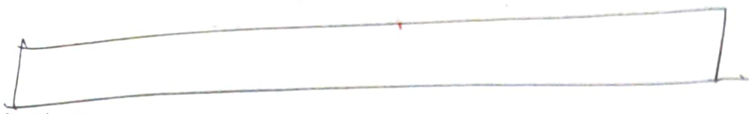
Navigation



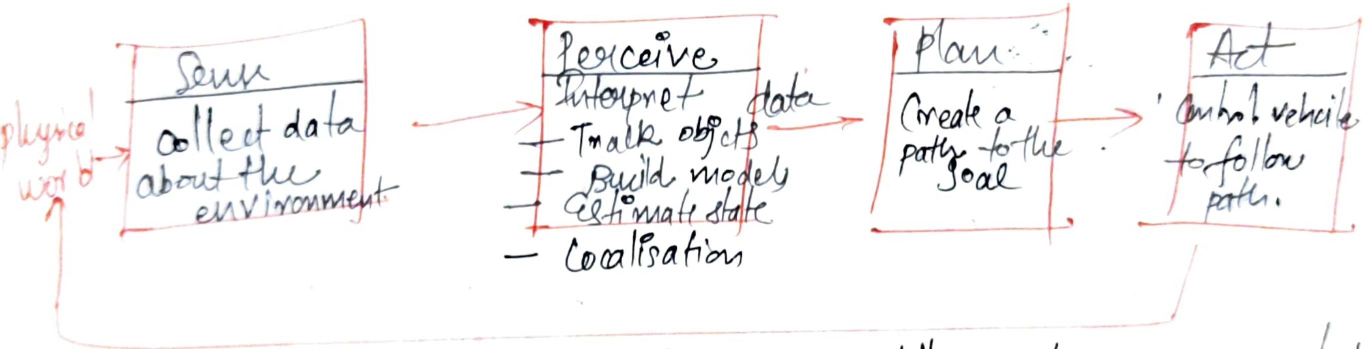
controlled by humans.

full autonomy

Heuristic approach
* Doesn't need complete environment info.

optimal approach
* Requires more environment info.
* planning is achieved through optimization

block diagram of autonomous System:-

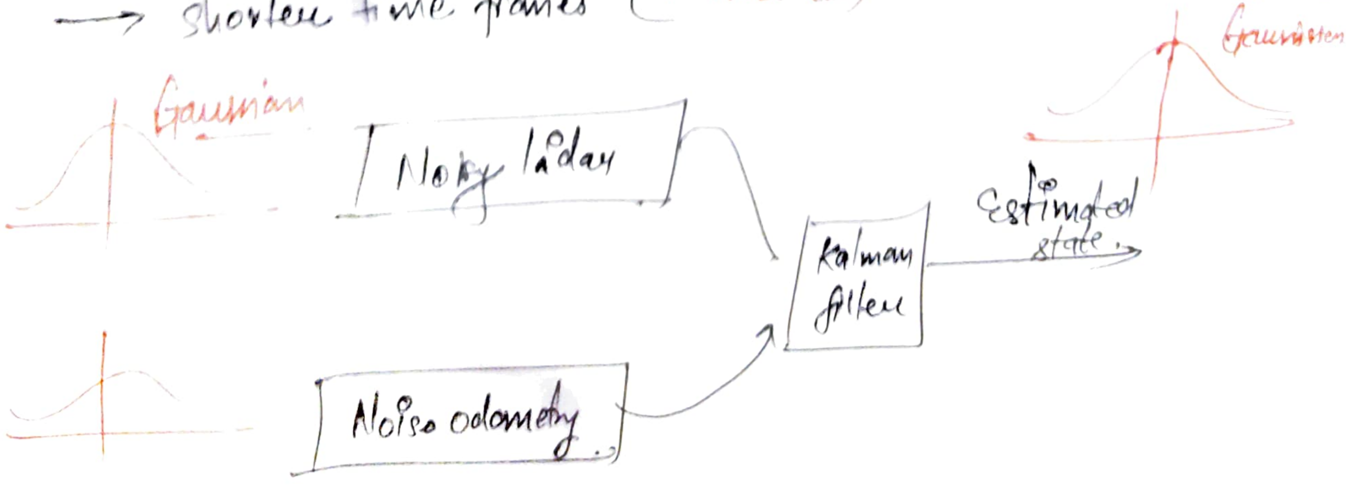


Estimating position and orientation of a mobile robot using a particle filter.

Dead reckon

If a future position is calculated using its past position and relative measurements like velocity etc.

→ shorter time frames (it is better)



Most cases in localization we don't get Gaussian distributed estimated state. This is the Localization problem!!

Monte Carlo localization

Adaptive Monte Carlo localization – (AMCL)
– recalculates the no. of particles after each gen^r so that you are not wasting computational resources

Well, here we have map. What if we don't have MAP!!
Here comes SLAM.
↓
Simultaneous Localisation And Mapping
using POSE graph Estimation.

Cyrill Course

mapping – modeling the environment.

Localisation :- Estimate the robot's poses given landmarks.

mapping :- Estimate the landmarks given the robot's poses.

SLAM Problems :-

Given : robots Controls :- $u_{1:T} = \{u_1, u_2, u_3, \ldots, u_T\}$

observations :- $z_{1:T} = \{z_1, z_2, z_3, \ldots, z_T\}$

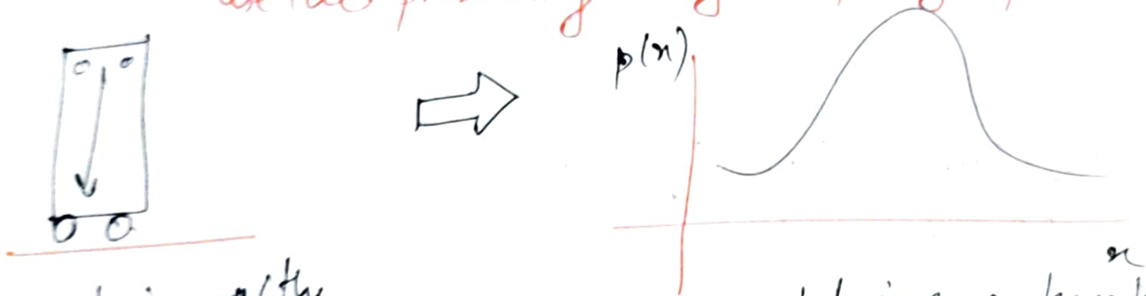
Wanted : map of environment - m

Path of robot $x_{0:T} = \{x_0, x_1, \ldots x_T\}$

reference.

Probabilistic approach:

use the probability theory to explicitly represent the uncertainty.

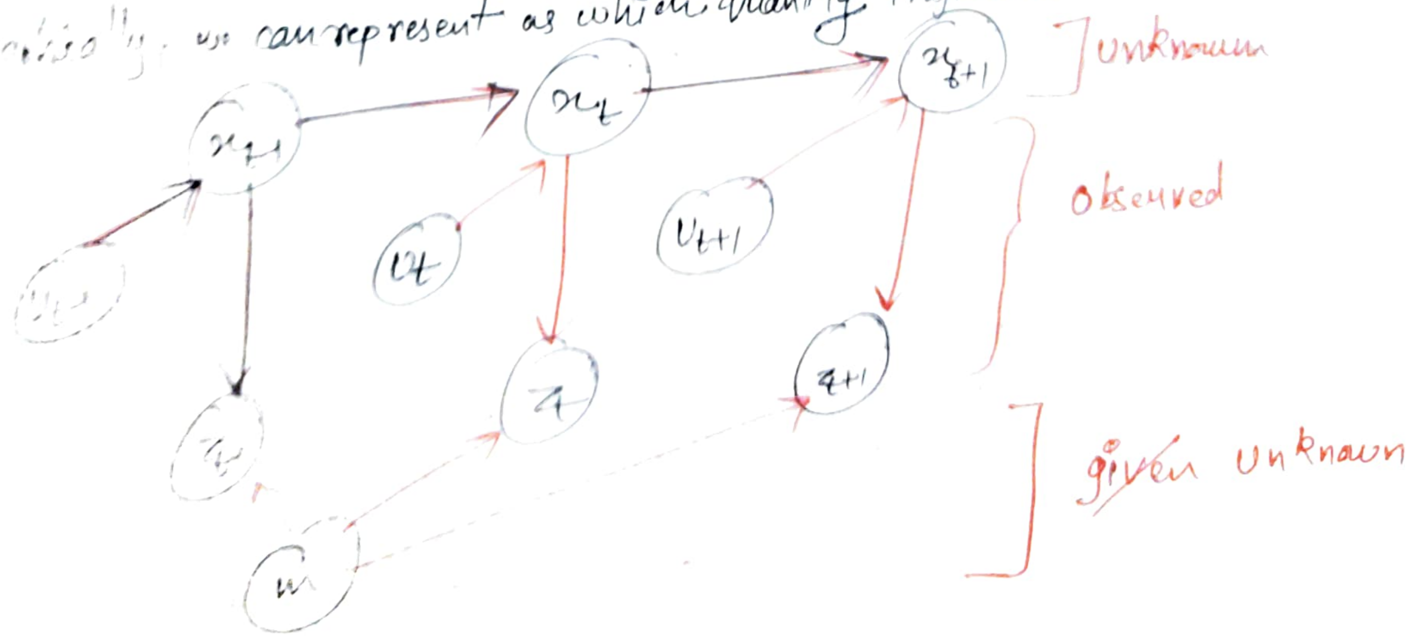


"robot in exactly here" " robot in somewhere here"

$$p(x_{0:T}, m \mid z_{1:T}, u_{1:T})$$

path map given observation controls.

graphically we can represent as which quantity influence the other



full SLAM vs. Online SLAM.

↳ complete trajectory estimation $p(x_{0:T}, m \mid z_{1:T}, u_{1:T})$

till current pos.

$p(x_t, m \mid z_{1:t}, u_{1:T})$

$$\Rightarrow \int_{x_0} \cdots \int_{x_{t-1}} p(x_{0:t}, m \mid z_{1:t}, u_{1:T})\, dx_{t-1} \cdots dx_0$$

→ solve the integral recursively

Motion model

— describes the relative motion of robot

$p(x_t \mid x_{t-1}, u_t)$

distribution, new pose, given, old pose, control.

→ Gaussian model

→ Non Gaussian model.

Homogenous Coordinates (H.C)

H.C. used in projective geometry.

A single matrix can represent affine transformation & projective transformation.

Def^n - representation 'x' of a geometric object is homogenous if x and λx represent the same object for $\lambda \neq 0$..

Eg-

$$x = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Infinitely Distant objects

It is possible to explicitly model infinitively distant points with finite coordinates

$$X_\infty = \begin{bmatrix} u \\ v \\ 0 \end{bmatrix}$$

Great tool when working with bearing-only sensors such as cameras.

3D points:-

$$X = \begin{bmatrix} u \\ v \\ w \\ t \end{bmatrix} = \begin{bmatrix} u/t \\ v/t \\ w/t \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} u/t \\ v/t \\ w/t \end{bmatrix}$$

homogenous . Euclidian

Projective transformation $[x' = MX]$

Imp $(\mathbb{P}^3)$ $x' = \underset{4\times4}{M} x$

translation : 3 parameter

$$M = \lambda \begin{bmatrix} I & t \\ 0^T & 1 \end{bmatrix} \qquad I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad t = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

$$0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Rotation : 3 parameters

$$M = \lambda \begin{bmatrix} R & 0 \\ 0^T & I \end{bmatrix}$$

R → Rotation matrix

Rotation matrix (RECAP)

$$R^{2D}(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$$R_z^{3D}(\kappa) = \begin{bmatrix} \cos(\kappa) & -\sin(\kappa) & 0 \\ \sin(\kappa) & -\cos(\kappa) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_x^{3D}(\omega) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\omega) & -\sin(\omega) \\ 0 & \sin(\omega) & \cos(\omega) \end{bmatrix}$$

$$\boxed{R^{3D}(\omega, \phi, \kappa) = R_z^{3D}(\kappa)\, R_y^{3D}(\phi)\, R_x^{3D}(\omega)}$$

$$R_y^{3D}(\phi) = \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix}$$

Combining translation and rotation: —

$$M = \lambda \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} \Rightarrow \text{(Rigid body)}$$

Similarity transformation : 7 params.

$$M = \lambda \begin{bmatrix} mR & t \\ 0^T & 1 \end{bmatrix}$$

Affine " : 9 "

(3 trans + 3 rot + 3 scale + 3 shear)

$$M = \lambda \begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix}$$

Bayes filter

$$p(\underset{\text{state}}{x} \mid \underset{\text{observation}}{z}, u)$$ ← controls

Recursive Bayes filter - 1

$bel(x_t) = p(x_t \mid z_{1:t}, u_{1:t})$

Def^n of belief

Bayes rule

$$bel(x_t) = p(x_t \mid z_{1:t}, u_{1:t})$$

$$= \eta \, p(z_t \mid x_t, z_{1:t-1}, u_{1:t}) \times p(x_t \mid z_{1:t-1}, u_{1:t})$$

$$= \eta \, p(z_t \mid x_t) \, p(x_t \mid z_{1:t-1}, u_{1:t})$$

$$= \eta \, p(z_t \mid x_t) \int_{x_{t-1}} p(x_t \mid x_{t-1}, z_{1:t-1}, u_{1:t}) \, p(x_{t-1} \mid z_{1:t-1}, u_{1:t}) \, dx_{t-1}$$

Law of total probability

Markov assumption.

$$= \eta\, p(z_t \mid x_t) \int_{x_{t-1}} p(x_t \mid x_{t-1}, u_t)\, p(x_{t-1} \mid z_{1:t-1}, u_{1:t-1})\, dx_{t-1}$$

(markov assumption)

$$\boxed{bel(x_t) = \eta\, p(z_t \mid x_t) \int_{x_{t-1}} p(x_t \mid x_{t-1}, u_t)\, \underbrace{bel(x_{t-1})}_{\text{recursive term}}\, dx_{t-1}}$$

Bayes filter is a two step process-

Prediction step :- $\overline{bel}(x_t) = \int \underbrace{p(x_t \mid u_t, x_{t-1})}_{\text{motion model}}\, bel(x_{t-1})\, dx_{t-1}$

correction step :- $bel(x_t) = \underbrace{\eta\, p(z_t \mid x_t)}_{\text{sensor or observation model.}}\, \overline{bel}(x_t)$

Kalman filters & friends

- Gaussians
- linear or linearised models.

Particle filter

- Non-parametric
- Arbitrary models (sampling required)

Velocity model

motion equation

Robot moves from (x, y, θ) to (x', y', θ')

velocity information $u = (v, \omega)$

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} -(v/\omega)\sin\theta + \frac{v}{\omega}\sin(\theta + \omega\Delta t) \\ (v/\omega)\cos\theta + (-v/\omega)\cos(\theta + \omega\Delta t) \\ \omega\Delta t \end{pmatrix}$$

x_{t-1}

Sensor based model

$$bel(x_t) = \eta \boxed{p(z_t|x_t)} \overline{bel}(x_{t-1})$$

assuming laser range finder.

KALMAN FILTERS

Slam is a state estimation problem.

→ Baye's filter is one tool for state estimation.

recap:- Prediction $\overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}$

correction $bel(x_t) = \eta\, p(z_t|x_t)\, \overline{bel}(x_t)$

Kalman filter is a implementation of Baye's filter.

Assumptions:- models are linear

— Distributions are gaussians.

Everything is Gaussian!

Properties : Marginalization and Conditioning

Given $x = \begin{pmatrix} x_a \\ x_b \end{pmatrix}$ $p(x) = N$

The marginals are Gaussians

(Gaussian distribution)

$p(x_a) = N$ $p(x_b) = N$

as well as the Conditionals

$p(x_a|x_b) = N$ $p(x_b|x_a) = N$

linear model

motion and observation model are linear functions.

(how world changes with zero i/p)

$$x_t = A_t x_{t-1} + B_t u_t + \epsilon_t \rightarrow \text{noise}$$

(A_t, B_t, C_t should be given)

$$z_t = C_t x_t + \delta_t$$

↳ how to obtain expected observation given x_t

$A_t \rightarrow (n \times n)$ matrix, describes how state evolves from $t-1$ to t without controls or noise.

$B_t \rightarrow$ matrix $(n \times l)$, describes how the control u_t changes the state from $t-1$ to t.

$n \rightarrow$ dimensionality of state.

$l \rightarrow$ dimensionality of our odometry

$C_t \rightarrow (k \times n) \rightarrow$, describes how to map the state x_t to an observation z_t.

$k \rightarrow$ dimension of observation

ϵ_t, $\delta_t \rightarrow$ random variable representing the process and measurement noise that are assumed to be independent and normally distributed with covariance R_t and Q_t respectively.

Linear motion model,

Motion under Gaussian noise leads to

$$p(x_t \mid u_t, x_{t-1}) = \eta \exp\left[-\frac{1}{2}\left[x_t - A x_{t-1} - B_t u_t\right]^T R_t^{-1} \left(x_t - A x_{t-1} - B_t u_t\right)\right]$$

$R_t \rightarrow$ describes noise.

Linear observation model

$$p(z_t \mid x_t) = \eta \exp\left[-\frac{1}{2}(z_t - C x_t)^T Q_t^{-1} (z_t - C x_t)\right]$$

$Q_t \rightarrow$ describes noise.

Kalman filter algorithm

1. Kalman filter $(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$:
2. $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$
3. $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$

} Prediction steps

4. $K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$
5. $\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$
6. $\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$

} correction steps.

7. return μ_t, Σ_t

mean

uncertainty (covariance matrix)

EKF Linearization: 1st order taylor Expansion

Prediction :-

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + \underbrace{\frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}}}_{=: G_t} (x_{t-1} - \mu_{t-1})$$

Correction:-

$$h(x_t) \approx h(\bar{\mu}_t) + \underbrace{\frac{\partial h(\bar{\mu}_t)}{\partial x_t}}_{=: H_t} (x_t - \bar{\mu}_t)$$

Jacobian matrices

Jacobian matrix
→ non square matrix $m \times n$ in general

given a vector valued funcⁿ

$$g(x) = \begin{pmatrix} g_1(x) \\ g_2(x) \\ \vdots \\ g_m(x) \end{pmatrix}$$

Jacobian matrix is

$$G_x = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \cdots & \frac{\partial g_1}{\partial x_n} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & \cdots & \frac{\partial g_2}{\partial x_n} \\ \vdots & & & \\ \frac{\partial g_m}{\partial x_1} & \frac{\partial g_m}{\partial x_2} & & \frac{\partial g_m}{\partial x_n} \end{bmatrix}$$

* Jacobian matrix is the orientation of the tangent plane to the vector-valued function at a given point.
* Generalises the gradient of a scalar valued funcⁿ.

Extended Kalman filter algorithm

1: Extended_Kalman_filter($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):

2. $\bar{\mu}_t = \underline{g(u_t, \mu_{t-1})}$

3. $\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$

4. $K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$

5. $\mu_t = \bar{\mu}_t + K_t(z_t - \underline{h(\bar{\mu}_t)})$

what we expect to observe

what we expect to observe

6. $\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$

7: return μ_t, Σ_t

$$\boxed{\begin{array}{l} A_t \leftrightarrow G_t \\ C_t \leftrightarrow H_t \end{array}}$$

EKF slam

(1) Application of EKF to SLAM.
(2) estimate robot's pose & locations of landmarks in environment.
(3) Assumption:— known correspondences.
(4) State space (for 2D plane) is

$$x_t = (\underbrace{x, y, \theta}_{\text{robot's pose}}, \underbrace{m_{1,x}, m_{1,y}}_{\text{landmark 1}}, \ldots, \underbrace{m_{n,x}, m_{n,y}}_{\text{landmark n}})^T$$

Filter Cycle

(1) State prediction
(2) Measurement prediction
(3) Measurement
(4) Data association
(5) Update.

State prediction

To be update

$$\underbrace{\begin{pmatrix} x_R \\ m_1 \\ \vdots \\ m_n \end{pmatrix}}_{\mu} \quad \underbrace{\begin{pmatrix} \Sigma_{x_R x_R} & \Sigma_{x_R m_1} & \cdots & \Sigma_{x_R m_n} \\ \Sigma_{m_1 x_R} & \Sigma_{m_1 m_1} & \cdots & \Sigma_{m_1 m_n} \\ \vdots & \vdots & & \vdots \\ \Sigma_{m_n x_R} & \Sigma_{m_n m_1} & \cdots & \Sigma_{m_n m_n} \end{pmatrix}}_{\Sigma \text{ (covariance)}}$$

remaining same

* prediction steps is linear in no. of landmarks.
(linear Complexity)

If there is any sensor update, it is really expensive operation if state space is large.

Concrete Example

set up

(1) Robot moves in the 2D plane.
(2) Velocity-based motion model.
(3) Robot observes point landmarks (x, y).
(4) Range-bearing Sensors.
(5) Known data association.
(6) Known number of landmarks.

Initialization

(1) Robot starts in its own reference frame (all landmarks unknown)
(2) $\boxed{2N+3 \text{ dimensions}}$

$$\mu_0 = \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 \end{pmatrix}^T$$

$$\Sigma_0 = \begin{pmatrix} 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \infty & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \infty \end{pmatrix}$$

at $t = 1$

$$\bar{\mu}_t = g(u_t, \mu_{t-1}) = g(u_1, \mu_0)$$

In velocity model,

Goal :- Update state space based on the robot's motion.

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \underbrace{\begin{pmatrix} -\frac{v_t}{\omega_t}\sin\theta + \frac{v_t}{\omega_t}\sin(\theta + \omega_t \Delta t) \\ \frac{v_t}{\omega_t}\cos\theta - \frac{v_t}{\omega_t}\cos(\theta + \omega_t \Delta t) \\ \omega_t \Delta t \end{pmatrix}}_{g_{x,y,\theta}(v_t, (x,y,\theta)^T)}$$

to the 2N+3 dimensional space

$$\underbrace{\begin{pmatrix} x' \\ y' \\ \theta' \\ \vdots \end{pmatrix} = \begin{pmatrix} x \\ y \\ \theta \\ \vdots \end{pmatrix} + \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \underbrace{0 & \cdots & 0}_{2N \text{ columns}} \end{pmatrix}^T}_{F_x^T} \begin{pmatrix} \\ \text{Let say (A)} \\ \\ \end{pmatrix}}_{g(u_t, x_t)}$$

landmarks positions are untouched while robot's pose is updated.

So $\bar{\mu}_t = g(v_t, \mu_{t-1})$ is done !!

Now moving to $\Sigma_t = G_t \Sigma_{t-1} G_t^T + R_t \rightarrow$ given

(Jacobian)

as said earlier function 'g' only affects the robot's motion and not the landmarks.

Jacobian of motion (3 X 3)

$$G_t = \begin{pmatrix} G_t^x & 0 \\ 0 & I \end{pmatrix}$$

Identity (2N X 2N)

$$G_t^x = \frac{\partial}{\partial(x,y,\theta)^T}\left[\begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} A \end{pmatrix}\right]$$

$$G_t^x = I + \frac{\partial}{\partial(x,y,\theta)^T}\begin{pmatrix} -\frac{v_t \sin\theta}{\omega_t} + \frac{v_t}{\omega_t}\sin(\theta+\omega_t\Delta t) \\ \frac{v_t}{\omega_t}\cos\theta - \frac{v_t}{\omega_t}\cos(\theta+\omega_t\Delta t) \\ \omega_t\Delta t \end{pmatrix}$$

$$= I + \begin{pmatrix} 0 & 0 & -\frac{v_t}{\omega_t}\cos\theta + \frac{v_t}{\omega_t}\cos(\theta+\omega_t\Delta t) \\ 0 & 0 & -\frac{v_t}{\omega_t}\sin\theta + \frac{v_t}{\omega_t}\sin(\theta+\omega_t\Delta t) \\ 0 & 0 & 0 \end{pmatrix}$$

$$\therefore G_t^x = \begin{pmatrix} 1 & 0 & -\frac{v_t}{\omega_t}\cos\theta + \frac{v_t}{\omega_t}\cos(\theta+\omega_t\Delta t) \\ 0 & 1 & -\frac{v_t}{\omega_t}\sin\theta + \frac{v_t}{\omega_t}\sin(\theta+\omega_t\Delta t) \\ 0 & 0 & 1 \end{pmatrix}$$

$$\therefore \bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$$

$$= \begin{pmatrix} G_t^x & 0 \\ 0 & I \end{pmatrix}\begin{bmatrix} \Sigma_{xx} & \Sigma_{xm} \\ \Sigma_{mx} & \Sigma_{mm} \end{bmatrix}\begin{pmatrix} (G_t^x)^T & 0 \\ 0 & I \end{pmatrix} + R_t$$

$$= \begin{bmatrix} G_t^x \Sigma_{xx} (G_t^x)^T & G_t^x \Sigma_{xm} \\ (G_t^x \Sigma_{xm})^T & \Sigma_{mm} \end{bmatrix} + R_t$$

$\therefore$ Prediction step DONE!!

$$\underbrace{F_x^T R_t^x F_x}_{R_t}$$

~~Fx~~ Overview of prediction steps $(\mu_{t-1}, \Sigma_{t-1}, v_t, z_t, c_t, R_t)$:

(2) $$F_x = \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \end{pmatrix}$$

(3) $$\bar{\mu}_t = \mu_{t-1} + F_x^T \begin{pmatrix} -\frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta}) + \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta} + \omega_t \Delta t) \\ \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta}) - \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta} + \omega_t \Delta t) \\ \omega_t \Delta t \end{pmatrix}$$

(4) $$G_t = I + F_x^T \begin{pmatrix} 0 & 0 & -\frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta}) + \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta} + \omega_t \Delta t) \\ 0 & 0 & -\frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta}) - \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta} + \omega_t \Delta t) \\ 0 & 0 & 0 \end{pmatrix}$$

(5) $$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + \underbrace{F_x^T R_t^x F_x}_{R_t}$$

Correction Step.

1. Known data association.
2. $c_t^i = j$: i-th measurement at time t observes the landmark with index j.
3. Initialize landmark if unobserved.
4. Compute the expected observation.
5. Compute the Jacobian of h
6. Proceed with computing the Kalman gain.

How does an observation looks like

Range bearing observation $z_t^i = (r_t^i, \phi_t^i)^T$

if landmark has not been observed.

$$\underbrace{\begin{pmatrix} \bar{\mu}_{j,x} \\ \bar{\mu}_{j,y} \end{pmatrix}}_{\text{observed location of landmark } j} = \underbrace{\begin{pmatrix} \bar{\mu}_{t,x} \\ \bar{\mu}_{t,y} \end{pmatrix}}_{\text{estimated robot's location}} + \underbrace{\begin{pmatrix} r_t^i \cos(\phi_t^i + \bar{\mu}_{t,\theta}) \\ r_t^i \sin(\phi_t^i + \bar{\mu}_{t,\theta}) \end{pmatrix}}_{\text{relative measurement.}}$$

Expected observation:-

Compute expected observation according to current estimate.

$$\delta = \begin{pmatrix} \delta_x \\ \delta_y \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{j,x} - \bar{\mu}_{t,x} \\ \bar{\mu}_{j,y} - \bar{\mu}_{j,y} \end{pmatrix}$$

$q = \delta^T \delta$ = square euclidian distance.

$$\hat{z}_t^i = \begin{pmatrix} \sqrt{q} \\ \tan^{-1}(\delta_y, \delta_x) - \bar{\mu}_{t,\theta} \end{pmatrix} = h(\bar{\mu}_t).$$

Compute the Jacobian.

$$^{low}H_t^i = \frac{\partial h(\bar{\mu}_t)}{\partial \bar{\mu}_t}$$

low dim. space $(x, y, \theta, m_{j,x}, m_{j,y})$

$$= \begin{pmatrix} \frac{\partial \sqrt{q}}{\partial x} & \frac{\partial \sqrt{q}}{\partial y} & \frac{\partial}{\partial \theta} & \frac{\partial}{\text{position of landmarks}} \\ \frac{\partial \operatorname{atan2}(..)}{\partial x} & \frac{\partial \operatorname{atan2}(..)}{\partial y} & \cdots & \end{pmatrix}$$

$$\frac{\partial \sqrt{q}}{\partial x} = \left(\frac{1}{2}\frac{1}{\sqrt{q}}\right) 2\delta_x(-1) = \frac{1}{q}\left(-\sqrt{q}\,\delta_x\right)$$

$$\therefore\ {}^{low}H_t^i = \frac{\partial h(\bar{\mu}_t)}{\partial \bar{\mu}_t} = \left(\frac{1}{q}\right)\begin{pmatrix} -\sqrt{q}\,\delta_x & -\sqrt{q}\,\delta_y & 0 & -\sqrt{q}\,\delta_x & \sqrt{q}\,\delta_y \\ \delta_y & -\delta_x & -q & -\delta_y & \delta_x \end{pmatrix}$$

Now map it to the high dimensional space.

$$H_t^i = {}^{low}H_t^i\, F_{x,j}$$

$$\Downarrow$$

$$F_{x,j} = \begin{array}{c} \overbrace{\begin{array}{ccc} x & y & \theta \end{array}}^{\text{maps}} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \cdots 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 1 & 0 & 0 \cdots 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 1 & 0 \cdots 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 0 & 0 \cdots 0 & 1 & 0 & 0 \cdots 0 \\ 0 & 0 & 0 & \underbrace{0 \cdots 0}_{2j-2} & \underbrace{0\ \ 1}_{\downarrow} & & \underbrace{0 \cdots 0}_{2N-2} \end{pmatrix} \end{array}$$

landmarks 1 to j-1

landmarks which are considered (here j^{th} landmark)

other landmarks

So, we can compute 4th step

$$K_t = \bar{\Sigma}_t H_t^T \left(H_t \bar{\Sigma}_t H_t^T + Q_t\right)^{-1}$$

$$\mu_t = \bar{\mu}_t + K_t\left(z_t - h(\bar{\mu}_t)\right)$$

$$\Sigma_t = \bar{\Sigma}_t * (1 - K_t H_t)$$

DONE ✔

Overview of Correction Step

(6) $Q_t = \begin{pmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\phi^2 \end{pmatrix}$

(7) for all observed features $z_t^i = (r_t^i, \phi_t^i)^T$ do

(8) $j = c_t^i$

(9) if landmark j never seen before

(10) $\begin{pmatrix} \bar{\mu}_{j,x} \\ \bar{\mu}_{j,y} \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{t,x} \\ \bar{\mu}_{t,y} \end{pmatrix} + \begin{pmatrix} r_t^i \cos(\phi_t^i + \bar{\mu}_{t,\theta}) \\ r_t^i \sin(\phi_t^i + \bar{\mu}_{t,\theta}) \end{pmatrix}$

11. end if

$$\delta = \begin{pmatrix} \delta_x \\ \delta_y \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{j,x} - \bar{\mu}_{t,x} \\ \bar{\mu}_{j,y} - \bar{\mu}_{t,y} \end{pmatrix}$$

$$q = \delta^T \delta$$

$$\hat{z}_t^i = \begin{pmatrix} \sqrt{q} \\ \operatorname{atan2}(\delta_y, \delta_x) - \bar{\mu}_{t,\theta} \end{pmatrix}$$

Compute $F_{x,j}$, $H_t^i = {}^{\text{low}}H_t^i \times F_{x,j}$

K_t^i, $\bar{\mu}_t$, $\bar{\Sigma}_t$ return μ_t, Σ_t