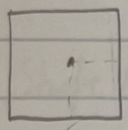
Forward Imaging Model : 3D to 2D.

world coordinates. $\underline{x}_w = \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix}$ → coordinates transformation → camera coordinates. $\underline{x}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$ → perspective projection → image coordinates $\underline{x}_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$

image plane → image sensor mapping

image plane has $(\hat{x}_i, \hat{y}_i)$ → image sensor has (u, v) → unit: pixel

$\hat{x}_i, \hat{y}_i$ unit: mm →

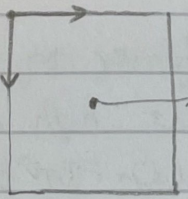


(pixel can be rectangular)

m_x, m_y: pixel densities (pixels / mm) in x & y directions.

$$u = m_x x_i = \underbrace{m_x f}_{f_x} \frac{x_c}{z_c}, \quad v = m_y y_i = m_y f \frac{y_c}{z_c}$$

we assume the location of principle point (centre of the image) as (O_x, O_y). This is unknown, and usually origin is default to top left corner.



$$u = f_x \frac{x_c}{z_c} + O_x, \quad v = f_y \frac{y_c}{z_c} + O_y$$

a scale factor which converts metres in 3D world coord into distances in pixels.

where $(f_x, f_y) = (m_x f, m_y f)$ are focal lengths in pixels in x & y directions.

f_x, f_y, O_x, O_y: intrinsic params of the camera. They represent camera's internal geometry.

params which align lens projection centre (the point of intersection from pinhole/optical axis ray to the sensor plane) wrt. centre of the sensor plane

intrinsic matrix: (perspective projection.)

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = \begin{bmatrix} f_x & 0 & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

camera intrinsic calibration: use $\begin{bmatrix} u \\ v \end{bmatrix}$, $\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$ to estimate focal lengths in pixels f_x, f_y & principal point in pixels (usually close to centre of the image) o_x, o_y.

where M_{int} is the intrinsic matrix,

$M_{int} = [K|0]$

& K is the calibration matrix,

$K = \begin{bmatrix} f_x & 0 & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix}$ which is an upper right triangular matrix.

$\begin{bmatrix} u \\ v \end{bmatrix}$ is the pixel coord. $\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$ is the 'camera coord.' use homogeneous 3D coord. $\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$ & homogeneous 4D coord. $\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$ to construct a matrix for easy computation.

in world coord.

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

→ in pixel coordinates. This is an approx. w/ few assumptions:

1. we use a pinhole camera model => ie: all light rays go through a single point (ie. (o_x, o_y)) instead of a lens.
2. => the camera is reduced to a 3D point (location at) the optical centre.

(& the optical centre is where all rays 'originate' from the pinhole camera).

optical axis is the vector from optical centre ⊥ to the image plane.

$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$ describe the 3D location of the obj. relative to the camera.

extrinsic parameters.

rotation matrix $R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$ → direction of $\hat{x}_c$ in world coordinate frame.

R is orthonormal:

NB: $\underline{u}$ & $\underline{v}$ are orthonormal iff:

1. $\underline{u}^T\underline{v} = 0$ 2. $\underline{u}^T\underline{u} = \underline{v}^T\underline{v} = 1$.

orthonormal matrix.

$R^{-1} = R^T \quad R^TR = RR^T = I$

extrinsic matrix : M_{ext}

$$\tilde{\underline{x}}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{M_{ext} = \begin{bmatrix} R_{3\times3} & \underline{t} \\ 0_{1\times3} & 1 \end{bmatrix}} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

$\tilde{\underline{x}}_c = M_{ext}\,\tilde{\underline{x}}_w$ ← coordinate transformation

$\begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix}$: world coord. position of obj. relative to world origin.

The transformation, M_{ext}, is a rigid body transformation (i.e. no shearing / deformation) that first rotate all points in world coord (using matrix R) then translate all points (using translation vector $\underline{t}$). s.t. From the camera's perspective, the world is now transformed as if the camera is fixed at the origin looking straight ahead.

world to camera , camera to pixel.

$M_{ext}\,\tilde{\underline{x}}_w = \tilde{\underline{x}}_c \qquad M_{int}\,\tilde{\underline{x}}_c = \tilde{\underline{u}}$.

projection matrix P:

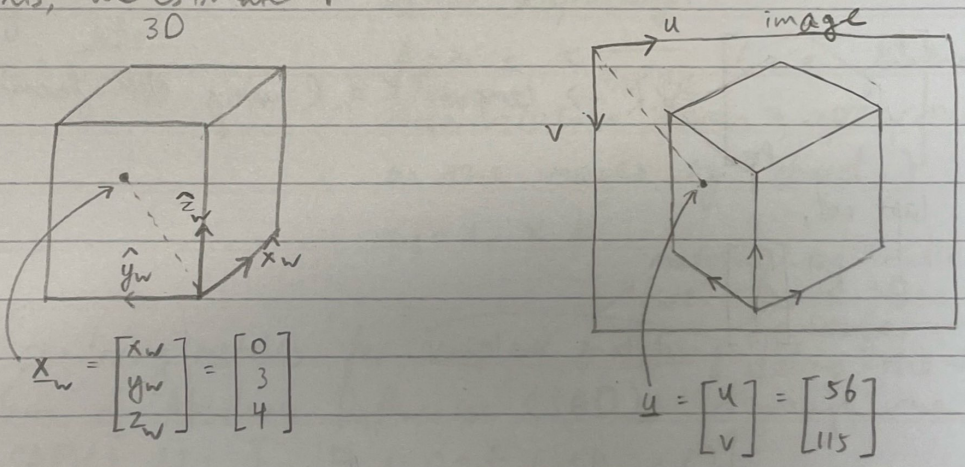
$$\tilde{u} = M_{int} M_{ext} \tilde{x}_w = P \tilde{x}_w$$

$$\begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = \underbrace{\begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \end{bmatrix}} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

we only need to know P to map all points in world coord. into the pixels in an image.

Camera calibration example

to identify the corresponse between 3D scene objects & image points, we estimate P:



$$\underline{x}_w = \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \\ 4 \end{bmatrix}$$

$$\underline{u} = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 56 \\ 115 \end{bmatrix}$$

for i^{th} point, unknown

$$\begin{bmatrix} u^i \\ v^i \\ 1 \end{bmatrix} = P \begin{bmatrix} x_w^i \\ y_w^i \\ z_w^i \\ 1 \end{bmatrix}$$

known

we rearrange s.t. we put all $\begin{bmatrix} u^i \\ v^i \end{bmatrix}$ & $\begin{bmatrix} x_w^i \\ y_w^i \\ z_w^i \end{bmatrix}$ into a single matrix A,

$\forall i \in \{1, \ldots, n\}$

and we rearrange matrix P into a vector $\underline{p}$:

$$\begin{bmatrix} P_{11} & P_{12} & \cdots \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix} \rightarrow \underline{p} = \begin{bmatrix} P_{11} \\ \vdots \\ P_{14} \\ P_{21} \\ \vdots \\ P_{24} \\ P_{31} \\ \vdots \\ P_{34} \end{bmatrix} \qquad A\underline{p} = 0$$

to estimate $\underline{p}$, $\min_{\underline{p}} \|A\underline{p}\|^2$ s.t $\|\underline{p}\|^2 = 1$

$A^T A\underline{p} = \lambda \underline{p}$ $\underline{p}$ is the eigenvector coresponding to the smallest eigenvalue λ of matrix $A^T A$.

? $\begin{bmatrix} P_{11} & & \\ & \ddots & \\ & & P_{33} \end{bmatrix} = KR \rightarrow$ compute K & R using QR factorisation

last col,

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

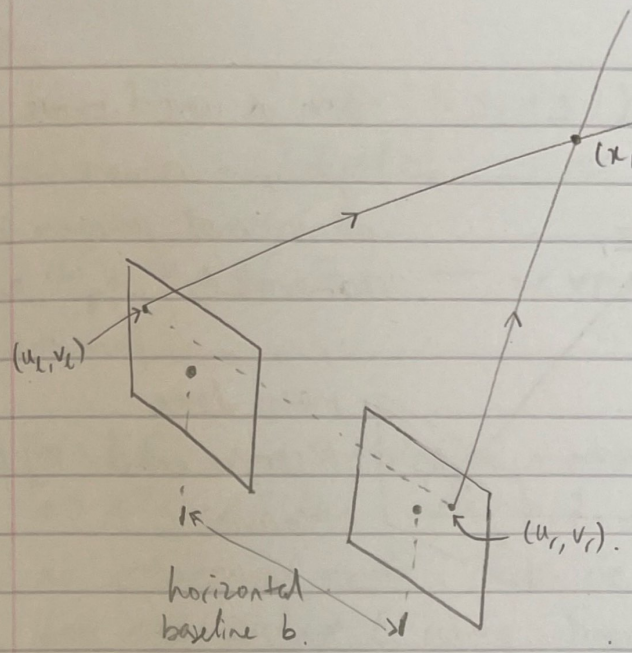
computing 2D → 3D : triangulation (need 2 imgs)

binocular vision : 2 cameras used to perceive depth by comparing 2 slightly diffrent views of the sane scene.

(stereo system : 2 camera system).

(Simplified stereo calibration) : 2 cameras place at the same height, b aparo. → horizontal baseline.

o<- - b - - ->o vs.



perspective projection eqn:

$$u_l = f_x \frac{x}{z} + o_x$$
$$v_l = f_y \frac{y}{z} + o_y$$

$$u_r = f_x \frac{x-b}{z} + o_x$$
$$v_r = f_y \frac{y}{z} + o_y$$

f_x, f_y, o_x, o_y, b are known since we've already calculated camera calibration at this point.

solve for (x,y,z):

$$\begin{cases} x = \frac{b(u_l - o)}{u_l - u_r} \\ y = \dots \\ z = \frac{bf_x}{u_l - u_r} \end{cases}$$

$u_l - u_r$: disparity.

depth $z \propto \frac{1}{u_l - u_r}$ (i.e. further away, then location doesn't appear that different on the images. eg. cloud.).

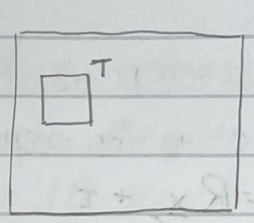
$u_l - u_r \propto b$.

use similarity metrics for template matching: matching the smaller 'patch' i.e. the template image with the larger image, by sliding the template through the img.

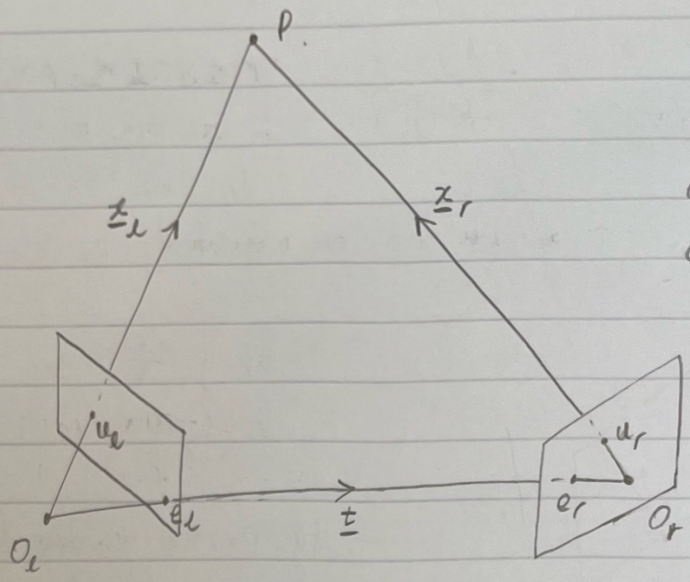
eg. Sum of Absolute Differences (SAD)

$$SAD(k,l) = \sum_{(i,j)\in T} |E_l(i,j) - E_r(i+k, j+l)|$$

for pixel $(k,l) \in L$ (large img); $(i,j) \in T$ (template).



SSD, NCC.



even in uncalibrate stereo system, we assume we know all internal params of 2 cameras. $f_x^{(l)}$, $f_y^{(l)}$, $o_x^{(l)}$, $o_y^{(l)}$, ", ", ", ", from the image's meta data.
⇒ camera matrix K_l & K_r is known.

O_l, O_r are camera centres (pinhole camera).
point P of the scene, O_l, O_r construct a unique <u>epipolar plane</u>.
plane O_lPO_r intersect the image planes at $u_l e_l$ are $u_r e_r$,
these 2 lines are known as <u>epipolar lines</u>.
camera's projection onto the other plane (i.e. the intersection w/ the image plane), at point e_l and e_r, are <u>epipoles</u>.
$\overrightarrow{O_lO_r}$ is the translation vector, $\underline{t}$.
we have vector $\underline{n}$ normal to the epipolar plane, $\underline{n} = \underline{t} \times \underline{x}_l$. (cross prod)

epipolar constraint: $\underline{x}_l \cdot (\underline{t} \times \underline{x}_l) = 0 = [x_l \; y_l \; z_l] \begin{bmatrix} t_y z_l - t_z y_l \\ t_z x_l - t_x z_l \\ t_x y_l - t_y x_l \end{bmatrix}$

$\equiv$

$$= [x_l \; y_l \; z_l] \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix} = 0$$

↓ translation matrix, T_x.
↳ the coord of P in left camera frame.

3D-to-3D translation of a point from left camera frame to its representation in the right camera frame.

$$\underline{x}_l = R\underline{x}_r + \underline{t}$$

$$\begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix} = \begin{bmatrix} r_{11} & & \\ & \ddots & \\ & & r_{33} \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

$$\underline{x}_l^T \cdot (T_x \times (R\underline{x}_r + \underline{t})) = \underline{x}_l^T \cdot (\underline{t}_x \times R\underline{x}_r + \underline{t} \times \underline{t}) \overrightarrow{0}$$

$$= \underline{x}_l^T \cdot T_x \times R\underline{x}_r \qquad \text{let } T_x \times R := E \text{ be the essential matrix.}$$

$$= \underline{x}_l^T \cdot E\,\underline{x}_r \leftarrow \text{our epipolar constraint.}$$

$E = T_x R$, where translation matrix T_x is skew-symmetric matrix. $\Leftrightarrow A^T = -A$ (aka, $a_{ij} = -a_{ji}$) eg. $a_{12} = -t_z = -a_{21}$. ; R is an orthonormal matrix. Thus, we can decompose T_x and R from their product E using Singular Value Decomposition.

$\underline{x}_l, \underline{x}_r$ are unknown. we use image coord u_l, u_r.

$$[u_l \; v_l \; 1] \underbrace{K_l^{-1^T} E K_r^{-1}} \begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix} = 0.$$

a 3x3 matrix:

fundamental matrix $F := K_l^{-1^T} E K_r^{-1} \Leftrightarrow E = K_l^T F K_r \Leftrightarrow T_x R$.

(first calc. F using $[u_l \; v_l \; 1] F \begin{bmatrix} u_l \\ v_l \\ 1 \end{bmatrix} = 0$. then calc. $E = K_l^T F K_r$,

lastly calc. T_x & R using SVD, once we work out $\underline{t}$ & R, we have calibrated the uncalibrated camera system).

(also, we find corresponding point pairs in l & r images using SIFT: $(u_l^{(1)}, v_l^{(1)}), \ldots, (u_l^{(m)}, v_l^{(m)})$.

Finding epipolar lines (aka. finding dense correspondence) ?

any point on left image, $(u_l^{(i)}, v_l^{(i)})$, corresponding to 1 point on the right epipolar line. So, we perform 1D search on the right epipolar line, to find the exact matching point.

given we calculated F, and $(u_l^{(i)}, u_l^{(i)})$, right epipolar line is given by: $a_l u_r + b_l v_r + c_l = 0$

where $a_l = f_{11} u_l + f_{21} v_l + f_{31}$, $b_l = f_{12} u_l + f_{22} v_l + f_{32}$,

$c_l = f_{13} u_l + f_{23} v_l + f_{33}$.

computing depth

Given the system of cameras are calibrated (relative position / translation from one frame to the other known, by $\underline{x}_2^T E \underline{x}_1$), we can work out the depth information from camera B given image produced by A

→ distortion coefficients.

$M_{int} \rightarrow M_{ext}$.

rectification image.

→ stereo

→ camera calibration : openCV documentation.

Keypoint detectors

keypoint: a distinct location in an image.
descriptor: a vector which summarises the local structure around the keypoint.

keypoint detectors: → tells me positions of local distinct points.

1. Harris.
2. Shi-Tomasi (sota, openCV) } structure matrix approaches.
3. Förstner

4. Difference of Gaussians ← compare between blurred images.

for structure matrix based approaches: for each pixel (x, y), compute the SSD in the local area window centred around (x, y) W_{xy} as:

$$f(x, y) = \sum_{(u,v) \in W_{xy}} (I(u,v) - I(u+\delta u, v+\delta v))^2$$

($I(u,v)$: a pixel in W_{xy}; $I(u+\delta u, v+\delta v)$: its neighbouring pixel)

where $I(u+\delta u, v+\delta v) \approx I(u,v) + [J_x \; J_y] \begin{bmatrix} \delta u \\ \delta v \end{bmatrix}$

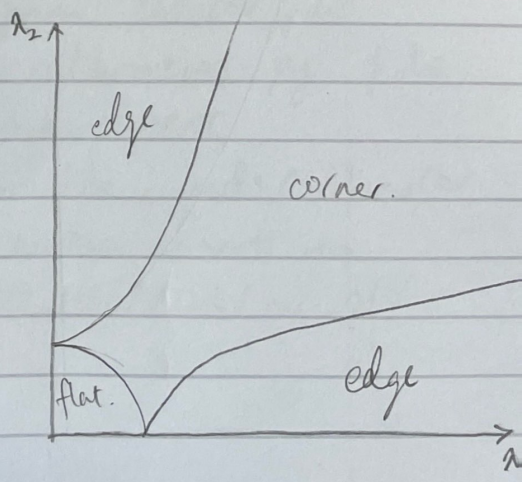
$$f(x, y) \approx \sum \left([J_x \; J_y] \begin{bmatrix} \delta u \\ \delta y \end{bmatrix} \right)^2 = \sum \begin{bmatrix} \delta u \\ \delta y \end{bmatrix}^T \underbrace{\begin{bmatrix} J_x^2 & J_x J_y \\ J_y J_x & J_y^2 \end{bmatrix}}_{\text{structure matrix, } M} \begin{bmatrix} \delta u \\ \delta y \end{bmatrix}$$

to detect the corner, criterion is that eigenvalues λ_1, λ_2 of the structure matrix are both big.

1. Harris corner detection:

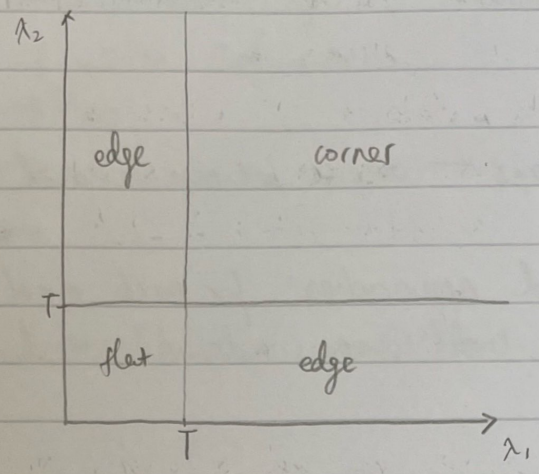
$$R = \det(M) - k(\text{trace}(M))^2$$
$$= \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2, \quad k = 0.04 \sim 0.06$$

$R \gg 0$: corner ($\lambda_1 \approx \lambda_2 \gg 0$)
$R < 0$: edge ($\lambda_1 \gg \lambda_2$ or $\lambda_2 \gg \lambda_1$)
$|R| \approx 0$: flat ($\lambda_1 \approx \lambda_2 \approx 0$)



2. suppose we set threshold T.
if smallest eigenvalue $\lambda_{min}(M) \geq T$: corner.

$$\lambda_{min}(M) = \frac{\text{trace}(M)}{2} - \frac{1}{2}\sqrt{(\text{trace}(M))^2 - 4\det(M)}$$



usual pratice: 1. RGB → greyscale 2. apply smoothing before doing corner detection

4. DoG: 1. apply Guassian smoothing of different kernel size, get imgs with different "degree of blurryness"
2. compare 2 blurry imgs and get their difference
(not only compare locally on the same img, but also w/ ↑ blurry / ↓ blurry imgs, to find the locally distinct key points).
3. compute maxima suppression

Feature descriptors

Feature descriptors: a vector which describes the patch around a keypoint.

1. SIFT: 16 patch × 8-bar histograms for gradient directions = 128-long vector.

we need to filter out incorrect match: use Lowe's Ratio Test

given we want to find best match for feature descriptor q, with either p_1 (best match) & p_2 (2^{nd} best match),

1. $d(q, p_1) < T$: Euclidean difference between q & p_1 must be less than threshold T.
2. $\frac{d(q, p_1)}{d(q, p_2)} < \frac{1}{2}$ only accept if p_1 is substantically better than p_2

2. Binary Descriptor: fast. good for online resources / videos

short binary str. 01011100...

- select a patch around a keypoint.
- select 2 pixels in that patch.
- compare the intensity of those 2 pixels: either 0 or 1.

$$b = \begin{cases} 1 & \text{if } I(s_1) < I(s_2) \\ 0 & \text{o/w} \end{cases}$$

Fast to compare: use Hamming distance

$d_H(B_1, B_2) = \text{sum}(\text{xor}(B_1, B_2))$

↙ bitstring!

example: BRIEF (256-bit binary feature descriptor)

cons: can't identify rotated same features.

solution: ORB (extension of BRIEF)

compared to SIFT, SIFT respond better to scale. (4096-bit) ORB (100× faster to compute than SIFT, 256-bit)

RANSAC

RANSAC - an algo which seperate in-lies (correctly identified pair of keypoints) w/ outlies (incorrectly identified pair of keypoints)

number of trails to succeed w/ prob. p is given by:

$$T = \frac{\log(1-p)}{\log(1-(1-e)^s)}$$

that all sampled points s are in-lies.

s: # sampled data point (this depend on my image-matching algo. eg. if I use 5-point algo. to determine the relationship between 2 images, $s = 5$. (larger the s, more difficult to draw these in-lies using trial-n-error approach). because all datapoints must be in-lies for RANSAC to draw correct results.

e: outlier ratio: $\frac{\#\text{outlies}}{\#\text{total datapoints}}$. (we don't always know this)

→ in terms of RANSAC, T ↑ exponentially with e & s! so, we would prefer an algo. with small s. eg. prefer 5-point algo. compared to 8-point algo.

→ RANSAC can't do multiple hypothesis - it's for finding the single best match (of sample points s, therefore single best correspondence between 2 images).

→ RANSAC used for computing eg. fundamental matrix, essential matrix. for computing visual odometry information.

DLT - extrinsic & intrinsic matrix calculation

DLT (direct linear transformation) map 3D world coord onto 2D image plane, assuming perfect lens. (since not counting in distortion, DLT is just an approx.)

camera calibration: compute intrinsic matrix / information

camera localisation: compute extrinsic information

we have 4 coord systems:

1. world, $S_o = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$ $X_o = \begin{bmatrix} X_o \\ Y_o \\ Z_o \end{bmatrix}$ R: rotation matrix
2. camera: origin @ projection centre of the camera $S_k = \begin{bmatrix} {}^kX \\ {}^kY \\ {}^kZ \end{bmatrix}$
3. image plane (2D plane which 3D world project onto. analogue) $S_c = \begin{bmatrix} c_x \\ c_y \end{bmatrix}$
4. sensor (pixel location: 'digital') $S_s = \begin{bmatrix} s_x \\ s_y \end{bmatrix}$

camera location: (X_o, R)

$x = PX$ x: 2D pixel coord X: 3D world

$$\underset{3\times4}{P} = K \underbrace{\begin{bmatrix} R & -RX_o \\ 0 & 1 \end{bmatrix}}_{3\times4} \rightarrow \text{aka. } M_{ext}: \begin{bmatrix} r_{11} & & & t_x \\ & \ddots & & t_y \\ & & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

K: 3×3 intrinsic

$\begin{bmatrix} R & -RX_o \\ 0 & 1 \end{bmatrix}$: projection & extrinsics.

we chain the transformation:

$$\begin{bmatrix} s_x \\ s_y \\ 1 \end{bmatrix} = {}^sH_c \; {}^cP_k \; {}^kH_o \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \qquad {}^kH_o = \begin{bmatrix} R & -RX_o \\ 0^T & 1 \end{bmatrix}$$

- ${}^sH_c$: image coord to sensor(pixel) coord
- ${}^cP_k$: transform from camera coord to image coord
- ${}^kH_o$: transform object coord to camera coord
- $\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$: 3D coord in world coord

(simplified) calibration matrix (aka. intrinsic matrix)

$$P = {}^sH_c \; {}^cK [R|-RK_o] = {}^sH_c \; {}^cK R [I_3|-X_o] = KR[I_3|-X_o]$$

- P: camera matrix. map world coord → image plane.
- ${}^cK$: (entire) calibration matrix
- where K^c is the calibration matrix for ideal camera,

$-X_o = \begin{bmatrix} -X_o \\ -Y_o \\ -Z_o \end{bmatrix}$: projection centre aka. pinhole location aka. 'origin of the camera', * in world coord *.

3D world coord → 3D camera coord is a rigid body transformation, done by translation & rotation only.

(distortion)

3. lens are imperfect: (eg. barrel effect: straight line mapped to curves on image plane. to address this issue, we apply non-linear transformation after all linear transformations. This non-linear transformation depend on location of the pixel as well as parameters.

eg. for barrel affect, the transformation required:

$$\begin{cases} {}^{a}x = x(1+q_1 r^2 + q_2 r^4) \\ {}^{a}y = y(1+q_1 r^2 + q_2 r^4) \end{cases}$$

r: radius from principle point in the image to the pixel.

q_1, q_2: params we need to calculate.

x, y: location of the pixel.

2. In the lecture, calibration matrix (K) is breakdown into: $K = {}^{S}H_{c}\,{}^{c}K$

$$= \begin{bmatrix} 1 & s & x_H \\ 0 & 1+m & y_H \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c & 0 & 0 \\ 0 & c & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c & cs & x_H \\ 0 & c(1+m) & y_H \\ 0 & 0 & 1 \end{bmatrix},$$

c: camera constant

s: sheer

m: scale difference

x_H, y_H: principle point.

$$P = K R [I_3 | -X_0] \quad (3\times 4)$$

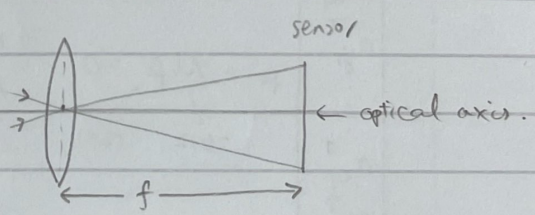
where $X_0 = \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix}$

K: intrinsics; R, X_0: extrinsics.

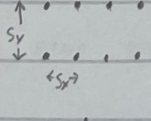
→ 1 calibration (intrinsic) matrix: $K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$

f_x, f_y are focal length in pixels. f is the focal length, in mm. — distance between the pinhole (optical centre) to the sensor plane.

f_x, f_y are focal length in pixel units.



s_x, s_y: sensor's pixel pitch: distance between the centre of adjacent pixels, measured in mm.



$$f_x = \frac{f}{s_x}, \quad f_y = \frac{f}{s_y}$$

eg. my sensor is 4000 × 3000 pixels, I may have $c_x = 1980$ px, $c_y = 1510$ px.

c_x, c_y: pincipal-point coordinates, the point which is projected from pinhole along optical axis onto the sensor plane. should be v. close to centre of the sensor/image, but usually off set by few mm. in pixel units.

DLT for calibration

(11 degrees of freedom 6 extrinsic, 5 intrinsic)

when we estimate extrinsic & intrinsic matrix, we're assuming affine camera: i.e. no non-linear distortion due to lens. Note when calibrating, should select 3D keypoints in the world that doesn't lie on the same plane. (otherwise estimation becomes unstable)

$x = PX \Rightarrow \begin{bmatrix} u \\ v \\ w \end{bmatrix} = P \begin{bmatrix} U \\ V \\ W \\ T \end{bmatrix}$, 2D & 3D coord in homogeneous coord. (since P is 3×4 matrix. 3D & 4D vector makes it convinient that single matrix P exist).

normalise it

$\begin{bmatrix} \frac{u}{w} \\ \frac{v}{w} \\ 1 \end{bmatrix} = P \begin{bmatrix} \frac{U}{T} \\ \frac{V}{T} \\ \frac{W}{T} \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$

Let $x_i = PX_i = \begin{bmatrix} P_{11} \\ P_{21} \\ P_{31} \end{bmatrix} X_i \begin{matrix} \rightarrow \text{vector A} \\ \rightarrow \text{vector B} \\ P_{34} \rightarrow \text{vector C.} \end{matrix}$

$\Rightarrow x_i = \begin{bmatrix} A^T \\ B^T \\ C^T \end{bmatrix} X_i = \begin{bmatrix} A^T X_i \\ B^T X_i \\ C^T X_i \end{bmatrix}$

$\Rightarrow x_i = \frac{u_i}{w_i} = \frac{A^T X_i}{C^T X_i}$, $y_i = \frac{v_i}{w_i} = \frac{B^T X_i}{C^T X_i}$

$\Rightarrow \begin{cases} -X_i^T A + x_i X_i^T C = 0. \\ -X_i^T B + y_i X_i^T C = 0. \end{cases}$ Let $p = \begin{bmatrix} A \\ B \\ C \end{bmatrix}$

rewrite as

$\begin{cases} a_{x_i}^T p = 0 \\ a_{y_i}^T p = 0. \end{cases}$ where $a_{x_i}^T = (-X_i^T, 0^T, x_i X_i^T)$ (4, 4, 4)

$a_{y_i}^T = (0^T, -X_i^T, y_i X_i^T)$.

stack it into one...

$\begin{bmatrix} a_{x_1}^T \\ a_{y_1}^T \\ \vdots \\ a_{x_I}^T \\ a_{y_I}^T \end{bmatrix} p = Mp = 0.$ (M: 2I×12, p: 12×1)

note M is known, p is what we want to solve.

realistically... $Mp = w$ for small w which we want min.

$\Rightarrow \hat{p} = \arg\min_p w^T w = \arg\min_p p^T M^T M p$

using SVD, $M = USV^T$ where S is a diagonal matrix w/ largest first, smallest last (0 would be ideal). (now P is found!)

V^T store the singular vectors (eigvectors) $\Rightarrow$ choose smallest eigvector for p.

DLT for calibration

now, we obtained 3×4 matrix P. how do we calc. K, R, X_0?
(aka. M_{int}). camera location in world coord.

$P = [KR \mid -KRX_0] = [H \mid \underline{h}]$

$\Rightarrow X_0$ (projection centre) $= -H^{-1}h$

$H = KR$, decompose using QR decomposition.
QR decomposition of $H^{-1} = (KR)^{-1} = R^{-1}K^{-1} = R^TK^{-1}$
then, rotation matrix $R = \underline{R}(z,\pi)R$ where $\underline{R}(z,\pi) = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
finaly, normalise $K = \frac{1}{K_{33}} KR(z,\pi)$.

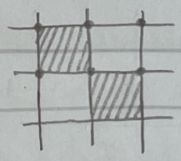
Zhang's method (find intrinsics) - calibration.

for $x = PX = KR[I_3 \mid -X_0]X$

- x: image point (sensor)
- X: world location
- K: intrinsics, which we want to find out using Zhang's method. (5 params)

using checkerboard for camera calibration (assuming we know size & structure - aka. flat surface)

For every image, we define a coord s.t. the checkerboard forms the XY plane $\Rightarrow$ All points on the checkerboard, has $Z = 0$.



$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} c & cs & x_H \\ 0 & c(1+m) & y_H \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

now, for checkerboard points, $Z = 0$.

$$\Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} & -K & \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

unknown which we need to calc

$$\Rightarrow \text{let } H = \begin{bmatrix} | & | & | \\ h_1 & h_2 & h_3 \\ | & | & | \end{bmatrix} = K \begin{bmatrix} | & | & | \\ r_1 & r_2 & t \\ | & | & | \end{bmatrix} \quad \Rightarrow \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = H \begin{bmatrix} X_i \\ Y_i \\ 1 \end{bmatrix}$$

Zhang's method

using DLT, $\begin{cases} a_{x_i}^T \underline{h} = 0 \\ a_{y_i}^T \underline{h} = 0 \end{cases}$

with $a_{x_i}^T = (-X_i, -Y_i, -1, 0, 0, 0, x_i X_i, y_i Y_i, x_i)$

$a_{y_i}^T = (0, 0, 0, -X_i, -Y_i, -1, y_i X_i, y_i Y_i, y_i)$

we need at least 4 points to determine estimate for H.

$H = K[\underline{r_1}, \underline{r_2}, \underline{t}]$, Let $B = K^T K^{-1}$. K is upper triangular $\Rightarrow$ K is invertible.

$[\underline{r_1}, \underline{r_2}, \underline{t}] = K^{-1}[\underline{h_1}, \underline{h_2}, \underline{h_3}]$ $\quad \begin{cases} \underline{r_1} = K^{-1}\underline{h_1} \\ \underline{r_2} = K^{-1}\underline{h_2} \end{cases}$

$\underline{r_1}, \underline{r_2}, \underline{r_3}$ are orthonormal $\Rightarrow \underline{r_1}^T \underline{r_2} = 0$, $\|\underline{r_1}\| = \|\underline{r_2}\| = 1$

$\Rightarrow$ constraints: $\begin{cases} \underline{h_1}^T \overbrace{K^{-T}K^{-1}}^{B} \underline{h_2} = 0 \\ \underline{h_1}^T K^{-T}K^{-1}\underline{h_1} - \underline{h_2}^T K^{-T}K^{-1}\underline{h_2} = 0 \end{cases}$

Cholesky decomposition: $chol(B) = AA^T$ where $A = K^{-T}$

define $\underline{b} = (b_{11}, b_{12}, b_{13}, b_{22}, b_{23}, b_{33})$

$B = \begin{bmatrix} b_{11} & \cdots \\ \vdots & b_{33} \end{bmatrix}$ coefficient

$\Rightarrow$ constraints: $\begin{cases} \underline{v_{12}}^T \underline{b} = 0 \\ \underline{v_{11}}^T \underline{b} - \underline{v_{22}}^T \underline{b} = 0 \end{cases} \Rightarrow \begin{pmatrix} \underline{v_{12}}^T \\ \underline{v_{11}}^T - \underline{v_{22}}^T \end{pmatrix} \underline{b} = 0$

2×6 matrix; $\underline{b}$: 1×6

Given n images, define matrix V s.t. $V\underline{b} = 0$.

$$\begin{matrix} \text{img 1} \rightarrow \\ \\ \\ \\ \text{img n} \rightarrow \end{matrix} \begin{pmatrix} \underline{v_{12}}^T \\ \underline{v_{11}}^T - \underline{v_{22}}^T \\ \vdots \\ \underline{v_{12}}^T \\ \underline{v_{11}}^T - \underline{v_{22}}^T \end{pmatrix} \underline{b} = 0. \quad \text{impose } \|\underline{b}\| = 1,$$

using SVD, smallest eigenvalue $= \underline{b}^* = \underset{\underline{b}}{\operatorname{argmin}} \|V\underline{b}\|$