Assignment 2

Name: Akshay Jain

Student ID : A20502846

Course Number: CS512

Semester : Spring 2022

SOLUTIONS

① Noise & Filtering.

Ans 1.a.) Signal to noise ratio is used in science & engineering that compares the level of a desired signal to the level of background noise. SNR is defined as the ratio of signal power to the noise power.

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

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

$$SNR\,[dB] = 10 \log_{10}(E_s/E_n)$$

i.e. 10 dB $\Rightarrow$ E_s is 10 times larger than E_n.

Ans 1 b) Gaussion noise is statistical noise having a probability density function equal to that of a normal distribution: also called as the Gaussian Distribution.

On the other hand the impulsive noises do not follow normal distribution. They are short duration instantaneous unwanted sharp noises that degrade the image.

Median filter is better because average filter is more sensitive to extremen values.

Ans 1 c)

filter

Image

After applying convolution filter on image having

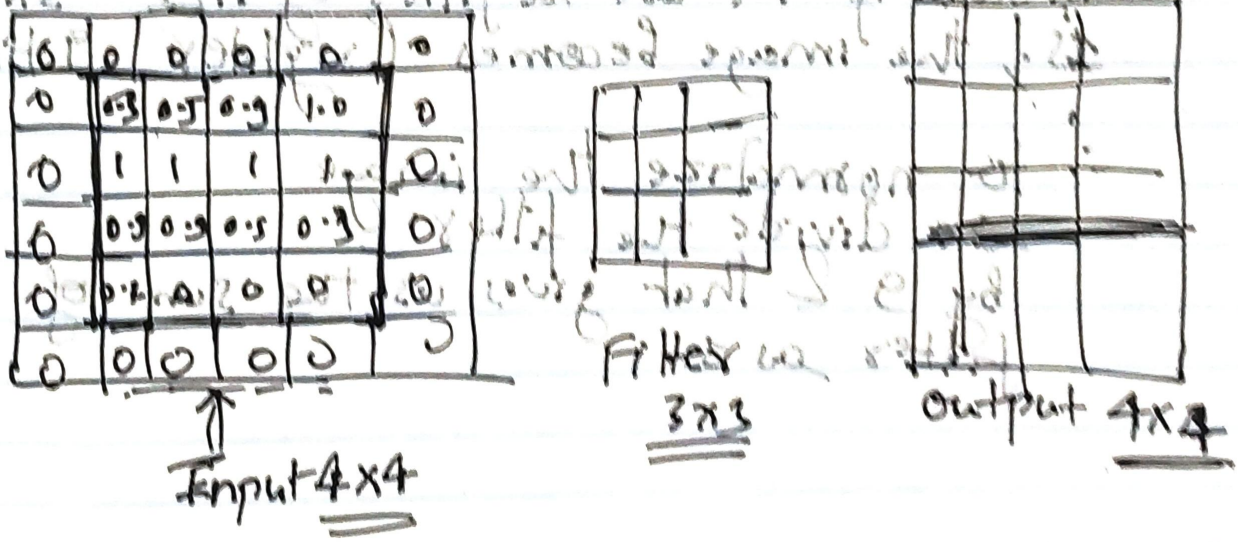
Note: (c) explanation

The values will not change on applying convolution filter but we can normalize the final value.

Ans: d) As convolution is a linear operation if we want the derivative of an image convolved with a filter we can first compute the derivative of the filter & then convolve it with the image.

Ans: e) Three different ways to handle boundaries during convolution are:

1) Zero Padding



2) Ignore the boundary values.

3) Mirror or Replicate.

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

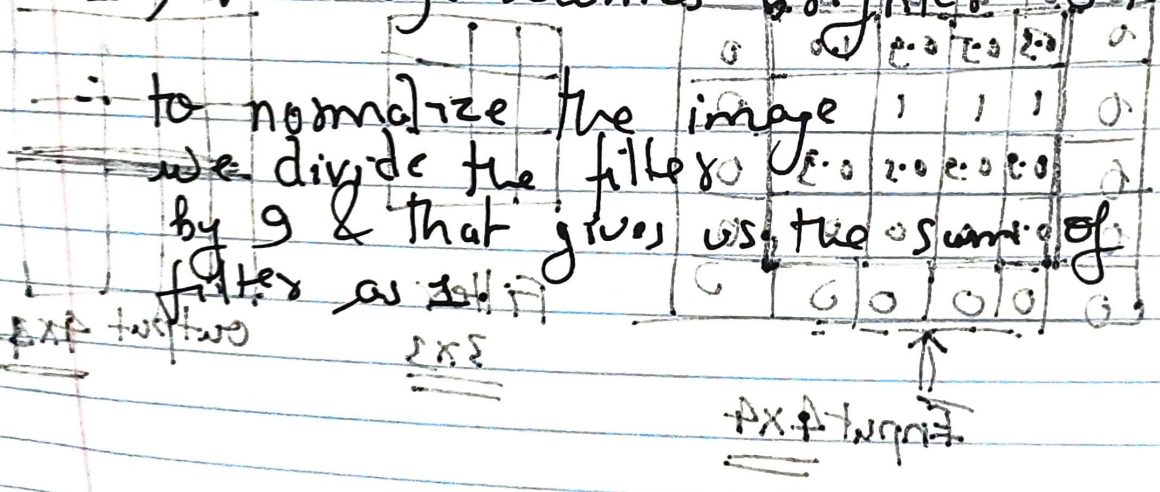
Ans 1) f)

Smoothing filter = 1/9

1	1	1
1	1	1
1	1	1

the sum of all values in the above filter is 9 because if we use a filter of size 3x3 & all 1s, the image becomes brighter 9 times.

∴ to normalize the image we divide the filter by 9 & that gives us the sum of new filter as 1.



Ans 1) g) Separable implementation

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

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

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

Instead of convolving with a 2D ~~convolution~~ Gaussian, convolve with 1D Gaussian along rows then along columns.

~~Ans 1) h)~~ So for $m \times n$ image & $m \times m$ filter

One 2D pass : $m \times n \times m^2$ operations.

Two 1D pass : $2 \times m \times n \times m$ operations.

Since, $2 \times m \times n \times m < m \times n \times m^2$ separable implementation is more efficient.

Ans I) ii) As $\sigma \leq \frac{m}{5}$

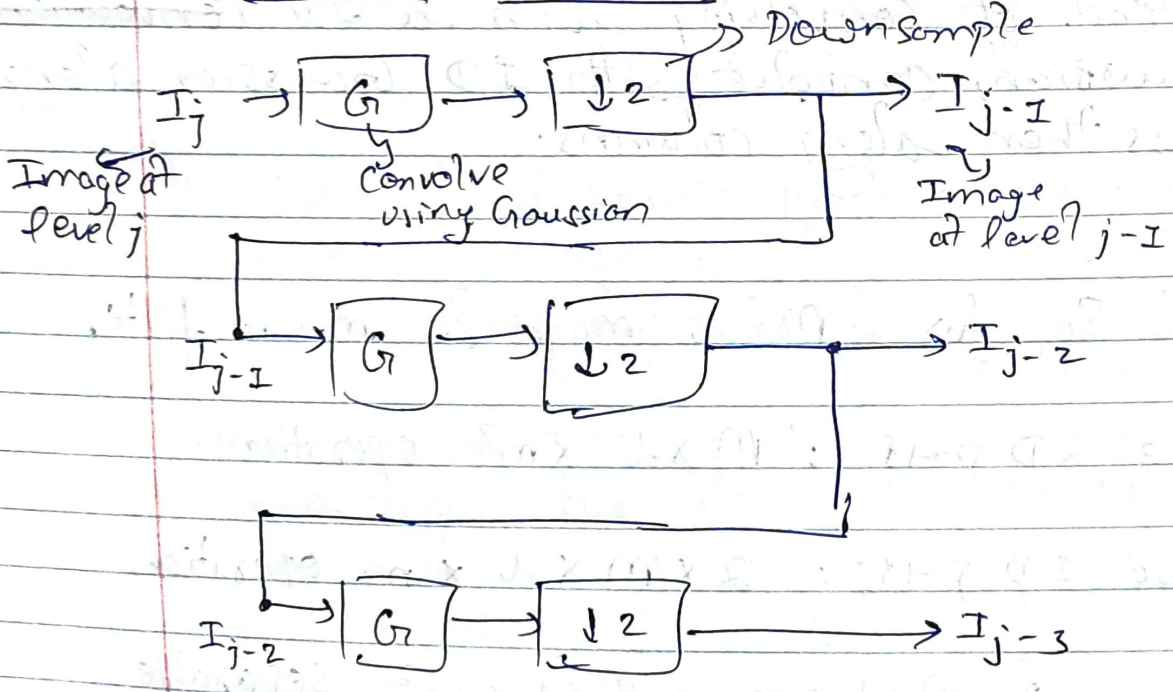
or $m \geq 5\sigma$

For $\sigma = 2$

$$\boxed{m \geq 10}$$

∴ The size of the filter must be 10.

Ans I) i) Gaussian Pyramids



Pyramid

I_j

I_{j-1}

⋮

I_0

Useful for analysis.

The number of layers is given as

$j = \log_2 m$.

Gaussian Pyramids are useful for multiple scale analysis.

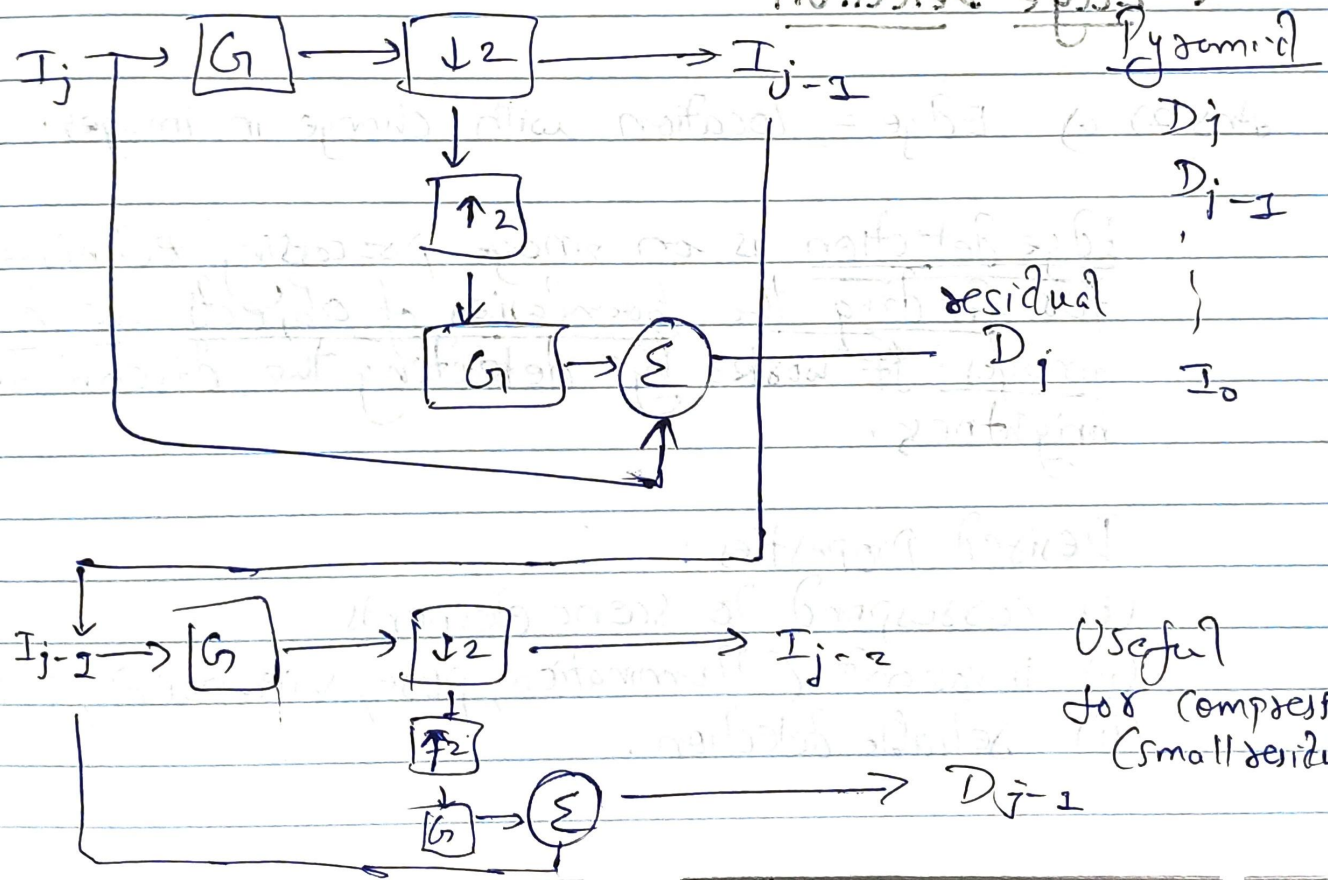
Total amount of additional processing done in Pyramid ⇒

$$m^2 + 1/4\, m^2 + 1/16\, m^2 + \cdots < 4/3\, m^2$$

$$\therefore\ 4/3\, m^2 - m^2 = 1/3\, m^2$$

So, we require 30 % extra additional processing compared to a single image.

Ans 1) j) Laplacian Pyramids



To form Laplacian Pyramids, we convolve the original image, downsample it by a factor of 2, again upsample it by a factor of 2, convolve again & then take the difference between original image & the latter one we get the residual D_j

For the next residual (D_{j-1}) we repeat the same process described above the image at level I_{j-1}

Laplacian pyramids are useful for compression (small residuals).

2. Edge Detection

Ans 2) a) Edge = location with change in images.

Edge detection is an image processing technique for finding the boundaries of objects within images. It works by detecting the discontinuies in brightness.

Desired properties:
(i) correspond to scene elements.
(ii) invariant (illumination, pose, viewpoint, scale).
(iii) reliable detection.

Ans 2) b) Edge Detection Steps :

1) Smooth to reduce noise (without affecting edges).
2) Enhance edges
3) Detect edges
4) Localize edges.

Smoothing : It helps to reduce the noise in the image & this make it easy to identify discontinuities.

Enhancing : It will slightly increase contrast.

Localization : It helps to find the precise location of the edges.

Ans 2) c) Image Gradient

Image : $I(x, y)$

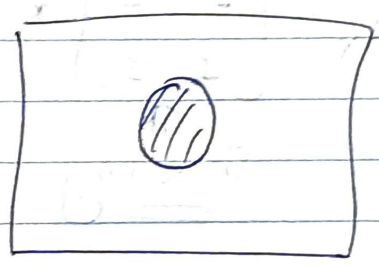
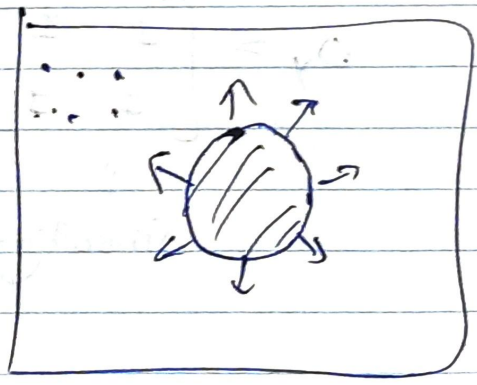


Image Gradient :

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

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

$$= \begin{bmatrix} I_x \\ I_y \end{bmatrix}$$



Two filters used for image gradient are:

1) Sobel filter → (Smooth & then take derivative)
It works by calculating the gradient of image intensity at each pixel within the image.
It finds the dir^n of the largest increase from light to dark & the rate of change in that direction.

2) Gaussian filter

It is a linear filter & usually used to blur the image or to reduce noise.

Ans 2) d) Sobel filter

First Smooth & then take the derivative (as mentioned in notes)

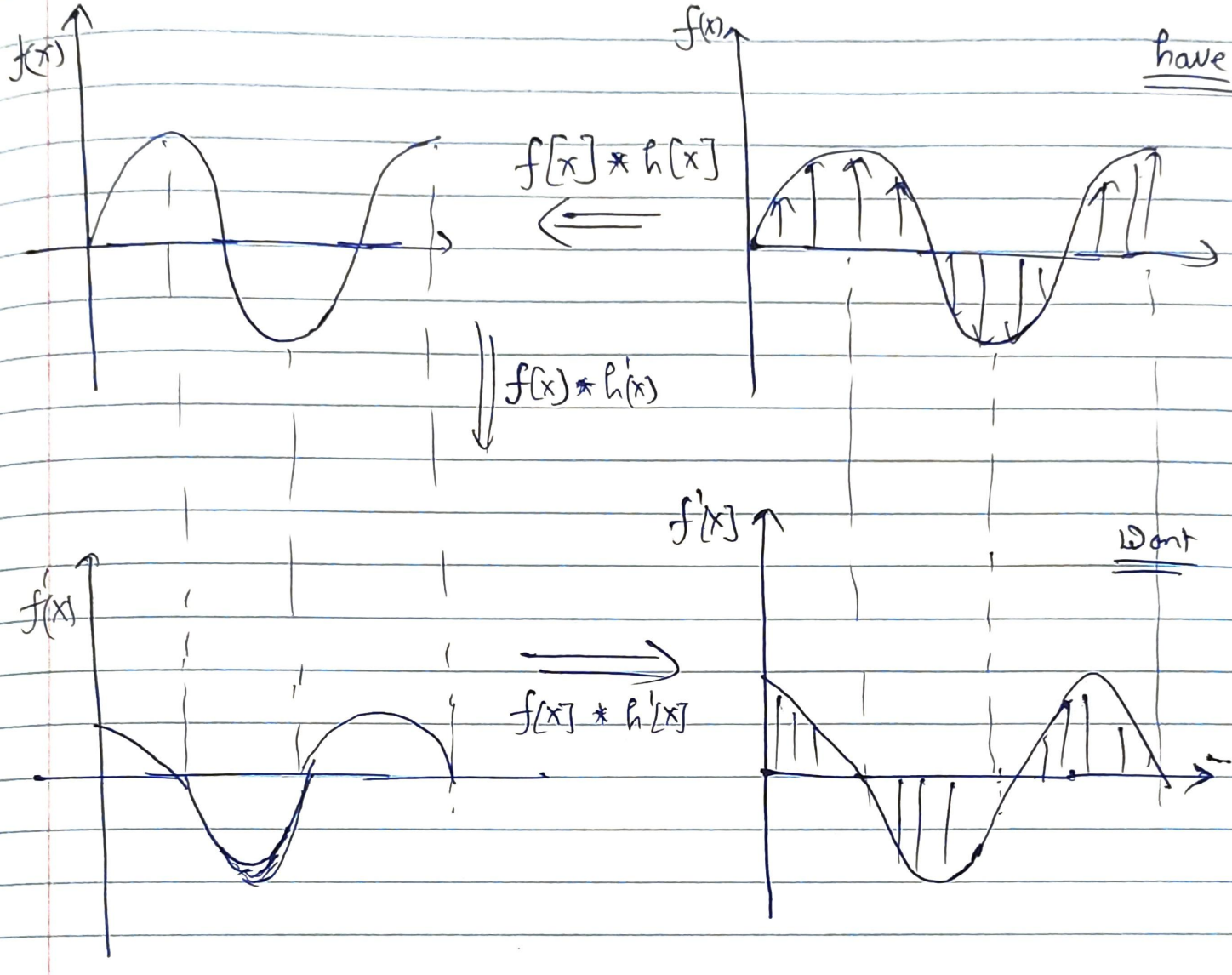
$$\Delta_x = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} * \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

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 ⇑

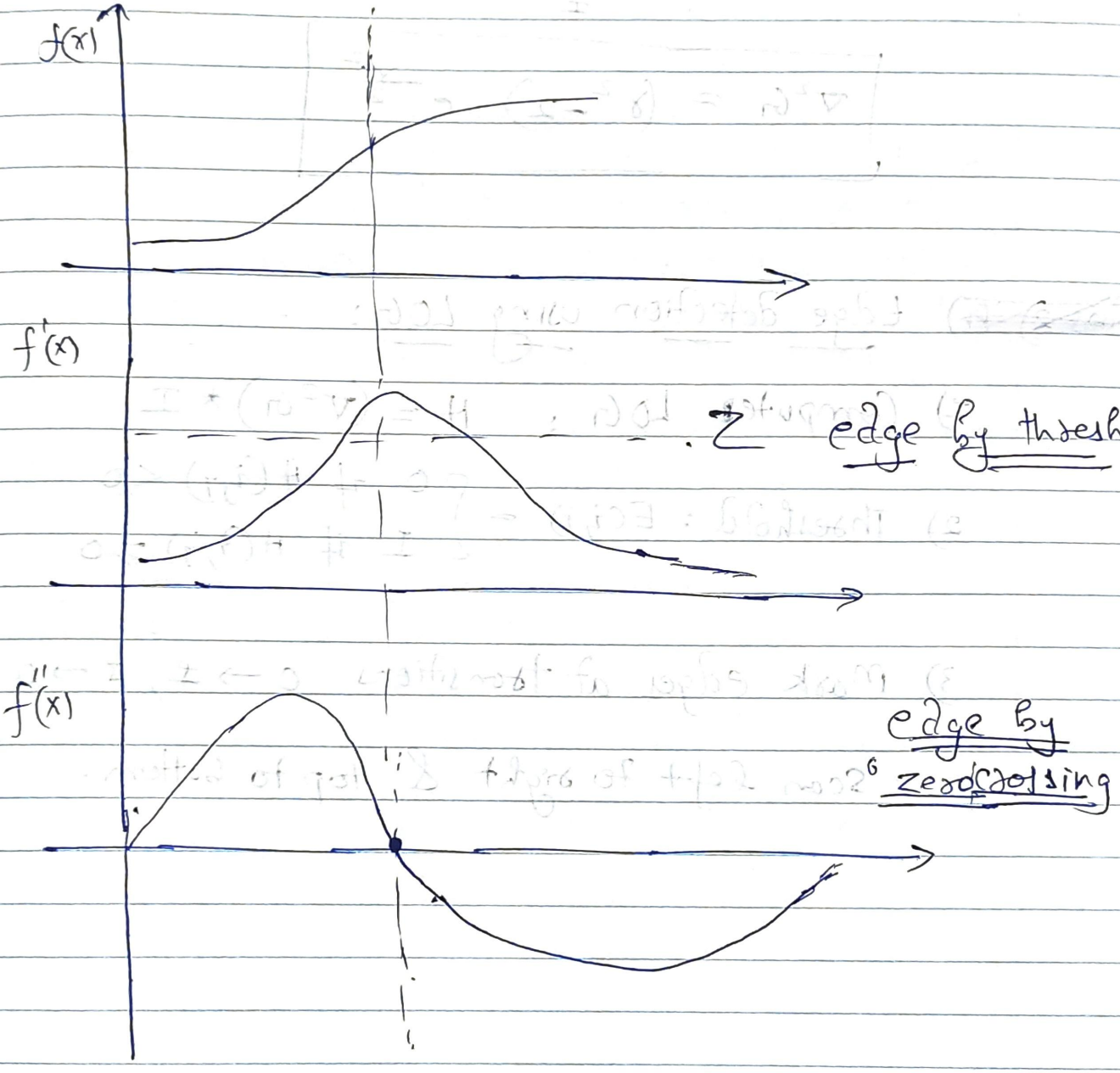
Ans 2) e) More accurate derivatives



We can take $f(x)$ & reconstruct a continuous function & then take its derivative & sample it to get $f'(x)$.

Ans 2) f) → The first order derivative detects the edges by looking for minimum & maximum of the image.

→ While the second order derivative searches for zero crossings.



Ans 3) a) Given $\sigma = 1$.

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

(for $\sigma = 1$)

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

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

b) 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 & top to bottom.

Ans 2) h) Standard Edge Detection:

It uses 3x3 filters, whose each pixels get the number giving the greatest rate of change in light intensity in the direction where intensity is changing fastest by calculating gradient magnitude for each pixel.

Whereas in:
Canny edge detection: It removes noise with a low pass filter first then apply a sobel filter. Next it performs non-maximum suppression to pick out the best pixel for edges when there are multiple possibilites in a local neighbourhood.

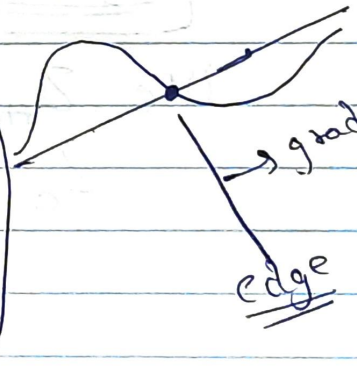
Conditions for detecting edge in canny:

1) Attempt detection only if gradient magnitude is large enough ($|n| > z$)

Note: Canny edge detection: detects edges at zero crossings of second order directional derivative taken along the ~~edges~~ gradient.

n = gradient

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



2) Smooth along the edges to preserve edge.

3) Alternative to zero crossing of

$\frac{\partial^2}{\partial n^2}(I * G)$ is maximum at $\frac{\partial}{\partial n}(I * G)$

Ans 2) i) **Non-Maximum Suppression**

Need: Local maximum at gradient magnitude in direction of gradient.

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

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

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

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

Hysteresis Thresholding

Use Z_H to start tracking &

Z_c to continue $(Z_H > Z_c)$

1) Initialize array of visited pixels.

$V(i,i) = 0$

2) Scan image T-B, L-R:

if $|V(i,i)|$ && $|\nabla I| > Z_H$ start tracking on edge.

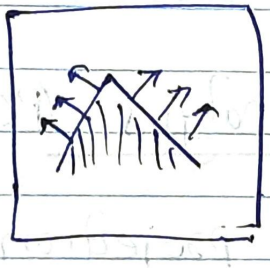
3) Search for additional neighbors in direction orthogonal to ∇I such that $|\nabla I| > Z_L$

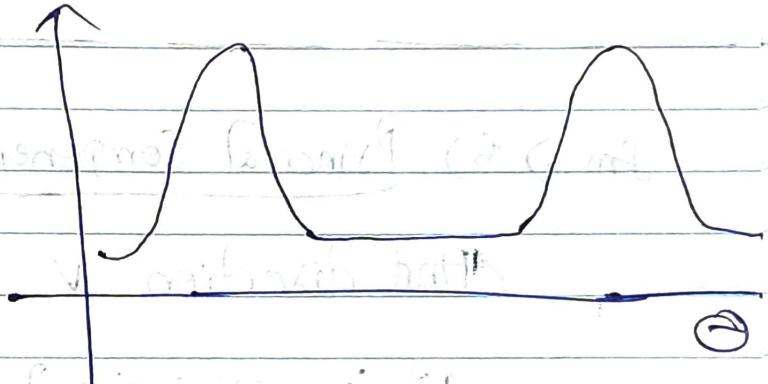
~~3. Edge Detection~~

3. Corner Detection

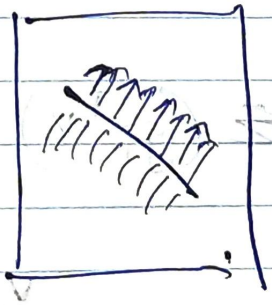
Ans 3) a)

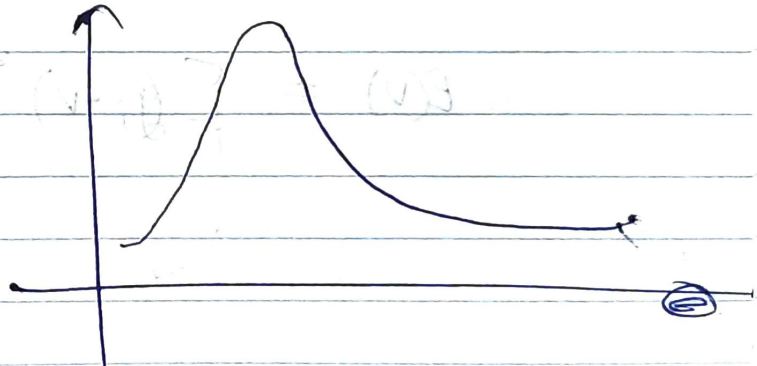
Corner





edge.





Corner = more than one direction in orientation histogram.

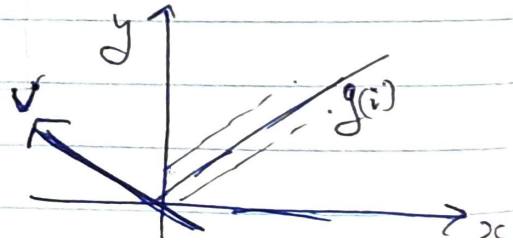
Given a window with image gradients, we want to decide whether it is a corner or edge:

1) Find correlation matrix of gradients at in local window.
2) Find edge values at correlation matrix.
3) Detect corner in window if eigen values are sufficiently large.

Ans 3) b) Principal Component Analysis (PCA)

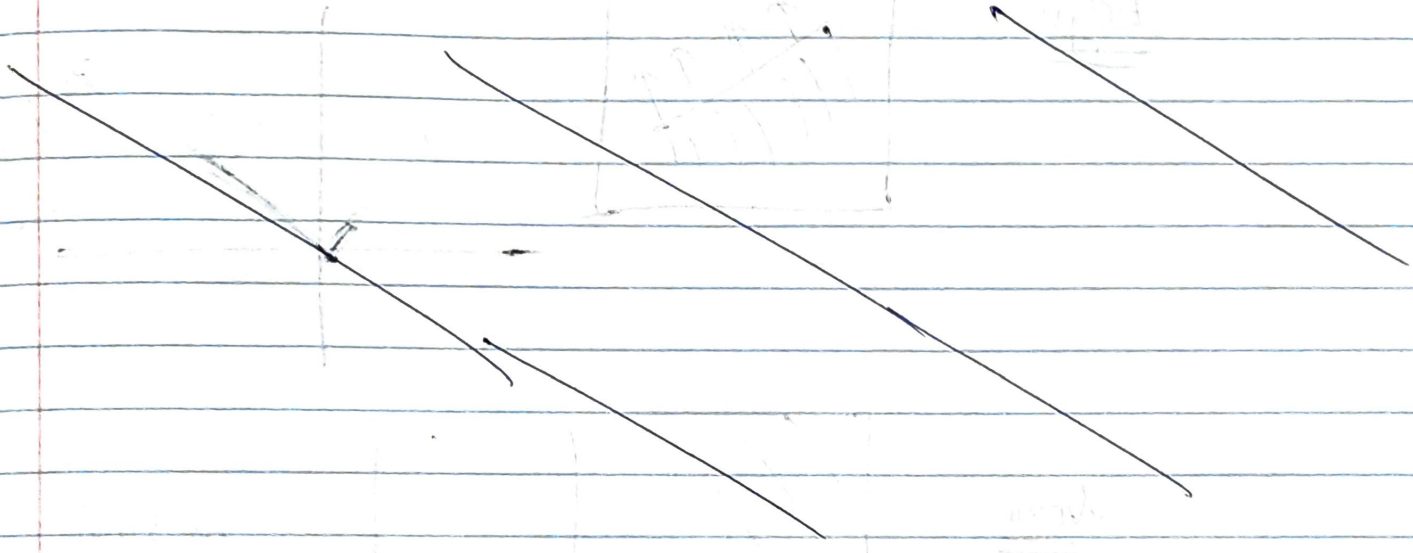
Find direction v s.t. projection of $\{g_i\}$ onto v is minimized.

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



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

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



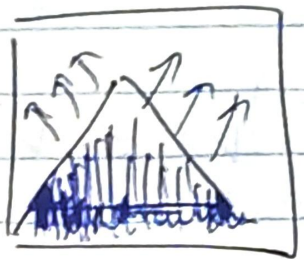
Ans 3) c) Correlation Matrix

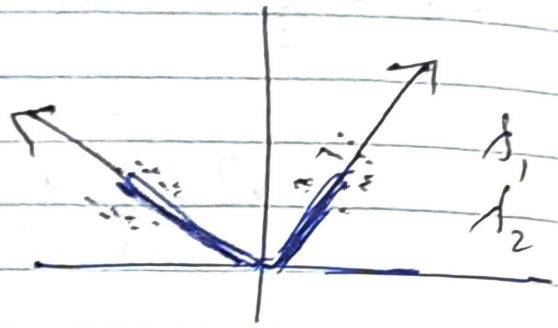
$$C = \sum_i g_i g_i^T = \begin{bmatrix} \Sigma x_i^2 & \Sigma x_i y_i \\ \Sigma x_i y_i & \Sigma y_i^2 \end{bmatrix}$$

$$= \begin{bmatrix} 1+1+1+1 & 1+2+3 \\ 1+2+3 & 1+4+9+16+ \\ & 1+4+9 \end{bmatrix}$$

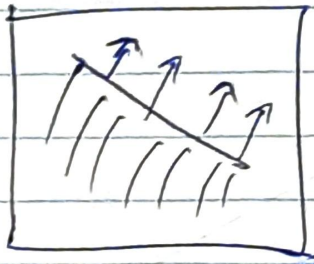
Ans 3) d) "Corner Detection"

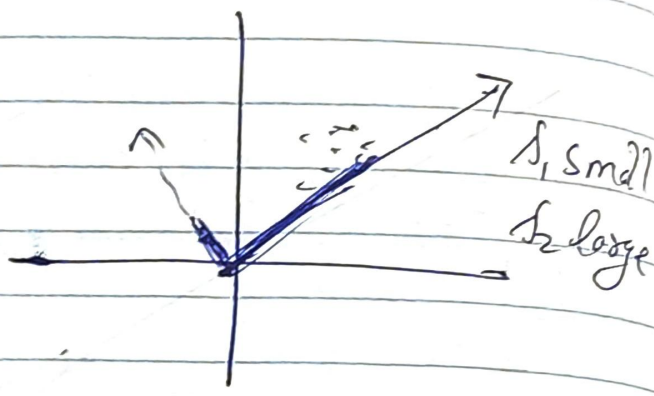
Corner



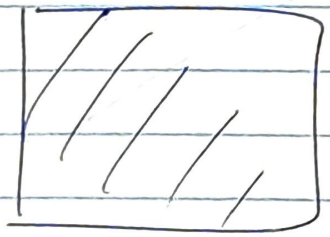


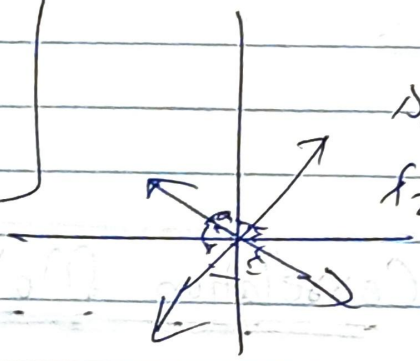
Edge





None





Corner if $\lambda_1 \cdot \lambda_2 > Z$

Ans 3) e) Non-maximum Suppression

1) Compute $\lambda_1 \cdot \lambda_2$ for all windows.
2) Select windows with $\lambda_1 \cdot \lambda_2 > Z$ & sort in decreasing order.

3) Select the top of the list as corners & delete all other corners in its neighborhood from the list.

4) Stop once detecting x % of the points as corners.

Ans 3) f) Harris Corner Detection

1) Compute correlation matrix C for windows.

~~2) (correlate)~~

2) Find Cornerness Measure

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

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

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

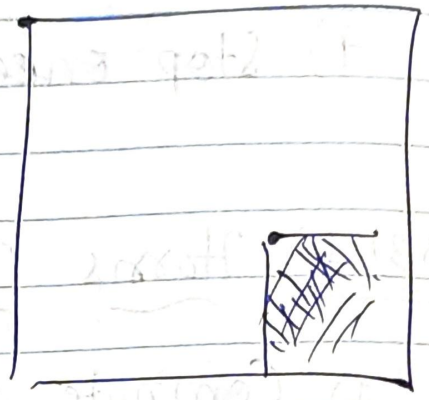
$\Downarrow$ 0 if $k = 0.5$ (Corner Detection)

$\Downarrow$ = 0 if $k=0$ (edge detection.)

Hence, using k we can avoid using λ_1 & λ_2.

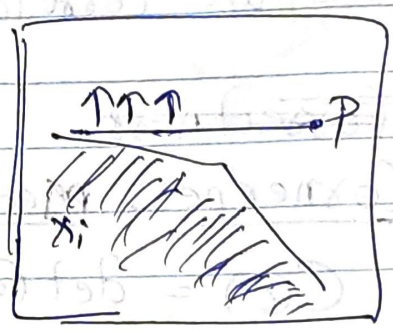
Ans 3) g) Corner Localization

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 & project the gradient at x_i onto $(x_i - P)$.

The "best" P will minimize the some of all projections.



Corner localizations Formula

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

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

$$= \sum_i \underbrace{(x_i - P)^T}_{1\times 2} \underbrace{\left(\nabla I(x_i) \nabla I(x_i)^T \right)}_{2\times 2} \underbrace{(x_i - P)}_{2\times 1}$$

(2×1 and 1×2 multiply to 2×2)

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

p^* : location of corner

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

C : correlation matrix.

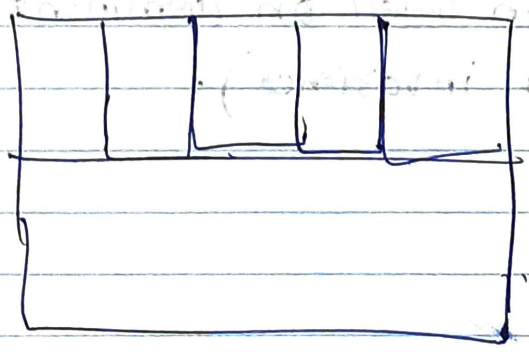
Since, we detected corner in the window

- $\lambda_1, \lambda_2 > z \Rightarrow C$ must be non-singular

---

Ans 3) b) Histogram of Oriented Gradients (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 of gradient magnitude).
3) Concentrate orientation histograms.

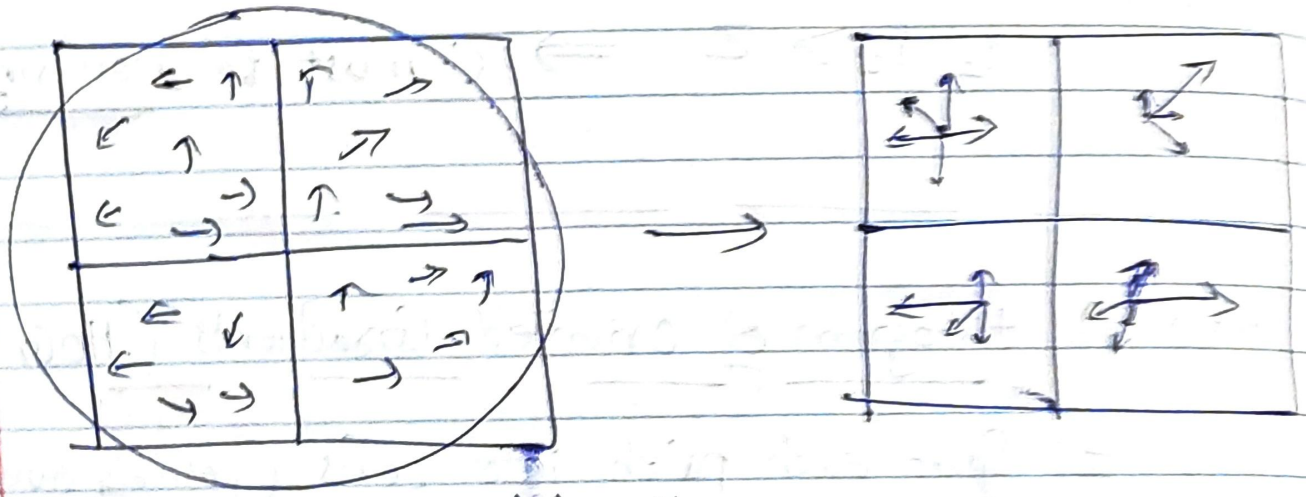


eg) 3x3 cell blocks where each cell has 6x6 pixels.

Requirements for a good characterization of feature points:

1) Translation invariance.
2) Rotation invariance.
3) Scale invariance
4) Illumination invariance

Ans 3) i) Scale Invariant Feature Transform (SIFT)



SIFT feature computation steps:

↳ break window into subparts.
↳ Compute gradient direction in each.
↳ Combine the gradients to form an oriented histogram.
↳ Align histogram based on dominant direction. (Rotation invariance).