ASSIGNMENT 1

A20442409
Akshay R.

Review Questions:

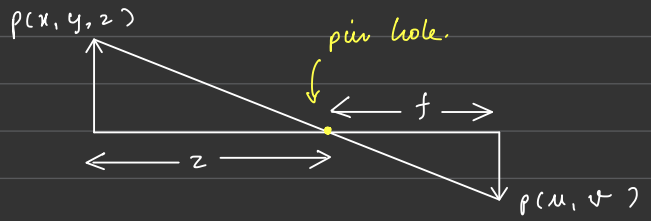
1. Geometric image formation.

(a) $f = 10 \qquad p = (3, 2, 1)$

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

$$= \frac{1}{1} \begin{bmatrix} -10 & 0 \\ 0 & -10 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \end{bmatrix} = \begin{bmatrix} -30 \\ -20 \end{bmatrix}$$

(b). **Model (1)**

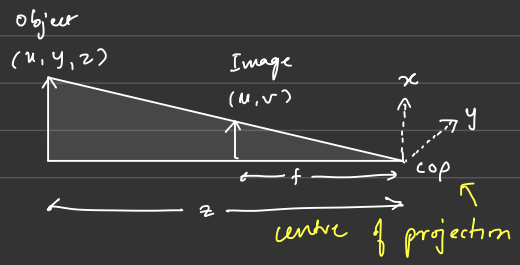


the transformed points (u, v) are given by

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

Model (2)

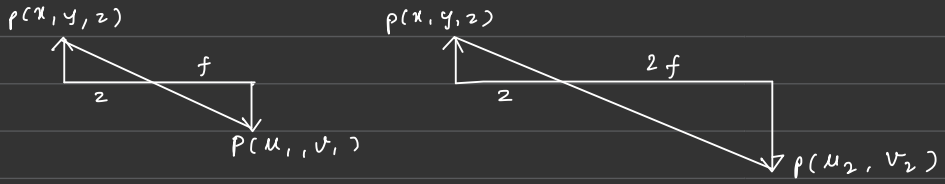
Here the point (u, v) is given by:



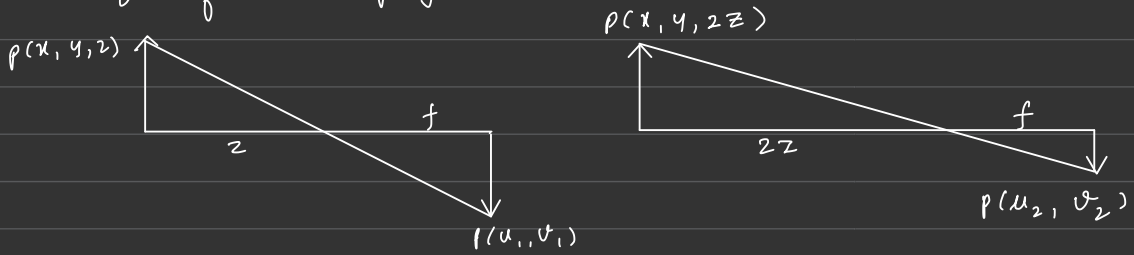
$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{z}\begin{bmatrix} f & 0 \\ 0 & f \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}$$

Model (1) corresponds better to the physical world. The other model, i.e, model (2) corresponds better for theoritical calculations. Omitting the -ve sign simplifies the calculation. However this is not a practical pin hole camera.

(c) When focal length increases the image size also increases.



When the distance to the object gets bigger the size of the projection decreases.



(d) Given point : $P(1, 1)$

2DH point : $(x, y, 1)$

$= (1, 1, 1)$

another 2DH point : (tx, ty, t)

for $t = 2$: $(2, 2, 2)$

(e) 2DH point : $(1, 1, 2)$

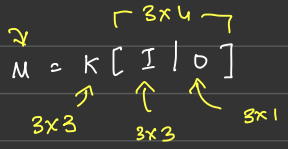
2D point : $(1/2, 1/2)$

(f) given 2DH point $(x, y, w) : (1, 1, 0)$

$w = 0$ represents direction of the line in the 2DH coordinate system that passes through the point (w, w, w) $\forall -\infty < 0 < \infty$ except for $w \neq 0$.

(g) By converting the image coordinates into 2DH coordinates we can omit the $1/z$ term while defining the points (u, v). Hence we can write a linear equation.

(h)



(i) $M = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 1 & 2 & 1 & 2 \end{bmatrix}$ $\quad p = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$

2DH $\qquad$ 3DH

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 1 & 2 & 1 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

p $\qquad$ p'

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

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

$$(u, v) = \left(\frac{18}{10}, \frac{46}{10}\right) = (1.8, 4.6)$$

2. Model transformations:

(a) $p(1, 1)$

$t = (2, 3)$

$$p' = \begin{bmatrix} I & t \\ 0 & 1 \end{bmatrix} p$$

(2DH) (2DH)

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

$$p' = \begin{bmatrix} 3 \\ 4 \\ 1 \end{bmatrix} \quad \therefore \quad p = (3, 4)$$

(b) ${}^{2DM}_{U}p = \left[\begin{array}{c|c} S & 0 \\ \hline 0 & 1 \end{array}\right] {}^{2DM}_{U}P = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix}$

$\therefore$ point $p : (2, 2)$

(c) $$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

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

after rotation we get the point $(0, \sqrt{2})$

(d) Rotation across an arbitrary point.

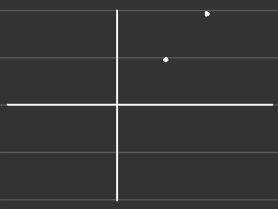
$R_{p,M}(\theta) = T(p)\, R_{M}(\theta)\, T(-p)$

point ↗ ↑ axis — translate back — rotate — translate

$$R_{(2,2)}(45°) = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 0 \\ 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 0 \\ 1/\sqrt{2} & 1/\sqrt{2} & -4/\sqrt{2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 2 \\ 1/\sqrt{2} & 1/\sqrt{2} & -4/\sqrt{2}+2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$



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

∴ The required point is $(2,\ 2-\sqrt{2})$.

(e) First rotate R then translate T
given transformation $p = Mp$.

projection matrix $\boxed{M = TR}$

(f) Given $M = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

transforming point p with M gives a scaled point
scaled by $(3,\ 2)$

(g) Given $M = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix}$

transforming point p with M gives a translated point
scaled by $(1,\ 2)$

(h) Given $M = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

The transformation matrix that will reverse the effects of this transformation is

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

(i) Given $M = R(\theta)\, T(x, y)$

$$M^{-1} = T^{-1}(x, y)\, R^{-1}(\theta)$$

$$= T(-x, -y)\, R(-\theta)$$

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

(j) A vector perpendicular to (a, b) is $(b, -a)$

$\Rightarrow$ a vector perpendicular to $(1, 3)$ is $(3, -1)$

(k) $\vec{a} = (1, 3)$ $\quad \vec{b} = (2, 5)$

$$\text{proj}_{\vec{b}} \vec{a} = \left(\frac{\vec{a} \cdot \vec{b}}{\|\vec{b}\|} \right)$$

$$= \frac{2 + 15}{\sqrt{29}}$$

$$\text{proj}_{\vec{b}} \vec{a} = \frac{17}{\sqrt{29}}$$

3. General camera model.

(a) We need different coordinate systems for camera, image and the world. This is because the camera may move and it would be inconsistant to compute all the points on the image using one coordinate system. Having different camera, world and image co ordinate systems ensures that we can efficiently translate image coordinates once the camera moves.

(b) $M_{c \leftarrow w}$ is first translation then rotation

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

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

rotation is orthogonal matrix so $R^T = R^{-1}$

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

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

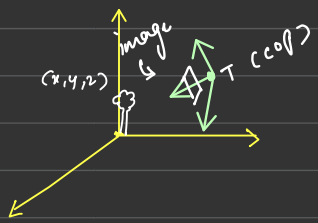
(R^T: 3×3, $-R^T T$: 3×1, 0: 1×3, 1: 1×1)

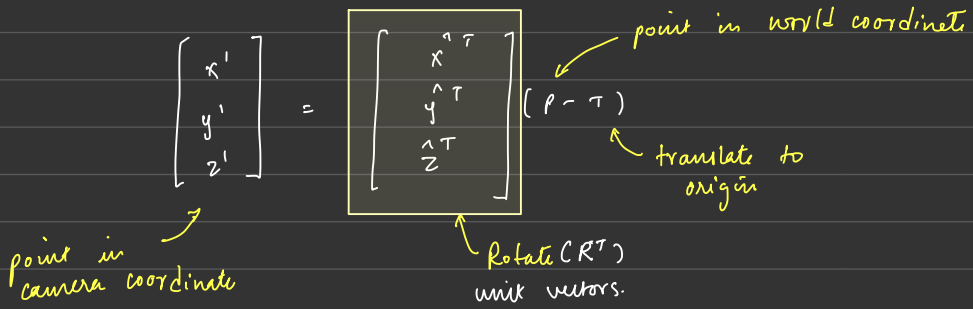
R^* → how much is the world rotated to obtain the allignment of camera

R → how much camera is rotated w.r.t the world.

$$\therefore \quad M_{c \leftarrow w} = \left[\begin{array}{c|c} R^* & T^* \\ \hline 0 & 1 \end{array}\right]$$

(c)
$$x' = (p - T) \cdot \hat{x} = \hat{x}^T (p - T)$$
$$y' = (p - T) \cdot \hat{y} = \hat{y}^T (p - T)$$
$$z' = (p - T) \cdot \hat{z} = \hat{z}^T (p - T)$$





(d) R^* → how much is the world rotated to obtain the allignment of camera

T^* → how much is the world translated to obtain the allignment of camera.

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

given $(u_0, v_0) = (512, 512)$

$$M_{i \leftarrow c} = \begin{bmatrix} k_u & 0 & 512 \\ 0 & k_v & 512 \\ 0 & 0 & 1 \end{bmatrix}$$

(f) $K^* [R^* \mid T^*]$

$$\underset{2DH}{p^{(i)}} = \underbrace{K^*}_{3\times3} \underbrace{[R^* \mid T^*]}_{3\times4} \underset{3DH}{p^{(w)}}$$

extrensic parameters

intrensic parameters

(g)

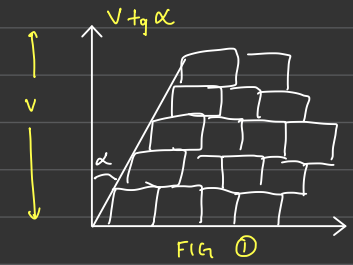


FIG ①

sheer in x

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

fifth independent parameter = 0

$$M_{int} = \begin{bmatrix} \alpha_u & & u_0 \\ & \alpha_v & v_0 \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & tg\,\alpha & \\ & 1 & \\ & & 1 \end{bmatrix} = \begin{bmatrix} \alpha_u & \alpha_u\, tg\,\alpha & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

Skew parameter is an internal parameter of the camera that exists. Fig ① shows an exaggerated setup of the camera pixels. This can be translated into the projection matrix by adding the term $tg\alpha$.

(h) Occurs with a camera with a large field of view

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

λ depends on what point is observed in the image

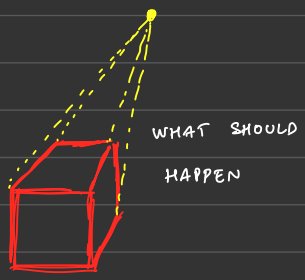
linear distortion coefficient

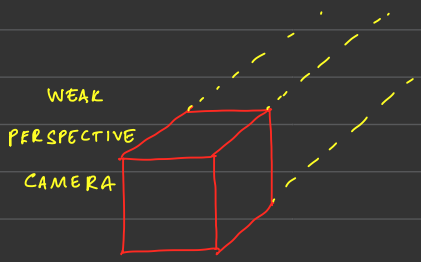
$$\lambda = 1 + k_1 d + k_2 d^2$$

distance from center — quadratic distortion coefficient

once k_1 and k_2 are determined the image can be warped to correct the distortion.

(i)





$$M_{\infty} = \begin{bmatrix} & & & \\ & & & \\ 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow (1)$$

Weak perspective cameras are those that lack the capacity to capture perception or depth of the image as shown in the image above. The projection matrix of a weak perspective camera looks like as shown in (1).

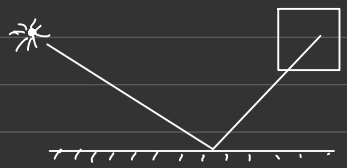
An affine camera is a special case of the projective camera and the transformation matrix looks like

$$\underset{3\times 4}{M_{affine}} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4. Color and photometric image formation

(a) Relate light in the scene (surface radiance) to light in the image (image irradiance)

$L(p)$ → power of light per unit area reflected from the surface (surface radiance)



$E(p)$ → power of light per unit area received to the image (image irradiance)

(b) The required equation is

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

- $E(p)$: light at the image
- $L(p)$: light at the surface
- d: diameter of the lens
- f: focal length
- α: angle between principal axis and surface normal

$d \uparrow \Rightarrow E(p) \uparrow$

$f \uparrow \Rightarrow E(p) \downarrow$

$\alpha \uparrow \Rightarrow E(p) \downarrow$

(c)

Lambard's cosine law states:

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

I_{ref}: intensity of reflection

I: Intensity of source

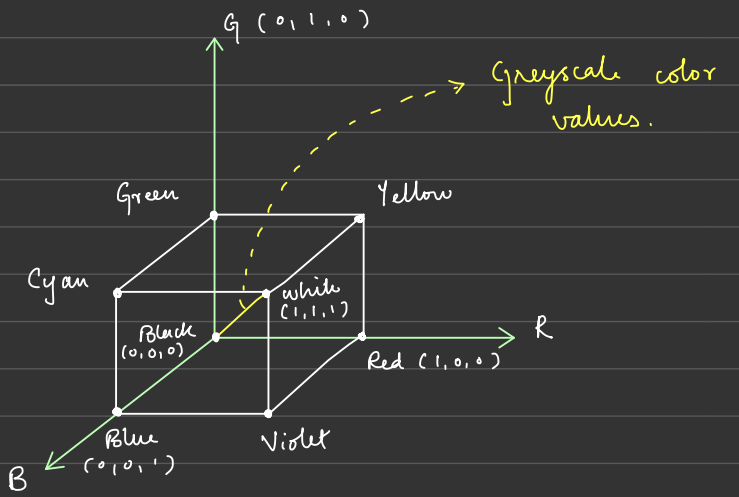
ρ: Surface albedo [0, 1]
= 0 poor reflector
= 1 good reflector

} reflection coefficient.

Albedo of a surface is a constant with respect to a particular surface. It ranges from (0 to 1) 0 meaning a poor reflector and 1 meaning a good reflector.

(d) We use RGB model to represent colors due to the reason that human eye uses RGB to perceive colors.

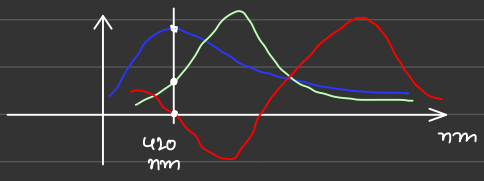
(e)

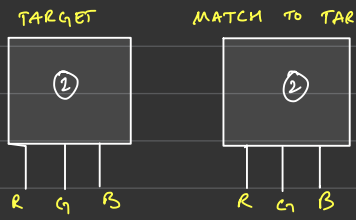


(f) RGB colors are mapped to real world colors by using CIE tables

* wavelength of RGB intensities are mapped to get the color combination.

645 nm (R)	526 nm (G)	444 nm (B)	color





* Several volunteers mapped the given RGB knobs ① and ② to obtain desired color

(g) In the YIQ model the Y gives the luminance. Early days when broadcast signal to analouge TVs were YIQ, a TV that could perceive only greyscale images used Y → luminance to display.

(h) LAB → means distance in perception space.

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

color comparison becomes easier. ⇓ C_1 is more similar to R

5. Noise and filtering:

(a) Signal to noise ratio:

* Measures the noise

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

σ_n^2 → variance for multiple frames of a static scene

(OR)

variance in a uniform image region.

$$SNR\ [db] = 10 \log_{10} \frac{E_s}{E_N}$$

10 db → E_s is 10 times larger than E_n

20 db → E_s is 20 times larger than E_n

3 db → E_s is 2 times larger than E_n.

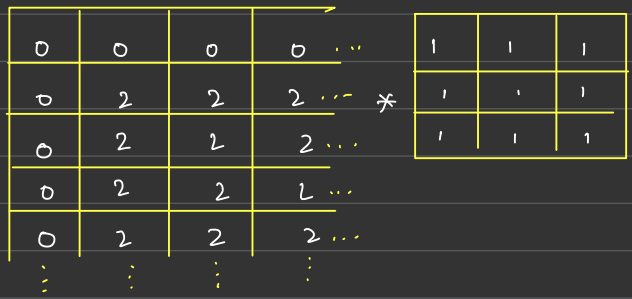
(b) Gaussian noise is a statistical noise present in images. This noise follow a gaussian distribution and is difined by this mathematical formula:

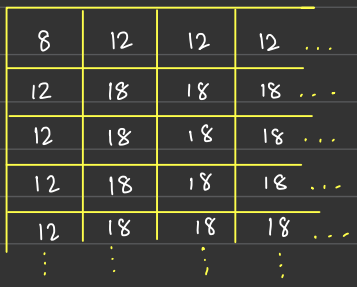
2D gaussian: $G_\sigma[x, y] = e^{-\frac{x^2 + y^2}{2\sigma^2}}$

Impulsive noise also known as salt and pepper noise are sudden spikes that occur in images. These noises are random and are best removed by using a median filter

(c) This can be analised in two scenarios

* with zero padding





* Without zero padding: Every cell will have the value of 18.

(d) We need $\frac{\partial}{\partial x}(I * F)$ (image: I, filter: F)

an effective way of computing this would be

$$\frac{\partial}{\partial x} I * F \quad \text{or} \quad I * \frac{\partial}{\partial x} F$$

(e) Padding : * zero padding
* Mirror replicate
* Ignore.

(f) A basic 3x3 Smoothing filter is as shown below:

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

Sum of all entries = 1

The sum = 1 since this filter should not increase or reduce the brightness but just perform Smoothing effect. Sum > 1 results in sharpening the image and sum < 1 results in reducing the intensity of the image.

(g)

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

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

$$= (I * G_y) * G_x \quad \text{(1D, 1D)} = I * G_x * G_y$$

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

2 1D pass: $2MNm$ operations $<$ 1 2D pass: MNm^2 operations

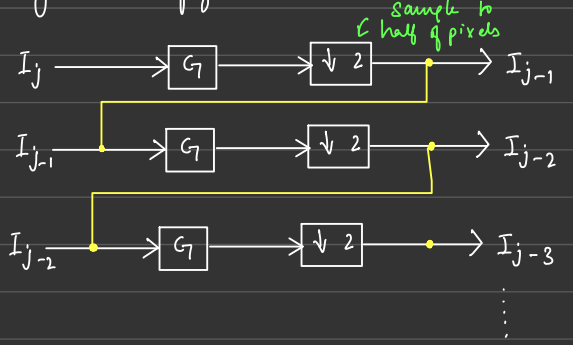
Not all filters are seperable. Gaussian filters are seperable as shown above.

(h) $\sigma = 2 \Rightarrow m \geq 5\sigma$

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

(i) Gaussian pyramids:

convolution helps get rid of aliasing



Pyramid

I_j

I_{j-1}

⋮

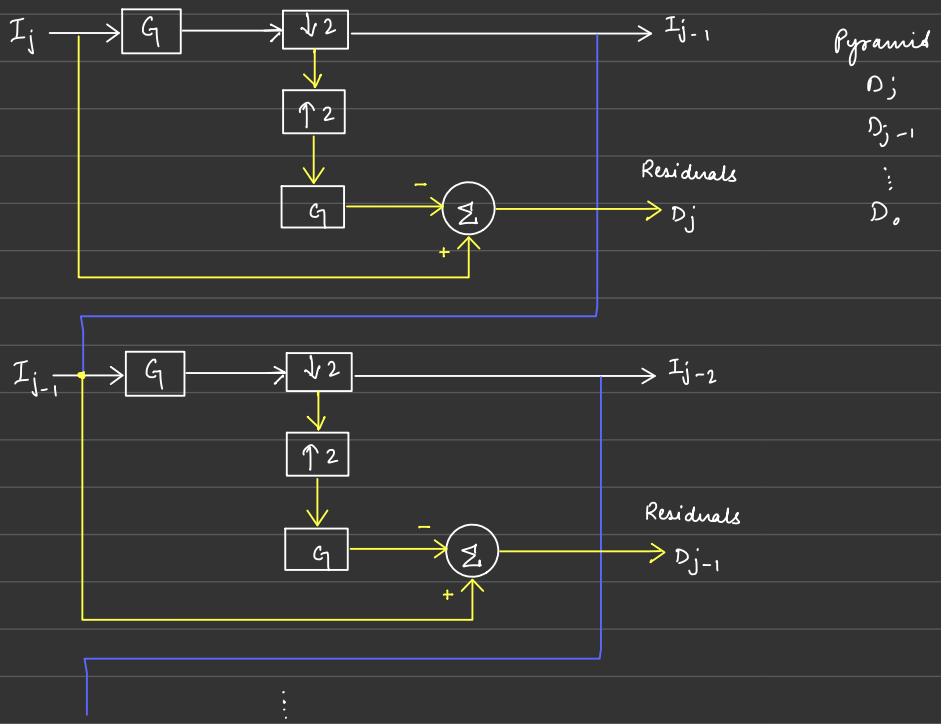
I_0

* useful for analysis

The above diagram shows how gaussian pyramids are created. Normal image pyramids consume $\frac{4}{3}m^2$ number of pixels where m is the size of filter. there an additional 33% processing is required when compared to a single image.

(j) We compute residuals in addition to gaussian pyramids to realize how downsampling effects the information retained. Below picture shows the structural skeleton of a laplacian pyramid.

Laplacian pyramids:



6. Edge detection:

(a) * Edges provides good description of the image.
* We need (desired properties)
- correspond to a scene element
- invariant
- reliable detection.

(b) Smoothing: Smoothing is performed on images to remove noise. This process has to be carefully applied such that we do not loose the edges

Enhancement: This is normally a derivative. We need to amplify the edges for easy edge detection.

Localization: We need to know exactly where the edges occur in the image.

(c)

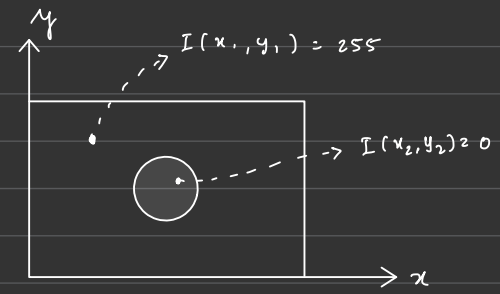
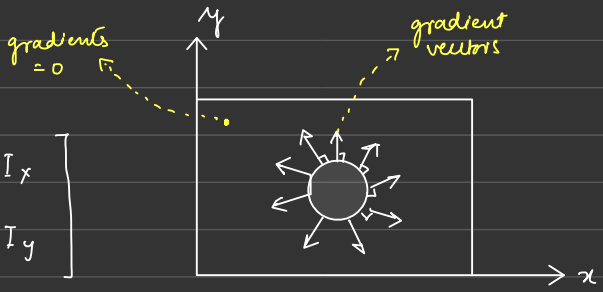


Image — 'intensity' $I(x, y)$

Image gradient

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



* Gradients provide information about change in intensity.
* Gradients are used to detect edges

any change in edges the gradient $\neq 0$.

(d) Sobel filter:

smoothing filter ↓ ; x derivative filter ↓

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

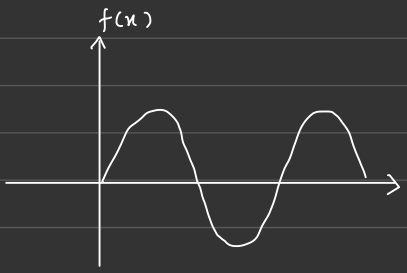
Sobel$_x$ filter ↓

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

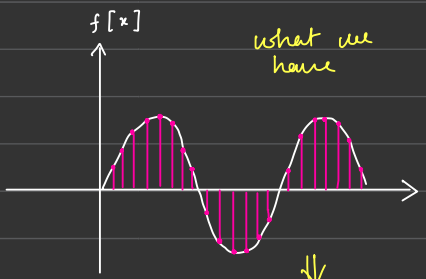
↑ smoothing filter ; ↑ y derivative filter

Sobel$_y$ filter ↑

(e)

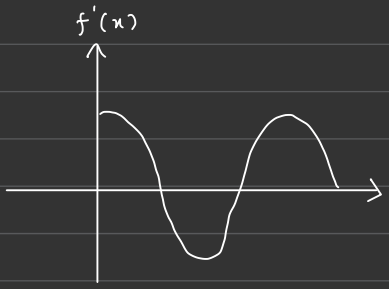


Step 1: $f[x] * h(x)$ (perfect interpolator)

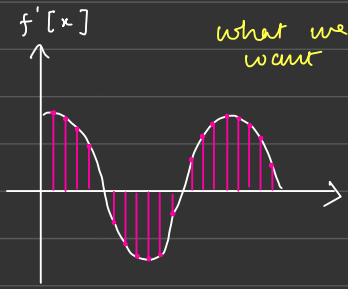


Step 2: Convolve $f[x] * h'(x)$

convolve with derivative of convolution filter $h'[x]$



Step 3: Sample $f[x] * h'[x]$



Since it is not practical to use a perfect interpolator we use a gaussian function instead.

$$I_x = I * G_x$$
$$I_y = I * G_y$$

Using seperable property of gaussian:

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

↙ 1D convolution with horizontal gaussian derivative

↙ 1D convolution with vertical gaussian

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

↖ 1D convolution with horizontal gaussian

↖ 1D convolution with vertical gaussian derivative.

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

Image * derivative * smooth

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

Image * derivative * smooth

$$G(x) = e^{-\frac{x^2}{2\sigma^2}} \rightarrow \text{Sample} \rightarrow \text{1D gaussian filter}$$

$$G'(x) = -\frac{2x}{2\sigma^2} e^{-\frac{x^2}{2\sigma^2}} \rightarrow \text{Sample} \rightarrow \text{1D gaussian derivative filter.}$$

$$-\frac{x}{4} e^{-\frac{x^2}{8}}$$

given $\sigma = 2$.

$\therefore$ Size of the filter should be > 10.

$G(0) = 1$

$G(1) = e^{-1/8} = G(-1)$

$G(2) = e^{-4/8} = G(-2)$

$G(3) = e^{-9/8} = G(-3)$

$G(4) = e^{-16/8} = G(-4)$

$G(5) = e^{-25/8} = G(-5)$

$G'(0) = 0$

$G'(1) = -0.25\, e^{-1/8} = -G'(1)$

$G'(2) = -0.5\, e^{-4/8} = -G'(2)$

$G'(3) = -0.75\, e^{-9/8} = -G'(3)$

$G'(4) = -e^{-16/8} = -G'(4)$

$G'(5) = -1.25\, e^{-25/8} = -G'(5)$

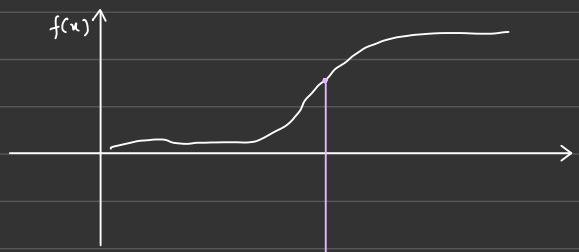
$G(x)$ ↓

①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩	⑪
$e^{-25/8}$	$e^{-16/8}$	$e^{-9/8}$	$e^{-4/8}$	$e^{-1/8}$	1	$e^{-1/8}$	$e^{-4/8}$	$e^{-9/8}$	$e^{-16/8}$	$e^{-25/8}$

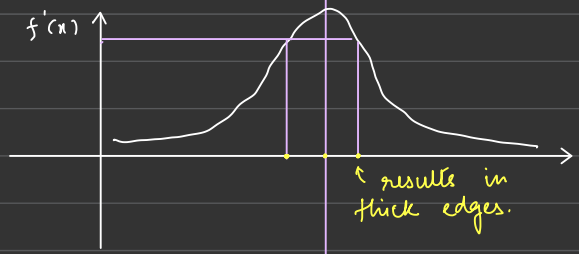
$G'(x)$ ↓

①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩	⑪
$1.25\, e^{-25/8}$	$e^{-16/8}$	$0.75\, e^{-9/8}$	$0.5\, e^{-4/8}$	$0.25\, e^{-1/8}$	0	$-0.25\, e^{-1/8}$	$-0.5\, e^{-4/8}$	$-0.75\, e^{-9/8}$	$-e^{-2}$	$-1.25\, e^{-25/8}$

(f).

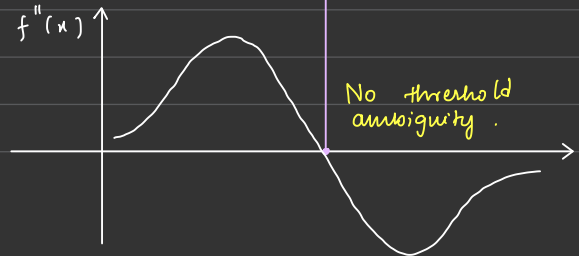


Edge.



Edge by threshold

↑ results in thick edges.



No threshold ambiguity.

Edge by zero crossing.

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

First order derivative → $I_x = I(x+1) - I(x)$ ← Describe kernel

Second order derivative → $I_{xx} = (I(x+1) - I(x)) - (I(x) - I(x-1))$

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

I_{xx}

1	-2	1

$I_{yy} =$

1
-2
1

$I_{xx} + I_{yy} =$

0	1	0
1	-4	1
0	1	0

(g) we have

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

given $\sigma = 1$

$$\nabla^2 G = (r^2 - 2) e^{-\frac{r^2}{2}} \qquad r^2 = x^2 + y^2.$$

$x = 0 \quad y = 0 \quad r^2 = 0 \quad \nabla^2 G_{(0,0)} = -2$

$x = 1 \quad y = 0 \quad r^2 = 1 \quad \nabla^2 G_{(1,0)} = -e^{-1/2} = \nabla^2 G_{(0,1)} = \nabla^2 G_{(}$

$x = 1 \quad y = 1 \quad r^2 = 2 \quad \nabla^2 G_{(1,1)} = 0 = \nabla^2 G_{(-1,-1)}$

$$\nabla^2 G = \begin{bmatrix} 0 & -0.6 & 0 \\ -0.6 & -2 & -0.6 \\ 0 & -0.6 & 0 \end{bmatrix}$$

Finding Edges:

1. Compute LOG : $H = (\nabla^2 G) * I$ (LOG)
2. Thresholding : $E(i,j) = \begin{cases} 0 & \text{if } H(i,j) < 0 \\ 1 & \text{if } H(i,j) \geq 0 \end{cases}$
3. Mark images at transitions $0 \rightarrow 1$, $1 \rightarrow 0$

Left → Right Top → bottom.

(h) Canny edge detection algorithm uses the directional derivatives of gradients instead of directly looking at the gradients for edge detection.

The condition for detecting an edge is to look for gradients > threshold. Only if the condition satisfys continue to find the directional derivative of this gradient else move on to the next gradient.

(i) Non maximum supression:

To determine the direction of tracking, we need to find the local maximum of gradient magnitude. We compute this by the following steps:

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

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

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

this yields values of θ^* as 0, 45, 90, 135 ....

Compute edge

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

Hysteresis thresholding:

* Use τ_H to start tracking and τ_L to continue ($\tau_H > \tau_L$)
  1. Initialize array of visited pixels to $V(i,j) = 0$
  2. Scan array top to bottom, left to right
     a. if $!V(i,j)$ && $|\nabla I| > \tau_H$: start tracking the edge
     b. Search for additional neighbours in direction orthogonal to ∇I such that $|\nabla I| > \tau_L$