4/2/25/ Tuesday.

Computer Vision:

Computer vision is a field of artificial intelligence (AI) that allows computers to see and understand the visual world.

Geometric Primitives:

2D Point:

- How to project a point from one space to another.
- Euclidean Space ⟶ Projective space.

Image, $I = (x, y)$ ↓ inhomogeneous $\quad \tilde{I} = (\tilde{x}, \tilde{y}, \tilde{w})$ ↓ homogeneous.

- 2D points represented using homogenous coordinates.

$$\tilde{X} = (\tilde{x}, \tilde{y}, \tilde{w}) = \tilde{w}(x, y, 1) = \tilde{w}\bar{x}$$

↓ scaling.

Why do we use augmented matrix representation?

$$\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots \end{bmatrix}$$

↘ form of augmented matrix

$(x, y, 1)$ → we are trying to capture an image at point (x, y) with the distance b/w camera & img being 1 unit. If it is at 3 units, the whole thing is ~~sc~~ scaled by 3 ($\tilde{w} = 3$).

↳ scaling.

2D Lines:

$\tilde{l} = (a, b, c)$

corresponding line equation:

$$\bar{x} \cdot \tilde{l} = ax + by + c$$

Intersection of two lines:

$$\tilde{x} = \tilde{l}_1 \times \tilde{l}_2$$

Line joining two points:

$$\tilde{l} = \tilde{x}_1 \times \tilde{x}_2$$

Cross product is used in both cases because we need vector.

★ $\tilde{a} \times \tilde{b} = [a] \times b$ → the vector we get is the line intersecting the lines $\tilde{a}$ & $\tilde{b}$.

$\tilde{a} \times \tilde{b}$: & another vector

$[a] \times b$: skew symmetric matrix. × vector

Q. How to make ~~vector~~ line $\tilde{l}$ to skew symmetric matrix?

↳

$\tilde{l} = (x, y, z)$

$$[\tilde{l}] = \begin{bmatrix} 0 & z & -y \\ -z & 0 & x \\ y & -x & 0 \end{bmatrix}$$

5/2/25/Wednesday.

eg: $l_1 \rightarrow y=1$

$\hookrightarrow (0 \quad 1 \quad -1)$

$\downarrow$

$0xx + 1xy + -1 = 0$

$\Rightarrow \quad y - 1 = 0$

$y = 1$

$l_2 \rightarrow x = 2$

$\hookrightarrow (1 \quad 0 \quad -2)$

$\downarrow$

$1xx + 0xy + -2 = 0$

$x - 2 = 0$

$x = 2.$

$l_3 \rightarrow$ line intersecting l_1 & l_2 ?

$\tilde{l}_1 \times \tilde{l}_2 = [\tilde{l}_1] \times \tilde{l}_2$

skew symmetric matrix form of l_1

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

$$= \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}$$

The point of intersection is $\rightarrow (2,1)$

augmented part.

Q. What do we get from getting the Icular vector to both the lines?

↳ We get the direction of intersection of both the lines. This is why we are doing cross product.

2D Transformations:

1. Translation (2 DoF)

↳ t_x + t_y.

- shifting in x-y plane.
- $x' = x + t$

OR

$$x' = [I \quad t]\bar{x}$$

$$\bar{x}' = \begin{bmatrix} I & t \\ 0^T & 1 \end{bmatrix}\bar{x}$$

I → identity matrix

↳ (3x3) x (3x1) = (3x1)

transformation matrix. | original point $\bar{x}$ | transformed point $\bar{x}'$

Translation matrix

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

2. Euclidean: (3 DoF)

1 DoF + 2 DoF = 3 DoF

- Combining rotation and translation
- $x' = Rx + t$

OR

$$x' = [R \quad t]\bar{x}$$

R → Orthonormal 2x2 rotation matrix.

$$\bar{x}' = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}\bar{x}$$

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$|R| = 1$

$R^TR = RR^T = I$

↳ orthonormal.

$$\underset{3x1}{\bar{x}'} = \underbrace{\begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}}_{3x3} \underset{3x1}{\bar{x}}$$

3. Similarity (4 DoF)

2 DoF + 1 DoF + 1 DoF = 4 DoF

- Translation & then scaled rotation
- $x' = sRx + t$ ↳ we are doing uniform scaling so 1 DoF. If it were non-uniform then 2 DoF.

OR

$$x' = [sR \quad t]\,\bar{x}$$

$$\bar{x}' = \begin{bmatrix} sR & t \\ 0^T & 1 \end{bmatrix} \bar{x}$$

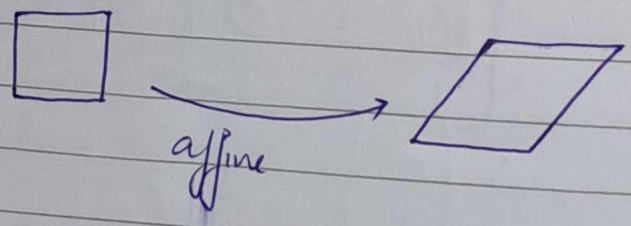
s - arbitrary scale factor.

4. Affine transformation (6 DoF)

- $x' = Ax$
- Affinity matrix → has six elements → 6 DoF

$$A = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix}$$

- When you apply shear, it causes affine transformation



- Parallelism is maintained. Orientation may be changed but lines which were parallel before would remain parallel after the transformation.

- $\bar{x}' = A \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix} \bar{x}$

5. Projective transformation / Homography (8 DoF)

- operates on homogeneous coordinates.

- $\tilde{x}' = \tilde{H} \tilde{x}$
  - ↳ arbitrary 3x3 homogeneous matrix. (the last element (h_{33}) is 1 so the DoF is 8 ~~3~~ not 9).

but lines [illegible]
parallel after the transformation.

- $\bar{x}' = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix} \bar{x}$

5. Projective transformation / Homography (8 DoF)

- operates on homogeneous coordinates.
- $\tilde{x}' = \tilde{H}\tilde{x}$
  - ↳ arbitrary 3x3 homogeneous matrix. (the last element (h_{33}) is 1 so the DoF is 8 ~~9~~ not 9).

12/02/25/Wednesday.

3D Transformations

Transformation	Matrix	DoF	Preserves	Icon.
translation	$[I \; t]_{3\times4}$	3	orientation	
Euclidean	$[R \; t]_{3\times4}$	6	lengths	
Similarity	$[sR \; t]_{3\times4}$	7	angles	
Affine	$[A]_{3\times4}$	12	parallelism	
Projective	$[\tilde{H}]_{4\times4}$	15	straight lines.	

Photometric Image Formation:

Sensor Functions:

1. Photoelectric conversion
2. Charge accumulation
3. Transfer Signal
4. Signal Detection
5. Analog to Digital conversion

↑ Functions done in order.

Lens → Image Sensor → ISP → Interface I/O → Image data

CMOS Camera.

Q. How to convert analog to digital?
↳ 1. Sampling
2. Quantization
3. Encoding

1. Sampling:

~~Digitalizing~~
Digitizing along (x, y) coordinates
Making grids

2. Quantization:

Digitizing along the amplitude.
↳ intensity of the image.

Q How many greyscale levels?
↳ 8 bit → 2^8 grayscale level quantization.
excluding white & black there are 254 grey scale levels.
255 ↓ ↳ 0

2. Quantization:

Digitizing along the amplitude.
↳ intensity of the image.

Q. How many grayscale levels?
↳ 8 bit → 2^8 grayscale level quantization.
excluding white & black there are 254 grey scale levels.
255 ↓ ↳ 0

17/2/25 Mon

Camera Calibration:

- process of determining specific camera parameters in order to complete operations with specified performance measure.

1. Intrinsic or Internal Parameters

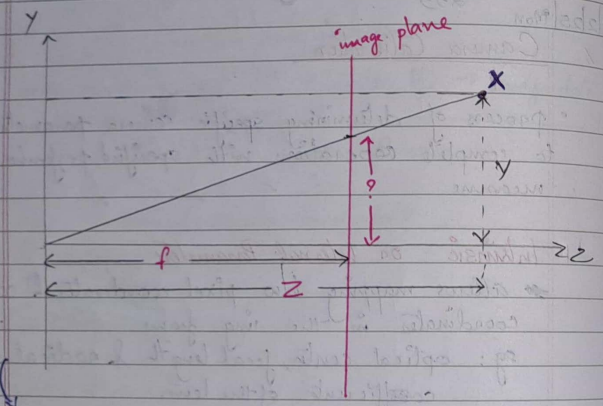
- allows mapping btw pixel coordinates & camera coordinates in the img frame.
  eg: optical center, focal length & radical distortion coefficients of the lens.

2. Extrinsic or External parameters

describes orientation & location of the camera.
Refers to rotation & translation of camera

Pinhole camera: • black box white screen & a pin hole.
• get inverted image.
• projecting object (3D) to a 2D plane.
• Thus we get a transformed 2D vector.

focal length: distance from camera center to the image plane.



$$[X \quad Y \quad Z]^T = \left[f \cdot \frac{X}{Z} \quad f \cdot \frac{Y}{Z}\right]^T$$

$$\frac{f}{z} = \frac{y}{f}$$

• The division by z captures perspective projection, meaning objects further from the camera appear smaller.

Camera Matrix:

A camera is a mapping b/w the 3D world & a 2D image.

$$x = P X$$

x: 2D image point; P: camera matrix; X: 3D world point

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix}_{3\times4} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

homogeneous matrix — camera matrix — homogeneous world point.

Camera Matrix ↓ decomposed into 2 matrices

Principal point coordinates; External calibration matrix

$$P = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}_{3\times4} \Rightarrow P = K[I|0]$$

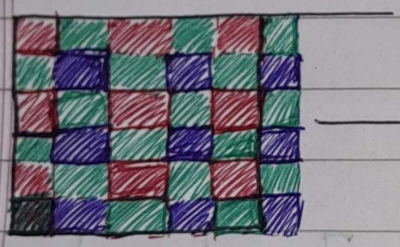
3×3 ↳ Internal calibration matrix.

☆ The camera is already aligned with the real world coordinates hence only the external calibration matrix is always $[I;0]$.
If it is not aligned, we can apply rotation & translation to align it.

☆ The camera is already aligned with the real world coordinates hence why the external calibration matrix is always [I|0].
If it is not aligned, we can apply rotation & translation to align it.

18/02/25/Tuesday.

- A digital camera separates R, G & B components using Bayer filter placed over the image sensor. Each pixel only captures one color (R, G or B).



→ Bayer filter

But you notice that some information is lost. So what to do?

↓

Interpolation: used to estimate unknown values b/w ~~data~~ known data points.

Types:

1. Nearest neighbour interpolation
2. Bilinear interpolation
3. Bicubic interpolation

etc.

$\frac{10}{3} \times \frac{2}{3}$

$= \frac{20}{9}$ +

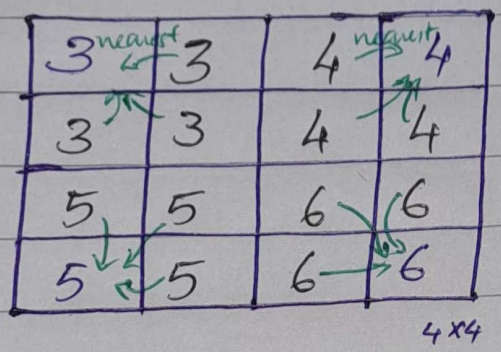
1. Nearest Neighbour Interpolation: (NNI).

3	4
5	6

  2×2

→ how to scale it by a factor of 2 using interpolation?

Step 1: Write down the corner points. We have 4 known values ↑ Now we find remaining using NNI.



Step 2: Now to a point consider the nearest point.

- Simplest method.

$\frac{11}{3} \times \frac{1}{3} +$

2. Bilinear Interpolation:

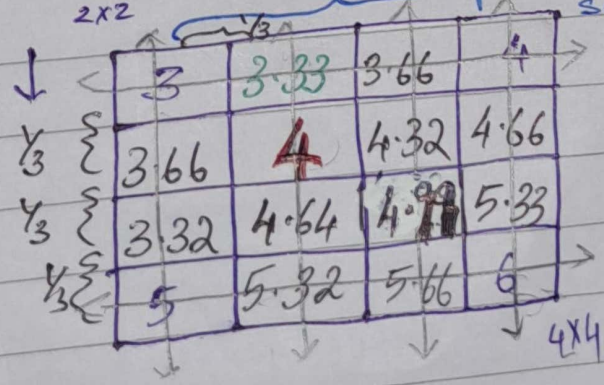
1 → distance.

3	4
5	6

  2×2

Task is the same. Again we know only the corner point.

→ assume dist remains same even after scaling.



Step 1: Find distance between known pixels in original matrix. Assume same dist in new matrix.

Step 2: Find dist b/w known & unknown pixels.

← Step 3: Multiply & Add. (Consider either vertically or horizontally)

$\frac{10}{3} \times \frac{2}{3} + \frac{16}{3} \times \frac{1}{3}$

$= \frac{20}{9} + \frac{16}{9} = \frac{36}{9} = 4$

$3 \times \left(1 - \frac{1}{3}\right) + 4 \times \left(1 - \frac{2}{3}\right)$

$= 3 \times \frac{2}{3} + 4 \times \frac{1}{3} = 2 + 1.33$

$= 3.33$

24/2/25 | Monday.

~~25/2 24/2/25 | Wednesday~~

3. Bicubic interpolation:

- Instead of 4 total points (as in bilinear), we'll be considering 16 neighbouring points.
- What if initially we only have 4 known points like in previous questions?
  - We pad the values to make it a 4x4 matrix.
  - We can use bilinear or any other method to make it 4x4 & then do bicubic interpolation.

$$\begin{bmatrix} P_1 & P_2 & P_3 & P_4 \\ P_5 & P_6 & P_7 & P_8 \\ P_9 & P_{10} & P_{11} & P_{12} \\ P_{13} & P_{14} & P_{15} & P_{16} \end{bmatrix}$$

$$P_{unknown} = P_1 q_1 + P_2 q_2 + P_3 q_3 + P_4 q_4$$

In bilinear $P_{unknown} = P_1 q_1 + P_2 q_2$

where q_1 & q_2 were inverse distance.

But in bicubic q_1, q_2, q_3, q_4 are polynomial functions based on ~~cont~~ Hermite interpolation.

Derivation:

$$P(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

$$P'(t) = a_1 + 2a_2 t + 3a_3 t^2$$

$$\rightarrow P(0) = f_0 = a_0 \quad -(1)$$

$$\rightarrow P(1) = f_1 = a_0 + a_1 + a_2 + a_3 \quad -(2)$$

$$\rightarrow P'(0) = f_0' = a_1 \quad -(3)$$

$$\rightarrow P'(1) = f_1' = a_1 + 2a_2 + 3a_3 \quad -(4)$$

$$f_1 = a_0 + a_1 + a_2 + a_3$$

$$= f_0 + f_0' + a_2 + a_3 \qquad f_1' - f_1 = -a_0 + a_2 + 2a_3$$

$$f_1' = a_1 + 2a_2 + 3a_3 \qquad a_2 + a_3 = f_1 - f_0 - f_0'$$

$$3f_1 = f_1' + 3a_0 + 2a_1 + a_2$$

$$3f_1 = f_1' + 3f_0 + 2f_0' + a_2$$

$$f_1 - f_0 - f_0' + f_1' - f_1 = -a_0 + 2a_2 + 3a_3$$

$$f_1 - f_0 - f_0' + f_1' - f_1 = -f_0 + f_1' - a_1$$

$$f_1 - f_0 - f_0' + f_1' - f_1 = -f_0 + f_1' - f_0'$$

$$0 = 0$$

(Z :)

$a_0 = f_0$

$a_1 = f_0'$

$f_1' - 2f_1 = a_1 + 2a_2 + 3a_3 - (2a_0 + 2a_1 + 2a_2 + 2a_3)$

$= -2a_0 - a_1 + a_3$

$f_1' - 2f_1 = -2f_0 - f_0' + a_3$

$a_3 = f_1' - 2f_1 + 2f_0 - f_0'$

Similarly find a_4.

Edge Detection:

Edges: abrupt changes in intensity, discontinuity in image brightness or contrast; usually edges occur on the boundary of two regions.

$a_0 = f_0$

$a_1 = f_0'$

$f_1' - 2f_1 = a_1 + 2a_2 + 3a_3 - (a_0 + 2a_1 + 2a_2 + 2a_3)$

$= -2a_0 - a_1 + a_3$

$f_1' - 2f_1 = -2f_0 - f_0' + a_3$

$a_3 = f_1' - 2f_1 + 2f_0 - f_0'$

Similarly find a_4.

Edge Detection:

Edges: abrupt changes in intensity, discontinuity in image brightness or contrast; usually edges occur on the boundary of two regions.

25/2/25 Tue

Masks for edge detection:

1. Prewitt operator
2. Sobel operator
3. Robinson Compass masks
4. Kirsch Compass masks
5. Laplacian operator

1. Prewitt Operator:

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

vertical edges — horizontal edges

- can detect only horizontal & vertical edges.

2. Sobel Mask:

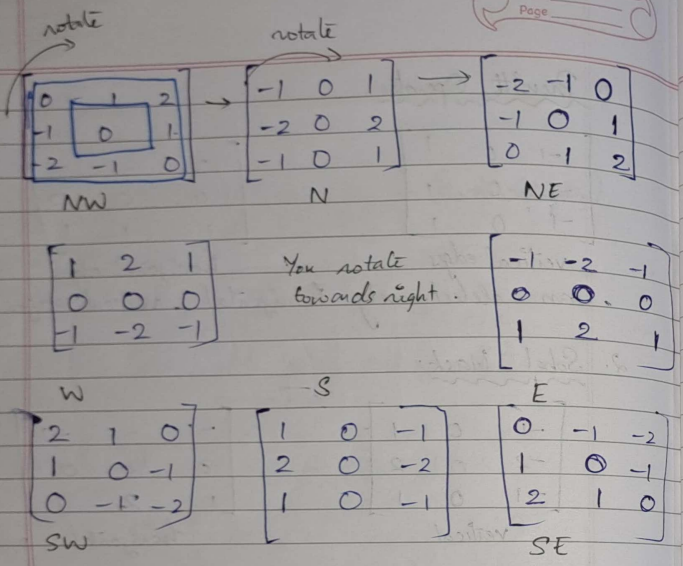
$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

vertical — horizontal

- Very similar to Prewitt.
- We can change -2 & 2. We could even give more weightage unlike Prewitt where it is a constant.

3. Robinson Compass Mask:

- 8 Masks each corresponding to: North, South, East, West, ~~No.~~ NE, NW, SE & SW.



4. Kirsch Compass:

Same as Robinson except more weightage.

5. Laplacian

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Negative Laplacian Positive Laplacian.

Canny's Edge Detection:

- Multi-step process that detects edges in images.

Img in Gray Level → Gaussian Smoothening → Take 1st derivative (application of Sobel filter over x & y separately) → Find Magnitude & Direction → Non-Maximal Suppression → Hysteresis Thresholding → Edge detected image

Gaussian filter (Gaussian Smoothening)

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

You apply this formula to get each value in the kernel given that you know x, y & σ.

coordinate is (0,0)

$$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

↳ sum of all elements in

As σ increases, degree of smoothening also increases

From Take 1st Derivative step you get G_x & G_y using sobel filter on horizontal & vertical done separately.

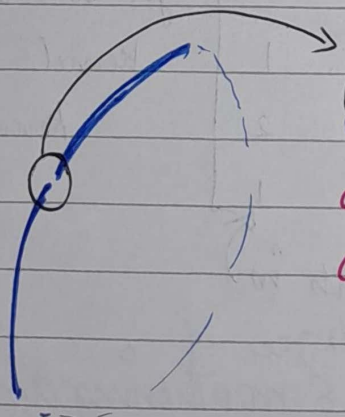
Now in nxt step

$$|G| = \sqrt{G_x^2 + G_y^2} \quad \rightarrow \text{magnitude}.$$

$$\theta = \tan^{-1}\left(\frac{G_y}{G_x}\right) \quad \rightarrow \text{direction}.$$

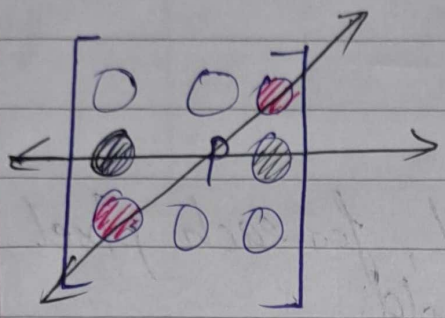
↓ to show how strong the edge is.

Using these data you suppress or do not suppress using non-maximal suppression. How?



for that point we select 2 nearest neighbours along the gradient then compare the magnitude.

3/3/25 Monday.



Say angle is 15° then you consider ⊘ these two points.
If it is 45° then you consider ⊘ these two points

4/3/25 Tuesday.

Segmentation:

- partitioning a digital image into multiple segments

Techniques:

1. Thresholding
2. Region growing
3. Edge-based segmentation
4. Clustering
5. Watershed
6. Graph based
7. Deep learning based.

1. Thresholding

1. Simple Thresholding

- ~~For all~~ Given a threshold, for each pixel compare the pixel value & threshold.
  - If pixel value ≥ threshold
    - → pixel value = 1
  - elif pixel value < threshold
    - → pixel value = 0
  - pixel value = threshold
    - → pixel value = 0 or 1 → You can decide.
- We have to manually assign threshold value.

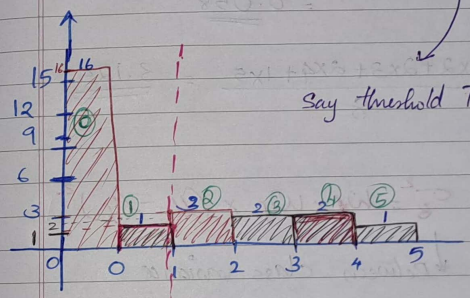
2. Otsu Thresholding:

- selects a threshold that minimizes the within class variance.
- We need to plot a histogram
  - ↳ plots frequencies.
- Plot frequencies of pixel intensities using histogram.

eg:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 2 & 3 & 0 \\ 0 & 2 & 3 & 4 & 0 \\ 0 & 1 & 4 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

histogram plot.



Say threshold T = 2.

T ≤ value → foreground class

T > value. → background class.

w_b = background class elements / total elements

$= \frac{16+1}{25} = \frac{17}{25} = \frac{68}{100} = 0.72$

$= \frac{68}{25 \times 4} = 0.68$

w_f = foreground class elements / total $= \frac{3+2+2+1}{25}$

$= \frac{8}{25} = 0.32$

$\mu_b = \frac{\sum \text{freq} \times \text{value}}{\text{total freq of background}}$

$= \frac{16\times0 + 1\times1}{17} = \frac{1}{17}$

$= 0.058$

$\mu_f = \frac{3\times2 + 2\times3 + 2\times4 + 1\times5}{8} = 3.125$

$\sigma_B^2 = w_b \times w_f \times (\mu_b - \mu_f)^2$

↳ Between class variance

$= 0.68 \times 0.32 \times (0.058 - 3.125)^2$

$= 2.04$

The σ_B^2 should be maximum.

Such a threshold must be chosen!

What if T = 3 ?

value	freq.
0	16
1	1
2	3
3	2
4	2
5	1

$w_b = \frac{16+1+3}{25} = \frac{20}{25} = \frac{80}{100} = 0.8$

$w_f = \frac{2+2+1}{25} = \frac{5}{25} = \frac{1}{5} = 0.2$

$\mu_b = \frac{16\times0 + 1\times1 + 3\times2}{20} = \frac{7}{20} = \frac{35}{100} = 0.35$

$\mu_f = \frac{3\times2 + 4\times2 + 5\times1}{5} = \frac{6+8+5}{5} = \frac{19}{5} = 3.8$

$\sigma_B^2 = 1.9044$

5/3/25 Wednesday.

2. Region growing:

ip: $\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 2 & 2 & 1 \\ 9 & 7 & 5 & 5 \\ 1 & 6 & 7 & 1 \end{bmatrix}$

For this also we need a threshold value, Let it be T.
Let T = 3.

let this be the initial seed point.

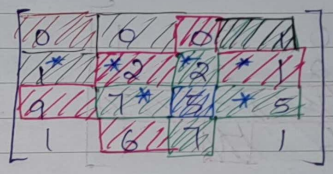
Step1: For the given img choose an initial seed point.

initial seed pixel intensity = 5

2. Look into neighbours. It can be 4-way (4 neighbours) or 8-way (8 neighbours).
We choose 4-way.
neighbours → 2, 7, 7, 5

3. If absolute diff. blw the neighbouring points & the initial seed point is ≤ threshold, then segment it into same segment (where seed point is present).
eg: 2 → |5-2| = 3 ≤ 3 → 2 is added to 5.
7 → |5-7| = 2 ≤ 3 → 7 is added to 2,5
5 → |5-5| = 0 ≤ 3 → 5 is added to 2,5,7,7

Thus it becomes:



T=3.

Now go to nxt point, initial seed = 7.
|7-2| = 5 ≥ 3 No
|7-9| = 2 ≤ 3 Yes
|7-6| = 1 ≤ 3 Yes.

initial seed = 5
|5-1| = 4 ≥ 3 No
|5-1| = 4 ≥ 3 No

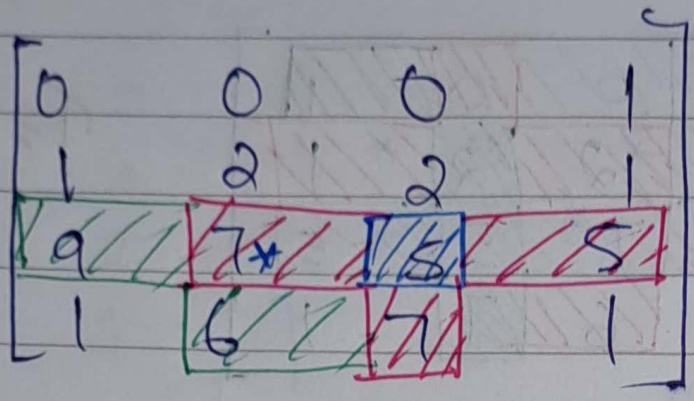
initial seed = 2
|2-0| = 2 ≤ 3 Yes
|2-1| = 1 ≤ 3 Yes
|2-2| = 0 ≤ 3 Yes

initial seed = 1
|1-1| = 0 ≤ 3 Yes

initial seed = 2
|2-0| = 2 ≤ 3 Yes
|2-1| = 1 ≤ 3 Yes

initial seed = 1
|1-0| = 1 ≤ 3 Yes

T=2.



S=5

$|5-5| = 0 \leq 2$ (Y)

$|5-7| = 2 \leq 2$ (Y)

$|5-2| = 3 > 2$ (N)

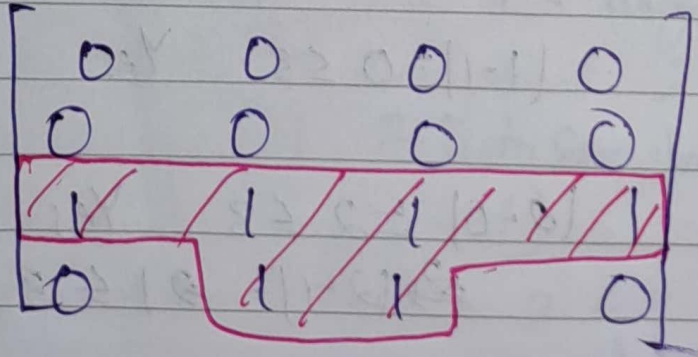
S=7

$|7-9| = 2 \leq 2$ (Y)

$|7-6| = 1 \leq 2$ (Y)

$|7-2| = 5 > 2$ (N)

Now make everything in segment to 1 & others to zero,



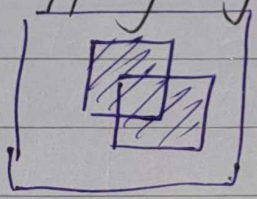
T=2

10/3/25 | Monday.

3. Watershed

Disadv. of region growing:

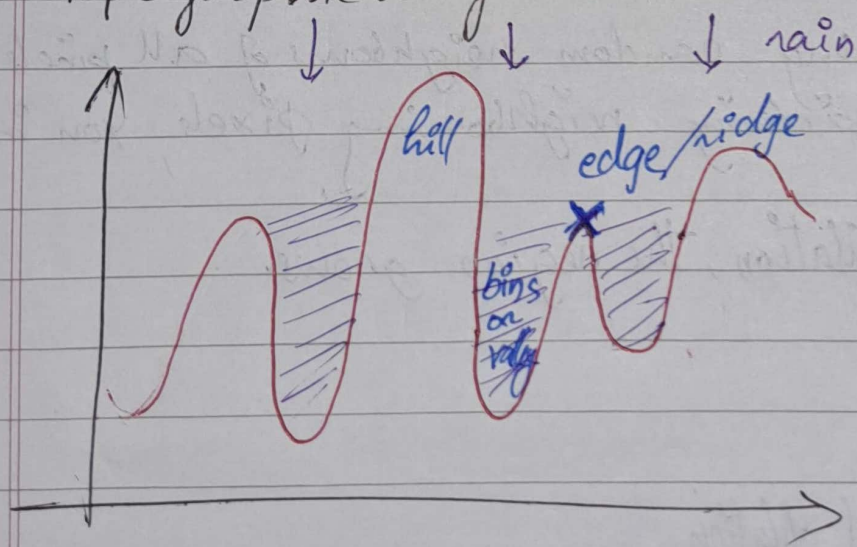
1. Computational complex:
   pixels that we have already marked as foreground might be checked again.
2. Overlapping objects.



- the overlapping object may be considered as a single object in this.

3. Watershed

- Topographical features



Steps:

1. RGB → binary image
   - we can use thresholding technique
2. Noise reduction.
   - Use Gaussian filter, we can do noise reduction.
3. Morphological operations
   - → Dilation
   - → Erosion

Dilation:

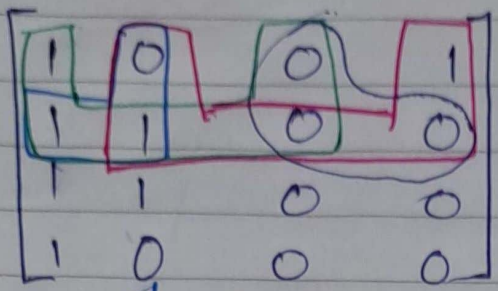
- structuring element
  - ↳ something similar to sobel filter, Laplacian etc.
- You do convolution
- Considering random neighbours of all pixels.
- By considering neighboring pixels, you increase width.
- After dilation, the region grows.

Erosion:

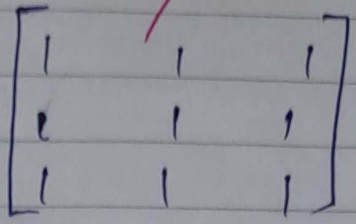
- Opposite of dilation
- shrinkage of region.

11/3/25/ Tuesday.

eg:



i/p img.

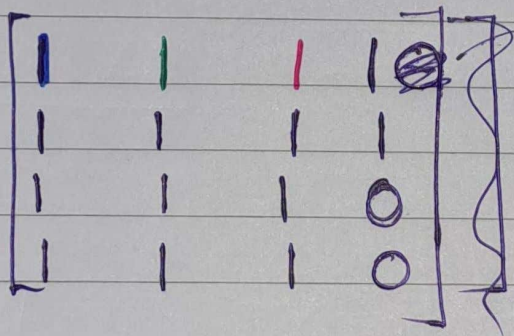


structuring element.

signifies all neighbours are to be considered.

Perform dilation.

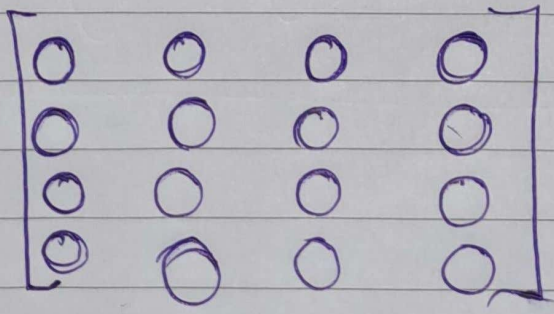
Ans: The structuring element is applied on all pixels. If any of the neighbours is one then take 1. Corner elements has only 3 neighbours.



→ the neighbours of that 0 was all zeroes but originally it was one.

We only change zeroes in dilation

Q. Do erosion:



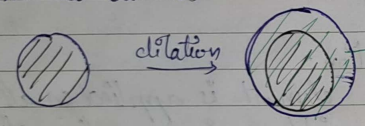
If even one neighbour is zero make the one to zero.

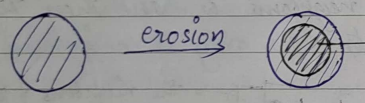
13/03/25 Thursday.

- By doing dilation we get sure background pixels.
- By doing erosion we get sure foreground pixels.

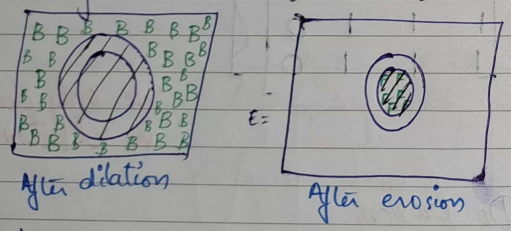
4. ~~Markers~~ Markers:

- We do dilation − erosion. How? ?

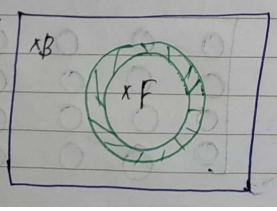




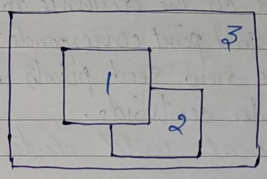
- Now you mark them - markers.



- Now D-E ↓



- What if two objects? Markers will be of 3 colors.



- Now you have marked the regions.

5. Flooding:

- We are doing flooding, so that each segment of the image has a different colour.
- Flooding starts from the markers.
- ~~How to fin~~
- How to find boundary? ?
  The region where the two colours intersect.

4. K-Means Clustering:

- K - no: of segments / clusters.
  k=3 ⇒ 2 objects + 1 background.
- Step1: Take random k seedpoints where k is number of clusters.
  Step2: Compute Euclidean dist b/w each point & a seed point.

Step 3: Assign each data point to a segment based on the distance to the seed point. (each seed point corresponds to one segment).

Step 4: Take new seed points ~~base~~ by finding cluster centroids.

Step 5: Repeat until cluster no longer change.

↳ How to do it in an image?

↳ Based on intensity of image pixels.

You do not consider coordinates instead just take absolute value of difference of the two pixel intensities as distance.

Step 3: Assign each data point to a segment based on the distance to the seed point. (Each seed point corresponds to one segment).

Step 4: Take new seed points ~~base~~ by finding cluster centroids.

Step 5: Repeat until cluster no longer changes.

↳ How to do it in an image?

Based on intensity of image pixels.

You do not consider coordinates instead just take absolute value of difference of the two pixel intensities as distance.

17/3/25/Mon UNIT-2

Feature Detection:

Currently we have segregated the foreground objects. Now we need to identify the features of the foreground objects.

1. Harris Corner Detection Algorithm:

- to detect corners in the given images.
- How?

$$\begin{bmatrix} 0 & 0 & 0 & 4 & 5 \\ 0 & 1 & 1 & 4 & 5 \\ 1 & 2 & 3 & 4 & 4 \\ 1 & 2 & 2 & 3 & 4 \\ 0 & 2 & 3 & 3 & 3 \end{bmatrix}$$

Step 1: Find the gradients using sobel or Prewitt or etc.

Eg: Gradient computation:

Prewitt $\rightarrow P_x \rightarrow \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$

Prewitt $\rightarrow P_y \rightarrow \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$

Assuming that we are doing 1D convolution.

$$\begin{bmatrix} 1 & 1 & 4 \\ 2 & 3 & 4 \\ 2 & 2 & 3 \end{bmatrix} \rightarrow \text{Using } P_x \quad \begin{bmatrix} -1 & 0 & 4 \\ -2 & 0 & 4 \\ -2 & 0 & 3 \end{bmatrix}$$

$$\rightarrow \text{Using } P_y \quad \begin{bmatrix} -1 & -1 & -4 \\ 0 & 0 & 0 \\ 2 & 2 & 3 \end{bmatrix}$$

Now make it 1D:

$$P_x = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix} \qquad P_y = \begin{bmatrix} -6 \\ 0 \\ 7 \end{bmatrix}$$

Gradient Computation:

(Just consider the part outside [ ] as padding) (Don't do this in exam. In exam if 5×5 matrix is given, then give enough padding to get 5×5 matrix as $I_x + I_y$).

$P_x \rightarrow [-1 \quad 0 \quad 1]$

$$\begin{bmatrix} 0 & 0 & 0 & 4 & 5 \\ 0 & 1 & 1 & 4 & 5 \\ 1 & 2 & 3 & 4 & 4 \\ 1 & 2 & 2 & 3 & 4 \\ 0 & 2 & 3 & 3 & 3 \end{bmatrix}$$

$[0 \quad 1 \quad 1] \circledast [-1 \quad 0 \quad 1]$

$0\times-1+1\times0+1\times1$

$$\begin{bmatrix} 1 & 3 & 4 \\ 2 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \leftarrow I_x$$

$0\times-1+1\times0+2\times1$

$P_y \rightarrow \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$ $I_y \rightarrow \begin{bmatrix} 2 & 3 & 0 \\ 1 & 1 & -1 \\ 0 & 0 & -1 \end{bmatrix}$

Step 2: Harris corner matrix, H

calculate: $H_1 = I_{x_1}^2 + I_{x_2}^2 + I_{x_3}^2 + \dots + I_{x_9}^2$

$H_1 = 1^2+3^2+4^2+2^2+2^2+1^2+1^2+1^2+2^2 = 1+9+16+4+4+1+1+4 = 10+20+11 = 41$

$$H = \begin{bmatrix} H_1 & H_2 \\ H_3 & H_4 \end{bmatrix}$$

calculate $H_4 = I_{y_1}^2 + I_{y_2}^2 + \dots + I_{y_9}^2$

$H_2 = H_3 = I_{x_1}I_{y_1} + I_{x_2}I_{y_2} + \dots + I_{x_9}I_{y_9}$

$H_2 = 2^2+3^2+0^2+1^2+1^2+-1^2+0^2+0^2+-1^2$

$= 4+9+1+1+1+0+1 = 17$

$H_3 = H_4 = 2+9+0+2+2+-1+0+0+-2$

$= 12$

$$H = \begin{bmatrix} 41 & 12 \\ 12 & 17 \end{bmatrix}$$

6/5/25/Tuesday.

How to distinguish blw corners & edges?

↳ When two edges intersect, it becomes a corner. Variation / intensity change occurs in multiple directions in the case of a corner.

Step 3: After finding Harris matrix, H find corner score R

$$C = \det(H) - k(\text{trace}(H))^2$$

↳ sum of diagonal elements (not sum)

C → corner score. k → sensitivity

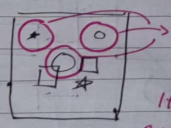
Step 4: Assign thresholds according to C.

★ BLOBs → Binary Large Objects.
→ A group of pixels which is sharing a set of features; can be considered as a collective component.
→ eg: each cell is considered as a BLOB
↳ human body cell.

★ LoG → Laplacian of Gaussian.
→ We can detect BLOBs using LoG.

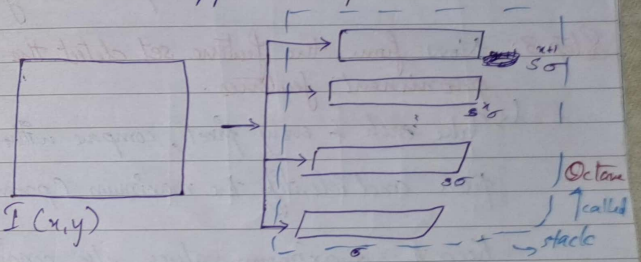
2. SIFT Algorithm → for det

- for detecting BLOBs.
- Scale Invariant Feature Transform
- In LoG operator, all BLOBs are of same scale.
↳ disadv. of LoG.



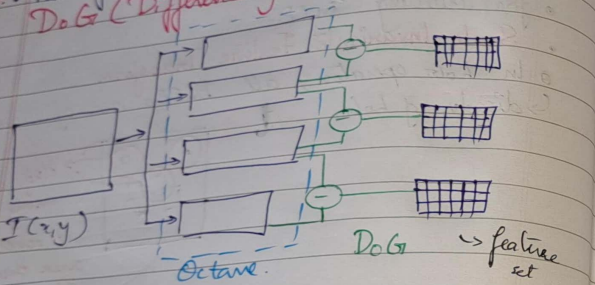
BLOB detected using LoG. It covers more area than needed for object.

- In SIFT this disadvantage is removed.
- Here Gaussian is applied multiple times inlike LoG.



Step 1: Gaussian is applied with different values of σ. The top image at the top of the stack is the most smoothened (max value of σ). This stack is called Octave.

Step 2: • Now after you have Octave, compute the DoG (Difference of Gaussians).

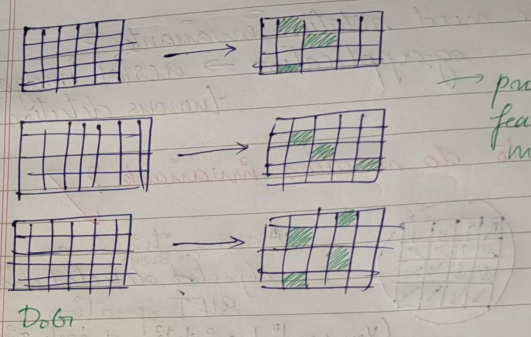


* We could have computed LoG instead of DoG but LoG is more computationally complex.

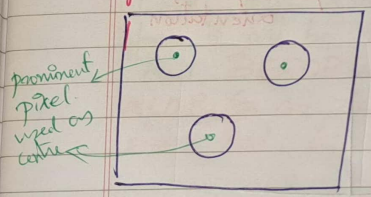
Step 3: Now from the feature set detect the most prominent features.

↳ Take each & every pixel, compare with the neighbouring pixels and identify the maximum (prominent value).

Keep the maximum values else consider those points as weak pixels.



Step 4: ✱ Now use these marked prominent features are centres to the BLOBs.

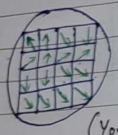


But here the BLOBs are of same size. How to make it diff size?

Make different octaves. Change the scale for each octave and then you'll get the diff: size BLOBs.

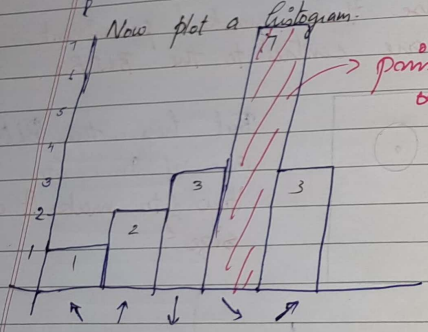
Do we need rotation invariante ?
↳ Yes eg: application → aerial images, tumour detection etc.

How to do rotation invariante ?



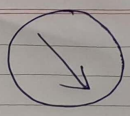
Assume this is an (or BLOB) object detected using SIFT operation
(You can find orientation of each pixel by calculating gradient)

Now plot a histogram.

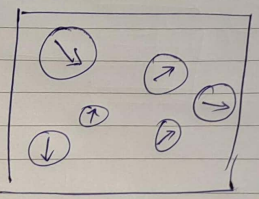


Now the highest frequency is taken as principal orientation.

Now assign the orientation of the whole BLOB as the principal orientation.

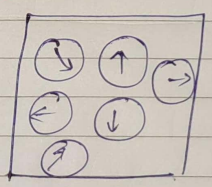


Now find orientation of all the BLOBs (like this.



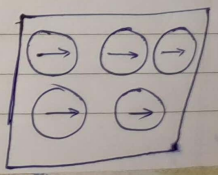
(Randomly assigned).

Now we can't have these BLOBs to have diff. orientation. We need it to align for it to be ∉ rotation invariante.
But BEFORE ALIGNING normalize the BLOBs as in make them of same size. (That is; normalize the Octave part)



(All BLOBs of same size)

Now rotate the BLOBs and align the orientation



13/5/25 Tuesday.

Why create octaves?
We are taking different levels of smoothening to create octaves. To detect the object at different scales.

n → no: of prominent feature points
m → each point represented with m dimensions

SIFT will be giving us a ~~multi dimension vector~~ n×m matrix which we use to describe each point in the image. Now after this we do feature mapping → each point represented using m dimensions.

To do feature mapping b/w ~~two~~ points?
Just do Euclidean distance of the two vectors.
So if we want to find match of a point P_{1A} in img A in img B, ~~first~~ just find Euclidean distance b/w point ~~P~~ P_{1A} of all points in img B.
↳ Brute force method. → not efficient. So,

We could do KNN instead.

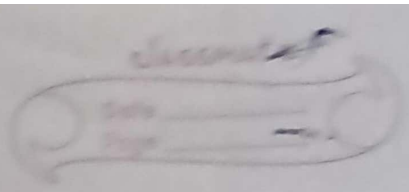
Img1 → SIFT → [n × 128] (n: points selected.., 128: no: of features)

Img2 → SIFT → [m × 128]

n can be greater than or less than m.

- Do KNN for each of the vectors (rows) in the matrices.

How do you know which vectors are neighbours? (within a matrix)
↳ By taking Euclidean distance. Then you compare ~~points~~ vectors with the two matrices. Then compare the neighbours as well. If they are similar then you can select that point as the point matched (feature mapping).

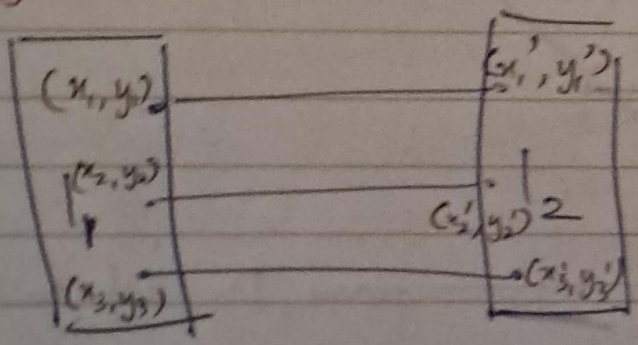


- Do KNN for each of the vectors (rows) in the matrices.

How do you know which vectors are neighbours?
↳ By taking Euclidean distance. Then you compare ~~each~~ vectors with the two matrices. Then compare the neighbours as well. If they are similar then you can select that point as the point matched (feature mapping).

28|5|25|Wednesday

Image Alignment:



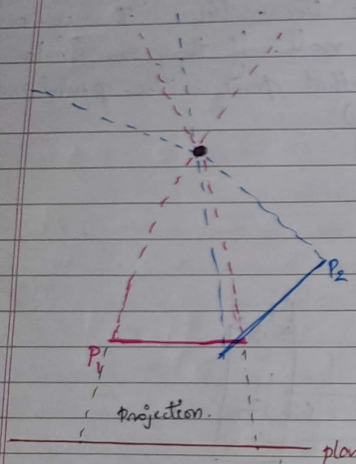
paired feature points which were matched using KNN.

Now you need to stitch it together. How?
↳ By doing transformation.

T → transformation matrix.

$x \rightarrow x'$

H → Homogenous transformation matrix
Projective transformation.
↳ 8 DoF.



Project all these viewpoints onto another plane.

$$\begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} = \begin{bmatrix} \tilde{x}_d \\ \tilde{y}_d \\ \tilde{z}_d \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix}$$

projected.

H 3×3

2nd img points → first img points

$(x_s, y_s) \longrightarrow (x_d, y_d)$

$$\tilde{x}_d = (h_{11} \times x_s) + (h_{12} \times y_s) + (h_{13} \times 1)$$
$$\tilde{y}_d = (h_{21} \times x_s) + (h_{22} \times y_s) + (h_{23} \times 1)$$
$$\tilde{z}_d = (h_{31} \times x_s) + (h_{32} \times y_s) + (h_{33} \times 1)$$

$$x_d = \frac{\tilde{x}_d}{\tilde{z}_d} = \frac{(h_{11} \times x_s) + (h_{12} \times y_s) + (h_{13} \times 1)}{(h_{31} \times x_s) + (h_{32} \times y_s) + (h_{33} \times 1)}$$

$$y_d = \frac{\tilde{y}_d}{\tilde{z}_d} = \frac{(h_{21} \times x_s) + (h_{22} \times y_s) + (h_{23} \times 1)}{(h_{31} \times x_s) + (h_{32} \times y_s) + (h_{33} \times 1)}$$

After rearranging ↓ $A = \begin{pmatrix} x_d \\ y_d \end{pmatrix}$ becomes

$$\begin{bmatrix} x_s & y_s & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_s & y_s & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} x_s & y_s & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_s & y_s & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = 0$$

A H

$$AH = 0$$

Now solve this & you will get H. (Use Eigen ~~value~~ vectors)

$$|A - \lambda I| = 0$$

& we will have a set of Eigen Vectors with us.

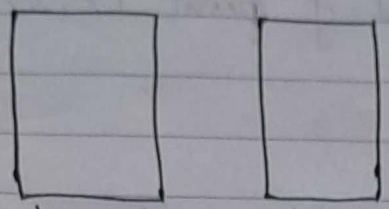
Now how to find H?
Use SVD. $A = U \Sigma V^T$

V^T → E.Vectors of $A^T A$ in decreasing order.

In order to find H, find a minimum 4 Pairs of feature match on 8 feature points (8 unknowns in H so 8 eqns required).

2/6/25/Monday.

Optical Flow:

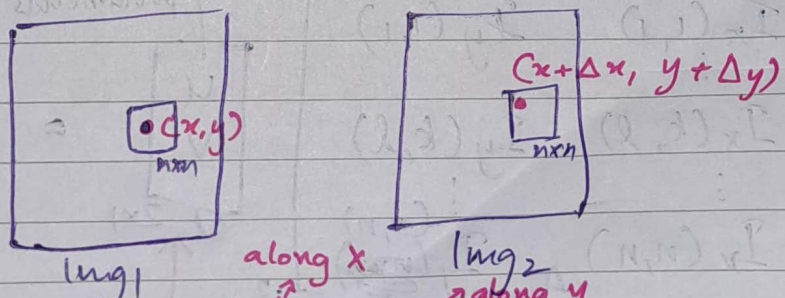


Video → Img_1 , Img_2
↓ ↓
Frame 1 Frame 2.
↓ ↓
t_1 t_2

Intensity change over time can be measured.
Must calculate change in intensity over time between frames. → Optical Flow

↓ why calculate?
↳ This will help us estimate motion.

~~Optical constraint~~
Optical flow constraint → to calculate this, we'll be considering patches in images. The patch must be in the same location in the images.
Then we check how a particular point from the patch changed



Displacement : $(\Delta x, \Delta y)$ (along x, along y)

Optical flow = $\left(\frac{\Delta x}{\Delta t}, \frac{\Delta y}{\Delta t}\right)$ → that is, velocity.

Let (k,l) be a point in the image.
You can find intensity I of point (k,l) from the image.
Now, find

$I_x(k,l)$ → change of intensity along x
$I_y(k,l)$ → change of intensity along y
$I_t(k,l)$ → change of intensity over time.

Now there are $n \times n$ pixels / points in the patch.
↳ n^2.

thus →

$$\begin{bmatrix} I_x(1,1) & I_y(1,1) \\ \vdots & \vdots \\ I_x(k,l) & I_y(k,l) \\ \vdots & \vdots \\ I_x(n,n) & I_y(n \times n) \end{bmatrix}_{n^2 \times 2}$$

Optical flow parameters

$$\begin{bmatrix} I_x(1,1) & I_y(1,1) \\ \vdots & \vdots \\ I_x(k,l) & I_y(k,l) \\ \vdots & \vdots \\ I_x(n,n) & I_y(n,n) \end{bmatrix}_{n^2 \times 2} \begin{bmatrix} u \\ v \end{bmatrix}_{2 \times 1} = \begin{bmatrix} I_t(1,1) \\ \vdots \\ I_t(k,l) \\ \vdots \\ I_t(n \times n) \end{bmatrix}$$

A — known — unknown — V — known

$$A \cdot U = B$$

left multiply A^T on both sides

$$A^T A \cdot U = A^T B$$

$$U = (A^T A)^{-1} A^T B$$

↳ This can be solved using least sq.s.

Finally you get $A^T A = \lambda_1, \lambda_2 \ldots$

Based on these λ values you get to know whether the patch you selected is good or not.

Good patch → There should be a variation but it should not be too much.

If the difference b/w two λ values are too high then that means variation is high; bad patch.

Also $A^T A$ should be invertible for this whole thing to work!

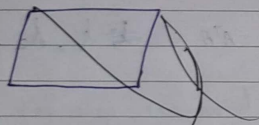
3/6/25 Tuesday.

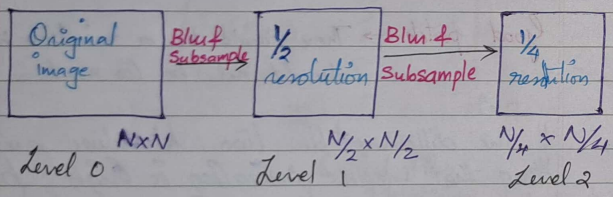
~~Jallat~~

Q. What if optical flow b/w ~~two~~ frames is too high?
↳ Use Resolution pyramids.

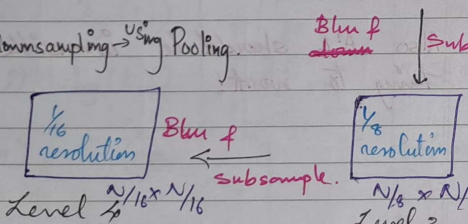
Resolution pyramids:

- also known as image pyramids.



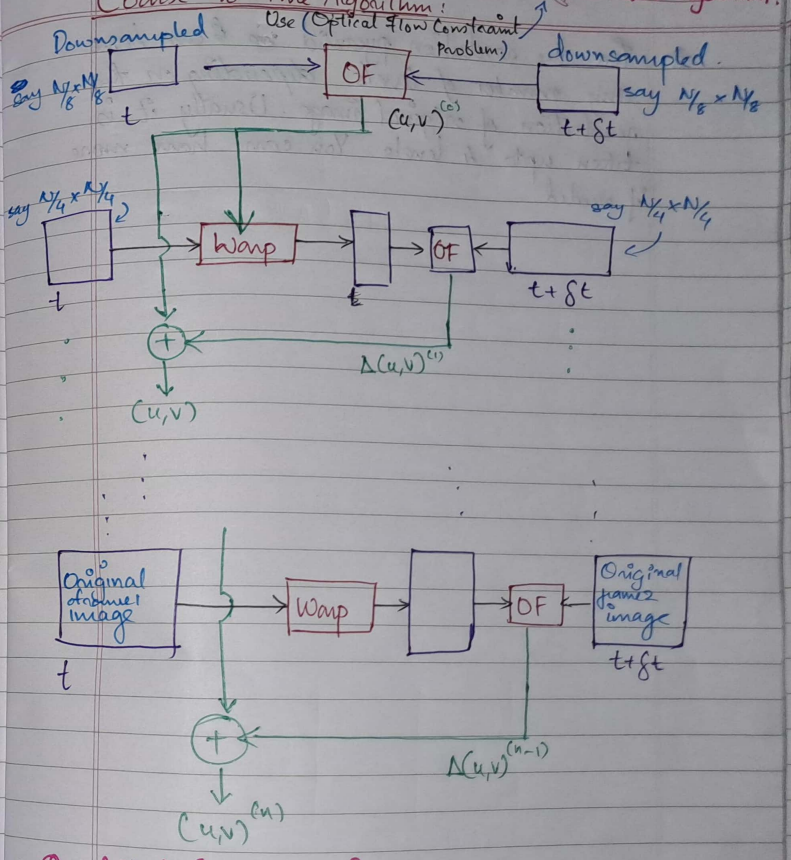
- Downsampling multiple times to get the pyramid.
- 

* Blur & Subsample → downsampling → using Pooling.



Coarse-to-Fine Algorithm:
Use (Optical flow Constraint Problem)

other name for ~~Lucas~~ Lucas-~~Kanade's~~ Algorithm.



Q. What is warp?
↳ Upsampling done using interpolation.

* For the resolution pyramid we can have any number of levels depending on the resolution of original image. Usually it is taken upto 4 levels. You can have more if needed.

* For the resolution pyramid we can have any number of levels depending on the resolution of original image. Usually it is taken upto 4 levels. You can have more if needed.

9/06/25 Monday.

Tasks in CV:

1. Object Detection:

bounding box → (x, y, w, h)
(x, y) → coordinates of point in bounding box
w → width, h → height.

Object detection locates and classifies multiple objects of interest within an image.
Where + What → bounding boxes + class labels

popular models:
- YOLO (You Only Look Once)
- Faster RCNN → Region based CNN
- SSD (Single Shot Detector)
- DETR (Detection Transformer)

2. Face Detection:

Locating human faces in images using bounding boxes.
Where → bounding boxes

Popular models:
- Haar cascades
- MTCNN
- RetinaFace

3. Face Recognition

Where + Who → bounding boxes + class labels

Popular models:
- FaceNet
- DeepFace
- ArcFace

4. Instance (Level) Recognition

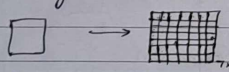
Identifying specific object instance rather than just a category.

eg: Base level recognition : monument / building
Instance level : Taj Mahal

5. Category Recognition
- AlexNet
- VGG

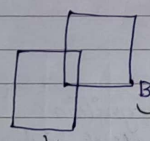
YOLO:

- single neural nlw to predict bounding boxes & associate class probabilities.
- image is divided into grids & each grid produces bounding boxes & their confidence scores.
- bounding box → (x, y, w, h, confidence)
- In YOLO v1 → image is divided into 7×7 grids. ↳ S=7



→ for each grid generate 2 bounding boxes
→ ~~acto~~ this was retrained on PASCAL VOC dataset. 20 classes.

- confidence score → ~~IO~~ IoU → Intersection Over Union

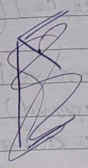


$IoU = \frac{A \cap B}{A \cup B}$

→ maximum IoU → better localization.

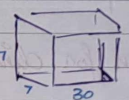
- Non maximum suppression → ~~suppose~~ only considering bounding boxes with more confidence scores.

- 24 conv layers & 2 fully connected layers.



1st Conv layer: 3 channels.
img size → 448 × 448 × 3
filter → 7×7 × 64 -s-2
of size. 64 filters stride = 2

Maxpool layer → 2×2 -s-2.

Final layer →  → 7×7×30

Why?

11/06/25 Wednesday.

20 classes are present. Usually it would give 20 nodes but here there are 30. Why?
Each grid in the input image is being mapped to a vector of 30 dimensions (img divided into 7×7 grids)
Original size of img: 448×448.

$\frac{448}{7} = 64 \times 64$ → size of 1 grid.

Each such grid mapped to a 30 dimensional vector.
Why 30?
↳ Each grid has 2 bounding boxes assigned to it

Bounding box ↓

BB1 ↓ $[x_1, y_1, w_1, h_1, c_1]$ $[x_2, y_2, w_2, h_2, c_2]$ BB2 ↓ (Appending both)

1×5 1×5

ie; → $[x_1, y_1, \ldots]$ 1×10 dimension vector.

↑ (append)

Now there are 20 classes → 1×20 dimension

Thus the output comes as a 30 dimensional vector.

Thus general eqn for dimension of last is ↓

$$S \times S \times (B \times 5 + N)$$

no: of grids ← no: of bounding boxes → no: of classes.

Viola Jones for Face Detection:

- Relatively slow to train, but can quickly detect objects accurately.
- Works well to detect human faces in real time.

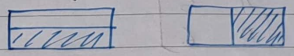
Steps:

1. Selecting Haar features

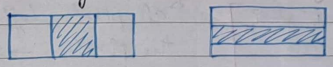
Steps:

1. Selecting Haar features

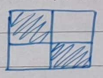
1. Edge features



2. Line features



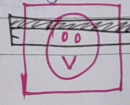
3. Four rectangle features



These filters can be used to detect various features of the face.

2. Creating an integral image.

- This step is done to reduce computation.
- It gives a fast & simple way to calculate the value of any haar-like feature.
- The value for location (x, y) on the integral image is the sum of the pixels above & to the left of the (x, y) on the original image plus itself.

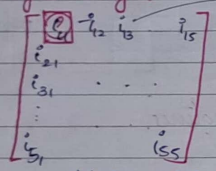


how to generate integral image?

↳ Input image:

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

A: ↳ integral image.



all pixels left & above & the pixel. → Sum 7. 5. Thus = 5.

left of 2, original pixel

$i_{12} = 5+2 = 7$

i_{33} = left pixels = 2+2 = 4
top pixels = 3+4 = 7
ori pixel = 1
12

ori. pi

$i_{33} = 2+2+4+3+5+2+1+5+1 = 25$

Thus final integral image becomes:

5	7	10	14	15
6	13	20	26	36
8	17	25	34	42
11	25	39	52	65
15	30	47	62	81

How is it benefiting?

↳ If normal addition of a patch of size N×N then no: of additions = $N^2 - 1$

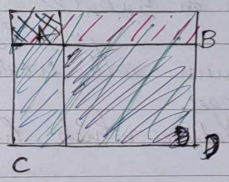
Say we need to find sum of value in patch O_{22} $\begin{bmatrix} 5 & 4 \\ 2 & 1 \end{bmatrix}$ ← original image

How to find using integral?

25 − 10 − 8 + 5 = 12 (25 is the sum of whole area, 8 & 10 are sum of vertical & horizontal areas 5 is subtracted twice so we add it)

Check above image for understanding.

↳ General.

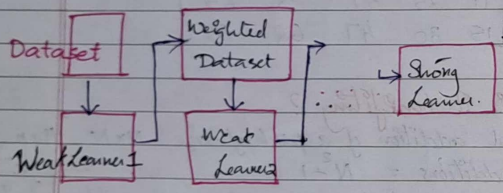


▨ = ▨ − ▨ − ▨ + ▨

↳ Patch$_{needed}$ = D − B − C + A

3. Adaboost training.

- Selecting a subset of prominent features from the huge set of features.
- Multiple weak learners used to create a strong learner.



- The algorithm sets a minimum threshold for determining a useful feature.

(How does this do feature selection?

→ Weak Learner 1 → w_1 → focuses of haar feature 1 → $h_1(x)$

w_2 → focuses on $h_2(x)$

w_n → focuses on $h_n(x)$

In final strong learner each of these haar features would have different weightages.

Take the haar features which have more weightage.

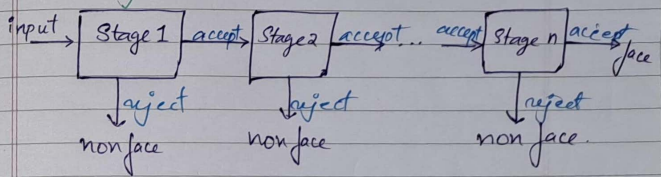
weightage

$$\alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x) + \ldots + \alpha_n h_n(x)$$

↳ Strong learner.

Now we have one strong classifier. Similarly we can generate multiple strong classifiers and then get make a decision. → Cascading

4. Cascading.



Each of these stages are strong classifiers. At any stage, if the classifier rejects it then with no 100% doubt we can say the image has no face. If it accepts go check for face in next classifiers, like that until n classifiers.

* For a single object there can be multiple bounding boxes in this case as well because the haar like features can be in any aspect ratio.

So what to do?

↳ Use Non-Maximum Suppression.