A
REPORT
ON

EKF AND PARTICLE FILTER BASED SIMULTANEOUS LOCALIZATION AND MAPPING OF MOBILE ROBOTS

Submitted in partial fulfillment of the requirements for

SUMMER RESEARCH INTERNSHIP

(ROBOTICS AND MACHINE INTELLIGENCE LAB)

By

DEEPESHWAR KUMAR

(U15EE054)

: Supervisor:

PROF. G.C. NANDI

DEPARTMENT OF INFORMATION TECHNOLOGY

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY

ALLAHABAD

May – July 2018

ABSTRACT

The importance of environment modelling as a prerequisite for the navigation of mobile robots cannot be understand. It is essential for a mobile robot to have the ability to autonomously build a model of environment, as human intervention is not practical in all circumstances. A robot has to address both the mapping and localization problems in order to solve the map learning problem. Due to coupling between these two tasks, autonomous map learning has been challenging. This report is primarily focused on implementation of SLAM based on Extended Kalman Filter (EKF) and Particle Filter (Fast SLAM).

The mapping and localization problem, called Simultaneous Localization and Mapping (SLAM) is defined as the process of concurrently building a map of the environment while using this map to obtain improved estimates of the robot's location. The SLAM algorithm begins with the mobile robot in unknown location, with no a priori knowledge of the map. The robot then uses a sensor to model the visible environment and simultaneously computes an estimate of its own position. Continuing in motion, the mobile robot eventually builds a model of the environment which is useful for navigation.

TABLE OF CONTENT

CHAPTER1: **Robot Motion Model**

1.1 Introduction

A differential drive robot is being used here for all the implementation of the algorithms. Robot is having lidar sensor on top for obtaining the measurement data from the environment and two encoder mounted motor for odometry data calculation. Arena is having cylindrical landmarks as obstacles. There is red circle marked on lidar top on the robot. While robot is moves in the arena, it is tracked by the overhead camera and gives us the reference trajectory. This is the external measurement system, not being part of the robot system, that gives us the reference trajectory of the robot position.

1.2 Odometry Description

Motion model of the robot is described here:

Description of the some of the important symbol is listed below:

$$R \rightarrow Instantaneous\ radius\ of\ curvature$$

$$l \rightarrow Distance\ covered\ by\ left\ wheel$$

$$r \rightarrow Distance\ covered\ by\ right\ wheel$$

$$w \rightarrow width\ of\ the\ robot$$

$$\alpha \rightarrow Turning\ angle$$

Steps for finding new pose from old pose and after control data being applied:

If $\boldsymbol{l} != \boldsymbol{r}$

α = (r - l)/w

R = l/ α

P = current pose of robot (x, y)

θ = current heading of robot

Instantaneous center of rotation:

$$C = \begin{bmatrix} Cx \\ Cy \end{bmatrix}$$

$$\begin{bmatrix} Cx \\ Cy \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} - (R + w/2) * \begin{bmatrix} sin\theta \\ -cos\theta \end{bmatrix}$$

New pose:

$$P' = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} Cx \\ Cy \end{bmatrix} + (R + w/2) * \begin{bmatrix} sin(\theta + \alpha) \\ -cos(\theta + \alpha) \end{bmatrix}$$

New heading:

$$\theta' = (\theta + \alpha) \bmod 2\pi$$

If $\boldsymbol{l = r}$:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + l * \begin{bmatrix} sin\theta \\ cos\theta \end{bmatrix}$$

$$\theta' = \theta$$

Here, width of the robot, w = 150 mm

Ticks to distance covered by wheel: 1count corresponds to 0.349 mm

Global frame is the coordinate system of lidar tracked by overhead camera.

Robot coordinate system is different from tracked coordinate system

Reference trajectory is obtained by tracking the red dots on lidar by overhead camera

And other trajectory is computed as pose from ticks using the motor ticks

- Transforming the robot's coordinate system to lidar coordinate system:

$$\begin{bmatrix} x \\ y \end{bmatrix} \mathrel{+}= \begin{bmatrix} sin\theta \\ cos\theta \end{bmatrix} * scanner\ displacement$$

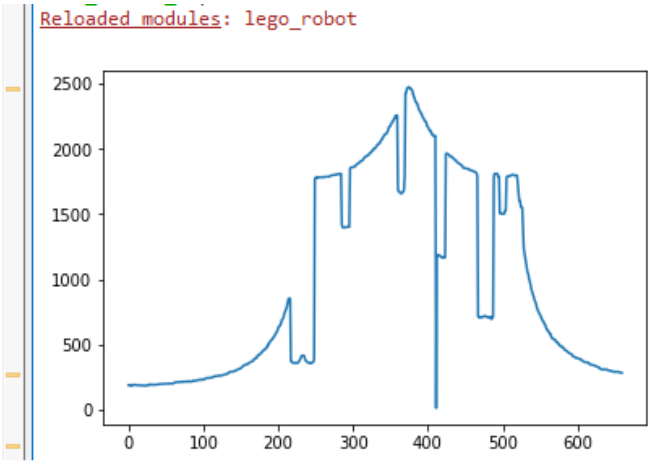
1.3 Laser Scan Description

Measurement of given landmarks using lidar:

There are 278 scans in whole trajectory and every scan have total of 660 beams.

Plotting a scan of the robot using matplotlib:

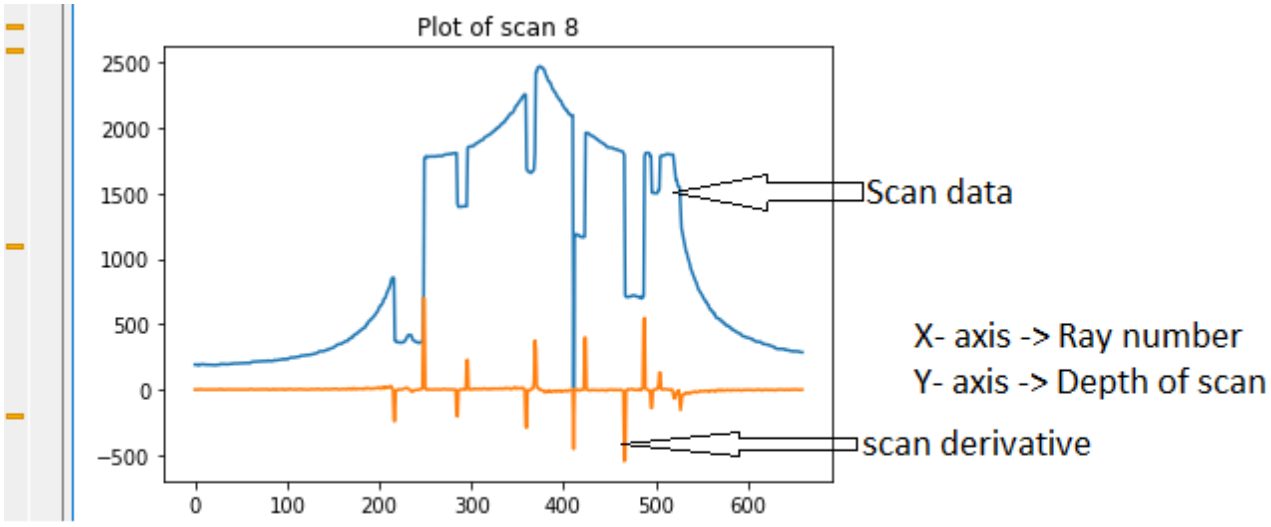
Plot of 8th scan:



X- axis -> Ray number(Total 660 beams)
Y- axis -> Depths of the scan

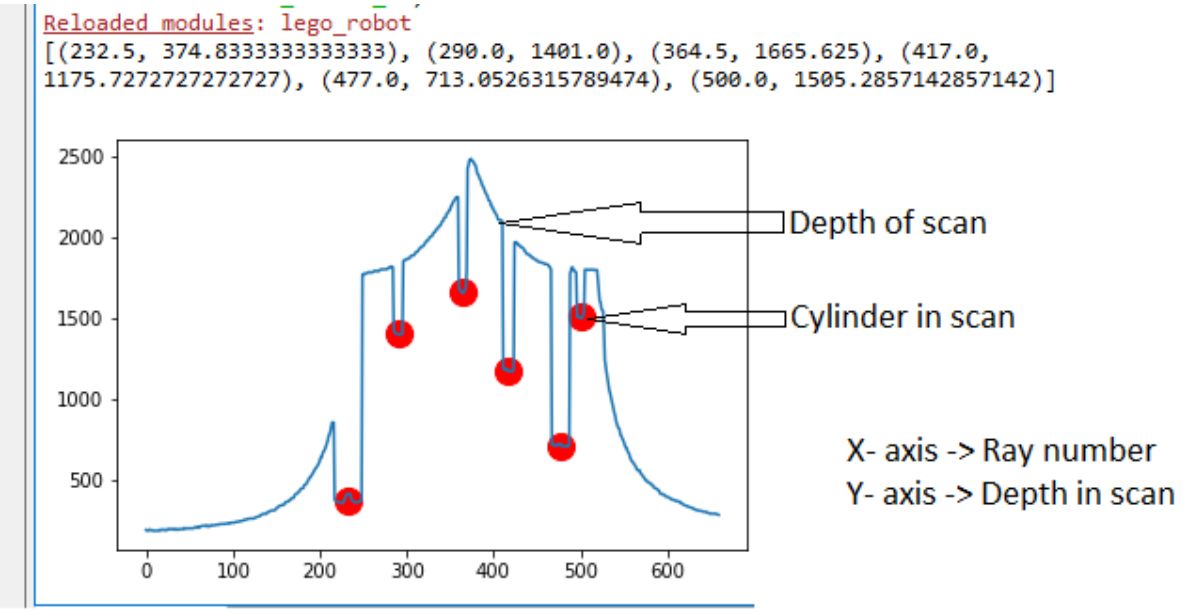
Spikes in the scan data depicts that there are some cylindrical landmarks in the range of laser scan. So, by finding the derivative of the scan data, we have idea of the landmarks in the scan. Also we need to ignore the invalid measurements.

Below plot depicts the scan derivative:

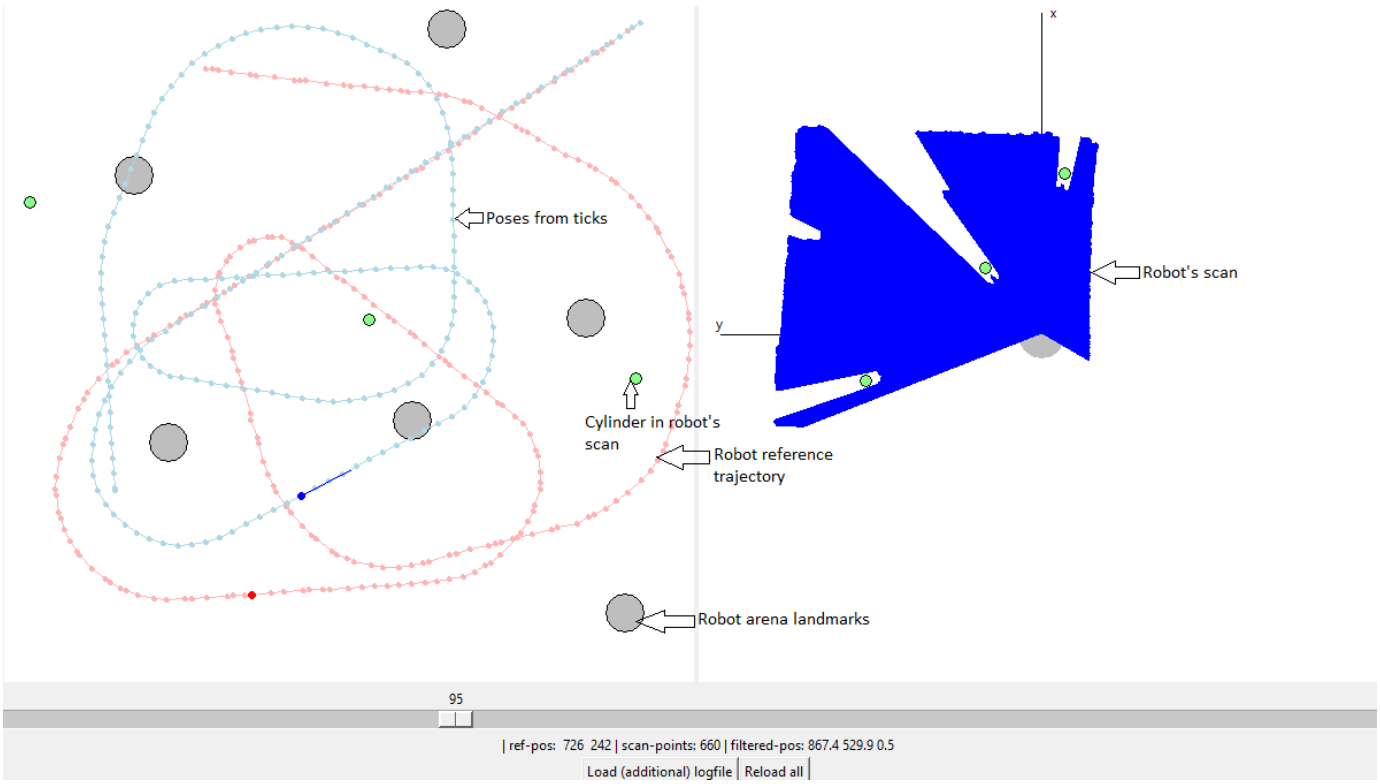


For each cylinder in the scan, now we have calculated its average ray number and average depth in the scan.

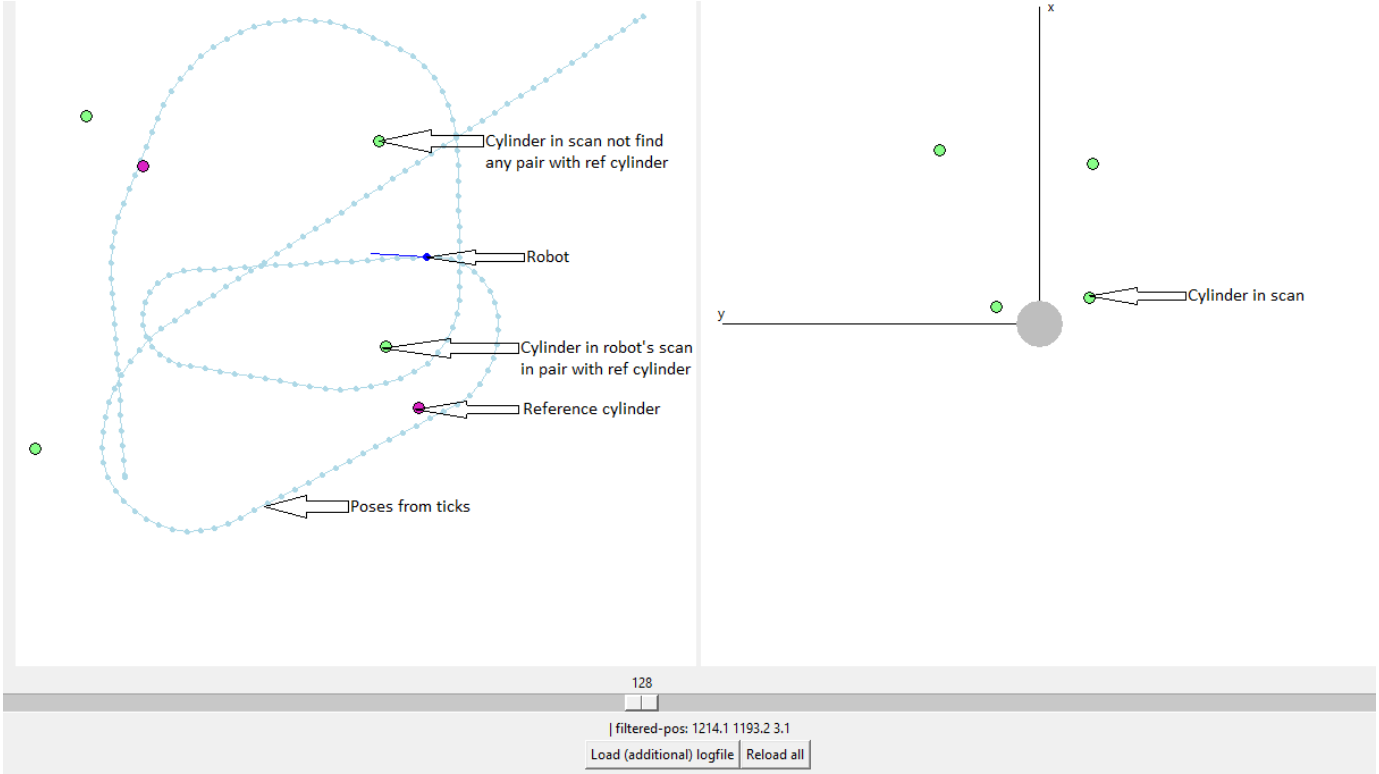
Average ray number and average depth for each cylinder in the scan is depicted below:



Given detected cylinder coordinate in terms of (Average ray number, average depth/distance), we have calculated its cartesian coordinates (x, y). This is polar to cartesian conversion with an added offset.



For each cylinder in the scan, we have found its cartesian coordinates in the world coordinate system. Then, we will find the closest point in the reference cylinder dataset and output it. Given a list of cylinders (points) and reference cylinders, for every cylinder we have find the closest reference cylinder and add the index pair (I, j), where I is the index of cylinder, and j is the index of the reference cylinder, to the result list.



CHAPTER2: **KALMAN FILTER**

2.1 Motion Model

When $r\ !=\ l$:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \left(R + \frac{w}{2}\right) * \ (\sin(\theta + \alpha) - sin\theta) \\ \left(R\ + \frac{w}{2}\right) * (-\cos(\theta + \alpha) + cos\theta) \\ \alpha \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = g(x, y,\ \theta,\ l,\ r)$$

$$X' = \ g(X, u)$$

When $r = l$:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} l * cos\theta \\ l * sin\theta \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = g(x, y,\ \theta,\ l,\ r)$$

$$X' = \ g(X, u)$$

2.2 Extended Kalman Filter Prediction step

EKF Prediction step:

$$\boldsymbol{X_t = g(X_{t-1}, u_t)}$$

$$\boldsymbol{\overline{\mu}_t = g(u_t, \mu_{t-1})}$$

$$\boldsymbol{\overline{\sum}_t = \ G_t \sum_{t-1} G_t^T + \ R_t}$$

$$g(x, y, \theta, l, r) = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \left(R + \frac{w}{2}\right) * \ (\sin(\theta + \alpha) - sin\theta) \\ \left(R\ + \frac{w}{2}\right) * (-\cos(\theta + \alpha) + cos\theta) \\ \alpha \end{bmatrix}$$

$$G_t \rightarrow Jacobian\ matrix\ of\ g = \frac{\partial g}{\partial X} = partial\ derivative\ of\ g\ matrix\ w.r.t\ states$$

$$G = \begin{bmatrix} \frac{\partial g_1}{\partial x} & \frac{\partial g_1}{\partial y} & \frac{\partial g_1}{\partial \theta} \\ \frac{\partial g_2}{\partial x} & \frac{\partial g_2}{\partial y} & \frac{\partial g_2}{\partial \theta} \\ \frac{\partial g_3}{\partial x} & \frac{\partial g_3}{\partial y} & \frac{\partial g_3}{\partial \theta} \end{bmatrix}$$

For $r\ != l$:

$$G = \begin{bmatrix} 1 & 0 & \left(R + \frac{w}{2}\right) * (\cos(\theta + \alpha) - cos\theta) \\ 0 & 1 & \left(R + \frac{w}{2}\right) * (\sin(\theta + \alpha) - sin\theta) \\ 0 & 0 & 1 \end{bmatrix}$$

For $r = l$:

$$G = \begin{bmatrix} 1 & 0 & -l * sin\theta \\ 0 & 1 & l * cos\theta \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_t = V_t \sum_{control} V_t^T$$

$$\sum_{control} = \begin{bmatrix} \sigma_l^2 & 0 \\ 0 & \sigma_r^2 \end{bmatrix}$$

$$\sigma_l^2 = (\alpha_1 * l)^2 + (\alpha_2 * (l - r))^2$$

$$\sigma_r^2 = (\alpha_1 * r)^2 + \left(\alpha_2 * (l - r)\right)^2$$

$$\alpha_1 = control\ motion\ factor$$

$$\alpha_2 = control\ turning\ factor$$

$$V_t = \frac{\partial g}{\partial control} = Partial\ derivative\ of\ g\ matrix\ w.r.t\ control$$

$$V_t = \frac{\partial g}{\partial control} = \begin{bmatrix} \frac{\partial g_1}{\partial l} & \frac{\partial g_1}{\partial r} \\ \frac{\partial g_2}{\partial l} & \frac{\partial g_2}{\partial r} \\ \frac{\partial g_3}{\partial l} & \frac{\partial g_3}{\partial r} \end{bmatrix}$$

For $r\ != l$:

$$\frac{\partial g_1}{\partial l} = \frac{w * r}{(r-l)^2} * (\sin(\theta+\alpha) - sin\theta) - \frac{r+l}{2*(r-l)} * cos(\theta+\alpha)$$
$$\frac{\partial g_2}{\partial l} = \frac{w * r}{(r-l)^2} * (-\cos(\theta+\alpha) + cos\theta) - \frac{r+l}{2*(r-l)} * sin(\theta+\alpha)$$

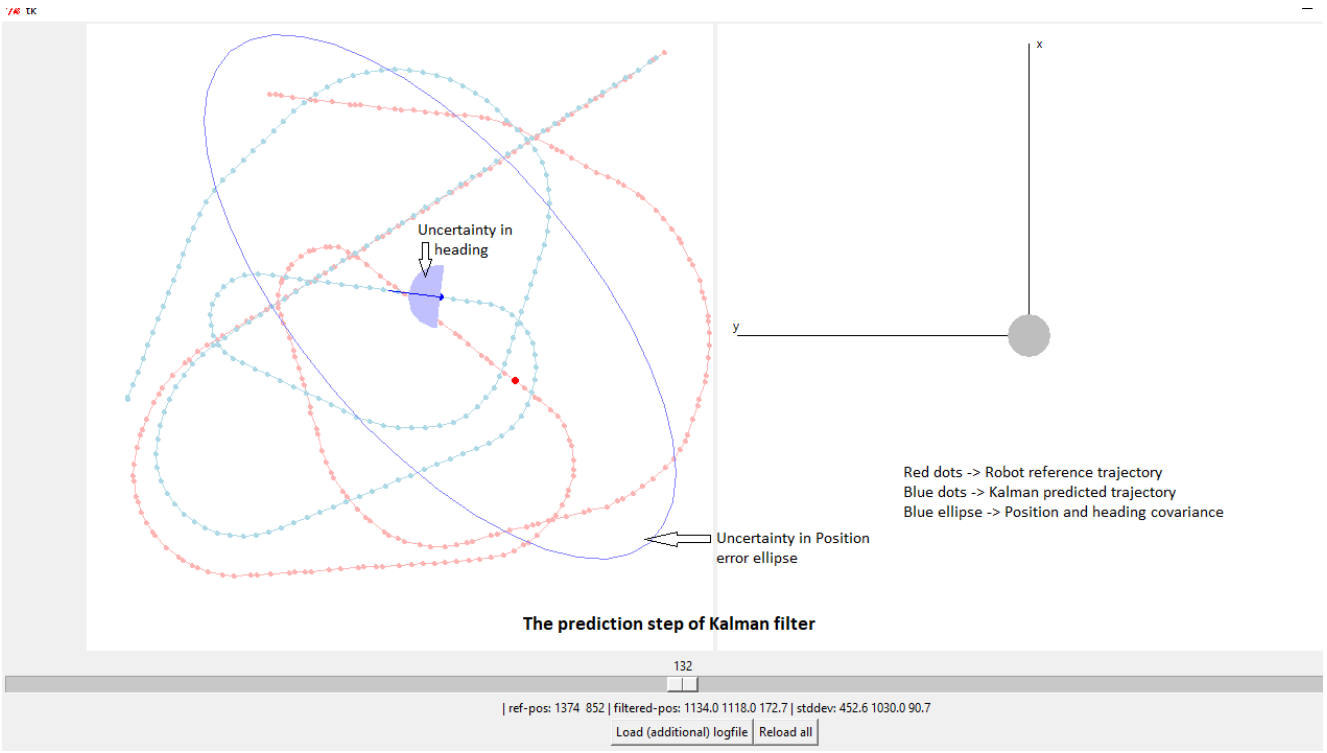
$$\frac{\partial g_3}{\partial l} = -\frac{1}{w}$$

$$\frac{\partial g_1}{\partial r} = \frac{-(w * r)}{(r-l)^2} * (\sin(\theta+\alpha) - sin\theta) + \frac{r+l}{2*(r-l)} * cos(\theta+\alpha)$$
$$\frac{\partial g_2}{\partial r} = \frac{-(w * r)}{(r-l)^2} * (-\cos(\theta+\alpha) + cos\theta) + \frac{r+l}{2*(r-l)} * sin(\theta+\alpha)$$

$$\frac{\partial g_3}{\partial r} = \frac{1}{w}$$

For $r = l$:

$$\frac{\partial g_1}{\partial l} = \frac{1}{2} * (cos\theta + \frac{l}{w} sin\theta)$$
$$\frac{\partial g_2}{\partial l} = \frac{1}{2} * (sin\theta - \frac{l}{w} cos\theta)$$
$$\frac{\partial g_3}{\partial l} = -\frac{1}{w}$$
$$\frac{\partial g_1}{\partial r} = \frac{1}{2} * (cos\theta - \frac{l}{w} sin\theta)$$
$$\frac{\partial g_2}{\partial r} = \frac{1}{2} * (sin\theta + \frac{l}{w} cos\theta)$$
$$\frac{\partial g_3}{\partial r} = \frac{1}{w}$$



2.3 Extended Kalman Filter Correction step

Correction step of Kalman Filter

$$\boldsymbol{z_t = h(X_t)}$$

$$K_t = \bar{\Sigma}_t \, H_t^T (H_t \, \bar{\Sigma}_t \, H_t^T + Q_t)^{-1}$$
$$\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$$
$$\Sigma_t = (I - K_t \, H_t) \, \bar{\Sigma}_t$$

- Transforming robot's body coordinate system into lidar coordinate system as all the measurement data is taken by lidar

$$d = scanner\ displacement$$

$$\begin{bmatrix} x_l \\ y_l \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + d * \begin{bmatrix} cos\theta \\ sin\theta \end{bmatrix}$$

- Consider a landmark positioned at $(x_m,\ y_m)$ is seen by lidar. So, the range and bearing angle$(alpha)$ of the landmark is given by:

$$range = \sqrt{(x_m - x_l)^2 + (y_m - y_l)^2}$$

$$alpha = \text{atan}\left(\frac{y_m - y_l}{x_m - x_l}\right) - \theta$$

$$Z = \begin{bmatrix} range \\ alpha \end{bmatrix} = h(x, y, \theta)$$

$$x_l = x + d * cos\theta$$

$$y_l = y + d * sin\theta$$

$$\Delta x = x_m - x_l$$

$$\Delta y = y_m - y_l$$

$$H = Jacobian\ matrix\ of\ h(x, y, \theta) = \frac{\partial h}{\partial state}$$

$$H = partial\ derivative\ of\ h\ matrix\ w.r.t\ state$$

$$H = \begin{bmatrix} \frac{\partial range}{\partial x} & \frac{\partial range}{\partial y} & \frac{\partial range}{\partial \theta} \\ \frac{\partial alpha}{\partial x} & \frac{\partial alpha}{\partial y} & \frac{\partial alpha}{\partial \theta} \end{bmatrix}$$

$$H = \begin{bmatrix} \frac{-\Delta x}{range} & \frac{-\Delta y}{range} & \frac{d}{range}(\Delta x * sin\theta - \Delta y * cos\theta) \\ \frac{\Delta y}{range^2} & \frac{-\Delta x}{range^2} & \frac{-d}{range^2}(\Delta x * cos\theta + \Delta y * sin\theta) - 1 \end{bmatrix}$$

$For\ calculation\ of\ Kalman\ Gain\ K_t$:

$$K_t = \overline{\sum}_t H_t^T\ (H_t \overline{\sum}_t H_t^T + Q_t)^{-1}$$

Here, Q_t = Measurement covariance matrix

$$Q_t = \begin{bmatrix} \sigma_{range}^2 & 0 \\ 0 & \sigma_{alpha}^2 \end{bmatrix}$$

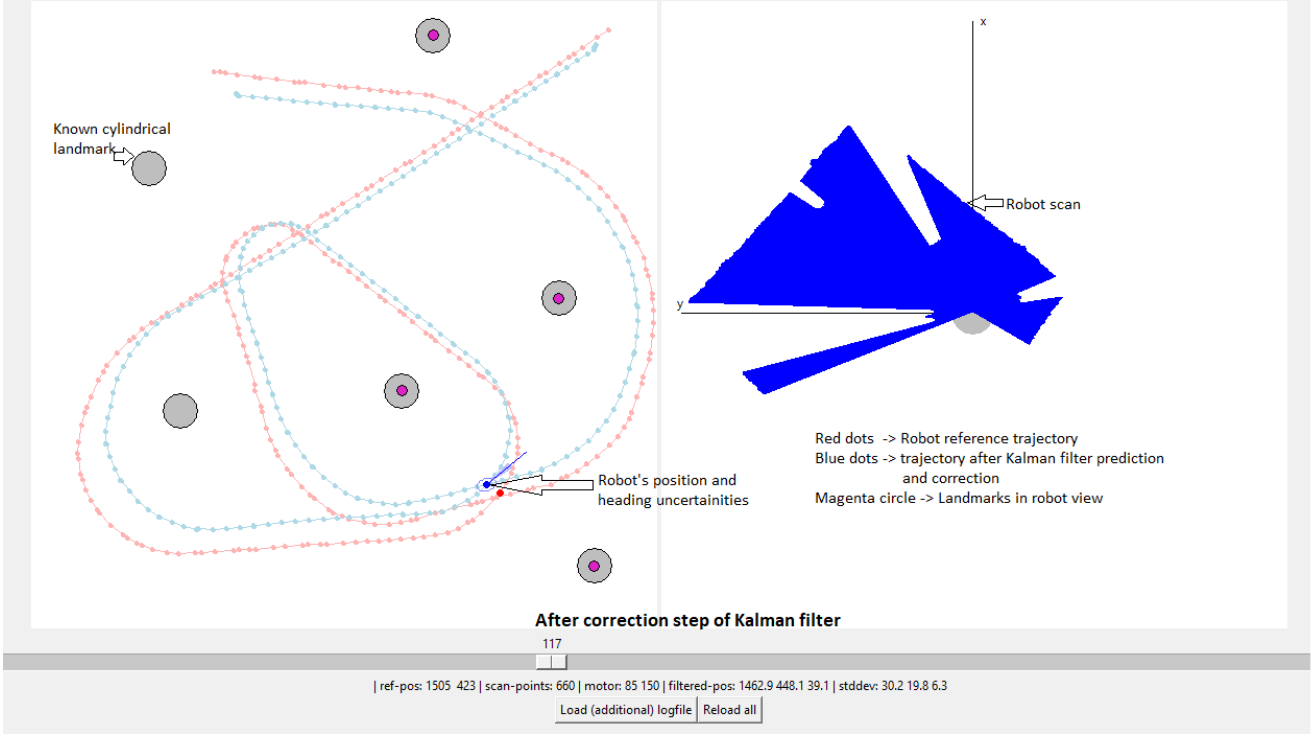
$$\sigma_{range}^2 = Distance\ measurement\ variance$$

$$\sigma_{alpha}^2 = Angle\ measurement\ variance$$

Corrected mean and covariance will be calculated as:

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

$$\sum_t = (I - K_t H_t) \overline{\sum}_t$$



CHAPTER3: **EKF SLAM**

3.1 Description of SLAM Problem

Given:

➔ The robot's controls

$$u_{1:T} = \{u_1, u_2, u_3 \dots, u_T\}$$

➔ Observations

$$z_{1:T} = \{z_1, z_2, z_3 \dots, z_T\}$$

Wanted:

➔ Map of the environment

$$m$$

➔ Path of the robot

$$x_{0:T} = \{x_0, x_1, x_2 \dots, x_T\}$$

Bayes Filter:

- Recursive filter with prediction and correction step

Prediction:

$$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\ bel(x_{t-1})\, dx_{t-1}$$

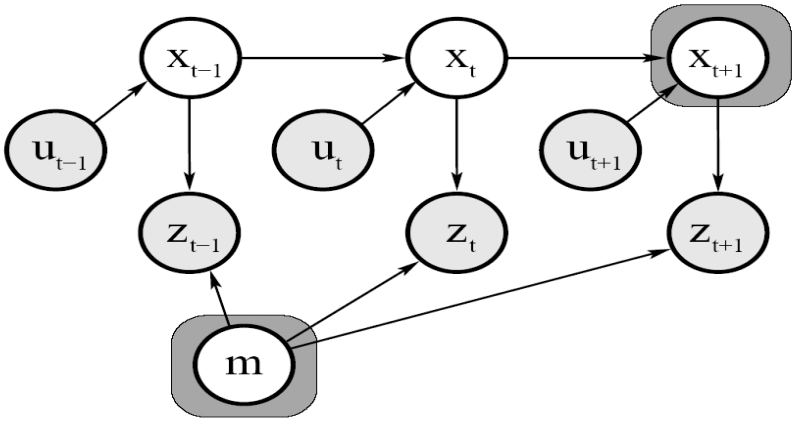
Correction:

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

EKF for online SLAM

The Kalman filter provides a solution to the online SLAM problem, i.e.

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



3.2 EKF Algorithm

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

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

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

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

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

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

return μ_t, Σ_t

EKF SLAM:

Now we will estimate the robot's pose as well as position of landmarks in the environment.

State space is:

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

State representation:

Map with n landmarks: $(3 + 2 * n)$ dimension state space

$$\underbrace{\begin{pmatrix} x_R \\ m_1 \\ \vdots \\ m_n \end{pmatrix}}_{\mu} \quad \underbrace{\begin{pmatrix} \Sigma_{x_R x_R} & \Sigma_{x_R m_1} & \cdots & \Sigma_{x_R m_n} \\ \Sigma_{m_1 x_R} & \Sigma_{m_1 m_1} & \cdots & \Sigma_{m_1 m_n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{m_n x_R} & \Sigma_{m_n m_1} & \cdots & \Sigma_{m_n m_n} \end{pmatrix}}_{\Sigma}$$

3.3 Prediction step of EKF SLAM

Main goal is to update state space based on robot's motion. Robot motion in plane is given by:

Motion model is given by:

When $r \; != \; l$:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \left(R + \frac{w}{2}\right) * \; (\sin(\theta + \alpha) - sin\theta) \\ \left(R \; + \frac{w}{2}\right) * (-\cos(\theta + \alpha) + cos\theta) \\ \alpha \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = g(x, y, \; \theta, \; l, \; r)$$

$$X' = \; g(X, u)$$

When $r = l$:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} l * cos\theta \\ l * sin\theta \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = g(x, y, \; \theta, \; l, \; r)$$

$$X' = \; g(X, u)$$

We will map it to $(3 + 2 * n)$ dimension space.

$$\begin{bmatrix} x' \\ y' \\ \theta' \\ x_1 \\ y_1 \\ . \\ . \\ . \\ x_n \\ y_n \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \\ x_1 \\ y_1 \\ . \\ . \\ . \\ x_n \\ y_n \end{bmatrix} + \begin{bmatrix} \left(R + \frac{w}{2}\right) * \ (\sin(\theta + \alpha) - sin\theta) \\ \left(R \ + \frac{w}{2}\right) * (-\cos(\theta + \alpha) + cos\theta) \\ \alpha \\ 0 \\ 0 \\ . \\ . \\ . \\ 0 \\ 0 \end{bmatrix}$$

Where $(x_1, y_1), (x_2, y_2)$ …, (x_n, y_n) are the position of landmarks.

Update covariance:

The function g only affects the robot's motion and not the landmarks

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

$$G_t^x = Jacobian\ of\ the\ motion\ (3 \times 3)$$

$$I = Identity\ Matrix\ (2N \times 2N\ )$$

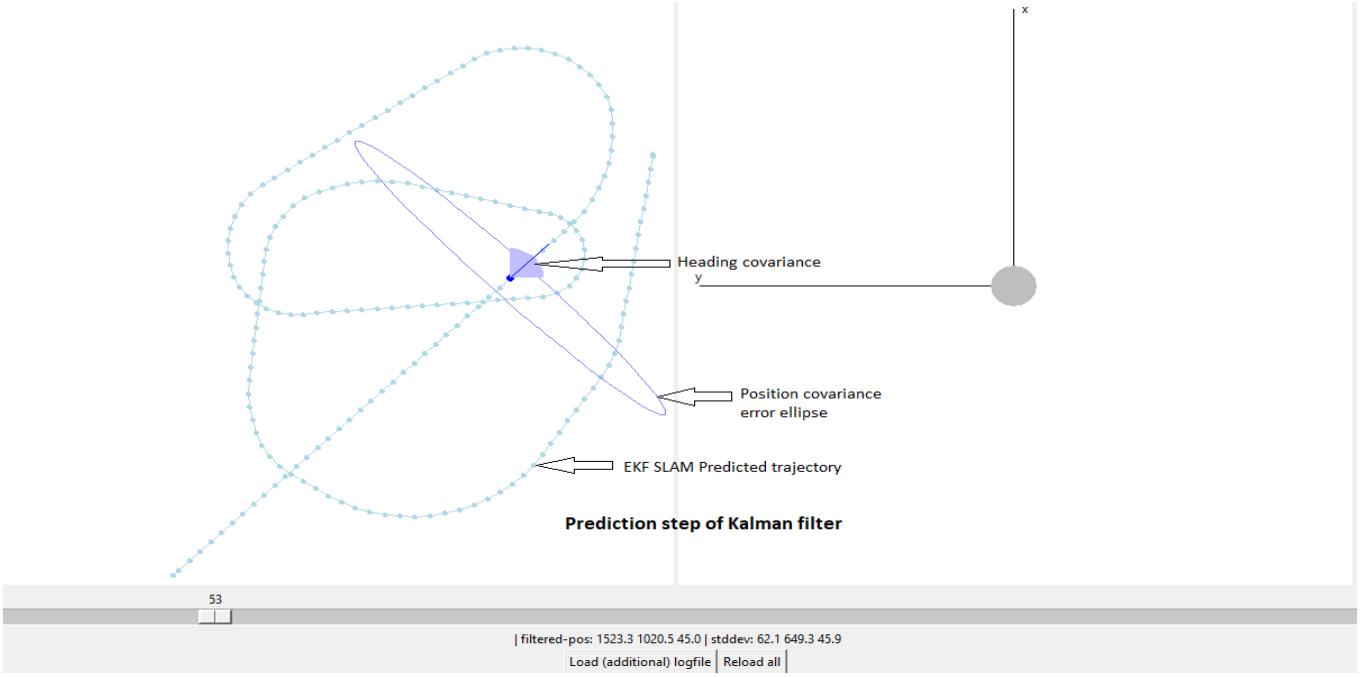
$$when\ l\ != r\text{:} \qquad G_t^x = \begin{bmatrix} 1 & 0 & \left(R \ + \frac{w}{2}\right) * (\cos(\theta + \alpha) - cos\theta) \\ 0 & 1 & \left(R + \frac{w}{2}\right) * \ (\sin(\theta + \alpha) - sin\theta) \\ 0 & 0 & 1 \end{bmatrix}$$

$$when\ l = r\text{:} \qquad G_t^x = \begin{bmatrix} 1 & 0 & -l * sin\theta \\ 0 & 1 & l * cos\theta \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_t = \begin{bmatrix} R_t^{old} & 0 \\ 0 & zeros \end{bmatrix}$$

$$R_t^{old} = V_t \sum_{control} V_t^T \ (3 \times 3)$$

$$zeros = zeros\ matrix\ (2N \times 2N)$$



3.4 Adding a Landmark

Enlarge the current state and covariance matrix to include one more landmark, which is given by its initial coordinates (x, y). Return the index of the newly added landmark. There is a need to augment the robot's state and covariance matrix. We will initialize the state with given initial coordinates and the covariance with 1e10 (as an approximation for infinity).

$$state = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \rightarrow after\ adding\ one\ landmark, state = \begin{bmatrix} x \\ y \\ \theta \\ x_1 \\ y_1 \end{bmatrix}$$

$$covariance = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{x\theta} \\ \sigma_{yx} & \sigma_y^2 & \sigma_{y\theta} \\ \sigma_{\theta x} & \sigma_{\theta y} & \sigma_\theta^2 \end{bmatrix} \rightarrow After\ adding\ one\ landmark,$$

$$covariance = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{x\theta} & 0 & 0 \\ \sigma_{yx} & \sigma_y^2 & \sigma_{y\theta} & 0 & 0 \\ \sigma_{\theta x} & \sigma_{\theta y} & \sigma_\theta^2 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{x1}^2 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{y1}^2 \end{bmatrix}$$

Initially assume $\sigma_{x1}^2\ and\ \sigma_{y1}^2$ tends to infinity.

3.5 Correction step of EKF SLAM

Correction step of Kalman filter

Get (x_m, y_m) of the landmark from the state vector corresponding to the called landmark index.

Observations matrix:

Initially

$$Z = \begin{bmatrix} range \\ alpha \end{bmatrix} = h(x, y, \theta)$$

Now after adding a landmark in the state:

$$h(x, y, \theta, x_m, y_m) \rightarrow H = \begin{bmatrix} \frac{\partial range}{\partial x} & \frac{\partial range}{\partial y} & \frac{\partial range}{\partial \theta} & \frac{\partial range}{\partial x_m} & \frac{\partial range}{\partial y_m} \\ \frac{\partial alpha}{\partial x} & \frac{\partial alpha}{\partial y} & \frac{\partial alpha}{\partial \theta} & \frac{\partial alpha}{\partial x_m} & \frac{\partial alpha}{\partial y_m} \end{bmatrix}$$

$$H = \begin{bmatrix} \frac{-\Delta x}{range} & \frac{-\Delta y}{range} & \frac{d}{range}(\Delta x * sin\theta - \Delta y * cos\theta) & \frac{\Delta x}{range} & \frac{\Delta y}{range} \\ \frac{\Delta y}{range^2} & \frac{-\Delta x}{range^2} & \frac{-d}{range^2}(\Delta x * cos\theta + \Delta y * sin\theta) - 1 & \frac{-\Delta y}{range^2} & \frac{\Delta x}{range^2} \end{bmatrix}$$

For n landmarks $H: 2 \times (3 + 2 * n)$ dimension

$$H = zeros((2, (3 + 2 * n))$$

$$H[0:2, 0:3] = H3$$

$$H[0:2, 3 + 2 * j : 3 + 2 * (j + 1)] = (-1) * H[0:2, 0:2]$$

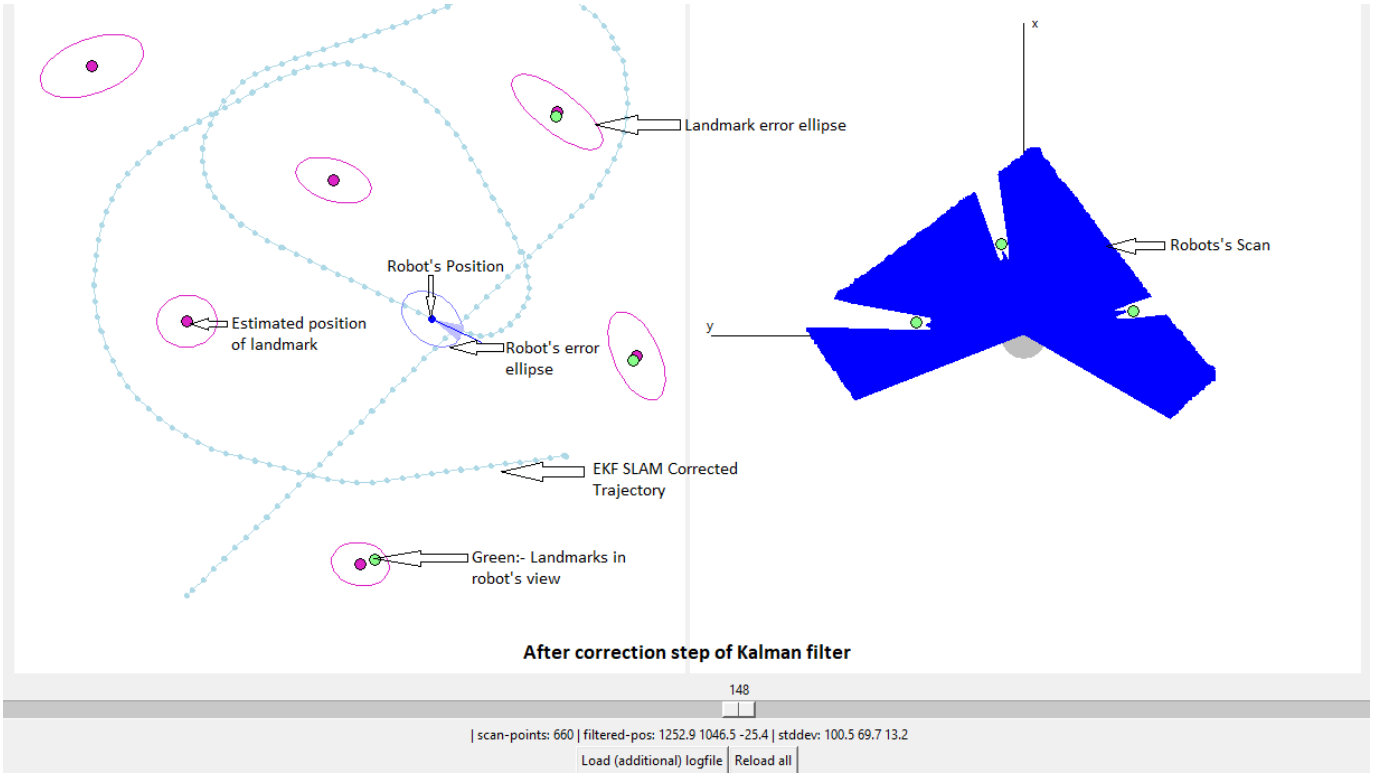
$$For\ calculation\ of\ Kalman\ Gain\ K_t:$$

$$K_t = \overline{\sum}_t H_t^T (H_t \overline{\sum}_t H_t^T + Q_t)^{-1}$$

And the calculation of corrected mean and covariance is as follows:

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

$$\sum_t = (I - K_t H_t) \overline{\sum}_t$$


Landmark error ellipse
Robot's Position
Estimated position
of landmark
Robot's error
ellipse
EKF SLAM Corrected
Trajectory
Green:- Landmarks in
robot's view
x
y
Robots's Scan
After correction step of Kalman filter
148
| scan-points: 660 | filtered-pos: 1252.9 1046.5 -25.4 | stddev: 100.5 69.7 13.2
Load (additional) logfile
Reload all

CHAPTER4: **Particle Filter**

4.1 Introduction

Brief Explanation:

- ➔ Non-parametric, recursive Bayes filter
- ➔ Posterior is represented by a set of weighted samples
- ➔ Not limited to Gaussians
- ➔ Proposal to draw new samples
- ➔ Wight to account for the differences between the proposal and the target
- ➔ Works well in low-dimensional spaces

4.2 Particle Filter Prediction

Particles: $x_t^{[i]}$ is a hypothetical state of the robot.

Set of random particles depicts the state of robot.

Consider a total of M particles in the space:

$$for\ i = 1\ to\ M$$

$$sample\ x^{[i]}\ \text{according to random.gaussian}(\mu, \sigma^2)$$

$$\text{return}\ \{x^{[1]}, x^{[2]} \dots, x^{[M]}\}$$

state of robot at time t is depicted by M particles:

$$x_t = \{x^{[1]}, x^{[2]} \dots, x^{[M]}\}$$

$$for\ m = 1\ to\ M$$

$$sample\ \overline{x}_t^{[m]} \sim P(x_t \mid x_{t-1}^{[m]}, u_t)$$

$$It\ is\ similar\ to\ \overline{x}_t^{[m]} = g\left(x_{t-1}^{[m]}, u_t\right)$$

$$where\ u_t = \begin{bmatrix} l_t \\ r_t \end{bmatrix}$$

$$Now\ sample\ l_t'\ according\ to\ random.gauss\left(l_t, \sigma_{l_t}^2\right)$$

$$And\ sample\ r_t'\ according\ to\ random.gauss\left(r_t, \sigma_{r_t}^2\right)$$

$$\sigma_l^2 = (\alpha_1 * l)^2 + (\alpha_2 * (l - r))^2$$

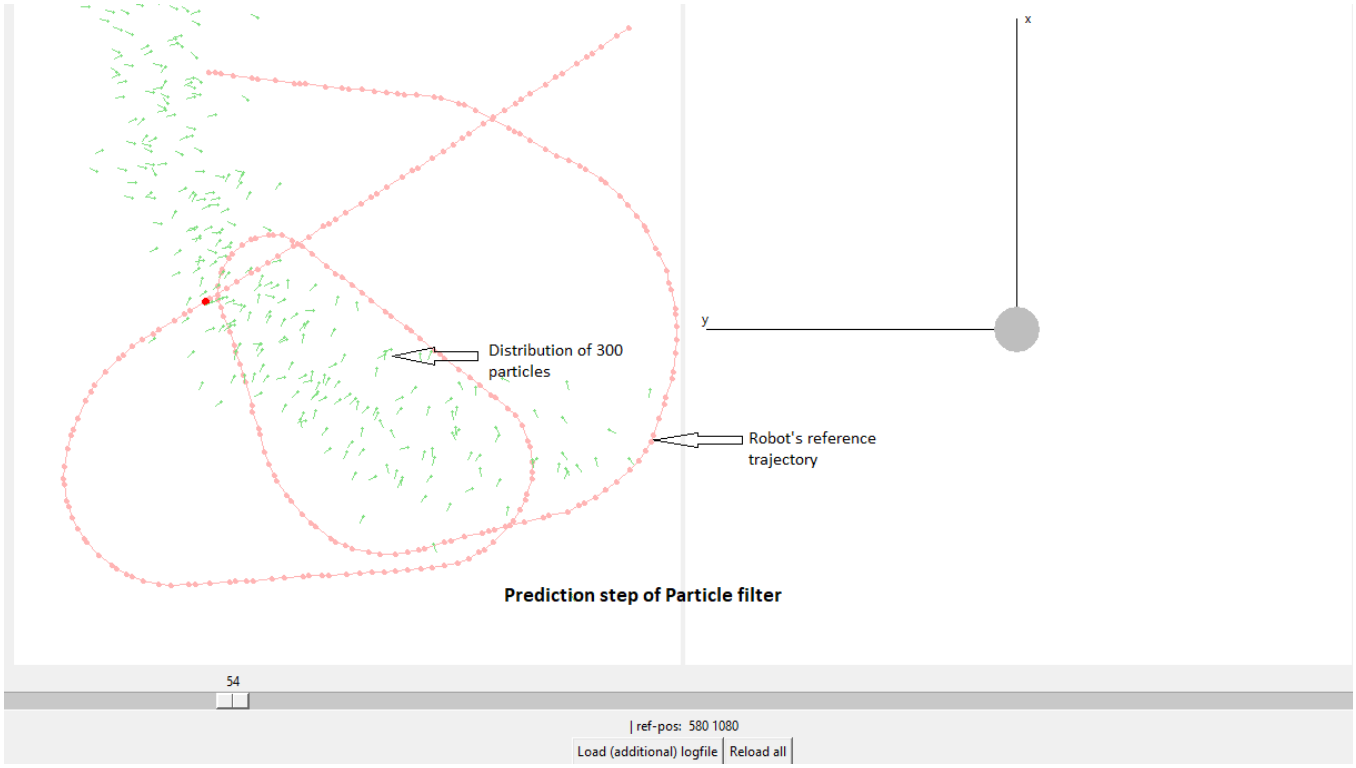
$$\sigma_r^2 = (\alpha_1 * r)^2 + \left(\alpha_2 * (l - r)\right)^2$$

$$\alpha_1 = control\ motion\ factor$$

$$\alpha_2 = control\ turning\ factor$$

$$\overline{x}_t^{[m]} = g(\overline{x}_{t-1}^{[m]}, \begin{bmatrix} l'_t \\ r'_t \end{bmatrix})$$

After prediction step of Particle filter, results is depicted in the below plot:



4.3 Particle Filter Correction

Now we will compute one weight for each particle. Consider a total of M particles.

$$for\ m = 1\ to\ M:$$

$$w_t^{[m]} = P(z_t \mid \overline{x}_t^{[m]})$$

4.3.1 Importance Sampling

- Draw *M* particles with replacement from the set $\{\overline{x}_t^{[i]}\}$ with probability proportional to to the weight $w_t^{[i]}$
- Before sampling, particles approximate $\overline{bel}(x_t)$

- After sampling, particles approximate $bel(x_t)$
- Particles with low weight will likely disappear

Importance sampling general technique:

$$w = \frac{target\ distribution}{proposal\ distribution} = \frac{bel(x_t)}{\overline{bel}(x_t)} = \frac{\alpha P(z|x)\overline{bel}(x_t)}{\overline{bel}(x_t)} = P(z|x)$$

We have laser scan measurement of detected landmark in terms of (range, bearing) denoted as (d, α). Also, we have predicted landmarks as assigned to the closest landmark in the arena (d', α').

The probability of measurement can be calculated as:

$$P(z|x) = P(d - d') * P(\alpha - \alpha')$$

Since there are 6 landmarks in the arena, so at a given pose,

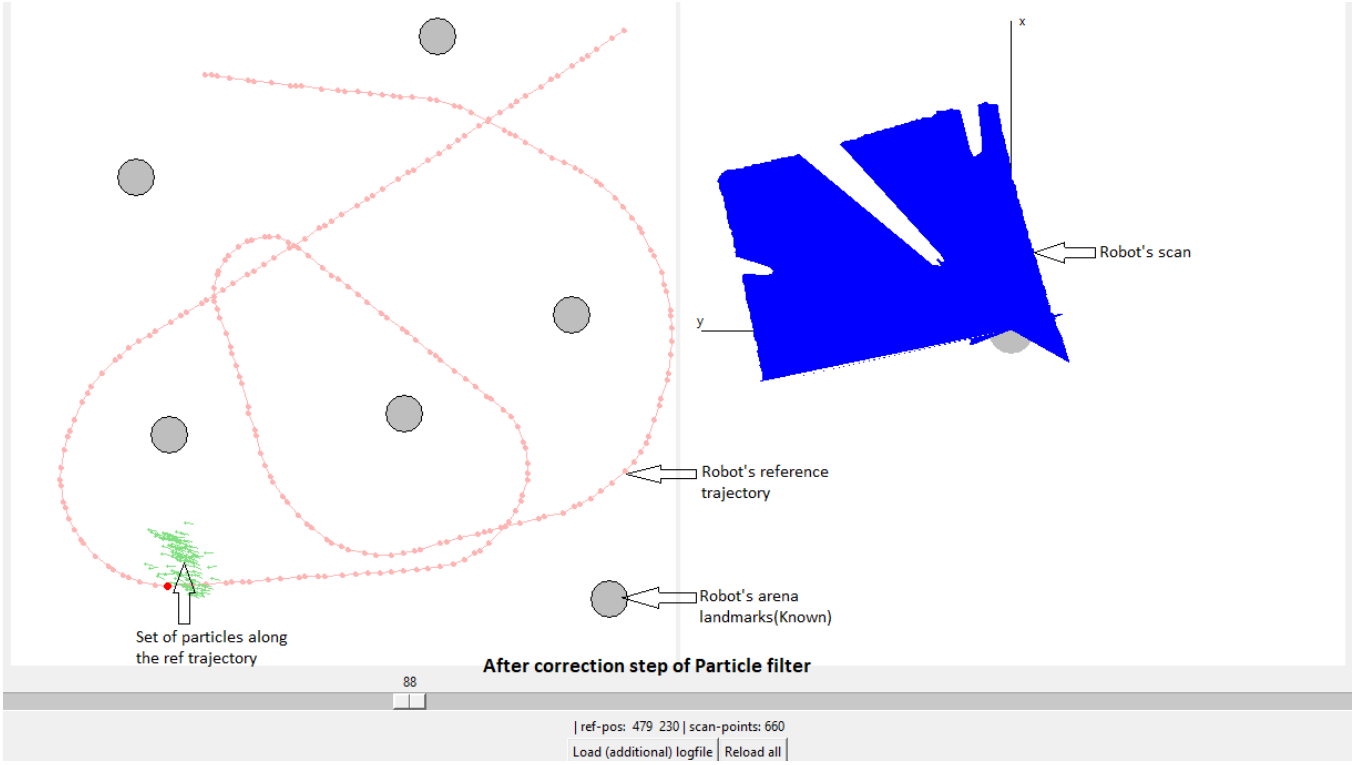
$$P(z_i|x)\ will\ be\ calculated\ for\ i\ varying\ from\ 1\ to\ 6$$

So, probability of all the measurement given the current pose will be:

$$P(z|x) = \prod_{i=1}^{6} P(z_i|x)$$

4.3.2 Resampling wheel

Particles with higher weights should be sampled more frequently (in proportion to their weight). It returns a list of particles which have been resampled, proportional to their weights.



4.4 Density Estimation

The Particle filter, prediction and correction step has been implemented. In addition to the filtered particles, now we will estimate the density. In which the mean position and heading is computed from all particles.

$$Estimated\ mean\ position: X_{mean} = \frac{1}{M}\sum_{i=1}^{M} x_i$$

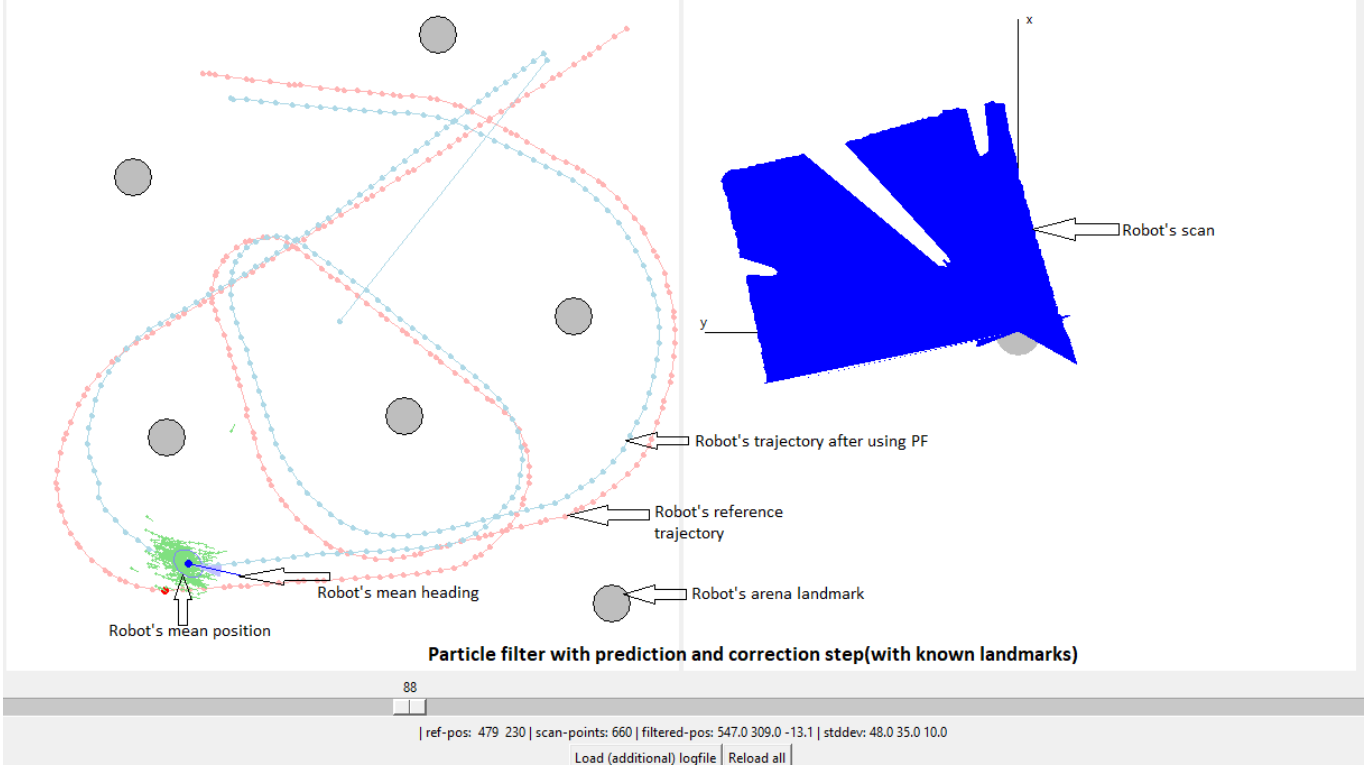
$$Regarding\ mean\ heading\ \theta, mean\ heading\ vector\ will\ be\ computed\ instead\ of\ mean\ angle:$$

$$V_x = \frac{1}{M}\sum_{i} cos\theta_i$$

$$V_y = \frac{1}{M}\sum_{i} sin\theta_i$$

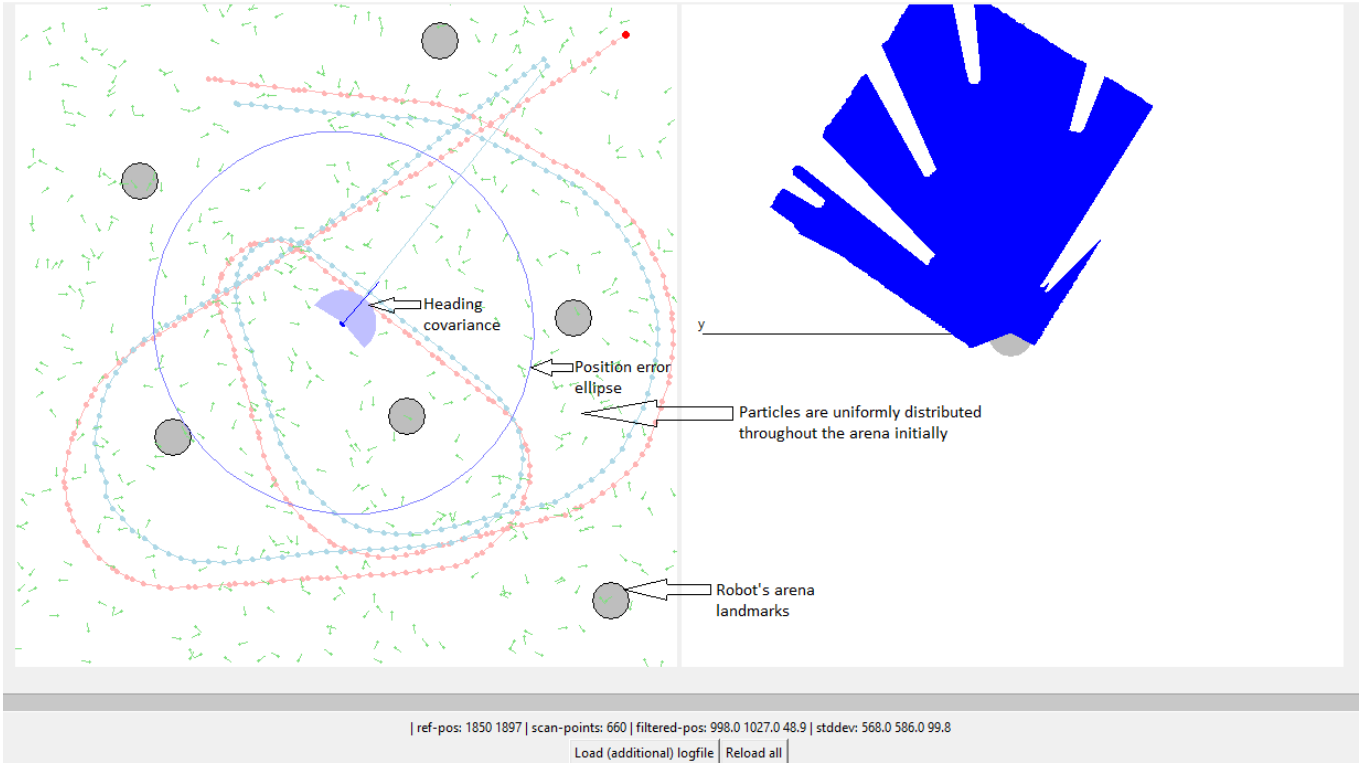
$$\theta_{mean} = atan2(V_y, V_x)$$

We will now output the state estimated from all particles. This will help us to know the mean trajectory of the robot after applying Particle filter.



4.5 Resampling Effects

- Resampling too often may lose diversity. Resampling too seldom waste particles in low probability regions.
- Solution of above problem is: No resampling if robot is static and integrate multiple measurements into weights. Use variance of the weights to determine if resampling is necessary.
- Without any knowledge of initial state, Particle filter manages to get very close to the true state very quickly.
- So, the particles are initialized uniformly distributed in the arena, and a larger number of particles are used to avoid the particle deprivation problem.


Heading
covariance
Position error
ellipse
Particles are uniformly distributed
throughout the arena initially
Robot's arena
landmarks
y
| ref-pos: 1850 1897 | scan-points: 660 | filtered-pos: 998.0 1027.0 48.9 | stddev: 568.0 586.0 99.8
Load (additional) logfile
Reload all

CHAPTER5: **Fast SLAM**

5.1 Particle Representation

A set of weighted samples

$$X = \{< x^{[i]}, w^{[i]} >\} for\ i = 1, \dots, M$$

Each sample is the hypothesis about the state.

For feature-based SLAM:

$$x = (x_{1:t}, l_{1,x}, l_{1,y}, \dots, l_{M,x}, l_{M,y})^T$$

Pose landmarks

Particle filters are effective in low dimensional spaces as the likely volumes of the state space need to be covered with samples.

If we use the particle set only to model the robot's path, each sample is a path hypothesis. For each sample, we can then compute an individual map of landmarks.

5.2 Factorization of Posterior

Factorization to exploit dependencies between variables:

$$P(a, b) = P(b \mid a) * P(a)$$

Factorization of SLAM posterior:

$$P(x_{0:t}, m \mid z_{1:t}, u_{1:t}) = P(x_{0:t} \mid z_{1:t}, u_{1:t}) * \prod_{i=1}^{N} P(m_i \mid x_{0:t}, z_{1:t})$$

$$\prod_{i=1}^{N} P(m_i \mid x_{0:t}, z_{1:t}) \rightarrow product\ (over\ all\ landmarks) of\ probability for\ the\ landmark\ location\ given\ the\ entire\ path\ and\ all\ measurements$$

- Represented using independent EKF (one per landmark)

This is the product of probability of each individual map feature given the path and measurements.

$$P(m|x_{0:t}, z_{1:t}) = \prod_{i=1}^{N} P(m_i \mid x_{0:t}, z_{1:t})$$

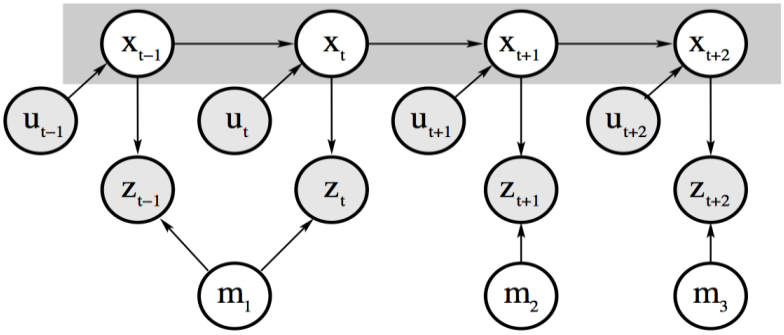
$$P(m|x_{0:t}, z_{1:t}) \rightarrow Probability\ for\ the\ map\ given\ the\ path\ and\ measurements$$

$$P(x_{0:t}, m \mid z_{1:t}, u_{1:t}) \rightarrow Represented\ using\ particles$$

We represent the distribution of the robot's path using particle filter and for each particle we represent the remaining part of posterior by a set of gaussian distribution (one distribution for each landmark).

Conditional Independence:

If the path $x_{0:t}$ is known then, locations of the landmarks are conditionally independent



- There are M particles and N landmarks:
- Each landmark is represented by a 2×2 EKF
- Each particle therefore has to maintain N individuals EKFs.

Particle 1:	$x_{0:t}^{[1]}$	$\mu_1^{[1]}, \sum_1^{[1]}$	$\mu_2^{[1]}, \sum_2^{[1]}$	……..	$\mu_N^{[1]}, \sum_N^{[1]}$
Particle 2:	$x_{0:t}^{[2]}$	$\mu_1^{[2]}, \sum_1^{[2]}$	$\mu_2^{[2]}, \sum_2^{[2]}$		$\mu_N^{[2]}, \sum_N^{[2]}$
		. ….	….		…
Particle M	$x_{0:t}^{[M]}$	$\mu_1^{[M]}, \sum_1^{[M]}$	$\mu_2^{[M]}, \sum_2^{[M]}$		$\mu_N^{[M]}, \sum_N^{[M]}$

5.3 Data Association

$$P(x_{0:t}, m \mid z_{1:t}, u_{1:t}, C_{1:t}) = P(x_{0:t} \mid z_{1:t}, u_{1:t}, C_{1:t}) * \prod_{i=1}^{N} P(m_i \mid x_{0:t}, z_{1:t}, C_{1:t})$$

$$C_{1:t} \rightarrow correspondences$$

- Each particle has to maintain individual data association.
- Fast SLAM maintains posterior over multiple data associations.

Fast SLAM solves Full SLAM problem:

Particle 1:	$x_{0:t}^{[1]}$	$\mu_1^{[1]}, \sum_1^{[1]}$	$\mu_2^{[1]}, \sum_2^{[1]}$	……..	$\mu_N^{[1]}, \sum_N^{[1]}$

Here

$$x_{0:t}^{[1]} \rightarrow depicts\ the\ full\ path$$

Also, Fast SLAM solves the online SLAM problem

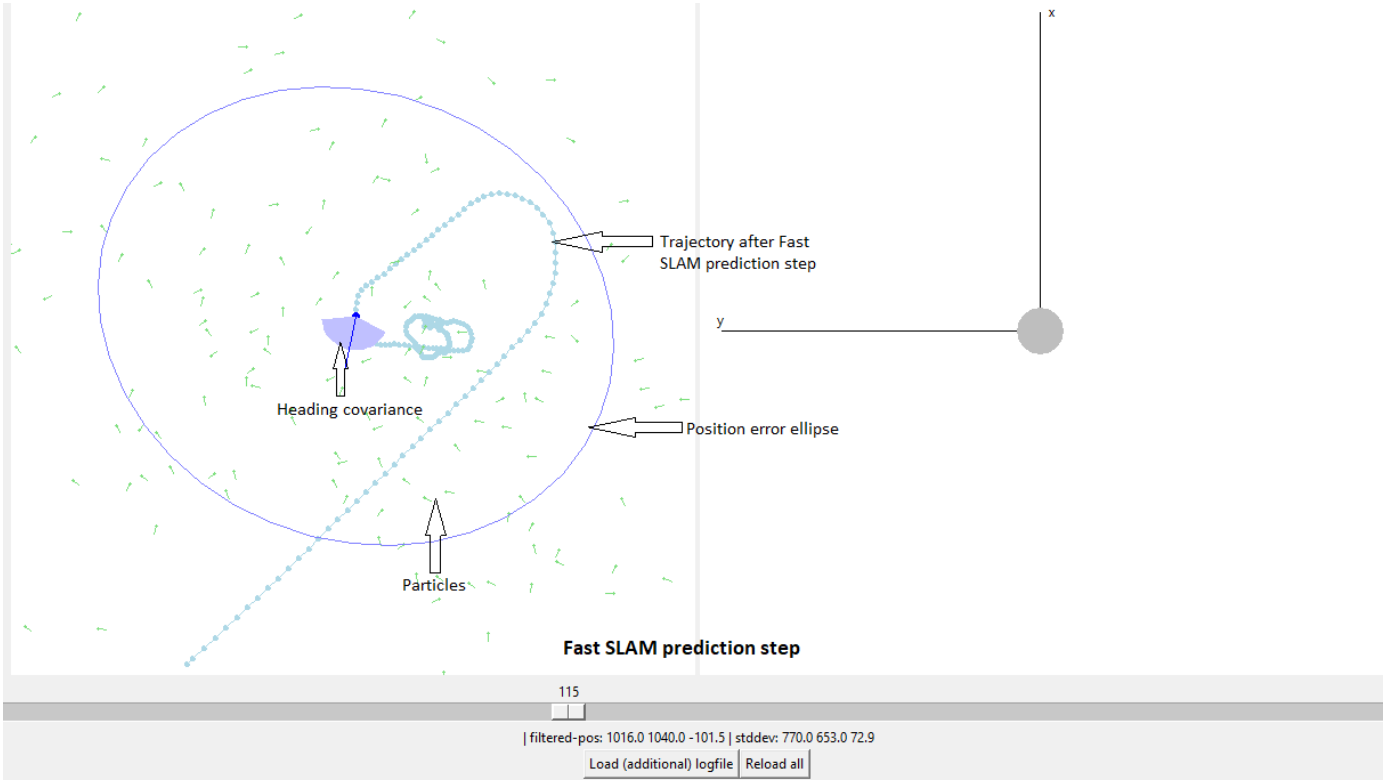
Particle 1:	$x_{0:t}^{[1]}$	$\mu_1^{[1]}, \sum_1^{[1]}$	$\mu_2^{[1]}, \sum_2^{[1]}$	……..	$\mu_N^{[1]}, \sum_N^{[1]}$

Here

$$x_{0:t}^{[1]} \rightarrow depicts\ only\ the\ last\ pose$$

5.4 Prediction step of Fast SLAM

For each particle, it predicts its new position as already done earlier.



5.5 Correction step of Fast SLAM

Update and compute weights (Cylinders) -> This function computes all the weights and it also updates all the particles.

Update particle (measurement)

Given a measurement, computes the likelihood that it belongs to any of the landmarks in the particle. If there are none, or if all likelihoods are below the minimum correspondence likelihood threshold, add a landmark to the particle. Otherwise, update the (existing) landmark with the largest likelihood.

Measurement contains one range and bearing angle corresponding to the measurement of single cylinder

Compute likelihood of the correspondence for any existing landmark

A. Compute likelihoods:

Expected measurement for landmark: It returns the expected distance and bearing measurement for a given landmark number and the pose of this particle.

Measurement function: Takes a $(x, y, theta)$ state and a (x, y) landmark, and returns the corresponding (range, bearing).

$$Expected\ measurement:$$

$$z_{expected} = h\big((x, y, \theta), m_k\big)\ where\ m_k \rightarrow k_{th}\ landmark's\ coordinate$$

Now computes Jacobian H of measurement function at the particle's position and the landmark given by landmark number.

$$H = \frac{\partial h}{\partial landmark}\ \ given\ (x, y\ , \theta, m_k)$$

Now computes the measurement covariance matrix

Covariance of the landmark measurement:

$$Q_l = H \sum_k H^T + Q_t$$

$$where\ Q_t \rightarrow variance\ due\ to\ actual\ measurement$$

$$Q_t = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\alpha^2 \end{bmatrix}$$

$$\sigma_r^2 \rightarrow variance\ in\ range\ measurment$$

$$\sigma_\alpha^2 \rightarrow variance\ bearing\ angle\ measurement$$

$$\sum_k \rightarrow variance\ in\ k_{th}\ landmark\ position(x, y)$$

$$H \sum_k H^T \rightarrow measurement\ variance\ due\ to\ landmark\ variance$$

For a given measurement and landmark number, we will calculate the likelihood that the measurement corresponds to the landmark.

$$\Delta Z = Z_{measured} - Z_{expected}$$

Likelihood is obtained from the probability density function of the Gaussian distribution.

$$l = \frac{1}{2\pi\sqrt{\det(Q_l)}} * \ e^{-\frac{1}{2}\Delta Z^T {Q_l}^{-1} \Delta Z}$$

For a given measurement, we will returns a list of all correspondence likelihoods (from index 0 to number of landmark -1).

B. Initialize new landmark

Until now our measurement see the true cylinder and it determines by computing the likelihood that the new landmark should be incorporated into its list of current landmark or not.
Using the known position, we have computed the position of the landmark and also computed appropriate covariance.

$$Z = h((x, y, \theta), m)$$

Now we want to get our landmark position from current pose and our measurement.

$$m = h^{-1}\big((x, y, \theta), Z\big) \; where \; Z \; is \; known \; range \; and \; bearing \; measurement$$

Scanner to world () function will compute the world coordinate (m) of our new landmark from the x, y coordinates in the scanner coordinate system.

Since, Jacobian H of measurement function is given by:

$$H = \frac{\partial h}{\partial landmark} \; given \; (x, y, \theta, m_k)$$

H -matrix gives us the information about how our landmark noise translates to a measurement noise.

Now our robot measures the range and bearing measurement, so the covariance should translate it to the noise of the landmark position.
So, there is a need to take inverse of $H \; matrix$. New covariance will be calculated as:

$$\sum = H^{-1} Q_t (H^{-1})^T$$

$$where \; Q_t \rightarrow measurement \; noise \; due \; to \; measurement \; device$$

C. Landmark update

Update a landmark's estimated position and covariance.

Due to our measurement process, position of the landmark will move close to the measurement.

So, error ellipse of the landmark will get smaller.

To update the position of the landmark, we need to use standard Kalman filter equation.

- To update the state:

$$Kalman\ gain, K = \sum_{old} H^T \left(H \sum_{old} H^T + Q_t\right)^{-1}$$

$$since, \qquad \left(H \sum_{old} H^T + Q_t\right) = Q_l$$

$$K = \sum_{old} H^T {Q_l}^{-1}$$

- New landmark positions and new covariance will be now computed as:

$$\mu_{new} = \mu_{old} + K\left(Z - h\left((x, y, \theta), \mu_{old}\right)\right)$$

$$Z \rightarrow Actual\ measurement$$

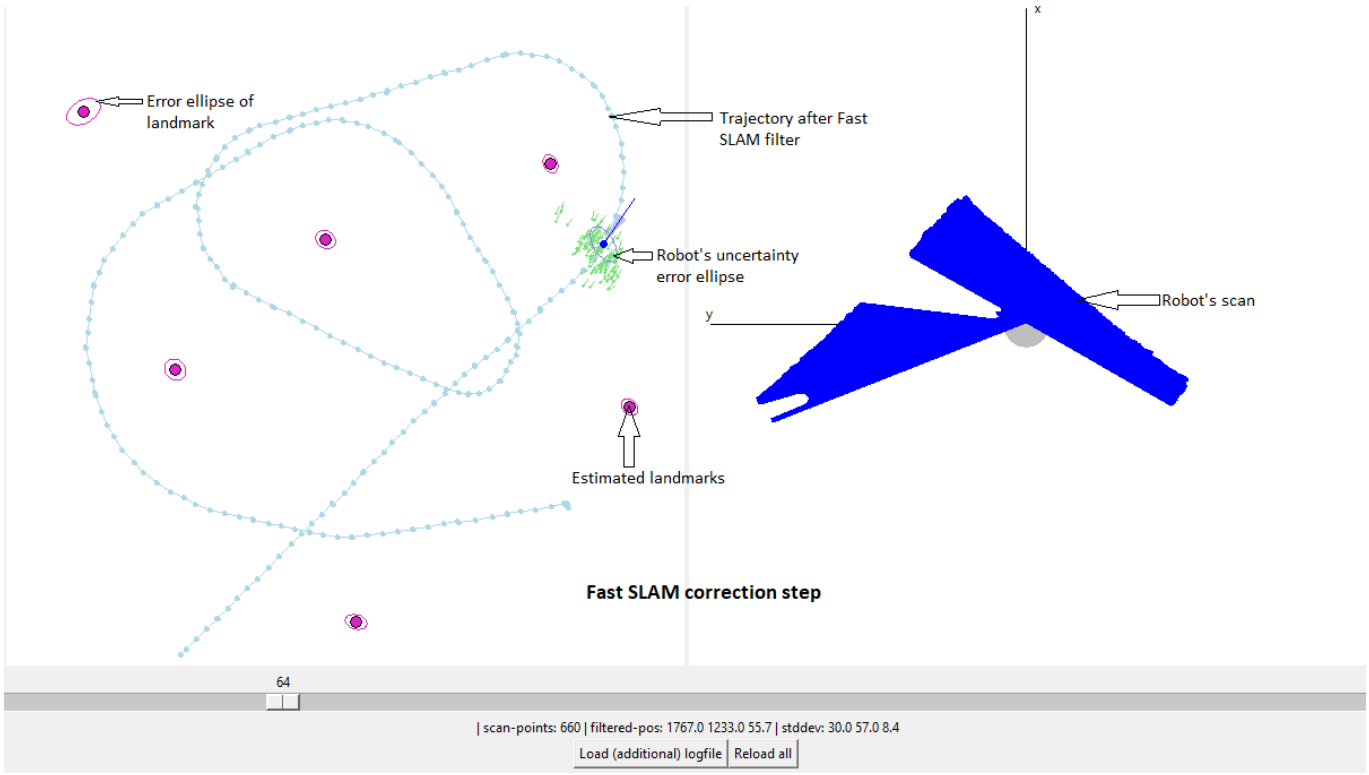
$$h\left((x, y, \theta), \mu_{old}\right) \rightarrow Expected\ measurement$$

$$\mu_{old} \rightarrow old\ position\ of\ the\ landmark$$

$$\sum_{new} = (I - KH)\sum_{old}$$

$$I \rightarrow identity\ matrix$$

$$\sum_{old} \rightarrow old\ covarianceof\ landmark$$


Error ellipse of landmark
Trajectory after Fast SLAM filter
Robot's uncertainty error ellipse
Robot's scan
x
y
Estimated landmarks
Fast SLAM correction step
64
| scan-points: 660 | filtered-pos: 1767.0 1233.0 55.7 | stddev: 30.0 57.0 8.4
Load (additional) logfile
Reload all

CHAPTER6: **CONCLUSION & FURTHER WORK**

So far you could have seen the report on the implementation of Kalman filter and Particle filter for solving the problem of SLAM. Kalman filter prediction and correction steps are the magic behind all the correcting the robot's trajectory as close to the reference trajectory when the positions of landmarks are known in the world coordinate system. Similarly, Particle filter is also able to correct the trajectory very efficiently. So, we proceed to solve the problem of SLAM as a egg-chicken problem, where we have to simultaneously generate the map of the environment and localize the robot also. Just we have the control data for definite interval of time. Here we have attempted to solve the SLAM problem through Extended Kalman filter and Particle filter as well. Particularly Particle filter SLAM is known as Fast SLAM and is best than EKF SLAM.

In near future, we will likely proceed to implement these algorithms for a autonomous robots traversing in the unknown environments and will continue to work on path planning and motion planning problem.

REFERENCE LIST

[1] SLAM Lectures by Claus Brenner
https://www.youtube.com/playlist?list=PLpUPoM7Rgzi_7YWn14Va2FODh7LzADBSm

[2] Artificial Intelligence for Robotics course by Sebastian Thrun
https://classroom.udacity.com/courses/cs373

[3] “Probabilistics Robotics” a book by Sebastian Thrun
https://docs.ufpr.br/~danielsantos/ProbabilisticRobotics.pdf

[4] SLAM Courses by Cyrill Stachniss
https://www.youtube.com/watch?v=U6vr3iNrwRA&list=PLgnQpQtFTOGQrZ4O5QzbIHgl3b1JHimN_