Preliminary progress on application of SLAM algorithm

Offline application on Ground Vehicle

Agraj Jain

Control/Robotics Research Laboratory

October 20, 2014

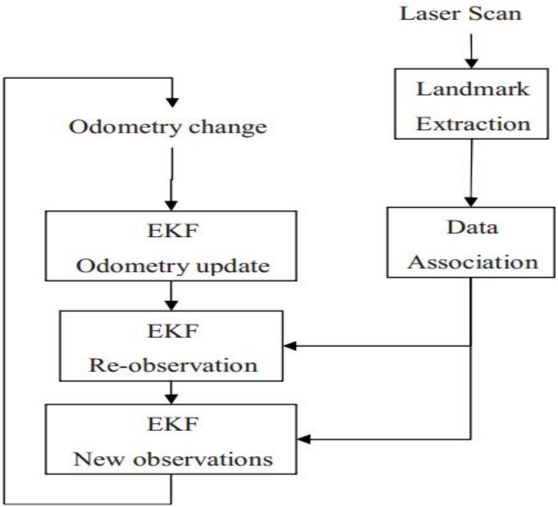
Introduction

- Can be implemented in many ways
- More of a concept than a simple algorithm
- Applicable to both 2D and 3D
- Consists of multiple parts:
    1. Landmark Extraction
    2. Data Association
    3. State Estimation
    4. State update
    5. Landmark Update

- Odometry:
  - 2 Encoders sampled at 1000 Hz
  - Velocity logged at a rate of 200 Hz
- Piccolo Laser Distance Sensor:
  - 6 m range
  - 5 Hz Rotation Speed
  - 1deg Angular Resolution

SLAM Process

The EKF SLAM



Moving from Deterministic to Probablistic models:

Bayes Filter:

Prediction:

$$p(\bar{x}_t) = \sum_{x_{t-1}} P(x_t \mid x_{t-1}, u_t).bel(x_{t-1}) = convolve(p(u), p(x_{t-1}))$$

Correction:

$$p(x_t) = \alpha.P(z_t \mid x_t).bel(\bar{x}_t) = mult(p(z), p(\bar{x}_t))$$

Gaussian Distribution:

$$N\left(x; \mu, \sigma^2\right) = \frac{1}{\sqrt{2\pi}.\sigma}.e^{-\frac{1}{2}.\left(\frac{x-\mu}{\sigma}\right)^2}$$

Extended Kalman Filter

Prediction

Motion model

If r $\neq$ l:

$$\alpha = \frac{r - l}{w} \qquad R = \frac{l}{\alpha}$$

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

If r $=$ l:

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

In general: $x' = g(x, u)$ Where:

$$x = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \qquad u = \begin{bmatrix} l \\ r \end{bmatrix}$$

The Prediction Step:

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

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

Where,

$$G_t = \frac{\partial g}{\partial state} \qquad V_t = \frac{\partial g}{\partial control}$$

$$\Sigma_{Control} = \begin{bmatrix} \sigma_l^2 & 0 \\ 0 & \sigma_r^2 \end{bmatrix}$$

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

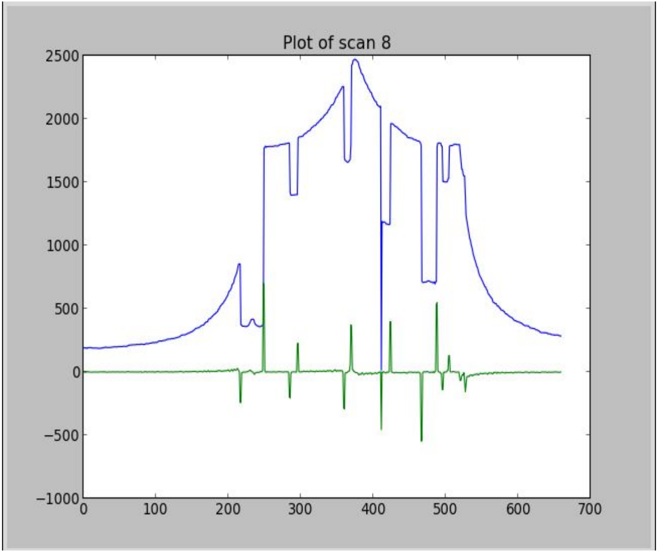
$$\sigma_r^2 = (\alpha_1.r)^2 + (\alpha_2.(l-r))^2$$

Landmark Detection:

- Use LIDAR data to identify landmarks.
- Unique to each arena and robot.
- Simplistic example is thresholding of the first derivative of the scan

Each time step we get a set of landmarks each of it identified as z_t

$$z_t = \begin{bmatrix} r \\ \alpha \end{bmatrix}$$


Plot of scan 8
2500
2000
1500
1000
500
0
−500
−1000
0
100
200
300
400
500
600
700

Predicting the measurement:

$$\bar{z}_t = h(x_m, y_m, \theta)$$

$$\bar{z}_t = \begin{bmatrix} r \\ \alpha \end{bmatrix} = \begin{bmatrix} \sqrt{(x_l - x_m)^2 + (y_l - y_m)^2} \\ \text{arctan}\left(\frac{y_l - y_m}{x_l - x_m}\right) - \theta \end{bmatrix}$$

Where, x_l, y_l are the co-ordinates of each landmark already on the map.

Extended Kalman Filter

Correction

Predicted measurement:

$$\bar{z}_t = h(\bar{\mu}_t)$$

Correction:

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

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

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

Where, K is the Kalman gain, and

$$H = \frac{\partial h}{\partial \mu} \qquad Q = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\alpha^2 \end{bmatrix}$$

Extended Kalman Filter

SLAM

When landmarks are not fixed constants, We modify the state vector from having only x,y,andθ to :

$$\mu = \begin{bmatrix} x \\ y \\ \theta \\ x_1 \\ y_1 \\ x_2 \\ y_2 \\ . \\ . \\ . \end{bmatrix} \quad \text{and} \quad \Sigma = \begin{bmatrix} \Sigma_{Old} & \Theta \\ \Theta & \begin{bmatrix} \sigma_{x1}^2 & 0 \\ 0 & \sigma_{y1}^2 \end{bmatrix} \end{bmatrix}$$

We pass only the first three elements of the state to the prediction step as the rest are independent of it.

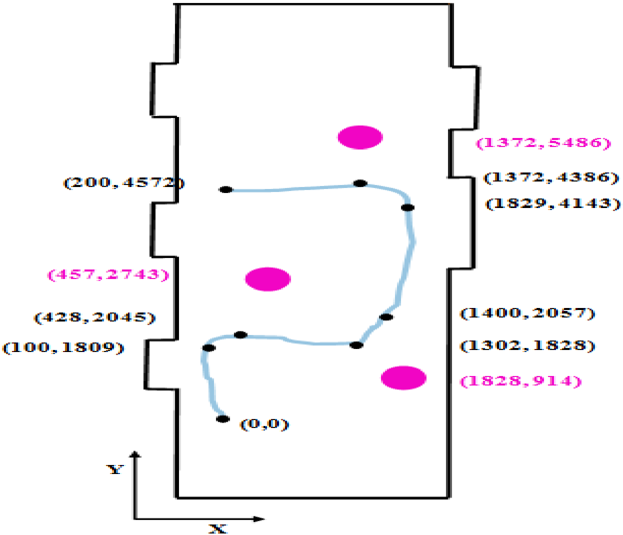
For the correction step we pass the whole state as now h is not only function of x,y,θ it is also dependent on x_1, y_1 etc.

Extended kalman Filter

Filter Constants:

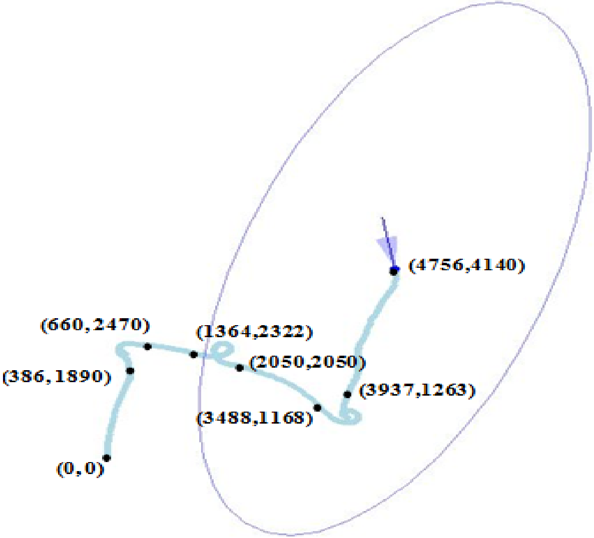
- α_1: Robot movement error factor
- α_2: Robot turning error factor
- σ_r: Sensor Distance Error
- σ_α: Sensor Angle Error

Actual data run for Reference



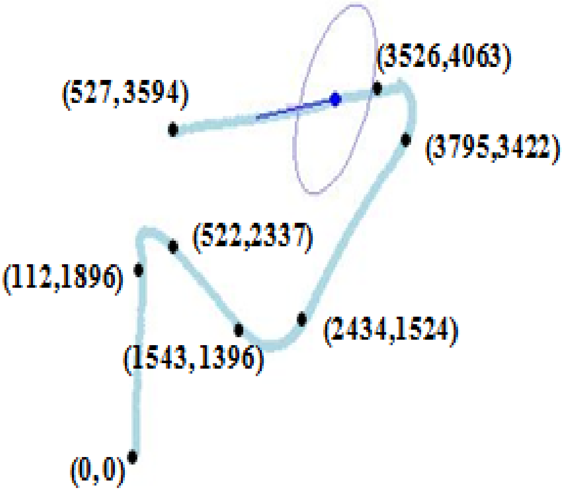
Results

Prediction using logged Velocity



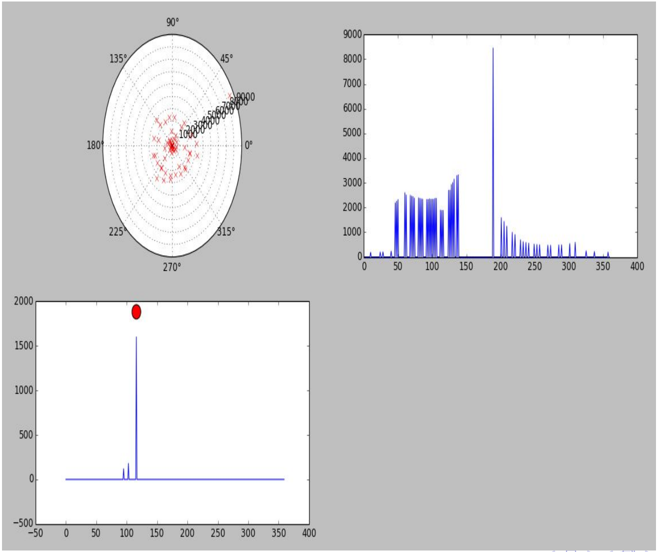
Results

Prediction using logged Velocity (After Temporary fix)



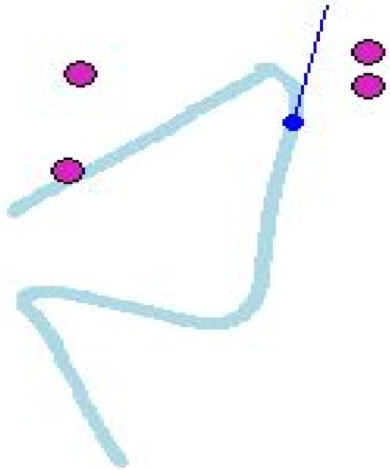
Results

Typical Scan



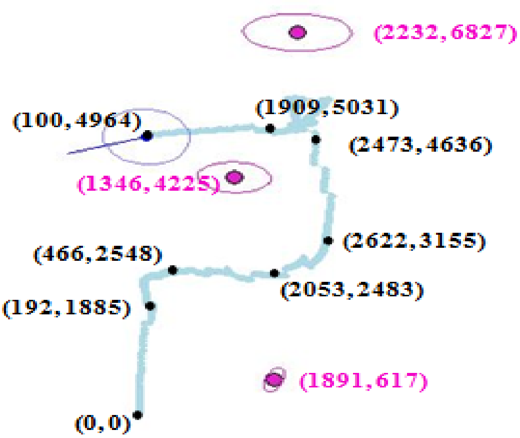
Results

Arena Design landmarks



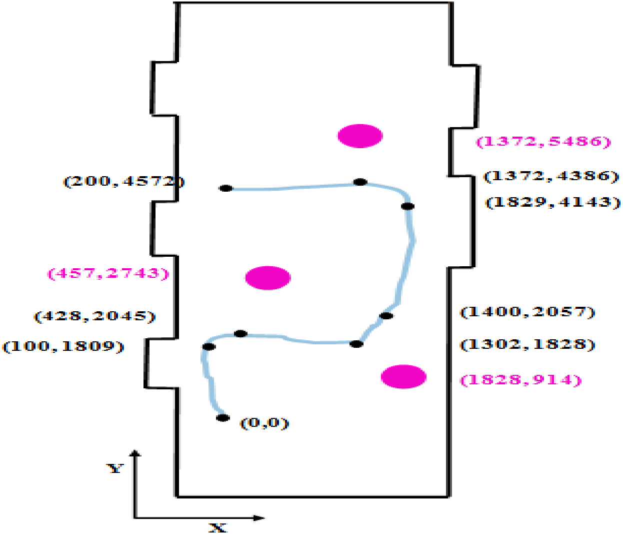
Results

Correction using logged Velocity (After Temporary fix)



Results

Actual data run for Reference



Further steps

- Implement Particle Filter and Fast SLAM
- Fix Encoder data processing to get a reasonable prediction
- Implement on-line version of the algorithm on the robot.
- Explore other landmark extraction algorithms
  - Recognizing walls, and using it as more landmarks
  - Mahalanobis Distance vs. Euclidean Distance
  - RANSAC algorithm for line extraction

Thank you