Name : Yash Manharbhai Patel

Student ID : A20451170

Course : CS512 – Computer Vision

Semester : Fall20

CS-512 - CV - Assignment-1

Ans-1) a) $f = 10$

$p = (3, 2, 1)$

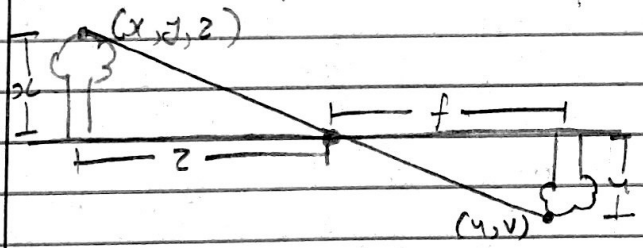
$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{z}\begin{bmatrix} -f & 0 \\ 0 & -f \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{1}\begin{bmatrix} -10 & 0 \\ 0 & -10 \end{bmatrix}\begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

$$= \begin{bmatrix} -30 \\ -20 \end{bmatrix}$$

$u = -30, \quad v = -20$

b)

Model 1: image plane is behind the center of projection.

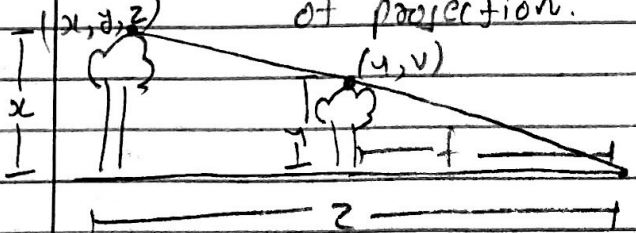


In this model we get converse image. (reverse oriented)

Here image coordinate is $(u, v) = \left(\frac{-fx}{z}, \frac{-fy}{z}\right)$

for every point in real-world we get points in image with negative sign.

Model 2: image plane is in front of the center of projection.



In this model we get get same oriented image

Here, image coordinate is $(u, v) = \left(\frac{fx}{z}, \frac{fy}{z}\right)$

So, orientation of the object is main difference b/w both models.

I would say Model 1 corresponds better to a physical pinhole camera. when model 2 helps to understand pinhole camera betterly

c) image coordinate, $(u, v) = \left(\frac{-fx}{z}, \frac{-fy}{z}\right)$

According to above equestion,

when the focal length gets bigger, projection of object gets bigger too, (zoom in) means coordinate of points gets bigger

when the distance to the object gets bigger. projection of object gets smaller too, (zoom out) means coordinate of points gets smaller

d) 2D point $(u, v) = (1, 1)$
2DH point $(u, v, w) = (1, 1, 1)$

default value of w is 1.

for another 2DH point multiply coordinates with n, where $n \neq 0$,
take $n = 2$,
$(u', v', w') = (2, 2, 2)$

corresponding 2D point,

$$(u'', v'') = \left(\frac{u'}{\omega'}, \frac{v'}{\omega'}\right) = \left(\frac{2}{2}, \frac{2}{2}\right) = (1, 1)$$
$$= (u, v)$$

e) 2DH point $(u, v, \omega) = (1, 1, 2)$

2D point $(u', v') = \left(\frac{u}{\omega}, \frac{v}{\omega}\right) = \left(\frac{1}{2}, \frac{1}{2}\right)$

f) 2DH point $(u, v, \omega) = (1, 1, 0)$

2D point $(u', v') = \left(\frac{u}{\omega}, \frac{v}{\omega}\right) = \left(\frac{1}{0}, \frac{1}{0}\right)$

whenever $\omega = 0$, we cannot convert 2DH point into original 2D point but, it shows the direction of a point It is called point of infinity.

g) In 3D to 2D projection,

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{z}\begin{bmatrix} f & 0 \\ 0 & f \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}, \quad \begin{aligned} u &= fx/z = fxz^{-1} \\ v &= fy/z = fyz^{-1} \end{aligned}$$

Here, parameter z^{-1} makes projection equation non-linear

In 3D to 2DH projection,

$$\begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad \begin{aligned} U &= fx \\ V &= fy \\ W &= z \end{aligned}$$

In 2DH to 2D

$$u = \frac{U}{W} = \frac{fx}{z}, \quad v = \frac{V}{W} = \frac{fy}{z}$$

So, here parameter W makes it possible to write projection equestion as a linear

h) $M = K[I|O]$

$\dim(M) = 3 \times 4$
$\dim(K) = 3 \times 3$
$\dim(I) = 3 \times 3$
$\dim(O) = 3 \times 1$

i) $M = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 1 & 2 & 1 & 2 \end{bmatrix}$, 3D point $P = [1, 2, 3]$
3DH point $P = [1, 2, 3, 1]$

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 1 & 2 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 18 \\ 46 \\ 10 \end{bmatrix}$$

2DH Point $P' = (u, v, w) = (18, 46, 10)$
2D Point $P' = (u', v') = \left(\frac{u}{w}, \frac{v}{w}\right) = \left(\frac{18}{10}, \frac{46}{10}\right)$
$= (1.8, 4.6)$

Ans-2 a) Point $(u, v) = (1, 1)$

translate $t = (t_u, t_v) = (2, 3)$

$$\begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_u \\ 0 & 1 & t_v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \Rightarrow P' = \left[\begin{array}{c|c} I & t \\ \hline 0 & 1 \end{array}\right] P$$

$$P' = T(t)\, P$$

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

$$= \begin{bmatrix} 3 \\ 4 \\ 1 \end{bmatrix}$$

Translated point $(u', v') = (3, 4)$

b) Point $(u, v) = (1, 1)$

Scale $S = (S_u, S_v) = (2, 2)$

$$\begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = \begin{bmatrix} S_u & 0 & 0 \\ 0 & S_v & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \Rightarrow P' = \left[\begin{array}{c|c} S & 0 \\ \hline 0' & 1 \end{array}\right] P$$

$$P' = S(s)\, P$$

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

Scaled point $(u', v') = (2, 2)$

c) point $(u, v) = (1, 1)$

rotate $\theta = 45^\circ$

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

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

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

$$= \begin{bmatrix} 0 \\ \sqrt{2} \end{bmatrix}$$

rotated point $(u', v') = (0, \sqrt{2})$

d) point $(u, v) = (1, 1)$
rotate $\theta = 45^\circ$
about the point $(2, 2)$

$$P' = T((2,2))\, R(45)\, T((-2,-2))\, P$$

$$T((-2,-2))\, P = P_1 = \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix}$$

$$R(45)\, P_1 = \begin{bmatrix} \cos 45 & -\sin 45 \\ \sin 45 & \cos 45 \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \end{bmatrix} = P_2$$

$$= \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} -1 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ -\sqrt{2} \end{bmatrix} = \begin{bmatrix} u_2 \\ v_2 \end{bmatrix}$$

$$T((2,2))\,P_2 = P' = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ -\sqrt{2} \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 2-\sqrt{2} \\ 1 \end{bmatrix} = \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix}$$

new coordinates $(u', v') = (2, 2-\sqrt{2})$

e) Suppose, point of an object is represented as p. and point after combined transformation is p'.

first rotate and then translate an object.

$$p' = TRp$$

$$p' = \left[\begin{array}{c|c} I & T \\ \hline 0 & 1 \end{array}\right] \left[\begin{array}{c|c} R & 0 \\ \hline 0 & 1 \end{array}\right] p$$

$$p' = \left[\begin{array}{c|c} R & T \\ \hline R & 1 \end{array}\right] p$$

f) $$M = \left[\begin{array}{cc|c} 3 & 0 & 0 \\ 0 & 2 & 0 \\ \hline 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{c|c} S & 0 \\ \hline 0 & 1 \end{array}\right] \neq \left[\begin{array}{c|c} I & T \\ \hline 0 & 1 \end{array}\right] \neq \left[\begin{array}{c|c} R & 0 \\ \hline 0 & 1 \end{array}\right]$$

scaling translate rotate

for 2DH points, the effect of applying this matrix to transform is scaling

It's not rotation, because in $R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$ diagonal values should be same

g) $M = \left[\begin{array}{cc|c} 1 & 0 & 1 \\ 0 & 1 & 2 \\ \hline 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{c|c} I & T \\ \hline 0 & 1 \end{array}\right] \neq \left[\begin{array}{c|c} S & 0 \\ \hline 0 & 1 \end{array}\right] \neq \left[\begin{array}{c|c} R & 0 \\ \hline 0 & 1 \end{array}\right]$

translate scale rotate

The effect is translate

h) $M = \left[\begin{array}{cc|c} 3 & 0 & 0 \\ 0 & 2 & 0 \\ \hline 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{c|c} S & 0 \\ \hline 0 & 1 \end{array}\right]$

scale

$M^{-1} = \left[\begin{array}{cc|c} 1/3 & 0 & 0 \\ 0 & 1/2 & 0 \\ \hline 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{c|c} S^{-1} & 0 \\ \hline 0 & 1 \end{array}\right]$

$M^{-1} = \frac{1}{\det(M)} \operatorname{adj}(M)$

$= \frac{1}{6} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 6 \end{bmatrix} = \begin{bmatrix} 1/3 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

i) $M = R(45)\,T(1,2)$

$M^{-1} = (R(45)\,T(1,2))^{-1} = T^{-1}(1,2)\,R^{-1}(45)$

$M^{-1} = T(-1,-2)\,R(-45)$

$M^{-1} = T(-1,-2)\,R^{T}(45)$

j) $V = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$

Suppose; $V' = \begin{bmatrix} a \\ b \end{bmatrix}$ is a vector perpendicular to V.

dot product of perpendicular vectors is 0.

$$V \cdot V' = 0 \Rightarrow \begin{bmatrix} 1 \\ 3 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \end{bmatrix} = 0$$

$$\Rightarrow a + 3b = 0$$
$$\Rightarrow a = -3b$$

take $b = 1$, $\Rightarrow a = -3$

so, perpendicular vector is, $V' = \begin{bmatrix} -3 \\ 1 \end{bmatrix}$

k) $\vec{V_1} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$, $\vec{V_2} = \begin{bmatrix} 2 \\ 5 \end{bmatrix}$

projection of V_1 on $V_2 = \left(\frac{\vec{V_1} \cdot \vec{V_2}}{|\vec{V_2}|^2} \right) \vec{V_2}$

$$= \left(\frac{1 \cdot 2 + 3 \cdot 5}{\sqrt{2^2 + 5^2}^2} \right) \begin{bmatrix} 2 \\ 5 \end{bmatrix} = \left(\frac{17}{29} \right) \begin{bmatrix} 2 \\ 5 \end{bmatrix}$$

projection vector is $(34/29, 85/29)$.

projection scaler is $17/\sqrt{29}$.

Ans-3)

a)

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

Here we first convert world coordinate system to camera coordinate system and the camera to image coordinate system.

general projection matrix uses different coordinate system for camera and image, because

i) use translation because image origin might be different than camera origin

ii) use scaling because it requires conversion to pixels

$$M_{i \leftarrow c} = \begin{bmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

(u_0, v_0) translation for converting to image coordinates

(k_u, k_v) is for pixel scaling.

b) $$M_{c \leftarrow w} = \left[\begin{array}{c|c} R & 0 \\ \hline 0 & 1 \end{array}\right]^{-1} \left[\begin{array}{c|c} I & T \\ \hline 0 & 1 \end{array}\right]^{-1}$$

$$= \left[\begin{array}{c|c} R^T & -R^T T \\ \hline 0 & 1 \end{array}\right] = \left[\begin{array}{c|c} R^* & T^* \\ \hline 0 & 1 \end{array}\right]$$

R and T are rotation and translation of camera coordinates w.r.t world coordinates

R^* and T^* are rotation and translation of world coordinates w.r.t camera coordinates

c) $V = (V_x, V_y, V_z) = (1,1,1)$

$\hat{x} = (V_x, 0, 0) = (1,0,0)$
$\hat{y} = (0, V_y, 0) = (0,1,0)$
$\hat{z} = (0, 0, V_z) = (0,0,1)$

rotation angle θ about V axis

$$R = I + \sin\theta \cdot Q + (1-\cos\theta) Q^2$$

$$Q = \begin{bmatrix} 0 & -V_z & V_y \\ V_z & 0 & -V_x \\ -V_y & V_x & 0 \end{bmatrix}$$

$$R_V(\theta) = \left[\begin{array}{c|c} R & 0 \\ \hline 0 & 1 \end{array}\right]$$

d) $M = \left[\begin{array}{c|c} R^* & T^* \\ \hline 0 & 1 \end{array}\right]$

$R^* = R^T = R^{-1}$
$T^* = -R^T T$

Here R^*, T^* are rotation and translation of world w.r.t camera coordinates.
where R and T are rotation and translation of camera w.r.t world coordinates.

c) K_u pixels/mm in x-direction.
K_v pixels/mm in y-direction.

$T = (u_0, v_0) = (512, 512)$ pixels

$$M_{i \leftarrow c} = \begin{bmatrix} K_u & 0 & u_0 \\ 0 & K_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} K_u & 0 & 512 \\ 0 & K_v & 512 \\ 0 & 0 & 1 \end{bmatrix}$$

f) $M = K^* [R^* | T^*]$

$$K^* = \begin{bmatrix} fK_u & 0 & u_0 \\ 0 & fK_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

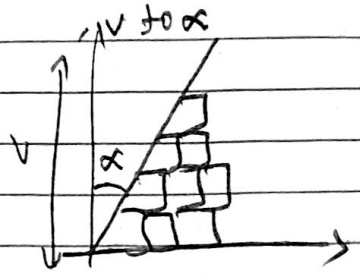
So, K^* contains intrinsic parameters of the camera.

Here, f is focal length.
K_u, K_v is scale (Pixels/mm)
u_0, v_0 is translation from camera to image.

$[R^* | T^*]$ contains extrinsic parameters of camera.
R^* and T^* are rotation and translation of world w.r.t. camera.

g) when optical sensors are not perpendicular to optical axis, then image might be skew.
when we include 2D skew parameter then we can make it correct.

$$M_{skew} = \begin{bmatrix} 1 & \tan\alpha & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



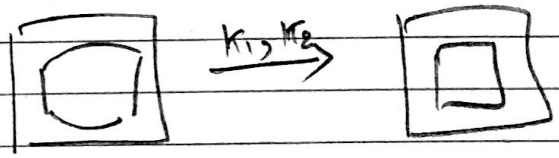
h) $p^{(i)} = \begin{bmatrix} 1/\lambda & 0 & 0 \\ 0 & 1/\lambda & 0 \\ 0 & 0 & 1 \end{bmatrix} K^* [R^* | T^*] P^{(w)}$

$\lambda = 1 + k_1 d + k_2 d^2$ — k_2: quadratic distortion coefficient; k_1: linear distortion coefficient

d = distance from center.

due to radial lens distortion we can see curve in the ~~straight~~ line projection.

This happens in wide angle cameras



i) In weak perspective camera we can't see perspective

$$M_{\infty} = \begin{bmatrix} & & & \\ & & & \\ & & & \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ ← last row is fixed

The last row is for homogeneous component.

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

$$e = |M_{\infty}P - MP| = \frac{\Delta}{d_0}(MP - P_0)$$

Δ: depth variation

$(MP - P_0)$: distance from center

d_0: distance from camera

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

computational model with combination of rotation, translation, scaling

It might not produce result like real camera

Ans-4

a)

$L(P)$ = Power of light per unit area reflects from surface
= Surface radiance

$E(P)$ = power of light per unit area received at the image.
= image irradiance

b)

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

$E(P)$: Light at image
$L(P)$: Light at surface
d: diameter of lens
f: focal length
α: angle b/w principal axis and surface normal

c) $I_{ref} = I \cdot S \cdot \cos\theta = I \cdot S \cdot (N \cdot L)$

S is surface albedo $\in [0, 1]$

Surface albedo is a coefficient of reflection which shows fraction of incident light that reflected from the surface.

If $S = 1$, surface reflects all lights
If $S = 0$, surface absorbs all lights

d) Red, Green and blue colors are sensitive to human eyes.
And RGB is an additive model means we can make other colors by adding proportion of R, G, B

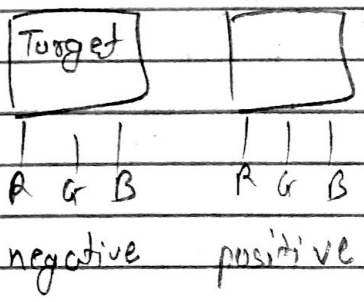
This RGB model is similar to human vision.

e) RGB color cube.

point (0,0,0) means no color will be reflected. so, it's a black color

point (1,1,1) means R, G, B colors will be reflected equally. So, it's a white color

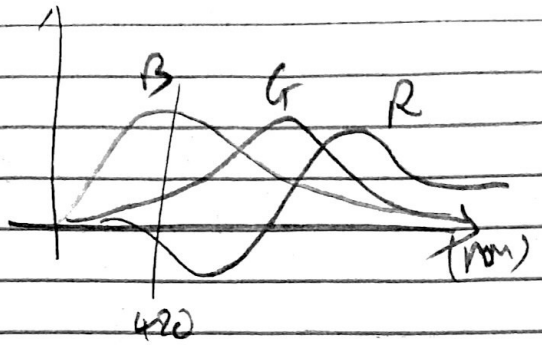
f) CIE tables are used to map RGB and real-world colors.



Here we use target RGB reference and another is to match with target.

as we twist the nobs it add colors in positive part and subtract colors from negative part

negative values are added to target to help match.



g) In XYZ color model

Y is for luminance which shows relative intensity of color that perceived by human eye.

Color with higher Y are perceived brighter.
Color with lower Y are perceived less brighter
color with equal y are perceived to have same brightness.

h) In RGB we can't find difference b/w two colors or can't compare colors with reference color accurately.

but if we convert these colors into LAB first then find difference or compare with reference color. then we get accurate result.

Ans-5] a)

$$SNR = \frac{\mathcal{E}_s}{\mathcal{E}_n} = \frac{\sigma_s^2}{\sigma_n^2} = \frac{\frac{1}{n}\sum_{i,j}(I(i,j)-\bar{I})^2}{\rho_n^2}$$

σ_s^2 = variance of signal in an Image.

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

$$SNR[db] = 10\log_{10}\left[\frac{\mathcal{E}_s}{\mathcal{E}_n}\right]$$

b) Gaussian noise having a probability density function equal to normal distribution. Impulse noice is a certain pulse that can have random values.

A median filter handles better impulsive noise than average filter because difference of impulse and other cells in filter is too high then averge filter won't work it will reduce but couldn't remove the impulse when median filter will remove an impulse by replacing it with median value in the filter.

c) Image having the value of 2 in each cell. 3x3 convolution filter having all 1-s in it entries.

Suppose 4×4

Image

2	2	2	2
2	2	2	2
2	2	2	2
2	2	2	2

zero padding →

4×4

8	12	12	8
12	18	18	12
12	18	18	12
8	12	12	8

mirror replicate →

4×4

18	18	18	18
18	18	18	18
18	18	18	18
18	18	18	18

Filter

1	1	1
1	1	1
1	1	1

Ignore →

2×2

18	18
18	18

d) we have an image I which is convolved with a filter f

So, new image is $I_c = I * f$

we need to find derivative of I_c

$I_c' = I * f'$

If we want derivative of convolved image then first take derivative of filter then convolve it with image.

for more efficiency first apply any x or y part of filter then apply another part y or x.

e) i) Zero padding => take zero value outside the boundries. means when you put center of a filter on a cell and if you don't find cells relevent to the cells in a filter then take zero as a cell value in a image.

0	0	0	0	0
0	1	2	3	
0	4	5	6	
0	7	8	9	
0				

ii) Mirror / Replicate => when you put a filter on a cell and if you don't find cells relevent to the cell in filter then take that cell's value which replicate it in the mirror.

1	1	2	3
1	1	2	3
4	4	5	6
7	7	8	9

iii) Ignore => when you put a filter on a cell and if you don't find cells relevan to the cells in filter then ignore those cells. In this case dimensions of image will reduce

1	2	3	4
5	6	7	8
9	10	11	12

f) basic 3x3 smoothing filter

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

, The sum of all entries in this filter is 1.
Here all cells' value are 1/9. which means every cells have same weight in filter window Sum is 1 because every pixel is represented with 0-255. So when we convolve filter with a window in an Image. then convolved value will also be in 0-255. It won't overflow.

g) Here I is an Image and $G(x,y)$ is 2D a gaussian filter. when we apply 2D filter.

Complexity : MNm^2

$M \times N$ image dimensions and m is filter's dimensions

$I' = I * G(x,y)$

Implement 2D convolution with two 1D convolution Here $G(x)$ and $G(y)$ is a 1D filter. when we apply two 1D filter

Complexity : $2MNm = MNm + MNm$

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

first apply 1D Gaussian filter on rows of image then apply 1D Gaussian filter on cols of image and vise a versa

So, two 1D convolution filter is more efficient

Yes, it is possible to implement any 2D filter in this way if one can define the relative weights for row filter and col filter in a way that no information will be lost and added and also take care about out of range values

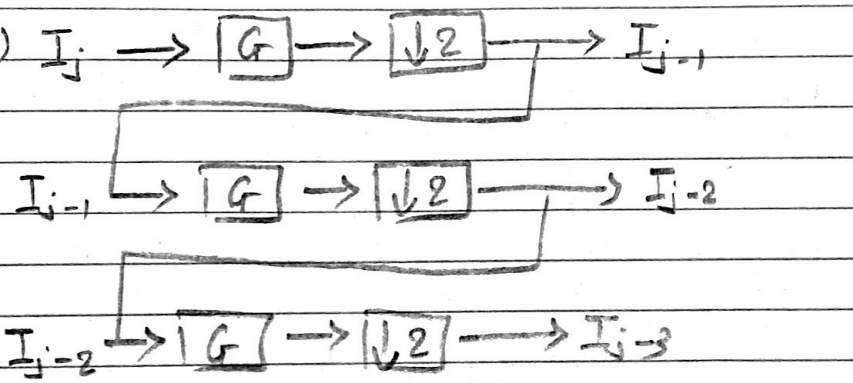
h) $\sigma = 2$

size of the filter, m should be greater than or equal to 5σ

$m \geq 5\sigma$

$m \geq 10$

size of the filter should be 10 or greater.

i) 

first take convolution of Input image with gaussian filter then take down sampling of 2 and get new image with reduced dimensions. repeat above step for every new image. This is how gaussian image pyramid produced.

Reason : when we want to find an object in an image we convolve an image with a filter. If we change size of an filter insted of image total complexity will be very high. but if we change size of an image insted of filter complexity in each step will reduce and total complexity will be less than above case.

Initial dimension of image is $m \times n$ and of filter is M.

complexity : mnM^2

next step complexity : $\frac{m}{2} \cdot \frac{n}{2} M^2 = \frac{mnM^2}{4}$

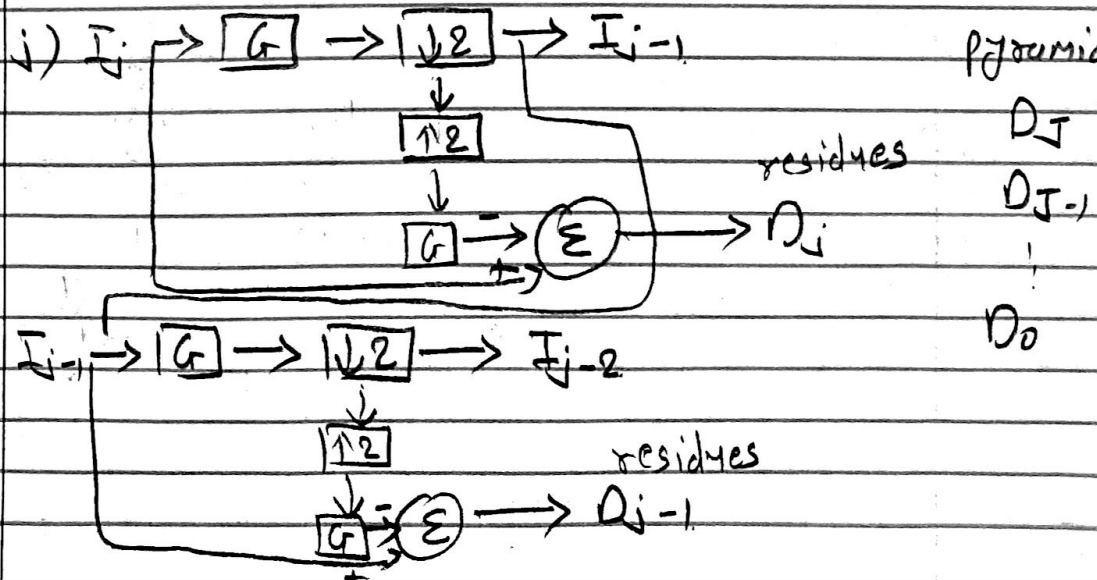
$\frac{m}{4} \cdot \frac{n}{4} \cdot M^2 = \frac{mnM^2}{16}$

$= \frac{mnM^2}{64}$

total no. of pixels processed :

$$mn + \frac{mn}{4} + \frac{mn}{16} + \ldots < \frac{4}{3} mn$$

The amount of additional processing done in a pyramid compared with a single image is $\frac{1}{3} mn$.



first take convolution of Input image with gaussian filter then take down sampling / Pooling of 2 and get new image. Then take up sampling and convolution with gaussian filter of last output and take difference of image and last output gives residues image.
repeat above step for every new image and get Laplacian pyramid of residue image.

Laplacian pyramid is used for analysis. like compression. we can know how much information is lost after compression of image using residual values.

Ans-6) a) Edge detection is useful to find the boundaries of objects within images

it can be found by detecting discontinuities in brightness.

desired properties of edge detection
- depth discontinuity
- normal discontinuity
- illumination discontinuity
- reflection discontinuity

b) - smoothing is required to reduce noise from the image without affecting edges.

- enhancement is required to enhance the quality of the edges in the image it is also called sharpening

- localization is required to find the exact location of an detected edge

c) Image gradient is a change in image from pixel to pixel in both x and y direction.

Image : $I(x,y)$

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

It can be found by taking partial derivative.

Image gradient is used to find pixel to pixel change in images from which we can detect edges in images

i) forward difference

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

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

$$\Delta_x = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \qquad \Delta_y = \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}$$

ii) central difference

$$\frac{\partial I}{\partial x} = \frac{I(x+h, y) - I(x-h, y)}{2h}$$

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

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

d) for smoothing we use 2x2 filter.

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

for derivative we use 2x2 forward difference filter.

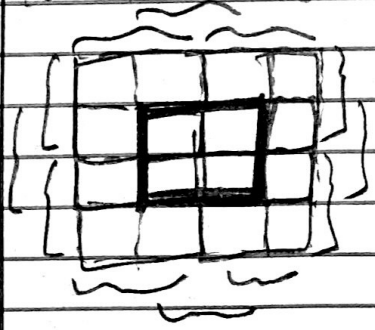
$$\Delta_x = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \qquad \Delta_y = \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}$$

Sobel filter is convolution of smoothing and forward difference gradient filter.

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

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

convolution of two 2x2 matrix



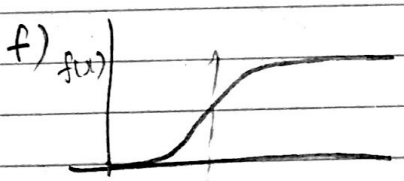
c) for more accurate derivative filter with an arbitary σ we can use gaussian derivative filter.

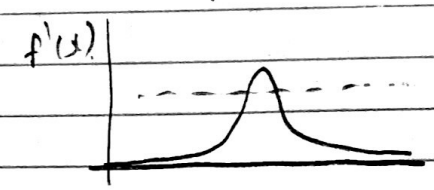
$$G(x) = e^{-x^2/2\sigma^2} \qquad G'(x) = \frac{-x}{\sigma^2} e^{-x^2/2\sigma^2}$$

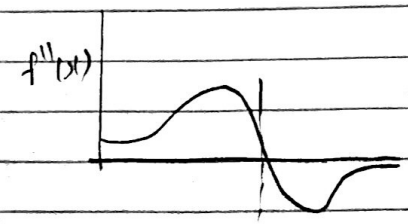
filter dimensions should be $m = 5\sigma \Rightarrow \sigma \leq \frac{m}{5}$

here $\sigma = 2 \Rightarrow m = 11$, filter dim should be odd

Index	Values	
0	+ 1.25 × 0.04	= +0.05
1	+ 1 × 0.14	= +0.14
2	+ 0.75 × 0.32	= +0.24
3	+ 0.5 × 0.61	= +0.305
4	+ 0.25 × 0.88	= +0.22
5	0 × 1	= 0
6	− 0.25 × 0.88	= −0.22
7	− 0.5 × 0.61	= −0.305
8	− 0.75 × 0.32	= −0.24
9	− 1 × 0.14	= −0.14
10	− 1.25 × 0.04	= −0.05

f) 





edge by threshold (first derivative)

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

magnitude $|\nabla I| = \sqrt{I_x^2 + I_y^2}$

$$E(i,j) = \begin{cases} 1 & \text{if } |\nabla I(i,j)| > Z \\ 0 & \text{o/w} \end{cases}$$

if magnitude is higher than threshold then there is an edge

edge by zero crossing
(second derivative)

$$\Delta I = \nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} = I_{xx} + I_{yy}$$

$$I_x = I(x+1) - I(x) \qquad \boxed{-1 \mid 1}$$

$$I_{xx} = I(x+1) - 2I(x) + I(x-1) \qquad \boxed{1 \mid -2 \mid 1}$$

when we take second derivative it will be zero at point which was maximum in first order derivative. So, we get zero crossing.

mark edges at transition.

$$E(i,j) = \begin{cases} 0 & \text{if } |\Delta I| < 0 \quad (0 \to 1) \\ 1 & \text{if } |\Delta I| \geq 0 \quad (1 \to 0) \end{cases}$$

a) $\sigma = 1$, $m = 5\sigma = 5$

$$G = e^{-r^2/2\sigma^2}, \quad r^2 = x^2 + y^2$$

$$\nabla^2 G = \frac{r^2 - 2\sigma^2}{\sigma^4} e^{-r^2/2\sigma^2} = G'' \Rightarrow G''_x = \frac{x^2 - \sigma^2}{\sigma^4} e^{-x^2/2\sigma^2}$$

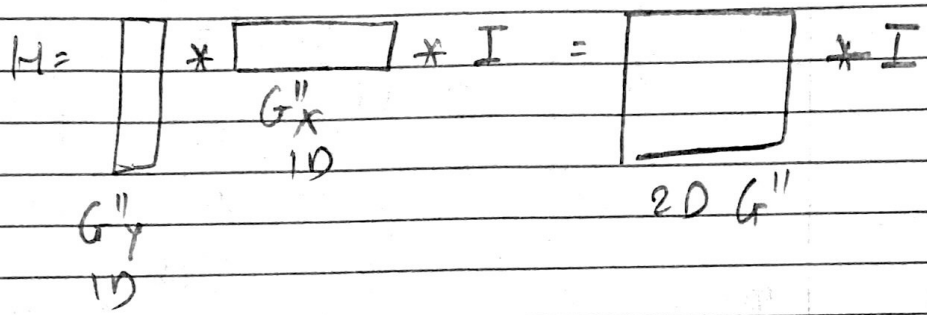
x	G''
0	$2 \times 0.14 = 0.28$
1	$-1 \times 0.61 = -0.61$
2	$-2 \times 1 = -2$
3	$-1 \times 0.61 = -0.61$
4	$2 \times 0.14 = 0.28$

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

$= \nabla^2 G * I$

$\nabla^2 G$ is a 2D filter

but we can apply two 1D filter



h) standard edge detection does not use a direction of gradient. It only finds gradient of every pixels and if magnitude is higher than threshold then it detects an edge at that pixel

Canny edge detection use a direction of gradient. It use two methods. one is to search local maximum of gradient magnitude. so when magnitude is max in local neighborhood then detect edges.
second is to when one cell is found with magnitude of gradient is larger then high threshold then search for additional neighbors in the orthogonal direction of gradient

difference :

standard edge detection gives broad edges.
canny edge detection gives thinner edges.

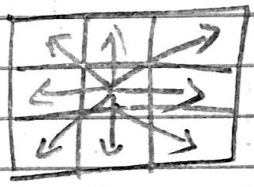
i) non-maximum suppression.

It detect edges when magnitude of gradient of cell is maximum in local neighborhood.

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

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

$$\theta^* = \tan^{-1}\left(\frac{\theta}{45}\right) \times 45$$



compare neighbor's gradient magnitude

$$E(i,j) = \begin{cases} 1 & \text{if } |\nabla(I * G)| \text{ is a local max} \\ 0 & \text{o/w} \end{cases}$$

Hysteresis thresholding

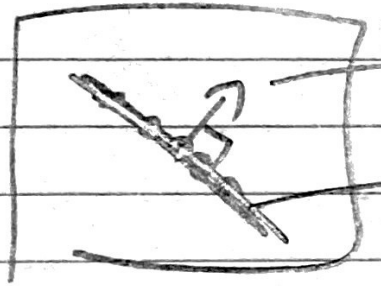
It use two threshold, T_H and T_L

$T_H > T_L$

It first marks every pixel unvisited.

Then scan image top-to-bottom, left-to-right.

If magnitude of gradient at current pixel is greater than Z_H then It starts tracking edge in orthogonal direction of gradient.



|gradient| > Z_H

search pixels on orthogonal direction those are possible on edge with threshold Z_L.

$|\nabla I| > Z_L$