Assignment-2

Name : SAI MANOHAR VEMURI
CWID : A20514848
Course Number : CS 512
Semester : SPRING 2023

1. Noise & filtering

a) SNR is a measure of quality of an image that represents ~~useful information~~ ratio of the useful information to the amount of background noise present in an image.
Noise can be in various forms such as Gaussian Noise, Salt & Pepper noise etc. Higher SNR values higher image quality.

To Compute SNR

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

Where,

σ_n^2 = it represents the variance for multiple frames of a statistic scene.

b) Gaussian Noise:- It ~~has~~ is a type of noise that has a normal distribution. It is caused by random fluctuations ~~in~~ in the Camera sensor.

→ The noise is spread evenly across the image with most values closer to the mean.

→ Median filter, Mean filter, Gaussian filter etc. can be used to remove the Gaussian noise.

Impulse Noise:- Impulse Noise, also known as Salt & Pepper Noise which is seen by fluctuations in pixel intensities. These pixels fluctuations result in pixel values that are significantly different from the surrounding pixels.

→ The main difference b/w Guassian Noise and Impulse Noise is the distribution.

Since
Median filter is best for handling the impulse Noise because it replaces the pixel value with the median of the surrounding pixels. While preserving the edges of the image.

c) If we have an image with the value 2 in each cell and have a 3×3 Convolution filter with having 1's in its entries, we get "18" in the feature map after the Convolution operation.

d) In order to do this operation more efficiently we use Linear Seperable filters where 2D Convolution operation can be broken down into two seperate 1D convolutions.

e) There are three different ways to handle boundaries during the Convolution:-

1. Zero padding:- In Zero padding we add Zeros values along the border of an image.
2. Mirror padding:- It is another way to increase the image size by reflecting the values along the border.

3. ignore padding :- In this type, the Convolution operation is performed only on the valid pixels of the image and does not add values to the border. It leads to shrink the feature map and ~~for~~ may loose the information near the edges of the image.

f) The basic Smoothing filter is defined as the average the values of its neighbouring pixels

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

The sum of all the entries in this filter would be equal to "1".
If we dont sum it as, everytime we apply Convolution operation, the image gets brighter and brighter.

g) Instead of Convolving with 2D Guassian, use 1D Guassian along the rows and then along the columns.

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} \cdot e^{-\left(\frac{x^2}{2\sigma^2}\right)} \cdot \frac{1}{2\sigma^2} \cdot e^{\left(\frac{-y^2}{2\sigma^2}\right)}$$

Where,
σ is the Standard deviation of Guassian function.
This seperately applies Convolution operation along x axis and along y axis.

~~Instead of~~

$M \times N$ image

$m \times n$ filter

One 2D pass $= M \times N \times m^2$

two 1D pass $= 2 \times M \times N \times m$

$$2MNm < MNm^2$$

Instead of Doing one big Convolution operation with large Sigma value we can do two Convolution operations with small Sigma values. The bigger the Sigma, the bigger the filter should be to capture the entire gaussian filter width

h)

$m \geq 5\sigma$

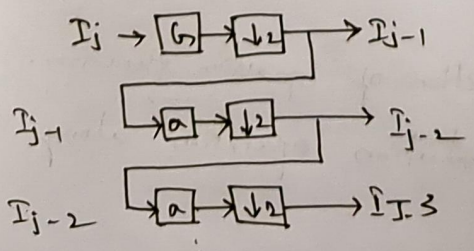
if $\sigma = 2$

$m \geq 10$

The filter m size should be ≥ 10

i) Gaussian image pyramid is produced by applying Gaussian filter to an original image and downsample it.

$$g_l(i,j) = \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m,n)\, g_{l-1}(2i+m, 2j+n)$$



The reason we generate pyramids is because features at some resolution may go undetected at one resolution can be easily detected at some other resolution.

The amount of procesing needed depends upon the image, Kernel Size and no. of levels.

Suppose we have $M \times N$ image size and $m \times m$ Kernel. We need to n level gaussian pyramid.

$= (MNm^2) \times n$

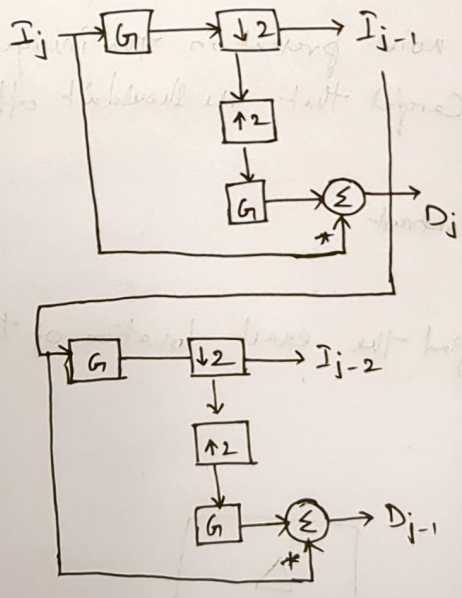
which is more than what we do for single image but at every level we downsample the image. So the image size will be reduced at each level.

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

$$\therefore \frac{4}{3}m^2 - m^2 = \frac{1}{3}m^2$$

So we need 30% additional procesing done in pyramid.

(i)



* First we Convolve the original image with the Gaussian filter
* Then we downsample it and again upsample it again and do the Convolution again.

* Then take the difference b/w original image and the final Convolved image

* We repeat the same process just as f_{j-1} level.

* Laplacian image pyramids is most widely used in image Compression

2. Edge Detection :-

a) Edge detection is useful for finding the boundaries of objects in the image. We find the gradients of an image and then find the magnitude of these gradients for each pixel which represent the strength and direction of gradient represent direction of edge.

Properties :-

* Correspond to Scene elements (no noise)
* invariant (illumination, pose, viewpoint, Scale)
* reliable detection

b) 1. Smoothing :- To reduce the noise present in the image but we have to be careful that we should it affect the edges

2. Enhance Edges :- Increases the Contrast

3 Detect Edges

4. Localize Edges. :- helps us to find the exact location of the edges.

c) Image gradient :-

Image :- $I(x, y)$

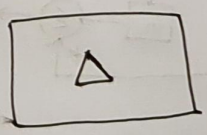


Image gradient :-

$$\nabla I(x,y) = \begin{bmatrix} \frac{dI}{dx} \\ \frac{dI}{dy} \end{bmatrix}$$

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

The most commonly used filters to find the image gradients are

1. Sobel filter :- It computes the gradients in the x & y direction separately. Used to detect the edges.
2. Prewitt filter :- It is similar to Sobel filter but has a different weight scheme.

d) for the Sobel filter, let's compute the image gradients for the x & y direction separately.

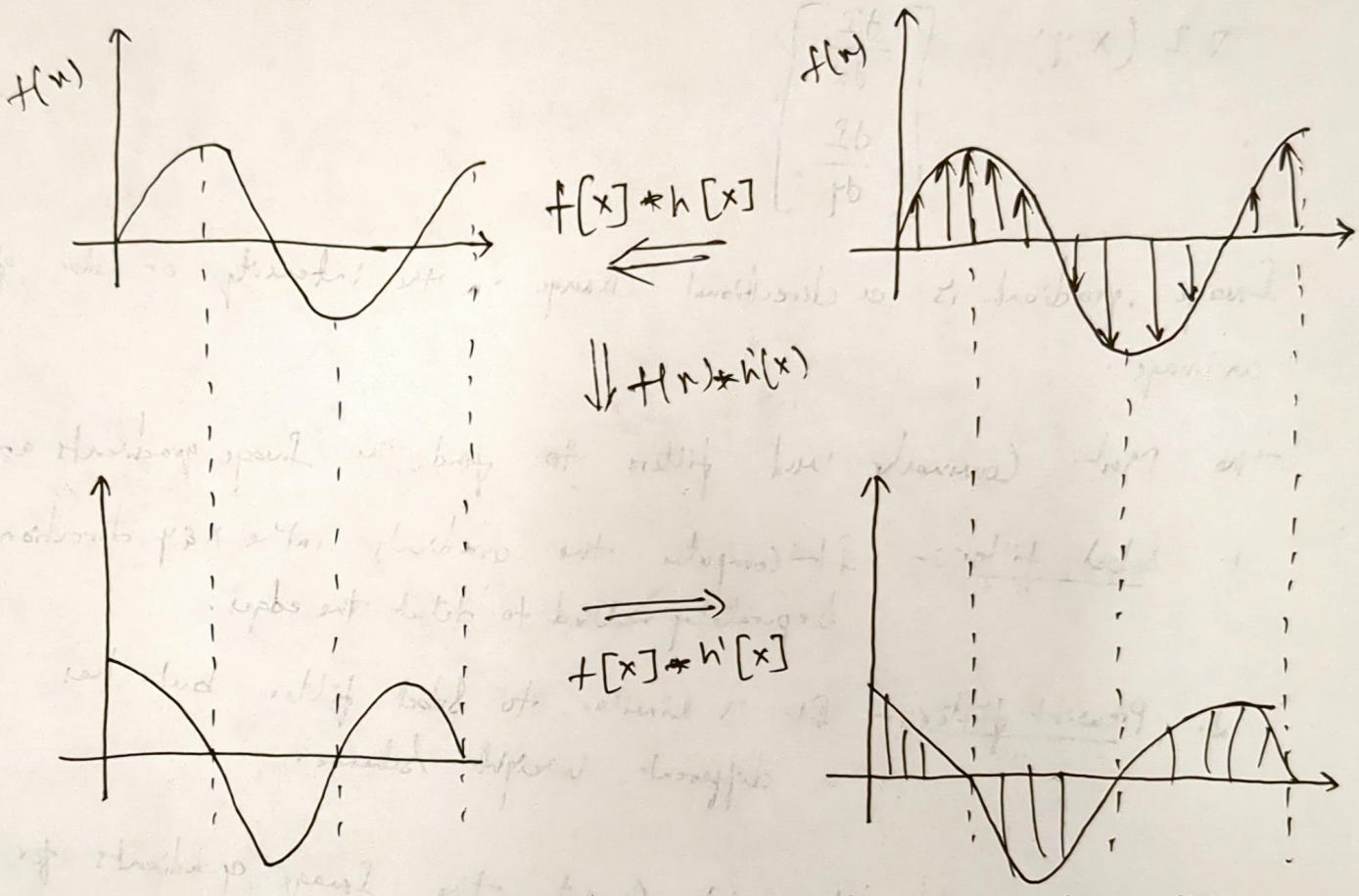
lets first smooth the image and find the derivative.

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

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

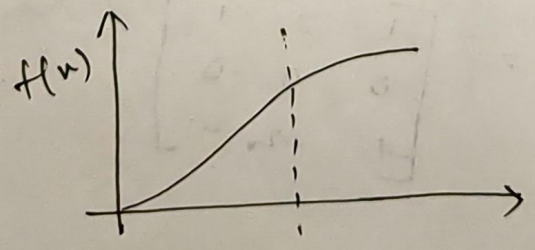
↑ Smoothing ↑ Derivative

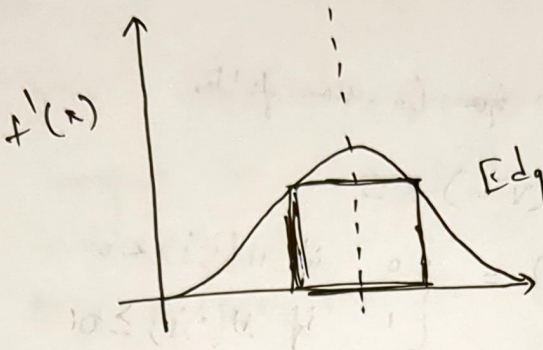
e) More Accurate derivatives :-



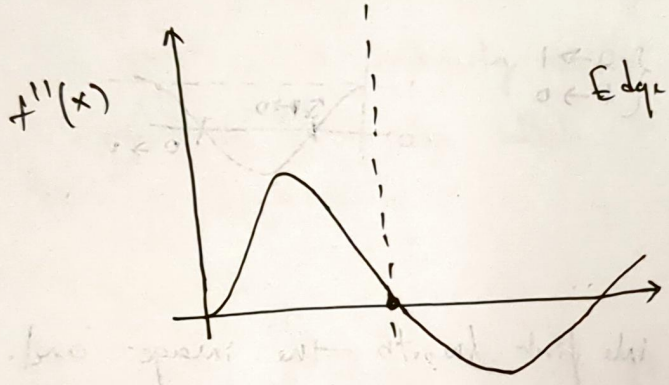
We can take f(x) and reproduce the continous function. Then take its derivative & sample it to get $f'(x)$

f) We can localize the edges using the Laplacian of Gaussian which computes the second order derivatives.





Edge by threshold after computing first order



Edge by zero Crossing

q) Given $\sigma = 1$

$$LoG = \nabla^2 G$$

$$G = e^{-\frac{r^2}{2\sigma^2}}, \quad \nabla = e^{-\frac{r^2}{2\sigma^2}} \left(\frac{-2r}{2\sigma^2}\right)$$

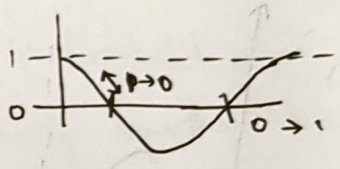
$$\nabla^2 G = \frac{r^2 - 2\sigma^2}{\sigma^4} \cdot e^{-\frac{r^2}{2\sigma^2}}$$

for $\sigma = 1$

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

Steps to Compute the LoG :-

* First we Smooth the Image with ~~Sobel~~ Gaussian filter
* Next Compute the LoG $\Rightarrow$ $H = (\nabla^2 G) * I$
* Compute the threshold $\Rightarrow$ $E(i,j) = \begin{cases} 0 & \text{if } H(i,j) < 0 \\ 1 & \text{if } H(i,j) \geq 0 \end{cases}$
* Mark edges at transitions $\begin{cases} 0 \to 1 \\ 1 \to 0 \end{cases}$



h) Standard Edge detection :- We first Smooth the image and find the the first order derivatives which represent rate of change of the image intensity in a specific direction which are used to detect the edges. They are Computed in x & y directions Seperately and magnitude is Calculated which indicates the strength of the edge. It produces thick edges.

Canny Edge Detection :- After Smoothing we Compute the gradient Magnitude using Sobel or other gradient operators. Then we perform Non maximum Supression which Compares the each pixel with its two neighboring pixels in the gradient direction. Then pixels which have the maximum magnitude are picked and rest all Supressed.

Condition for detecting the edge in canny edge detection :-

* detect edges only if gradient magnitude is large enough ($|n| > t$)
* Compare the magnitude ~~in the gradient orientation~~ of its neighboring pixels in the gradient orientation and find the max of them and rest all suprened.
* Use two threshold values : high threshold and low threshold values. edge pixel with intensity higher than high threshold value is classified as strong edge while pixels intensity lower than low threshold value is suprened.

i)

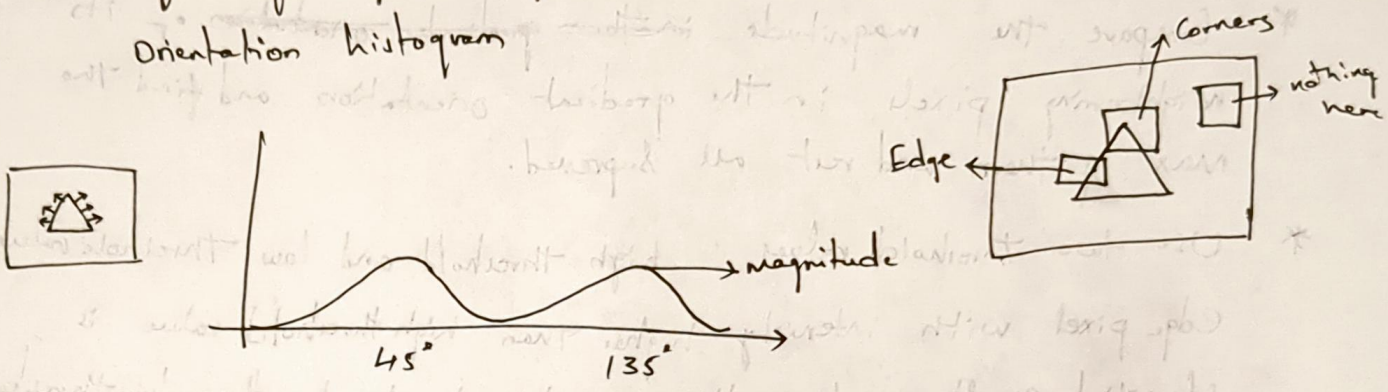
Non-Maximal suprention :- It is used to thin the edges.

By taking the ~~the~~ neighboring pixels, we compare the magnitude in the gradient orientation.

Hysteresis thresholding :- We take two threshold values ~~and~~ high & low values. ~~pixel magnitudes where are intensities~~ Any edge pixel above this level is classified as strong edge pixel and any pixel which is lower is classified as weak edge pixel.

3. Corner Detection

a) After finding the gradient directions we plot it in the Orientation histogram



It will check for the magnitude increment/decrement in the histogram.

Algorithm :-

1. find the correlation matrix of the gradients in the window.
2. find Eigen values of correlation matrix
3. detect Corner in window if the eigen values are above the threshold.

b) PCA can be used to detect these corners by analysing the variations in the image gradients. Here principle components are the eigenvectors of the correlation matrix and the represent the directions of maximum variance in the data.

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

$$= \sum_i (v^T g_i)(g_i^T v) = \sum_i v^T g_i g_i^T v$$

$$= v^T \left(\sum_i g_i g_i^T\right) v = v^T C v$$

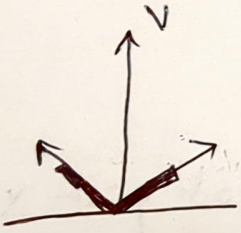
c) Correlation matrix of Image gradients

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

x_i^2	y_i^2	$x_i y_i$
0	0	0
0	1	0
0	4	0
0	9	0
0	16	0
1	0	0
1	1	1
1	4	2
1	9	3

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

d) λ_1 & λ_2 are the eigen values of the gradient matrix for corner detection, this λ_1 & λ_2 should be large.



$\lambda_1 \lambda_2 > z$ should be large.

e) Non Maximum Supression :-

i) Compute the $\lambda_1 \lambda_2$ for all window

ii) Select windows with $\lambda_1 . \lambda_2 > z$ & sort them in descending order

iii) Then select the top of the list as corners & delete all other corners in its neighborhood from the list.

iv) Stop once deleting x'l. of the points as corners.

f) Harris Corner detection

i) Compute the correlation matrix for each window

ii) Then find the Corner measure

$$C(C) = \det(C) - k\,tr^2(C)$$

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

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

$= 0$ if $k=0.5$ (Corner detection)

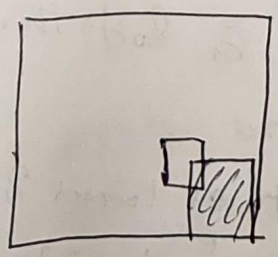
$= 0$ if $k=0$ (edge detection)

Hence using k we can avoid using λ_1, λ_2

g) Corner localization

To determine if P is the corner Connect each point x_i to P & project the gradient at x_i onto $(x_i - P)$.

The "best" P will minimize the sum of all projections



Corner localization formula

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

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

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

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

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

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

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

↑ Correlation matrix

$\lambda_1, \lambda_2 > t \Rightarrow$ C should be non Singular since we detected the corner in the window.

(A) (i) Split each patch into cells

(ii) Create orientation histogram in each cell using the gradient directions

(iii) This histogram is used to find the dominant direction.

Requirements for Good characteristics of feature points

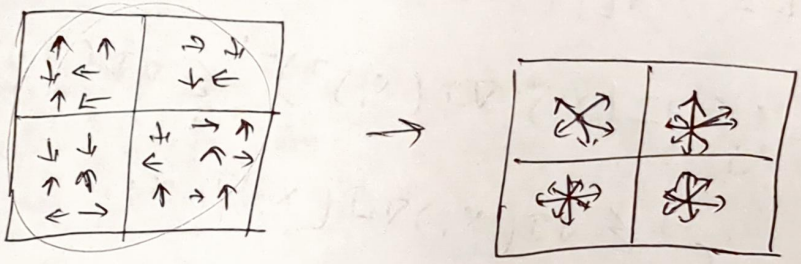
i) translation invariance

ii) Rotation invariance

iii) Scaling invariance

iv) Illumination invariance.

i) Scale Invariant feature Transform (SIFT)



Steps:-

i) break window into sub windows

ii) ~~Align the~~ Compute gradient direction in each ~~corner~~ subwindow

iii) Combine gradients to form an orientation histogram

iv) Align histogram based on dominant direction.