CS-512 - ASSIGNMENT 2
COMPUTER VISION

RAKSHITH CHURCHAGUNDI AMARNATH
CWID: A20424771

① NOISE AND FILTERING:

(a) Explain How to estimate the signal to noise ratio (SNR) in an image.

Soln: SNR in an image can be estimated by the ratio of energy of signal to the energy of noise in an image [How much noise is there in the 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, E_S = Energy of the Signal

E_n = comparitive energy of noise

σ_S^2, σ_n^2 = Variance of Signal, noise

$\Sigma_{(i,j)}$ = Sum of all pixels in the image

$\bar{I}$ = average value of the image intensity

$I_{(i,j)}$ = intensity of images

$$SNR[db] = 10\log_{10}\left(\frac{E_s}{E_n}\right)$$

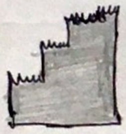
(b) Explain the difference between Gaussian and impulsive noise. which filter handles better impulsive noise: an averaging filter or a median filter.

Solⁿ:

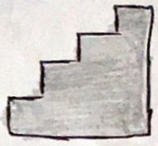
	Gaussian Noise	Impulsive noise
(i)	The values taken by noise are distributed.	The values taken by noise can be unwanted, instantaneous sharp sounds
(ii)	This can be used as additive white noise that can inturn generate additive white Gaussian noise.	This noise can be used to enhance the quality of noisy signals in order to achieve robustness in pattern recognition and adaptive control system.

(iii) **Example:** Image →

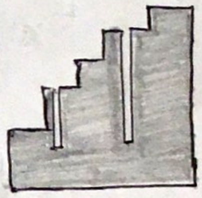
with Noise →



Example: Image →



with Noise →



* Median filter handles better impulsive noise

(C) Given an image having the value of 2 in each cell, write the value of the pixels in this image after applying a 3x3 convolution filter having all 1-s in its entries

Solⁿ:

filter

1	1	1
1	1	1
1	1	1

Image

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	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 matrix :-

	0	1	2	3	4	5	6
0	8	12	12	.	.	.	.
1	12	18	18	.	.	.	.
2	12	18	18	.	.	.	.
3	.	.	.	.	.	.	.
4	.	.	.	.	.	.	.
5	.	.	.	.	.	.	.
6	.	.	.	.	.	.	.

$(0,0) = (1\times 0) + (1\times 0) + (1\times 0) + (2\times 1) + (2\times 1)$
$+ (1\times 0) + (2\times 1) + (2\times 1)$

$= 8$

$(0,1) = (1\times 1) + (0\times 1) + (1\times 2) + (1\times 2) + (1\times 2) + (1\times 2)$
$+ (1\times 2) + (1\times 2)$
$= \underline{12}$

$(0,2) = (0\times 1) + (0\times 1) + (0\times 1) + (2\times 1) + (2\times 1)$
$+ (2\times 1) + (2\times 1) + (2\times 1) + (2\times 1)$
$= \underline{12}$

$(1,0) = \underline{12}$

$$I_A(i,j) = I(i,j) * A(i,j) = \sum_{h=+m/2}^{m/2} \sum_{K=-n/2}^{n/2} A(h,K)\, I(i-h, j-K).$$

$I(1,1) = (1\times 2) + (1\times 2) + (1\times 2) + (1\times 2) + (1\times 2) + (1\times 2) + (1\times 2) + (1\times 2)$
$= \underline{18}$

$I(1,2) = (1\times 2) + (1\times 2) + (1\times 2) + (1\times 2) + (1\times 2) + (1\times 2)$
$= \underline{12}$

$I(2,1) = (1\times 2) + (1\times 2) + (1\times 2) + (1\times 2) + (1\times 2) + (1\times 2)$
$= \underline{12}$

(d) Given that we need the derivative of an image convolved with a filter, explain how the operation can be applied more efficiently.

Sol^n :- let I be the image and f be the filter.

$\frac{d}{dx}(I*f) = \frac{d}{dx} I * f = I * \frac{d}{dx}(f)$ is a convolution property.

This is we get as the derivative of an image which can be used to separate the convolution of one or more images

(e) Explain the 3 different ways to handle boundries during convolution

3 different ways are

(i) <u>zero padding</u> :- Here, we assume zero everytime the filter falls outside an image. while performing convolution

<u>example</u>:-

$$I = \begin{array}{|c|c|c|c|} \hline 1 & 2 & 3 & . \\ \hline 4 & 5 & 6 & . \\ \hline 7 & 8 & 9 & . \\ \hline . & & . & . \\ \hline \end{array} \qquad f = \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

<u>convolution</u>:-

$$\begin{array}{c|c|c|c|c|} 0 & 0 & 0 & 0 & \\ \hline 0 & 1 & 2 & 3 & . \\ \hline 0 & 4 & 5 & 6 & . \\ \hline 0 & 7 & 8 & 9 & . \\ \hline & . & . & . & \\ \hline \end{array}$$

(ii) <u>Replicate</u>:- Here, we will repeat the last (nearest) values of the image, when the filter falls outside of an image

<u>ex</u>:- Consider the following image

$$\begin{array}{c|c|c|c|c|} 2 & 2 & 3 & 4 & \\ \hline 2 & 2 & 3 & 4 & . \\ \hline 5 & 5 & 6 & 7 & . \\ \hline 8 & 8 & 9 & 10 & . \\ \hline & . & . & . & . \\ \hline \end{array}$$

(iii) Ignore:- Here we will ignore the the part of the filter which falls outside out of an image and apply convolution to the rest.

(f) Write a basic 3X3 smoothing filter, what is the sum of all entries in this filter? Explain the reason for the sum of to be selected as it is?

Sol^n:- The Basic 3X3 Smoothing filter is as follows:-

1	1	1
1	1	1
1	1	1

The sum of all entries in the above filter is 9 which means it has 9 times more intensity. we don't need that much high intensity. So we will normalize it by multiplying $1/9$ to the filter.

So, filter now becomes $\frac{1}{9} * \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$.

The Noise increases when Standard deviation increases.

(g) Explain how to implement a 2D convolution with Gaussian using 2-1D convolution filters. which option is more efficient? Is it possible to implement any 2D filter in this way?

Sol^n:- 2D Gaussian filter can be obtained by breaking into 2-1D filters.

convolution.

$I_G = I * G$ (2D gaussian)   2D - $G(x,y) = e^{\frac{-x^2+y^2}{2\sigma^2}}$

we can break it down as,

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

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

$$\therefore (I \times G_y) * G_x = I * G_x * G_y$$

if $M \times N$ is an image and $m \times m$ is a filter, then

one 2D Pass - $\boxed{M \times N \times m^2}$ operation

two 1D pass - $\boxed{2 \times M \times N \times m}$ ← operation

$\underline{2MNm < MNm^2}$

It is not possible to implement any 2D filter in this way as we deal with exponents which inturn makes it different for different values

(h) Give a 1D Gaussian filter with $\sigma = 2$, what should be the size of this filter?

Sol^n:- Given $\sigma = 2$,

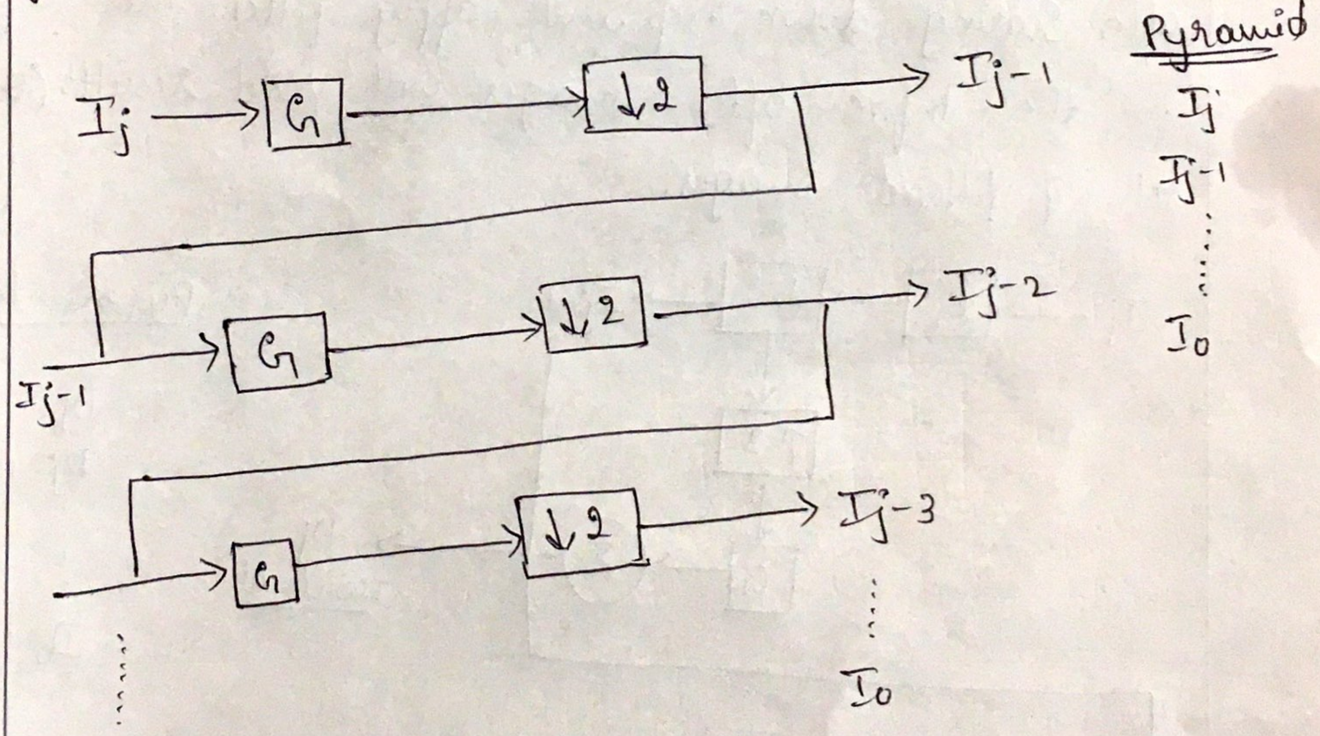
Size of filter, $m = 5\sigma$

$= 5 \cdot 2$

$= \underline{\underline{10}}$ → Size of the filter.

(i) Explain how a Gaussian image pyramid is produced. what is the reason for producing such pyramids? what is the amount of additional processing done in a pyramid compared with a single image.

Soln: Gaussian image is produced by convolving the image with a gaussian filter, then scaling it down. output ~~contains~~ is produced as a set of filtered images after this process is repeated.



where I_j = image at level J

G = Gaussian filter.

$\downarrow 2$ = Sampling with 2

Smoothing and sampling are involved in Gaussian Pyramids.

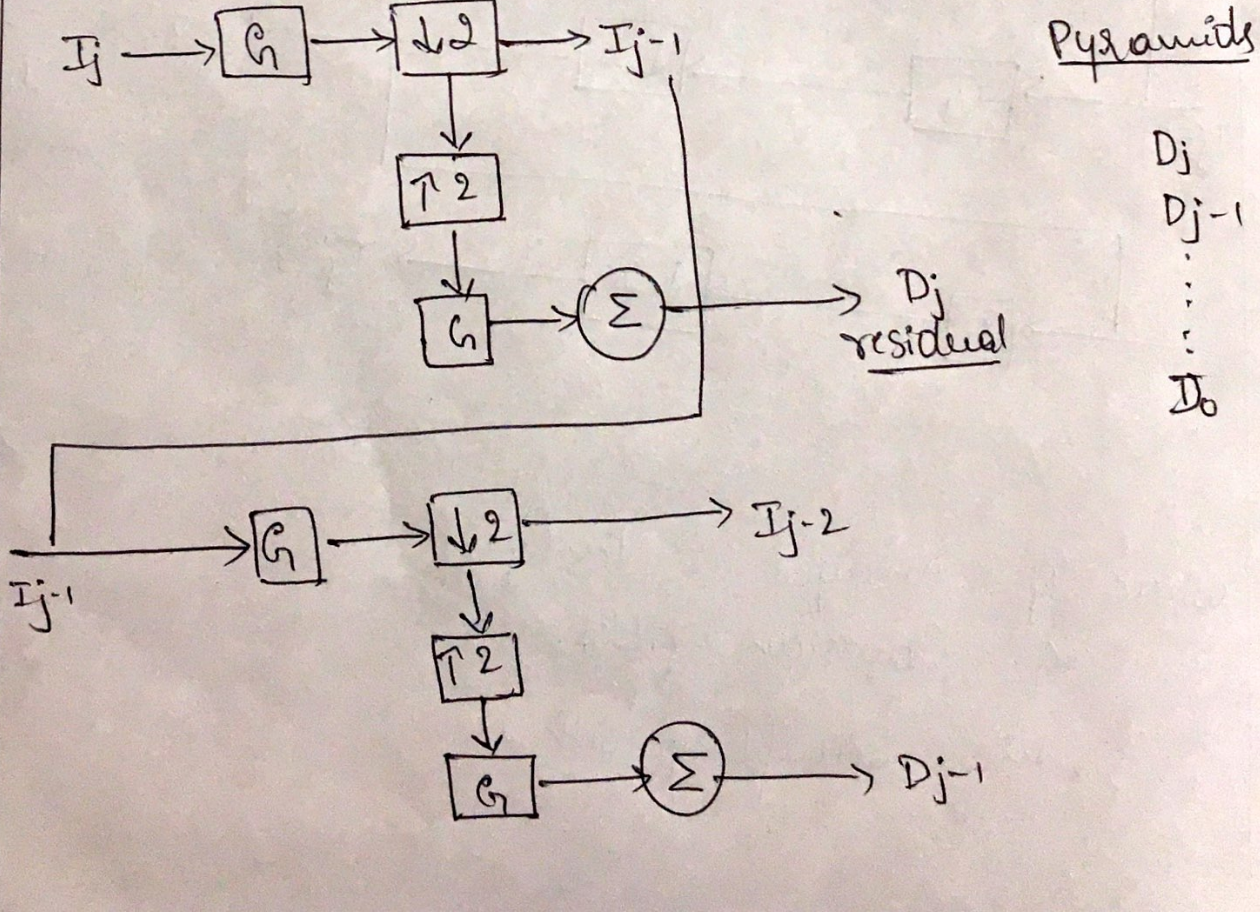
Gaussian pyramids are used to overcome the destruction of an image and for analysis purposes.

$m = 2^j \Rightarrow j = \log_2 m$

* Amount of additional processing done in a pyramid compared with a single image is 30%.

(j) Explain how the laplacian pyramids are produced and its use.

Soln: Convolution of an image with a gaussian filter, then after scaling down & up and apply filter. The difference in the before-after images will yeild results in set of filtered images.



where I_j = Image ~~wi~~ at level j;
G = gaussian filter.

Ex: Consider this image,

$$\begin{matrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 7 & 9 & 1 & 2 \end{matrix} \rightarrow \boxed{\downarrow 2} \rightarrow \begin{matrix} 1 & 3 \\ 7 & 1 \end{matrix} \rightarrow \boxed{\uparrow 2}$$

$$\downarrow$$

$$\begin{matrix} 1 & 1 & 3 & 3 \\ 1 & 1 & 3 & 3 \\ 7 & 7 & 1 & 1 \\ 7 & 7 & 1 & 1 \end{matrix}$$

Laplacian pyramids are useful for compressing an image (small residuals)

(2) EDGE DETECTION

(a) Why is edge detection useful? What are the desired properties of edge detection?

Soln: Edge detection is useful in characterizing images and local neighbouring points in it.

Properties:-

* Image should not have any discontinuity.
* Image should not contain any kind of noise.
* Image should be aligned correctly.

(b) Explain the basic steps of edge detection and need for them :- Smoothing, enhancement, localization.

Solⁿ:-

* **Smoothing:-** It's done to reduce the amount of noise in the image

  Need:- its needed to get appropriate and exact results by removing noise.

* **Enhancing the edges :-**

  Images are enhanced by improving the quality of an image

  Need:- To make edges more visible. its important so that it ~~to~~ removes the possibility of blur's in the image

* **Detection of edges:-** Edges need to be detected once the quality of an image is improved.

  Need:- The edges with strong values are taken into consideration, ~~so~~ Many points have Non zero Value for gradient and not every edges are for usefully.

* **Localize edges :-** the edges are contained (Localized) in this final step.

  Need:- To predict the orientation of edges, location prediction of edges should be estimated with SubPixel resolution.

(c) Describe two filters for computing the image gradient. What is the meaning of an image gradient? What is it used for?

Sol^{n}:

Two filters:-

* Forward differences:-

$$\frac{\partial}{\partial x} I(x,y) = \frac{I(x+h,y) - I(x,y)}{h}, \quad h=1$$

$$= I(x+1,y) - I(x,y)$$

This is obtained by ~~the~~ taking the differences of current and the previous elements.

Convolution masks,

$$\Delta X = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \qquad \Delta y = \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}$$

* Central differences:-

$$\frac{\partial}{\partial x} I(x,y) = \frac{I(x+h,y) - I(x-h,y)}{h}, \quad h=1$$

$$= I(x+1,y) - I(x-1,y)$$

This is obtained by the differences of the next and previous elements.

convolution mask, $\Delta X = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$, $\Delta y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

Image gradient is a directional change in the intensity or color in an image.

Edge detection is the main application of an image gradient.

$$\nabla I(x,y) = \begin{bmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{bmatrix} = \begin{bmatrix} I_x \\ I_y \end{bmatrix}$$

(d) Explain how the sobel filter can be produced from a smoothing filters.

Soln:- The smoothing and taking a derivative of it leads to a soble filter

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

Smoothing filter   derivative (x)   Soble filter

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

Smoothing filter   Derivative (y)   Soble filter.

Scanned with CamScanner

(c) Explain how to generate a more accurate derivative filter with an arbitrary σ. write the element of a filter for more accurate derivative computation with $\sigma = 2$.

Solⁿ:

More accurate derivative filters are calculated as $I_x = \underset{\text{image}}{I} * \underset{\text{derivative}}{G'(x)} * \underset{\text{smoothing}}{G(y)}$

Image is convolued with horizontal gaussian derivative and with a vertical gaussian filter.

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

$$I_y = I * G'(y) * G(x)$$

Here, its convolved with vertical derivative and with a horizontal gaussian.

$$G(y) = \frac{-y}{\sigma^2} e^{\frac{-y^2}{2\sigma^2}}, \quad G(x) = e^{-x^2/2\sigma^2}$$

$$I_x = I * \left(\frac{-x}{\sigma^2}\right) e^{\frac{-x^2}{2\sigma^2}} * e^{\frac{-y^2}{2\sigma^2}}$$

when $\sigma = 2$

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

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

(f) Explain how an edge can be localized using the first or second order derivative of the image.

Soln: 2nd order derivative of the image.

$$\Delta I = \nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} = I_{xx} + I_{yy}$$

$$I_x = I(x+1) - I(x)$$

$$I_{xx} = (I(x+1) - I(x)) - (I(x) - I(x-1))$$

$$= I(x+1) - I(x) - I(x) + I(x-1)$$

$$= I(x+1) - 2I(x) + I(x-1)$$

$$\Rightarrow \begin{array}{|c|c|c|} \hline 1 & -2 & 1 \\ \hline \end{array}$$

$$I_y = I(y+1) - I(y)$$

$$I_{yy} = [I(y+1) - I(y)] - [I(y) - I(y-1)]$$

$$= I(y+1) - I(y) - I(y) + I(y-1)$$

$$= I(y+1) - 2I(y) + I(y-1)$$

$$\Rightarrow \begin{array}{|c|} \hline 1 \\ \hline -2 \\ \hline 1 \\ \hline \end{array}$$

$$I_{xx} + I_{yy} = \begin{array}{|c|c|c|} \hline 1 & -2 & 1 \\ \hline \end{array} + \begin{array}{|c|} \hline 1 \\ \hline -2 \\ \hline 1 \\ \hline \end{array}$$

$$= \begin{array}{|c|c|c|} \hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array}$$

(g) let $\sigma = 1$, write the laplacian of Gaussian (LOG) filter using this σ, Explain how to use LOG to detect edges.

Sol^n:

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

we Know, $G = e^{-r^2/2\sigma^2}$ $\quad [r^2 = x^2 + y^2]$

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

when $\sigma = 1$

$$H = \left[\frac{r^2 - 2(1)^2}{1^4} e^{\frac{-r^2}{2(1)^2}} \right] * I$$

$$= \left[r^2 - 2 \; e^{-r^2/2} \right] * I.$$

Edge detection using LOG.

(i) compute LOG using $H = \nabla^2 G * I$.

(ii) Find the threshold, $E(i,j) = \begin{cases} 0 & \text{if } H(i,j) < 0 \\ 1 & \text{if } H(i,j) \geq 0 \end{cases}$

(iii) Mark transition of edges $0 \to 1$, $1 \to 0$ [Scaning left to right and top to bottom]

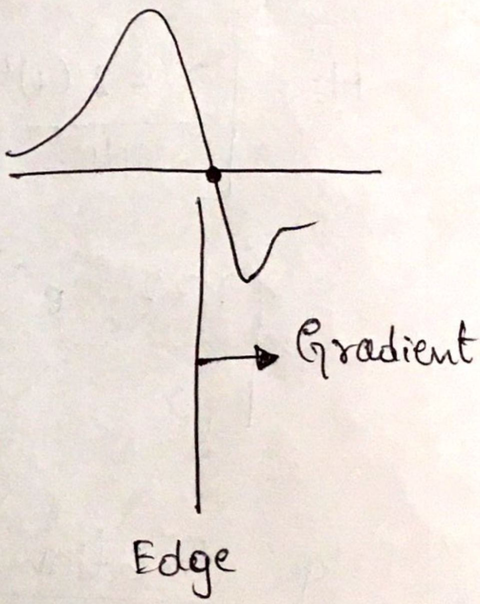
(h) Explain the main difference b/w the canny edge detection algorithm and a standard edge detection that does not use directional derivatives. What is the condition for detecting an edge candidate in canny?

Sol^n:- The main difference b/w them is that it doesn't use directional derivatives in detecting edges at zero crossings of 2nd order derivatives taken along the gradient.

n: gradient

if $|n| > \tau$ detect edges at zero crossing of

$$\frac{\partial^2}{\partial x^2}(I * G)$$



Condition:- Attempt detection only when ~~the~~ it has a larger gradient magnitude.

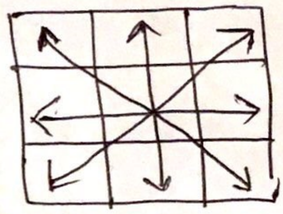
$(|n| > \tau)$

(i) Explain the non-maximum suppression and thresholding parts of the canny algorithms.

Solⁿ:- NON- maximum suppression:-

Its used to find the largest edge. local maxima of gradient magnitude in direction of gradient

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



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

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

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

Hysteresis thresholding:-

use τ_H to track,

τ_L to proceed furthur. $(\tau_H > \tau_L)$

Step 1):- initialize array of visited pixels.

$V(i,j) = 0.$

Step 2):- Scan image from top to bottom and left to right.

Step 3:- Search for additional neighbours in directional orthogonal to ∇I such that $|\nabla I| > \tau_L$.

Scanned with CamScanner