* Image :-

Image is a 2D function $f(x, y)$, where (x, y) are spatial co-ordinates and the value of f at any point (x, y) is proportional to the brightness of the image

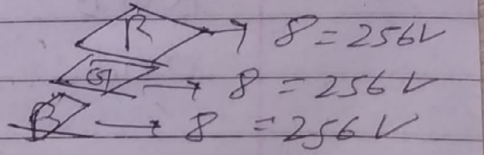
1 bit = 2 values (0 and 1)
8 bit = 2^8 values = 256

* Digital image :-

Image having discrete values

* Color depth :-

To store 256 values of R, G and B we need 8 bits

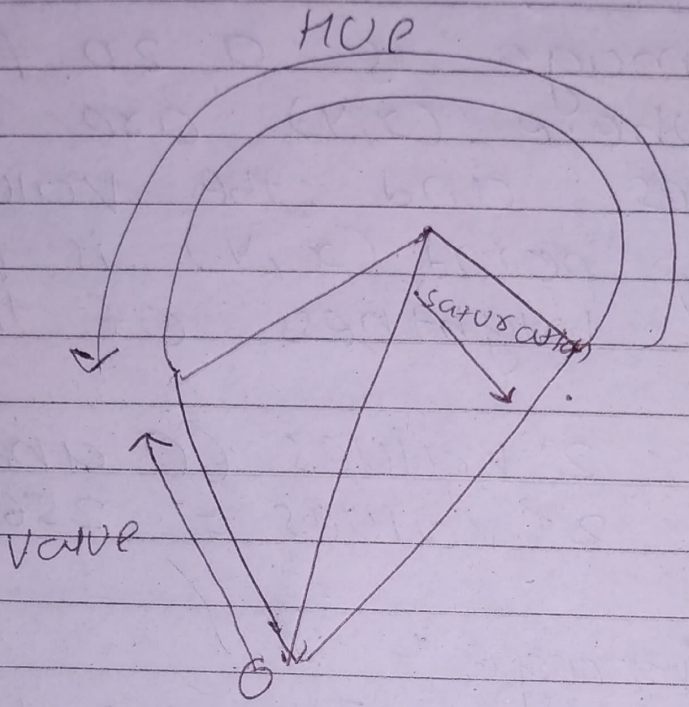


$3 \times 8 = 24$ bits

No. of bits used by a pixel

* Grayscale image :- Gray values :- Gray levels = brightness = Intensity
:- It has single gray (0, 255)

* HSV Color Space



Hue :- represents color
saturation = depth of color
value :- brightness of color

* Drawbacks of RGB

:- RGB holds the value of color and brightness together

suppose we are using 2 images first image is light and the 2nd is darker one so it will detect different colors

LAB color space

L = Intensity
a = color components ranging from Green to Magenta

b = color component ranging from Blue to yellow

* Transforms

1] Euclidian transform or Isometric

2] Affine transform

3] Projective transform

Transform :-
Modify spatial relationship between pixels. Image can be moved, rotate, can be stretched

* Euclidian transform

:- Can be moved in x, y axis and also rotate irrespective of any pixels.
Have 3 degree of freedom.

Characteristics :-
Distance constant
angle constant
Shapes constant

Normalize = Divide

Gaussian blur > blur

Filter = Kernel

If there is no normalizing then the image will get white.

Low pass filter :- Smoothing of an image

cv.filter2D(src, ddepth, kernel, anchor, border-type)

Note :- Kernal always replaces the center value but incase if we want to place value near to the center then the anchor is used

* box filter 2D :- ~~Fo~~ To increase the intensity of image.

* Gausian blur :- ~~Weight~~ weight is more near to center and less far away from the center

* Median blur :- Reduction in noise
In beauty filter we can use median blur

Gausian blur :- Have contrast values

* bilateral :- preserves edges while blurring

~~(d) parameter :-~~

bilateral function makes sure that only those targeted pixels having intensity almost same as target pixel are considered

~~(d) :- At how~~

(d) = At how much distance we have to considered the values

Sigma color :- deviation of color

increase in Sigma color
= = = = increase in mixing of colors.

~~Sigma space~~

Sharpening :-

Take gausian blur

Subtract gausian blur from original image make sure to maintain 1

gamma = intensity

* Image :- Described as a 2D function at $f(x,y)$ where (x,y) are spatial co-ordinates and the value of f at any point (x,y) is proportional to the brightness or gray levels of the image.

1 bit = 2 values
8 bit = 2^8 values =
(1 byte)

* Difference between digital image and Image

Digital image :- Takes discrete values at any pt on the function.

* RGB model

i] Additive model
ii] Each pixel of each channel value ranges from 0 to 255

Total depth = 8^3 = 24 bits

* Grayscale image → consist of single array

(1) 1 channel because we dont require any R, G, B channels to represent this Grayscale image.

If 1000 = h
600 = w

Total size = 1000 × 600 × 1 (single channel)
= 6 lakh bytes

* RGB image

Total size = 1000 × 600 × 3 (channels)
= 18 lakh bytes

* Color spaces

Representation of colors →

i] CMYK [cyan magenta yellow key]

1] subtractive color model

2] K = key = black color
(a) It is used as a subtractive binary
(b) It is used because of cost effectiveness
(c) Instead of using 3 colors to make black ink.

* Half Toning

: process of painting image using less ink

* HSV color space

: Hue saturation value

Hue = color
Saturation = amount of gray
value = brightness

* Nompy

check version np.__version__

* Interpolation

: Linear, Area, cubic, nearest, sinosoi

Transform :- modify spatial relation -ship b/w pixels

(1) Isometric or Euclidian transform
3 Degrees of freedom
x, y, Rotate

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

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

Distance remains same
Angles remains same
shapes remains same

(2) Affine transform

parallel lines may be sheared
Squares may become parallelogram

* Convolution :-

Basically applying mathematical operation to each pixel

Kernel = Filter

Low pass filter :- Attenuates high frequency component → smoothening

High pass filter :- Attenuates low frequency component

* Box filter :- Increases the intensity

* Gausian & any other averaging filter

* Median blur :- Reduction of noise

* Bilateral :- Blurring + preserving edge

* Edge detection → intensity

The process of edge detection involves detecting sharp edges in the image and producing a binary image.

*Sobel - x

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

* ~~Sobel~~ Sobel - y

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

* Sharr - x

Gradient is always ⊥ to edge direction

$$|G| = \sqrt{G_x^2 + G_y^2}$$

$$\theta = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

* Canny edge detection

(1) Noise Reduction → If there is noise in image it will affect the edge detection. as it is a change in high and low intensity

(2) Gradient calculation →

(3) Non-maximum suppression

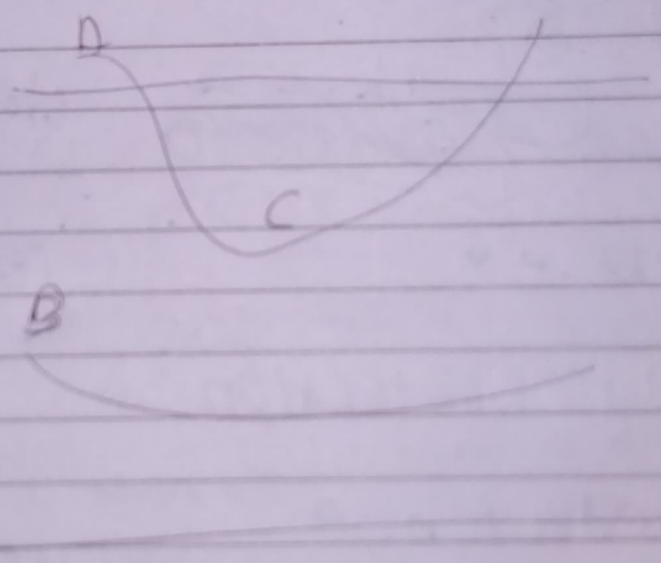
(4) Double thresholding and edge tracking by Hysteresis → those edges having intensity gradient more than maximal are edges

The algorithm goes through all the points on the image and finds the pixel with the maximum value of gradient in the edge direction

(5) Hysterisis :-

We have removed the edges lying above maxval and minval but what about the edges lies b/w maxval and minval

To deal with this ~~the~~ hysterisis is used



If they are connected to sure edge pixels they are considered to be part of edges. otherwise they are also discarded

↗ interior ↗ external

* Intrinsic and Extrinsic parameters

Extrinsic parameter :- 3D where is the camera in the 3-D world

θ = $\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix}$ → 1° → 3D orientation $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$ → where is camera

3D orientation where camera is looking to

$$\begin{bmatrix} x \\ y \\ z \\ \alpha \\ \beta \\ \gamma \end{bmatrix} = 6D \text{ vector}$$

* Intrinsic :- How to map a 3D world point on to the 2D image

parameters :-

Camera constant, x-y scale difference
focal length in x and y
principal point :- The pixel in image of which optical axis passes

Projection matrix = $M_{int} \; M_{ext}$

$$P = \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \end{bmatrix} = \begin{bmatrix} f_x & 0 & 0_x & 0 \\ 0 & f_y & 0_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{32} & P_{33} \end{bmatrix} = \begin{bmatrix} f_x & 0 & 0_x \\ 0 & f_y & 0_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

P = KR

K → Intrinsic

R → Rotation

K is an upper triangular matrix

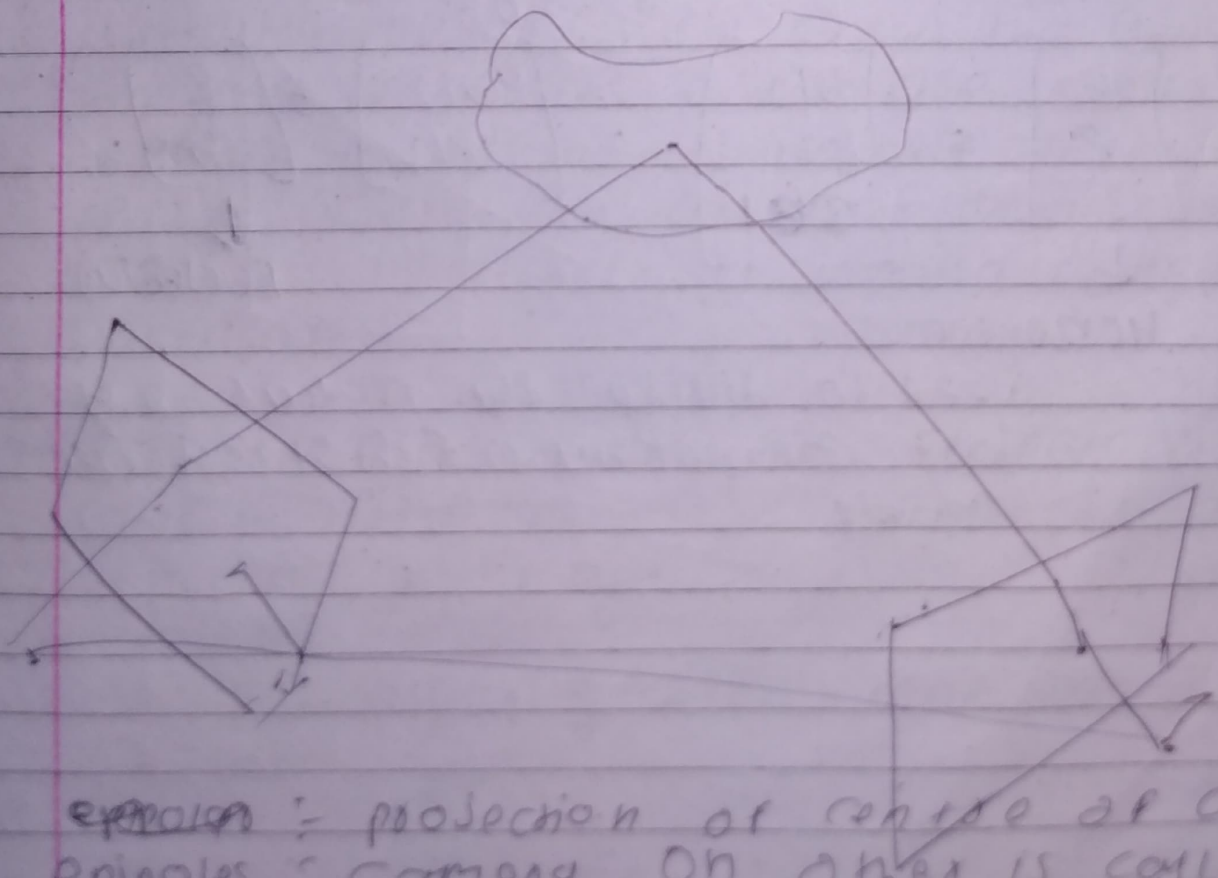
$$\begin{bmatrix} p_{14} \\ p_{24} \\ p_{34} \end{bmatrix} = \begin{bmatrix} fx & 0 & Ox \\ 0 & fy & Oy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} tx \\ ty \\ tz \end{bmatrix} = Kt$$

$$t = K^{-1} \begin{bmatrix} p_{14} \\ p_{24} \\ p_{34} \end{bmatrix}$$

translational matrix

* Epipolar

Relative position of camera to other camera



epipolar : projection of centre of one
epipoles : camera on other is called
epipoles

Homogeneous co-ordinates

co-ordinates for projective geometry
It helps in mapping point from 3d plane to 2d image

① They can be expressed in various matrix

② point at infinity

$$\begin{bmatrix} 2 \\ 4 \end{bmatrix} \longrightarrow \begin{bmatrix} 2 \\ 4 \\ 1 \end{bmatrix} \rightarrow 3D$$

Euclidian → Homogeneous

$$\begin{bmatrix} 4 \\ 8 \\ 2 \end{bmatrix} = \begin{bmatrix} 4/2 \\ 8/2 \\ 2/2 \end{bmatrix} = \begin{bmatrix} 4/2 \\ 8/2 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$

Homogeneous → Euclidian

Need to divide by matrix by last component of homogeneous Matrix

[illegible]

• [illegible] dimension

Disparity

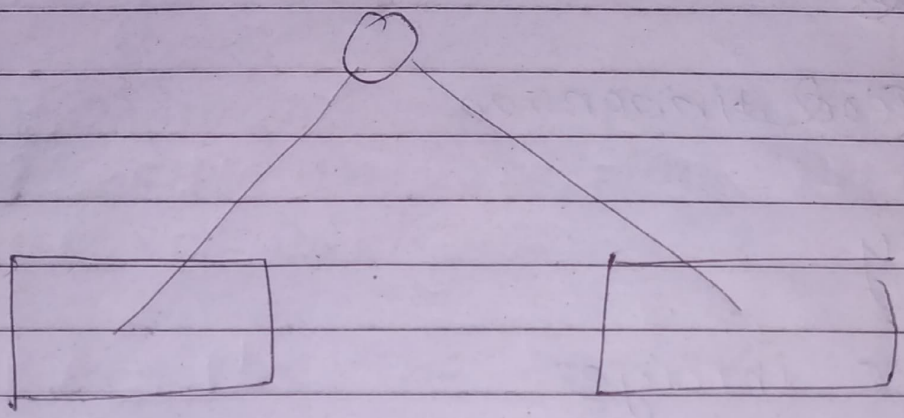
2 stereo images

Lighted shade = object in front
Disparity map refers to the apparent pixel difference or motion between a pair of stereo images.

~~If one eye is open and other is~~ Closing and opening ~~both~~ each eye of simultaneously which shows the change in distance : objects which are closer will appear to jump a significant distance and objects which are far away will move a little bit. It is also called as parallax

Difference in location of objects in corresponding two images as seen by the left and right eye

Using disparity, depth map can be obtained



$$Z \propto \frac{1}{(x_2 - x_1)}$$

Disparity map is basically a pixel difference between a pair of stereo images

~~Baseline~~
~~Disparity~~

Fundamental + essential

✗ ~~Fundamental~~ Matrix

= To display relation between two stereo images
fundamental matrix are used for uncalibrated camera
essential matrix are used for calibrated camera.

Point in img ① ← $x' F x'' = 0$ → point in img ②

→ 8 point algorithm

F coplanarity constraint

$x' E x'' = 0$ → 5 point algorithm

✗ Fundamental matrix

M_1, M_2 = matrix of intrinsic parameters of left and right camera

$\bar{P}_1, \bar{P}_2$ = pixel coordinates of P_1 and P_2

$\bar{P}_1 = M_1 P_1, \quad \bar{P}_2 = M_2 P_2$

$P_2^T E P_1 = 0$

$(M_2^{-1} \bar{P}_2) E (M_1^{-1} \bar{P}_1) = 0$

$P_2^{-T} F \bar{P} = 0$

$F = (M_2^{-1})^T E M_1^{-1}$

= ~~$(M_2^{-1})^T E M_1^{-1}$~~

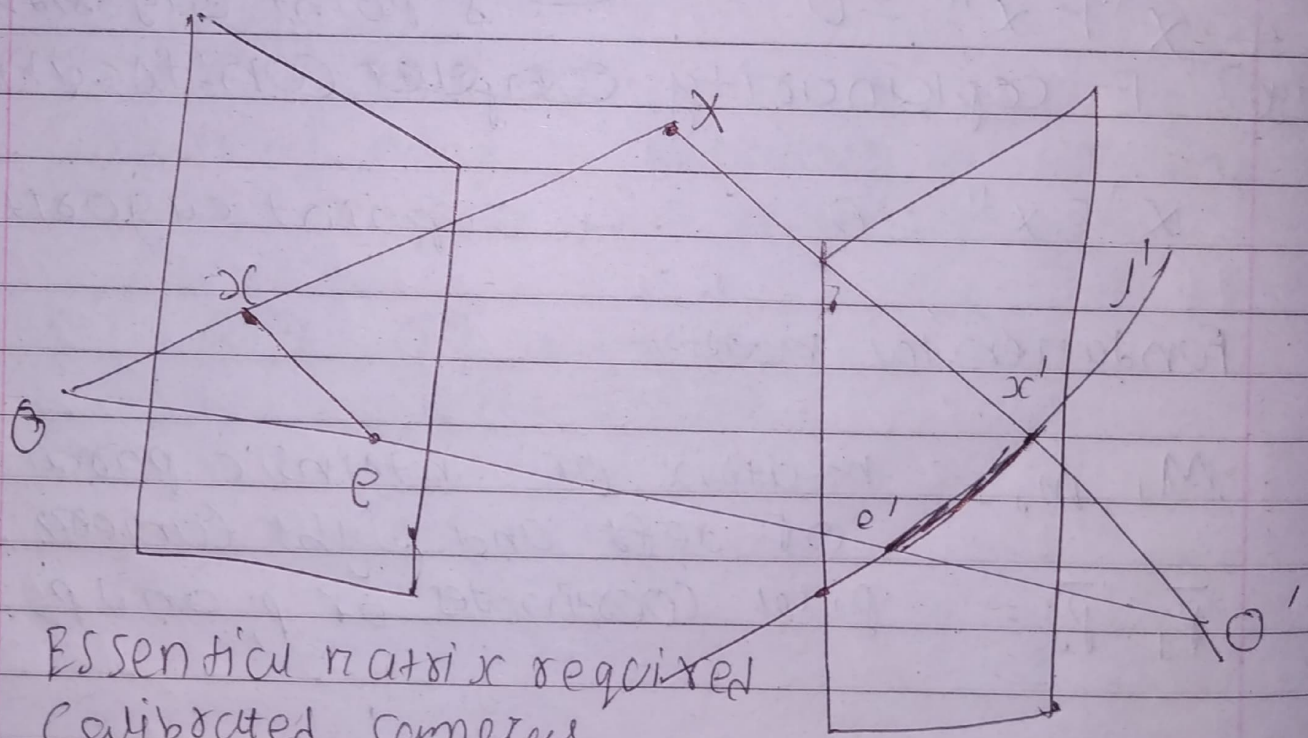
Has both extrinsic and intrinsic parameter

Rank 2

Essential matrix

Essential matrix = 3 x 3 Matrix consist of epipolar geometry

$EX = \lambda'$



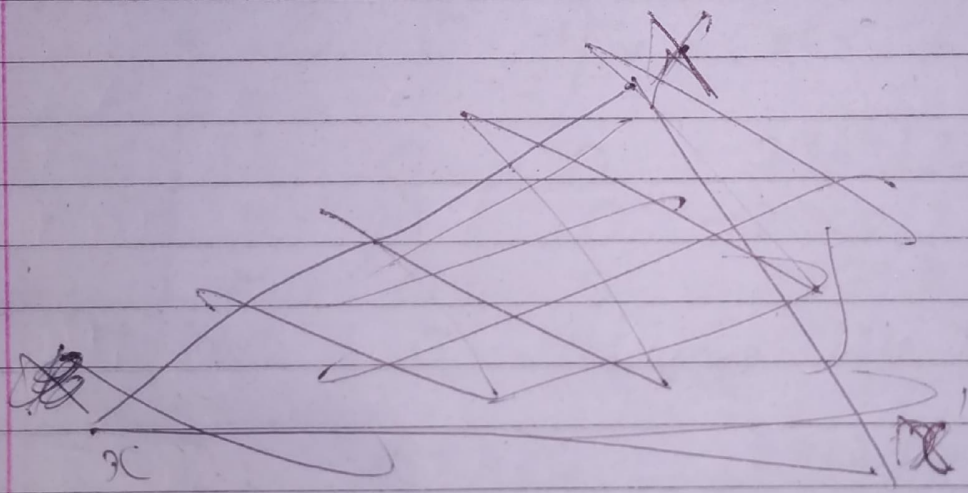
Essential matrix required Calibrated cameras because inner cameras parameter cannot be variable

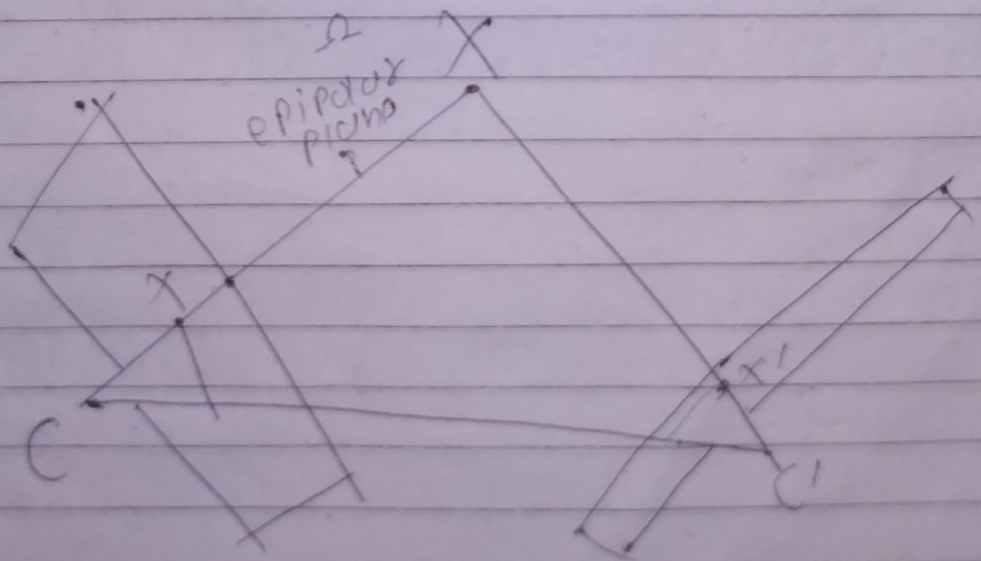
Q2 * Epipolar Geometry and fundamental matrix

* It is independent of scene, structures

* It depends upon camera internal parameters

* Fundamental matrix is a 3×3 matrix of rank 2





STEREO RECONSTRUCTION

* Scaling, Translation, Rotation

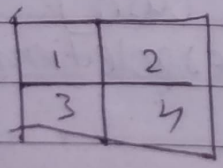
* Optical zoom :- moving the lens so that it increases the magnification of light.

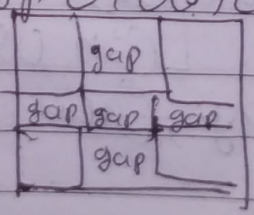
* Digital zoom :- It is simply interpolation the image after it has been acquired

* Scaling refers to change in height and width.

Increasing and decreasing the pixels in an image

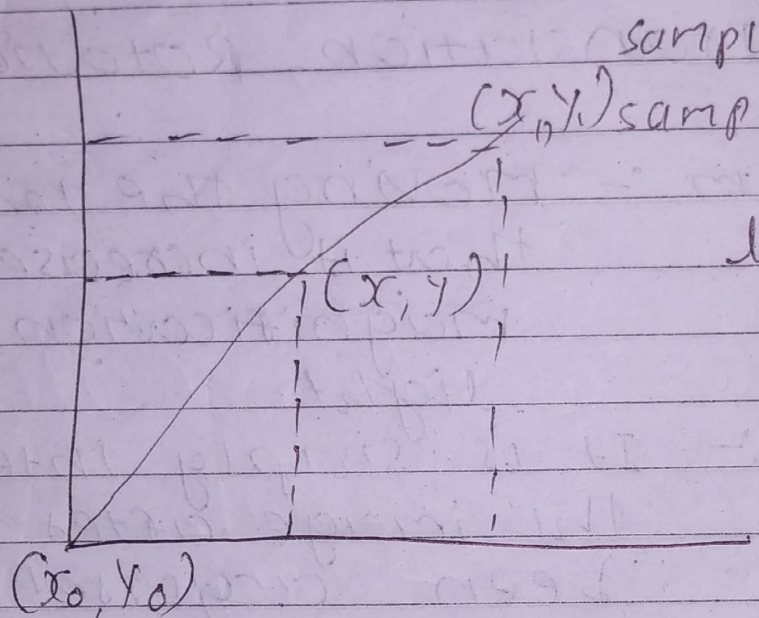
When we increased the size of the image the gap between the pixels increase to fill those gaps ~~the~~ ~~no~~ we need interpolation





Interpolation is the process of estimating the values of a continuous function from discrete samples

image is a function



(i) Linear Interpolation
(ii) Area -11-
(iii) Cubic -11-
(iv) Nearest neighbour -11-
(v) sinusoidal interpolation

Code part :- (1) Take any image
cv.resize has 5 parameters like
img, fx, fy, interpolation
fx, fy = factors
size = None [can use this also]
cv.resize(image, (300, 300), interpolation = cv.INTER_LINEAR)

Cubic interpolation is much better

Translation

* → Shifting the image in co-ordinate space

→ adding certain values to the (x, y) co-ordinates

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

t_x = translation along x axis
t_y = translation along y axis

Code : ① Read the image
matrix = 2 rows, 3 columns
matrix = numpy.float32([[1,0,100],[0,1,100]])

Apply matrix using warp affine

translated = cv.warpAffine(image, matrix, (image.shape[0], image.shape[1]+100))

warpAffine = rows, columns

* Rotation

$$\begin{bmatrix} \alpha & \beta & (1-\alpha)\cdot center.x - \beta \cdot center.y \\ -\beta & \alpha & \beta \cdot center.x + (1-\alpha)\cdot center.y \end{bmatrix}$$

$\alpha = scale \cdot \cos\theta$

$\beta = scale \cdot \sin\theta$

- 2 parameters can be change
  scale, angle

θ = angle with which it is rotating

Matrix = cv.getRotationMatrix2D
((h/2, w/2), 10, 1)

Transforms :- Euclidian
Affine and projective

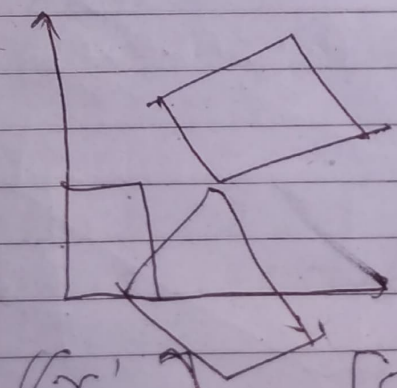
Geometric transform

(1) Modify spatial relationship b/w pixels

(2) Image can be shifted, Rotated can can be stretched.

Isometric Or Euclidian transforms

Whenever image is translated Or rotated around any point is called as Euclidian transform



$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & tx \\ \sin\theta & \cos\theta & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}$ new coordinates after translation

$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$ co-ordinates before translation

$$\begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}$$ θ = rotation angle

~~Reflection~~

Euclidian transform is a subset of affine transform.

1. Distance preserved
2. Angles preserved
3. Shapes preserved

Affine transformation

There are 6 degree's of freedom

square won't be square

parallel lines preserved but may be sheared

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

2 degrees for translation, one for rotation, one for scaling, one for scaling direction, one for scaling ratio

using 2 x 3 matrix I can rotate shear, translate and ~~rote~~ scale the image

Affine transform

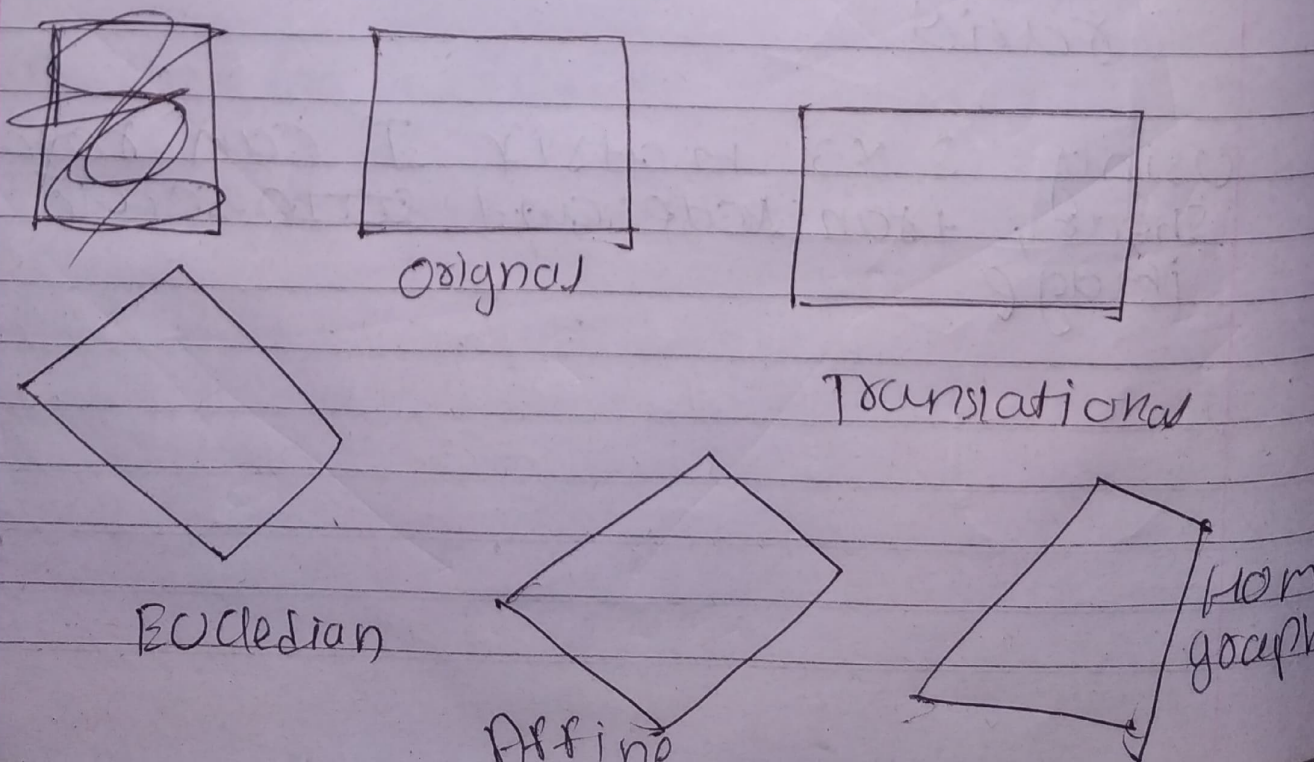
X, Y translation ✓
Shearing ✓
Translation ✓
scaling ✓
scaling ratio ✓
rotation ✓

projective transform

It is a 3×3 matrix

$$C\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Euclidean VS Affine VS projective Transform



Intrinsic and Extrinsic parameters

Extrinsic :-

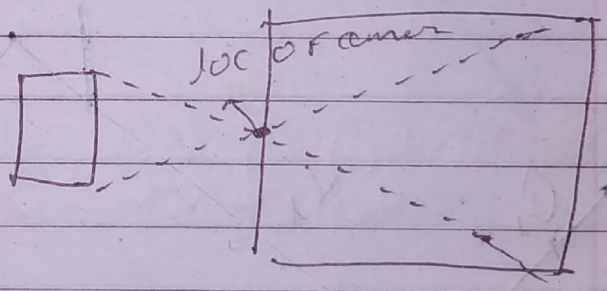
(1) where is my camera in 3D world

$$X_0 = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \qquad \theta = \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix}$$

3D orientation where is camera looking to.

It is a 6D freedom vector

$$\begin{bmatrix} X \\ Y \\ Z \\ \alpha \\ \beta \\ \gamma \end{bmatrix}$$



Intrinsic :-

(1) How a point in a 3D world is mapped on a 2D image.

(2) 4 or 5 parameters, camera constant, scale difference principal point

Mapping of a point on 2D

$(x, y, z) \longrightarrow (x, y)$

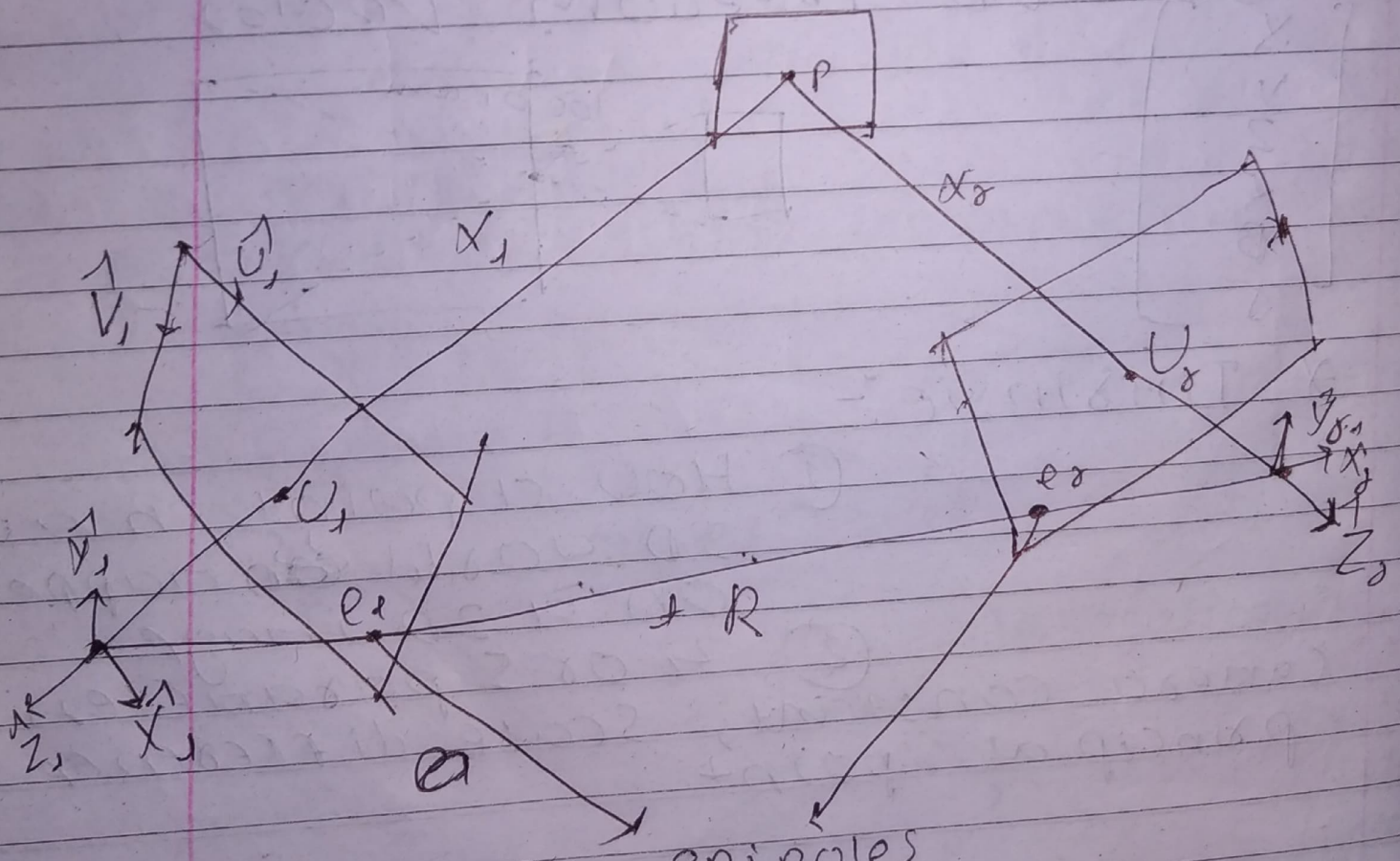
$x = P X$ ← 3D coordinate

↑ 2D coordinate

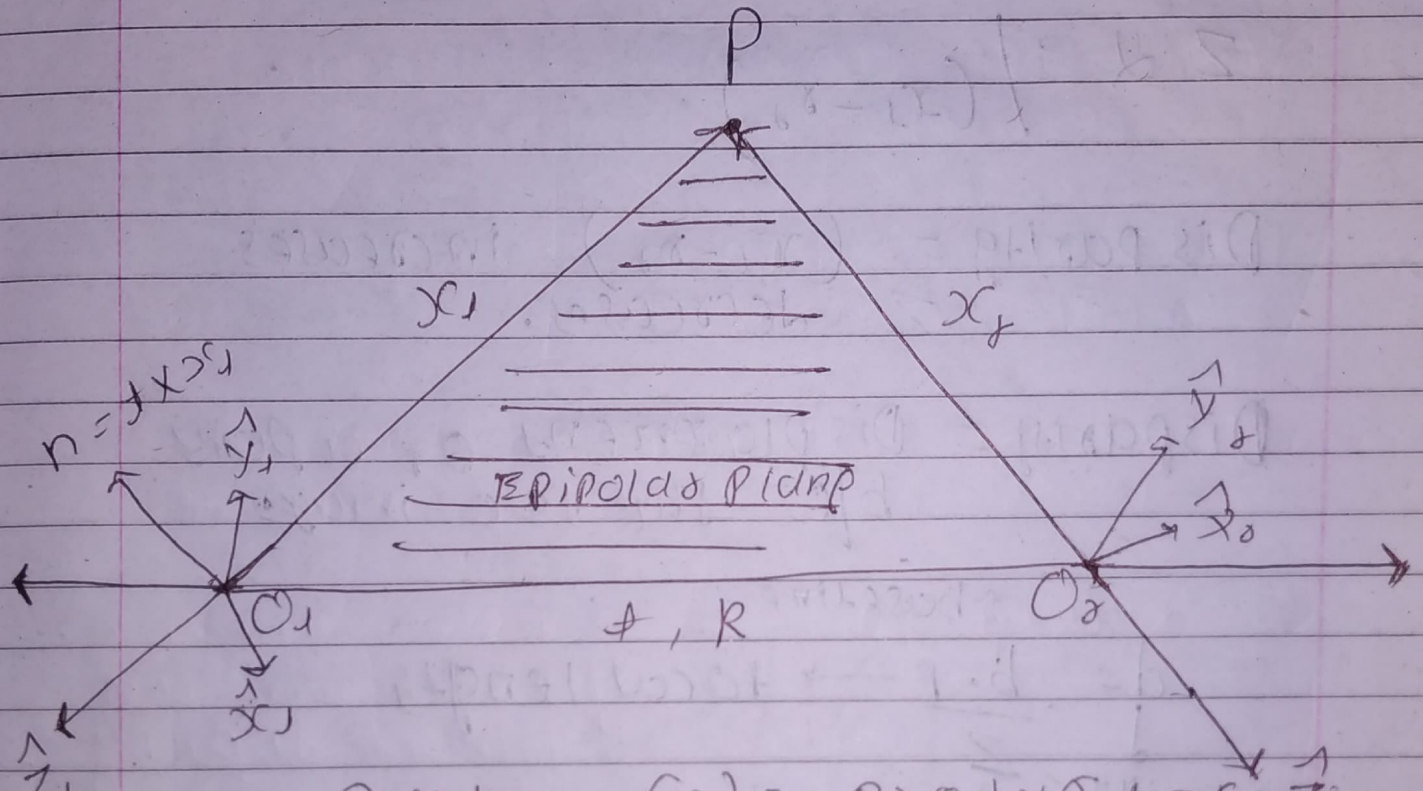
↑ projection matrix

Epipolar geometry:-

Relative position of 1 camera with other camera.



The triangular plane called as epipolar plane
Every scene point lies on a unique epipolar plane.



Normal (n) = product of $\hat{z}_2$ translational x y_1

$x_1 \cdot (t \times x_1) = 0$

Disparity and depth

depth is inversely proportional to disparity

$$Z \propto 1/(x_1 - x_2)$$

Disparity = $(x_1 - x_2)$ increases
Z decreases.

Disparity = Displacement of a point b/w the two images

$$d = \frac{b \cdot f}{Z}$$

b → baseline
f → focal length
d → disparity
Z → depth

* Algorithm

(1) Import Libraries such as opencv Numpy, Matplotlib

(2) Read left and right image respectively in grayscale

(3) Find Disparity map using StereoBM_create

(a) pass the parameters such as number of disparities, block size

(i) numDisparities :- It is the difference b/w maximum and minimum disparity

This parameter must be divisible by 16

(ii) BlockSize :- It is the size of the matching window to find the corresponding pixels in a rectified stereo image pair

If directly computes disparity it returns 16 bit signed single channel CV16S cons

* Other parameters that can be used :-

(i) prefilter Type :- parameter which is used to decide the type of filtering. It actually enhances the texture before passing to the block matching algorithm.

(ii) uniqueness ratio :- If the best matching disparity is not better than the other disparity in the search range, the pixel is filtered out

(iii) speckle window :- Speckles are produced near to the boundaries where the matching window catches the foreground and the background. To get rid of this we apply speckle window

(4) convert the image [disparity] into float 32 format and normalize it by dividing it by numdisparities

[As it directly computes disparity it returns 16 bit signed single channel]

(5) convert Disparity map into point cloud by using reprojectImageTo3D

parameters :- Disparity map

Q = 4x4 prespective transformation matrix

Q matrix can be obtained by using stereo.rectify and can be manually created

$$Q = \begin{bmatrix} 1 & 0 & 0 & -width/2 \\ 0 & 1 & 0 & -height/2 \\ 0 & 0 & 0 & focal\ length \\ 0 & 0 & -1/baseline & doffs/b \end{bmatrix}$$

doffs :- x difference of principle points, doffs = cx1 - cx0

reprojectImageTo3D returns 3 channel image representing 3D surface. That is for each pixel (x y)

(6) seperate those channels

(7) using hstack to stack the channels horizontally

(9) Reshape the Channels

(10) By using matplotlib library plot the image

(11) Visualize the same file in mesh lab by converting it into ply file