ST2ESD- Estimation et Décision

Hussein Hijazi
hussein.hijazi@intervenants.efrei.net

Travaux Pratiques

efrei PARIS
Ecole d'ingénieurs du numérique

- **Objectif:**

Dans ce TP, vous estimerez de manière récursive la position d'un véhicule le long d'une trajectoire à l'aide des mesures disponibles et d'un modèle d'état de mouvement. Le véhicule est équipé d'un type très simple de capteur LIDAR, qui renvoie des mesures de portée et de l'angle correspondant à des points de repère individuels dans l'environnement. Les positions globales des points de repère sont supposées connues à l'avance. Nous supposerons également la connaissance de l'association des données, c'est-à-dire quelle mesure appartient à quel point de repère.

- **Modèle d'état de mouvement:**

Le modèle d'état de mouvement du véhicule reçoit des lectures d'odométrie de vitesse linéaire et angulaire en tant qu'entrées et produit l'état (c'est-à-dire la pose 2D) du véhicule:

$$\mathbf{x}_k = \mathbf{x}_{k-1} + T \begin{bmatrix} \cos\theta_{k-1} & 0 \\ \sin\theta_{k-1} & 0 \\ 0 & 1 \end{bmatrix} \left(\begin{bmatrix} v_k \\ \omega_k \end{bmatrix} + \mathbf{w}_k \right), \quad \mathbf{w}_k = \mathcal{N}(\mathbf{0}, \mathbf{Q})$$

- $\mathbf{x}_k = [x_k \;\; y_k \;\; \theta_k]^T$ est la pose 2D actuelle du véhicule
- v_k et ω_k sont les lectures d'odométrie de vitesse linéaire et angulaire, que nous utilisons comme entrées du modèle
- Le bruit de l'état $\mathbf{w}_k$ a une distribution normale (moyenne nulle) avec une matrice de covariance constante $\mathbf{Q}$.

- **Modèle d'observation ou de mesure:**

Le modèle de mesure relie la position actuelle du véhicule aux mesures de portée et de l'angle du LIDAR $\mathbf{y}^l{}_k=[r\ \phi]^T$.

$$\mathbf{y}_k^l = \begin{bmatrix} \sqrt{(x_l - x_k - d\cos\theta_k)^2 + (y_l - y_k - d\sin\theta_k)^2} \\ atan2\,(y_l - y_k - d\sin\theta_k, x_l - x_k - d\cos\theta_k) - \theta_k \end{bmatrix} + \mathbf{n}_k^l\,, \quad \mathbf{n}_k^l = \mathcal{N}(\mathbf{0}, \mathbf{R})$$

- x_l et y_l sont les vrais coordonnées du point de repère l
- x_k et y_k et θ_k représentent la pose actuelle du véhicule
- d est la distance connue entre le centre du moteur et le télémètre laser (LIDAR)
- Le bruit de mesure de point de repère $\mathbf{n}^l{}_k$ a une distribution normale (moyenne nulle) avec une covariance constante $\mathbf{R}$.

- **Mise en Route:**

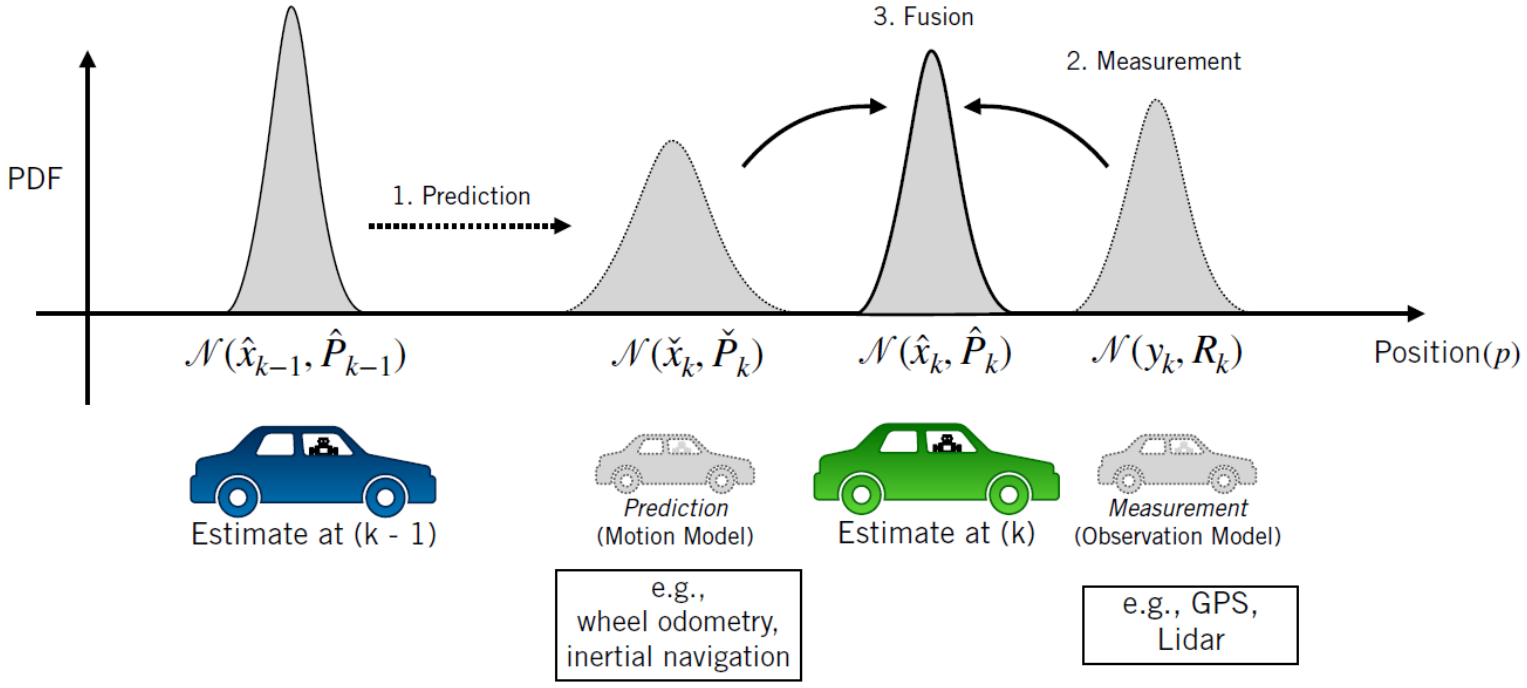
Étant donné que les modèles ci-dessus sont non linéaires, nous recommandons d'utiliser le filtre de Kalman étendu (FKE) comme estimateur et poursuite d'état $\mathbf{x}_k = [x_k \quad y_k \quad \theta_k\ ]^T$. Plus précisément, vous devrez fournir du code implémentant les étapes suivantes:

- l'étape de prédiction, qui utilise des mesures d'odométrie et le modèle d'état de mouvement pour produire une estimation prédite de pose avec sa covariance d'erreur à un instant donné
- l'étape de correction, qui utilise les mesures de distance et d'angle fournies par le LIDAR pour corriger les estimations de pose avec sa covariance d'erreur.

- **Mise en Route:**

Étant donné que les modèles ci-dessus sont non linéaires, nous recommandons d'utiliser le filtre de Kalman étendu (FKE) comme estimateur et poursuite d'état $\mathbf{x}_k = [x_k \quad y_k \quad \theta_k]^T$. Plus précisément, vous devrez fournir du code implémentant les étapes suivantes:

- l'étape de prédiction, qui utilise des mesures d'odométrie et le modèle d'état de mouvement pour produire une estimation prédite de pose avec sa covariance d'erreur à un instant donné
- l'étape de correction, qui utilise les mesures de distance et d'angle fournies par le LIDAR pour corriger les estimations de pose avec sa covariance d'erreur.



- **Décompressez les données:**

Commençons par décompressez les données les données disponibles mydata.mat dans le dossier data. (fonction matlab: load)

```
t = double(data.t);  % timestamps [s]
x_init  =  % initial x position [m]
y_init  =  % initial y position [m]
th_init =  % initial theta position [rad]

%input signal

v  =  % translational velocity input [m/s]
om =  % rotational velocity input [rad/s]

% bearing and range measurements, LIDAR constants

b =    % bearing to each landmarks center in the frame attached to the laser [rad]
r =    % range measurements [m]
l =    % x,y positions of landmarks [m]
d =    % distance between robot center and laser rangefinder [m]
```

Note:

Pour que les estimations d'orientation coïncident avec les mesures d'angle de LIDAR, il est également nécessaire de convertir toutes les valeurs θ estimées dans l'interval $(-\pi, \pi]$ en utilisant la fonction matlab wrapToPi

- **Paramètres d'initialisation:**

Maintenant que nos données sont chargées, nous pouvons commencer à configurer les choses pour notre solveur. L'un des aspects les plus importants de la conception d'un filtre consiste à déterminer les matrices de covariance du bruit d'état **Q** et de mesure **R**, ainsi que la pose initial et sa matrice de covariance. Nous définissons les valeurs ici :

```
v_var = 0.01;  % translation velocity variance
om_var = 0.01;  % rotational velocity variance
r_var = 0.1;  % range measurements variance
b_var = 0.1;  % bearing measurement variance

Qkm =          % Process noise covariance
Rk =           % measurement noise covariance

x_est = zeros(3, length(v));  % estimated states, x, y, and theta
P_est = zeros( 3, 3, length(v));  % state covariance matrices

x_est(:,1) =     % initial state
P_est(:,:,1) = diag([0, 0, 0]); % initial state covariance
```

- **Étape de correction de l'KFE:**

Tout d'abord, implémentons la fonction de mise à jour par des mesures, qui prend une mesure de point de repère disponible l et met à jour l'estimation prédite de la pose $\breve{\mathbf{x}}_k$. Pour chaque mesure de point de repère reçue à un pas de temps donné k, vous devez implémenter les étapes suivantes:

$$\mathbf{y}_k^l = \begin{bmatrix} \sqrt{(x_l - x_k - d\cos\theta_k)^2 + (y_l - y_k - d\sin\theta_k)^2} \\ atan2\,(y_l - y_k - d\sin\theta_k, x_l - x_k - d\cos\theta_k) - \theta_k \end{bmatrix} + \mathbf{n}_k^l, \quad \mathbf{n}_k^l = \mathcal{N}(\mathbf{0}, \mathbf{R})$$

- Calculer les jacobiennes du modèle de mesure à $\breve{\mathbf{x}}_k$:

$$\mathbf{y}_k^l = \mathbf{h}(\mathbf{x}_k, \mathbf{n}_k^l)$$

$$\mathbf{H}_k = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}_k}\right|_{\breve{\mathbf{x}}_k, 0}, \quad \mathbf{M}_k = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{n}_k}\right|_{\breve{\mathbf{x}}_k, 0}$$

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial x_1} & \cdots & \frac{\partial \mathbf{f}}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

- Calculer le gain de Kalman:

$$\mathbf{K}_k = \check{\mathbf{P}}_k \mathbf{H}_k^T \left(\mathbf{H}_k \check{\mathbf{P}}_k \mathbf{H}_k^T + \mathbf{M}_k \mathbf{R}_k \mathbf{M}_k^T\right)^{-1}$$

- Corriger l'estmation prédite:

$$\check{\mathbf{y}}_k^l = \mathbf{h}\left(\check{\mathbf{x}}_k, \mathbf{0}\right)$$
$$\hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k \left(\mathbf{y}_k^l - \check{\mathbf{y}}_k^l\right)$$

- Corriger la matrice de covariance d'erreur:

$$\hat{\mathbf{P}}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\, \check{\mathbf{P}}_k$$

```
function [x_hat, P_hat] = EKF_Correction_Step(lk, rk, bk, P_check, x_check, R, d)

%Compute measurement Jacobian Hk and Mk
xl = lk(1);
yl = lk(2);
xk = x_check(1);
yk = x_check(2);
thk = wrapToPi(x_check(3));

Hk = …
Mk = …




%Compute Kalman Gain
Kk =

%Correct predicted state (remember to wrap the angles to [-pi,pi])
ykl_check = zeros(2,1);

ykl = zeros(2,1);

x_hat = zeros(3,1);

%Correct covariance
P_hat =
```

- **Étape de prédiction de l'KFE:**

Maintenant, implémentons la fonction de prédiction, en définissant l'étape de prédiction du FKE à l'aide du modèle d'état de mouvement fourni :

$$\mathbf{x}_k = \mathbf{x}_{k-1} + T \begin{bmatrix} \cos\theta_{k-1} & 0 \\ \sin\theta_{k-1} & 0 \\ 0 & 1 \end{bmatrix} \left(\begin{bmatrix} v_k \\ \omega_k \end{bmatrix} + \mathbf{w}_k \right), \quad \mathbf{w}_k = \mathcal{N}(\mathbf{0}, \mathbf{Q})$$

- Calculer les jacobiennes du modèle d'état par rapport à la pose à l'instant précédent $\boldsymbol{x}_{k-1}$ et par rapport au bruit d'état $\mathbf{w}_k$:

$$\mathbf{x}_k = \mathbf{f}\ (\mathbf{x}_{k-1}, \mathbf{u}_k\ , \mathbf{w}_k\ )$$

$$\mathbf{F}_{k-1} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}_{k-1}}\right|_{\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k, 0}, \quad \mathbf{L}_{k-1} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{w}_k}\right|_{\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k, 0}$$

- Mettre à jour la pose avec les lectures d'odométrie:

$$\check{\mathbf{x}}_k = \mathbf{f}\left(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k\ , \mathbf{0}\right)$$

- Calculer la matrice de covariance d'erreur de la prédiction:

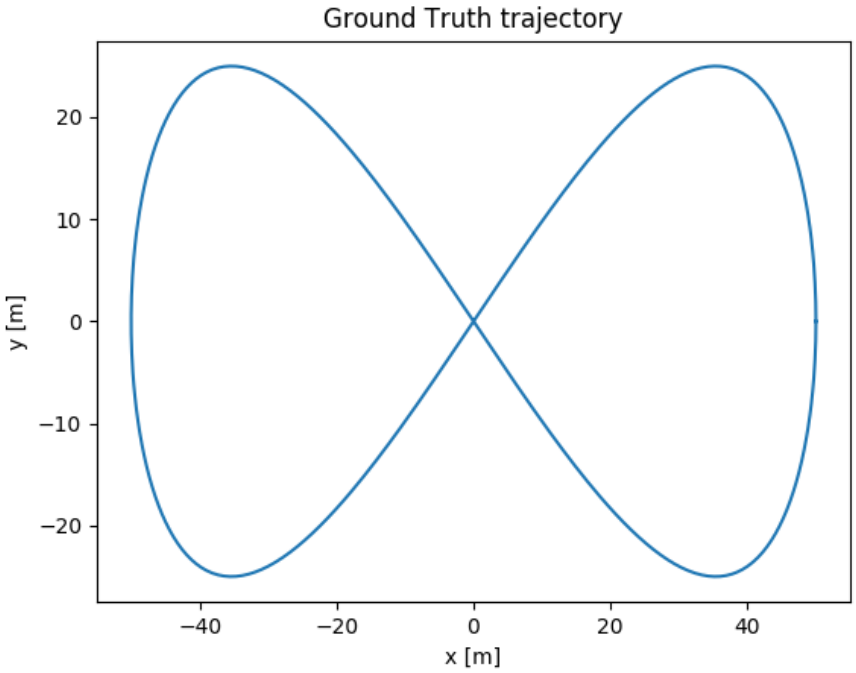
$$\check{\mathbf{P}}_k = \mathbf{F}_{k-1} \hat{\mathbf{P}}_{k-1} \mathbf{F}_{k-1}^T + \mathbf{L}_{k-1} \mathbf{Q}_{k-1} \mathbf{L}_{k-1}^T$$

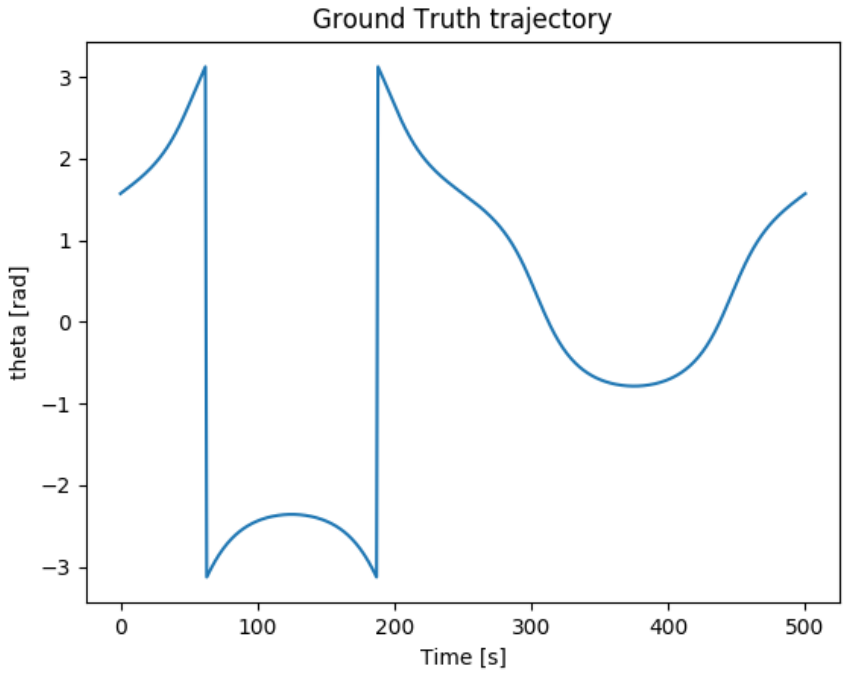
```
function [x_check, P_check] = EKF_Predection_Step(delta_t, vk,omk, P_hat, x_hat, Qkm)

%Update state with odometry readings (remember to wrap the angles to [-pi,pi])
th_prev = wrapToPi(x_hat(3));

x_check = …

%Compute Process model Jacobians with respect to last state
F_km = …

%Compute Process model jacobian with respect to noise
L_km = …

%Prediction covariance
P_check = …
```

```
t = double(data.t);  % timestamps [s]
x_init  =  % initial x position [m]
y_init  =  % initial y position [m]
th_init =  % initial theta position [rad]
%input signal
v  =  % translational velocity input [m/s]
om =  % rotational velocity input [rad/s]
% bearing and range measurements, LIDAR constants
b =    % bearing to each landmarks center in the frame attached to the laser [rad]
r =    % range measurements [m]
l =    % x,y positions of landmarks [m]
d =    % distance between robot center and laser rangefinder [m]
v_var = 0.01;  % translation velocity variance
om_var = 0.01;  % rotational velocity variance
r_var = 0.1;  % range measurements variance
b_var = 0.1;  % bearing measurement variance
Qkm =         % Process noise covariance
Rk =          % measurement noise covariance
x_est = zeros(3, length(v));  % estimated states, x, y, and theta
P_est = zeros( 3, 3, length(v));  % state covariance matrices
x_est(:,1) =     % initial state
P_est(:,:,1) = diag([0, 0, 0]); % initial state covariance
for k=2:length(t) % start at 2 because we've set the initial estimation
    delta_t = t(k) - t(k - 1);  %time step (difference between timestamps)
    [x_check, P_check] = …

    %Update state estimate using available landmark measurements
    for i =1: size(r,2)
        [x_check, P_check] = …
    end
%Set final state predictions for timestep
x_est(:, k) = x_check;
P_est(:, :,k) = P_check;
end
```

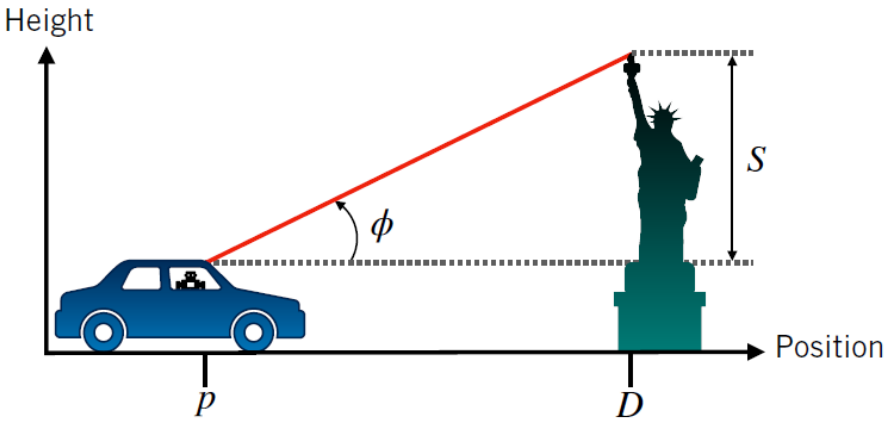
- **<u>Traçons les estimations des poses résultantes:</u>**

La trajectoire résultante (positionnement et orientation) devrait ressembler étroitement à la vraie trajectoire, avec des "sauts" mineurs dans l'estimation de l'orientation dus à l'enveloppement d'angle.





- **Annexe sur EKF et les calcules des matrices jacobiennes:**



$$\mathbf{x} = \begin{bmatrix} p \\ \dot{p} \end{bmatrix} \qquad \mathbf{u} = \ddot{p}$$

S and *D* are known in advance

$$\mathbf{x}_k = \mathbf{f}_{k-1}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1})$$

current state, previous state, inputs, process noise

$$\mathbf{y}_k = \mathbf{h}_k(\mathbf{x}_k, \mathbf{v}_k)$$

measurement, state, measurement noise

Motion/Process model

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1})$$
$$= \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \mathbf{x}_{k-1} + \begin{bmatrix} 0 \\ \Delta t \end{bmatrix} \mathbf{u}_{k-1} + \mathbf{w}_{k-1}$$

Landmark measurement model

$$y_k = \phi_k = h(p_k, v_k)$$
$$= \tan^{-1}\left(\frac{S}{D - p_k}\right) + v_k$$

Noise densities

$$v_k \sim \mathcal{N}(0,\ 0.01) \qquad \mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, (0.1)\mathbf{1}_{2\times 2})$$
$$\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k) \qquad \mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)$$

Motion/Process model

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1})$$
$$= \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \mathbf{x}_{k-1} + \begin{bmatrix} 0 \\ \Delta t \end{bmatrix} \mathbf{u}_{k-1} + \mathbf{w}_{k-1}$$

Landmark measurement model

$$y_k = \phi_k = h(p_k, v_k)$$
$$= \tan^{-1}\left(\frac{S}{D - p_k}\right) + v_k$$

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial x_1} & \cdots & \frac{\partial \mathbf{f}}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Motion model Jacobians

$$\mathbf{F}_{k-1} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}_{k-1}}\right|_{\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbf{0}} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$$

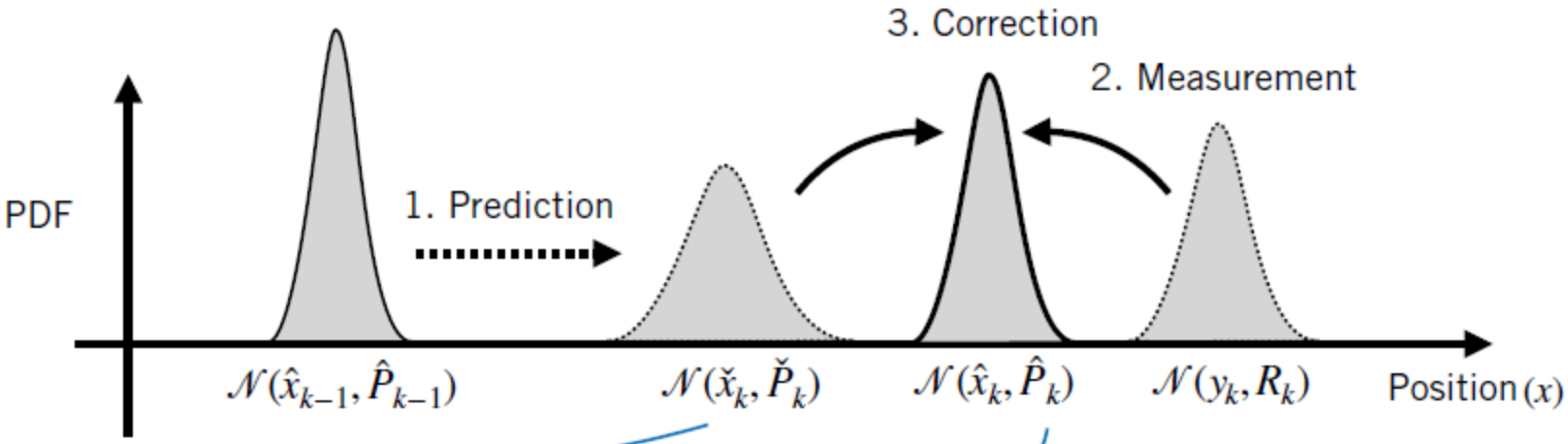
$$\mathbf{L}_{k-1} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{w}_{k-1}}\right|_{\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbf{0}} = \mathbf{1}_{2\times 2}$$

Measurement model Jacobians

$$\mathbf{H}_k = \left.\frac{\partial h}{\partial \mathbf{x}_k}\right|_{\check{\mathbf{x}}_k, \mathbf{0}} = \begin{bmatrix} \frac{S}{(D - \check{p}_k)^2 + S^2} & 0 \end{bmatrix}$$

$$M_k = \left.\frac{\partial h}{\partial v_k}\right|_{\check{\mathbf{x}}_k, \mathbf{0}} = 1$$

- **EKF Algorithme:**



1. Prediction

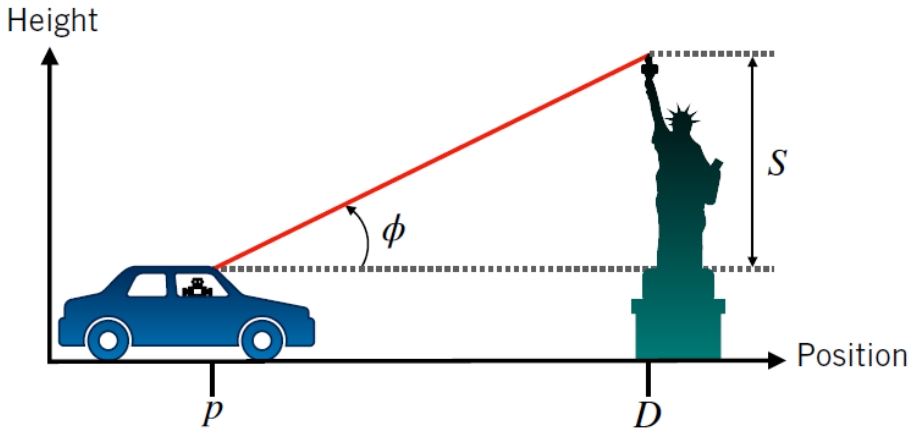
$$\begin{cases} \check{\mathbf{x}}_k = \mathbf{f}_{k-1}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbf{0}) \\ \check{\mathbf{P}}_k = \mathbf{F}_{k-1}\hat{\mathbf{P}}_{k-1}\mathbf{F}_{k-1}^T + \mathbf{L}_{k-1}\mathbf{Q}_{k-1}\mathbf{L}_{k-1}^T \end{cases}$$

2/3. Measurement & Correction

$$\begin{cases} \mathbf{K}_k = \check{\mathbf{P}}_k\mathbf{H}_k^T(\mathbf{H}_k\check{\mathbf{P}}_k\mathbf{H}_k^T + \mathbf{M}_k\mathbf{R}_k\mathbf{M}_k^T)^{-1} \\ \hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k(\mathbf{y}_k - \mathbf{h}_k(\check{\mathbf{x}}_k, \mathbf{0})) \\ \hat{\mathbf{P}}_k = (\mathbf{1} - \mathbf{K}_k\mathbf{H}_k)\check{\mathbf{P}}_k \end{cases}$$

$\check{\mathbf{x}}_k$ Prediction (given motion model) at time k

$\hat{\mathbf{x}}_k$ Corrected prediction (given measurement) at time k



Data

$$\hat{\mathbf{x}}_0 \sim \mathcal{N}\left(\begin{bmatrix}0\\5\end{bmatrix}, \begin{bmatrix}0.01 & 0\\0 & 1\end{bmatrix}\right)$$

$\Delta t = 0.5$ s

$u_0 = -2\ [m/s^2]$ $\quad y_1 = \pi/6$ [rad]

$S = 20\ [m]$ $\quad D = 40\ [m]$

Using the Extended Kalman Filter equations, what is our updated position?

$\hat{p}_1$

Prediction

$$\check{\mathbf{x}}_1 = \mathbf{f}_0(\hat{\mathbf{x}}_0, \mathbf{u}_0, \mathbf{0})$$

$$\begin{bmatrix} \check{p}_1 \\ \check{\dot{p}}_1 \end{bmatrix} = \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 5 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} (-2) = \begin{bmatrix} 2.5 \\ 4 \end{bmatrix}$$

$$\check{\mathbf{P}}_1 = \mathbf{F}_0 \hat{\mathbf{P}}_0 \mathbf{F}_0^T + \mathbf{L}_0 \mathbf{Q}_0 \mathbf{L}_0^T$$

$$\check{\mathbf{P}}_1 = \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0.01 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0.5 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.36 & 0.5 \\ 0.5 & 1.1 \end{bmatrix}$$

Correction

$$\mathbf{K}_1 = \check{\mathbf{P}}_1\mathbf{H}_1^T(\mathbf{H}_1\check{\mathbf{P}}_1\mathbf{H}_1^T + \mathbf{M}_1\mathbf{R}_1\mathbf{M}_1^T)^{-1}$$

$$= \begin{bmatrix} 0.36 & 0.5 \\ 0.5 & 1.1 \end{bmatrix} \begin{bmatrix} 0.011 \\ 0 \end{bmatrix} \left(\begin{bmatrix} 0.011 & 0 \end{bmatrix} \begin{bmatrix} 0.36 & 0.5 \\ 0.5 & 1.1 \end{bmatrix} \begin{bmatrix} 0.011 \\ 0 \end{bmatrix} + 1(0.01)(1) \right)^{-1}$$

$$= \begin{bmatrix} 0.40 \\ 0.55 \end{bmatrix}$$

$$\hat{\mathbf{x}}_1 = \check{\mathbf{x}}_1 + \mathbf{K}_1(\mathbf{y}_1 - \mathbf{h}_1(\check{\mathbf{x}}_1, \mathbf{0}))$$

$$\begin{bmatrix} \hat{p}_1 \\ \hat{\dot{p}}_1 \end{bmatrix} = \begin{bmatrix} 2.5 \\ 4 \end{bmatrix} + \begin{bmatrix} 0.40 \\ 0.55 \end{bmatrix} (0.52 - 0.49) = \begin{bmatrix} 2.51 \\ 4.02 \end{bmatrix}$$

Bonus!

$$\hat{\mathbf{P}}_1 = (\mathbf{1} - \mathbf{K}_1\mathbf{H}_1)\check{\mathbf{P}}_1$$

$$= \begin{bmatrix} 0.36 & 0.50 \\ 0.50 & 1.1 \end{bmatrix}$$