Notion

Date :

Localisation :- Goal is to determine our position wrt world frame.

=> Scan matching → key building block for most slam algos which use laser rays finder, RGB.

SLAM →

Scan matching → Building Block of slam.

=> Cartographer captures submaps → and integrates all submaps.

=> Localisation using odometery :-

⇓

Determine state of robot wrt environment (one's starting point)

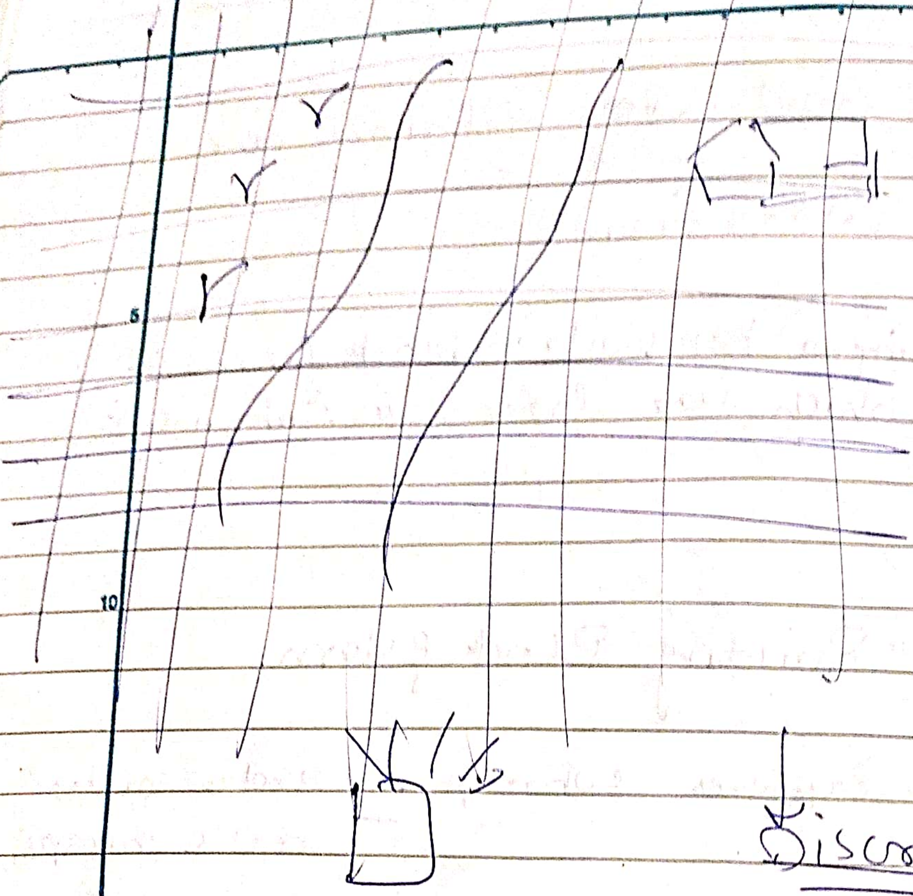
Map representations :-

→ what kind of obstacles we have / where they are

Occupancy Grid Map → focus for small scale robots

Point cloud map → real world passenger vehicles

Date :

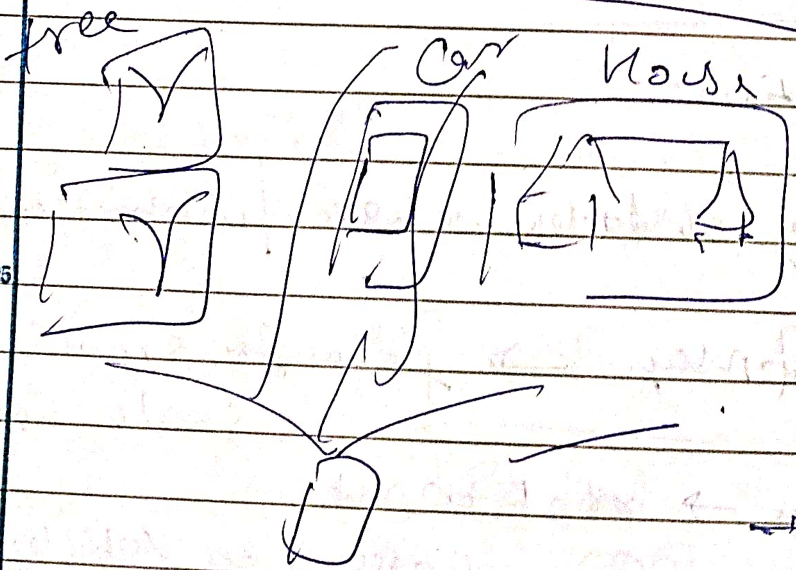


Discretize map

(Lidar Sensors)

Feature map → No discretization

Sematic Map → Each obstacle given meaning



usually

→ Combined with other map rep^n

→ Detection and Position estimate of object

Date :

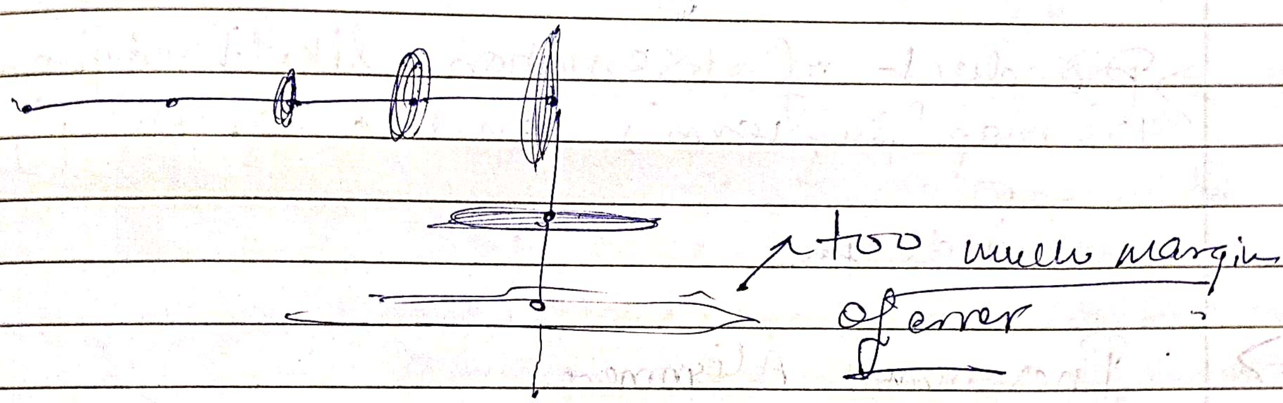
Localization (contd.)

① **odometery:-** Start at known pose and integrate control and motion measurements to estimate the current pose.
(relying on IMU, wheel encoder).

acc^n

$$\text{acc}^n \rightarrow \boxed{\int} \xrightarrow{\text{velocity}} \boxed{\int} \xrightarrow{\text{position}}$$

Uncertainity increases over time.



② Scan matching (improve pose estimate), incremental scan alignment, takes scan one after other and tries to align them so that their best match on top of each other is found.

- pose, offers correction in ordometery estimation.

* Pose correction using scan matching:-

Maximize liklihood of current pose relative to previous pose and map.

Sensor observation model

$$x_t^* = \arg\max_{x_t} \{ P(z_t \mid x_t, m_{t-1}) \, P(x_t \mid u_{t-1}, x^*_{t-1}) \}$$

map constructed so far.

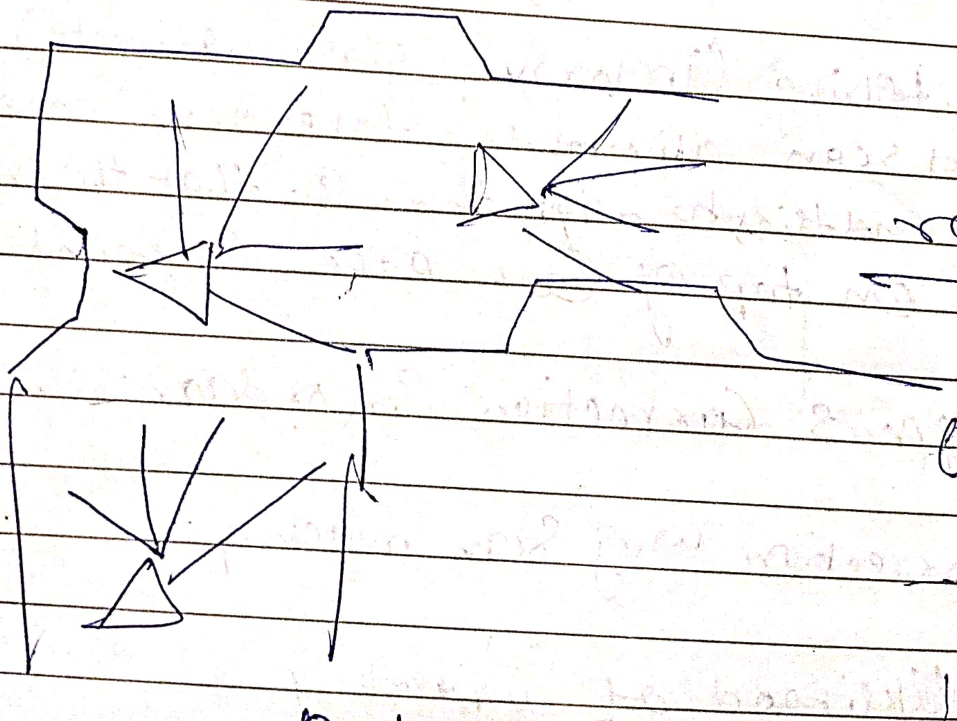
current measurement

motion model. → robot motion

⇒ Product of observation likelihood given the map (first term) and odometry)

builds so far

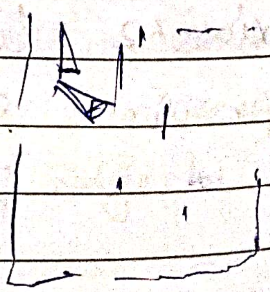
⇒ Incremented Alignment



result

Goal is to align these two scans

we have robot taking successive scans



Date :

get some association
b/w points of both scan.

To search for best overlap is to do it exhaustively by incremental alignment.

To much complexity and timetaking

⇓ other ways

one way.

Iterative closest point algorithm. (ICP)

previous scan ↔ current scan

scans of both

Tries to get some association b/w points of both scans and finds appropriate translation and rotation which maximise squared distances b/w points.

so they max overlap

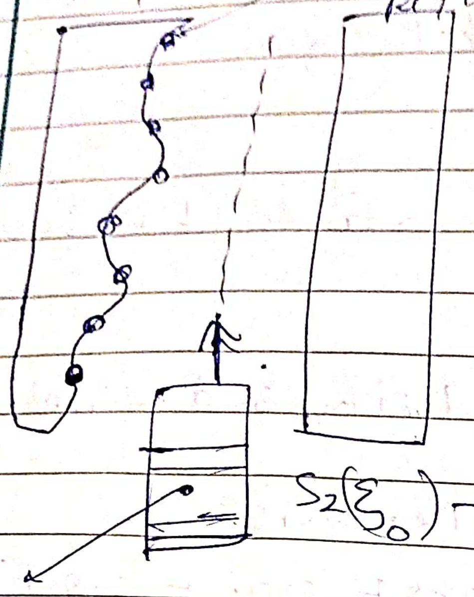
RANSAC → outlier rejection →

eliminate them { points in one scan don't have partner in other

Scan matching improves pose-estimate substantially.

$\rightarrow S_1(\xi_0) \rightarrow$ world frame

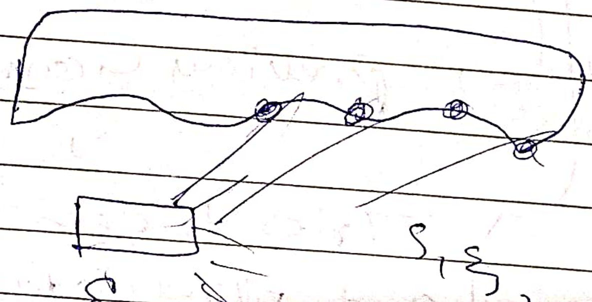
Co-ordinates of impact point in car refrence frame. from lidar rays



$S_2(\xi_0) \rightarrow$ Car frame

$\xi_0 \rightarrow$ at time zero (position)

At $\xi_1 \longrightarrow$



$S_1 \xi_1$

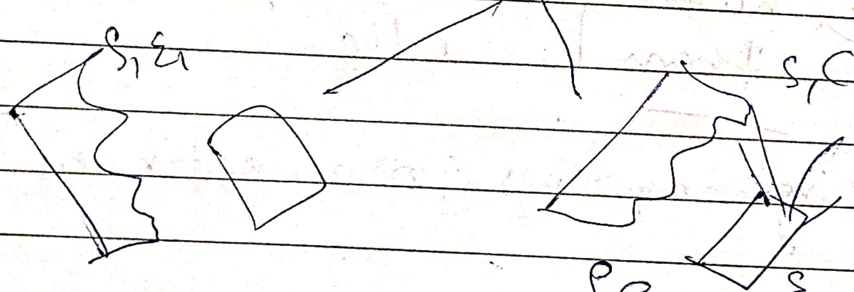
ξ (position wrt frame)

Second scan

Can be anywhere

we don't know where the car is

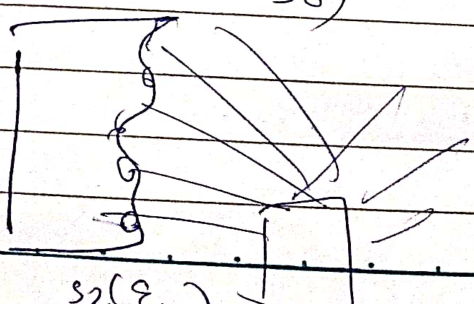
$S_1 \xi_1$



$S_1(\xi_1)$

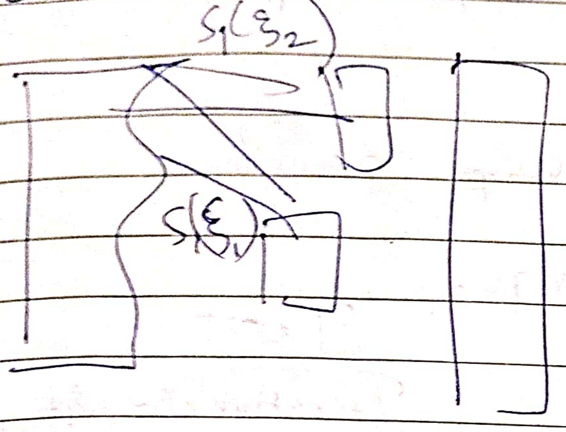
Second scan ξ_1

$S_\varphi(\xi_0)$



$S_2(\xi_{..})$

most likely car position is the car position which gives best overlap between previous scan and current scan



$$\xi^* = \arg\min \sum_{i=1}^{n} \left[1 - M(s_i(\xi))\right]^2$$

Measure of match

impact coordinates of i-th step in world frame

~~Total steps (n=1084)~~

→ Scanning → fast rate

→ Scanning ⟷ car speed

Homogenous surface bad for scan matching.

⇒ Scenery doesn't change much as you drive

Best match (measure of match) says $S_i(\xi_i) = S_i(\xi_{i+1})$

or $\xi_i = \xi_{i+1}$

Car thinks itself is not moving.

Date :

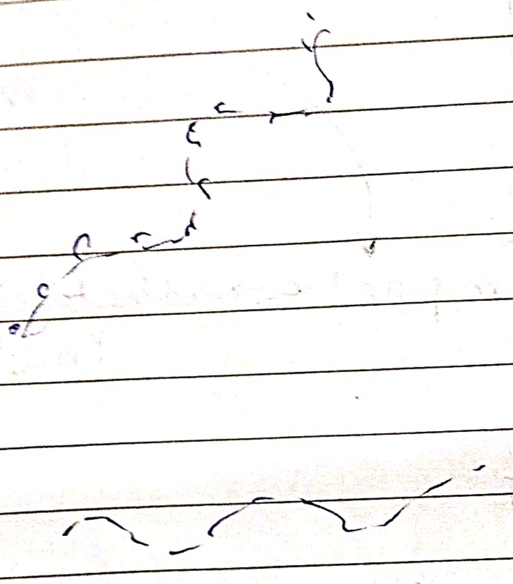
Scanmatching in a map → in pre-build map of environment → algorithm will match current scene to the scenes in map.

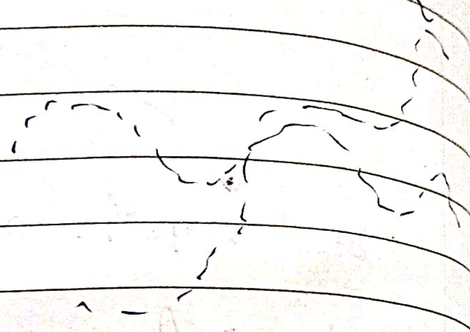
Localisation using scanmatching (Intuition)

Iterative Closest point Algo.

ICP

Iterative closest pt. transformation





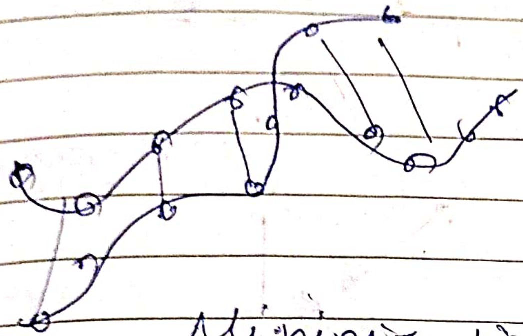
Two scans of 2-D point cloud.

compute center of mass and shift the point clouds on top of each other

goal is to compute transformation b/w two point cloud.

ICP (1) align COM

step (2) Data Association step.

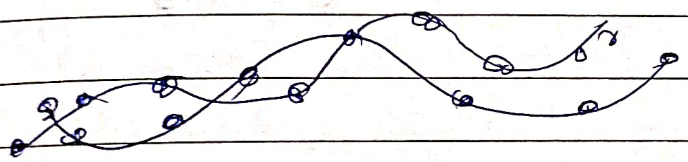


Minimize distance b/w point pairs.

Compute correspondences from P to Q

for every P_i we search closest q_j to it.

(3) Compute rotation and translation needed using SVD



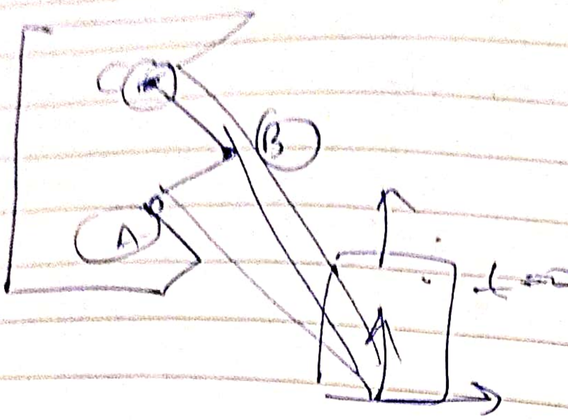
outliers → perfectly aligned scans

→ not partner of point in scan1 to scan2

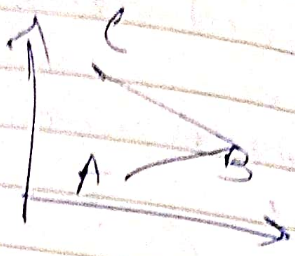
eliminate these using RANSAC → for faster computation (convergence)

Date :

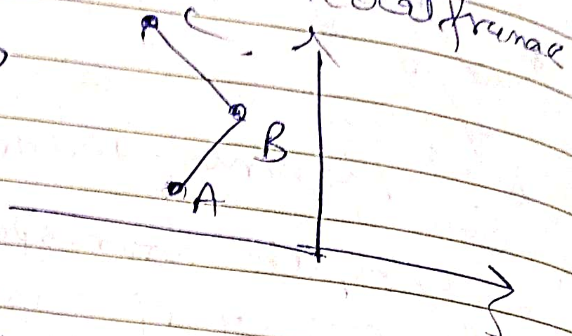
Q



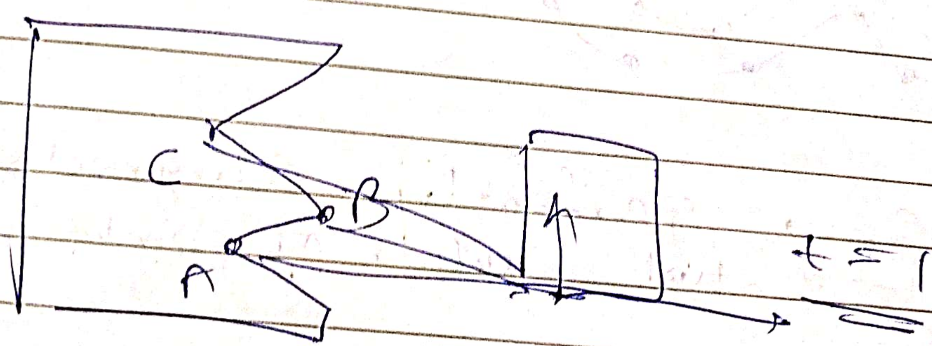
Global frame

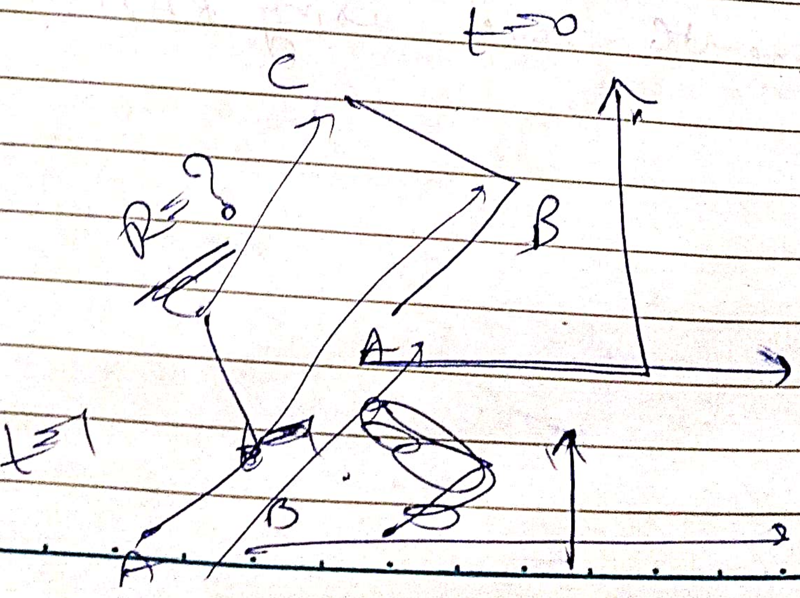


Local frame



At t = 1 we assume robot is moving in random directions ⟶ distances to landmarks have changed





Date :

$R \rightarrow$ how much the two has moved in time.

Assume closest pts correspond to each other] Iteratively find best R $\leftarrow$ Correspondence match

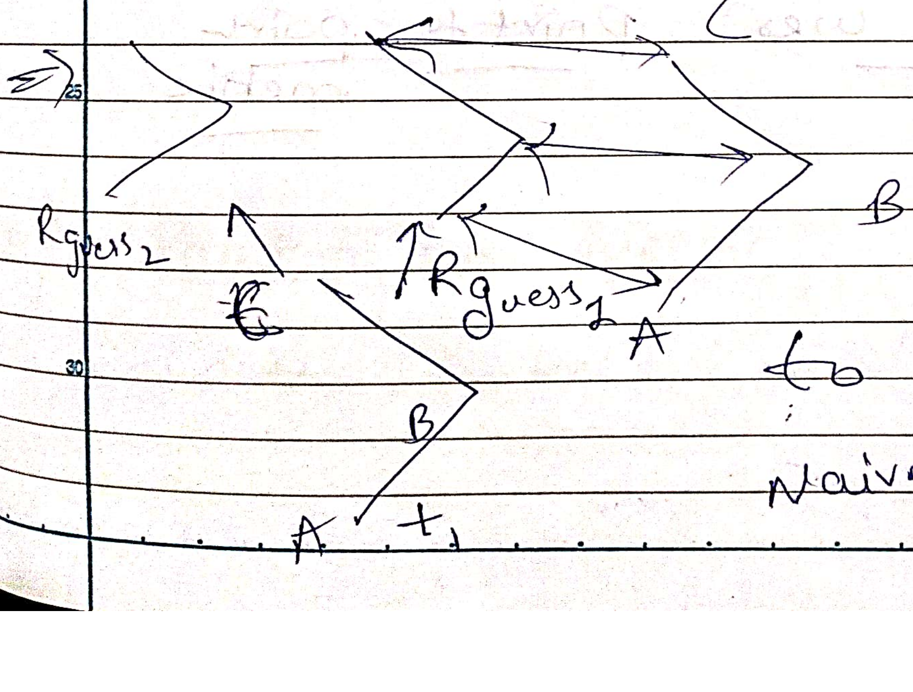
1. Make guess of R.
2. For each point in new scan ($t=k+1$) we find closest point in previous set ($t=k$)

   $\Downarrow$

   (Correspondence search)

3. Make another "better guess" of R
4. Set next guess to be current guess, repeat steps 2–4 until converges

"Better guess" $\rightarrow$ metrix $\rightarrow$ Match function



R_{guess1} or R_{guess2} to ?

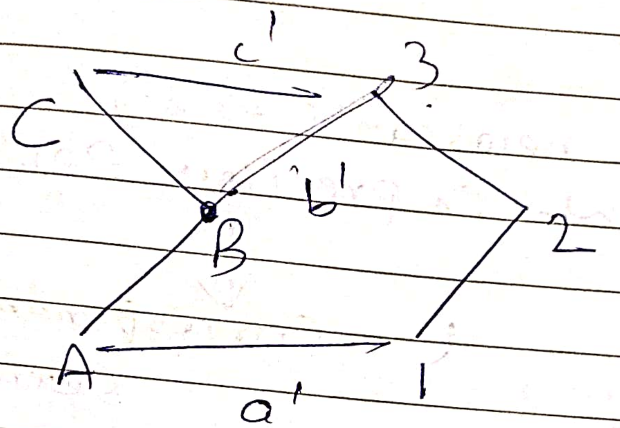
$\Downarrow$

Naive metrix $\rightarrow$ sum of squared distances.

Date :

This metric is robust → → Noise in data
→ absence of pts in Scan 1 or 2.

Next Metric ⟶ Point to point metric.



closest point to B → 3

projected point from line seprew 2 → 3

$$Score = |a'|^2 + |b'|^2 + |c'|^2$$

$$Score_{guess\,1} < Score_{guess\,2}$$

Vanilla ICP uses Point to point metric

Date :

Basic Alignment Problem with known correspondence

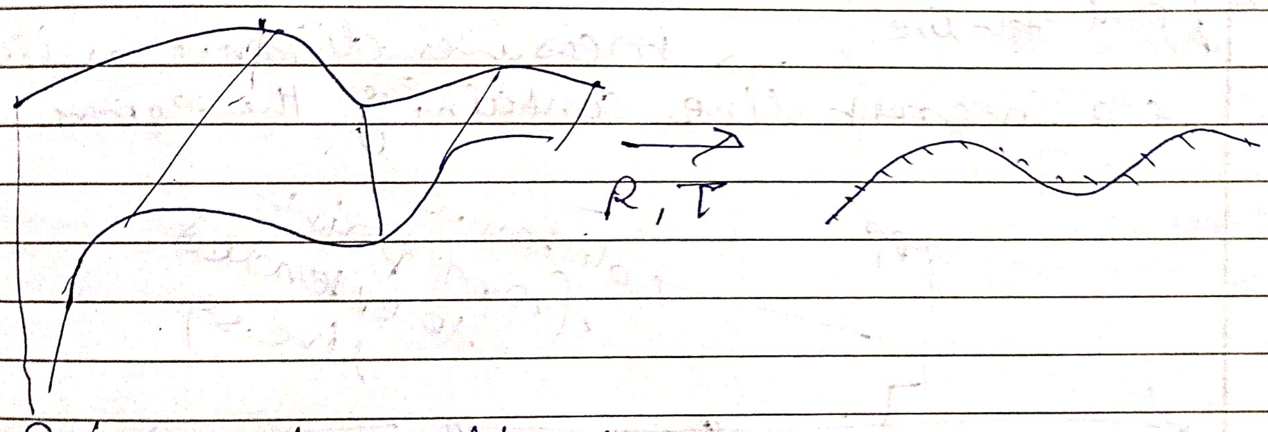
$Y = \{y_1, \ldots y_n\}$ $X = \{x_1, x_2 \ldots x_n\}$

with correspondences $C = \{i, j\}$

Wanted.

$$\sum_{(i,j) \in Z} \| y_i - (R x_j - t) \|^2 \rightarrow \min.$$

Key idea :- if correct correspondences are known, the correct relative rotation/translation can be compelted easily.



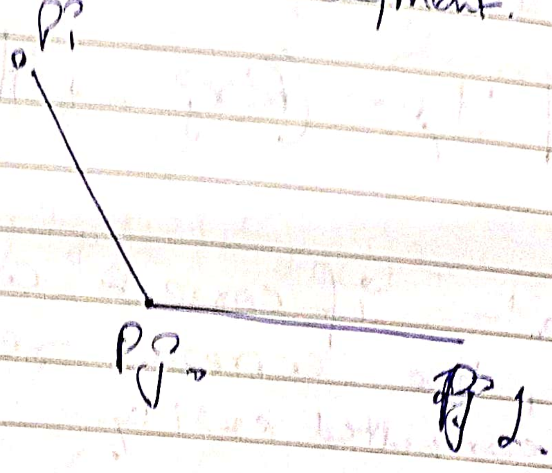
Point to line Metric.

Tricks to speed correspondence search

• P_i (point from current scan)

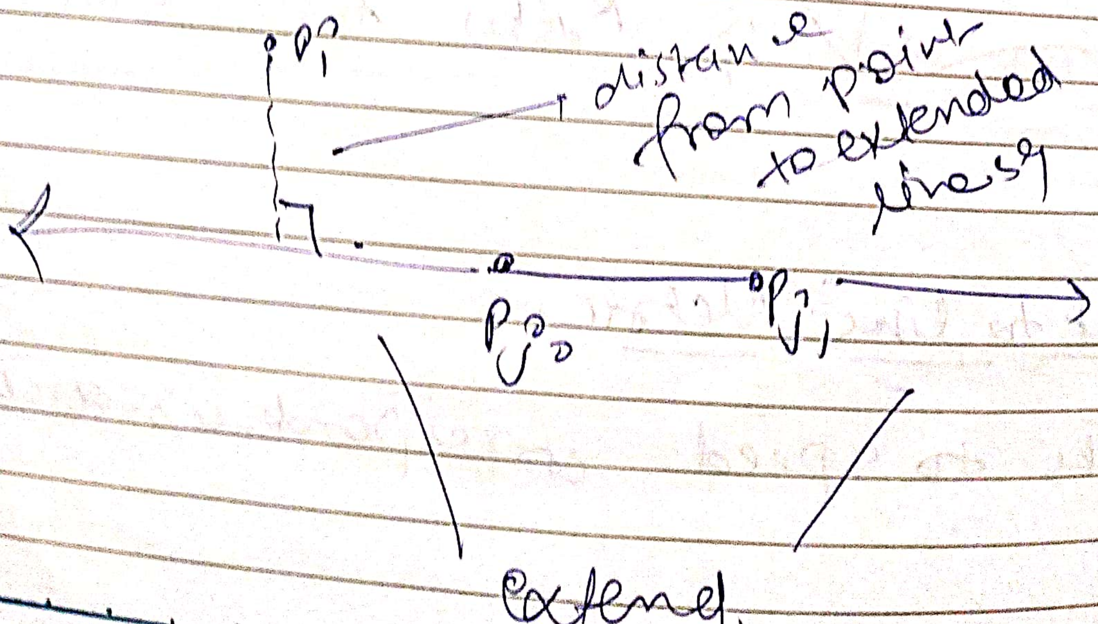
P_{j0} P_{j1} } points from prev. scan

Point-to-point measures distance as distance to nearest point on the segment.



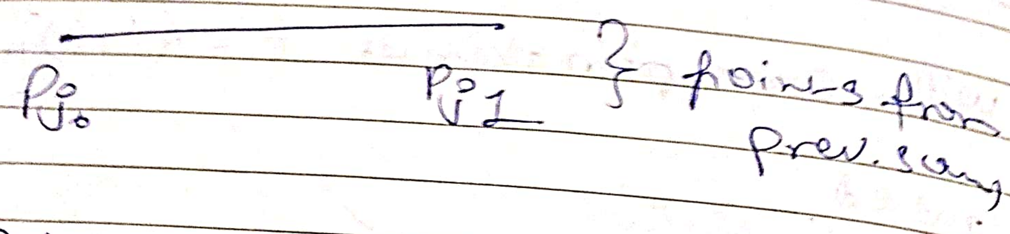
Point-to-line.

→ measures distance as distance to nearest line containing the segment

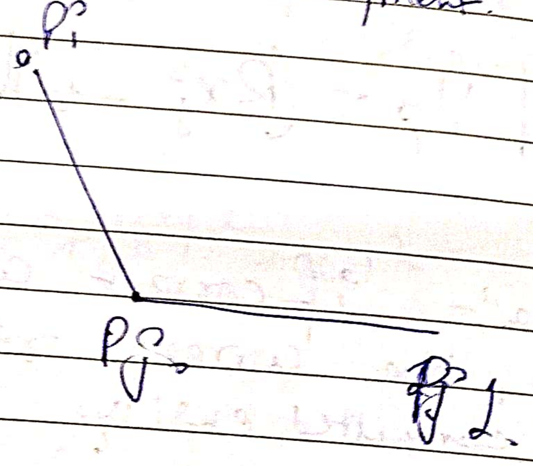


Date :

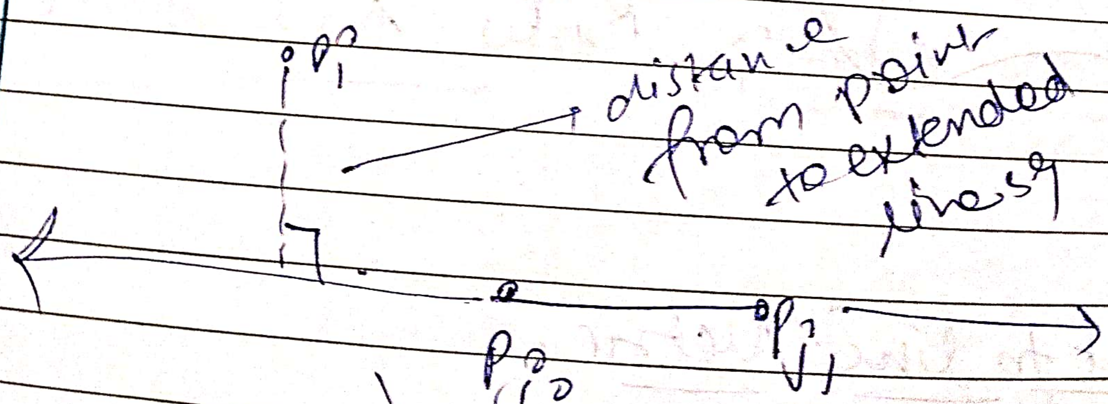
• P_i (point from current scan)

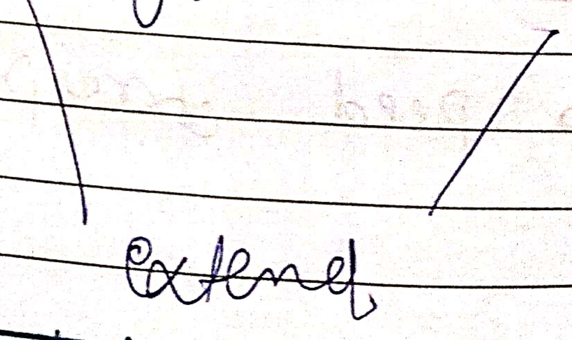


Point to point measures distance as distance to nearest point on the segment.

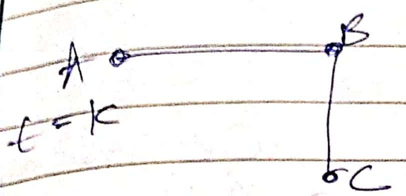


Point to-line.
→ measures distance as distance to nearest line containing the segment.

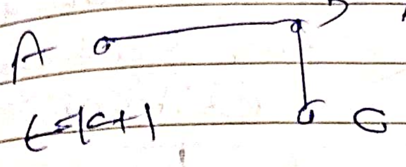




fastness of point to line metric



Assume transforms are vertical translations



to line

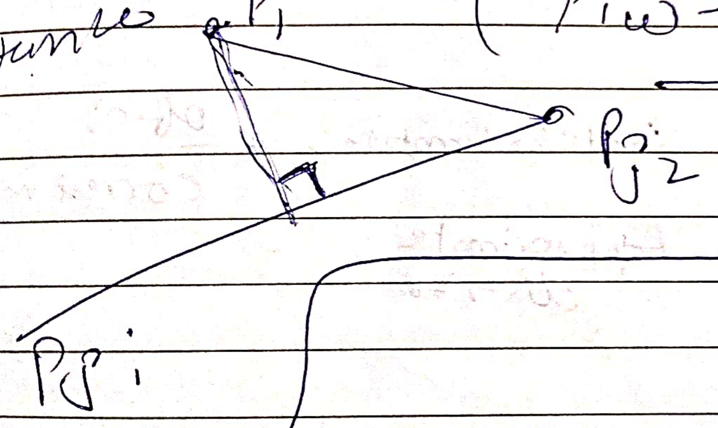
Sum of distances

→ update transform func.

$$J(q_{K+1}, C_K) = \min \left(\sum_i n_i^T \left[R(\theta_{K+1}) P_i + t_{K+1} - P_{j_1}^i \right] \right)$$

distance

$$(P_{iw} - P_{j_2}) \cdot n$$



unit normal to line connecting points in previous scans.

Date :

State Estimation

$$p(B|A) = \underbrace{\frac{p(A|B)\cdot p(B)}{p(A)}}$$

$$p(B|A, c_1, c_2 \ldots c_n)$$

$$P(B|A, c_1, \ldots c_n) = \frac{p(A|B, c_1 \ldots c_n)\, P(B|c_1 \ldots c_n)}{P(A|c_1 \ldots c_n)}$$

Approximate of distribution:-

$$\underbrace{Bel}(x_t) = P(x_t \mid O_t, U_{t-1}, O_{t-1} \ldots)$$

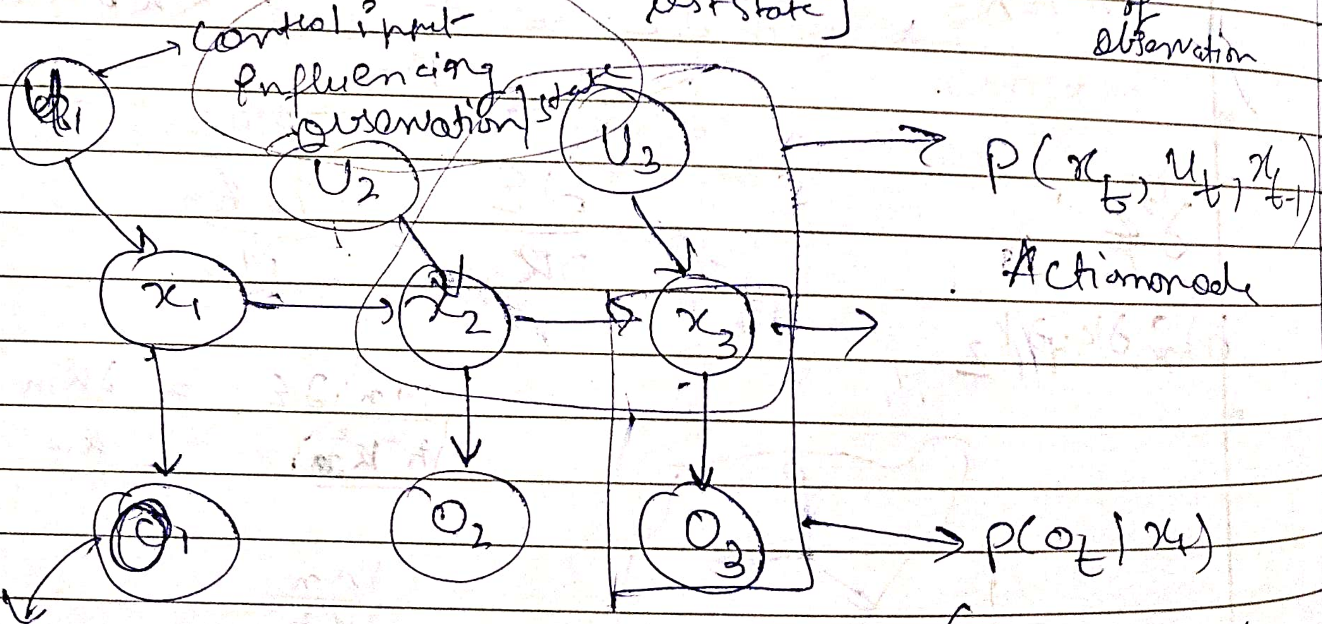
Belief of Robot state

Observation

Control input

history of observation

Markov model [Dependent on last state]



observation which will be influenced by the state

Sensor model

Likelihood model sensor

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Date :

for our sensor

$$P(O_t \mid X_1 = x_1 \ldots X_t = x_t) = P(O_t \mid X_t = x_t)$$

for our sensor, liklihood is independent of past state

$$P(X_{t+1} \mid X_1 = x_1 \ldots X_t = x_t) = P(X_{t+1} \mid X_t = x_t)$$

future state independent of past state, only on current state

Action model $\longleftrightarrow$ control input

Sensor model $\longleftrightarrow$ observation

$$Bel(x_t) = P(x_t \mid O_t, u_{t-1}, O_{t-1} \ldots)$$

prior

$$bel(x_t) = P(x_t \mid O_{1:t-1}, u_{1:t})$$

$$\int P(x_t \mid x_{t-1}, O_{1:t-1}, u_{1:t})\, P(x_{t-1} \mid O_{1:t-1}, u_{1:t})\, dx_{t-1}$$

Markov.

$$\int P(x_t \mid x_{t-1}, O_{1:t-1}, u_{1:t})\, bel(x_{t-1})\, dx_{t-1}$$

} Markov

$$\int \underbrace{p(x_t \mid x_{t-1}, u_t)}_{\text{action model}} \, bel(x_{t-1}) \, d(x_{t-1})$$

Step ② Correction with observation term

$$\underbrace{bel(x_t)}_{\text{posterior}} = p(x_t \mid o_{1:t-1}, o_t, u_{1:t})$$

$$= \frac{p(o_t \mid o_{1:t-1}, x_t, u_{1:t}) \, p(x_t \mid o_{1:t-1}, u_{1:t})}{p(o_t \mid o_{1:t-1}, u_{1:t})}$$

} Markov prop.

$$\frac{p(o_t \mid x_t) \, \overbrace{p(x_t \mid o_{1:t-1}, u_{1:t})}^{\text{prior}}}{\underbrace{p(o_t \mid o_{1:t-1}, u_{1:t})}_{\text{Normalisation term}}}$$

posterior   prior

Date :

$$bel(x_t) = P(o_t \mid x_t) \left[\begin{array}{c} \text{Action model} \\ \text{output} \end{array} \right]$$

$$P(o_t \mid o_{1:t-1}, u_{1:t})$$

normalisation Col.

$$\underbrace{bel(x_t)}_{\text{posterior}} = \eta \underbrace{P(o_t \mid x_t)}_{\text{observation}} \underbrace{\overline{bel}(x_t)}_{\substack{\text{control} \\ \text{prior}}}$$

Assume state and noise to be gaussian distribution.

Variants of Bayes filter :
1. Kalman filter
2. Extended kalman filter
3. Unscented kalman filter.

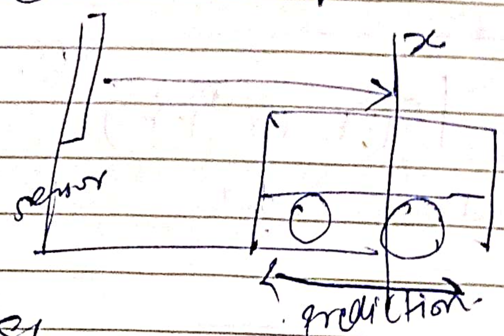
Conditional distribution :- If two set of variables are jointly gaussian then conditional distribution of one set conditioned on other set is Gaussian.

If choosing Gaussian prior would result in Gaussian posterior.

→ Particle filter

use sample based method to estimate distribution

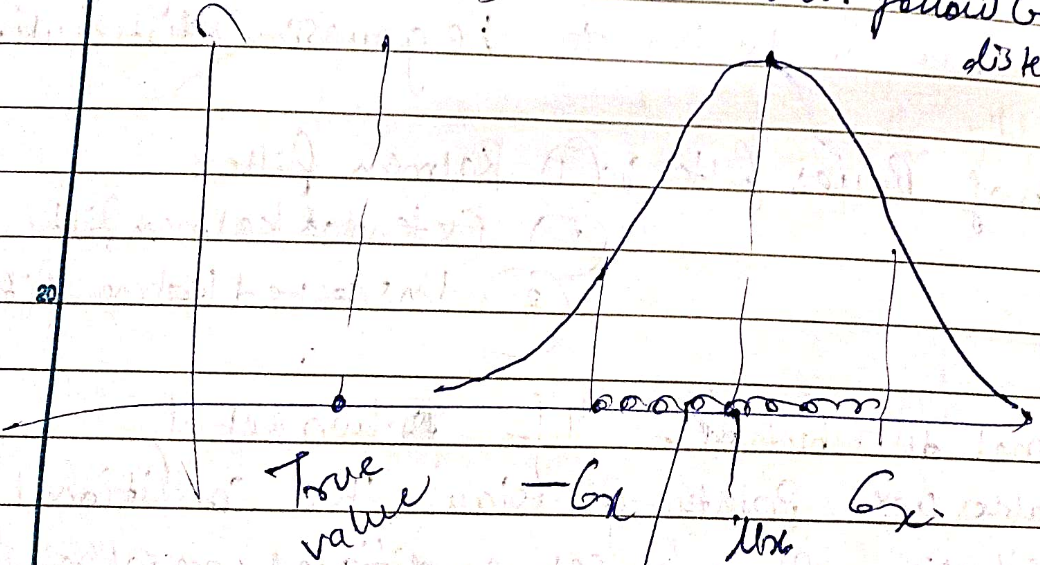
EKF example



State: position of car.

Obs = measurement from sensor to pole on car

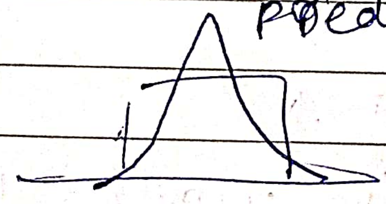
Assumption:- State and obs. follow Gaussian distribution



prediction

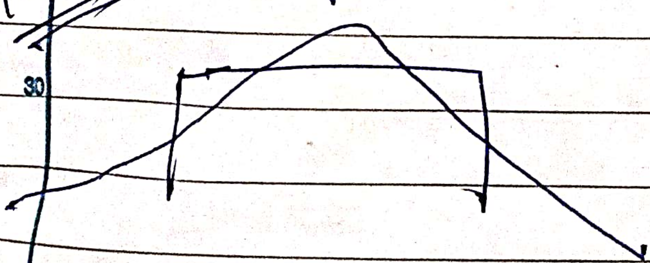
mean of measurements = μ_x

$t=0$



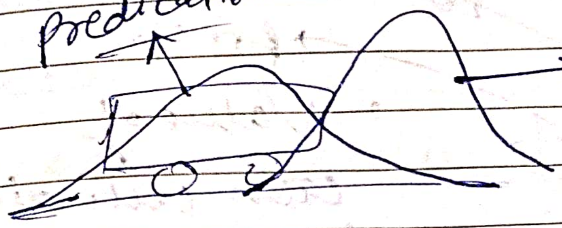
$t=\Delta s$

prediction → we become uncertain about car's position because of noisy control input



Correction of prediction with observation

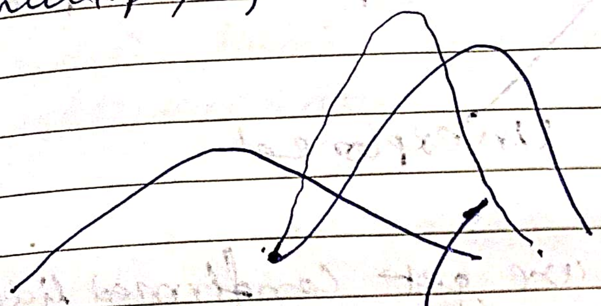
Predication



measurement/observation

Corresponds to uncertainty in noisy measurement

Combined knowledge of system is provided by multiplying two PDFs!



new PDF provides best estimate of location of car.

* Particle filters used for monte carlo localisation.

Hector Slam.

Overview of slam:- (1) laser scan at time t_0

(2) Register scan of initial map.

(3) change in time/ position (4) Laser scan at next time instant.

(5) Estimate pose change from transformation of new laser scan

(6) Align laser scan ~~to~~ new pose estimate.

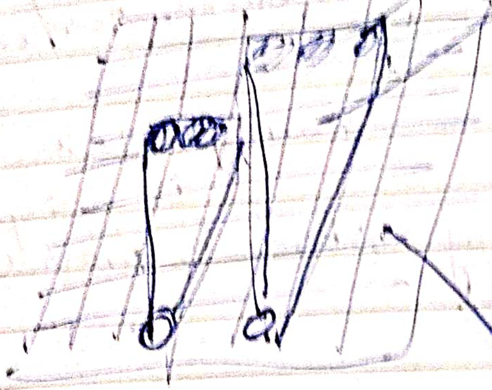
(7) map update

current laser scans are correlated with map observed to estimate change in position

with which we update map

Occupancy Grid mapping

→ occupied
→ free
→ Value of each cell indicates whether the cells occupied, free cell, or ~~unoccupied~~ Un-Explored



Unexplored

→ The cells where we get confirmed lidar hits over several elevations → are considered occupied cells [walls].

Measurement:-

$m_{xy} = 1$ LiDar hit

$m_{xy} = 0$ No hit.

Map cell:- $Z = 1, 0, -1$

$1 \rightarrow$ occupied, $0 \rightarrow$ unexplored, $-1 \rightarrow$ free

Measurement model:-

$p(Z \mid m_{xy}) \rightarrow$ probability of cell being occupied/unoccupied given lidar measurement being 1 or 0.

Indicates estimate of certainity in data obtained from lidar measurements

Date :

→ People moving [uncertainty in measurement

<u>Scan matching</u>

(1) Update map cells using first laser scan.

(2) When car proceeds further → goal is to find pose change from previous measurements, we do this by finding transformation between new laser scan and the previously registered map.

(3) Hector slam :-

$$\xi^* = \arg\min \sum_{i=1}^{n} \left[1 - M(S_i(\xi))\right]^2$$

measure of match

Goal is to minimise function.

we might endup local minima

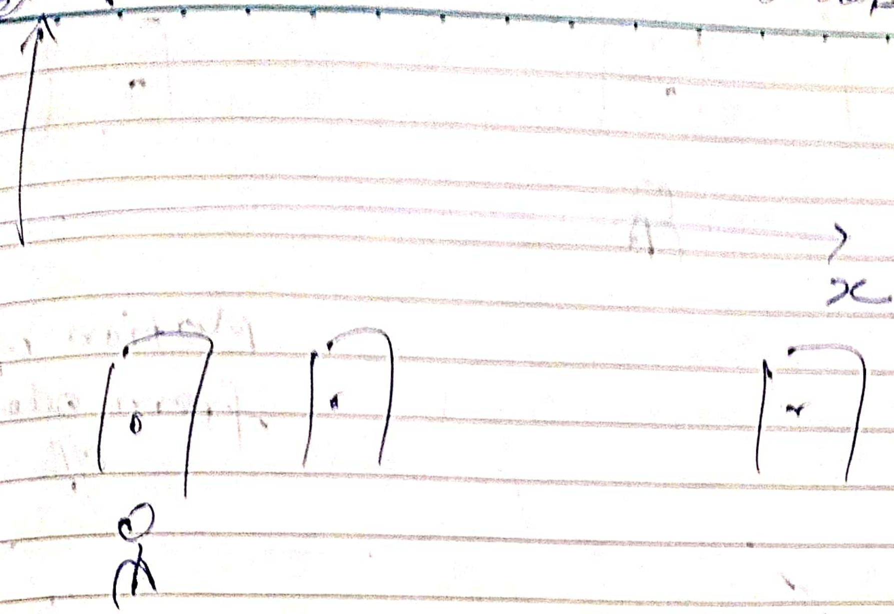
How to fix?

Equation $\xi^* = (\ )$ is optimised firstly for coarser grids. to find pose estimate.

Estimate is then used as an input to optimisation function ($\xi^* = (\ )$) of higher resolution maps.

∴ If Hectorslam ($\frac{5cm}{grid}$).

Algorithm iterates from 20cm → 15cm → 5cm belief state (bel) ⟶ Prob. of robot being on 'x'.



At t=1 ⟶ Robot senses door.

probability that sensor gives values regarding position of doors ⟶ given present position

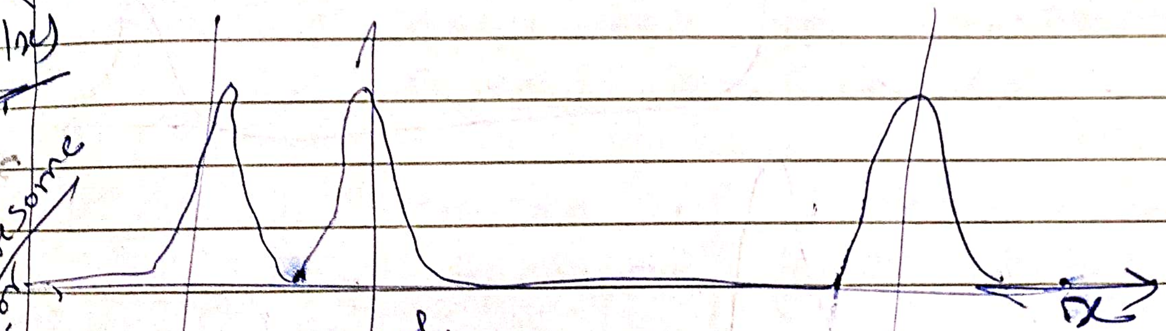
$p(z \mid x)$

↓

senses measurement ⟹ it is door or not

Bel(x)

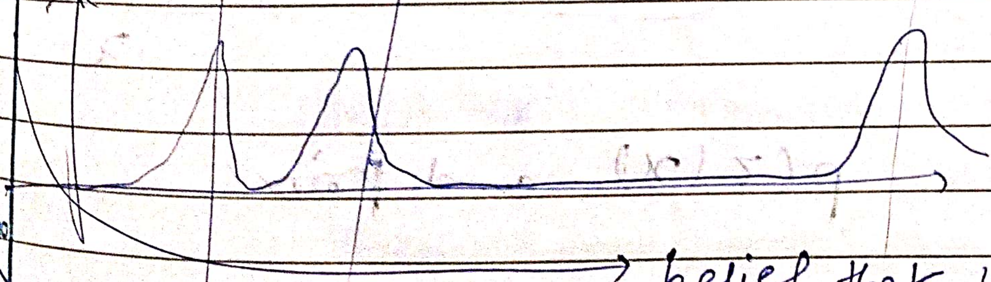
P(z|x)



At t=1 we are at some door

⟶ it can be any of the door ⟶ So gaussian peaks at all those 'x' values where doors are present

bel(x)

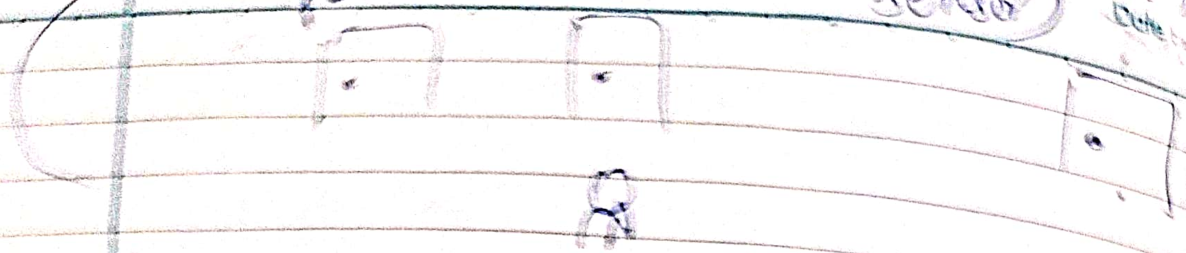


⟶ belief that we are at some 'x' position

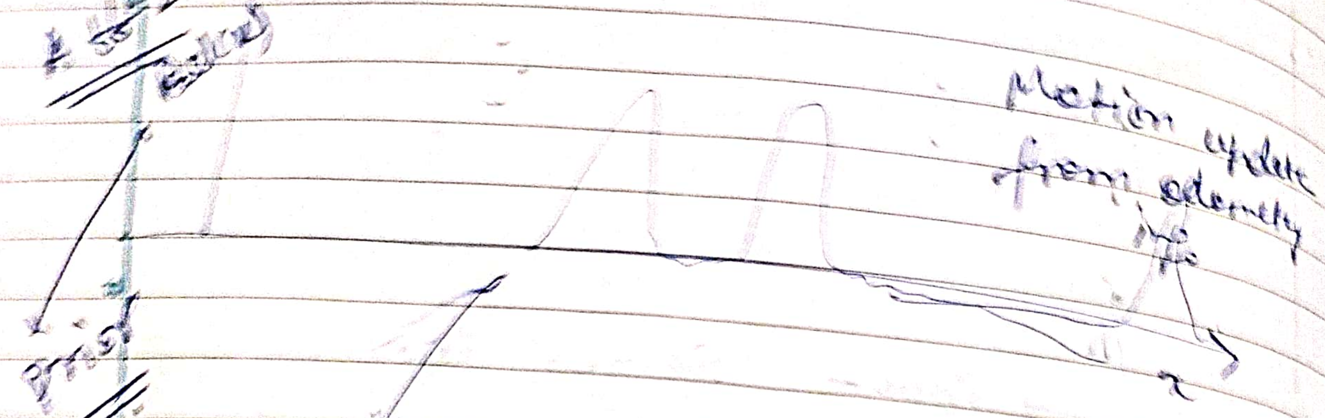
⟶ posterior belief

⟶ after measurement has been taken.

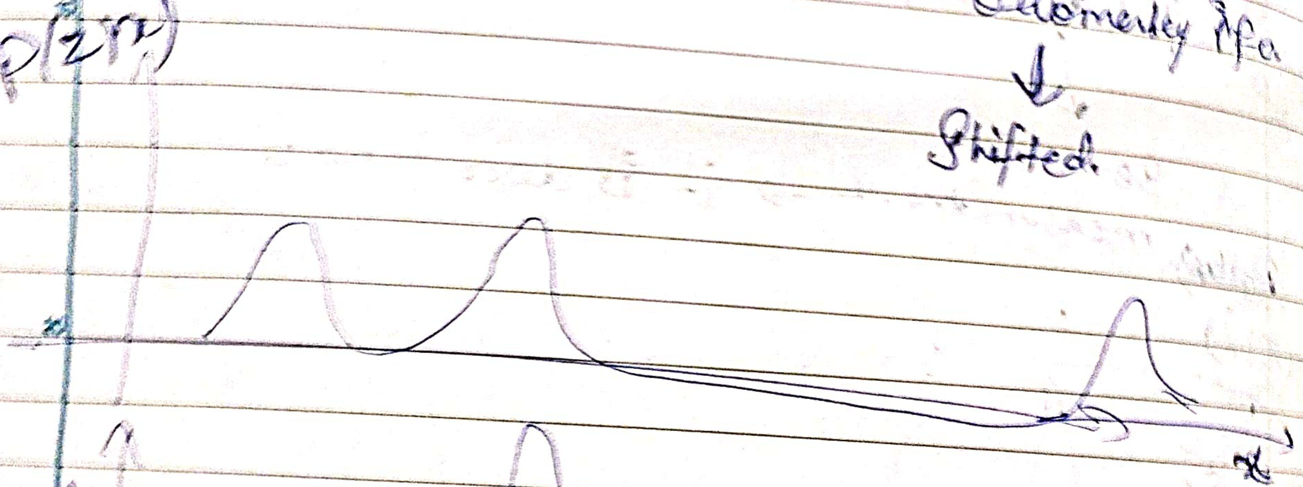
has ... position attained by (odometry + sensor) → Not convolution

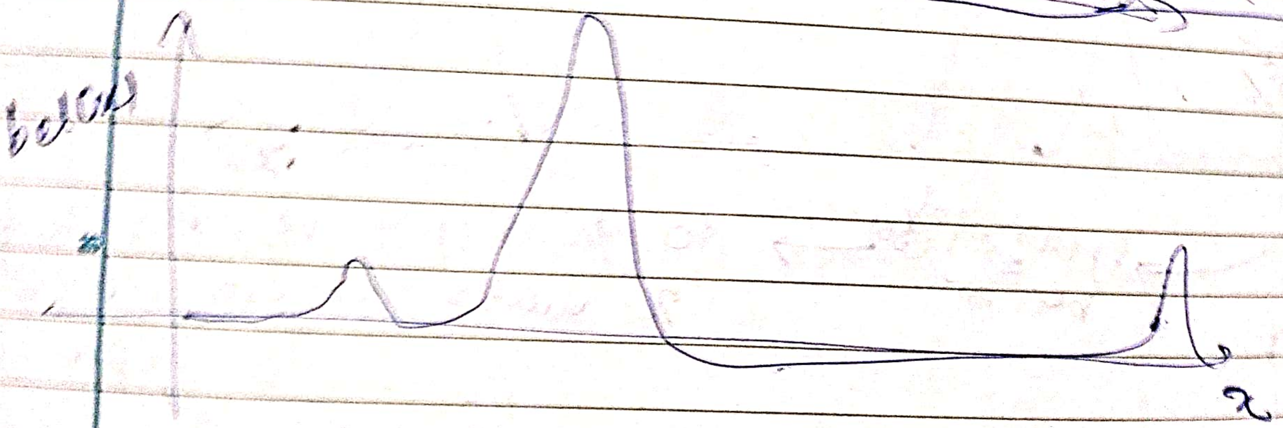


Motion update from odometry info



Previous belief state ~~updated~~ updated from odometry info
↓
Shifted.





Convolve $p(z|x)$ and prior

$$\underbrace{Z + (\text{prior})_{\text{var}}}_{\text{addition of R.V.}} \longrightarrow \text{Convolution of densities.}$$

Date :

Montecarlo Localisation (Particle filter) → we already have a pre-build map

$1 - M(\xi)$ — metric of measure of match ← we can define this to be anything.

Say correlation

$$S = \frac{\sum_m \sum_n (A_{mn} - \bar{A})(B_{mn} - \bar{B})}{\sqrt{\left(\sum_m \sum_n (A_{mn} - \bar{A})^2\right)\left(\sum_m \sum_n (B_{mn} - \bar{B})^2\right)}}$$

score

A, B

$A \longrightarrow$ map build before from scan matching.

$B \longrightarrow$ new scan corresponding to each particle.

$\bar{B}, \bar{A} \longrightarrow$ mean of all pixels in occupancy grid $z = 1, 0, -1$

of all respective map

$B_{mn}, A_{mn} \longrightarrow (m,n)^{th}$ coordinate in occupancy grid.

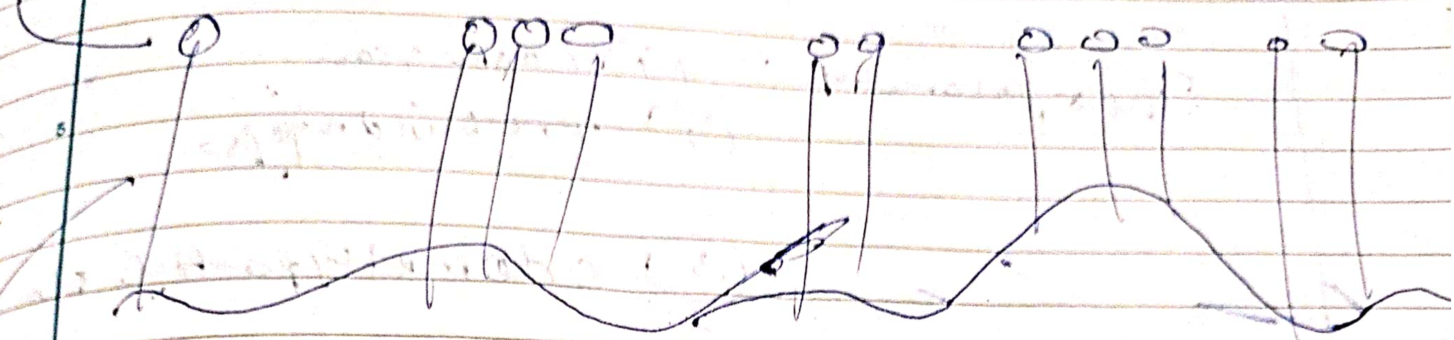
$S \longrightarrow$ measure match [correlation] between map (pre build) and new scan for each particle.

$M(\xi)$ → that function

→ of typical particles

Date :

Particle filter Resampling



discretise.

weights for each particle change and few particles become more prominent than others we further divide this particles into more smaller particles

resampling

by substitut… new particles near the large…

a new set of particles have higher density near ~~these particles~~ prominent particles (which determine loss)