bunch of vectors and we try to find a principal direction which aligns best with all vectors. We use PCA to assess number of principal directions

1. We first find correlation matrix of gradients in local windows
2. Find eigen values of correlation matrix
3. Detect corner in window if eigen values are sufficiently large

Q3.b). In PCA, to find principal directions for of gradient orientations, we find directions that will minimize the projection on them, so that they are orthogonal to previous directions. Using PCA, we find direction V such that projection of $\{g_i\}$ onto V is minimized.

$$E(V) = \sum_i (g_i \cdot V)^2 = \sum_i (g_i^T V)(g_i^T V)$$

$$E(V) = V^T C V$$

where V is a 2D vector & C is the correlation matrix.

$$C = \sum_i g_i g_i^T = \begin{bmatrix} \Sigma x_i^2 & \Sigma x_i y_i \\ \Sigma x_i y_i & \Sigma y_i^2 \end{bmatrix}$$

where $g_i = [x_i, y_i]^T$

$$\therefore \quad V^* = \underset{V}{\operatorname{argmin}}\, E(V).$$

$$\frac{d}{dV} E(V) = 0$$

$$\therefore \quad \nabla E(V) = 0 \quad \begin{cases} \frac{d}{dx} E = 0 \\ \frac{d}{dy} E = 0 \end{cases}$$

gradient

$$2CV = 0$$

Q3. c) correlation matrix $C = \begin{bmatrix} \Sigma x_i^2 & \Sigma x_i y_i \\ \Sigma x_i y_i & \Sigma y_i^2 \end{bmatrix}$

$$C = \begin{bmatrix} 1+1+1+1 & 1+2+3 \\ 1+2+3 & 1+4+9+1+4+9 \end{bmatrix}$$

$$C = \begin{bmatrix} 4 & 6 \\ 6 & 44 \end{bmatrix} = \begin{bmatrix} 4 & 6 \\ 6 & 44 \end{bmatrix}$$

Q3. d). Corner is detected if

$\lambda_1 \cdot \lambda_2 > T$

i.e. dot product of eigen values should be larger than threshold. λ_1 & λ_2 both should be large

Q3 e). Non maximum supression

1. compute λ_1, λ_2 for all windows
2. select windows with $\lambda_1 \cdot \lambda_2 > T$ and sort in decreasing order
3. Select the top of the list as corner and delete all other corners in its neighbourhood from the list
4. Stop once detecting X% of the points as corners.

Q3. f) Harris corner detection.

$$G(C) = \frac{\det(C)}{} - k\,tr^2(C)$$

$$= \lambda_1 \cdot \lambda_2 - k(\lambda_1 + \lambda_2)^2$$

$$= \lambda_1\lambda_2 - k(\lambda_1^2 + 2\lambda_1\lambda_2 + \lambda_2^2)$$

$$= (1-2k)\lambda_1\lambda_2 - k(\lambda_1^2 + \lambda_2^2)$$

$\lambda_1\lambda_2(1-2k) \gtrsim 0$ if $k = 0.5$ ⇒ corner detection

$k.(\lambda_1^2 + \lambda_2^2) \gtrsim 0$ if $k = 0$ ⇒ edge detection.

So, without computing λ_1 & λ_2,
if $k = 0.5$ $G(C)$ detects edges
if $k = 0$ $G(C)$ detects corners.

Harris corner detection will detect corners where $G(C)$ is high.

Q3. g) Corner localization means given that there is a corner in a window, find its location. To determine if 'p' is the corner connect each point x_i to p and project the gradient at x_i onto $(x_i - p)$

$$E(p) = \sum_i \left(\nabla I(x_i) \cdot (x_i - p)\right)^2$$

$$E(p) = \sum_i (x_i - p)^T \left(\nabla I(x_i) \nabla I(x_i)^T\right)(x_i - p)$$

$$p^* = \underset{p}{\operatorname{argmin}}\ E(p)$$

$$\nabla E(p) = 0$$

$$-2 \Sigma_i \left(\nabla I(x_i) \nabla I(x_i)^T\right)(x_i - p) = 0$$

$$\Sigma \nabla I(x_i) \nabla I(x_i)^T p = \Sigma \nabla I(x_i) \nabla I(x_i)^T x_i$$

$$p^* = \left(\Sigma_i \nabla I(x_i) \nabla I(x_i)^T\right)^{-1} \sum_i \nabla I(x_i) \nabla I(x_i)^T x_i$$

$$= C^{-1} \sum_i \nabla I(x_i) \nabla I(x_i)^T x_i$$

↑ location of corner

C → correlation matrix

If we detected corner in the window $\lambda_1 . \lambda_2 > \tau \Rightarrow C$ must be non singular.

So, for the formula to be true, C should be non-singular.

(Q3. h) Characterization of feature points using HOG

① split each patch into cells (possibly overlapping)
② Create orientation histogram in each cell (using edge or gradient directions, possibly weighted by distance from center or gradient magnitude.
③ Concatenate orientation histograms

Requirements for a good characterization of feature points :-

① translation variance ⇒ local window
② rotation variance → histograms
③ Scale variance → pyramid
④ illumination variance → gradients

(Q3. i) Scale Invariant Feature Transform (SIFT)

① Use weighted sum of gradient directions to create orientation histogram's M cells, then concatenate
② Align histogram based on dominant direction (rotation invariance)