Name: Arpit Hasmukhbhai Patel
ID: A20424085
sem: Fall-18
sub: Computer Vision

[1] Noise and filtering

(a) $SNR = \frac{E_s}{E_n} = \frac{\sigma_s^2}{\sigma_n^2} = \frac{1/n \sum_{ij} (I(i,i) - \bar{I})^2}{\sigma_n^2}$

- we can calculate σ_n^2 with variance for multiple frames of a static scene

or

variance in a uniform image region.

- This is how we can calculate signal to noise ratio

(b) Gaussian noise

- It is statistical noise. It has probability density function equal to the gaussian distribution.

* Impulsive noise

- Impulsive noise adds very different values to some specific pixels. The Noise is of two types (i) positive impulsive and (ii) negative impulsive.

⇒ Median fiter handles better impulsive noise.

(c)

1	1	1
1	1	1
1	1	1

filter

2	2	2	2	- -
2	2	2	2	- -
2	2	2	2	- - -
2	2	2	2	- - -
!	!	!	!	

image

0	0	0	0	0
0	18	18	18	18 - - -
0	18	18	18	18 - - -
0	18	18	18	18 - - -
	18	18	18	18 - - -

(d) We need derivative of image convolved with filter so first we do derivative of filter then convolve with an image. It is more efficient compared with the derivative of image convolved with filter

(e) We can handle boundries during convolution in 3 ways:
(i) zero pailing
(ii) Mirror / Replicate
(iii) Ignore.

(f) smoothing filter

$\frac{1}{9}$	1	1	1
1	1	1
1	1	1

→ In this 3×3 filter the sum is 9 so the intensity becomes 9 times so we multiply by 1/9.

(g) Separable Implementation

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

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

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

$$= I * G_x * G_y$$

→ This is two 1D Gaussian

M×N image
m×m filter

one 2D pass : $M \times N \times m^2$
Two 1D pass : $2 \times M \times N \times m$

$2MNm < MNm^2$

So, two 1D pass is efficient

(h) $\sigma \leq \frac{m}{5}$

$m \geq 5\sigma$

for $\sigma = 2$

$m \geq 10$

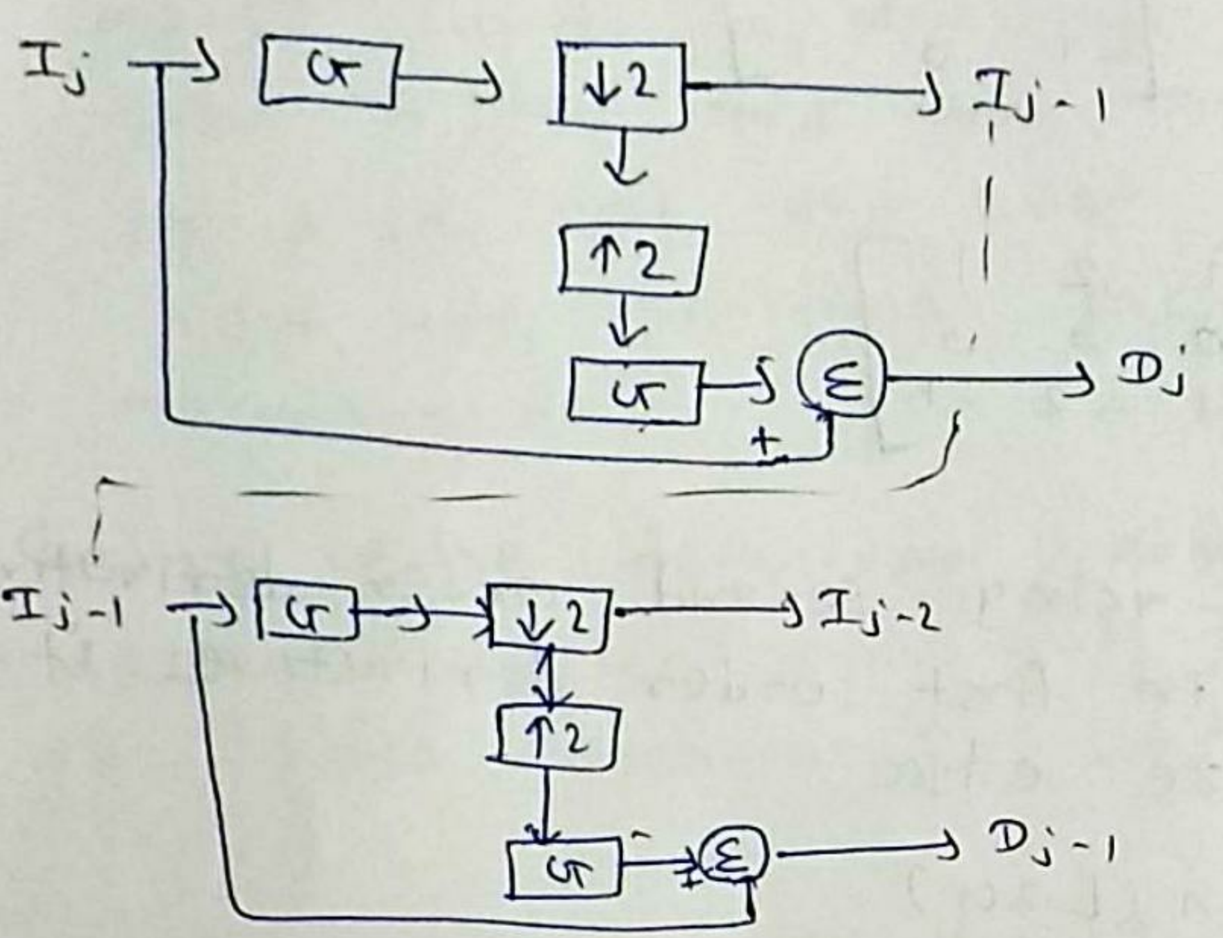
So, the of filter is $m \geq 10$

(i) Gaussian Pyramid is produced by shrinking image.
- Instead of changing window size we can the size of image

$$m^2 + \frac{1}{4}m^2 + \frac{1}{16}m^2 + \cdots < \frac{4}{3}m^2$$

- so it adds only 30% extra pixel.

(j) Laplacia Pyramid



→ This is how laplacian pyramid is produced
→ It is useful for compression.

[2] Edge detection

(a) With edge detection we can get more details and understanding of an image.

⇒ Properties of edge detection.
(1) Scene elements
(2) invariant (illumination, pose, scale, viewpoint)
(3) reliable detection.

(b) edge detection steps are:
(i) Smooth to reduce noise (without affecting edges)
(ii) Enhance edges
(iii) Detect edges
(iv) Localize edges (for precise location of edges)

(c) sobel filter & Gaussian

- Image gradient is a directional change in the intensity or color in image.
- It is used to detect edges.

(d) first smooth and then take derivative

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

smoothing derivative

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

smoothing derivative

(f) We can localize edge using second order derivative by zero crossing but in first order derivative it is difficult to localize edges.

(g) Laplacian of Gaussian (LoG)

- smooth with a gaussian before applying laplacian

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

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

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

for $\sigma = 1$

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

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

* 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 of transitions $0 \to 1$, $1 \to 0$
(scan left to right & top to bottom)

h) Standard Edge detection (Sobel filter)

→ It has 3×3 filter where gradient magnitude is for each pixel, a number giving the greatest rate of change in the light intensity in direction where intensity chang fastest.

→ Canny edge detection goes further by removing noise with a low pass filter first then sobel filter and then non-matimym suppression to pick out the best pixel for edges where there are multiple possibilities in a local neighbourhood.

⇒ Canny edge detection conditions.

✗ attempt detection only if gradient magnitude is large enough ($|m| > \tau$)

(j) Non-maximum Suppression.

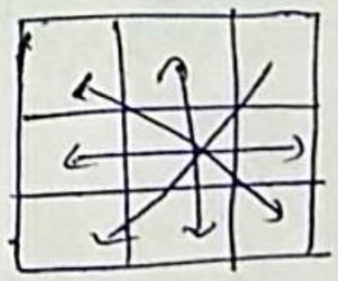
- Local maximum of gradient magnitude in direction of gradient

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

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

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

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



compare neighbors after discretization

* Hysteresis Thresholding

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

1) Initialize array to visited 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 neighbors in direction orthogonal to ∇I such that $|\nabla I| \leq \tau_e$

(e)

Element of filter for more accurate derivative with $\sigma \geq 2$

$m = 15$

$m \geq 50$

$m \geq 10$

$G(x) = G(y)$

$\frac{1}{4.988}$

0.002	0.011	0.04	0.135	0.32	0.606	0.88	1	0.88	0.6	0.32	0.135	0.04	0.011	0.002
							$x=0$							

$G'(x) = G'(y)$

-7	-6	-5	-4	-3	-2	-1	$x=0$	1	2	3	4	5	6	7
0.0035	0.016	0.05	0.135	0.24	0.303	0.22	0	-0.22	-0.303	-0.24	-0.135	0.05	-0.016	-0.0035