Probabilistic Robotics - Least Squares → based on Robotics @ Sapienza University of Rome (Prof. Dr. Giorgio Grisetti)

Least Squares Introduction

Maximum Likelihood Estimation (MLE)

$$x^* = \arg\max_x p(x|z) \equiv X_{\text{STATE}}$$

x^*: most likely values of this distribution

very complex in a probabilistic way

→ Bayes Rule: $p(x|z) = \frac{p(z|x)\,p(x)}{p(z)}$

- $p(z|x)$: OBSERVATION MODEL
- $p(x)$: PRIOR OVER POSSIBLE STATE
  - if we do not know anything → becomes a UNIFORM DISTRIBUTION = CONSTANT
  - $\left.\frac{p(x)}{p(z)}\right|_{p(x)\text{ uniform dist}}$ would also be a constant

$$p(x|z) \propto p(z|x) = \prod_i p(z^{[i]}|x)$$

Gaussian assumption (measurements → Gaussian noise)

assuming independent measurements

$$p(z^{[i]}|x) = \mathcal{N}\left(z^{[i]};\ h^{[i]}(x),\ \Sigma^{[i]}\right)$$

- $z^{[i]}$: MEASUREMENT
- $h^{[i]}(x)$: prediction of the measurement
- x: STATE
- $p(z^{[i]}|x)$: LIKELIHOOD

$$\propto \exp\left(-\underbrace{\left(h^{[i]}(x) - z^{[i]}\right)^T}_{\text{error function: } e^{[i]}(x)} \cdot \Sigma^{[i]-1} \cdot \left(h^{[i]}(x) - z^{[i]}\right)\right)$$

$\Omega^{[i]} = \Sigma^{[i]-1}$: information matrix

Through Gaussian assumption:
- MAXIMIZATION ⇒ MINIMIZATION
- PRODUCT ⇒ SUM

$$x^* = \arg\max_x \prod_i p(z^{[i]}|x)$$

$$= \arg\max_x \prod_i \exp\left(-e^{[i]}(x)^T \cdot \Omega^{[i]}(x) \cdot e^{[i]}(x)\right)$$

log-likelihood

$$= \arg\min_x \sum_i e^{[i]}(x)^T \cdot \Omega^{[i]}(x) \cdot e^{[i]}(x)$$

$e^{[i]}(x)$ is not a linear function (usually) → requires iterative minimization!

Gauss-Newton

→ iterative minimization of $F(x) = \sum_i e^{[i]}(x)^T \cdot \Omega^{[i]} \cdot e^{[i]}(x)$

$x \leftarrow x + \Delta x$ → each iteration refines the current estimate by applying a perturbation

⇒ perturbation obtained by minimizing a quadratic approximation in Δx

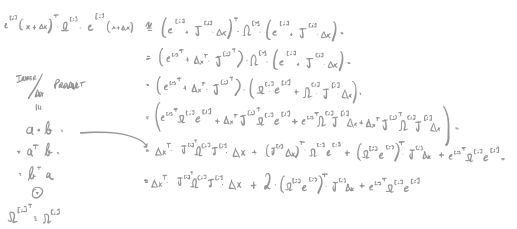
$$F(x + \Delta x) \simeq \Delta x^T H \Delta x + 2 b^T \Delta x + c$$

Quadratic form = Higher Dimension of 2nd-grade Polynom

$$e^{[:]}(x^* + \Delta x) \cong e^{[:]}(x^*) + \left.\frac{\partial e^{[:]}(x)}{\partial x}\right|_{x=x^*} \cdot \Delta x$$

$e^{[:]}(x^*)$ = $e^{[:]}(x)$; $\left.\frac{\partial e^{[:]}(x)}{\partial x}\right|_{x=x^*}$ = $J^{[:]}$

$$F(x+\Delta x) = \sum_i e^{[:]}(x+\Delta x)^T \cdot \Omega^{[:]} \cdot e^{[:]}(x+\Delta x)$$



$$\cong \sum_i \left[\Delta x^T \cdot \underbrace{J^{[:]T} \Omega^{[:]} J^{[:]}}_{H^{[:]}} \cdot \Delta x + 2 \cdot \underbrace{\left(\Omega^{[:]} e^{[:]}\right)^T J^{[:]}}_{b^{[:]T}} \cdot \Delta x + \underbrace{e^{[:]T} \Omega^{[:]} e^{[:]}}_{c^{[:]}} \right]$$

$$b^{[:]} = J^{[:]T} \cdot \Omega^{[:]} e^{[:]}$$

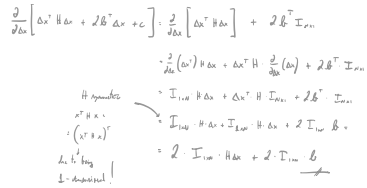
$$\sum_i e^{[:]T} \Omega^{[:]} e^{[:]} = \chi_e$$

CUMULATIVE ERROR

$$\Delta x^* = \underset{\Delta x}{\operatorname{argmin}} \; F(x^* + \Delta x)$$

$$\cong \underset{\Delta x}{\operatorname{argmin}} \left(\Delta x^T H x + 2 b^T \Delta x + c \right)$$

$$\Leftrightarrow \quad \emptyset = \frac{\partial \left[\Delta x^T H x + 2 b^T \Delta x + c \right]}{\partial \Delta x} \quad \Leftrightarrow$$



$$\Leftrightarrow \; -\cancel{2} \cdot \cancel{I_{1\times n}} \cdot b = \cancel{2} \cdot \cancel{I_{1\times n}} \cdot H \Delta x \; \Leftrightarrow$$

$$\Leftrightarrow \; -b = H \Delta x$$

if $y = mx + b$ w/ $e^{[:]}$ affine

$\Downarrow$

1 iteration is enough to find the minimum !

$\left(\frac{\partial \chi}{\partial x} = m \equiv \text{constant coefficient}\right)$

1 ITERATION

$H \leftarrow \emptyset$; $b \leftarrow \emptyset$

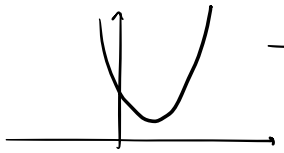
FOR EACH MEASUREMENT, UPDATE H AND b:

- $e^{[:]} \leftarrow h^{[:]}(x^*) - z^{[:]}$
- $J^{[:]} \leftarrow \left.\frac{\partial e^{[:]}(x)}{\partial x}\right|_{x=x^*}$ —— $\dim(J) = \dim(z^{[:]}) \times \dim(x)$
- $H \leftarrow H + J^{[:]T} \Omega^{[:]} J^{[:]}$ —— $\dim(H) = \dim(x) \times \dim(x)$
- $b \leftarrow b + J^{[:]T} \Omega^{[:]} e^{[:]}$ —— $\dim(b) = \dim(x) \times 1$

$\Delta x \leftarrow$ solve $(H \Delta x = -b)$ —— update estimate w/ perturbation

if x had 1 dimension:

$$F(x+\Delta x) \cong h \cdot x^2 + 2b \cdot x + c$$



→ HOW TO FIND THE MINIMUM ?

derivative → $\emptyset$

$X^* \longleftarrow X^* + \Delta x$

Methodology

1. Identify the state space X
   - Qualify the domain
   - Find a locally Euclidean parametrization ⟵ given that Gauss-Newton assumes linear parametrization!
2. Identify the measurement space Z
   - Qualify the domain
   - Find a locally Euclidean parametrization
3. Identify the prediction function $h(x)$

IF SUFFICIENTLY GOOD, needed a strategy to improve initial pose information!

Gauss-Newton in SLAM

- Calibration
- Registration
  - Cloud to Cloud (ICP)
  - Image to Cloud (Projective ICP)
- Global Optimization
  - Pose SLAM
  - Bundle Adjustment

- Data association assumed to be known...
- GN alone not sufficient to solve a full problem
- NEEDED A STRATEGY TO COMPUTE DATA ASSOCIATION

Example: Iterative Closest Point (ICP) 2D

→ find a transformation that minimizes distance between corresponding points

≡ pose of the world w.r.t. the laser coordinate frame (with respect to)

Leads to more simple calculations than the other way around!

1. State (domain)

$X \in SE(2)$ ⟵ 2D homogeneous transformation

$x = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$ $\left.\begin{matrix} \\ \\ \end{matrix}\right\} = t$ (translation)
⟵ rotation

$R(\theta) = \begin{bmatrix} c_\theta & -s_\theta \\ s_\theta & c_\theta \end{bmatrix}$ ⟶ $R'(\theta) = \frac{\partial R}{\partial \theta} = \begin{bmatrix} -s_\theta & -c_\theta \\ c_\theta & -s_\theta \end{bmatrix}$

$h^{[i]}(x) = R(\theta)\, p^{[i]} + t$ → measurement function

$\frac{\partial h^{[i]}(x)}{\partial x} = \begin{pmatrix} \frac{\partial h^{[i]}(x)}{\partial t} & \frac{\partial h^{[i]}(x)}{\partial \theta} \end{pmatrix}$ ⟵ measurement Jacobian

$\frac{\partial h^{[i]}(x)}{\partial t} = I_{2\times 2}$

$\frac{\partial h^{[i]}(x)}{\partial \theta} = R'(\theta)\, p^{[i]}$

2. Measurements (domain)

prediction of the measurement based on the state

$z \in \mathbb{R}^2$

$z = \begin{pmatrix} x \\ y \end{pmatrix}$ ⟶ $h^{[i]}(x) = R(\theta)\, p^{[i]} + t$

Rotation matrix: $R(\theta) = \begin{bmatrix} c_\theta & -s_\theta \\ s_\theta & c_\theta \end{bmatrix}$

(in this case)

$\frac{\partial e^{[i]}(x)}{\partial x} = \frac{\partial h^{[i]}(x)}{\partial x}$ ⟸ does not depend on the state (in this case)

$\Rightarrow e^{[i]}(x) = \left(h^{[i]}(x) - z^{[i]} \right)$

$[e, J] = \text{ERROR AND JACOBIAN}(x, p, z)$

4. Iterative Closest Point (ICP)

$[chi, X\_new] = \text{ICP2D}(x, P, Z)$ ⟶ 1 iteration

$X \leftarrow \emptyset$

$H \leftarrow \emptyset_{3\times3}$

$b \leftarrow \emptyset_{3\times1}$

For $i = 1 : \#\text{measurements}$

  $[e, J] = \text{ERROR AND JACOBIAN}(x, p, z)$

  $H \leftarrow H + J^T \cdot J$

  $b \leftarrow b + J^T \cdot e$

  $X \leftarrow X + e^T \cdot e$

END

$dx \leftarrow -H \setminus b$

$X\_new \leftarrow X + dx$

$\theta \leftarrow x\_new(3)$

$x\_new(3) \leftarrow \text{atan2}(\sin(\theta), \cos(\theta))$

END

assumed to be an identity matrix, given no knowledge on the measurement's uncertainty

$\longrightarrow X = \sum_i e^{[i]T} \Omega^{[i]} e^{[i]}$

⇓

CUMULATIVE ERROR

tells us how far we are from the optimum

ERROR SHOULD BE $\emptyset$

⇑

→ set point to same location

→ set my point close to same location

⟶ → set initial guess poses

Observations: ⟶ How to test?

- Spawn a set of random points in 2D
- define the location of the robot
- Compute synthetic measurements from that location

Example: ICP 2D Bearing Only

1. State (dimensions the same)

$x \in SE(2)$

$x = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$ ⟶ $\Delta x = \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{pmatrix}$

needed this euclidean parametrization due to standard ICP update

being $X += \Delta X$

⊕

$F(x+\Delta x) \cong \Delta x^T H \Delta x + 2b^T \Delta x + c$

2. Measurement

$z \in \mathbb{R}$

$z = \theta$ ⟶ $h_{ICP}^{[i]}(x) = R(\theta) p^{[i]} + t$

$h_{Bearing\_only}^{[i]}(x) = \text{atan2}\left(h_{ICP}^{[i]}(x)_y, h_{ICP}^{[i]}(x)_x\right)$

3. Jacobian

$$e^{[:1]}(x) = h^{[:]}_{\text{BEARING-ONLY}}(x) - z^{[:]}_{\text{BEARING}} \longrightarrow \frac{\partial e^{[:1]}(x)}{\partial x} = \frac{\partial h^{[:2]}_{\text{BEARING-ONLY}}(x)}{\partial x}$$

↑ also does not depend on the state

$$= \frac{\partial h_{\text{BEARING-ONLY}}}{\partial y}\bigg|_{y = h^{[:1]}_{\text{LASER}}(x)} \cdot \frac{\partial h^{[:1]}_{\text{LASER}}(x)}{\partial x}\bigg|_{x = x^k}$$

↓ REMAINS THE SAME

$$\frac{\partial h_{\text{BEARING-ONLY}}}{\partial y} = \left(\frac{-y}{x^2+y^2}, \frac{x}{x^2+y^2} \right) =$$

$$= \frac{1}{\rho^T \cdot \rho} \cdot \left(-\rho(2), \rho(1) \right)$$

4. Bearing-only ICP → only changes the T and z_hat (in measurement prediction)

only observable the angle when observing a point (from a certain unknown position) ≡ Relative angle between Robot and point

Final Observations

Why just Gauss-Newton?

- usually, works well w/ good initial guess and semi-convex cost functions (1 global minimum)
- even though is considered a 1st-order algorithm, the fact of using the quadratic-form approximation makes almost 2nd-order algorithm
- Levenberg-Marquardt interpolates between [Gauss-Newton, Gradient Descent] ⇒ also known as damped least-squares

usually, more intended for more complex problems

Least Squares Application: Odometry Calibration

Algorithm (One Iteration)

$H \leftarrow \emptyset$
$b \leftarrow \emptyset$

For each measurement:

$$e^{[i]} \leftarrow h^{[i]}(x^*) - z^{[i]}$$

$$J^{[i]} \leftarrow \left.\frac{\partial e^{[i]}(x)}{\partial x}\right|_{x=x^*}$$

$$H \leftarrow H + J^{[i]T} \cdot \Omega^{[i]} \cdot J^{[i]}$$

$$b \leftarrow b + J^{[i]T} \cdot \Omega^{[i]} \cdot e^{[i]}$$

End

$\Delta x \leftarrow \text{solve}(H \Delta x = -b)$ → linear system!

$x^* \leftarrow x^* + \Delta x$

Methodology

1. Identify the state space x
   - qualify domain
   - find locally Euclidean parametrization
2. Identify the measurement space z → what can we observe about environment or system
   - qualify domain
   - find locally Euclidean parametrization
3. Identify the prediction functions $h(x)$

Problem Formulation: Odometry Calibration

- robot moves through an environment getting odometry measurements u_i (affected by a systematic error)
- for each u_i, we have u_i^* (ground-truth) provided by us by an external sensor
- $f_i(x)$ w/ some bias parameters X, return an unbiased odometry reading u_i'

$$u_i' = f_i(x) = \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{pmatrix} \cdot u_i$$ → linear function

1. State

$X \in \mathbb{R}^9$

$X = (x_{11}\ x_{12}\ x_{13}\ x_{21}\ x_{22}\ x_{23}\ x_{31}\ x_{32}\ x_{33})^T$

↓ odometry intrinsic parameters (supposing we do not have any knowledge on the kinematic steering geometry)

2. Measurement

$u_i \in \mathbb{R}^3$

$u_i = \begin{pmatrix} u_{i,x} \\ u_{i,y} \\ u_{i,\theta} \end{pmatrix}$

$$e^{[i]}(x) = u_i^* - \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{pmatrix} \cdot u_i$$

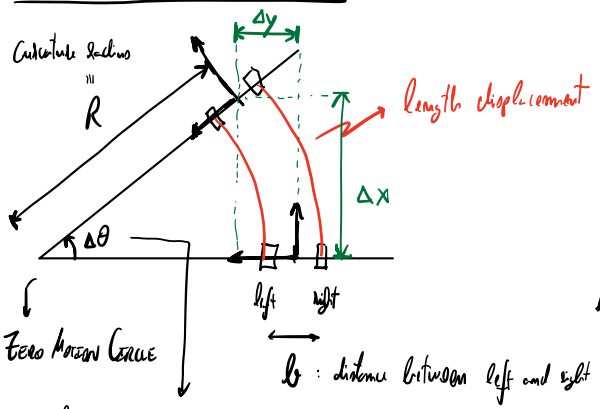
$u_i^* \equiv h^{[i]}(x)$; the matrix product term is $z^{[i]}(x)$

↑ our odometry measurement converted to an unbiased estimate (supposedly...)

$$\frac{\partial e^{[i]}(x)}{\partial x} = - \begin{pmatrix} u_i^T & & \\ & u_i^T & \\ & & u_i^T \end{pmatrix}_{3\times 9}$$

⇓ Does not depend on the state ⟶ Converges in a single iteration!

Calibrating a Unicycle (e.g., Differential Drive)



Curvature Radius = R

Zero Motion Circle

Length displacement

b : distance between left and right wheels

BASELINE

change in the attitude of the robot

$r = \Delta\theta \cdot (R + b/2)$

$l = \Delta\theta \cdot (R - b/2)$

} motion on the ground wheels

INVERT MAPPING:

$$\begin{cases} r = \Delta\theta \cdot (R + b/2) \\ l = \Delta\theta \cdot (R - b/2) \end{cases} \Leftrightarrow \begin{cases} R = \frac{r}{\Delta\theta} - \frac{b}{2} \\ R = \frac{l}{\Delta\theta} + \frac{b}{2} \end{cases} \Leftrightarrow \begin{cases} R = \frac{2r - b\cdot\Delta\theta}{2\Delta\theta} \\ R = \frac{2l + b\cdot\Delta\theta}{2\Delta\theta} \end{cases}$$

$\Delta\theta = \frac{r-l}{b}$

$$\Delta\theta = \frac{r - l}{b}$$

$$R = \frac{r + l}{2\Delta\theta}$$ — radius of the circular motion

ENCODER TO MOTOR FROM

$r^{[:]} = K_r \cdot t_r^{[:]}$ → encoder ticks

$l^{[:]} = K_l \cdot t_l^{[:]}$

$$\Delta\theta^{[:]} = \frac{r^{[:]} - l^{[:]}}{b}$$

$$\Delta x^{[:]} = \frac{r^{[:]} + l^{[:]}}{2} \cdot \frac{\sin(\Delta\theta^{[:]})}{\Delta\theta^{[:]}}$$

$$\Delta y^{[:]} = \frac{r^{[:]} + l^{[:]}}{2} \cdot \frac{1 - \cos(\Delta\theta^{[:]})}{\Delta\theta^{[:]}}$$

(real model of a robot)

EXACT INTEGRATION

$\Delta x = R \cdot \sin(\Delta\theta)$

$\Delta y = R \cdot (1 - \cos(\Delta\theta))$

What if $\Delta\theta \to 0$? (singularity of the model when $\Delta\theta \to 0$)

1st ALTERNATIVE:

IF $|\Delta\theta| \leq 1e\text{-}6$

$\Delta x^{[:]} = \frac{r^{[:]} + l^{[:]}}{2}$

$\Delta y^{[:]} = 0$

ELSE

...

END

2nd ALTERNATIVE: if $\Delta\theta$ small ⇒ Taylor expansion

$$P_x(\theta)$$

$$\frac{\sin\theta}{\theta} \cong 1 - \frac{\theta^2}{6} + \frac{\theta^4}{120} - \frac{\theta^6}{5040} \dots$$

$$\frac{1-\cos\theta}{\theta} \cong 0 + \frac{\theta}{2} - \frac{\theta^3}{24} + \frac{\theta^5}{720} \dots$$

$$P_y(\theta)$$

$\Delta_+^{[:]} = r^{[:]} + l^{[:]}$

$\Delta_-^{[:]} = r^{[:]} - l^{[:]}$

$\Delta\theta^{[:]} = \frac{\Delta_-^{[:]}}{b}$

$\Delta x^{[:]} = \frac{\Delta_+^{[:]}}{2} P_x(\Delta\theta^{[:]})$

$\Delta y^{[:]} = \frac{\Delta_+^{[:]}}{2} P_y(\Delta\theta^{[:]})$

indeed, NOT PHYSICAL (only due to the mathematical formulation of the model)

manifesting the singularity of the unicycle model

usually, up to 4th-order is enough

most popular is the simplest model, when:

$$\Delta x^{[:]} = \frac{r^{[:]} + l^{[:]}}{2} \cdot \frac{\sin(\Delta\theta^{[:]})}{\Delta\theta^{[:]}} \cong \frac{r^{[:]} + l^{[:]}}{2}$$

$$\Delta y^{[:]} = \frac{r^{[:]} + l^{[:]}}{2} \cdot \frac{1 - \cos(\Delta\theta^{[:]})}{\Delta\theta^{[:]}} \cong 0$$

ICP Formulation

1. State

$$X = \begin{pmatrix} K_R \\ K_L \\ b \end{pmatrix} \in \mathbb{R}^3$$

$\Updownarrow$

encoder to meter factors & baseline

$$\Delta\theta^{[i]} = \frac{K_R \cdot t_R^{[i]} - K_L \cdot t_L^{[i]}}{b}$$

$\Downarrow$

$$\frac{\partial \Delta\theta}{\partial X} = \begin{pmatrix} \frac{t_R^{[i]}}{b} & -\frac{t_L^{[i]}}{b} & -\frac{K_R \cdot t_R^{[i]} - K_L \cdot t_L^{[i]}}{b^2} \end{pmatrix}$$

$$\frac{\partial P_x}{\partial \Delta\theta} = -\frac{\theta}{3} + \frac{\theta^3}{30} - \frac{\theta^5}{840}$$

$$\frac{\partial P_y}{\partial \Delta\theta} = \frac{1}{2} - \frac{\theta^2}{8} + \frac{\theta^4}{144}$$

2. Measurement — encoder ticks (left and right wheel)

$$\mu_i^* = \begin{pmatrix} \mu_{i,x}^* \\ \mu_{i,y}^* \\ \mu_{i,\theta}^* \end{pmatrix}$$

$\Downarrow$

ground-truth measurements given by an external system

$$e_i^{[i]} = \mu_i^* - \begin{pmatrix} \frac{\Delta_+^{[i]}}{2} \cdot P_x(\Delta\theta^{[i]}) \\ \frac{\Delta_+^{[i]}}{2} \cdot P_y(\Delta\theta^{[i]}) \\ \Delta\theta^{[i]} \end{pmatrix}$$

$$\Delta\theta^{[i]} = \frac{\lambda^{[i]} - \beta^{[i]}}{b} = \frac{K_R \theta_R^{[i]} - K_L \theta_L^{[i]}}{b}$$

$$\Delta_+^{[i]} = \lambda^{[i]} + \beta^{[i]} = K_R t_R^{[i]} + K_L \theta_L^{[i]}$$

$$\Delta_-^{[i]} = \lambda^{[i]} - \beta^{[i]} = K_R \theta_R^{[i]} - K_L t_L^{[i]}$$

$\Downarrow$

again, the chain rule!

$$\frac{\partial P_x(\Delta\theta^{[i]})}{\partial X} = \left.\frac{\partial P_x}{\partial \Delta\theta}\right|_{\Delta\theta = \Delta\theta^{[i]}} \cdot \left.\frac{\partial \Delta\theta}{\partial X}\right|_{X = X^*}$$

$\Leftarrow$

$$\frac{\partial P_y(\Delta\theta^{[i]})}{\partial X} = \left.\frac{\partial P_y}{\partial \Delta\theta}\right|_{\Delta\theta = \Delta\theta^{[i]}} \cdot \left.\frac{\partial \Delta\theta}{\partial X}\right|_{X = X^*}$$

$\oplus$

$$\frac{\partial \Delta_+}{\partial X} = \begin{pmatrix} t_R^{[i]} & t_L^{[i]} & \emptyset \end{pmatrix}$$

$$\frac{\partial \Delta_-}{\partial X} = \begin{pmatrix} t_R^{[i]} & -t_L^{[i]} & \emptyset \end{pmatrix}$$

$\swarrow$

with these equations, you are now able to compute the final Jacobian

(do not forget of multiplication derivation rule: $(a \cdot b)' = a' b + a b'$)

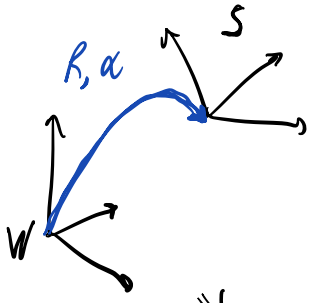
3D Point Registration

3D Example ICP orientation in 3D — find a transform that minimizes the distance between corresponding points ⟸ points + query + rough data association

State — $x \in SE(3)$

$x = (x\ y\ z\ \alpha_x\ \alpha_y\ \alpha_z)^T$

- 6D Euclidean vector
- t = translation
- α = rotation around 3 axes



Measurements — $z \in \mathbb{R}^3$ ⟶ observation from the sensor coordinate frame

$h^{[i]}(x) = R(\alpha) \cdot p^{[i]} + t$

- Prediction function
- $p^{[i]}$ = point in the world frame

$$\text{Error} - e^{[i]} = h^{[i]} - z^{[i]}$$

$$\frac{\partial e}{\partial x} = \frac{\partial h}{\partial x}$$

└ due to $z^{[i]}$ appears as a constant

⇓

$R(\alpha)$ ⟶ rotation as a composition of the rotations along $x - y - z$

$$R(\alpha) = R_x(\alpha_x)\, R_y(\alpha_y)\, R_z(\alpha_z)$$

$$R_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c & -s \\ 0 & s & c \end{pmatrix} \quad R_y = \begin{pmatrix} c & 0 & s \\ 0 & 1 & 0 \\ -s & 0 & c \end{pmatrix} \quad R_z = \begin{pmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$R_x' = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -s & -c \\ 0 & c & -s \end{pmatrix} \quad R_y' = \begin{pmatrix} -s & 0 & c \\ 0 & 0 & 0 \\ -c & 0 & -s \end{pmatrix} \quad R_z' = \begin{pmatrix} -s & -c & 0 \\ c & -s & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

⇓

$$h^{[i]}(x) = R_x(\alpha_x)\, R_y(\alpha_y)\, R_z(\alpha_z) \cdot p^{[i]} + t$$

$$\frac{\partial h^{[i]}(x)}{\partial x} = \begin{pmatrix} \frac{\partial h^{[i]}(x)}{\partial t} & \frac{\partial h^{[i]}(x)}{\partial \alpha_x} & \frac{\partial h^{[i]}(x)}{\partial \alpha_y} & \frac{\partial h^{[i]}(x)}{\partial \alpha_z} \end{pmatrix}$$

$$\frac{\partial h^{[i]}}{\partial t} = I$$

$$\frac{\partial h^{[i]}}{\partial \alpha_x} = R_x' R_y R_z\, p^{[i]}$$

$$\frac{\partial h^{[i]}}{\partial \alpha_y} = R_x R_y' R_z\, p^{[i]}$$

$$\frac{\partial h^{[i]}}{\partial \alpha_z} = R_x R_y R_z'\, p^{[i]}$$

⇓

Least-Squares exactly the same way!

$$H \mathrel{+}= J^T \cdot J$$

$$b \mathrel{+}= J^T \cdot e$$

$$\chi \mathrel{+}= e^T \cdot e$$

— in the example, removed the information matrix for clarity / simplicity

$$J^T \cdot \underbrace{I}_{\Omega} \cdot J$$

how to test it?

1. start by the initial guess = ground-truth → error should be $\emptyset$
2. then, vary the initial guess and/or measurement noise

ICP Linear Regression

How to approach the problem without iteration?

ICP is made in 1 iteration when $\frac{\partial e}{\partial x}$ = CONSTANT ! (- converges in 1 iteration; - Jacobian does not change between iterations)

In the case of 3D registration, rotations in prediction function make Jacobian dependant on the initial guess.

⇓

We can relax the constraint that R is a rotation matrix (MANIFOLD → not Euclidean)

$$R(\alpha) \cong \begin{pmatrix} r_1 & r_2 & r_3 \end{pmatrix}$$

Over-parametrization of the state ! (rotation matrix ≈ 12 parameters)

$$x^T = \begin{pmatrix} r_1^T & r_2^T & r_3^T & t^T \end{pmatrix}$$

$$h^{[i]}(x) = R\rho^{[i]} + t = \begin{pmatrix} r_1^T \\ r_2^T \\ r_3^T \end{pmatrix} \rho^{[i]} + t = \underbrace{\begin{pmatrix} \rho^{[i]T} & & \\ & \rho^{[i]T} & \\ & & \rho^{[i]T} \end{pmatrix}}_{M^{[i]}} \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix} + t$$

$r_j^T \rho^{[i]}$: NOT LINEAR → CAN BE TRANSPOSED ⇓

$$x^T = \begin{pmatrix} r_1^T & r_2^T & r_3^T & t^T \end{pmatrix}$$

$$h^{[i]}(x) = M^{[i]} \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix} + t$$

$$\frac{\partial h^{[i]}(x)}{\partial x} = \begin{pmatrix} M^{[i]} & | & I \end{pmatrix}$$

3×12 matrix NOT DEPENDING on x ↓ minimum found in 1 iteration of least squares

if initial transform is IDENTITY ⊕ $\Omega = I$

$$H = \begin{pmatrix} \sum_i M^{[i]T} M^{[i]} & \sum_i M^{[i]T} \\ \sum_i M^{[i]} & \sum_i I \end{pmatrix}$$

$$b = \sum_i \begin{pmatrix} M^{[i]} & | & I \end{pmatrix}^T \cdot \left(h^{[i]}(x) - z^{[i]} \right)$$

→ 1 iteration finds 12 parameters that minimize the error

in terms of basis of rotation matrix

BUT

• R_1, R_2, R_3 are not orthonormal !

How to find a rotation matrix "close" to the solution reported by the Least-squares ?

SVD decomposition
(singular value decomposition)

linear part of the affine transform returned by least squares → $A = \begin{pmatrix} R_1^T \\ R_2^T \\ R_3^T \end{pmatrix} = U D V^T$

D: diagonal matrix (if identity, A is a rotation matrix)

U, V: orthonormal matrices

rotation matrix "close" to A (if D close to I) → $R = UV^T$

if D "far" from I,
solution reported through linear regression not good !

• noise
&
• data association (outliers)

→ may delay solution by linear relaxations !

→ the condition of R matrix may be checked by inspecting the diagonal matrix

→ few further iterations of non-linear least squares are usually beneficial to refine the solution

Robust Estimators

Gauss-Newton

$$F(x) = \sum_i e^{[i]}(x)$$

$$e^{[i]}(x) = e^{[i]}(x)^T \cdot \Omega^{[i]} \cdot e^{[i]}(x) = \left\| e^{[i]}(x) \right\|^2_{\Omega^{[i]}} \longrightarrow \text{Omega - L2 norm}$$

$$\int \|v\|^2_\Omega = v^T \Omega v$$

squared norm of the vector, modulated by a weight matrix Ω

⇓

GN find the x that minimizes the sum of the squared L2 Omega norm of the errors:

- L2-norm assigns to each term a cost that is QUADRATIC in the magnitude of the error vectors ⟶ outliers will pull away the optimal solution quadratically (overweight correctness of the solution ⇒ ↑↑ error)
- a linear transformation of a vector that assigns an omega-norm or a plain squared norm (≡ Euclidean distance)

$\Omega = L L^T$ (Cholesky Decomposition) ← positive definite

$$\|v\|^2_\Omega = v^T \Lambda v = v^T L L^T v = \|v\|^2 \quad \text{(L2 norm)}$$

($v' = L^T v$)

What about L1 norm?

- grows linearly w/ length of a vector
- $\|v\|_\Omega = \sqrt{v^T \Omega v}$

⟶ might present some benefits (does "not" quadratically overweight wrong measurements, that may hinder fits)

BUT INVARIANCES of Gaussian no longer .....

Outliers

⟶ in general, some associations will not be correct ⊕ apriori no knowledge which associations are correct!

2 strategies:

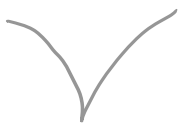
- Consensus (e.g., RANSAC) — find larger set of measurements that simultaneously agree with a solution
- M estimators — assume initial guess is good enough and consider "less" the large error terms

M ESTIMATORS — minimize $F(x) = \sum_i \rho\left(\underbrace{\left\| e^{[i]}(x) \right\|_{\Omega^{[i]}}}_{u^{[i]}(x)} \right)$

ROBUST KERNEL

↓

- ⟶ $\rho(u)$ is a SCALAR MONOTONICALLY INCREASING function that maps L1-Omega norm of the error to a scalar
- ⟶ if $\rho(u) = u^2$ ⇒ plain GN algorithm

	$f(r)$ $\longrightarrow$ $\rho = f$, $u = r$	stylized functions	$w(r)$ ⇝ inverse functions to standard cost-functions ?
L^2	$\frac{1}{2} r^2$		1
L^1	$\lvert r \rvert$		$\frac{1}{\lvert r \rvert}$
L^2 thrne (= TRUNCATED)	$\begin{cases} \frac{1}{2} r^2 & , \text{IF } \lvert r \rvert \le k \\ k^2/2 & , \text{OTHERWISE} \end{cases}$		$\begin{cases} 1 & , \text{IF } \lvert r \rvert \le k \\ \varnothing & , \text{OTHERWISE} \end{cases}$
L^1 thrne	$\begin{cases} \lvert r \rvert & , \text{IF } \lvert r \rvert \le k \\ k & , \text{OTHERWISE} \end{cases}$		$\begin{cases} 1/\lvert r \rvert & , \text{IF } \lvert r \rvert \le k \\ \varnothing & , \text{OTHERWISE} \end{cases}$
Huber	$\begin{cases} r^2/2 & , \text{IF } r \le k \\ k(\lvert r - k/2 \rvert) & , \text{OTHERWISE} \end{cases}$		$\begin{cases} 1 & , \text{IF } r \le k \\ k/\lvert r \rvert & , \text{OTHERWISE} \end{cases}$
Tukey	$\begin{cases} k^2/6 \cdot \left(1 - \left(1 - \left(r/k\right)^2\right)^3\right) & , \text{IF } r \le k \\ k^2/6 & , \text{OTHERWISE} \end{cases}$		$\begin{cases} \left(1 - \left(r/k\right)^2\right)^2 & , \text{IF } r \le k \\ \varnothing & , \text{OTHERWISE} \end{cases}$
Cauchy	$\frac{k^2}{2} \cdot \log\left(1 + \left(r/k\right)^2\right)$		$\frac{1}{1 + \left(r/k\right)^2}$
Geman-McClure	$\frac{r^2/2}{1 + r^2}$		$\frac{1}{\left(1 + r^2\right)^2}$

$\Downarrow$

How to interpolate arbitrary robust cost functions into GN?

$$\frac{\partial \lVert e(x) \rVert_\Omega^2}{\partial x} = 2 \cdot e^T(x) \Omega \frac{\partial e(x)}{\partial x} \longrightarrow \text{to minimize a function we need to set its derivation to } \varnothing !$$

$$\frac{\partial \lVert e(x) \rVert_\Omega^2}{\partial x} \cdot \frac{\partial}{\partial x}\left(e(\omega)^T \Omega e(\omega)\right) \cdot \frac{\partial}{\partial x}\left(e(\omega)\right) \cdot \text{product rule} \cdot \left(\frac{\partial e(\omega)}{\partial x}\right)^T \Omega e(\omega) + e(\omega)^T \Omega \frac{\partial e(\omega)}{\partial x} \cdot x^T \Omega x \cdot (x^T \Omega x)^{-1} \cdot e^T(\omega) \Omega \frac{\partial e(\omega)}{\partial x} + e^T(\omega) \Omega \frac{\partial e(\omega)}{\partial x} \cdot 2\, e^T(\omega) \Omega \frac{\partial e(\omega)}{\partial x}$$

$\Downarrow$

$$\frac{\partial \lVert e(x) \rVert_\Omega}{\partial x} = \frac{\partial \sqrt{\lVert e(x) \rVert_\Omega^2}}{\partial x} = \frac{1}{2 \lVert e(x) \rVert_\Omega} \cdot 2 e^T(x) \Omega \frac{\partial e(x)}{\partial x} = \frac{1}{u(x)} \cdot \frac{\partial \lVert e(x) \rVert_\Omega^2}{\partial x}$$

Scalar (inverse of L1-norm) ⊗ derivative of the L2-norm

$$\frac{\partial \rho(u(x))}{\partial x} = \frac{\partial \rho(u)}{\partial u}\bigg|_{u=u(x)} \cdot \frac{1}{u(x)} \frac{\partial \|e(x)\|_\Omega^2}{\partial x}$$

$$= \frac{\partial \rho(u)}{\partial u}\bigg|_{u=u(x)} \cdot \frac{1}{u(x)} e^T(x)\,\Omega\,\frac{\partial e(x)}{\partial x}$$

→ derivative of the error weighted by the cost function (robustified error)

$\gamma(x)$ → incorporates "all the wiggly things" that multiply the derivative of L2-norm (just a scaling factor)

$$\frac{\partial \rho(u(x))}{\partial x} = \gamma(x)\, e^T(x)\,\Omega\,\frac{\partial e(x)}{\partial x}$$

$$\frac{\partial \|e(x)\|_\Omega^2}{\partial x} = 2\, e^T(x)\,\Omega\,\frac{\partial e(x)}{\partial x}$$

- → scaling factor in the Ω matrix
- → changed at each iteration, depending on u ⇒ solved in the feed back algorithm in iterative

Robust GN (1 iteration)

$H \leftarrow 0$
$b \leftarrow 0$

FOR EACH MEASUREMENT
- $e^{[i]} \leftarrow h^{[i]}(x^*) - z^{[i]}$
- $J^{[i]} \leftarrow \frac{\partial e^{[i]}(x)}{\partial x}\Big|_{x=x^*}$
- $u^{[i]} \leftarrow \sqrt{e^{[i]T} e^{[i]}}$
- $\gamma^{[i]} \leftarrow \frac{1}{u^{[i]}} \frac{\partial \rho(u)}{\partial u}\Big|_{u=u^{[i]}}$
- $H \leftarrow H + J^{[i]T} \gamma^{[i]} \Omega^{[i]} J^{[i]}$
- $b \leftarrow b + J^{[i]T} \gamma^{[i]} \Omega^{[i]} e^{[i]}$

END

$\Delta x \leftarrow \text{solve}\ (H \Delta x = -b)$

$x^* \leftarrow x^* + \Delta x$

RANSAC

3D registration → in reality, UNKNOWN CORRESPONDENCES ⊕ UNKNOWN INITIAL GUESS

⇓

find a transformation that minimizes the distance between corresponding points

TOO COMPLICATED IN THIS FORMULATION...

(brute force, greedy approach may be very slow...)

EXAMPLE: feature detector / extractor on images ⇒ EXTRACT POINTS OF INTEREST

⇓

DESCRIPTION OF THESE POINTS

(should be invariant to rotation, translation, scale, viewpoint, ...)

‖

vector in the multi-dimension space = f (neighborhood of interest point)

‖

mini-image around point of interest

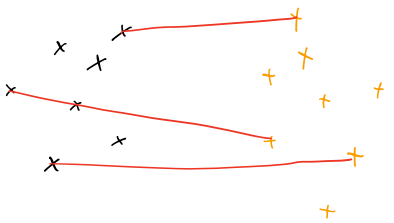
⊕ (typically) #good correspondences << #total correspondences

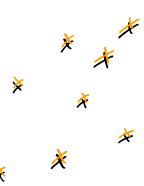
(usually, in regions where gradient noise substantially)

leads to light errors / outliers / wrong correspondences

⇓ WHAT IF ?... we know a minimal set of correct correspondences

1. find initial guess of our system by using tools learned so far (a "crude" solution)
2. w/ initial guess, find more "good" correspondences (by aligning the point clouds)
3. determine solution by considering all "good" correspondences, and dropping the bad ones





HOW MANY POINTS FOR MINIMUM SET ?

HOW TO SELECT A PAIR OF POINTS ?

↓

RANSAC (sampling approach)

⇓

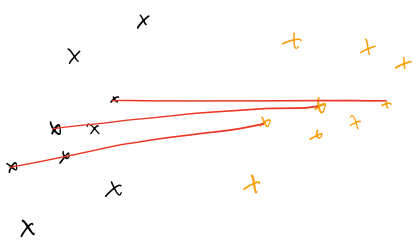
RANDOM SAMPLE CONSENSUS (RANSAC)

FOR N TIMES:

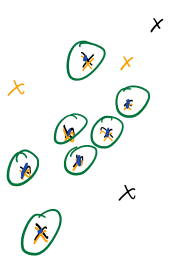
1. Sample (randomly ?...) a minimal set of correspondences among the candidate ones
2. w/ minimal set, compute an initial alignment
3. use alignment to determine the number of good / bad correspondences
4. compute the "consensus" of the guess as a function of number of inliers and error
5. repeat above steps N times, and each time keep "best" solution

↳ Completely Randomly ?... or exploit invariance of the data...

e.g. point cloud invariant to distance between points



3D point registration requires minimum of 3 points ! (are sufficient for a crude estimation)



#INLIERS

ERROR

⇓

consensus = f (# inliers, error)

χ^2 correspondences

⇒ RANSAC IS A SCHEMA (not a closed algorithm)

What do we need to implement a RANSAC scheme?

- procedure to seek for correspondences (the better the correspondences, the less iterations N are needed)
- procedure to SMARTLY select a set of PSEUDO-RANDOM (worst case: uniform) set between correspondences
- procedure to compute a solution, immune to poor initial guess → e.g., in the ICP case, Linear relaxation to determine registration w/ depending on the
- procedure to count the inliers

⇓

→ Correspondences – usually done exploiting the appearance of features

- usually leads to few "good" correspondences
- correspondence search characterized by an "INLIER RATIO" $\omega = \frac{\#\text{good matches}}{\#\text{model points}}$

↓ (3D point registration example)

↳ assume "ALL" points match with all other points, so we have $N \cdot N$ correspondences (WORST CASE)
↳ inlier ratio is $1/N$ (VERY BAD)

→ Determining a solution – initial guess is poor ⇒ linear relaxation to compute the initial solution (immune from initial guess)

$$X^T = (r_1^T\ r_2^T\ r_3^T\ t^T) \qquad h^{[i]}(x) = \underbrace{\begin{pmatrix} p^{[i]} & & \\ & p^{[i]} & \\ & & p^{[i]} \end{pmatrix}}_{M^{[i]}} \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix} + t$$

$$e^{[i]}(x) = h^{[i]}(x) - z^{[i]}$$

$$\Rightarrow H = \begin{pmatrix} \sum_i M^{[i]T} M^{[i]} & \sum_i M^{[i]T} \\ \sum_i M^{[i]} & \sum_i I \end{pmatrix} \qquad b = \sum_i \begin{pmatrix} M_i^T \\ I \end{pmatrix} e^{[i]}$$

$$\Rightarrow A = \begin{pmatrix} r_1^T \\ r_2^T \\ r_3^T \end{pmatrix} = U \Sigma V^T \qquad R = V V^T$$

→ Before feeding RANSAC, we might prune correspondences (IN THE GENERATION PHASE) → instead of a completely random one

↳ in absence of reasonable appearance-based correspondence, DISTANCE IS INVARIANT IN THE SPACE!

⇓

1. select triplet of points in the world reasonably distant from each other
2. select triplet that have more or less the same distances

↳ NOT PART OF RANSAC! (simple optimization that might help us pruning wrong correspondences)

⇓

typically feature-based matching may lead ratios $\omega > 0.2$

→ selecting inliers → e.g., given 3 correspondences, "count" inliers (more inlier points should correspond to a better solution)

→ how many rounds run RANSAC?

RANSAC random process ⇒ never guaranteed that found solution is correct one ← if consistently selecting same solution (.....

⇓

What is the probability of having the correct solution?

ω : inliers ratio of the correspondences

m : # points required to compute initial solution (minimum number of needed points)

p : desired probability of success

⇓

(assuming independent selection)

ω^m ——— probability of picking m inliers

$1 - \omega^m$ ——— probability of not picking m inliers

$(1 - \omega^m)^K$ ——— probability of not picking m inliers in K rounds

$1 - p = (1 - \omega^m)^K$ ——— probability of failure

$K = \frac{\log(1-p)}{\log(1-\omega^m)}$ ——— # trial required

Example : $p = 0.8$

$m = 3$ ⇒ $K = \frac{\log(1-0.8)}{\log(1-0.1^3)} \simeq 1608$ iterations —— that is why computing first combination for initial guess

$\omega = 0.1$

Least Squares and Uncertainty

Maximum Likelihood Estimation (MLE)

$$x^* = \arg\max_x \; p(x|z)$$

$$p(x|z) = \frac{p(z|x)\,p(x)}{p(z)} \propto p(z|x) = \prod_i p(z^{[i]}|x)$$

(Bayes rules; consider a uniform distribution; $\prod_i$: independence)

measurements affected by Gaussian noise (Gaussian assumption)

$$\Rightarrow \quad p(z^{[i]}|x) = \mathcal{N}\left(z^{[i]};\, h^{[i]}(x),\, \Sigma^{[i]}\right)$$

$$\propto \exp\left(-\underbrace{\left(h^{[i]}(x) - z^{[i]}\right)^T}_{e^{[i]}(x)} \cdot \Sigma^{[i]-1} \cdot \left(h^{[i]}(x) - z^{[i]}\right)\right)$$

Ω: error function

$$\Downarrow$$

$$x^* = \arg\max_x \prod_i p(z^{[i]}|x)$$

$$= \arg\max_x \prod_i \exp\left(-e^{[i]}(x)^T \cdot \Omega^{[i]} \cdot e^{[i]}(x)\right)$$

$$= \arg\min_x \sum_i e^{[i]}(x)^T \cdot \Omega^{[i]} \cdot e^{[i]}(x)$$

Goal → derive an estimate of the uncertainty of the solution around the optimum reached by Gauss-Newton (obtain some statistics of x^*)

- chain rule to get $p(x,z)$
- Conditioning on z

$$z^{[i]*} = h^{[i]}(x^*)$$

$$h^{[i]}(x^* + \Delta x) \cong z^{[i]*} + \underbrace{\left.\frac{\partial h^{[i]}(x)}{\partial x}\right|_{x=x^*}}_{J^{[i]}} \cdot \Delta x$$

(Taylor expansion)

$$p(z^{[i]}|\Delta x + x^*) \sim \mathcal{N}\left(J^{[i]}\Delta x + z^{[i]*};\, \Omega^{[i]-1}\right)$$

(Covariance)

describes the conditional distribution of the measurement given the perturbation around the optimal state

$$\Downarrow$$

1. joint distribution over the entire measurement (stack of all possible measurements) (Joint distribution)

m: stack of all possible measurements

$$p(z \mid \Delta x + x^*) \sim \mathcal{N}\left(\mu_z, \Omega_z^{-1}\right)$$

$$\mu_z = \begin{pmatrix} J^{[1]}\Delta x + z^{[1]*} \\ J^{[2]}\Delta x + z^{[2]*} \\ \vdots \\ J^{[K]}\Delta x + z^{[K]*} \end{pmatrix} = \underbrace{\begin{pmatrix} J^{[1]} \\ J^{[2]} \\ \vdots \\ J^{[n]} \end{pmatrix}}_{J} \Delta x + \underbrace{\begin{pmatrix} z^{[1]*} \\ z^{[2]*} \\ \vdots \\ z^{[K]*} \end{pmatrix}}_{z^*}$$

→ conditional over all measurements is again a multivariate Gaussian

$$\Omega_z = \begin{pmatrix} \Omega^{[1]} & & \varnothing \\ & \Omega^{[2]} & \\ & & \ddots \\ \varnothing & & \Omega^{[K]} \end{pmatrix}$$

→ summary independence of measurements

2. joint (x, z)

We know

$$p(x) = \mathcal{N}(x; x^*, \Sigma_x)$$

$$p(z \mid x) = \mathcal{N}(z; J\underbrace{(x - x^*)}_{\Delta x} + z^*, \Sigma_z)$$

Want to compute

$$\rightarrow \quad p(x, z) = \mathcal{N}\left(\mu_{x,z}, \Sigma_{x,z}\right)$$

$$\Downarrow$$

$$\mu_{x,z} = \begin{pmatrix} x^* \\ z^* \end{pmatrix}$$

$$\Omega_{x,z} = \begin{pmatrix} J^T \Omega_z J + \Omega_x & -J^T \Omega_z \\ -\Omega_z J & \Omega_z \end{pmatrix}$$

3. Chain rule

do not know anything about my state

$$\Sigma_x = \infty \Rightarrow \Omega_x = \varnothing$$

$$p(\Delta x) = \mathcal{N}(\varnothing, \Sigma_x)$$

given that a perturbation around optimal!

(NOTE THAT WE ARE AT THE OPTIMAL)

$$\Downarrow$$

instead of working in x, we work on Δx, that has the same covariance but it is centered on $\varnothing$

$$p(\Delta x, z) \sim \mathcal{N}\left((0, z^*)^T, \Lambda_{x,z}^{-1}\right)$$

should be familiar given that is a part of kalman-update machinery

(+) no Σ_x given that was set to $\emptyset$

$$\Omega_{x,z} = \begin{pmatrix} J^T \Omega_z J & -J^T \Omega_z \\ -\Omega_z J & \Omega_z \end{pmatrix} \longrightarrow \text{joint information matrix of measurements and states}$$

$$= \begin{pmatrix} \Omega_{xx} & \Omega_{xz} \\ \Omega_{zx} & \Omega_{zz} \end{pmatrix}$$

$$\Downarrow$$

$$\Omega_{xx} = J^T \Omega_z J$$

$$= \begin{pmatrix} J^{[2]T} & J^{[2]T} \cdots & J^{[K]T} \end{pmatrix} \cdot \begin{pmatrix} \Lambda^{[2]} & & \\ & \Lambda^{[2]} & \\ & & \ddots \\ & & & \Omega^{[K]} \end{pmatrix} \cdot \begin{pmatrix} J^{[1]} \\ J^{[2]} \\ \vdots \\ J^{[K]} \end{pmatrix}$$

$$= \sum_{K=1}^{K} H^{[K]}$$

$$= H \longrightarrow \text{same matrix from kalman-update}$$

$$\Downarrow$$

$p(x|z)$ is Gaussian w/ parameters $p(x) = \mathcal{N}^{-1}(x;\ V_{x|z},\ \Omega_{x|z})$

$$\begin{cases} V_{x|z} = V_x - \Omega_{xz} \cdot z \\ \Omega_{x|z} = \Omega_{xx} \longrightarrow \text{the information of the conditional is just the } xx \text{ block} \end{cases}$$

$$\Downarrow$$

$$\mu_{x|z} = x^* + \Sigma_{xz} \Sigma_{zz}^{-1} (z - z^*) \longrightarrow \text{the further } \underset{\text{prediction}}{\text{from}} \text{ the measurement the more mean changes}$$

$$z \triangleq z^*$$

$$\mu_{x|z} \triangleq x^*$$

$$\Sigma_{x\theta} = \Omega_{xx}^{-1}$$

$$\approx H^{-1}$$

From notes

- if dimension (measurement) = 1 and dimension (state) = 10,

$$J^{T}_{10\times1}\ \Omega_{1\times1}\ J_{1\times10} = \left[\ \right]_{10\times10} \longrightarrow \text{Rank} = 1$$

↓

in the best of the cases, will qualify only 1 dimension of the Hessian

- at minimum, 10 measurements to describe the state $\longrightarrow$ BUT NOT A SUFFICIENT CONDITION !
- is it really only needed 3 points or something else ?

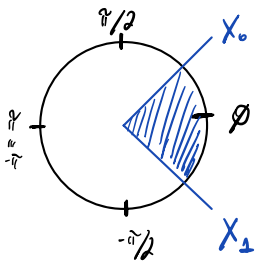
2 points ⊕ information

(e.g., 1 coordinate of 3 point)

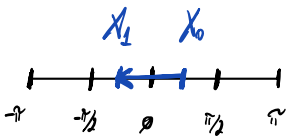
LIE: SOLVES ON MANIFOLDS → NON-EUCLIDEAN DOMAINS // euclidean = map to $\mathbb{R}^n$ and can measure distance consistent to euclidean norm

4D NON-EUCLIDEAN SPACES

2D ANGLES



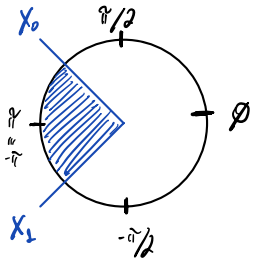
usually... → Regular subtraction

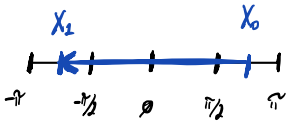


→ IS THIS APPROACH CORRECT?

Locally Euclidean parametrization
(e.g., map 2D angles in the interval $[-\pi, \pi]$)

VERSUS



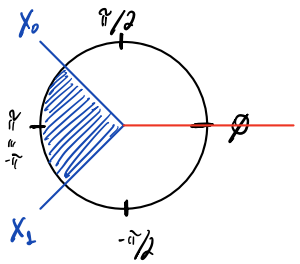


→ direction of the distance is the OPPOSITE one!
↓
→ leads to the wrong result...

↓↓

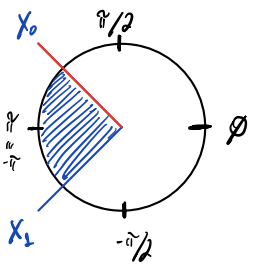
specially in the 3D problem formulation, NORMALIZATION is very tricky!

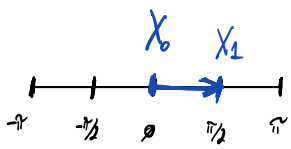
PROBLEM:



⇒ choosing this as origin of our mapping caused the invalidity of the angle difference

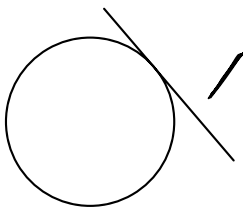
Why not putting X_0 as the origin?





makes the singularity fall away!

What about in the 3D case?



→ tangent plane would be the chart around X_0

4D COMPUTE DIFFERENCES (⊟)

X_0: start point, on manifold
X_1: end point, on manifold
Δx: difference, on chart

1. compute a chart around X_0 ($X_0 \to \emptyset$)
2. compute the location of X_1 on the chart (but the chart remains euclidean)
3. measure the difference between points in the chart

$X_0 = X_1$
$X_1 \boxminus X_0 = \Delta x$

Box-minus operator

undo the process

Applying the differences

X_0: start point, on manifold

Δx: difference, on chart

X_1: end point, on manifold reachable from X_0 by moving of Δx on the chart

→ 1. Compute Δx on chart around X_0
2. move of Δx on the chart and go back to the manifold
3. encapsulate, on manifold reachable from X_0, by moving of Δx on the chart

↳ $X_0 \boxplus \Delta x = X_1$

the formulation must ensure that $X_0 \boxplus \Delta x\big|_{\Delta x = \emptyset} = X_0$!

How to embbed the previous formulations into Least Squares?

$H \leftarrow \emptyset$

$b \leftarrow \emptyset$

→ straight-forward ?

$+ \Rightarrow \boxplus$

$- \Rightarrow \boxminus$

...

↳ NOT QUITE EXACTLY !...

FOR EACH MEASUREMENT:

$e^{[i]} \leftarrow h^{[i]}(x^*) - z^{[i]}$

$J^{[i]} \leftarrow \frac{\partial e^{[i]}(x)}{\partial x}\Big|_{x = x^*}$ ← evaluate at the current optimum

$H \leftarrow H + J^{[i]T} \cdot \Omega^{[i]} \cdot J^{[i]}$

$b \leftarrow b + J^{[i]T} \cdot \Omega^{[i]} \cdot e^{[i]}$

END

$\Delta x \leftarrow$ solve $(H \Delta x = -b)$

$x^* \leftarrow x^* + \Delta x$

↓

error function: $e^{[i]}(x) = h^{[i]}(x) \boxminus z_i$

(if space euclidean, $\boxminus \equiv -$)

→ euclidean distance !

Taylor expansion: $e^{[i]}(X \boxplus \Delta x) = e^{[i]}(x) + \underbrace{\frac{\partial e^{[i]}(X \boxplus \Delta x)}{\partial \Delta x}\Big|_{\Delta x = \emptyset}}_{J^{[i]}} \cdot \Delta x$

$e^{[i]}$

always lie on the manifold

linearization point $\equiv X$

(in 1 iteration, the origin of our chart does not have)

current optimum $\equiv X$

$\left(X \boxplus \Delta x\big|_{\Delta x = \emptyset} = X\right)$!

⊕ allows us to use the box operators mechanisms

increments: $X \leftarrow X \boxplus \Delta x$

Manifold Least Squares

$H \leftarrow \emptyset$

$b \leftarrow \emptyset$

FOR EACH MEASUREMENT:

$e^{[i]} \leftarrow h^{[i]}(x^*) \boxminus z^{[i]}$

$J^{[i]} \leftarrow \left. \frac{\partial e^{[i]}(x^* \boxplus \Delta x)}{\partial \Delta x} \right|_{\Delta x = \emptyset}$

$H \leftarrow H + J^{[i]T} \cdot \Omega^{[i]} \cdot J^{[i]}$

$b \leftarrow b + J^{[i]T} \cdot \Omega^{[i]} \cdot e^{[i]}$

END

$\Delta x \leftarrow$ solve $(H \Delta x = -b)$

$x^* \leftarrow x^* \boxplus \Delta x$

—— 2 differences:

- Jacobian by differentiating $\left. \frac{\partial e^{[i]}(x^* \boxplus \Delta x)}{\partial \Delta x} \right|_{\Delta x = \emptyset}$
- Box plus operation: $x^* \leftarrow x^* \boxplus \Delta x$

$\oplus$

X^* does not need to be a vector

can be a data structure

$\Downarrow$

only Δx needs to be a vector!

Methodology

1. State space X
   - Quality the domain
   - Define an Euclidean parameterization for the perturbation
   - Define the box plus operation ($\boxplus$)
2. Measurement space (s) z
   - Quality the domain
   - Define an Euclidean parameterization for the perturbation
   - Define the box minus operation ($\boxminus$)
3. Identify the prediction functions $h(x)$
4. Define the error functions $e(x)$

→ What really is? not how we represent it

Transformations $3D \Rightarrow SE(3)$ — 4 "coordinate" but normal constrained to unit sphere

line / fitting / plane ....

↳ is not euclidean [normal + distance of the plane to the origin; vector to origin to minimum plane ...

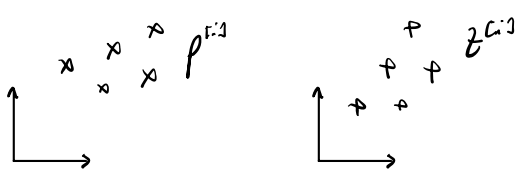
↳ euclidean parameterization for perturbation SHOULD BE MINIMUM

(e.g., minimal motions: azimuth + angle + move along the normal)

ANOTHER EXAMPLE: Lines ⟷ Lines (REGISTRATION)

- edge detection on the image
- ↓ line extraction
- We would want to register lines w/ projective line registration

Example ICP optimization in 2D



set of 2D measurements in robot frame

$\oplus$

rough known correspondences

$\Rightarrow$

Goal:

find transformation of the world w.r.t. robot coordinate frame that minimizes the distance between points

→ State

$X \in SE(2)$, $\begin{bmatrix} R & t \\ \emptyset & 1 \end{bmatrix}$ —— manifold representation vs homogeneous representation → We can represent in any form! (ANY CONVENIENT FORM)

$\Delta x = (\underbrace{\Delta x \ \Delta y}_{\Delta t} \ \Delta\theta)^T$ ⟶ euclidean representation for the chart of the PERTURBATION VECTOR
↳ if $\emptyset$, X should stay the same

$\Delta X = v2t(\Delta x) = \begin{bmatrix} \Delta R & \Delta t \\ \emptyset & 1 \end{bmatrix}$

↑ convenient function that converts a perturbation vector Δx into a matrix

⇓

$X \boxplus \Delta x = v2t(\Delta x) \cdot X$ ⟵ left multiplication → but it could be right-side multiplication (by inverting $v2t(\Delta x)$)

$= \Delta X \cdot X$ ⟶ WILL REMAIN IN THE MANIFOLD $SE(2)$!

⊕

IF $\Delta x = (\emptyset \ \emptyset \ \emptyset)^T$, $\Delta X = \begin{bmatrix} I_{2\times2} & \emptyset_{2\times1} \\ \emptyset & 1 \end{bmatrix}$, THEN $X \boxplus \Delta x \Big|_{\Delta x = \emptyset} = X$ ⟵

→ **Measurements**

$z \in \mathbb{R}^2$ ⟶ are euclidean ⇒ NO NEED TO DEFINE THE BOX MINUS OPERATOR $\boxminus$

$h^{[:]}(x) = R\,p^{[:]} + t$

$= X\,p^{[:]}$ ⟶ using a transformation matrix...

$h^{[:]}(x \boxplus \Delta x) = (x \boxplus \Delta x)\,p^{[:]}$

$= v2t(\Delta x)\, X\, p^{[:]}$

$\tilde{p}^{[:]}$ ⟶ note that in the Jacobian, the derivatives are relative to Δx and not X !

$= R(\Delta\theta)\,\tilde{p}^{[:]} + \Delta t$

→ **Jacobian**

due to z not depending on the state ...

$h^{[:]}(x \boxplus \Delta x) = R(\Delta\theta)\,\tilde{p}^{[:]} + \Delta t \quad \Longrightarrow \quad \frac{\partial e^{[:]}(x)}{\partial \Delta x} = \frac{\partial h^{[:]}(x)}{\partial \Delta x}$ ↓

⇓

$$\left.\frac{\partial h^{[:]}(x \boxplus \Delta x)}{\partial \Delta x}\right|_{\Delta x = \emptyset} = \left.\begin{pmatrix} \frac{\partial^{[:]} h(\cdot)}{\partial \Delta t} & \frac{\partial h^{[:]}(\cdot)}{\partial \Delta\theta} \end{pmatrix}\right|_{\Delta x = \emptyset}$$

$\frac{\partial h^{[:]}(\cdot)}{\partial \Delta t} = I_{2\times2}$

$$\frac{\partial \tilde{\rho}^{(i)}}{\partial \Delta\Theta} = R'(\emptyset)\,\tilde{\rho}^{(i)} = \begin{pmatrix} \emptyset & -1 \\ 1 & \emptyset \end{pmatrix} \tilde{\rho}^{[i]} = \begin{pmatrix} -\tilde{\rho}^{[i]}\cdot y \\ \tilde{\rho}^{[i]}\cdot x \end{pmatrix}$$

↓ New symmetric matrix

Uncertainty of the Solution → the manifold formulation returns a probabilistic distribution over the reconstruction vector!

H^{-1} : represents covariance of the solution around the origin of the chart

$$\Delta x \sim \mathcal{N}(\Delta x ;\ \emptyset ,\ H^{-1})$$

But, how to obtain the covariance matrix at the domain of the manifold?

$$X = X^* \boxplus \Delta x \longrightarrow X \sim g_{X^*}(\Delta x)$$

X: also a random variable ⟸ function over random variable; Δx: random variable

↓ Approximation of a Gaussian in X

→ Linearization: $J_x = \left.\frac{\partial X^* \boxplus \Delta x}{\partial \Delta x}\right|_{\Delta x = \emptyset}$ — 1st order error propagation (≡ Linearization)

$$X \sim \mathcal{N}(X ;\ X^* ,\ J_x H^{-1} J_x^T)$$

↑ note that $H = \frac{\partial e}{\partial \Delta x}$ ↺ it's in the error domain

→ Unscented Transform: $X^{(i)} = X^* \boxplus \Delta x^{(i)}$ | where $\Delta x^{(i)}$ are the sigma points extracted from Gaussian distribution in the chart

≡ project sigma points $\Delta x^{(i)}$ through the function $g_{X^*} \equiv \boxplus$ operator

Measurement Uncertainty

$$\Delta z^{[i]} \sim \mathcal{N}(\emptyset ,\ \Sigma_{\Delta z}^{[i]})$$

↑ chart perturbation

$$z^{[i]} = \mu_z^{[i]} \boxplus \Delta z^{[i]}$$

↑ measurement mean

Remap $h^{[i]}$ and $z^{[i]}$ to euclidean perturbation capable w/ information matrix

≡

→ $e^{[i]}(x) \sim \mathcal{N}(h^{[i]}(x) \boxminus \mu_z^{[i]} ,\ \Sigma_{\Delta z}^{[i]})$ — assumed to be in the manifold

Manifold Least Squares

$H \leftarrow \emptyset$

$b \leftarrow \emptyset$

FOR EACH MEASUREMENT:

$c^{[:]} \leftarrow h^{[:]}(x^*) \boxminus \mu_z^{[:]}$

$J^{[:]} \leftarrow \left. \frac{\partial e^{[:]}(x^* \boxplus \Delta x)}{\partial \Delta x} \right|_{\Delta x = \emptyset}$

$H \leftarrow H + J^{[:]T} \cdot \Omega^{[:]} \cdot J^{[:]}$

$b \leftarrow b + J^{[:]T} \cdot \Omega^{[:]} \cdot e^{[:]}$

END

$\Delta x \leftarrow \text{solve}\ (H \Delta x = -b)$

$x^* \leftarrow x^* \boxplus \Delta x$

Conclusions

- Least-squares on manifold leads to a more robust formulation of non-linear least squares on non-Euclidean spaces

- $\boxplus$, $\boxminus$ encapsulate operations on the manifolds

$\oplus$

usually, leads to easier computations (a lot of things become constant)

ICP OPTIMIZATION ON MANIFOLD (3D)

State

$$X = [R \mid t] \in SE(3)$$

$$\Delta x = (\underbrace{\Delta x, \Delta y, \Delta z}_{\Delta t} \; \underbrace{\Delta\alpha_x, \Delta\alpha_y, \Delta\alpha_z}_{\Delta\alpha})^T \in \mathbb{R}^6$$

6-dimension vector $\Rightarrow$ the manifold homogeneous matrix $(SE(3))$

$$\longrightarrow X \boxplus \Delta x = \exp(\Delta x) \cdot X = \begin{bmatrix} R(\Delta\alpha) R & \mid & R(\Delta\alpha) t + \Delta t \end{bmatrix}$$

Measurements

$z \in \mathbb{R}^3$ — Euclidean vector $\Rightarrow$ no need for box-minus operation

$$h^{[i]}(X \boxplus \Delta x) = (X \boxplus \Delta x)\, p^{[i]} = R(\Delta\alpha) \cdot \underbrace{[R p^{[i]} + t]}_{\hat{p}^{[i]}} + \Delta t$$

$$e^{[i]}(X \boxplus \Delta x) = h^{[i]}(X \boxplus \Delta x) - z^{[i]}$$

$$= R(\Delta\alpha) \cdot \hat{p}^{[i]} + \Delta t - z^{[i]}$$

Rotation Matrices $\longrightarrow R(\alpha) = R_x(\alpha_x) R_y(\alpha_y) R_z(\alpha_z)$

$$R_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_{\alpha_x} & -s_{\alpha_x} \\ 0 & s_{\alpha_x} & c_{\alpha_x} \end{pmatrix} \quad R_{\alpha_y} = \begin{pmatrix} c_{\alpha_y} & 0 & s_{\alpha_y} \\ 0 & 1 & 0 \\ -s_{\alpha_y} & 0 & c_{\alpha_y} \end{pmatrix} \quad R_{\alpha_z} = \begin{pmatrix} c_{\alpha_z} & -s_{\alpha_z} & 0 \\ s_{\alpha_z} & c_{\alpha_z} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$R_x' = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -s_{\alpha_x} & -c_{\alpha_x} \\ 0 & c_{\alpha_x} & -s_{\alpha_x} \end{pmatrix} \quad R_{\alpha_y} = \begin{pmatrix} -s_{\alpha_y} & 0 & c_{\alpha_y} \\ 0 & 0 & 0 \\ -c_{\alpha_y} & 0 & -s_{\alpha_y} \end{pmatrix} \quad R_{\alpha_z}' = \begin{pmatrix} -s_{\alpha_z} & -c_{\alpha_z} & 0 \\ c_{\alpha_z} & -s_{\alpha_z} & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$R_{x=0}' = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \quad R_{y=0}' = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} \quad R_{z=0}' = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Jacobians $\longrightarrow$ linearizing around the 0 of the chart simplifies the calculations!

$$J^{[i]} = \left.\frac{\partial e^{[i]}(X \boxplus \Delta x)}{\partial \Delta x}\right|_{\Delta x = 0}$$

$$= \begin{pmatrix} \frac{\partial e^{[i]}(\cdot)}{\partial \Delta t} & \frac{\partial e^{[i]}(\cdot)}{\partial \Delta\alpha} \end{pmatrix}$$

$$\frac{\partial \Delta t}{\partial \Delta t} = I_{3\times 3}$$

$$\Longrightarrow \quad R(\Delta\alpha) = R_x(\Delta\alpha_x) R_y(\Delta\alpha_y) R_z(\Delta\alpha_z)$$

$$= \begin{pmatrix} \frac{\partial \Delta t}{\partial \Delta t} & \frac{\partial}{\partial \Delta \alpha}\left(R(\Delta\alpha)\,\tilde{p}^{[:]}\right) \end{pmatrix}$$

note that

$$\frac{\partial R(\Delta\alpha)}{\partial \Delta\alpha_x} = R_x'(\Delta\alpha_x)\,R_y(\alpha_y)\,R_z(\Delta\alpha_z)$$

$$\frac{\partial R(\Delta\alpha)}{\partial \Delta\alpha_y} = R_x(\Delta\alpha_x)\,R_y'(\alpha_y)\,R_z(\alpha_z)$$

$$\frac{\partial R(\Delta\alpha)}{\partial \Delta\alpha_z} = R_x(\Delta\alpha_x)\,R_y(\alpha_y)\,R_z'(\Delta\alpha_z)$$

$$J^{[:]} = \left(\frac{\partial \Delta t}{\partial \Delta t} \quad \frac{\partial R_x(\Delta\alpha_x)}{\partial \Delta\alpha_x}\tilde{p}^{[:]} \quad \frac{\partial R_y(\Delta\alpha_y)}{\partial \Delta\alpha_y}\tilde{p}^{[:]} \quad \frac{\partial R_z(\Delta\alpha_z)}{\partial \Delta\alpha_z}\tilde{p}^{[:]} \right)\Bigg|_{\Delta x = \emptyset}$$

NOTE THAT J IS EVALUATED ON $\Delta x = \emptyset$!

$$R_j(\Delta\alpha_j)\Big|_{\Delta\alpha_j = \emptyset} = I_{3\times 3} \; !$$

$$\frac{\partial R(\Delta\alpha)}{\partial \Delta\alpha_j}\Big|_{\Delta\alpha_j = \emptyset} = \frac{\partial R_j(\Delta\alpha_j)}{\partial \Delta\alpha_j}\Big|_{\Delta\alpha_j = \emptyset}$$

$$= \begin{pmatrix} I_{3\times 3} & \begin{bmatrix} \emptyset & \tilde{p}^{[:]}_{\cdot z} & -\tilde{p}^{[:]}_{\cdot y} \\ -\tilde{p}^{[:]}_{\cdot z} & \emptyset & \tilde{p}^{[:]}_{\cdot x} \\ \tilde{p}^{[:]}_{\cdot y} & -\tilde{p}^{[:]}_{\cdot x} & \emptyset \end{bmatrix} \end{pmatrix}$$

$$= \begin{pmatrix} I_{3\times 3} & -\left[\tilde{p}^{[:]}\right]_\times \end{pmatrix}$$

EXAMPLE:

IF $(\chi > \text{Killobold})$
  $e \leftarrow \sqrt{\text{Killobold}} / \chi$
  $\chi \leftarrow$ Killobold
ELSE
  e remains the same
END

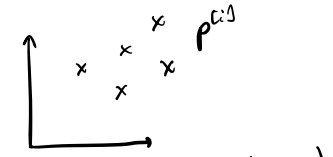
↳ OUTLIERS – wrong correspondences ⇒ ROBUST KERNELS !

↳ CONCLUSIONS
- using manifolds not necessarily makes the derivation more complex ⊕ convergence usually improves !
- using trimmed kernels might help in case of outliers at the cost of slower convergence

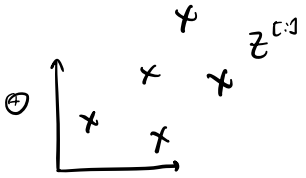
Registration on a Manifold

Standard example in Robotics: registration in a world domain (time, homogeneous transformation + scale, ...)

Similarity in $\mathbb{R}^3$



Known map (configuration of artifact) ⊕ roughly known correspondences



observed points screed! (zoom factor)

→ We want to find a transformation ($Sim3$) that minimizes distance between corresponding points

→ if we know the scale, we would be able to register the observed points (may be also possible w/ linear relaxation!)

State

$$X = \begin{pmatrix} R & t \\ \emptyset & 1 \end{pmatrix} \in Sim3$$

- X: ONLY A DATA STRUCTURE
- s: scaling factor (usually, take as 1)

→ member of group of similarities ⊕ allows us to compose the state X as regular transformations

$$\Delta x = (\underbrace{\Delta x \ \Delta y \ \Delta z}_{\Delta t} \ \underbrace{\Delta\alpha_x \ \Delta\alpha_y \ \Delta\alpha_z}_{\Delta\alpha} \ ; \ \Delta s)^T$$

$\log s$ → do not forget that $X \boxplus \Delta x\big|_{\Delta x = \emptyset} = X$!

vector to similarity

$$v2s(\Delta x) = \begin{pmatrix} R(\Delta\alpha) & \Delta t \\ \emptyset & \exp(\Delta s) \end{pmatrix}$$

↑ invert the operation so that, when $\Delta x = \emptyset$, because s the identity

in our mapping will do $\log(s)$ → usual trick in manifold

$$X \boxplus \Delta x = v2s(\Delta x) \cdot X$$

SICP: Operations

$$X = \begin{pmatrix} R & t \\ \emptyset & s \end{pmatrix} \in Sim3$$

$$X \cdot p = s R p + t$$

$$X_1 \cdot X_2 = \begin{pmatrix} R_1 & t_1 \\ \emptyset & s_1 \end{pmatrix}\begin{pmatrix} R_2 & t_2 \\ \emptyset & s_2 \end{pmatrix} = \begin{pmatrix} R_1 R_2 & R_1 t_2 + t_1 s_2 \\ \emptyset & s_1 s_2 \end{pmatrix}$$

Measurements – are just points! ⇒ no need to define the box-minus operator $\boxminus$

$z \in \mathbb{R}^3$

$h^{[:]}(x \boxplus \Delta x) = (X \boxplus \Delta x)\, p^{[:]}$.

$= \exp(\Delta s)\left(R(\Delta\alpha) \cdot \left[s\underbrace{(Rp + t)}_{\tilde{p}^{[:]}}\right] + \Delta t\right)$

→ prediction functions for the similarity elements

STEP: ERRORS AND JACOBIANS

$e^{[:]}(X \boxplus \Delta x) = h^{[:]}(x \boxplus \Delta x) - z^{[:]}$

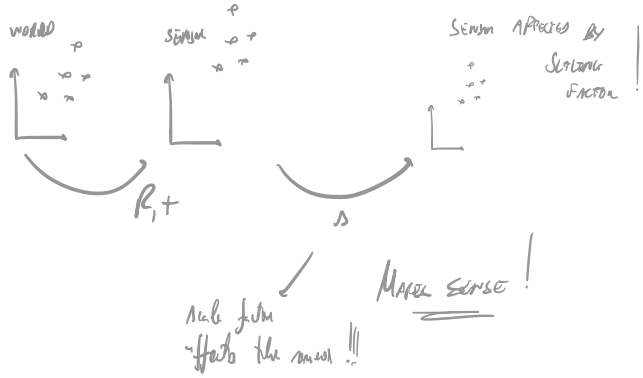
$J^{[:]} = \left.\frac{\partial e^{[:]}(X \boxplus \Delta x)}{\partial \Delta x}\right|_{\Delta x = 0}$

$= \begin{pmatrix} I_{3\times3} & -[\tilde{p}^{[:]}]_\times & \tilde{p}^{[:]} \end{pmatrix}$

— a question is why in the code they have $z_{hat} = s*(Rp+t)$?

$\exp(\Delta s)\left(R(\Delta\alpha) \cdot \left[s(Rp + t)\right] + \Delta t\right) =$

$= \exp(\Delta s)\left(R(\Delta\alpha)\, sRp + sR(\Delta\alpha)t + \Delta t\right)$



EXERCISE: N points in the space

Measurements = direction vectors of these points expressed in the reference frame of the sensor ($\tilde{=}$ bearing-only registration in $SE(3)$)

⇕ generalization of camera problems

$z \in S^2$ → we may map S^2 in $\mathbb{R}^3$ (OVER PARAMETRIZATION) $\Rightarrow z = (x\ y\ z)^T$ s.t. $\|z\| = 1$

$h^{[:]}(x) = \frac{Rp^{[:]} + t}{\|Rp^{[:]} + t\|}$ → NORMALIZED VECTOR

SUBJECT TO

with this formulation, we would not need the $\boxplus$ operator

State: $X = [R|t] \in SE(3)$ — 3D transformation of the world w.r.t. robot coordinate frame

$\Delta x = (\underbrace{\Delta x\ \Delta y\ \Delta z}_{\Delta t}\ \underbrace{\Delta\alpha_x\ \Delta\alpha_y\ \Delta\alpha_z}_{\Delta\alpha})^T \in \mathbb{R}^6$ — euclidean parametrization of the perturbation vector for the state

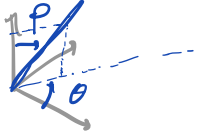
$X \boxplus \Delta x = v2t(\Delta x)\, X$ — $\begin{bmatrix} R(\Delta\alpha) & \Delta t \\ 0 & 1 \end{bmatrix}\begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R(\Delta\alpha)R & R(\Delta\alpha)t + \Delta t \\ 0 & 1 \end{bmatrix}$

$= \left[R(\Delta\alpha)R \mid R(\Delta\alpha)t + \Delta t\right]$

Measurements: Domain is a manifold (S^2 = sphere)

$z = (x, y, z)^T \in S^2$

$z_1 \boxminus z_2 : \Delta z = (\theta, \phi)$ ——— relative angle on azimuth and elevation → indeed, to represent a vector in a 3D space, we only require 2 components (minimal parametrization)



1st alternative (complicated)

1. construct a rotation matrix $R(z) = R(\theta) R(\phi)$ ——— $= R_z(\theta) R_y(\phi) = \begin{bmatrix} c_\theta & -s_\theta & 0 \\ s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} c_\phi & 0 & s_\phi \\ 0 & 1 & 0 \\ -s_\phi & 0 & c_\phi \end{bmatrix} =$

$= \begin{bmatrix} c_\theta c_\phi & -s_\theta & c_\theta s_\phi \\ s_\theta c_\phi & c_\theta & s_\theta s_\phi \\ -s_\phi & 0 & c_\phi \end{bmatrix}$

2. compute only a rotation matrix for both z_2, z_2 (assuming the original parametrization $z = (\theta, \phi) \in S^2$)
3. compute the rotation difference as $R_2^T R_1$
4. extract azimuth and elevation from rotation difference

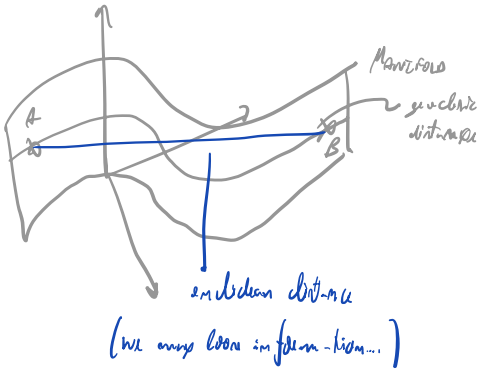
$\theta = \text{atan2}(-r_{12}, r_{22})$

$\phi = \text{atan2}(-r_{31}, r_{33})$

→ these formulas are according to the slides.....
but considering that these are derived on a single rotation, not $K_i^T K_i$ ...

2nd alternative (simplified): treat z as euclidean (but losing the benefit of operating on a geodesic distance)

replace geodesic distance with euclidean distance



(we enjoy loss in formulation...)

$\text{normalize}(v) = \frac{v}{\sqrt{v^T v}}$

$\frac{\partial \, \text{normalize}(v)}{\partial v} = \frac{v' \|v\| - v \|v\|'}{\|v\|^2}$

$(v^T v)' = \left(x^2 + y^2 + z^2\right)' = 2 v^T$

$\|v\|' = \frac{\partial}{\partial v}\left(\sqrt{v^T v}\right) = \frac{1}{2} \cdot (v^T v)' \cdot \frac{1}{\sqrt{v^T v}} = \frac{v^T}{\sqrt{v^T v}} = \frac{v^T}{\|v\|}$

$= \frac{I \|v\| - \frac{v v^T}{\|v\|}}{\|v\|^2} = \frac{1}{\|v\|} I - \frac{1}{\|v\|^3} v v^T$

$\left(\begin{matrix} x \\ y \\ z \end{matrix}\right)' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = I$!

Error function: $e^{[:]}(x) = h^{[:]}(x) - z^{[:]}$

$h^{[:]}(x) = \text{normalize}\left(\underbrace{R p^{[:]} + t}_{\tilde{p}^{[:]}}\right) = \frac{\tilde{p}^{[:]}}{\|\tilde{p}^{[:]}\|}$

$h^{[:]}(x \boxplus \Delta x) = \text{normalize}\left(\underbrace{\Delta R \, \tilde{p}^{[:]} + \Delta t}_{h_{tmp}}\right)$

Jacobians: chain rule!

$$\left.\frac{\partial e^{[i]}(x \boxplus \Delta x)}{\partial \Delta x}\right|_{\Delta x = 0} = \left.\frac{\partial \text{normalize}(\cdot)}{\partial l}\right|_{l = \tilde{p}^{[i]}} \cdot \left.\frac{\partial l_{\text{imu}}^{[i]}(x \boxplus \Delta x)}{\partial \Delta x}\right|_{\Delta x = 0}$$

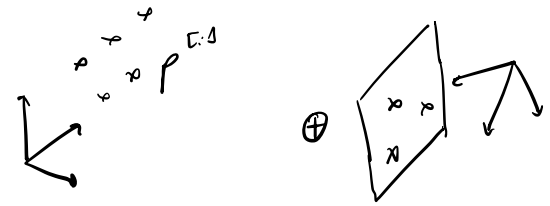
$$= \frac{1}{\|\tilde{p}^{[i]}\|^3} \cdot \left(I \cdot \|\tilde{p}^{[i]}\|^2 - \tilde{p}^{[i]} \tilde{p}^{[i]\top}\right) \cdot \left(I \quad -[\tilde{p}^{[i]}]_\times\right)$$

↓

skew matrix

(derivation of a rotation matrix is related to the skew matrix)

Realtime ICP ≡ projective registration ⇒ the focus is how to develop this type of application



Set of known points in the world frame

⊕

Set of 2D image projections of these points

w/ unknown data association !

⇒ We want to find the relative pose of the observer that minimizes the distance between prediction and measurement on the image plane

⇓

MINIMIZE OVER IMAGE PLANE ⇒ reprojection error !

⇒ How to proceed?

- constant simulation environment
- synthesis of the sensor model
- data association
- least squares
- integrate these parts

⊕ after each step do validation !

1. Simulation environment

- should be easy to debug models and show what is wrong
- to test the components of the system, convenient to create a SANDBOX ENVIRONMENT ⇒ [isolated controlled code changes / outright experimentation from production environment]
  - e.g. do not test together least squares and data association (test each model on its own)
- all we need is a set of random 3D points
  - HOWEVER, if I am working with a camera, I can recognize really easily a shape ! (INSTEAD OF RANDOM POINTS)

To render simulation more realistic.... ⇒ put points along specific patterns in the space (e.g., segments)

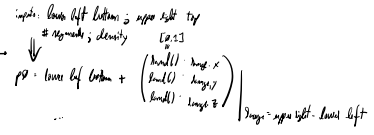
⇓

Running an edge extractor on an image

→ if we test and start to receive weird shapes and/or distortion, our camera models is wrong

point_utils.cpp:

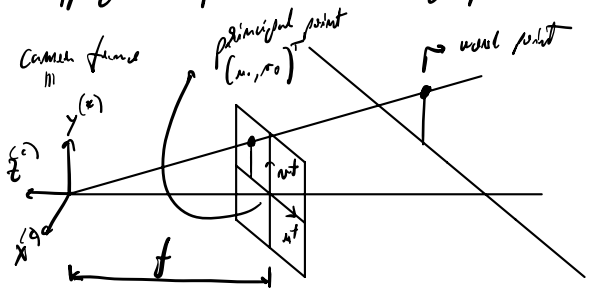
- generate points in the world (make World)
- draw points on opencv images (drawPoints)
- draw set of correspondences between 2D points (drawCorrespondences)
- ...

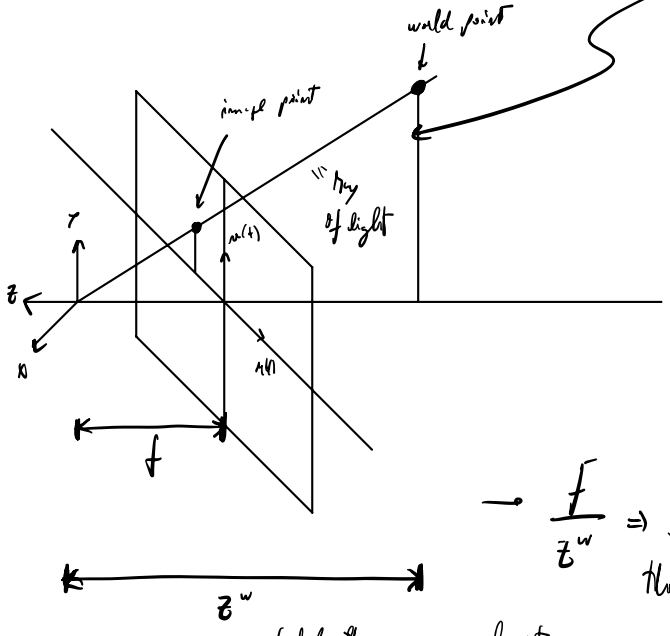


2. Sensor model (Pinhole Camera) — Camera model ≡ mapping 3D points to 2D image points

$p_w = (x \;\; y \;\; z)^T$ ——— world point

$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f\frac{x_w}{z_w} + u_0 \\ f\frac{y_w}{z_w} + v_0 \end{pmatrix}$ ——— image point ⇒





f: focal length

$(u_0, v_0)^T$: principal point / center of projection

⇓

in practice, the origin of coordinates in the image plane may not be the principal point! (that is why the offset $(u_0, v_0)^T$)

$\rightarrow \frac{f}{z^w}$ ⇒ re-scaling due to the image plane!

z^w

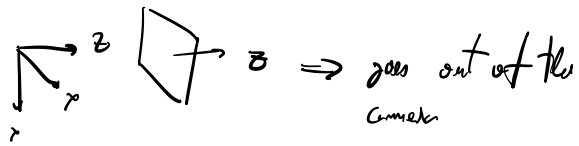
3D ⇒ 2D : ok!

2D ⇒ 3D : only ray / direction from the center that goes through the camera center (non projection)

$$\begin{pmatrix} X_{cam} \\ Y_{cam} \\ Z_{cam} \end{pmatrix} = \begin{pmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{pmatrix} \cdot p_w$$

focal length, principal point

p_{cam} — $K \equiv$ camera matrix (intrinsic parameters)



PROJECTION

$$p_{img} = p_{proj}(p_{cam}) = \begin{pmatrix} \frac{X_{cam}}{Z_{cam}} \\ \frac{Y_{cam}}{Z_{cam}} \end{pmatrix} \equiv \text{HOMOGENEOUS DIVISION}$$

$$= p_{proj}(K p_w)$$ — image point expressed with camera matrix and projection

(assuming the camera and world reference systems are the same)

However, if reference systems of the world and camera are not the same, we can compute the transformations as follows:

$$p_{img} = p_{proj}(K X p_w)$$

↑ world w.r.t. camera transformation

- in C++, we have to design a class that will implement the camera functionalities: Camera.cpp / .hpp
  - # rows, # cols → to check if point is inside image plane or not!
  - $z \leq 0$ → we cannot project points that are behind us !!!

pose of world w.r.t. camera
camera matrix
$\Longrightarrow$ compute image coordinates of a world point, according to the attributes

- testing the sensor model: `./camera_test`
  - constant simulated environment with bunch of 3D lines
  - place a camera in the world
  - move the camera with the arrow keys (W, S – move in front and backwards, a, d – rotate)
  - see the image perceived by the camera

3. Data Association — nearest neighbor strategy (if we have reasonable guess of camera pose ⊕ points sparse enough)

given a transform
X of world w.r.t. camera:
- project each point of the world onto the image
- assign to each of these points the closest measured point on the image

$O(n^2)$ w/ naive implementation $\Rightarrow$ distance map
- measurements do not change during ICP
- constant when receiving measurement
- query distance map in each iteration

⊕
- image coordinates lie on a plane
- # points potentially large

$\Downarrow$

Deriving error alignment ...

1. compute distance map on the image based on the measured points (JUST ONCE)
2. at each iteration, assign to the measured point the world point whose projection falls closest to it

- in C++, image size, image points, max_distance $\Rightarrow$
  - initialization to compute distance map
  - query for the correspondences
- to test the data association, extend `./camera_test` by showing the distance map ⊕ computed associations

4. Least-Squares

- State: $X = [R|t] \in SE(3)$

$$\Delta x = (\underbrace{\Delta x \; \Delta y \; \Delta z}_{\Delta t} \; \underbrace{\Delta\alpha_x \; \Delta\alpha_y \; \Delta\alpha_z}_{\Delta\alpha})^T$$

$$X \boxplus \Delta x = v2t(\Delta x) \cdot X$$

- Measurements (Euclidean):

$$z^{[m]} = \begin{pmatrix} u^{[m]} & v^{[m]} \end{pmatrix}^T \in \mathbb{R}^2$$

→ measurements are real $\mathbb{R}^2$ $\Rightarrow$ belong to the Euclidean space!

- Prediction: $h^{[m]}(x) = \text{proj}\left(K \underbrace{X_p^{[m]}}_{h_{ICP}^{[m]}(x)}\right) \Rightarrow p^{\nu[m]} = h_{ICP}^{[m]}(x) = X_p^{[m]}$

  $p_{cam}^{\wedge[m]} = K\, p^{\sim[m]}$

  ↑ camera coordinates (before homogeneous division)

- Error: $e^{[m,m]}(x) = h^{[m]}(x) - z^{[m]}$

  $= \text{proj}\left(K\, p^{\sim[m]}\right) - z^{[m]}$

  $e^{[m,m]}(x \boxplus \Delta x) = \text{proj}\left(K\, h_{ICP}^{[m]}(x \boxplus \Delta x)\right) - z^{[m]}$

- Jacobians: apply chain rule for derivation!

$$\left.\frac{\partial e^{[m,m]}(x \boxplus \Delta x)}{\partial \Delta x}\right|_{\Delta x = 0} = \left.\frac{\partial \text{proj}(\cdot)}{\partial p}\right|_{p = \hat{p}_{cam}^{[m]}} \cdot K \cdot \underbrace{\left.\frac{\partial h_{ICP}^{[m]}(x \boxplus \Delta x)}{\partial \Delta x}\right|_{\Delta x = 0}}_{J_{ICP}^{[m]}}$$

$$= J_{\text{proj}}\left(\hat{p}_{cam}^{[m]}\right) \cdot K \cdot J_{ICP}^{[m]}$$

where,

$$J_{ICP}^{[m]} = \begin{pmatrix} I_{3\times 3} & \left[-p^{\nu[m]}\right]_\times \end{pmatrix}$$

$$J_{\text{proj}}(p) = \begin{pmatrix} \frac{1}{z} & 0 & -\frac{x}{z^2} \\ 0 & \frac{1}{z} & -\frac{y}{z^2} \end{pmatrix}$$

ldlt ⇒ cholesky decomposition w/ diagonal

- in C++, class to implement least-squares solver:

  - world-points
  - image-points
  - current camera pose
  - Kernels threshold = ROBUST KERNEL

  ⇒ [initialization (& reset)
  single one iteration method (given the correspondences)

- How to test the least squares module? [define a problem with perfect associations!
  ./ pico-solver-test ⇒ allows keys move the camera pose but performs one single least squares iteration

(+)

[validate first the error function (if the system is at the optimum, error should be 0!)
compute first the numeric Jacobians (or use autodiff)
only if needed (for speed) compute the analytic jacobians

5. System Integration

- only when each single module has been validated!
- ./picp_complete_test
- damping on least-squares: problem not always well conditioned → damping prevents that!

↳ if even still slow, CUDA may help to speed up ....

Essentials in Computer Vision

→ Epipolar Geometry on Unit Sphere → determine up to a scale the relative transform between 2 images, given a set of lines / rays passing through the same projective center

Lines

Camera = sense directions of objects in the world (image projection of a point ≡ ray passing through the center of the camera and reaching the point on the image plane)

↓

but cannot sense distance to the sensed object (at least, not directly!)

$$l_0 = p_0 + d_0 \cdot s_0$$

where,

p_0 : offset → may be a generic point on the line

d_0 : direction → unit vector

s_0 : scalar abscissa ⇒ allows spanning the points in the line

⇓ transform by $T : R, t$

$$T = [R|t]$$

$$l_1 = T \cdot l_0 = \underbrace{R p_0 + t}_{p_1} + \underbrace{R d_0}_{d_1} \cdot s_0 \equiv$$

[transforms the point p_0 by T

direction just rotated → not subject to the translation of the transformation

⇒ However

over parametrization of the line!

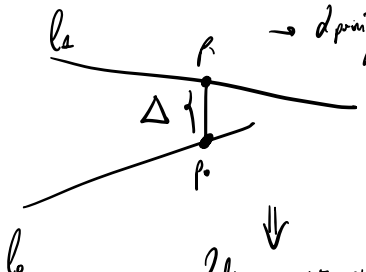
just needs 4 components (4DoF)

⇓

put 6 DoF only to be convenient

How to determine the closest point pairing of 2 lines?

- analytical ways
- least-squares machinery!



→ 2 points that form the closest distance between 2 lines

⇓

2 lines NOT NECESSARILY INTERSECTING

$$\Delta(s_0, s_1) = p_0 + d_0 s_0 - p_1 - d_1 s_1$$

$$= \underbrace{(p_0 - p_1)}_{\Delta p} + \underbrace{(d_0 \,|\, -d_1)}_{\Delta D} \cdot \underbrace{\begin{pmatrix} s_0 \\ s_1 \end{pmatrix}}_{s}$$

⇓

$$s^* = \arg\min_s \| \Delta p + \Delta D s \|^2$$

PSEUDO-INVERSE

$$\Delta p = -\Delta D s \Leftrightarrow s^* = -(\Delta D)^{\dagger} \cdot \Delta p$$

two abscissa to which form closest points between 2 lines l_0, l_1 $\equiv s^* = \begin{pmatrix} s_0^* \\ s_1^* \end{pmatrix} \Leftarrow s^* = -(\Delta D^T \cdot \Delta D)^{-1} \cdot \Delta D^T \cdot \Delta p$

Line Intersection

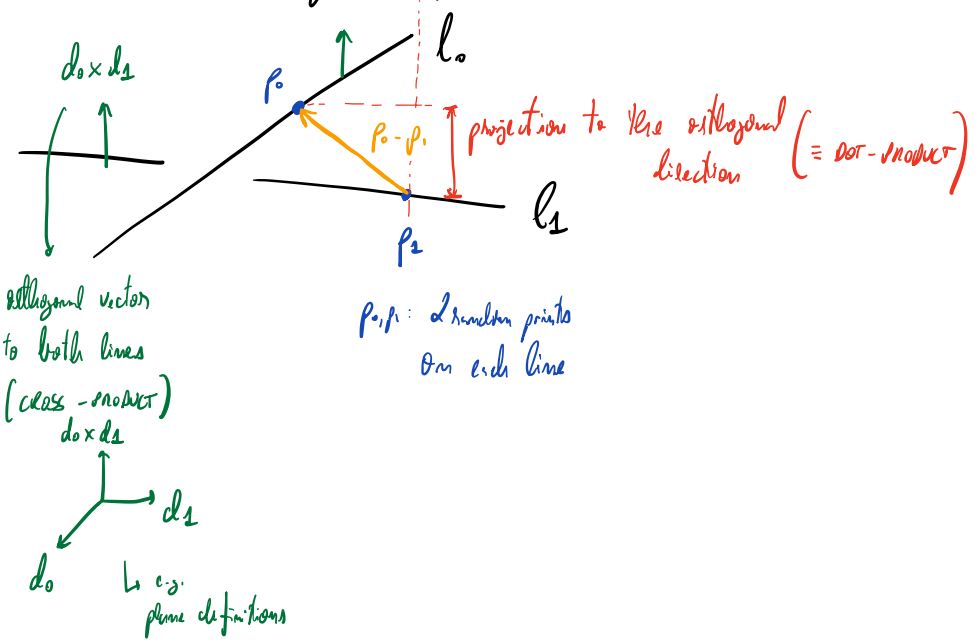
$l_0 : p_0, d_0$

$l_1 : p_1, d_1$

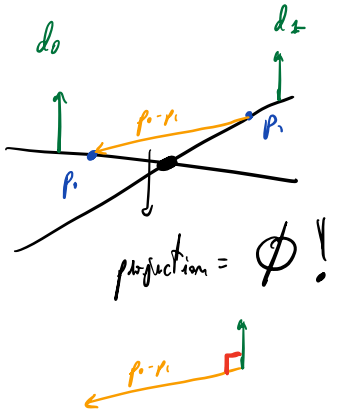
⟶ intersection condition specifies that { difference vector between any 2 points in the lines (so we choose p_0, p_1) ; along the vector $d_0 \times d_1$ orthogonal to both lines } | should be $\emptyset$

$$\emptyset = (p_0 - p_1) \cdot (d_0 \times d_1)$$ ⟶ condition verified when l_0, l_1 intersect (when parallel may also be valid, but in that case, $d_0 \times d_1 = \emptyset$ ...?)

what does it mean geometrically ?



vs



⇓

Both lines / rays pass through a camera center in $p = \emptyset$:

$l_0 : \emptyset, d_0$

$l_1 : \emptyset, d_1$

Considering the second camera being at position $T = [R|t]$ w.r.t. first camera

$l_1' : t, R d_1$ = l_1 viewed by camera 1 in the reference frame of camera $\emptyset$

⇓

$$\emptyset = t \cdot (R d_1 \times d_0)$$ ——— intersection condition

$$= d_0 \cdot (t \times (R d_1))$$ ——— cross-product identity : $a \cdot (b \times c) = c \cdot (a \times b)$

$$= d_0 \cdot [t]_\times R d_1$$ ——— cross-product to skew

$$= d_1^T (R^T [-t]_\times) d_0$$ ——— transpose and negate

E ⟷ Essential Matrix ⟶ admits ∞ solutions

(if we multiply by scalar, nonzero value)

⇓

e.g., just scales in Reconstruction

not meaningful ⇒ due to $=\emptyset$!

$\emptyset = d_1^T E d_0 \equiv$ Epipolar Constraint

d_0, d_2 : 2 directions that intersect unit sphere

⊕

Only 5 DoFs = $R \oplus t$ up to a scale

⇓ Extracting R and t from E → contains both R and t : $E = R^T [-t]_\times$, $\text{rank}(E) \leq 2$

FULL RANK ↑ ; rank ≤ 2 ↑

$= U S V^T$ s.t. $S = \begin{pmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \emptyset \end{pmatrix}$ ← 3^{rd} singular value at $\emptyset$

1. Compute Singular Value Decomposition (SVD)
2. use the scaling matrix $W = \begin{pmatrix} -1 & 1 & \\ & & 1 \end{pmatrix}$ ⟶ $WS = \begin{pmatrix} -\sigma_2 & \sigma_1 & \\ & & \sigma_3 \end{pmatrix}$
3. extract R and t by identification
4. 4 solutions for : t, $-t$, W, W^T
5. choose one by triangulating the points and pick the one w/ most points in front of the camera

$WSW^T = S$

$E = UWSW^T V^T =$

$= \underbrace{UWV^T}_{R} \cdot \underbrace{VSW^T V^T}_{[t]_\times}$

$V^T \cdot V = I$

↑

V = orthonormal matrix

"rotation"

⇓ Estimating E w/ 8 points

$E = \begin{pmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{pmatrix}$ ⟶ $e = \begin{pmatrix} e_{11} \\ e_{12} \\ e_{13} \\ \vdots \\ e_{33} \end{pmatrix}$

$d'^T E d = \emptyset$ — epipolar constraint

$e^* = \arg\min_e \sum_i \| d'^{[:]^T} E d^{[:]} \|^2$ — TRIVIAL SOLUTION : $E = \emptyset \Rightarrow$ enforce $\|e\| = 1$

norm of all elements in matrix

⇓

$d'Ed = \underbrace{\begin{pmatrix} x'x & x'y & x't & y'x & y'y & y't & t'x & t'y & t't \end{pmatrix}}_{A} \cdot \underbrace{\begin{pmatrix} e_{11} \\ e_{12} \\ \vdots \\ e_{33} \end{pmatrix}}_{e}$

$e^* = \arg\min_e \sum_i e^T A^{[:]^T} A^{[:]} e$

$$= \underset{e}{\operatorname{argmin}} \; e^T \underbrace{\left(\sum_i A^{[:,j]T} A^{[:,1]} \right)}_{H_{9\times 9}} e$$

subject to (s.t.) $e^T e = 1$ ⟶ we need to enforce this constraint!

⇓

Constrained minimization using Lagrange multipliers:

$$\mathcal{L}(x, \lambda) = f(x) + \langle \lambda, g(x) \rangle$$

↓ simple case

$$= f(x) + \lambda \cdot g(x)$$

⇓

$g(x) = 0$ is the constraint of the minimization problem!

$$\frac{\partial \mathcal{L}}{\partial x} = 0 \quad , \quad \frac{\partial \mathcal{L}}{\partial \lambda} = 0$$

or

$$\frac{\partial f(x)}{\partial x} + \lambda \cdot \frac{\partial g(x)}{\partial x} = 0 \quad \text{AND} \quad g(x) = 0$$

⇓

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

- $e^T H e$ = FUNCTION TO MINIMIZE
- $e^T e - 1$ = EQUATION OF THE CONSTRAINT, $e^T e = 1$

$$\frac{\partial \mathcal{L}(e,\lambda)}{\partial e} = 2He - 2\lambda e = 0$$ ⟶ does not this ring a bell? EIGEN VECTOR PROBLEM

$$\boxed{He = \lambda e}$$, where e is a eigenvector of H

$$\frac{\partial \mathcal{L}(e,\lambda)}{\partial \lambda} = e^T e - 1 = 0$$

choose which one makes the cost function smaller!

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

⇓

pick eigen vector that corresponds to the smallest eigen value!

- using the 8 points algorithm, we need 8 correspondences!
  - E only has 5 dofs
  - there are more complex solutions that require less points (7, 6, and 5) to compute E
  - in fact, the algorithm is commonly used to estimate the FUNDAMENTAL MATRIX F, that is computed when the camera matrix K / intrinsic parameters

$$E = (K')^T F K$$

| if 2 cameras have different intrinsic parameters

⇒ HOW TO GET T

- from set of points in the image, compute vectors (or something proportional to them) by undoing effects of camera matrix K
- estimate essential matrix from these directions
- compute 4 solutions for R and t, and for each solution, triangulate the points

⇓

choose solution w/ most admissible points
- in front of camera
- in correct fraction

↳ if points too far, we may not want to triangulate that point

PRE-CONDITIONING ⟶ e.g., scale the points around the origin ⟶ typically helps (specially, in noisy conditions)

See src/octave/tools/projective_geometry!

Multi-Point Registration

→ assume identifiable landmarks whose position is unknown (for now)
→ landmarks observed by 3D sensor from multiple positions

Goal: ⟹ • landmark locations • robot positions

State: X

$X = \{ X_R^{[1]}, \dots, X_R^{[N]}, X_l^{[1]}, \dots, X_l^{[M]} \}$ ≡ data structure

(robot) $X_R^{[n]} \in SE(3)$: $X_R^{[n]} = [R^{[n]} \mid t^{[n]}]$

(landmark) $X_l^{[m]} \in \mathbb{R}^3$: $X_l^{[m]} = \begin{pmatrix} x^{[m]} & y^{[m]} & z^{[m]} \end{pmatrix}^T$

for example

Components: vector of transformation matrices; matrix of $3 \times M$ points (in octave)

The increments are represented by a large vector containing the minimal perturbation for each state variable:

$\Delta x \in \mathbb{R}^{6N+3M}$: $\Delta x = \left(\Delta x_R^{[1]T}, \dots, \Delta x_R^{[N]T}, \Delta x_l^{[1]T}, \dots, \Delta x_l^{[M]T} \right)^T$

$\Delta X_R^{[n]} \in \mathbb{R}^6$: $\Delta x_R^{[n]T} = \underbrace{(\Delta x, \Delta y, \Delta z^{[n]}}_{\Delta t^{[n]}}, \underbrace{\Delta \alpha_x^{[n]}, \Delta \alpha_y^{[n]}, \Delta \alpha_z^{[n]}}_{\Delta \alpha^{[n]}})^T$

$\Delta x_l^{[m]} \in \mathbb{R}^3$: $\Delta x_l^{[m]T} = \underbrace{(\Delta x^{[m]} \; \Delta y^{[m]} \; \Delta z^{[m]})^T}_{\Delta t^{[m]}}$

each column is a landmark

useful helpers to access individual blocks inside state:

v_idx_pose = 1 + (pose_index - 1) × 6

v_idx_land = 1 + (num_poses) × 6 + (land_idx - 1) × 3

Boxplus operator:

$X' = X \boxplus \Delta x$ ⟵ we extend the definition of the operator for $SE(3)$ where the $\boxplus$ is adapted to be applied for the individual perturbations for each variable block

$X_R^{[n]'} = X_R^{[n]} \boxplus \Delta x = v2t(\Delta x_R^{[n]}) X_R^{[n]}$

$X_l^{[m]'} = X_l^{[m]} + \Delta x^{[m]}$

Measurements and Predictions — assuming 1 measurement ≡ 1 observation of a landmark

$z^{[n,m]} \in \mathbb{R}^3$: $z^{[n,m]} = \begin{pmatrix} x^{[n,m]} & y^{[n,m]} & z^{[n,m]} \end{pmatrix}^T$

- n: robot position in which landmark was seen
- m: observed landmark

Prediction and actual measurements similar to ICP:

$h^{[n,m]}(x) = X_R^{[n]} X_l^{[m]}$

$e^{[n,m]}(x) = X_R^{[n]} X_l^{[m]} - z^{[n,m]}$

$e^{[n,m]}(x \boxplus \Delta x) = v2t(\Delta x_R^{[n]}) X_R^{[n]} (X_l^{[m]} + \Delta x_l^{[m]}) - z^{[n,m]}$

→ only appears $\Delta x_R^{[n]}$ and $\Delta x_l^{[m]}$ due to when we perturbe other robot poses and landmarks do not affect the combination $[n, m]$!

⇩ JACOBIANS — prediction only depends on robot pose n and landmark m, mostly will be $\emptyset$

$$\left.\frac{\partial e^{[n,m]}(x \boxplus \Delta x)}{\partial \Delta x}\right| = \left(\dots \emptyset_{3\times 6} \dots \left.\frac{\partial e^{[n,m]}(x \boxplus \Delta x)}{\partial \Delta x_n^n}\right| \dots \emptyset_{3\times 6} \; \emptyset_{3\times 3} \dots \left.\frac{\partial e^{[n,m]}(x \boxplus \Delta x)}{\Delta x_l^m}\right| \dots \emptyset_{3\times 3} \right)$$

$$\tilde{z}^{[n,m]} = \chi_n^{[n]} \chi_l^{[m]}$$

$$\left.\frac{\partial e^{[n,m]}(x \boxplus \Delta x)}{\partial \Delta x_n^n}\right|_{\Delta x = \emptyset} = \underbrace{\left(I_{3\times 3} \;\middle|\; L \cdot \tilde{z}^{[n,m]} \right]_\times}_{J_n^{n,m}}$$

$$\left.\frac{\partial e^{[n,m]}(x \boxplus \Delta x)}{\partial \Delta x_l^m}\right|_{\Delta x = \emptyset} = \underbrace{R^{[n]}}_{J_l^{n,m}}$$

$$J^{[n,m]} = \left(\dots \emptyset_{3\times 6} \dots J_n^{[n,m]} \dots \emptyset_{3\times 6} \; \emptyset_{3\times 3} \dots J_l^{[n,m]} \dots \emptyset_{3\times 3} \right)$$

$$\Downarrow$$

$$J_{3\times(6\cdot N + 3\cdot M)} \Big|_{\text{column} \leftrightarrow \text{measurements to one landmarks}}$$

⇩ H MATRIX AND b VECTORS — have only few non zero blocks → we know it is sparse

$$\Downarrow$$

only modify the blocks, do not copy THE ENTIRE MATRIX !

$$H^{[n,m]} = J^{[n,m]^\top} \Omega^{[n,m]} J^{[n,m]}$$

$$= \begin{pmatrix} \dots & J_n^{[n,m]^\top} \Omega^{[n,m]} J_n^{[n,m]} & \dots & J_n^{[n,m]^\top} \Omega^{[n,m]} J_l^{[n,m]} \\ & \vdots & & \vdots \\ \dots & J_l^{[n,m]^\top} \Omega^{[n,m]} J_n^{[n,m]} & \dots & J_l^{[n,m]^\top} \Omega^{[n,m]} J_l^{[n,m]} \end{pmatrix}$$

$$b^{[n,m]} = J^{[n,m]^\top} \Omega^{[n,m]} e^{[n,m]} = \begin{pmatrix} \vdots \\ J_n^{[n,m]^\top} \Omega^{[n,m]} e^{[n,m]} \\ \vdots \\ J_l^{[n,m]^\top} \Omega^{[n,m]} e^{[n,m]} \\ \vdots \end{pmatrix}$$

Degrees of freedom → state configuration X has the same error of all states obtained by applying a rigid transform to all variables (∞ solutions)

if moving everything, would still obtain a $\emptyset$ error

↓

UNDER-DETERMINED SYSTEM

(6 DoF of a transformation)

→ eliminate redundant variables ⇒ force some variables to be locked in the origin

impose 1 robot pose as fixed →
- reduce the system by suppressing the rows and columns of the variables that stay fixed
- solve reduced system
- propagate computed perturbation to all variables that have not been eliminated

→ adding a prior ⇒ add another constraint that we force to stay where we want ($\approx$ defining the origin)

$e_{prior} = \Delta X_0^{[1]}$

$J_{prior} = I$

$H_{prior} = \Omega_{prior}$

$b_{prior} = \Omega_{\Delta X}$

$\Delta x \to 0 \Rightarrow b_{prior} \to \emptyset$

SOFT-CONSTRAINT

Example: [we are there w/ certain probability (e.g., GPS); we might receive GPS measurements along the way]

→ add to the final H a 6×6 (LARGE) information matrix in correspondence of first variable

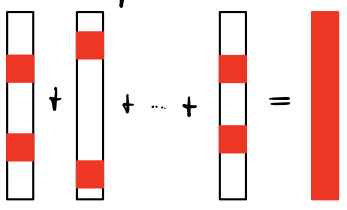
$\uparrow \Omega_{prior}$ ⇒ stiffen the constraint on the system ⇒ the more confident we are in our prior

EXAMPLE:

```
dx = zeros(system_size, 1);
H(1:pose_dim, 1:pose_dim) += eye(6) * large_value;
...
dx = -H\b;
[Xr, Xl] = boxPlus(Xr, Xl, num_poses, num_landmarks, dx);
```

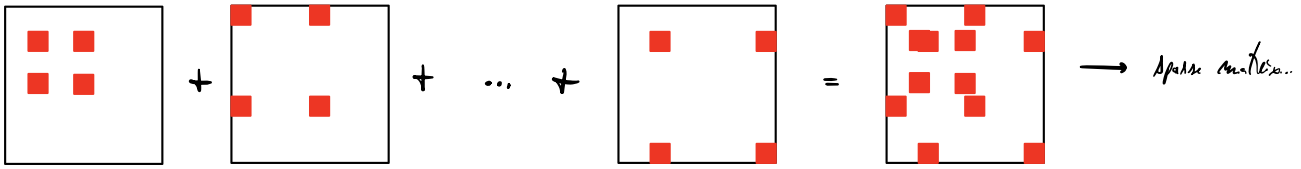
Structure of H: location of non-zero elements

$b = \sum b^{[m,m]}$



→ full diagonal matrix

$H = \sum H^{[m,n]}$



→ sparse matrix...

matrix representation of the graph!

⇓

COMPLEX MULTI-ROBOT MULTI-LAYERED MAPPING

- measurement independence leads to a sparse structure of H
- structure of H does not change during iterations
- it can be efficiently solved by using sparse methods

⇓

SPARSE IS THE KEY FOR MODERN SLAM

Multi-Pose Registration and Graph-SLAM

↳ variant of the multi-pose registration problem determining configuration of $SE(2)$ and/or $SE(3)$ configurations that are mutually observing each other

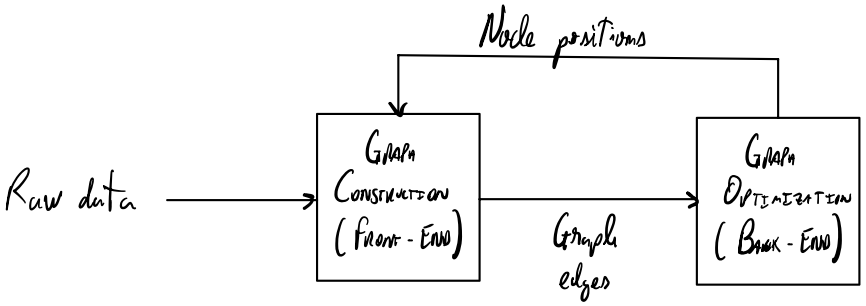
Graph-based SLAM — problem described as a graph

node = robot position ⊕ laser measurement that the robot has seen at that position

edge between 2 nodes ≡ data-dependent spatial constraint between the nodes

⇓

being able to compute relative poses of two nodes based on their measurements and optimize that by "moving" the nodes... → Solution to Graph SLAM



⇒ computes/optimizes the position of its nodes which is maximally consistent with the observations in the edges

↓ focus of this class

the front-end gives constellation of "masses" (robot positions) ⊕ "springs" (equilibrium of positions) ≡ 2 connected masses

a consistent map helps determining the new constraints by reducing the search space

Loop closure constraints: relation between 2 poses from scan matching

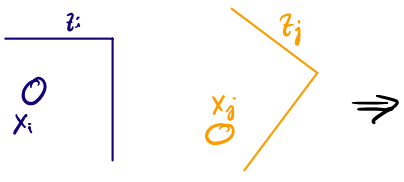
What does the Graph look like?

- n nodes, $x = x_{1:n}$ → each node x_i is a 2D or 3D transformation representing the robot position at time t_i

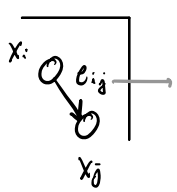
- there is a constraint e_{ij} between the node x_i and the node x_j

IF

- robot observed the same part of the environment from both x_i and x_j ⊕ via common observation, it construct an "virtual measurement" about the position x_j seen from

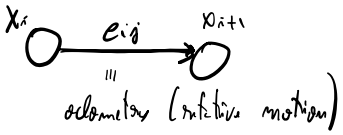


⇒

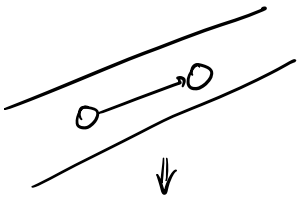


registration of the environment info of 2 poses x_i and x_j

- positions are subsequent in time and there is an odometry measurement

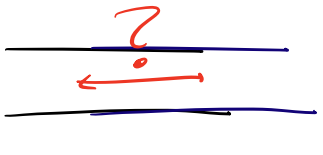


- edge information matrices - Ω_{ij} ⇒ encode the uncertainty of the edge (accounts the nature of the observations)
  - example: laser scan matching on featureless corridor



⇒ in one direction, we cannot retrieve information from scan matching

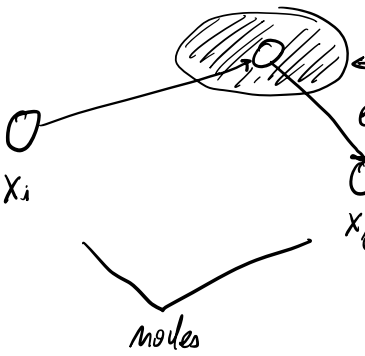
Ω_{ij} is very important in this case !



↑ certainty information

⇒ Pose Graph

$\langle z_{ij}, \lambda_{ij} \rangle$ = edge constraint ⟶ now, measurement z_{ij} becomes an object of $SE(3)$ (rotation matrix ⊕ translation vector)



GOAL: find the assignment of poses to the nodes of the graph which minimizes the negative Log-Likelihood of the observations

$$\hat{x} = \operatorname{argmin} \sum_{ij} e_{ij}^T \Omega_{ij} e_{ij}$$

→ State: collection of robot poses

$$x : x = \{x_1^{[1]}, \dots, x_n^{[N]}\}$$

$$x_n^{[n]} \in SE(3) : X^{[n]} = \left[R^{[n]} \mid t^{[n]} \right]$$

(State definition)

$$\Delta x \in \mathbb{R}^{6N} : \Delta x = \left(\Delta x_1^{[1]T}, \dots, \Delta x_n^{[N]T} \right)^T$$

$$\Delta x_n^{[n]} \in \mathbb{R}^6 : \Delta x_n^{[n]} = \left(\underbrace{\Delta x^{[n]}, \Delta y^{[n]}, \Delta z^{[n]}}_{\Delta t^{[n]}}, \underbrace{\Delta\alpha_x^{[n]}, \Delta\alpha_y^{[n]}, \Delta\alpha_z^{[n]}}_{\Delta\alpha^{[n]}} \right)^T$$

(Euclidean parameterization of the state perturbation)

$$X' = X \boxplus \Delta x$$

$$X_n^{[n]'} = X_n^{[n]} \boxplus \Delta x_n^{[n]}$$

$$= v2t\left(\Delta x_n^{[n]}\right) X_n^{[n]}$$

(Box ⊞ operator definition)

→ Measurements and Predictions

$$Z^{[i,j]} \in SE(3) \cdot \quad Z^{[i,j]} = \left[R^{[i,j]} \mid t^{[i,j]} \right]$$

↑ measurement is now an object of $SE(3)$

$$\Downarrow$$

$$l^{[i,j]}(x) = \left(X_n^{[i]}\right)^{-1} \cdot X_n^{[j]}$$

↑ pose of the robot w.r.t. world frame at time instant j

$\left(X_n^{[i]}\right)^{-1} \cdot X_n^{[j]} = H_j^i$ — relative transformation of robot pose j w.r.t. robot pose i

$\left(X_n^{[i]}\right)^{-1} = A^i_{world}$ $\quad X_n^{[j]} = H^{world}_j$

to map it back from the manifold to euclidean!

$$e^{[i,j]}(x) = \left(X_n^{[i]}\right)^{-1} \cdot X_n^{[j]} \boxminus Z^{[i,j]} = t2v\left(\left(Z^{[i,j]}\right)^{-1} \cdot \left(X_n^{[i]}\right)^{-1} \cdot X_n^{[j]}\right)$$

$Z^{[i,j]}$ measurement coordinate frame

prediction expressed in the measurement coordinate frame

relative position of the prediction relative to the measurement!

$\left(H^i_{j,estimate}\right)^{-1} \cdot H^i_j = H^{j,estimate}_i \cdot H^i_j$

$l^{[i,j]}$ — difference between transformations

$$e^{[i,j]}(x \boxplus \Delta x) = t2v\left(\left(l^{[i,j]}\right)^{-1} \cdot \left(v2t(\Delta x_n^{[i]}) \cdot X_n^{[i]}\right)^{-1} \cdot \left(v2t(\Delta x_1^{[j]}) \cdot X_n^{[j]}\right)\right) \longrightarrow \dim\left(e^{[i,j]}\right) = 6$$

→ Jacobians: prediction depends only on the observed robot poses $(x^{[i]})$ and $(x^{[j]})$ → mostly the rest will be $\emptyset$

$$\frac{\partial e^{[i,j]}(x \boxplus \Delta x)}{\partial \Delta x} = \left(\dots \emptyset_{6\times 6} \dots \frac{\partial e^{[i,j]}(x \boxplus \Delta x)}{\partial \Delta x_1^{[i]}} \dots \frac{\partial e^{[i,j]}(x \boxplus \Delta x)}{\partial \Delta x_1^{[j]}} \dots \emptyset_{6\times 6} \dots \right)$$

$$\frac{\partial e^{[i,j]}(x \boxplus \Delta x)}{\partial \Delta x_1^{[i]}}\Big|_{\Delta x_1^{[i]} = \emptyset} = J_i^{[i,j]} \qquad \frac{\partial e^{[i,j]}(x \boxplus \Delta x)}{\partial \Delta x_1^{[j]}}\Big|_{\Delta x_1^{[j]} = \emptyset} = J_j^{[i,j]}$$

$$J^{[i,j]} = \left(\dots \emptyset_{6\times 6} \dots J_i^{[i,j]} \dots \emptyset_{6\times 6} \dots J_j^{[i,j]} \dots \emptyset_{6\times 6} \dots \right)$$

not easy to compute by hand...

(problem is on the $t2v$ helper function...)

$\Downarrow$

TRANSFORMATION TO VECTOR

(mapping from the manifold back to a euclidean parameterization)

→ **Information Matrices:** measurements live on a non-Euclidean space ⇒ we need to bend information matrices

$$\hat{Z}^{[i,j]} = X_i^{[:]-1} X_j^{[:]} \qquad \text{(prediction)}$$

$$J_e^{[i,j]} = \left. \frac{\hat{Z}^{[i,j]} \boxminus Z}{\partial z} \right|_{z = Z^{[i,j]}} \qquad \text{(derivation of the error w.r.t. measurement)}$$

$$\tilde{\Omega}_{ij}^{[i,j]} \leftarrow \left(J_e^{[i,j]} \Omega^{[i,j]-1} J_e^{[i,j]\top} \right)^{-1} \equiv \text{adapted information matrix for one iteration}$$

$\Omega^{[i,j]-1}$: information matrix w/ 12 numbers (9 for rotation, 3 for translation?...)

project it back to an error with dimension 6

→ **H matrix and b vector**

$$H^{[i,j]} = J^{[i,j]\top} \tilde{\Omega}^{[i,j]} J^{[i,j]} = \begin{pmatrix} \vdots & & \vdots & \\ \cdots & J_i^{[i,j]\top} \tilde{\Omega}^{[i,j]} J_i^{[i,j]} & \cdots & J_i^{[i,j]\top} \tilde{\Omega}^{[i,j]} J_j^{[i,j]} \cdots \\ \vdots & & \vdots & \\ \cdots & J_j^{[i,j]\top} \tilde{\Omega}^{[i,j]} J_i^{[i,j]} & \cdots & J_j^{[i,j]\top} \tilde{\Omega}^{[i,j]} J_j^{[i,j]} \cdots \\ \vdots & & \vdots & \end{pmatrix}$$

$$b^{[i,j]} = J^{[i,j]\top} \tilde{\Omega}^{[i,j]} e^{[i,j]} = \begin{pmatrix} \vdots \\ J_i^{[i,j]\top} \tilde{\Omega}^{[i,j]} e^{[i,j]} \\ \vdots \\ J_j^{[i,j]\top} \tilde{\Omega}^{[i,j]} e^{[i,j]} \\ \vdots \end{pmatrix}$$

both only give a few non zero blocks...

Chordal Distance

$SE(3)$ highly non-linear (bad for the derivatives) ⟶ simplify the problem using the chordal distance:

↳ what does it mean to subtract 2 transformation matrices?

- 2 translations ≡ difference in position
- 2 rotations ≡ each rotation is 3 orthogonal vectors

⟸ 12-dimension vector

difference on the chordal distance ≡ regular minus to express difference between transformations

$$X = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \longrightarrow \text{flatten}(X) = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ t \end{pmatrix}$$

$$R = (r_1 \; r_2 \; r_3)$$

$$g^{[i,j]}(x) = \left(X_R^{[i]}\right)^{-1} \cdot X_R^{[j]}$$

$$h^{[i,j]}(x) = \text{flatten}\left(g^{[i,j]}\right)$$

$$e^{[i,j]}(x) = \text{flatten}\left(\left(X_R^{[i]}\right)^{-1} X_R^{[j]}\right) - \text{flatten}\left(Z^{[i,j]}\right) \equiv \text{Chordal Residuals and Errors}$$

$$\frac{\partial e^{[i,j]}(x \boxplus \Delta x)}{\partial \Delta x} = \left(\cdots \underset{12\times 6}{0} \cdots \frac{\partial e^{[i,j]}(x \boxplus \Delta x)}{\partial \Delta x_R^{[i]}} \cdots \underset{12\times 6}{0} \cdots \frac{\partial e^{[i,j]}(x \boxplus \Delta x)}{\partial \Delta x_R^{[j]}} \cdots \underset{12\times 6}{0} \cdots \right)$$

error now becomes 12 dimension

easier to compute w/ pen and paper

note that the number of active variables remain 6! $(\Delta x, \Delta y, \Delta z, \Delta\alpha_x, \Delta\alpha_y, \Delta\alpha_z)$

expanding the jacobians at the perturbations ...

$$e\left(X_i \boxplus \Delta x_i,\ X_j \boxplus \Delta x_j\right) = X_i^{-1}\ \nu 2 t\left(\Delta x_i\right)^{-1}\ \nu 2 t\left(\Delta x_j\right) X_j$$

$$\nu 2 t(\Delta x)^{-1} = \nu 2 t(-\Delta x), \text{ for small } \Delta x !$$

$$e\left(X_i \boxplus \Delta x_i,\ X_j \boxplus \Delta x_j\right) = \underbrace{\begin{pmatrix} R_i^T & -R_i^T t_i \\ 0 & 1 \end{pmatrix}}_{X_i^{-1}} \underbrace{\begin{pmatrix} R(-\Delta\alpha_i) & -\Delta t_i \\ 0 & 1 \end{pmatrix}}_{\nu 2 t(\Delta x_i)^{-1}} \underbrace{\begin{pmatrix} R(\Delta\alpha_j) & \Delta t_j \\ 0 & 1 \end{pmatrix}}_{\nu 2 t(\Delta x_j)} \underbrace{\begin{pmatrix} R_j & t_j \\ 0 & 1 \end{pmatrix}}_{X_j}$$

$$\Downarrow$$

the derivative is evaluated at $\Delta x = 0$!

$$\left.\frac{\partial e\left(X_i \boxplus \Delta x_i,\ X_j \boxplus \Delta x_j\right)}{\partial \Delta x_i}\right|_{\Delta x = 0} = \frac{\partial}{\partial \Delta x_i}\left(\begin{pmatrix} R_i^T & -R_i^T t_i \\ 0 & 1 \end{pmatrix}\begin{pmatrix} R(-\Delta\alpha_i) & -\Delta t_i \\ 0 & 1 \end{pmatrix}\begin{pmatrix} I_{3\times 3} & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} R_j & t_j \\ 0 & 1 \end{pmatrix}\right) =$$

$$= \frac{\partial}{\partial \Delta x_i}\left(\begin{pmatrix} R_i^T & -R_i^T t_i \\ 0 & 1 \end{pmatrix}\begin{pmatrix} R(-\Delta\alpha_i) & -\Delta t_i \\ 0 & 1 \end{pmatrix}\begin{pmatrix} R_j & t_j \\ 0 & 1 \end{pmatrix}\right)$$

$$\left.\frac{\partial e\left(X_i \boxplus \Delta x_i,\ X_j \boxplus \Delta x_j\right)}{\partial \Delta x_j}\right|_{\Delta x = 0} = \frac{\partial}{\partial \Delta x_i}\left(\begin{pmatrix} R_i^T & -R_i^T t_i \\ 0 & 1 \end{pmatrix}\begin{pmatrix} I_{3\times 3} & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} R(\Delta\alpha_j) & \Delta t_j \\ 0 & 1 \end{pmatrix}\begin{pmatrix} R_j & t_j \\ 0 & 1 \end{pmatrix}\right) =$$

$$= \frac{\partial}{\partial \Delta x_j}\left(\begin{pmatrix} R_i^T & -R_i^T t_i \\ 0 & 1 \end{pmatrix}\begin{pmatrix} R(\Delta\alpha_j) & \Delta t_j \\ 0 & 1 \end{pmatrix}\begin{pmatrix} R_j & t_j \\ 0 & 1 \end{pmatrix}\right)$$

$$\Downarrow$$

only differ the $-$!

?

(in the slides, they say $\frac{\partial e(x_i \boxplus \Delta x_i,\ x_j \boxplus \Delta x_j)}{\partial \Delta x_i} = -\frac{\partial e(x_i \boxplus \Delta x_i,\ x_j \boxplus \Delta x_j)}{\partial \Delta x_j}$)

Focusing on computing the derivative w.r.t. Δx_j, being Δx_i its opposite:

$$e\left(X_i,\ X_j \boxplus \Delta x_j\right) = \begin{bmatrix} R_i^T & -R_i^T t_i \end{bmatrix}\begin{bmatrix} R(\Delta\alpha_j) & \Delta t_j \end{bmatrix}\begin{bmatrix} R_j & t_j \end{bmatrix}$$

$$= \begin{bmatrix} R_i^T & -R_i^T t_i \end{bmatrix}\begin{bmatrix} R(\Delta\alpha_j) R_j & R(\Delta\alpha_j) t_j + \Delta t_j \end{bmatrix} =$$

Indeed, if you have made all the computations, it would verify this....

$$= \left[R_i^T R(\Delta\alpha_j) R_j \;\middle|\; R_i^T R(\Delta\alpha_j) t_j + R_i^T \Delta t_j - R_i^T t_i \right] =$$

$$= \left[R_i^T R(\Delta\alpha_j) R_j \;\middle|\; R_i^T \left(R(\Delta\alpha_j) t_j + \Delta t_j - t_i \right) \right]$$

↓

derivation of g w.r.t. Δx_j originates a 4×4 matrix $\Rightarrow$ ONLY THE FIRST 3 ROWS MATTER!

Recalling the formulations of a rotation generic matrix:

$$R(\alpha) = R_x(\alpha_x) R_y(\alpha_y) R_z(\alpha_z)$$

$$R_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c & -s \\ 0 & s & c \end{pmatrix} \qquad R_y = \begin{pmatrix} c & 0 & s \\ 0 & 1 & 0 \\ -s & 0 & c \end{pmatrix} \qquad R_z = \begin{pmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$R_x' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -s & -c \\ 0 & c & -s \end{pmatrix} \qquad R_y' = \begin{pmatrix} -s & 0 & c \\ 0 & 1 & 0 \\ -c & 0 & -s \end{pmatrix} \qquad R_z' = \begin{pmatrix} -1 & -c & 0 \\ c & -s & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$R_{x_0}' = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \qquad R_{y_0}' = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} \qquad R_{z_0}' = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

(R_{x_0}': $\alpha_x = 0$)

$$g(X_i, X_j \boxplus \Delta x_j) = \left[R_i^T R(\Delta\alpha_j) R_j \;\middle|\; R_i^T \left(R(\Delta\alpha_j) t_j + \Delta t_j - t_i \right) \right]$$

↳

$$\frac{\partial g}{\partial \Delta\alpha_x} = \left[R_i^T R_{x_0}' R_j \;\middle|\; R_i^T R_{x_0}' t_j \right] \qquad \frac{\partial g}{\partial \Delta t_x} = \left[\mathbf{0}_{3\times 3} \;\middle|\; R_i^T \cdot (1\,0\,0)^T \right]$$

$$\frac{\partial g}{\partial \Delta\alpha_y} = \left[R_i^T R_{y_0}' R_j \;\middle|\; R_i^T R_{y_0}' t_j \right] \qquad \frac{\partial g}{\partial \Delta t_y} = \left[\mathbf{0}_{3\times 3} \;\middle|\; R_i^T \cdot (0\,1\,0)^T \right]$$

$$\frac{\partial g}{\partial \Delta\alpha_z} = \left[R_i^T R_{z_0}' R_j \;\middle|\; R_i^T R_{z_0}' t_j \right] \qquad \frac{\partial g}{\partial \Delta t_z} = \left[\mathbf{0}_{3\times 3} \;\middle|\; R_i^T \cdot (0\,0\,1)^T \right]$$

⇓

$$\left. \frac{\partial h(X_i, X_j \boxplus \Delta x_j)}{\partial \Delta x_j} \right|_{\Delta x = 0} = \begin{pmatrix} dh_x & dh_y & dh_z & dh_{\alpha_x} & dh_{\alpha_y} & dh_{\alpha_z} \end{pmatrix}$$

$\text{flatten}\left(\frac{\partial e}{\partial \Delta t_x}\right)$ $\text{flatten}\left(\frac{\partial e}{\partial \Delta t_y}\right)$ ..... $\text{flatten}\left(\frac{\partial e}{\partial \Delta t_x}\right)$ .....

④

$$\left.\frac{\partial h(X_i \boxplus \Delta x_i,\ X_j)}{\partial \Delta x_i}\right|_{\Delta x = 0} = -\left.\frac{\partial h(X_i,\ X_j \boxplus \Delta x_j)}{\partial \Delta x_j}\right|_{\Delta x = 0}$$

⇓ find Jacobian w.r.t. Δx_j

$$r_{x_0}' = \text{flatten}\left(R_i^T R_{x_0}' R_j\right)$$

$$r_{y_0}' = \text{flatten}\left(R_i^T R_{y_0}' R_j\right)$$

$$r_{z_0}' = \text{flatten}\left(R_i^T R_{z_0}' R_j\right)$$

$$J_j^{[i,j]} = \begin{bmatrix} \mathbb{0}_{9\times 3} & \left[r_x' \mid r_y' \mid r_z'\right] \\ R_i^T & -R_i^T \lfloor t_j \rfloor_\times \end{bmatrix}$$

$$= \left[R_i^T \begin{pmatrix}1\\0\\0\end{pmatrix} \middle| R_i^T \cdot \begin{pmatrix}0\\1\\0\end{pmatrix} \middle| R_i^T \cdot \begin{pmatrix}0\\0\\1\end{pmatrix} \right]$$

$$R_i^T R_{x_0}' t_j = R_i^T \cdot \begin{pmatrix} 0 \\ -t_{j,z} \\ t_{j,y} \end{pmatrix}$$

$$R_i^T R_{y_0}' t_j = R_i^T \cdot \begin{pmatrix} t_{j,z} \\ 0 \\ -t_{j,x} \end{pmatrix}$$

$$R_i^T R_{z_0}' t_j = R_i^T \cdot \begin{pmatrix} -t_{j,y} \\ t_{j,x} \\ 0 \end{pmatrix}$$

⇓

$$\left[\begin{pmatrix} 0 \\ -t_{j,z} \\ t_{j,y} \end{pmatrix} \middle| \begin{pmatrix} t_{j,z} \\ 0 \\ -t_{j,x} \end{pmatrix} \middle| \begin{pmatrix} -t_{j,y} \\ t_{j,x} \\ 0 \end{pmatrix} \right] = -\lfloor t_j \rfloor_\times$$

⇒ **CONCLUSIONS** → factor graph is the combination of these slides!

- we manage to formulate the pose graph as a least-squares problem
- pose-pose constraints ≈ from a position we know the position of another robot
- USE THE CHORDAL DISTANCE for pose-pose measurements
- all considerations on sparsity and low rank made for the pose-landmark problem STILL HOLD

Factor Graphs

Large problems may involve thousand of thousand of nodes .... ⊕ their associated measurements and possible landmarks ....

examples
- pose - glyphs
- pose - landmark
- calibration
- ... (e.g. combination of one of above)

⟶ the loses the problem, the slows the solution!

all boils down to linear systems ⇒ 1000 SE(3) poses ≡ 6000 × 6000 matrices ⇒ 6000 variables

⇓ the inversion may be HELL ....

⇓

However, despite ↑ # variables, a single measurement is typically only affected by a subset of state variables (LOCALITY OF THE PROBLEM)

- landmark measurement ≈ observed position $X_R^{[m]}$ ⊕ position of the landmark $X_l^{[m]}$
- pose - pose measurement ≈ observed position $X_R^{[i]}$ ⊕ observed position $X_R^{[j]}$

⇒ **Factor Graph as a Graphical Representation** ⟶ not a method!

graphical formalization to formulate / describe a problem ⇒ Goal: highlight the dependency of a measurement from a subset of state variables

State variables : $X^T = \left(x^{[1]^T}, \dots, x^{[n]^T} \right)$

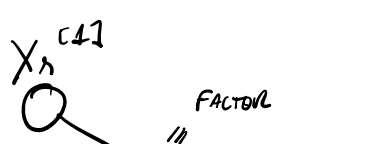
Error function for measurement $z^{[k]}$: $e^{[k]}\left(X^{[k]}\right)$ ⟶ is what we want to minimize (the prediction function h is only a means to an end)

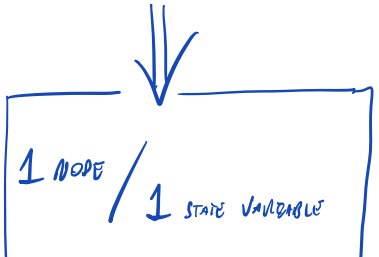
Subset of state variables affecting $z^{[k]}$: $X^{[k]^T} = \left(x^{[k_1]^T}, \dots, x^{[k_q]^T} \right) \subset \left(x^{[1]^T}, \dots, x^{[n]^T} \right)$

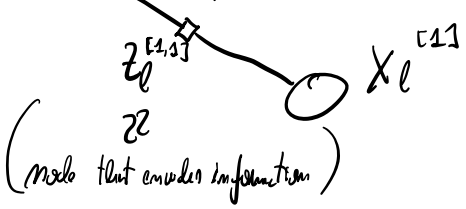
Log-likelihood : $F(x) = \sum_{k \in C} e^{[k]}\left(x^{[k]}\right)^T \cdot \Omega^{[k]} \cdot e^{[k]}\left(x^{[k]}\right)$

Optimum : $x^* = \arg\min_x F(x)$

⟶ **Pose - Landmark SLAM:**







(node that encodes information)

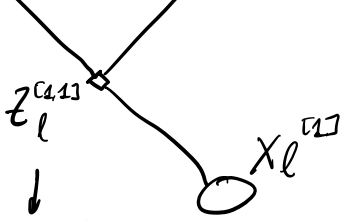
$\Downarrow$

Error function associated to measurement $z_l^{[1,1]}$ only depends on robot pose $x_r^{[2]}$ and landmark $x_l^{[1]}$

1 NODE / 1 MEASUREMENT

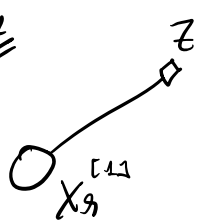
1 EDGE BETWEEN VARIABLE AND MEASUREMENT ONLY IF THEY ARE DEPENDENT

OR



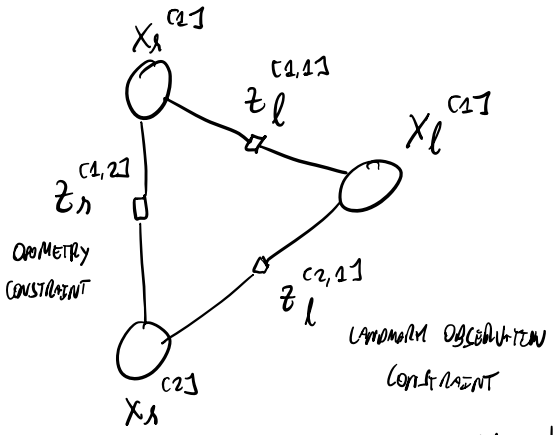
this factor now has 3 dependencies

OR



in the case of GPS, the error factor only dependent on the current robot position

→ Landmark + Odometry SLAM



⇒ FACTOR GRAPH REPRESENTATION

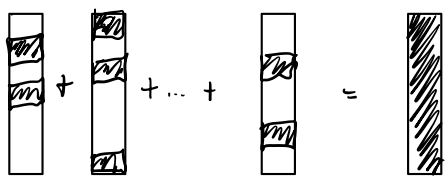
- Can represent arbitrary factored functions
- In our case, we restrict to log-likelihood
- 2 variable nodes are connected through a measurement
  factors can be seen as hyper-edges connecting multiple variables

⇒ STRUCTURE → contribution to the H matrix

contribution for a measurement $z^{[k]}$ will affect only the ith components in $x^{[k]}$ (subset)

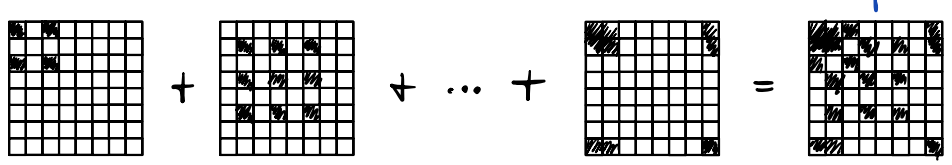
structure of H only depends on the structure of the measurements [preallocate before the iterations
do not allocate memory for 0 blocks

$$b = \sum b^{[k]}$$



→ DENSE VECTOR

$$H = \sum H^{[k]}$$



FULL DIAGONAL
each measurement correlates a fixed # variables with q^2 blocks

$q \equiv$ dimension of the factor

SPARSE MATRIX?
Yes, mostly empty!

$\Downarrow$

non-zero blocks in H depends on # measurements

bounded by
- # poses
- perception range
- landmark density

typically → # measurements grow linearly w/ trajectory length

Solving Sparse Systems

Usually, cholesky factorization used to solve a sy mmetric positive definite system of equations ⇒ WHY? given numerically stable solution if H symmetric

$H \Delta x = -b$ ——— linear system we want to solve

$H = L L^T$ ——— cholesky (L lower triangular)

$LL^T \Delta x = -b$

y

$L y = -b$ ——— solve for y by forward propagation

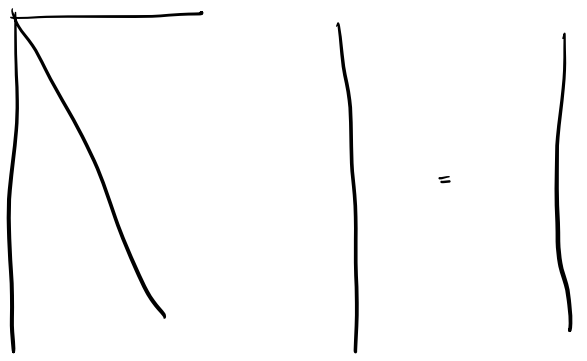
$L^T \Delta x = y$ ——— solve for x by backward propagation

⇒ H sparse DOES NOT MEAN L sparse !

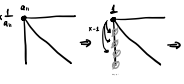
L sparse depends on non-Ø elements in H

⊕

strongly influenced by the order of elements in H



⇓

we need to triangularize this matrix:  → with this way, we may end w/a ↑↑ dense systems

What if we swap rows and/or columns to try to get a final L sparse matrix ?

⇓

→ Permutations and Cholesky: PERMUTATION MATRIX

- encodes a reordering of the variables
- elements are either Ø or 1
- exactly 1 non-zero for each row
- exactly 1 non-zero for each column
- inverse of permutation is its transpose

(SPECIAL CASE OF AN ORTHONORMAL MATRIX!)

if a matrix is sparse, there is a reordering of the variables that renders the Cholesky factor maximally sparse

↳ computing such ordering is NP complete ⇒ efficient heuristics solve the problem!

$H \Delta x = -b$ ——— system we want to solve

$P H P^T P \Delta x = -Pb$ ——— apply permutation

($P^T P = I$)

$\underbrace{P H P^T}_{H'} \underbrace{P \Delta x}_{\Delta x'} = -\underbrace{Pb}_{b'}$

$H' \Delta x' = -b'$ ——— solve the system under permutation

$H' = L' L'^T$ ——— Cholesky decomposition of H' (sparse)

$L' L'^T \Delta x' = -b'$

($L' \Delta y' = \Delta x'$)

$L' \Delta y' = -b'$

$L'^T \Delta x' = \Delta y'$

$\Delta x = P^T \Delta x'$ ——— Recover Δx by applying inverse permutation

Comments — approached complex multi-robot multi-landmark problem where:

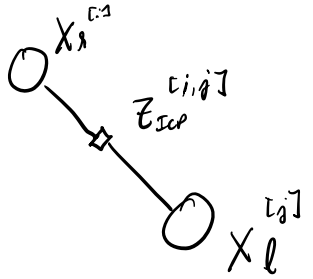
- measurement independence ⇒ H sparse
- structure of H does NOT change between iterations
- can be efficiently solved using sparse methods
- Cholesky IS NOT THE ONLY POSSIBLE WAY ⌣ e.g., QR factorization
- sparse methods rely on finding an ordering that keeps triangular system sparse!

Total Least Squares

Same methodology while also taking into account sparse structure!

given that in SLAM we are usually mixing several types of pose constraints → $X : {}^{W}T_{R}$ robot w.r.t. world

→ Pose - Landmark constraint

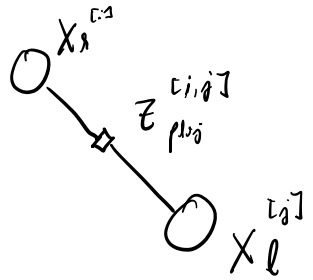


$$h_{lcp}^{[i,j]}(x) = X_s^{[i]^{-1}} X_l^{[j]} = R_n^{[i]^T}\left(X_l^{[j]} - t_n^{[i]}\right)$$

$$e_{lcp}^{[i,j]}\left(x_1^{[i]} \boxplus \Delta x_s^{[i]},\ X_l^{[j]} \boxplus \Delta x_l^{[j]}\right) =$$

$$= h_{lcp}^{[i,j]}\left(x_1^{[i]} \boxplus \Delta x_s^{[i]},\ X_l^{[j]} \boxplus \Delta x_l^{[j]}\right) - z_{lcp}^{[i,j]}$$

→ Pose - Landmark (2D projection) constraint



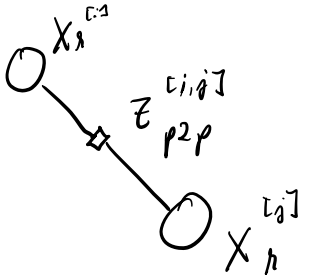
$$h_{proj}^{[i,j]}(x) = \pi\left(K\left(X_s^{[i]^{-1}} X_l^{[j]}\right)\right) =$$

$$= \pi\left(K R_s^{[i]^T}\left(X_l^{[j]} - t_s^{[i]}\right)\right)$$

$$e_{lcp}^{[i,j]}\left(x_1^{[i]} \boxplus \Delta x_s^{[i]},\ X_l^{[j]} \boxplus \Delta x_l^{[j]}\right) =$$

$$= h_{proj}^{[i,j]}\left(x_1^{[i]} \boxplus \Delta x_s^{[i]},\ X_l^{[j]} \boxplus \Delta x_l^{[j]}\right) - z_{proj}^{[i,j]}$$

→ Pose - Pose constraint



$$h_{p2p}^{[i,j]}(x) = \text{flatten}\left(X_s^{[i]^{-1}} X_s^{[j]}\right)$$

$$e_{p2p}^{[i,j]}(x) = h_{p2p}^{[i,j]}(x) - \text{flatten}\left(z_{p2p}^{[i,j]}\right)$$

$$e_{p2p}^{[i,j]}\left(X_s^{[i]} \boxplus \Delta x s^{[i]},\ X_1^{[j]} \boxplus \Delta x s^{[j]}\right) =$$

$$= h_{p2p}^{[i,j]}\left(X_s^{[i]} \boxplus \Delta x s^{[i]},\ X_1^{[j]} \boxplus \Delta x s^{[j]}\right) - \text{flatten}\left(z_{p2p}^{[i,j]}\right)$$