HomeWork 2

Name: Karan Bhatiya
Student Id: A 20424290
Course Number: CS512
Semester: Fall - 18

1. Noise & Filtering

a)

$$SNR = \frac{E_s}{E_n} = \frac{\sigma_s^2}{\sigma_n^2} = \frac{\text{variance of pixels in seq. of images}}{\text{variance of pixels in uniform images}}$$

$$SNR[db] = 10 \log_{10} \frac{E_s}{E_n}$$

$$= \frac{\frac{1}{n} \sum_{i,j} (I(i,j) - \bar{I})^2}{\sigma_n^2}$$

where, $\bar{I}$ ⇒ average value of the image
σ_n^2 ⇒ variance in a uniform region

If SNR = 1 ⇒ very bad quality of ~~cam~~ image
because the image has equal amount of noise to signal.

We consider good camera Image, as where signal is 100 times greater than noise.

b)

Gausian Noise	Impulsive Noise
1) Arises in digital images eg. sensor noise	1) is also called as salt & pepper
2) Caused by poor illumination & High Temperature	2) dark pixel in bright region and vice versa.

Median filter Handle impulsive noise better than all other noise. It removes all the sharp skip that is occur due to sudden change in the intensity.

c) Convolution filter = $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

$I = \begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix}$ We apply zero padding

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 2 & 2 \\ 0 & 2 & 2 & 2 \\ & 2 & 2 & 2 \end{bmatrix} * \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$I(0,0) = 0*1+0*1+0*1+0*1+2*1+2*1+0*1+2*1+2*1$

$= 8$

$$\therefore \begin{bmatrix} 8 & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix}$$

new Image after convolution

we have to apply on each pixel.

$I(0,1) = 0*1+0*1+0*1+2*1+2*1+2*1+2*1+2*1+2*1$

$= 12$

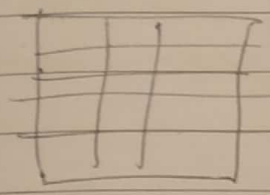
$I(0,2) = 8$ $I(1,2) = 12$ $I(2,2) = 8$

$I(1,0) = 12$ $I(2,0) = 8$

$I(1,1) = 18$ $I(2,1) = 12$

$$\therefore \begin{bmatrix} 8 & 12 & 8 \\ 12 & 18 & 12 \\ 8 & 12 & 8 \end{bmatrix}$$

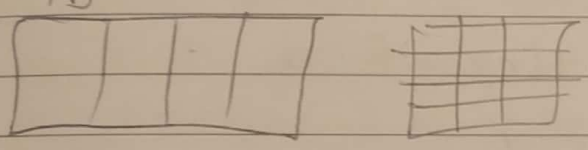
d) For more efficiency the derivative of an image can be convolude with 1D gaussian instead of using 2D gaussian



$M \times N$ $\quad m \times m$

$= M \times N \times m^2$ no. of operation

1D



$M \times N$

$M \times N \times m =$ no. of operation

We have to apply ~~to~~ two 1D gaussian to get same result as one 2D gaussian

$\therefore 2(M \times N \times m)$

$2(M \times N \times m) < M \times N \times m^2$ } It show How efficient 1D gaussian is.

e) 3 different ways to handles boundaries during convolution as follows:

1) <u>Zero padding</u>

to calculate the value at position (0,0) we add zero padding to make it 3*3

0	0	0		0
0	1	2	3	0
0	3	4	2	0
0	2	5	1	0
	0	0	0	

2) <u>mirror replicate</u>

We apply mirror replicate to their neighbour pixel

	4	5		1
4	4	5	1	1
6	6	7	2	2
4	4	5	3	3
	4	5	3	

3) Ignore boundary

we ignore the boundaries. While calculation we take the value of boundaries as 0

1	2	5
3	4	6
4	2	5

f) 3 x 3 smoothing filter

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

filter is mask with positive entries that sum to ~~9~~ $9/9 = 1$

~~the~~ The reason for the sum to be selected as it is, that we don't want to change the intensity of the image after applying this filter

g) $$I_G = \underbrace{I * G}_{\text{2D gaussian}} = \sum_i \sum_j I(i,j)\, e^{-\frac{i^2+j^2}{2\sigma^2}}$$

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

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

$$= I * G_x * G_y$$

~~The results~~

Instead of convoluing with 2D gaussian convolue with 1D gaussian along rows then along columns.

The result we get are same.

The option is efficient is applying 2 1-D gaussian.
because it takes less no. of operation compared to 2D gaussian

$$\underbrace{2MNm}_{\text{two 1D}} < \underbrace{MNm^2}_{\text{one 2D}}$$

Yes, it is possible to implement any 2D filter in this way.

h) $\sigma = 2$ Let the size of filter be m $\sigma \leq m/5$ m = size of filter

$\cancel{G(x) = e^{-x^2/2\sigma^2}}$ $\therefore$ $\cancel{\sigma = 2}$

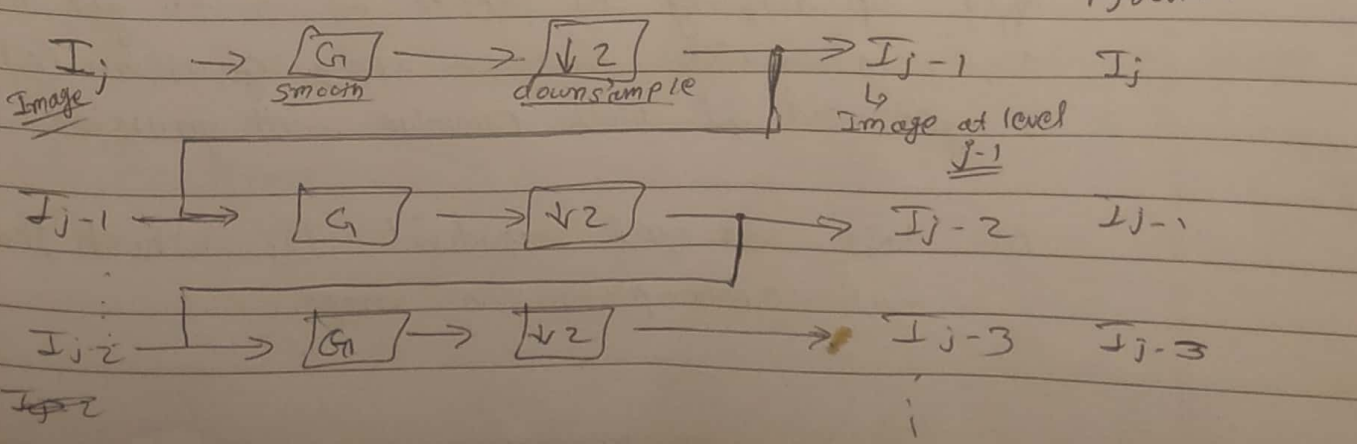
$\cancel{= e^{-x^2/2(2)^2} = e^{-x^2/8}}$

$2 \leq m/5$

$m \geq 10$

$\therefore$ $\boxed{\text{Size of filter} \geq 10}$

i) Pyramid

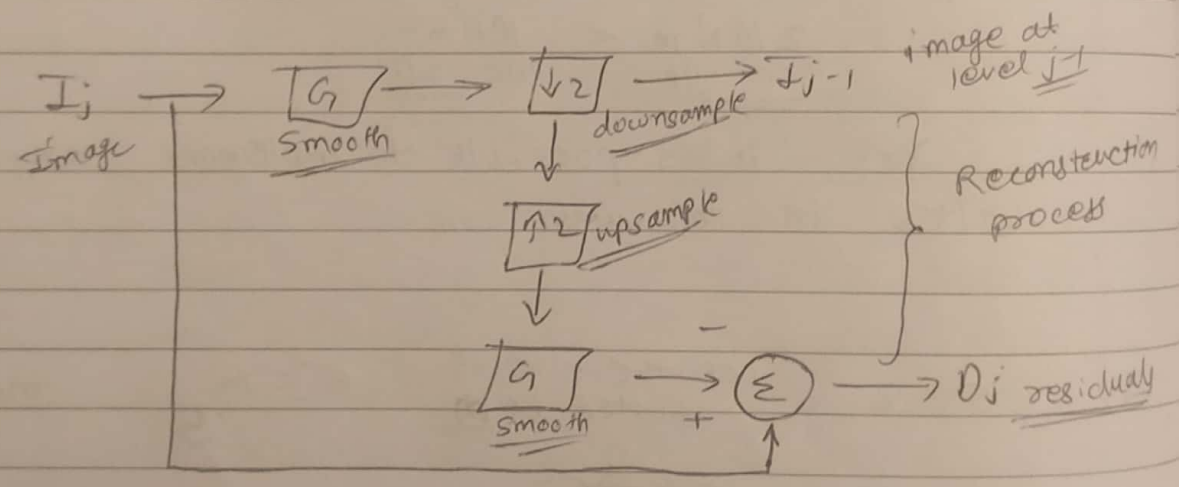


To get gaussian pyramid, 1st we apply gaussian convolution on Image & then we apply downsampling by factor of 2 on the image. Here we get 1st frame of Image.

Then similary we proceed to get every frame of the image.

We build these pyramid for analysis purpose. It's become useful for analysis of each frame. ~~after~~ on

j)



first we apply gaussian, then we apply downsampling on image. Here we lost some information because of downsampling, so we apply upsampling by factor of 2 to get back lost information. Now, we apply gaussian (G) to get rid of block convolve with gaussian.

At last, we get residual D_j which shows small corruption in Image.

~~We use~~

To get reconstructed image we add all residual with low frequency I_0

$$\therefore \left.\begin{array}{l} I_0 \\ \vdots \\ D_{j-1} \\ D_j \end{array}\right\} \text{it gives us complete reconstructed image.}$$

We used Laplacian Pyramid in Image Compression.

~~2 Edge Decti~~

2 Edge Detection :

a) ~~Usef~~ Goal of the Edge Detection:

- We can extract important features from the edges of an image.
  ex. corners, lines, curves

- We get more detailed info of an image

~~Conc~~

Properties of edges:

1) Edge Orientation
   - Edge Normal - unit vector in the direction of maximum intensity change
   - Edge Direction - unit vector ⊥ar to the edge normal

2) Edge Position
   - image position at which edge is located

3) Edge Magnitude
   - How rapid is the intensity variation across the edge along edge normal.

4) Edges should be consistent

5) Edges should be invariant and correspond to scene elem

b)

1) Smoothing
   : suppress as much ~~as~~ noise as possible, without destroying true edges

2) Enhancement
   : apply a filter to enhance the quality of the edges in the image (sharpening)

3) Localization
   : determine the exact location of an edge.

c) Image Gradient:

- Gradient is a vector which has certain magnitude & direction.

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

$$|\nabla I| = \sqrt{I_x^2 + I_y^2} \qquad \theta = \tan^{-1}\left(\frac{I_y}{I_x}\right)$$

1) Forward Difference

$$\frac{\partial}{\partial x} I(x,y) = \frac{I(x+h,y) - I(x,y)}{h} \quad \Big\} \text{ derivative w.r.t. to } x$$

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

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

2) Central Difference

$$\frac{\partial(x,y)}{\partial x} = \frac{I(x+h,y) - I(x-h,y)}{2h}$$

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

$$\Delta x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad \Delta y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

- Why Useful:
  - Magnitude - provides information about strength of the ~~image~~ edge.
  - Direction of gradient is always perpendicular to the direction of the edge.

d) Sobel filter

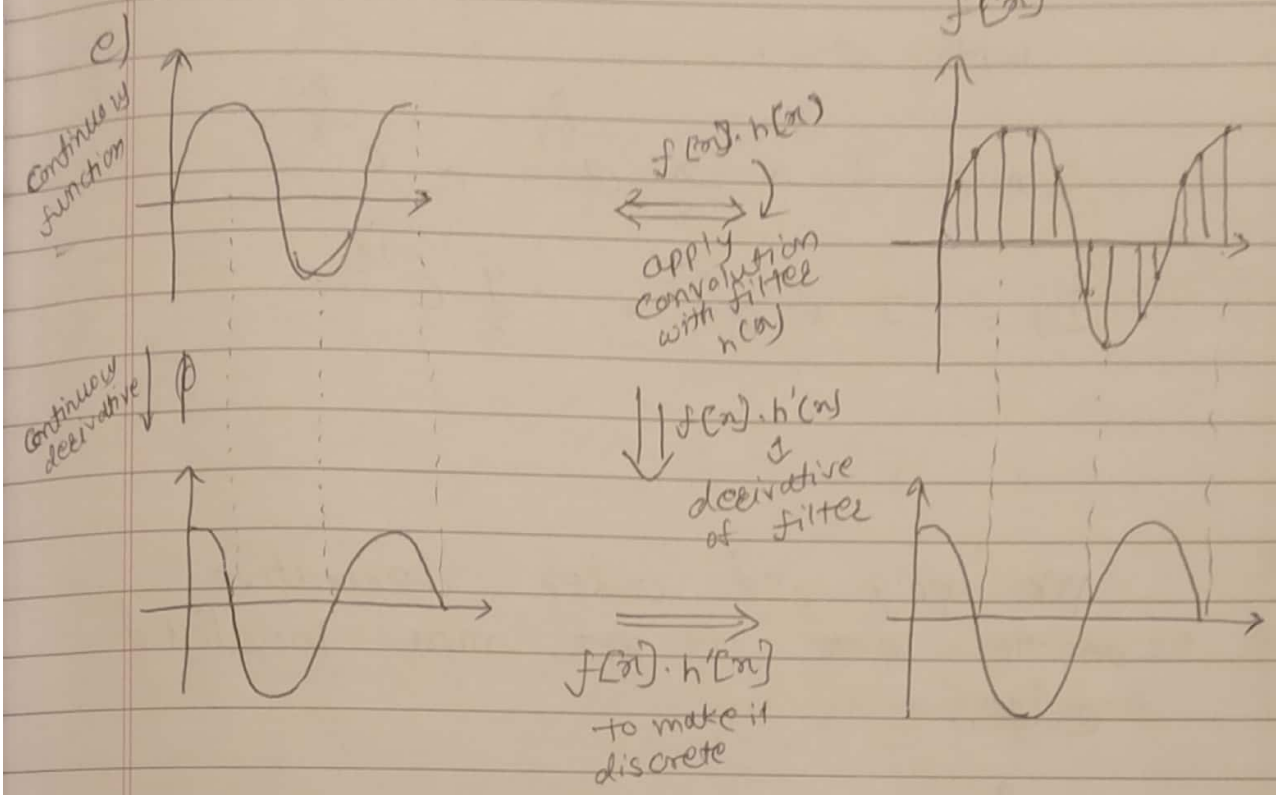
First we do smoothing & then take derivative

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

to smooth — to compute derivative

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

e)



$$I_x = I * G_x$$
$$I_y = I * G_y$$

$$\therefore \; I_x = I * G'[x] * G[y]$$

1D Convolution with Horizontal gaussian derivative

$$G[x] = e^{-x^2/2\sigma^2} \qquad G'(x) = -\frac{2x}{2\sigma^2} e^{-x^2/2\sigma^2}$$
$$= \frac{-x}{\sigma^2} e^{-x^2/2\sigma^2}$$

$$I_x = I * \frac{-x}{\sigma^2} e^{-x^2/2\sigma^2} * e^{-y^2/2\sigma^2} =$$

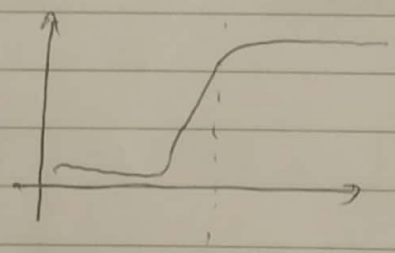
$$I_y = I * e^{-x^2/2\sigma^2} * \frac{-y}{\sigma^2} e^{-y^2/2\sigma^2}$$

with $\sigma = 2$

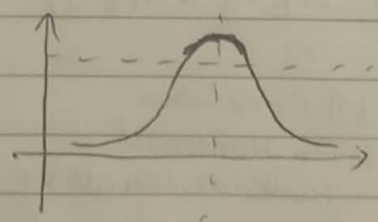
$$I_x = I * \frac{-x}{8} e^{-x^2/8} * e^{-y^2/8}$$

$$I_y = I * e^{-x^2/8} * \frac{-y}{8} e^{-y^2/8}$$

f) We apply 2nd order derivative on the image to get more localized edge.

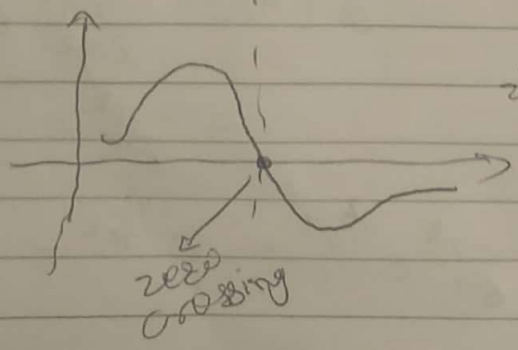


edge



1st order derivative

not localized bcz several edges are above threshold



2nd order derivative

Here all the edges at zero crossing

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

$$= I_{xx} + I_{yy}$$

$$I_{xx} = (I(x+1) - I(x)) - (I(x) - I(x-1)) = I(x+1) - 2I(x) + I(x-1)$$

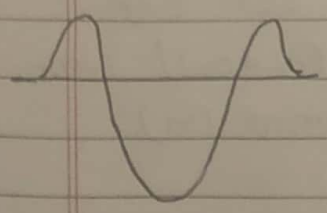
$$= \begin{bmatrix} 1 & -2 & 1 \end{bmatrix}$$

$$\therefore \quad I_{yy} = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

$$\nabla^2 I = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$ laplacian can be implemented using this mask

g) $\sigma = 1$

$$H = \nabla^2 (I * G) = \underbrace{\nabla^2 G}_{\text{LOG}} * I$$



$$G = e^{-r^2/2\sigma^2} \qquad [r^2 = x^2 + y^2]$$

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

1D structure obtained by LOG.

at $\sigma = 1$

$$\nabla^2 G = \frac{e^2 - 2}{1} e^{-r^2/2}$$

Edge Detection using LOG:

1) Compute LOG $\quad H = \nabla^2 G * I$

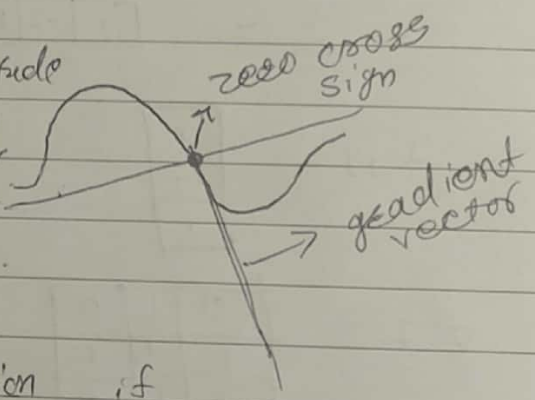
2) Threshold $\quad E(i,j) = \begin{cases} 0 & \text{if } H(i,j) < 0 \\ 1 & \text{if } H(i,j) \geq 1 \end{cases}$

3) mark edges at transition } scan left to right, top to bottom

h) It detect edges at zero crossing of 2nd order directional derivative taken along the gradient

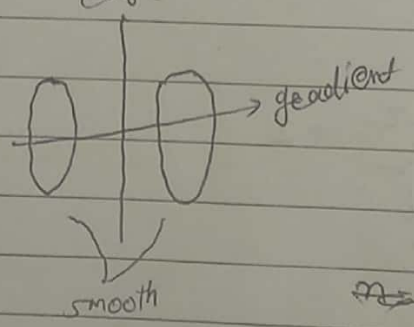
if $|n| > \tau$

if gradient magnitude is above threshold, then we will attempt to detect the edges.



Attempt detection if only $|n| > \tau$

Smooth along edges to preserve edge



$n = \nabla(I * G)$

$\frac{\partial}{\partial n}(I * G) = n \cdot \nabla(I * G)$

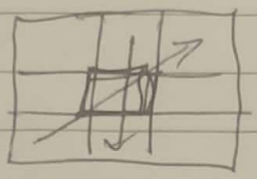
if $|\nabla(I*G)| > \tau$ set edge at maximum of $|\nabla(I*G)|$ along the direction of gradient (n).

i) <u>Non-Maximum Suppression:</u>

It Help us to detect the maximum of gradient in the direction of gradient

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

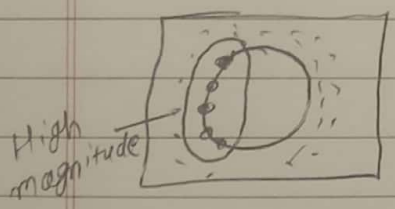
$\theta = \tan^{-1}(I_y / I_x)$ $\theta^* = \text{round}(\theta / 45) * 45$



Compare neighbours after discretization

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

Hysteresis Thresholding:



$\tau_H \Rightarrow$ edge to ~~detect~~ track
Uses High Threshold

Why High Threshold?
$\Rightarrow$ if we keep low threshold, lot of noise edges.

$\tau_L \Rightarrow$ Another start value to keep tracking
~~$\tau_H \Rightarrow$~~

1) Intially array of visited pixels
$v(i,j) = 0$

2) Scan image top to bottom, left to right
if $v(i,j)$ && $|\nabla I| > \tau_H$
start tracking an image

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