Date 10/10/2020

CS-512 Assignment 1 — fall 2020

Image formation and filtering

Review Name - Kinjal Kachi

Q1) Geometric Image Foundation.

a) Let $(f) = 10$ be the focal length of a camera. Let $p = (3, 2, 1)$ be a world point. find the co-ordinate of the point p when projecting it on the image. Assume that the projections is done in camera co-ordinates so there is no need for a transformation between co-ordinate systems.

→ $f = 10$; $P_w = (3, 2, 1)$. $\quad x_w, y_w, z_w$

Representation of projections using linear equations in homogeneous co-ordinates :-

$$u = \frac{-fx}{z} = \frac{(-10) \times 3}{1}$$

$$v = \frac{-fy}{z} = \frac{(-10) \times 2}{1}$$

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} -f & & \\ & -f & \\ & & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

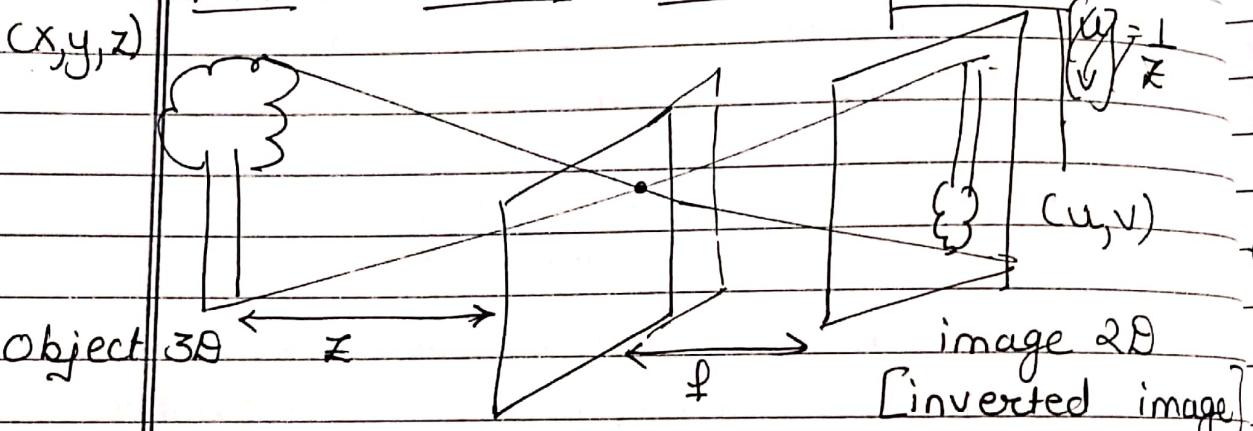
$\therefore u = \frac{u}{w} = \frac{-fx}{z}$

$v = \frac{v}{w} = \frac{-fy}{z}$

$\therefore$ Image projections. $= (-30, -20)$

Q2) Explain difference between pinhole camera where image plane is behind the centre of projection & the pinehole camera model where the image plane is in front of the centre of projection. Which model corresponds better to a physical pinhole camera model? How is the other model justified?

⇒ Pinhole Camera Model:



f = focal length.

focal length controls the size of the image, bigger the distance, bigger the image

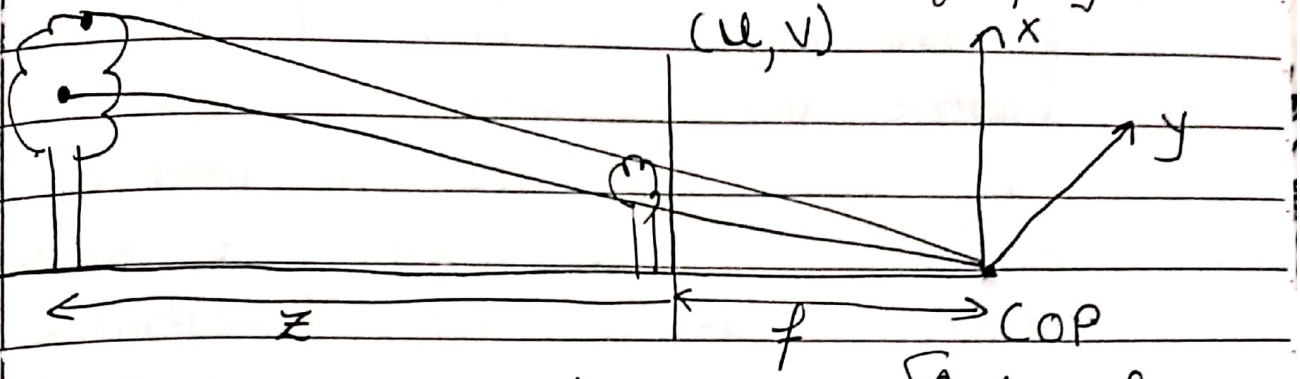
Balanced Matrix Representation:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1/f \end{bmatrix} \begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} \qquad u = \frac{fX}{Z}, \quad v = \frac{fy}{Z}$$

matrix

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{Z} \begin{bmatrix} -f & 0 \\ 0 & -f \end{bmatrix} \begin{bmatrix} X \\ Z \end{bmatrix}$$ Equation I

⇒ Alternative Model where image plane is in front of the centre of projection:-



Object 3D image 2D [non-inverted image] [Center of Projection]

Equation for alternative model:-

Equation (2)

$$\frac{x}{z} = \frac{u}{f} \Rightarrow u = f\frac{x}{z}$$

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

$$\Rightarrow \begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{z}\begin{bmatrix} f & 0 \\ 0 & f \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}$$

- In Alternative model to the Camera co-ordinate System is centered at COP [Center of Projection].

→ In this model there is no hole, the image is formed by connecting the points on the object to the centre of projection. When the rays of the object points hits the image plane the image is formed.

→ Unlike the previous "Regular Pinhole Camera where image plane is behind the COP" the images are non-inverted in this case.

Hence, the alternative model, having image plane in front of C.O.P is better and justified

As you can see in previous page Equation 1 represents our regular Pinhole Camera and Equation 2 represents alternative model. These equations are geometrically equivalent and the equation 2 for the alternative model also shows that there is no inversion.

Q1 c)a) ~~Explain.~~ w' Explain what happens to the projection of an object when the focal length get bigger?

→ Focal length (f) controls the size of the image, hence bigger focal length gives bigger image projection of an object.

1c)b what happens when distance to object gets bigger.

→ when distance to object (z) gets bigger then we get a smaller projection of an object.

Q1 d) 2D point (1, 1),

∴ its co-ordinates in homogeneous coordinates (2DH)

⇒ 2D → 2DH

(1, 1) ⟶ (1, 1, 1).

(x, y) ⟶ (tx, ty, t)

Another set of homogeneous co-ordinates (2DH) we that corresponds to the same 2D points could be :-

(2, 2, 2) or (3, 3, 3) ... where t is 2 or 3 respectively.

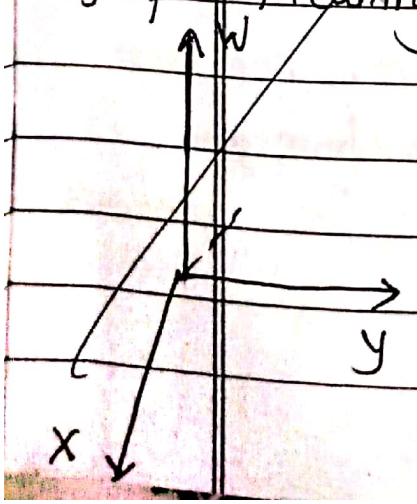
Q1 e) Given :- 2DH (1, 1, 2) find 2D point corresponding to it

$(x, y, w) \longrightarrow \left(\frac{x}{w}, \frac{y}{w}\right)$.

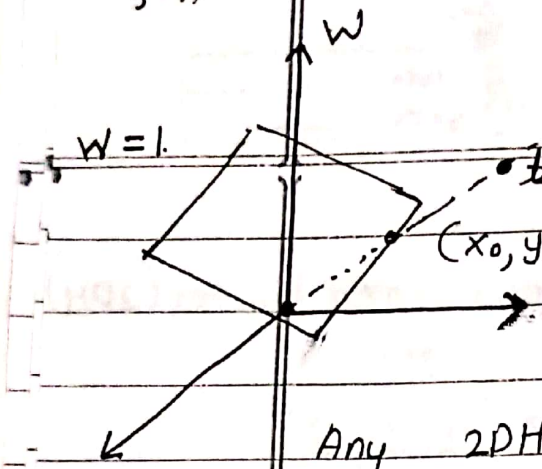
$\left(\frac{1}{2}, \frac{1}{2}\right)$

(0.5, 0.5). 2D points

Q1) f Meaning of 2DH point (1, 1, 0).



Q1 f) Explain meaning of the 2DH point (1, 1, 0)



Point in 2D = line in 2DH.

homogenity = intersecting with plane $w = 1$.

Any 2DH point which has $w=0$ is called point of infinity, because when we try to homogeneous it $\left(\frac{1}{0}, \frac{1}{0}\right)$ it gives an infinite value.

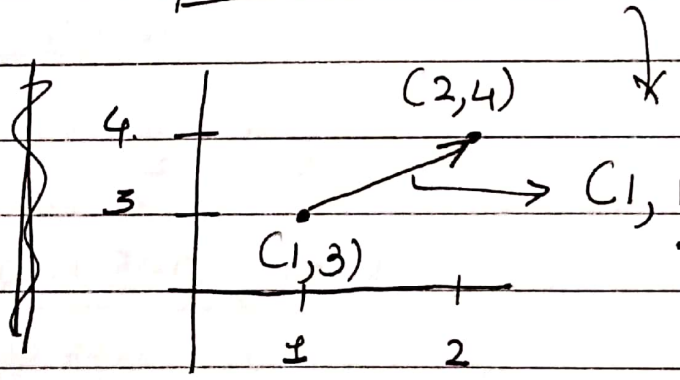
→ It does not represent real point in 2D, it represents direction.

Example:-

Take 2 pts (1, 3, 1) and (2, 4, 1)

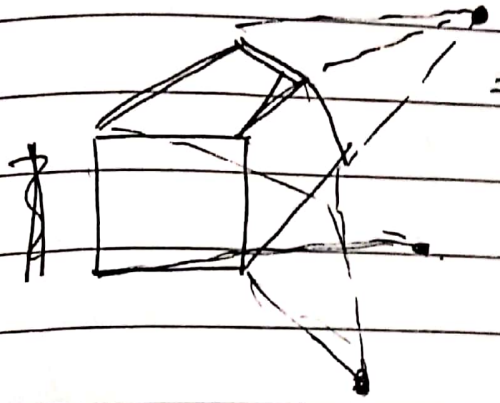
∴ (2, 4, 1) - (1, 3, 1)

gives (1, 1, 0) in homogenous coordinates and represent direction. in 2D coordinates



Q1 g. Why it is possible to write the non-linear projection equa" as a linear equa" in homogenous co-ordinates.

Points in infinity are Vanishing points. Vanishing points are projection of direction



Here as w is 0. it shows a vanishing point. A vanishing point is the projection of a point in infinity.
As you can see in the diagram the parallel lines are assumed to meet in a vanishing point.

Q1)g Representing projection equation as linear equation is 2D - co-ordinate system is not possible because. in that equation we divide by z :- $\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{z}\begin{bmatrix} f & 0 \\ 0 & f \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}$ $\left.\begin{matrix} u = \frac{fx}{z}, \\ v = \frac{fy}{z} \end{matrix}\right\}$

But projection equation is represented using linear equation in Homogeneous co-ordinates as follows:-

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} f & & \\ & f & \\ & & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix} \Rightarrow \left.\begin{matrix} u = fx \\ v = fy \\ w = z. \end{matrix}\right\}$$

$\therefore$ $u = \frac{fx}{z}$, $v = \frac{fy}{z}$ solve projection equation as above.

$u = fx$
$v = fy$
$w = fz$ } Projected point in 2D

To Homogeneous the projection point in 2D we divide by w

$$\therefore u = \frac{u}{w} = \frac{fx}{z}$$

$$v = \frac{v}{w} = \frac{fy}{z}$$

These (U, V) are projected points in 2D and same projected equations as before but are represented linearly in homogeneous co-ordinates

Q1) h. Projection matrix : ~~M = K[I]~~

$M = K[I|0]$

$M \rightarrow 3 \times 4$ matrix

$0 \rightarrow 3 \times 1$ matrix

$I \rightarrow 3 \times 3$ matrix.

$K \rightarrow 3 \times 3$ matrix

$M \rightarrow 3 \times 4$ matrix

Projection matrix

Q1) i

$$M = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 1 & 2 & 1 & 2 \end{bmatrix}$$

3D point $P = [1, 2, 3]$, find the co-ordinate of 2D point p obtained by projecting P using M.

Given point in 3DH co-ordination :-

$[1\ 2\ 3\ 1]$.

$\therefore P_{2DH} = M\ P_{3DH}$

$$\rightarrow \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 1 & 2 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} 1+4+9+4 \\ 5+12+21+8 \\ 1+4+3+2 \end{bmatrix} = \begin{bmatrix} 18 \\ 46 \\ 10 \end{bmatrix}$$

$$P_{2D} = \begin{bmatrix} \frac{x}{w} \\ \frac{y}{w} \end{bmatrix} = \begin{bmatrix} \frac{18}{10} \\ \frac{46}{10} \end{bmatrix} = \begin{bmatrix} 1.8 \\ 4.6 \end{bmatrix}$$

Q2) Modelling x' formaⁿ

a) point (1, 1) find its co-ordinates after translating it by (2, 3)

→ x' formation matrix compulaⁿ

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & & tx \\ & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$x' = x + tx$$
$$y' = y + ty$$
$$1 = 1.$$

$$\therefore \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\therefore \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 3 \\ 4 \\ 1 \end{bmatrix}$$

$$\Rightarrow P_{2D} = \left(\frac{3}{1}, \frac{4}{1}\right) \Rightarrow (3, 4)$$

2. b) point $P = (1, 1)$
Scale by $(2, 2)$.

Perform Computation using x' formaⁿ matrix

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix}$$

$$P_2 = \left(\frac{2}{1}, \frac{2}{1}\right) \Rightarrow (2, 2)$$

2c] Given points (1, 1).,
find co-or. after rotating it by 45°.

2D - Rotation :

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$ } 2D rotation about origin

$$= \begin{bmatrix} \cos 45 & -\sin 45 \\ \sin 45 & \cos 45 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

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

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \sqrt{2}/2 & -\sqrt{2}/2 \\ \sqrt{2}/2 & \sqrt{2}/2 \end{bmatrix}$$

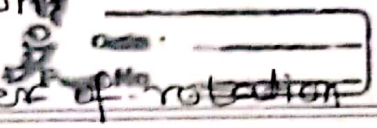
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{1\sqrt{2}}{2} + \frac{1\sqrt{2}}{2} \end{bmatrix} \Rightarrow \begin{bmatrix} 0 \\ \frac{2\sqrt{2}}{2} \end{bmatrix} \Rightarrow \begin{bmatrix} 0 \\ \sqrt{2} \end{bmatrix}$$

$$\therefore (x', y') = (0, \sqrt{2})$$

$(P'x, P'y)$ = Co-ordinates of point after rotation.

(Px, Py) = Points to be rotation

(Ox, Oy) = Co-ordinates of center of rotation



Q2d. point (1, 1), find co-ordinates after rotating it by 45° about point (2, 2).

$$\begin{bmatrix} P'x \\ P'y \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} Px - Ox \\ Py - Oy \end{bmatrix}$$

Apply:- (Rotation by 45° and then Translation) = (T)(R)

$$\therefore P'x = \cos 45 \times (1-2) - \sin 45 (1-2) + Ox$$

$$= \frac{\sqrt{2}}{2} \times (-1) - \left(\frac{\sqrt{2}}{2} \times -1\right) + 2$$

$$= -\frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2} + 2$$

$$\boxed{P'x = 2}$$

$$P'y = \sin 45 (1-2) + \cos 45 (1-2) + 2$$

$$= \frac{\sqrt{2}}{2} \times (-1) + \frac{\sqrt{2}}{2} \times (-1) + 2$$

$$= -\frac{\sqrt{2}}{2} + \left(-\frac{\sqrt{2}}{2}\right) + 2$$

$$= -\frac{2\sqrt{2}}{2} + 2$$

$$\boxed{P'y = 2 - \sqrt{2}}$$

Q2) e

Rotation + Translation :-

This transformation is also known as 2D rigid body motion. or It can be written as :- $x' = Rx + t$

or

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

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

is an orthonormal rotation matrix with $RR^T = I$ and $|R| = 1$.

Q2) f $M_{2DH} = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

⇒ The above matrix applied to point P indicates scaling by the ~~factor of~~ 3 is x- direction, by ~~factor of~~ 2 in y direction ~~and by 1 in Z direction~~

Q2) g. $M_{2DH} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix}$

⇒ The above matrix applied to point P indicates translation by (1, 2).

Q2)h $M_{2DH} = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

→ The above matrix is a Scaling matrix, hence the Inverse of the Scaling matrix can be represented as follows:-

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

Q2)i $M = R(45)\,T(1,2)$ be transforma^n matrix in Homogeneous co-ordinate. Rotation by 45° & transla^n by (1, 2)

$$R_4^{-1}(\theta) = R_4^T(\theta)$$

$\therefore$ if $R_4(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$

$$R_4^T(\theta) = R_4^{-1}(\theta) \Rightarrow \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

$$T^{-1}(t_x, t_y) = T(-t_x, -t_y) = T(-1, -2)$$

$$R^{-1}T^{-1} = \begin{bmatrix} \cos 45 & \sin 45 \\ -\sin 45 & \cos 45 \end{bmatrix} \begin{bmatrix} -1 \\ -2 \end{bmatrix}$$

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

$$\Rightarrow \begin{bmatrix} \frac{-1 \times \sqrt{2}}{2} + \frac{-2 \times \sqrt{2}}{2} \\ \frac{-1 \times \sqrt{2}}{2} + \frac{-2 \times \sqrt{2}}{2} \end{bmatrix}$$

Q2) j Vector perpendicular to vector (1, 3)

$$\Rightarrow \begin{bmatrix} -\sqrt{2}/2 & * & \sqrt{2} \\ & \frac{3\sqrt{2}}{2} & \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} \sqrt{2}/2 \\ -3\sqrt{2}/2 \end{bmatrix}$$

j) Vector perpendicular to j(1,3)

$\therefore 1x + 3y = 0$

$\therefore$ x and y should be such that it should equate to zero

$\therefore x = -3$ and $y = 1$

$(a) \cdot b = \|a\| \cdot \|b\| \cdot \cos\theta = \|a\| \cdot \|b\| \cdot \cos(90^\circ) = 0$

$\therefore$ Vector $(-3, 1)$ is perpendular to $j(1,3)$

Q2) Find projection of vector (1,3) onto the direction defined by vector (2,5)

$$\text{Proj}_{(b)}(a) = \frac{(a \cdot b)}{|b^2|} b$$

$$a \cdot b = (1, 3) \cdot (2, 5) = 2 + 15 = 17$$

$$|b^2| = (2\ 5) = 4 + 25 = 29$$

$$\therefore \frac{(a \cdot b)}{b^2} \times b = \frac{17}{29} \times [2, 5]$$

$$\Rightarrow \left[\frac{17}{29} \times 2, \frac{17}{29} \times 5\right]$$

$$\Rightarrow [1.172, 2.93]$$

Scalar Projection ?

Q3] General Camera Model:

a) The transformation between world and camera is obtained by aligning the camera with the world. For this we assume the camera is rotated by R and translated by T with respect to world, to move from world co-ordinates to camera co-ordinates. However, this 3D to 2D projection equation is in same co-ordinate system. But we want to relate 3D points in world co-ordinate system to 2D points in image co-ordinates (pixels). Hence we need a General Camera Model:-

$$\underset{\text{2DH.}}{\begin{bmatrix} U \\ V \\ W \end{bmatrix}} = \underset{(3\times4)}{\begin{bmatrix} f & & & \\ & f & & \\ & & 1 & \end{bmatrix}} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \underset{(3DH)}{\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}} = K[I|0] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Have: $p^{(c)} = K[I|0]\, p^{(c)}$

Want: $p^{(i)} = M_{i \leftarrow w}\, p^{(w)}$

b) Given that the camera is rotated by R & translated by T w.r.t world, write the transformation matrix that will convert coordinates from world to camera coordinates

⇒ In Camera Co-ordinates:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} f & & \\ & f & \\ & & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = K[I|0] \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Have: $p^{(c)} = K[I|0]\ P^{(c)}$

Want: $p^{(i)} = M_{i \leftarrow w}\ P^{(w)}$

$\therefore p^{(i)} = M_{i \leftarrow c}\ p^{(c)} = M_{i \leftarrow c} K[I|0] P^{(c)}$

$= M_{i \leftarrow c} K[I|0] M_{c \leftarrow w}\ P^{(w)}$

when camera rotated & translated w.r.t world

2DH 2DH

$p^{(i)} = \underset{3\times3}{M_{i \leftarrow c}}\ P^{(c)}$ ↓ 3×4 matrix

$= \underset{3\times3}{M_{i \leftarrow c}} \cdot K[I|0]\ P^{(c)}$ 3DH.

↓

$= M_{i \leftarrow c}\ K[I|0]\ (M_{c \leftarrow w}\ P^{(w)})$

when camera is translated by T & rotated by R w.r.t world along camera with wold

$M_{c \leftarrow w} = \tilde{R}^{-1} \tilde{T}^{-1}$

$$M_{c \leftarrow w} = \tilde{R}^{-1} \tilde{T}^{-1}$$

$$= \begin{bmatrix} R & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & T \\ 0 & 0 \end{bmatrix}^{-1}$$

$$= \begin{bmatrix} R^T & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & -T \\ 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} R^T & -R^T T \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R^* & T^* \\ 0 & 1 \end{bmatrix}$$

$R^*, T^* \Rightarrow$ Rotation & Translation of world w.r.t camera

$R, T \Rightarrow$ Rotation & Translation of camera w.r.t world

(c) Given three unit vectors $\hat{x}, \hat{y}, \hat{z}$, write rotation matrix describing rotn of camera w.r.t world

$\rightarrow$ $R^T = \begin{bmatrix} \hat{x}_c^T \\ \hat{y}_c^T \\ \hat{z}_c^T \end{bmatrix}$

Q3)d $M = \begin{bmatrix} R^* & T^* \\ 0 & 1 \end{bmatrix}$

→ R^* and T^* are Rotation and Translation of world with respect to Camera.

Q3)e k_u = pixels per mm in the x-direction,
k_v pixels per mm in the y-direction.
$(u_0, v_0) = (512, 512)$ pixels.
Write the transformation matrix that will convert camera co-ordinates to image co-ordinates.

$$M = \begin{bmatrix} k_v & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} k_v & 0 & 512 \\ 0 & k_v & 512 \\ 0 & 0 & 1 \end{bmatrix}$$

Q3)f $M = K^* [R^* \mid T^*]$

K^* = Intrinsic Camera parameter:

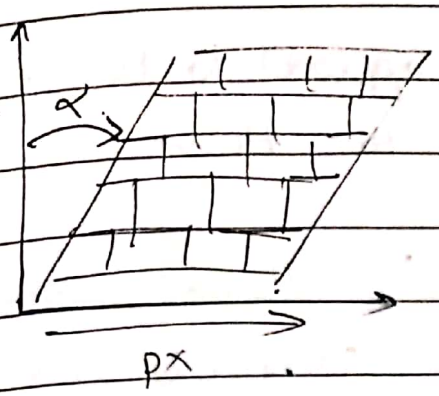
$$K = \begin{bmatrix} \alpha_u & & u_0 \\ & \alpha_v & v_0 \\ & & 1 \end{bmatrix}$$

focal length in pixels

R^*, T^* = extrinsic camera parameters

$p_x, p_y \Rightarrow$ Scale in x & y respectively relating pixels to mm

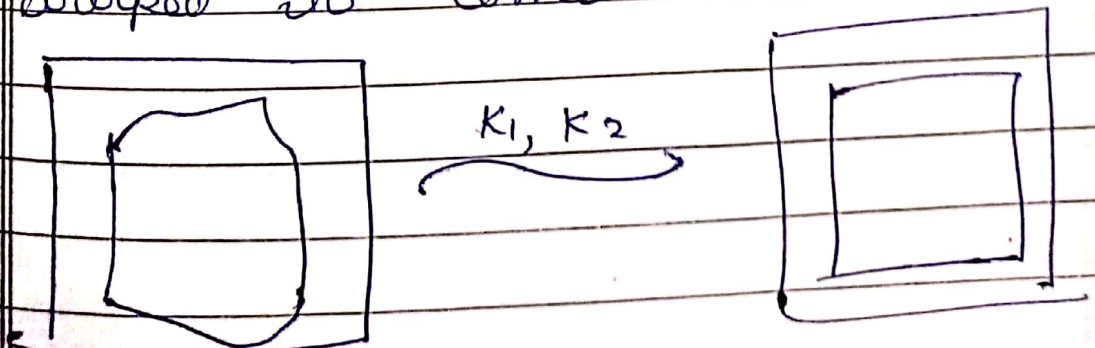
(Q3)g:



Due to the way the cells are scanned to produce the image there could be a little shift in the pixels. There could be a little shear of skewed x'formation.
Skew parameter is used where one needs sub-millimeter accuracy like some industrial inspection of computer chips or printed circuit boards..
Also it could happen that the sensor is not placed perpendicular to optical axis, hence 2D skew is included to camera model.

(Q3)h Radial lense Distortion

→ Once Radial lense Distortion paramaters (K_1, K_2) are determined, the image can be warped to correct the distortion.



- The straight lines appear to be curved in the image due to Radial lense distortion. This is seen in wide angle cameras such as security camera.

$$\Rightarrow \quad p^{(i)} = \begin{bmatrix} 1/\lambda & & \\ & 1/\lambda & \\ & & 1 \end{bmatrix} K^* \begin{bmatrix} R^* & | & T^* \end{bmatrix} P^{(w)}$$

$$\lambda = 1 + K_1 d + K_2 d^2.$$

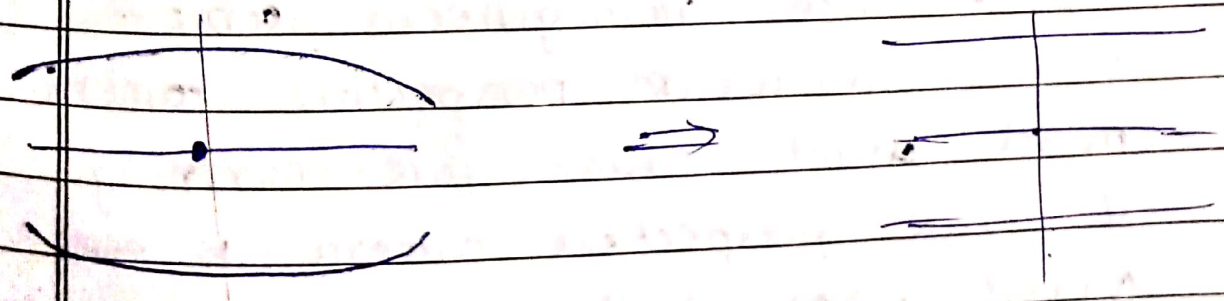
$K_1 \rightarrow$ linear distortion coefficient

$K_2 \rightarrow$ quadratic distortion coefficient.

$d \Rightarrow$ distance from centre.

* Complications added by adding Radial lense distortions

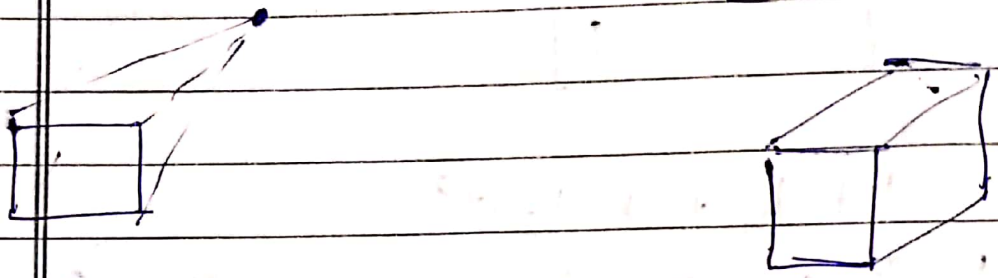
→ Radial lense causes the shrinkage as you go away from the centre.



Q3) (i) Weak-perspective ~~of~~ camera and affine Cameras.

Perspective. Weak

M (3×4) $M_{aff} = \begin{bmatrix} & & & \\ & & & \\ 0 & 0 & 0 & 1 \end{bmatrix}$



Perspective camera gives fore-shortning or a vanishing point.

In weak-perspective camera, there are no such effects of perspective such as fore-shortning.

⇒ 0 0 0 1 in last line of M_{aff} model of weak-perspective camera ~~causes~~ ~~fore~~ produce homogeneous compo.

→ 0 0 0 1 in weak perspective camera model won't cause fore-shortning.

→ A weak perspective camera is ~~same~~ correct when depth variation in the scene is small compared with distance from

camera.

$e = |M_\infty P - MP| = \frac{\Delta}{d_o} (MP - P_o)$.

$\Delta \rightarrow$ depth variation.

$d_o \rightarrow$ distance from camera

$MP, P_o \rightarrow$ distance from Centre.

✱ Affine Camera

$M_{affine} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ 0 & 0 & 0 & 1 \end{bmatrix}$.

Affine.

→ $a, b \ldots h \rightarrow$ arbitary & numbers.

→ Rotation & and translation create the matrix

→ It is a computation model, & the numbers $a, b, \ldots h$ donot ensure that the camera is similar to the real camera model.

→ Projections under an affine camera is a linear mapping on non-homogeneous co-ordinates composed with translation

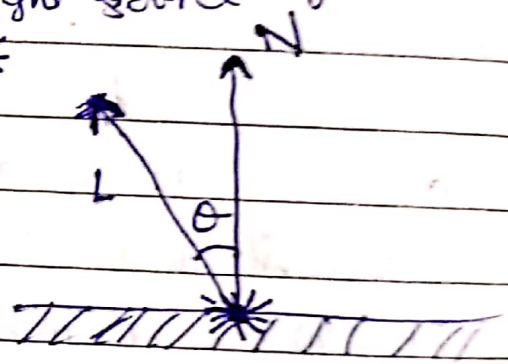
Q4] Color and Photometric image formation

a) Surface Radiance and Image Radiance

Surface Radiance :-

→ For surface radiance rotate the light source intensity to reflected light intensity.

→ Lambertian surface are diffused reflection.
→ They take light and reflects light uniformly.



$\cos\theta = N \cdot L$

$N \cdot L < 0 \Rightarrow$ surface not visible

$I_{ret} = I \cdot \rho \cdot \cos\theta$

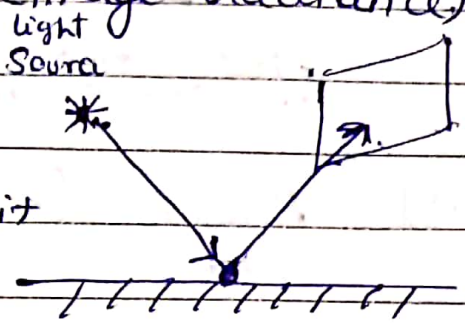
(Intensity of reflection) $= I \cdot \rho \cdot (N \cdot L)$

(intensity of source) {reflection coefficient}

ρ = Surface albedo $\in [0, 1]$

Q4] Surface Radiance & Image Radiance.

Q4a] * Relate light in the scene (surface radiance) to light in the image (image radiance)



L(P) = Surface radiance.
→ Power of light per unit area reflection from surface.

E(P) = Power of light per unit area received at the image.
This is image Radiance.

Q4b] $E(P) = L(P) \frac{\pi}{4} \left(\frac{d}{f}\right)^2 (\cos\alpha)^4$

f → focal length, bigger (f), smaller image radiance.
E(P) → light of image.
L(P) → light of surface.
α → angle between principal axis and surface normal
d → diameter of lense.
d ↑ = E(P) ↑ α ↑ = E(P) ↓
f ↑ = E(P) ↓

Q4 c) "ρ" is Albedo of a surface. It's range is between 0 and 1. It* is the fraction of incident light that the surface reflects.. It is intensity of the source.

Q4) d Reason for using RGB color model

→ Humans ~~eye~~ have R, G, B receptors. They imitate real colors with combination of R, G, B receptors. Human eyes have RGB sensors and are more sensitive to green.

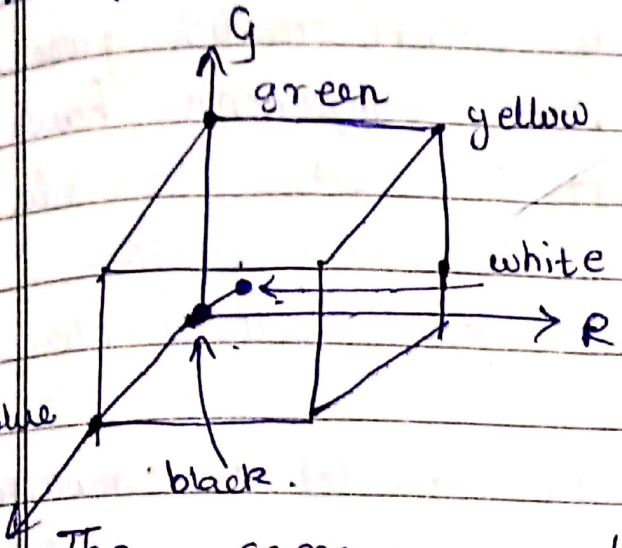
Q4) e What are colors along the line that connects (0, 0, 0) with (1, 1, 1)

→ Colors along (0, 0, 0) & (1, 1, 1) are grayscale colors

(0, 0, 0) give white and (1, 1, 1) give black.

Q4] Explain the way by which RGB colors are mapped to real world colors.

→

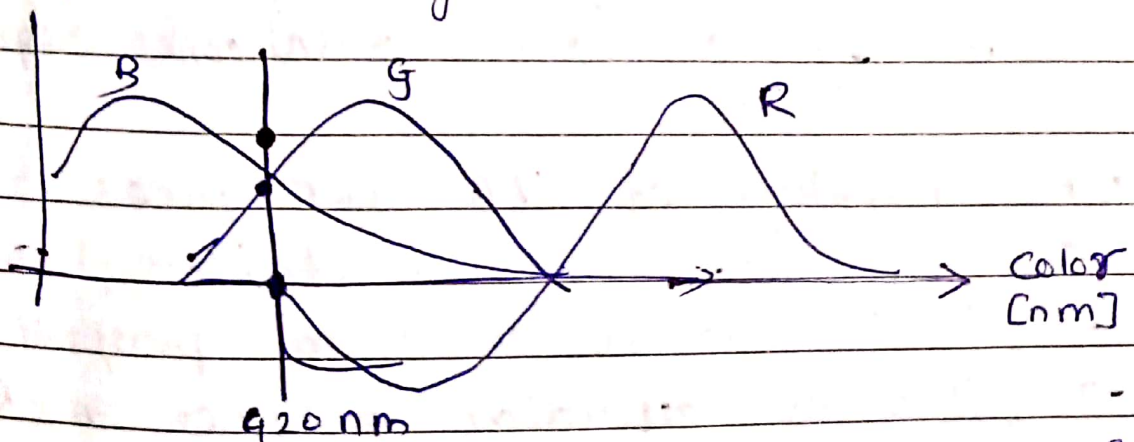


RGB: additive model

The cameras and displays & our human eyes work with the R, G, B sensor

→ The R, G, B colors are added with different proportions to get the desired target colours

→ Then for example: to produce color Brown which is at 420 nm: mix the beam lights of R, G, B in the desired proportions. Originally



Originally R is at 645nm; G visible at 526 nm & B is at 444nm

When one of the components has the strongest intensity, the color is a hue near the primary color (reddish, greenish, bluish) & when two components have the same strongest intensity the colour is a hue of a secondary color (shade of cyan, magenta, yellow) etc.

Q4)g CIE RGB color model & conversion to XYZ model.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.49 & 0.31 & 0.20 \\ 0.177 & 0.813 & 0.011 \\ 0.00 & 0.01 & 0.94 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}.$$

→ Importance of chrominance Y.
→ Y corresponds to relative luminance; Y also carries color information related to eye's "M" (yellow-green) cone response

Q4)h Advantage of LAB color Space.
→ Non-linear conversion of RGB
→ Works similar to human perspection
→ Measures euclidian distance in RGB Space doesnot correspond to human perception

Eg:- Euclidean distance in RGB might be large for similar colors and could be treated at very different two colors.

→ whereas distance in LAB space does correspond to perception of humans

RGB → $L^*a^*b^*$ → compare colors.

$$|C_1 - R| < |C_2 - R|$$

Eg: skin colors where C_1 is more similar to R.
C_1, C_2 → Colors and R is reference.
If $C_1 - R$ is small then C_1 is similar to R.
L, a, b can be used in skin colors present of different body parts, which can differentiated correctly in L, a, b

Q5] Noise filtering:

a) SNR is energy of signal to energy of noise
E_s = energy of signal
E_n = energy of noise
If E_s is high Higher SNR means good signal, lower SNR means poor signal and more noise.

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

2 ways to handle variace of noise in image

σ_n^2 ① variance for multiple frames of a static scene.

OR

② Variance in uniform image region.

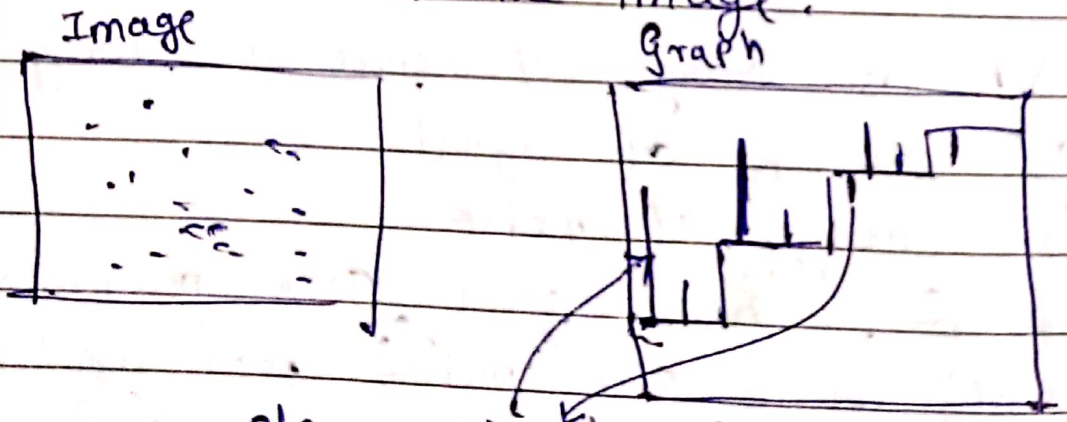
$\bar{I}$ ⇒ avg intensity of image
$I_{i,j}$ ⇒ Intensity at each point.
n ⇒ no. of pixels

(a)

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

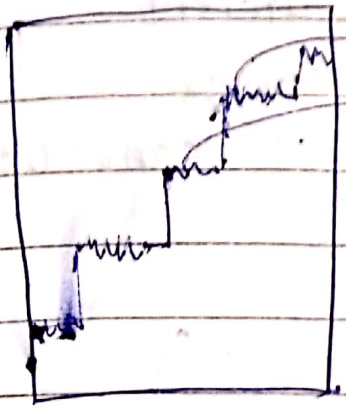
If 0dB SNR means 'Es is 10 times large than E

b) Impulsive noise is also called as salt and pepper noise. It appears as small dots on the image.



clear spikes of the noisy dots in the image represented in the graph

Gaussian noise has gaussian distribution on noise in the image.



→ Remove the noise using smoothing.
→ Median-filters are better for removing impulsive noise. As they are calculated by first sorting all the pixel value from the surrounding neighborhood into numerical order and then replacing the pixel by the middle pixel value, thus eliminating the outlying values either high or low.

c) -convolution filter:-

(C) (3×3)

1	1	1
1	1	1
1	1	1

Let us consider the image is:-

(3×3) (I)

2	2	2
2	2	2
2	2	2

$\Rightarrow I * C_{filter}$

$\Rightarrow \begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 6 \\ 6 \\ 6 \end{bmatrix} = 18$

d) Given need of derivative of an image convolved with a filter explain how the operation be more efficient.

⇒ Assume: $f \rightarrow$ filter; $g \rightarrow$ image

⇒ Using the property of convolution

$$\frac{d}{dx}(f \times g)$$

$$= \left(\frac{d}{dx} f\right) * (g) = f * \left(\frac{d(g)}{dx}\right)$$

⇒ Hence using this property take derivate of one of the component and then apply convolution.

e) Ways to Handle boundaries during convolution:

① Zero padding :- Pad zero to the boundaries:-

0	0	0	0
0	5	6	7
0	8	9	1
0	1	2	3

↑ Enlarge the image and pad zeros & then use convolution.

② Mirror padding

⇒ mirror eliminates boundary effect

⇒ Disadv: unnecessary amplification of image.

	5	6	7
5	5	6	7
8	8	9	10
11	11	12	2

(3) Ignore:

5	6	7
8	9	10
1	2	2

→ Ignore the boundaries pixels, hence the resultant image gets smallers.

→ Store ~~resultant~~ ~~the~~ results in a new image or in an flood array.

(b) Smoothing filter (3x3) = $\frac{1}{9}$ × [1 1 1; 1 1 1; 1 1 1]

⇒ Sum of all the weights = 9

→ We add up all the weights to decrease the intensity after each convolution

(g) 2D Gausian filters using 1D Gaussian

⇒ Use Separable Implementation:

Instead of convolving with 2D Gaussian convolve with 1D Gaussian along rows then along columns

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

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

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

⇒ which one is better?

Suppose: $M \times N$ image & $m \times m$ filter.

→ ~~Has one~~

∴ Complexity :- of using :-

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

Two 1D passes : $2 \times M \times N \times m$ operations

∴ $2MNm < MNm^2$

Hence using two 1D Gaussian filter is better.

⇒ T

~~Q(b) 1D Gaussian filter :- with $\sigma = 2$~~

~~Size of filter.~~

→ Is it possible to ~~implement~~ 2D Gaussian filter in this way?

→ Yes, first take convolution ~~ation~~ "y" direction and then take convolution along "x" direction.

→ Also we can use Repeated application

~~Q(c) 1D Gaussian filter with $\sigma = 2$~~

~~Find size of filter~~

of

⇒ It is possible to implement 2D Gauss^n filter using two 1D Gaussians such that take convolution in 'y' direction first then in x direction

→ Also we can use repeated applica^n of Gaussian when

$$I * G_1 * G_2 = I * G_{\sqrt{\sigma_1^2 + \sigma_2^2}}$$

extra or point

$$\Rightarrow 2 \times (M \times N \times m^2) < (M \times N \times (2m)^2)$$

Q5)h 1D Gaussian filter with $\sigma = 2$

∴ size of filter.

∴ filter with $m = 5\sigma$

∴ $m \rightarrow$ size of filter

∴ $m = 5 \times 2 = 10$

⇒ size of filter is 10.

Q5)i How Gaussian image pyramid produced?

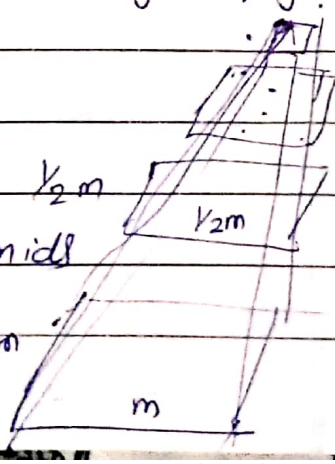
→ It is produced using Multiple Scale Analysis techniques using image pyramids

→ To where:-

$m = 2^J \Rightarrow J = \log_2^m$

$m \rightarrow$ size of image window.

$J \Rightarrow J$ levels of image pyramids

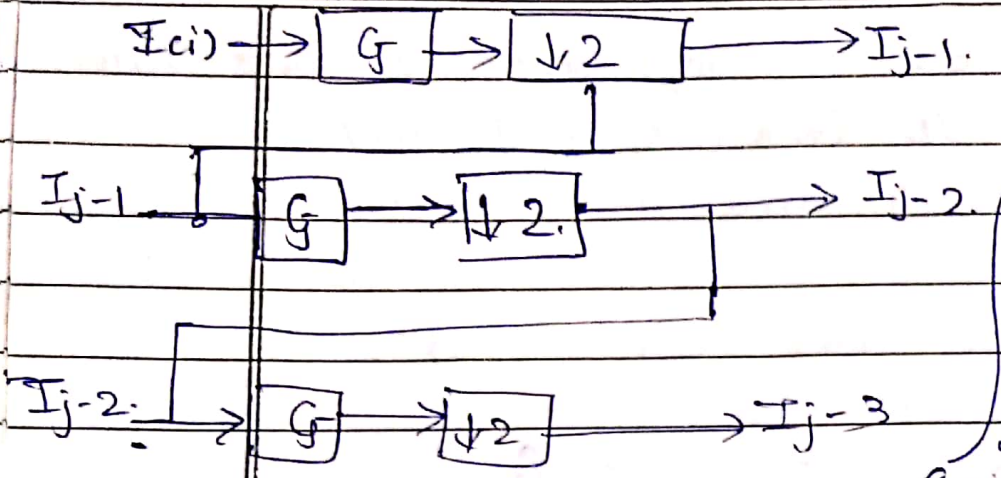


total # pixels:-

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

⇒ Analyse the image at multiple scales, ~~use~~ pool the image & apply convolution at each ~~scale~~.

→ Gaussian pyramid.



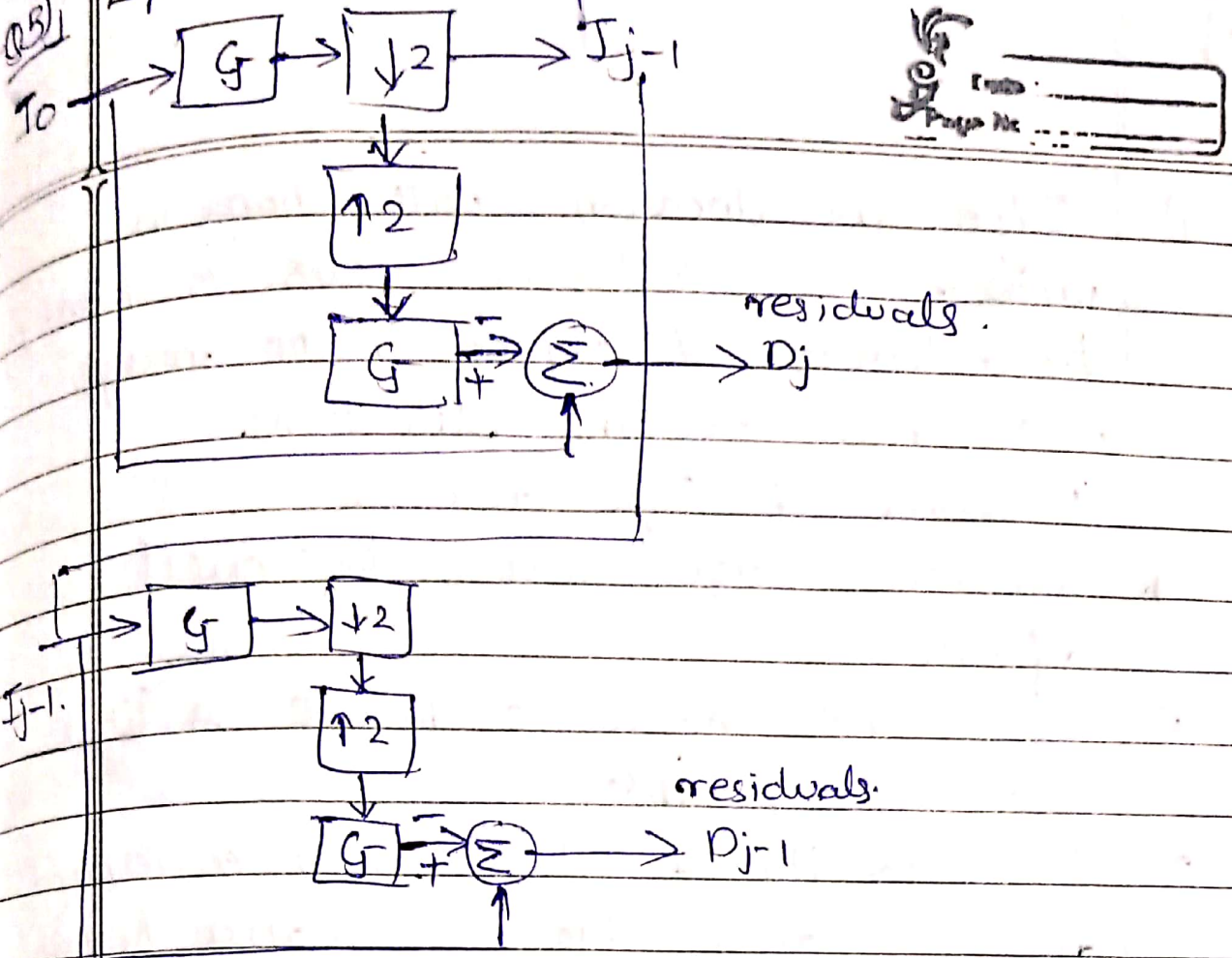
while pooling the images at each level, there is more sampling and may result in aliasing, hence use Gaussian filter resulting into such Gaussian ~~image pyramids~~

☆ Reason to Create ~~image~~ such pyramids

→ In order to analyse objects of varying size in an image we use image ~~pyrag~~ pyramids. ~~We~~ use Gaussian image pyramids to remove the effect of aliasing due to sampling the images to these pyramids.

→ Amount of processing which is extra is ~~less & less~~ ~~4~~ less than $\frac{4}{3}m^2$ where m is ~~max~~ size of image, hence we add 30% more pixels to the image.

Q5) Laplacian pyramids and its use?



→ Used mainly for image compression.

→ In these pyramids there is both upsampling and downsampling of the image; To regain the lost pixels downsampling in first step, we again ~~upsampling~~ do upsampling and smoothen it using Gaussian filter. Then we take the resultant image & compare with original image (Io) and get residuals by taking difference between them. Then we make pyramid of these residuals. This is laplacian pyramids. The residuals are small and close to zero, hence it is easier to compress the image in this way.

Q6] Edge Detection.

a) Edge is location with change in image. Edge Detection helps to identify boundaries of object in the image with such changing locations.

Properties of Edge detection:

1. Produced edges should be well defined,
2. Background should contribute as little noise as possible.
3. The intensity of edges should correspond as close as possible to what a human would perceive.

b) Basic Steps of Edge Detection.

→ Smoothing
→ Enhance edges.
→ Detect edges.
→ Localize edges are 4 basic steps of Edge Detection.

Smoothning :- is required to reduce noise without affecting the edges.

Enhancement :- It is calculated using the gradient magnitude. Enhancement gives emphasis to the points where intensity of change is more.

Localization: Determins the exact location of an edge.

Q6.d) Two filters to compute image gradient.

→ ① forward difference filter.
② Central difference filter

forward difference filter

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

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

$h=1$

$$\Delta x = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \quad ; \quad \Delta y = \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}$$

⇒ Calculate gradient of next pixel from current pixel.

Here $h=1$, because take the immediate next pixel

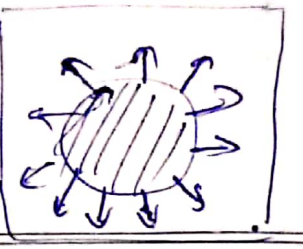
Central difference filter: Calculate gradient ~~w.r.t two both of pre~~ of both previous and next pixel w.r.t current pixel.

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

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

filter used = $\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$

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



→ Image gradient & it's use.

→ Image gradient is a directional change in the intensity or ~~color~~ of in an ~~img~~ image. ~~The gradient~~ Hence it is used to calculate the change of intensity of an image. (maximum change of)

(d) ~~Sobo~~ Sobel filter.

→ Smooth and the ~~take~~ Derivative. ~~where:-~~ ~~sm~~ to ~~create~~ the Sobel filter. This filter takes convolution of smoothing and derivative filters.

① Smoothing filter = $\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$

② take derivative in X and Y direction:-

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

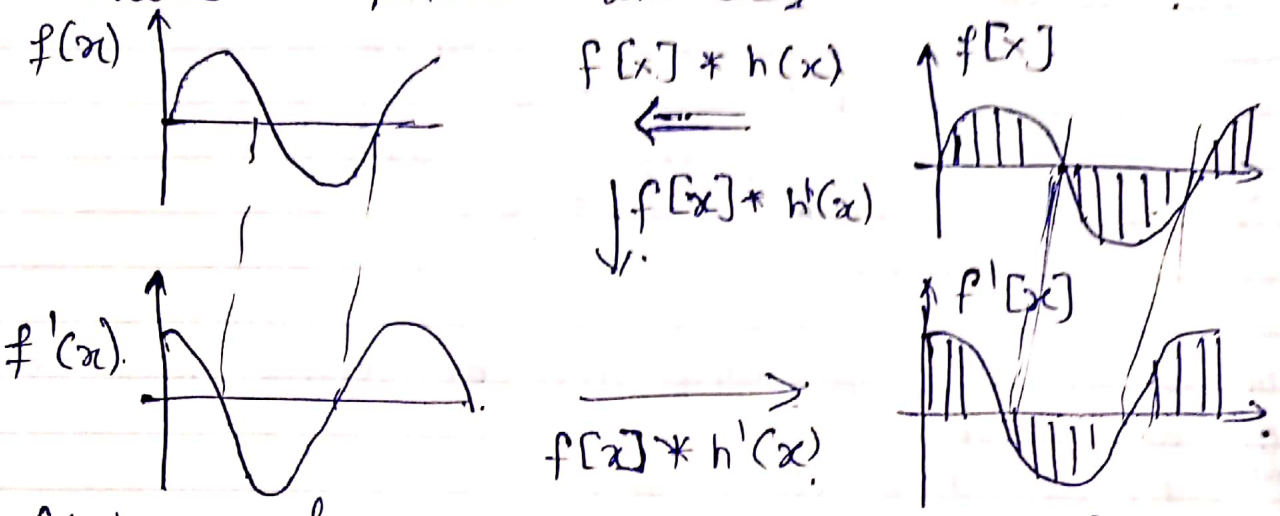
$$\therefore \Delta x = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

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

⇒ This filter will take gradient in on direction and smoothen ~~th~~ in opposite direction.

Q e] Generate more accurate derivative filter with an arbitary σ. Elements of a filter for more accurate derivative computation with σ = 2

Ans:



h(x) is a function which reconstructs to continuous function.

In computers we have discrete image, our desired output is to get a sample derivative. Discrete images cannot give sample derivates, hence we use h(x) and reconstrud the image in continuous form and then find derivative of it.

$\therefore I_x = I * G_x$; $I_y = I * G_y$.

Use separable property of Gaussian:

$I_x = I * G'[x] * G[y]$

$I_y = I * G'[y] * G[x]$

G'[x], G'[y] ⇒ 1D convolⁿ with horizontal & vertical Gaussian respectively.

G[x]

G'[x] → 1D Convolⁿ with Gaussian Derivative.

G'[y] → 1D convolⁿ with vertical Gaussian Derivation

G[y] → 1D convolⁿ with verticle Gaussian

G[x] → 1D convolⁿ with horizontal Gaussian.

1. $I_x = I * \left(\frac{-x}{\sigma^2}\right) e^{-x^2/2\sigma^2} * e^{-y^2/2\sigma^2}$

$\therefore$ when ~~σ^2~~ $\sigma = 2$ $\therefore$ $\sigma^2 = 4$.

$= I * \left(\frac{x}{4}\right) e^{-x^2/8} * e^{-y^2/8}$

$I_y = I * \left(\frac{-y}{4}\right) e^{-y^2/8} * e^{-x^2/8}$.

(f) Localization of edge using first or second order derivative of an image.

→ 2nd order derivative of image :-

$\Delta I = \nabla^2 I = I_{xx} + I_{yy}$.

$I_x = I(x+1) - I_x$.

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

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

$= I(x+1) - 2I(x) + I(x-1)$

$\therefore I_{xx} = \boxed{1, -2, 1}$

$I_y = I(y+1) - I_y$

$I_{yy} = [I(y+1) - I(y)] - [I(y) - I(y-1)]$

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

$= I(y+1) - 2I(y) + I(y-1)$.

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

0	1	0
1	-4	1
0	1	0

(g) Assume $\sigma=1$, write laplacian of Gaussian. (LOG) filter using σ. Explain how to use LOG to detect edges.

⇒ $N = \nabla^2 (I * G) = \nabla^2 G * I$

Now :-

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

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

here $\sigma = 1$.

$\therefore N = \left[(r^2 - 2)\, e^{-x^2/2} \right] * I$

* Edge Detection using LOG ⇒.

* Step (1): Compute LOG, $H = \nabla^2 G * I$

Step (2): find the 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 transition $0 \to 1$, $1 \to 0$.

Scan ~~fro~~ images from left to right & top to bottom

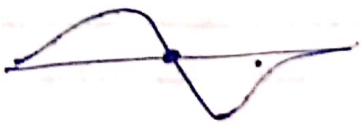
h) Difference between Canny Edge Detection and Standard Edge Detection. That does not use directional derivatives. What is condition for detecting edge candidate in Canny?

⇒ Standard edge Detection does not use directional derivatives to detect edges at zero crossings of second order directional derivative taken along gradient.

n = gradient.

if $|M| > T$ detect

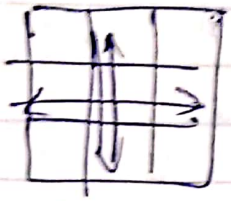
edges at zero crossing of $\frac{d^2}{dx^2}(I * G)$



Condition ⇒.

If gradient magnitude is large enough ($|\nabla m| > \gamma$), detection should be attempted

Non-maximum Suppression ⇒
Helps to find local maximum of gradient magnitude in direction of gradient.



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

$\theta = \tan^{-1}\left(\frac{I_y}{I_x}\right)$ ⇒ direction of gradient

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

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

θ is angle of gradient using which we can find direc^n.
To find maximum value we need to compare with ~~neig~~ neighbouring value.

Hysteresis thresholding ⇒
use T_H to short tracking and T_L to continue.
∴ $(T_H > T_L)$.
① Initialize array of visited pixel. $V(i,j) = 0$
② Scan image from left to right and top to bottom if $|V(i,j')|$ && $|\nabla I| > T_H$ then start tracking on edge.
③ Search for additional neighbors in direction orthogonal to ∇I such that $|\nabla I| > T_L$.
One threshold to start tracking and another threshold to continuing tracking.