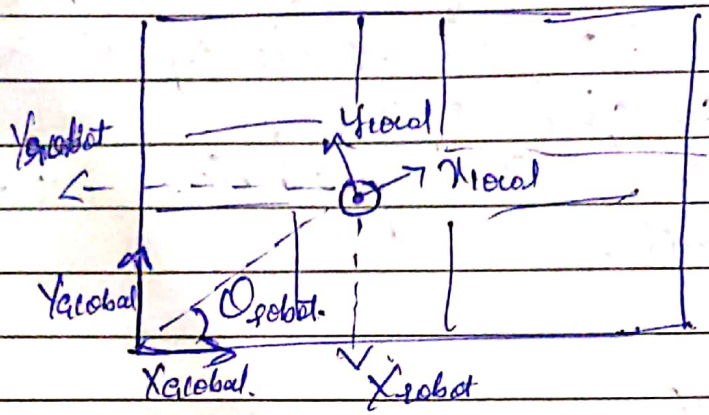
4) Particle filter



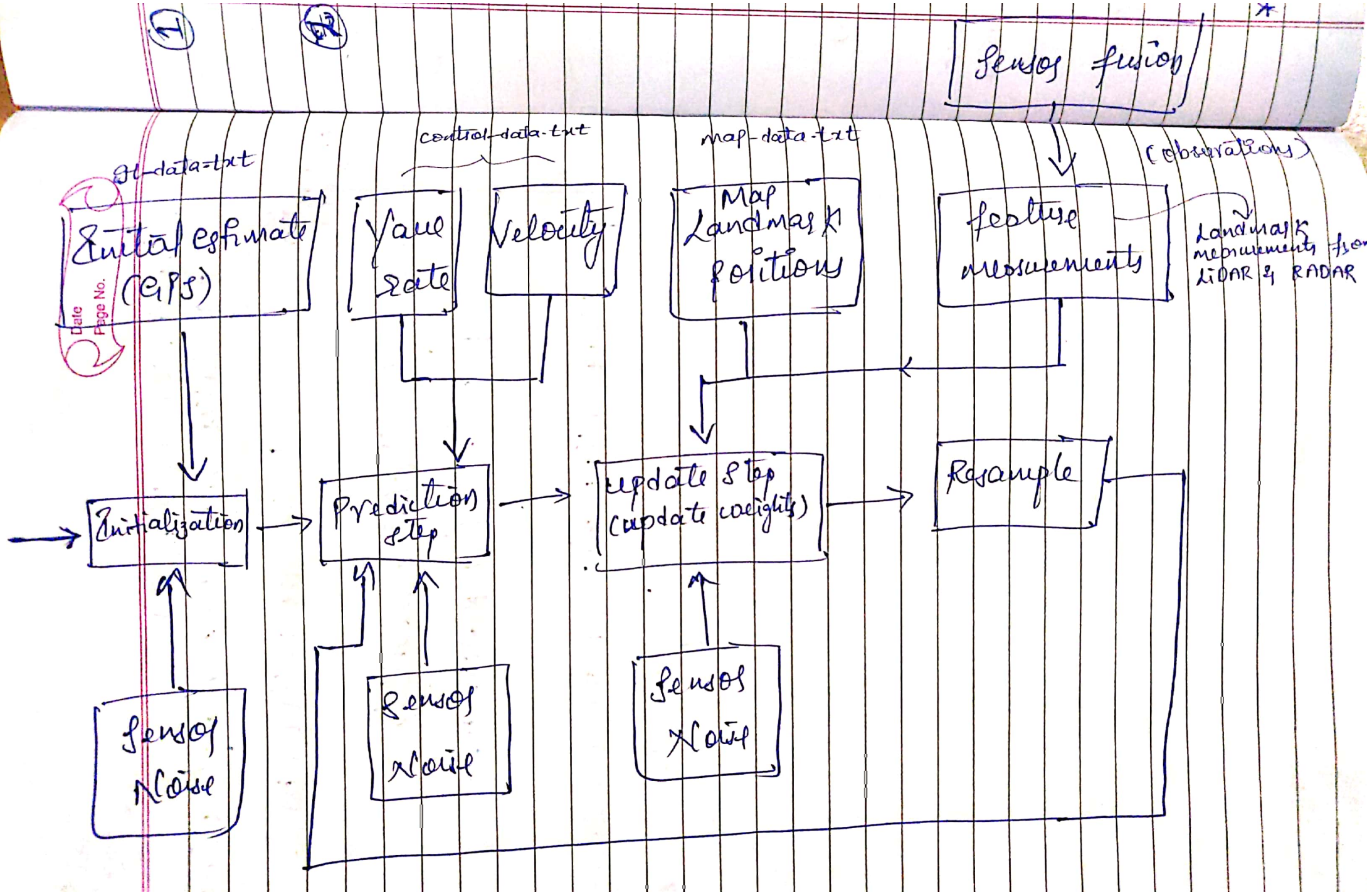
Vector of each particle = $\begin{bmatrix} x_{particle} \\ y_{particle} \\ \theta_{particle} \\ Weight \end{bmatrix}$

Where, Weight = robot actual pose − Particle predicted pose

✓ Importance of particle depends on weight.
✓ Bigger the weight of particle, more accurate it is.

✓ Particle with large weights are more likely to survive during resampling process, where as other more likely to die.

✓ finally after many iterations & resampling particles converge & estimate robot pose.



Date
Page No.

Particle filter Pseudocode (GPS)

Particle filter(X_{t-1}, u_t, z_t)
- X_{t-1} → previous belief
- z_t → sensor measurement
- u_t → Actuation (Yaw rate, command velocity)

Initialization ← $\bar{X}_t = X_t = \emptyset$

we estimate our position from sensor (GPS or i/p, later we refine this estimate (Lidar) to localize our vehicle.

for $m = 1$ to M:

Prediction ← Sample $x_t^{[m]} \sim p(x_t | u_t, x_{t-1}^{[m]})$

we add control i/p's (Yaw rate, velocity) to all the particles.

Update ←
$$w_t^{[m]} = p(z_t | x_t^{[m]})$$
$$\bar{X}_t = \bar{X}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$$

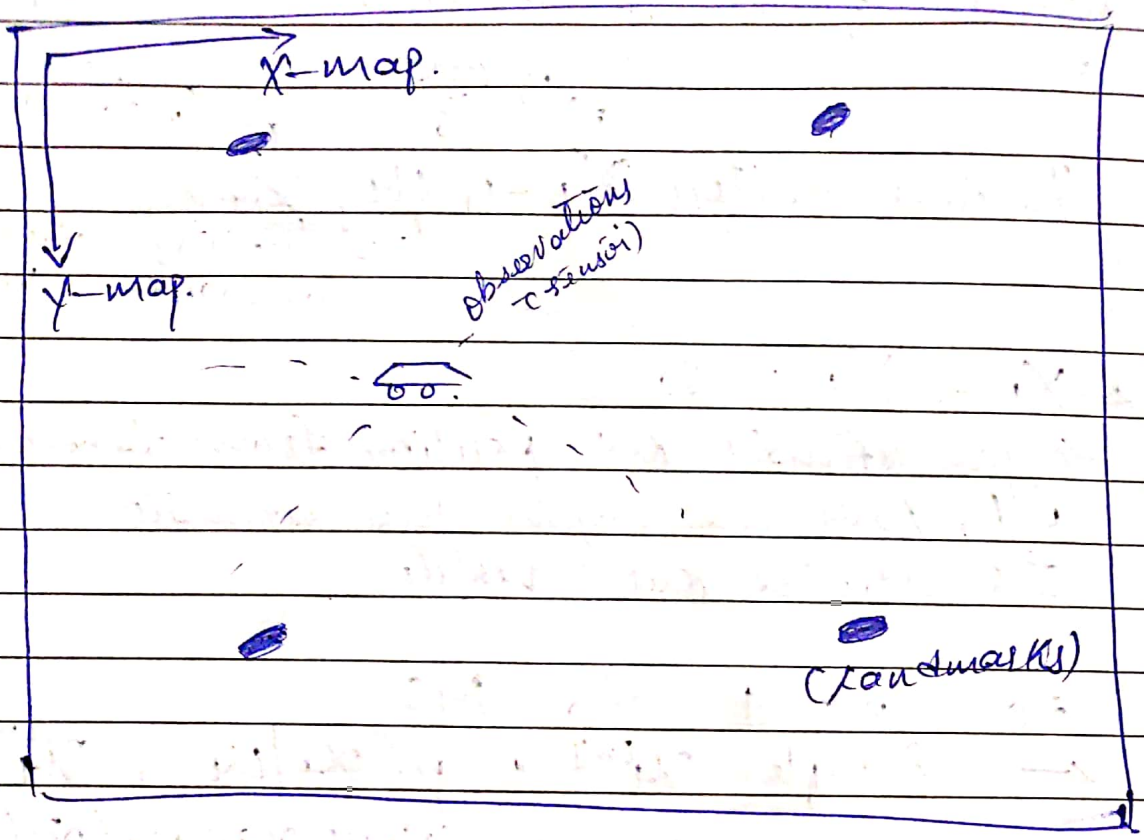
we update our particle weights using map landmark positions and feature measurements

end for

Resampling ←
for $m = 1$ to M:
draw i with probab $\propto w_t^{[i]}$
add $x_t^{[i]}$ to X_t
end for.

return X_t

returns new particle set. Now we have refined estimate of the vehicles position based on i/p evidence



Known.

$z_{1:t}$ → observations from sensors (range measurements, bearing, images --)

$u_{1:t}$ → Controls (Yaw, pitch, Yaw rate, velocity ---)

(Landmarks) m → map (grid maps, feature maps --)

Unknown. x_t → represents the pose (position (x, y) + orientation (θ))

$$\text{belief}(x_t) = p(x_t | z_{1:t}, u_{1:t}, m)$$

Date
Page No.

(1) Initialization

We Initialize our particles using GPS i/p, this step is impacted by noise.

(2) Prediction step.

We predict where the vehicle will be at the next time step based on yaw rate & velocity.

Equations for updating position (x, y) and yaw angle (θ)

$$x_f = x_0 + \frac{v}{\dot{\theta}} [\sin(\theta_0 + \dot{\theta}(dt)) - \sin(\theta_0)]$$

$$y_f = y_0 + \frac{v}{\dot{\theta}} [\cos(\theta_0) - \cos(\theta_0 + \dot{\theta}(dt))]$$

$$\theta_f = \theta_0 + \dot{\theta}(dt)$$

where,

(x_0, y_0) → Initial position in x & y direction (m)

(x_f, y_f) → Final " " " " (m)

v → Velocity (m/s)

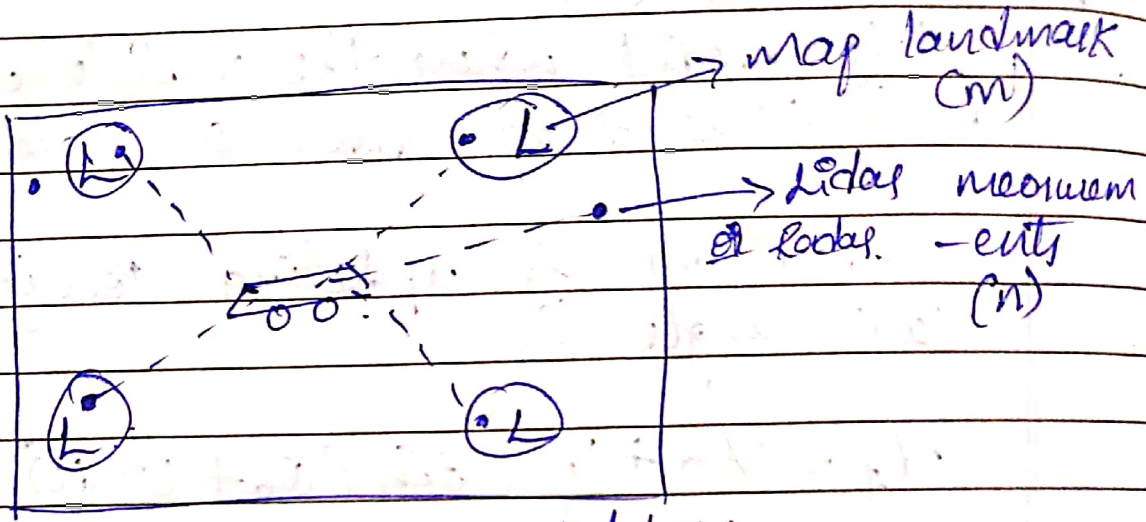
(θ_0, θ_f) → Initial & Final yaw angle.

$\dot{\theta}$ → Yaw rate

* Yaw rate & Velocity are control i/p's.

Sensor Fusion

Data association problem → It is the problem of solving landmarks measurements to the objects in the real-world like map landmarks.



we pick the nearest neighbours to the landmarks.

③ update step
we update particle measurements based on LiDAR and RADAR readings of landmarks.

feature measurements → Measurement from fused sensor data of LiDAR & RADAR sensors

$$p(x_t|z_{1:t}) = p(z_t|x_t) \cdot p(x_t|z_{1:t-1})$$

where, z_t = sensor measurement @ time t
x_t = state at time t

Date
Page No.

✓ we update weights of each particle using multi-variate Gaussian probability density function for each measurements.

$$w = \prod_{i=1}^{m} \frac{\exp\left(-\frac{1}{2}(x_i - H_i)^T \Sigma^{-1} (x_i - H_i)\right)}{\sqrt{|2\pi\Sigma|}}$$

x_i = measurement of i^{th} particle. (landmark)
H_i = predicted measurement of i^{th} particle.
Σ = Covariance of measurements

• ✓ Landmark measurement (or sensor measurement) (from sensors)

→ for the map landmark

m = total no. of measurements for 1 particle

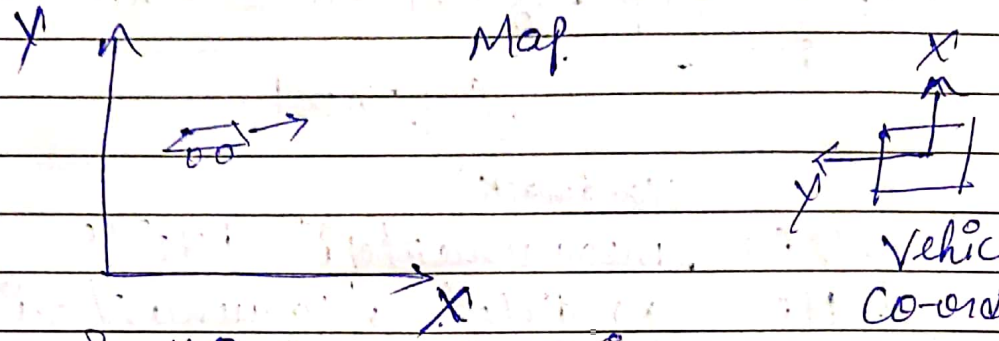
Σ = Covariance matrix

Eg: for LiDAR, $\Sigma = \begin{bmatrix} \sigma_{px}^2 & 0 \\ 0 & \sigma_{py}^2 \end{bmatrix}$

for RADAR, $\Sigma = \begin{bmatrix} \sigma_\rho^2 & 0 & 0 \\ 0 & \sigma_\nu^2 & 0 \\ 0 & 0 & \sigma_\phi^2 \end{bmatrix}$

* map landmarks will be in map's coordinate
* So, convert landmark measurements (i.e sensor measurements) from vehicle co-ordinate to map's coordinate with respect to our particle.

✓ Transform Landmark (sensor) measurements from vehicle's coord to map's coordinates using Homogeneous Transformation function



In this case $\theta = -90^\circ$ (sensor measured)

(x_c, y_c) → observation co-ordinates

(x_p, y_p) → particle "

θ → rotation angle (in our case -90)

(x_m, y_m) → map's co-ordinates

$$\begin{bmatrix} x_m \\ y_m \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & x_p \\ \sin\theta & \cos\theta & y_p \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_c \\ y_c \\ 1 \end{bmatrix}$$

$$\Rightarrow x_m = x_p + (\cos\theta \times x_c) - (\sin\theta \times y_c)$$

$$\Rightarrow y_m = y_p + (\sin\theta \times x_c) + (\cos\theta \times y_c)$$

✓ Next, associate each transformed observation with map landmark positions

To do this we must associate the closest landmark to each transformed observation

Date
Page No.

✓ Calculating the particle's final weight

$$p(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-\left(\frac{(x-\mu_x)^2}{2\sigma_x^2} + \frac{(y-\mu_y)^2}{2\sigma_y^2}\right)}$$

(μ_x, μ_y) = nearest map landmark.

(x, y) = Transformed landmark measurements (from sensor)

✓ Calculating errors (performance measurement of filter)

$$\text{Error}_{\text{weighted}} = \frac{\sum_{i=1}^{M} w_i \sqrt{|P_i - g|}}{\sum_{i=1}^{M} w_i}$$

& here we consider weights which we've calculated previously (ie not $p(x,y)$ 😉)

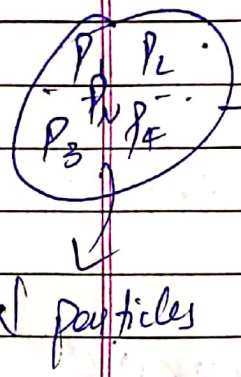
④ Resampling

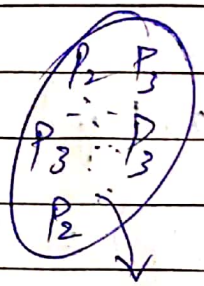
	Particles	Poses	weights	Normalized weights.
N (1000)	P_1	(x_1, y_1, θ_1)	w_1	$\alpha_1 = w_1 / W$
	P_2	(x_2, y_2, θ_2)	w_2	$\alpha_2 = w_2 / W$
	:	:	:	:
	P_N	(x_N, y_N, θ_N)	w_N	$\alpha_N = w_N / W$
			$W = \sum_i w_i$	$\sum_i \alpha_i = 1$

Randomly draw new particles from old one with replacement in proportion to importance weight.

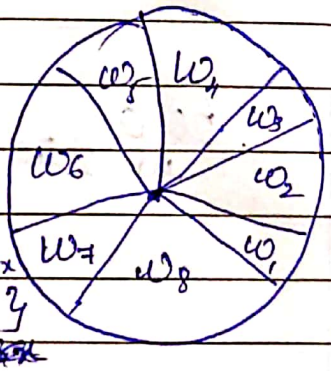
Pick particles with high probability (α)

Resampling 'N' times





Resampling wheel

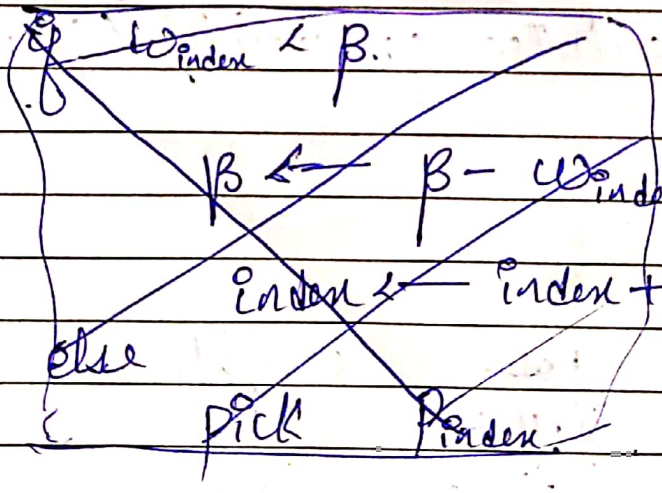


index = random [1 - - N]

$\beta = 0$.

for i = 1 - - N

$\beta \leftarrow \beta + \text{random}[0 - \cdot 2 * w_{max}]$



Step - Command Resampling Power.

while w[index] < beta:

beta = beta - w[index]

index = index + 1

Select p[index]

✓ init(x, y, theta, std[ ])
(GPS+IMU)
(gt_data.txt).

✓ prediction(dt, velocity, yaw_rate, vel & yaw_rate uncert)
IMU
(control_data.txt).

* gt_data.txt → GPS+IMU (x, y, θ) in Global Co-ordinates

* control_data.txt → Velocity & yaw rate (IMU)

* map_data.txt → .map landmark (x, y) & map_id in Global Co-ordinates

* Observation folder →
(observation_ _ .txt)

contains local coordinates (x, y), i.e car to the landmark measurements from LiDAR & RADAR fused data.

* only map_data.txt → structured in map.h

rest all are structured in helper_functions.h

Date
Page No.

✓ dataAssociation (predicted landmarks, observations)
(Observation folder)
observation_..txt.

✓ updateWeights (sensor_range, std_landmarks[], landmarksObs observations, map_landmarks)
(map_data.txt)

✓ In observation folder
observation_..txt → contains (x, y) vehicle or local coordinates, measurement from vehicle to landmarks.
(obtained from LIDAR & RADAR fused data)

→ It contains multiple coordinates in each .txt file (for data association purpose)

✓ weighted_mean_error (gt_x, gt_y, gt_theta)
gt_data.txt.
(GPS+IMU)