**CSE-483: Mobile Robotics** **Monsoon 2019**

# Assignment-4 Report

*Prepared by: Anirudha Ramesh, Aryan Sakaria*

Ego-localization in world of robot is a fundamental task in mobile robotics. Traditionally with no visual input or any other alternate input/observation other than actuator input, we can get position of robot in it's world very accurately given accurate motion sensor output from actuators. However, in the real world there is a significant uncertainty owing to noise in collection of motion data from actuators alone, due to which the robot can no longer very accurately pinpoint its position. To combat and overcome this uncertainty that exists with our independent 'motion model', we introduce a new 'sensor' model which is based on visual/some other kind of camera input, combining which we get more accurate results.

## 1 Method

**Nature of uncertainty**
Errors arise due to measurement, and less so, computation usually (owing to precision and sorts). Errors arising due to measurement can be broadly assumed to be random following a normal distribution. Thus, the noise in our system can be modelled as a Gaussian

**Setting / Environment**
To combine our motion and sensor model, we introduce 'landmarks', which are points on the map we already know the exact position of. The sensor model looks at the landmark and the current predicted position and tries to calculate the error between where our robot thinks it is and where it actually is based on relative position of landmark when we are in predicted position and relative position of landmark in our actual position.

**Kalman Filter**
Kalman filter is built on the assumption that all transformations are linear, and a linear transformation of a gaussian remains a gaussian. Therefore Kalman filters as they are cannot be directly applied to Non-Linear Dynamic Systems. Owing to this the Extended Kalman filter is introduced.

**EKF(filter design)**
To resolve problem raised above, we use the concept of local linearization which we get via the first order taylor expansion of the Non-Linear Dynamic Systems.

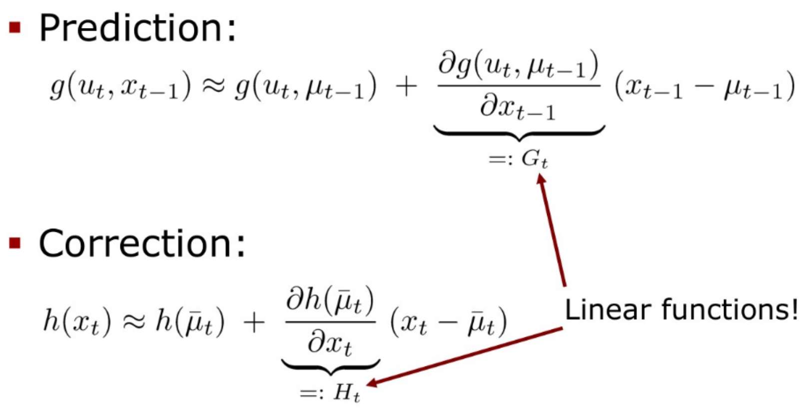


Figure 1: Courtesy: Cyrill Stachniss slides

The algorithm we implemented is as follows:

1: **Algorithm Extended_Kalman_filter**($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):
2: $\bar{\mu}_t = g(u_t, \mu_{t-1})$
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: $\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$
6: $\Sigma_t = (I - K_t\, H_t)\, \bar{\Sigma}_t$
7: **return** $\mu_t, \Sigma_t$

Figure 2: Courtesy: Probabilistic Robots

**Jacobians and other matrices**
$R_t$ is a 3*3 diagonal matrix that represents odometry noise. The diagonal elements are [0.00442026, 0.00442026, 0.00818609] where the first two elements are variance in translational speed readings, and the third element is variance in rotational speed readings. $Q_t$ is a 34*34 diagonal matrix that represents observation noise. $Q_{ii}$ is $r_{var}$ if i is even, otherwise it is $b_{var}$
The matrices $G_t$ and $H_t$ are jacobian matrices of the motion and sensor model respectively. Below are the equations for the same.

Motion Model

$\vec{x}_k = g(\vec{x}_{k-1}, \vec{u}_k)$ where $\vec{x}_i = \begin{bmatrix} x_i \\ y_i \\ \theta_i \end{bmatrix}$ and $\vec{u}_i = \begin{bmatrix} v_i \\ \omega_i \end{bmatrix}$

$$G_t = \frac{\partial g(\vec{x}_{k-1}, \vec{u}_k)}{\partial \vec{x}_{k-1}}$$

$$= \frac{\partial}{\partial \vec{x}_{k-1}} \left\{ \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{bmatrix} + dt \begin{bmatrix} \cos\theta_{k-1} & 0 \\ \sin\theta_{k-1} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_k \\ \omega_k \end{bmatrix} \right\}$$

$$= I + dt * \frac{\partial}{\partial \vec{x}_{k-1}} \left\{ \begin{bmatrix} v_k \cos\theta_{k-1} \\ v_k \sin\theta_{k-1} \\ \omega_k \end{bmatrix} \right\}$$

$$= I + dt * \begin{bmatrix} 0 & 0 & -a \\ 0 & 0 & b \\ 0 & 0 & 0 \end{bmatrix}$$ where $a = v_k \sin\theta_{k-1}$, $b = v_k \cos\theta_{k-1}$

Final Motion-model Jacobian:

$$\begin{bmatrix} 1 & 0 & -dt * v_k * \sin\theta_{k-1} \\ 0 & 1 & dt * v_k * \cos\theta_{k-1} \\ 0 & 0 & 1 \end{bmatrix}$$

Figure 3: Motion Model Jacobian

## Sensor Model

The sensor model w.r.t. $l^{th}$ landmark at timestep $k$ is as follows:

$$\begin{bmatrix} r_k^l \\ \phi_k^l \end{bmatrix} = \begin{bmatrix} \sqrt{\{(x_l - x_k - d\cos\theta_k)^2 + (y_l - y_k - d\sin\theta_k)^2\}} \\ \text{atan2}(y_l - y_k - d\sin\theta_k,\ x_l - x_k - d\cos\theta_k) - \theta_k \end{bmatrix} = h(\vec{x}_k)$$

The sensor jacobian is a $2 \times 3$ matrix corresponding to 1 landmark.

$$H = \frac{\partial h(\vec{x}_k)}{\partial \vec{x}_k} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \end{bmatrix}$$

$$\boxed{\begin{array}{l} x\text{-diff} = (x_l - x_k - d\cos\theta_k) \\ y\text{-diff} = (y_l - y_k - d\cos\theta_k) \end{array}}$$

$$h_{10} = \frac{y\text{-diff}}{(x\text{-diff})^2 + (y\text{-diff})^2} \ ; \quad h_{11} = \frac{-x\text{-diff}}{((x\text{-diff})^2 + (y\text{-diff})^2)}$$

$$h_{12} = -d \times \frac{\{x\text{-diff} \times \cos(\theta_k) + y\text{-diff} \times \sin(\theta_k)\}}{(x\text{-diff})^2 + (y\text{-diff})^2} - 1$$

$$h_{00} = \frac{-x\text{-diff}}{\sqrt{x\text{-diff}^2 + y\text{-diff}^2}} \ ; \quad h_{01} = \frac{-y\text{-diff}}{\sqrt{(x\text{-diff})^2 + (y\text{-diff})^2}}$$

$$h_{02} = \frac{(x\text{-diff} \times d \times \sin(\theta_k) - y\text{-diff} \times d \times \cos(\theta_k))}{\sqrt{(x\text{-diff})^2 + (y\text{-diff})^2}}$$

These $2 \times 3$ matrices are stacked together to form a $34 \times 3$ matrice (for all 17 landmarks)
In case an observation for range is 0, a zeros matrix is appended instead

Figure 4: Jacobian for the sensor model


Figure 5: plot for ground truth vs odometry

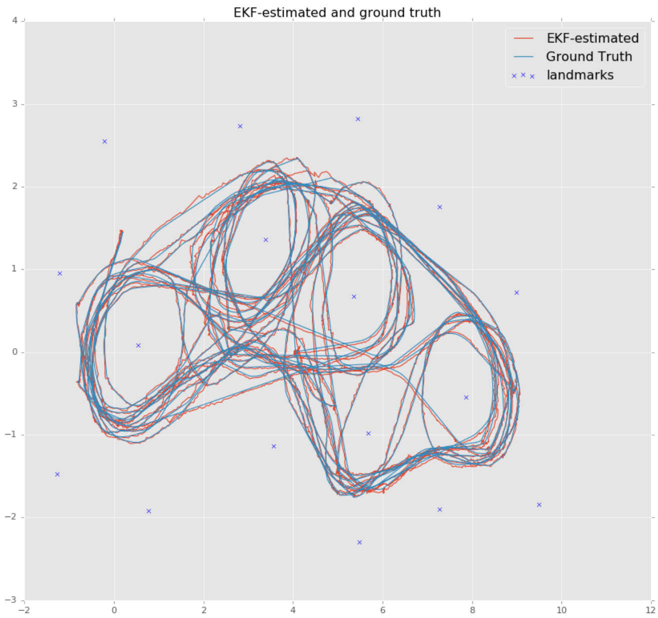


Figure 6: plot for EKF-estimated trajectory vs ground truth