Essentials in Computer Vision

- Pick Demon ⇒ goal: how to integrate models into each other
  - register camera from a set of 2D projections → very used in computer vision
    by knowing how the world looks like

- exercise 3D registration:
  generic observer → just the direction

  direction vector are belong to a non Euclidean space → surface of a sphere (S^2)
  ↓
  measurements space would lie on a manifold

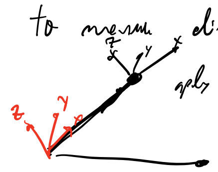
How to proceed?

- traditional one by defining the $\boxplus$ operator

$$X \boxplus \Delta x : [R(\Delta\alpha) \mid R(\Delta\alpha)t + \Delta t] \longrightarrow \text{when } \Delta\alpha = 0, \; \cancel{t(\Delta\alpha)} = t$$

$z \in S^2 \Rightarrow$ unit vector (over parametrization, given that we only need 2 DoF to describe a unit vector)
⇓

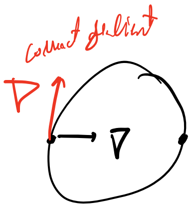
$R_1 : R(\theta) R(\varphi_1)$ — to measure distance in geodetic
azimuth, elevation?



apply a rotation so that x aligns with first point $(x \,/\!/\, z_1)$

$y \perp x$ ④ y lying on the plane x_0

- learn something on the gradient



Main Role of Epipolar Geometry on Unit Sphere

- determine up to a scale the relative transformation between 2 images / a generic set of lines passing through the same projective center

  w/o knowing the points!

Rays in lines

4D Lines

Camera = measure directions of objects in the world (image projection of a point = ray passing through the center of the camera and reaching the point in the plane) but cannot sense distance to the target/object

LADY BUG

$l_0 = p_0 + d_0 \cdot \lambda_0$ → scalar

↓ 1 generic point on a line

↓ unit vector (direction)

⇓ overparametrization of the line
just need 2 components (4DoF) → keep 6DoF due to being convenient

Transforme line

→ determine a new set of parameters

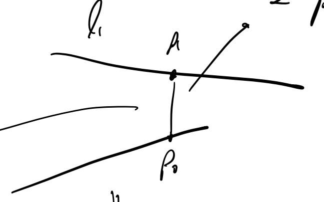
$$l_1 = T \cdot l_0 = R p_0 + t + R d_0 \lambda_0$$

$T = [R|t]$

↓ direction just rotated
not subject to the translation part of a transformation

How to determine closest points passing by 2 lines:

- analytical ways
- least-squares machinery!



2 points that form closed distance between 2 lines (4 a line)

⇓ not necessarily intersected

Δ = delta between 2 points

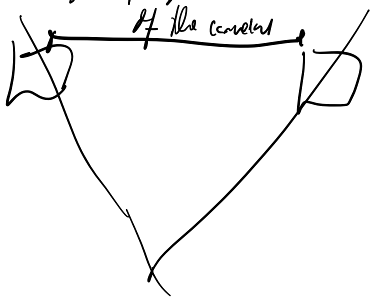
$$\arg\min \| \Delta_p + \Delta_d \lambda \|^2 = \text{minimum norm solution}$$

⇓ easily solved by least squares

$$\lambda^* = \begin{bmatrix} \lambda_0^* \\ \lambda_1^* \end{bmatrix}$$ → two distances to reach closest points

↓ very complicated way for triangulation problem ...?

[illegible] [illegible]



Intersection condition between two lines: → must be fulfilled while registering 2 LiDAR sensors

of all lines converging to the same point intersection → also possible to register

$$\emptyset = (p_0 - p_1) \cdot (d_0 \times d_1)$$ → ORTHOGONALITY CONDITION → it verifies when 2 lines intersect

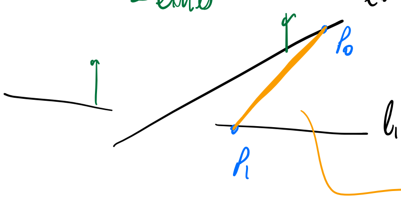
(when parallel may also be $\emptyset$, but $d_0 \times d_1 = \emptyset$ ...)

$d_0 \times d_1 \equiv$ orthogonal to the 2 lines vector

↓

Cross product ??

(plane definition)



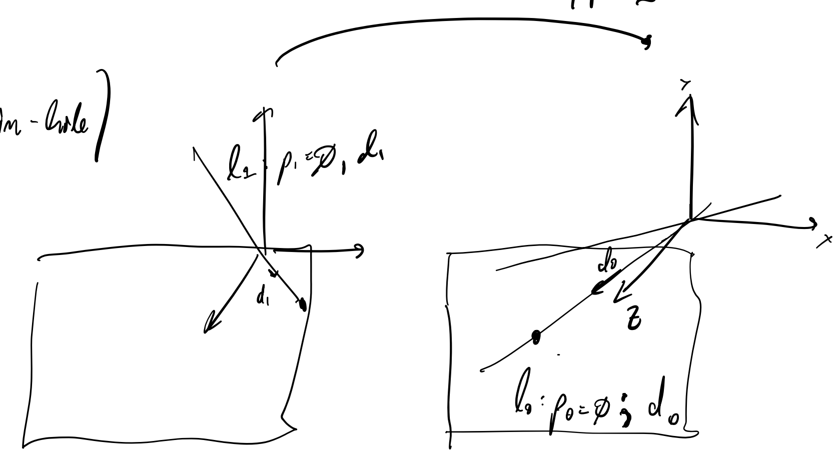
do not intersect

project to the orthogonal direction (= dot-product)

2 random point on the line

$T = [R|t]$

all rays pass through the center of my sensor (pin-hole)



$$\emptyset = (p_0 - p_1) \cdot (d_0 \times d_1)$$

⇓

$$t \cdot (R d_0 \times d_1) =$$

⇓

$$l_0' = t, R d_0$$

↓ in the camera frame 1

CROSS-PRODUCT IDENTITIES

$$= d_1 \left(t \times (R d_0) \right)$$

cross-product = skew - matrix

$$= d_1 \cdot [t]_\times R d_0$$

we overspeci skew matrix!

$$= d_0^T \underbrace{\left(R^T [t]_\times \right)}_{} d_1$$

d_0, d_1 2 directions they intersect must be

ESSENTIAL MATRIX → admits ∞ solutions

(if we multiply by a scalar, remains valid)

put scalar in the translation

from E, how to determine R and t ⇐ (only 5 DoF) = $R \oplus t$ up to a scale

t defined up to a scale

$$\boxed{E = R^T [t]_\times} \rightarrow \mathrm{rank}(E) \leq 2$$

full rank

rank at most 2

svd decomposition of E

$$= U S V^T$$

$$S = \begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & 0 \end{bmatrix}$$

3rd singular value which is 0

W rotates the components of σ_1 and σ_2

+
our game is to decompose E into R and t

→ start from SVD

→ $W S W^T = S$ ⇒ can substitute in the formula

$V^T V = I$ ⇒ can be used in the middle

4 solutions for W, W^T, t, t^T!

↳ if $W, -W$ or W^T ≡ 4 possible solutions

↳ we should also consider t and $-t$

R, t → triangulate the points ⇒ figure it out where most of the point fall in front of the camera

⇓

How?

if direction is exiting, ←

Δ(origin) is POSITIVE!

How to estimate E?

$d'^T E d = \emptyset$ ≡ epipolar constraint!

↳ make infinitesimal small that all lines intersect

$e^* = \dots$ $\qquad E = \emptyset \Rightarrow$ TRIVIAL SOLUTION

in general, we have many directions that intersect

enforce $\|e\| = 1$

norm of ALL elements in $\underline{e}$!

(where $e = (e_{11}\ e_{12} \cdots e_{21}\ e_{22} \cdots e_{33})$ 1×9 vector

$d' = (x'\ y'\ z')^T$

$d = (y\ x\ z)$

s.t. = subject to

$e^T e = 1 \rightarrow$ we need to enforce the constraint!

$H_{9\times 9}$

Constrained minimization $\Rightarrow$ Lagrange multipliers!

$$\mathcal{L}(e, \lambda) = e^T H e - \lambda(e^T e - 1)$$

function to minimize $(f(e))$

equation of the constraint

$$\begin{cases} \dfrac{\partial}{\partial e} = 2He - 2\lambda e = \emptyset \\ \dfrac{\partial}{\partial \lambda} = e^T e - 1 = \emptyset \end{cases}$$

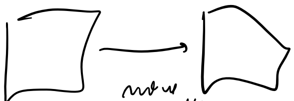
$\Rightarrow$ does not ring a bell?

EIGEN VECTOR PROBLEM

e is an eigenvector of H

which one to choose?

$$\boxed{He = \lambda e}$$



move a little

choose which one make cost function smaller!

$$e_j^T H e_j = e_j^T \lambda_j e_j = \lambda_j \quad \underline{\underline{\text{cost}}}$$

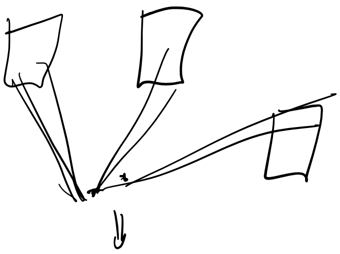
pick eigen vector that corresponds to the smallest eigen value!

1° – E

2° – R, t that correspond to E

3° – triangulate pts

4° – pick solution max # pts in front (up to a scale!)



may be drift over time (if move very little, 99 cases)

8 point algorithm — epipolar geometry / fundamental matrix (8 DoF? confirm!)

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

this one was missing!

src/octave/tools/epipolar_geometry:

instead of putting K, put eye(3) every will be consistent

triangulated

Lemma — if too far, we may not want to triangulate that point right-handed condition? ...

$$E = K^T F K$$ (Fundamental → essential)

Preconditioning? ...

test_epipolar.m ⇒ same "test_epipolar.m"
X = generateTransform()

#n_points

testTriangulatePoints(X, 100)

↓

many zero why? ⇒ ILLEGAL CONDITION!

P1 ~~P2~~ - ground-truth | pts behind the camera

P - triangulated points

test Transform ⇒ generate 2 sets of points
particular image w/ certain amount of noise

How to compare 2 translations? - divide point wise the two translations

= if proportional ok

20 points not getting visual estimate

are projected ⇒ some of them are outside !

(may be in the back of the mesh)

when using

preconditioning works... why?

200, 1, false — more influence estimate......

less noise

⇓

PRECONDITIONING — rule the things?

↳ typically helps!

puts most of the points in the centre