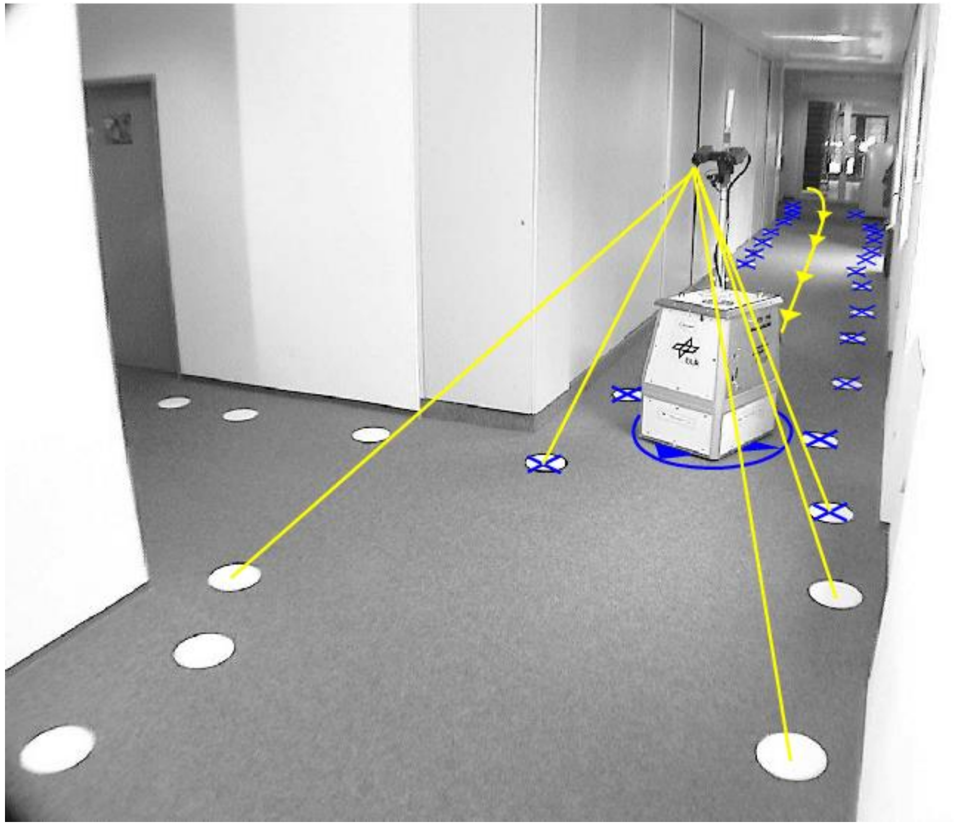
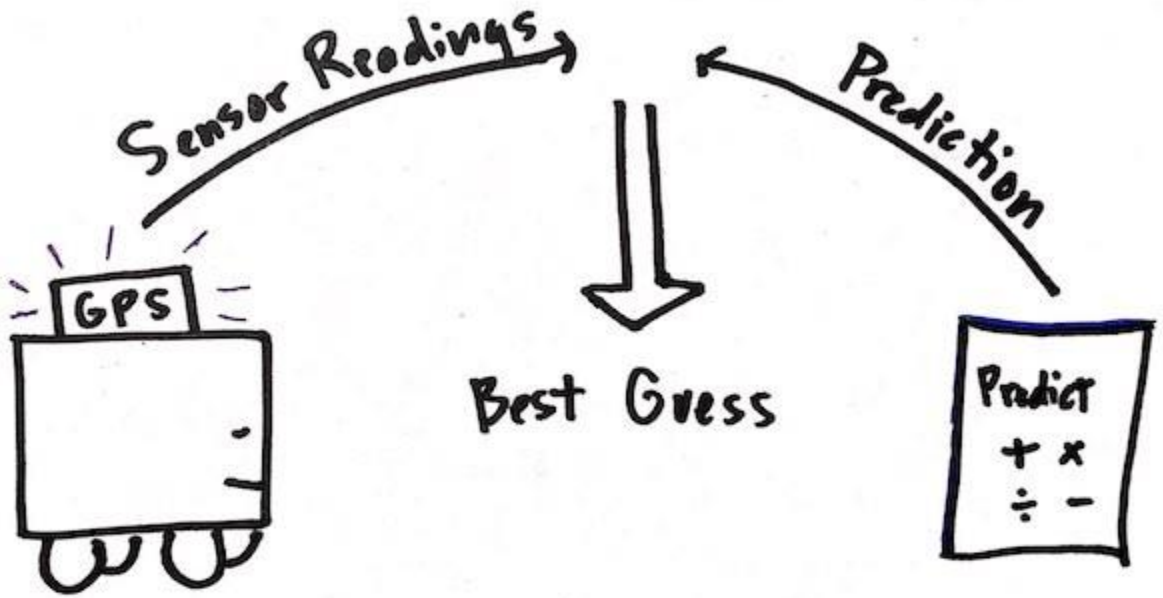
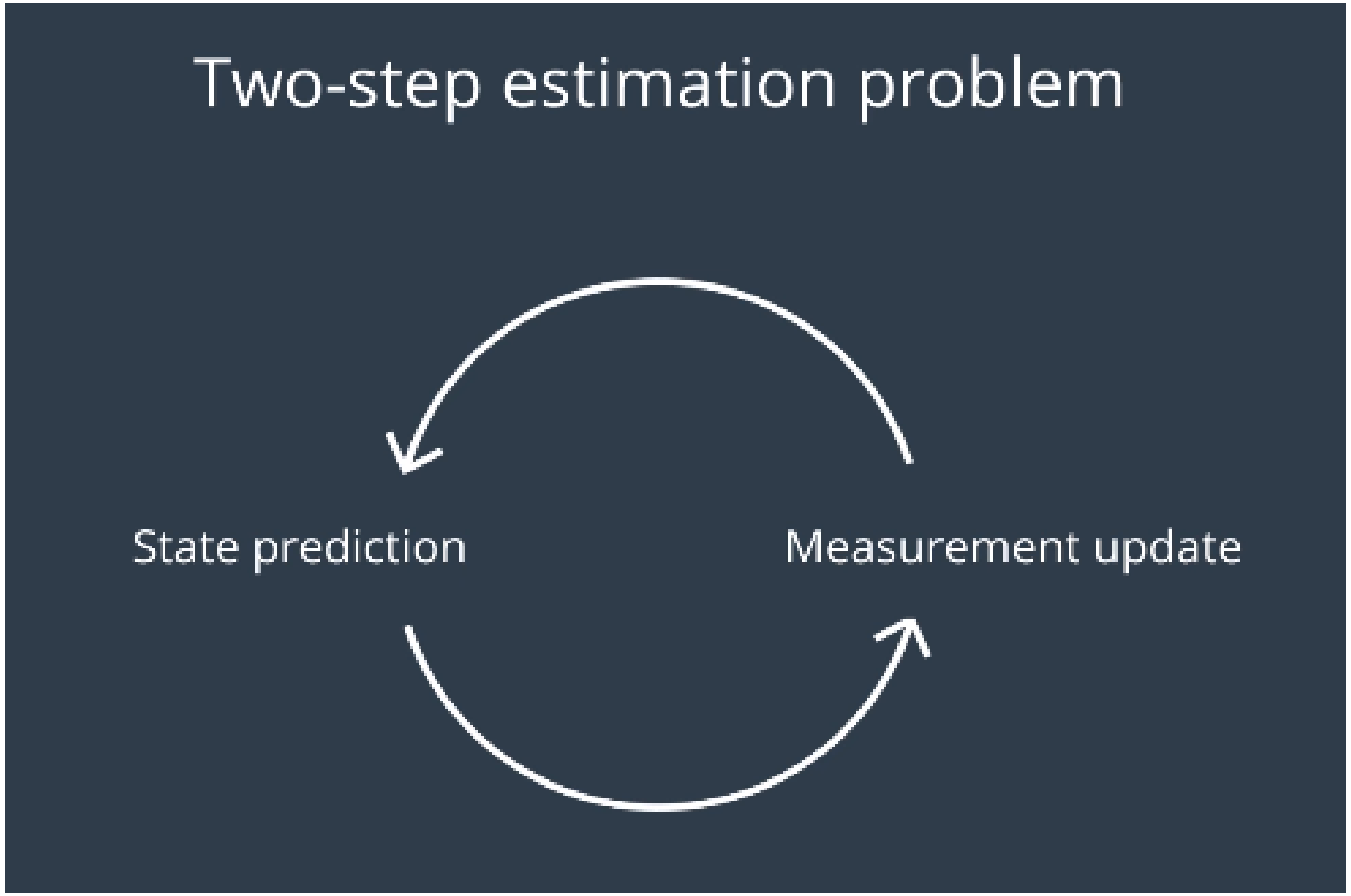
Kalman filter

Localization problem

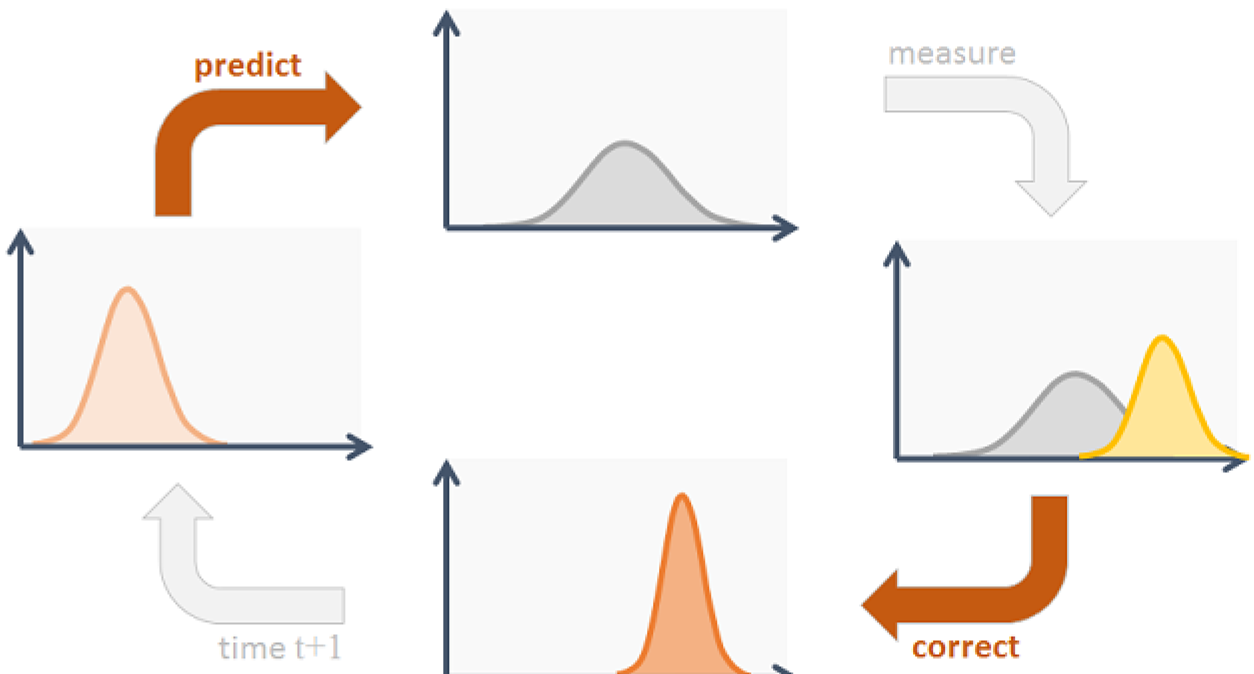


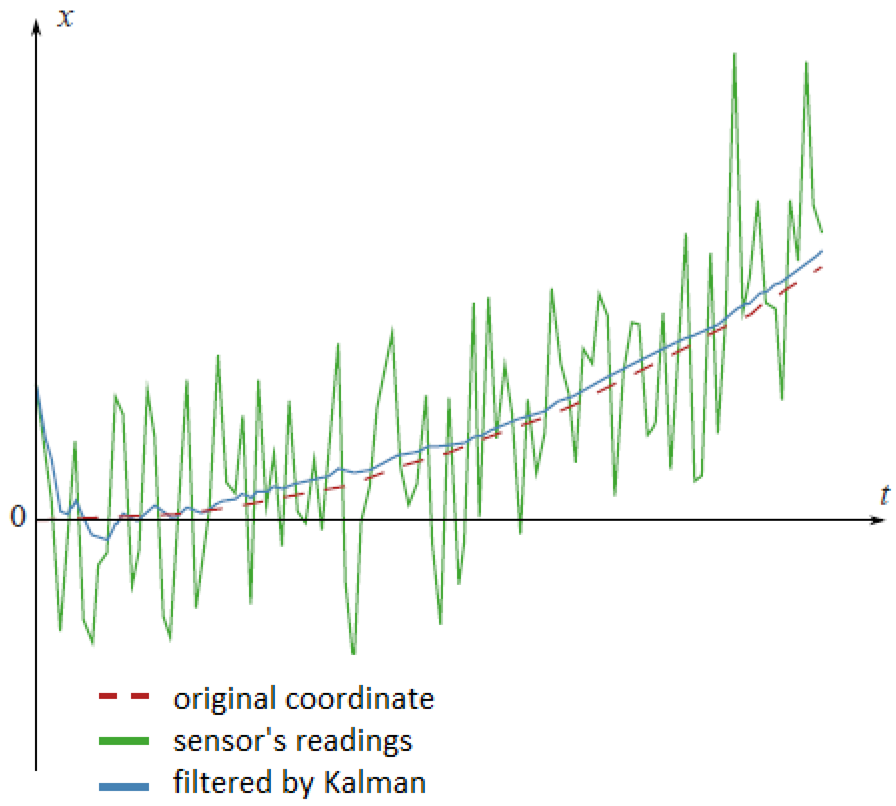




State estimate

State Estimator

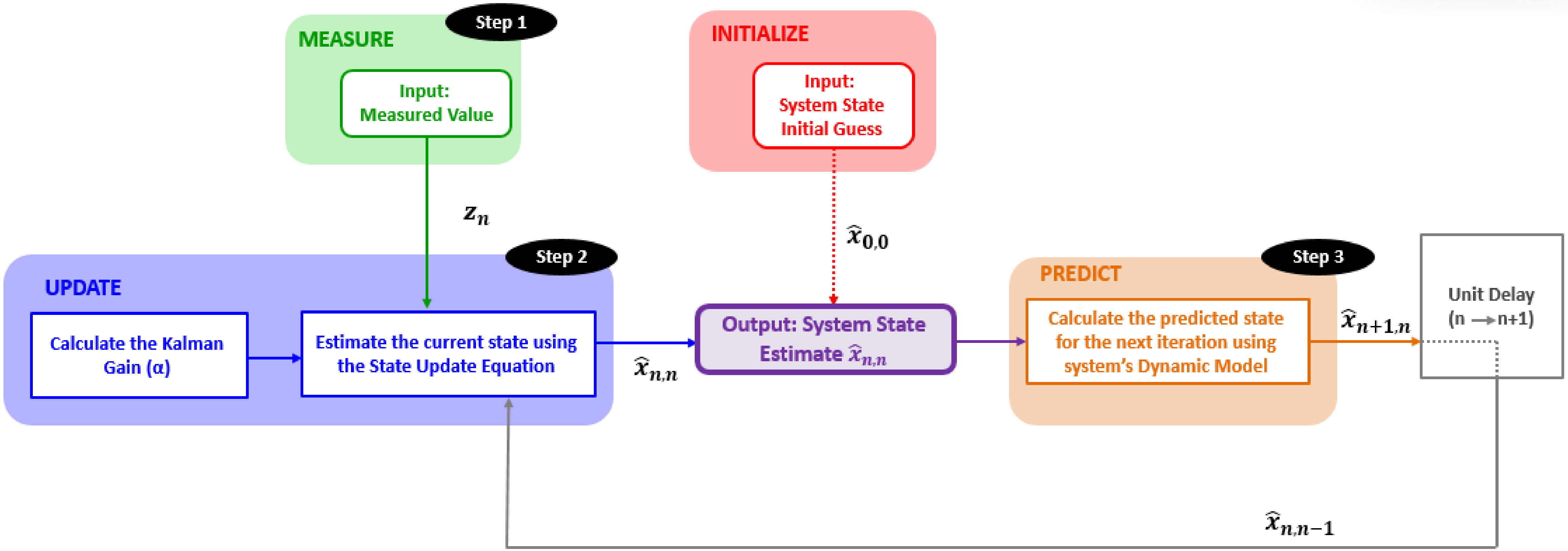




The Black Magic of the Kalman Filter

- This is the Estimation Algorithm of the Kalman filter
- Estimating the current state from the predicted values plus Kalman Gain multiplied by the Error

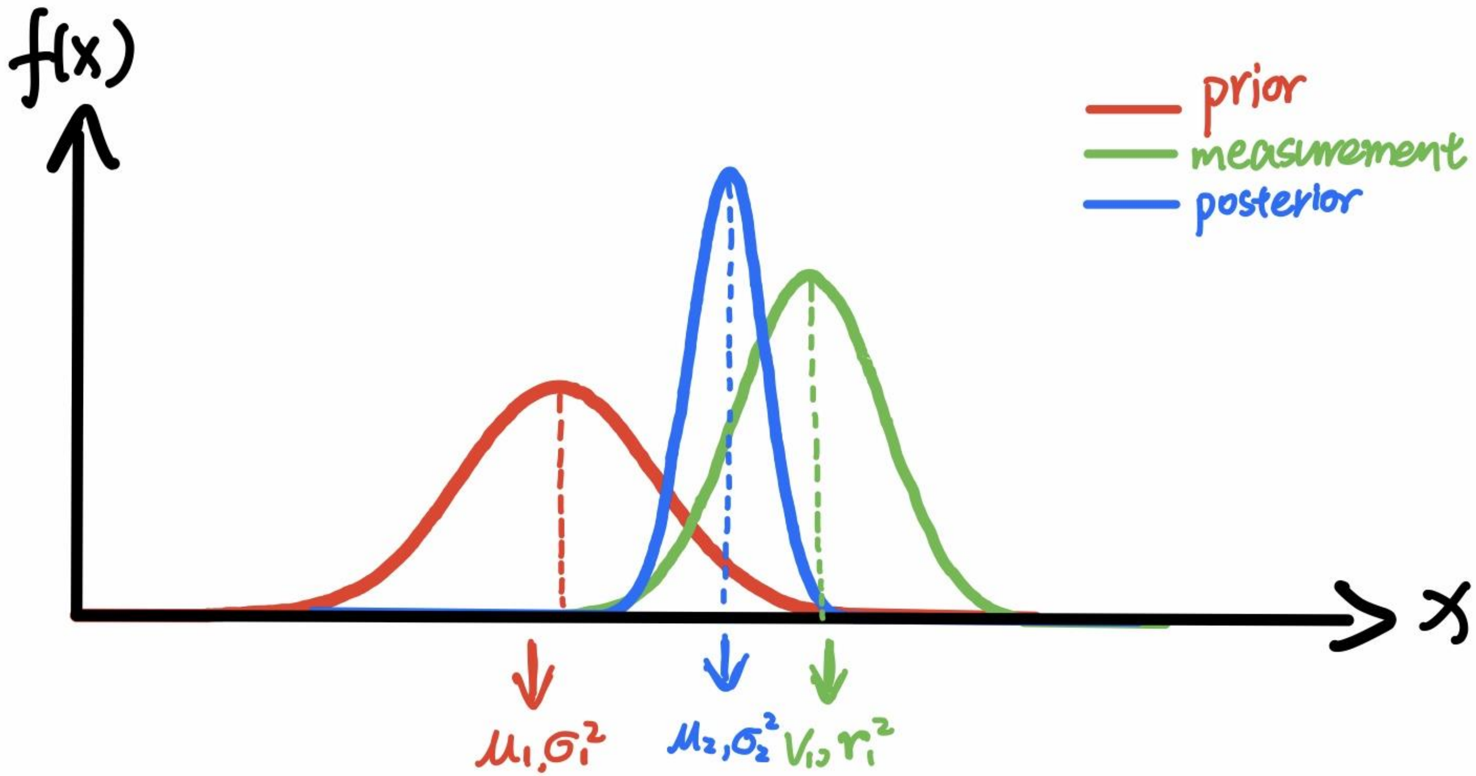
$$\boxed{\text{The estimate of the current state}} = \boxed{\text{Predicted value of the current state}} + \boxed{\text{Factor}} \times \left(\boxed{\text{Measurement}} - \boxed{\text{Predicted value of the current state}} \right)$$


MEASURE
Step 1
Input:
Measured Value
INITIALIZE
Input:
System State
Initial Guess
z_n
$\hat{x}_{0,0}$
UPDATE
Step 2
Calculate the Kalman Gain (α)
Estimate the current state using the State Update Equation
$\hat{x}_{n,n}$
Output: System State Estimate $\hat{x}_{n,n}$
PREDICT
Step 3
Calculate the predicted state for the next iteration using system's Dynamic Model
$\hat{x}_{n+1,n}$
Unit Delay
(n → n+1)
$\hat{x}_{n,n-1}$

How it works?

Designing a Kalman Filter

The system position



Kalman Filter types

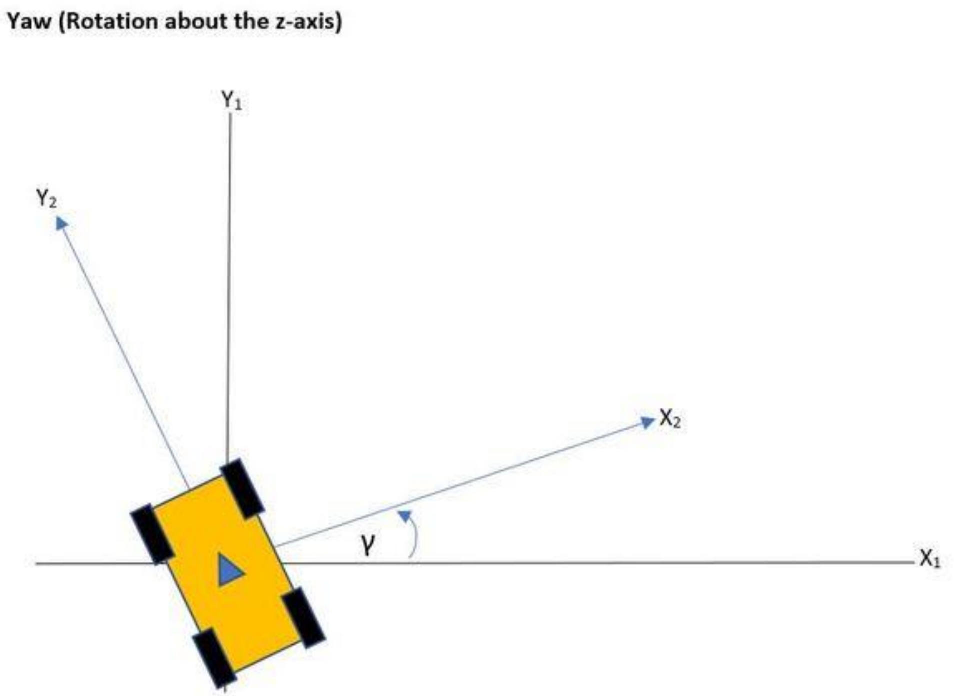
- Linear Kalman Filter
- Extended Kalman filter (Multi-dimentional)

Kalman Filter
Impl. with Python

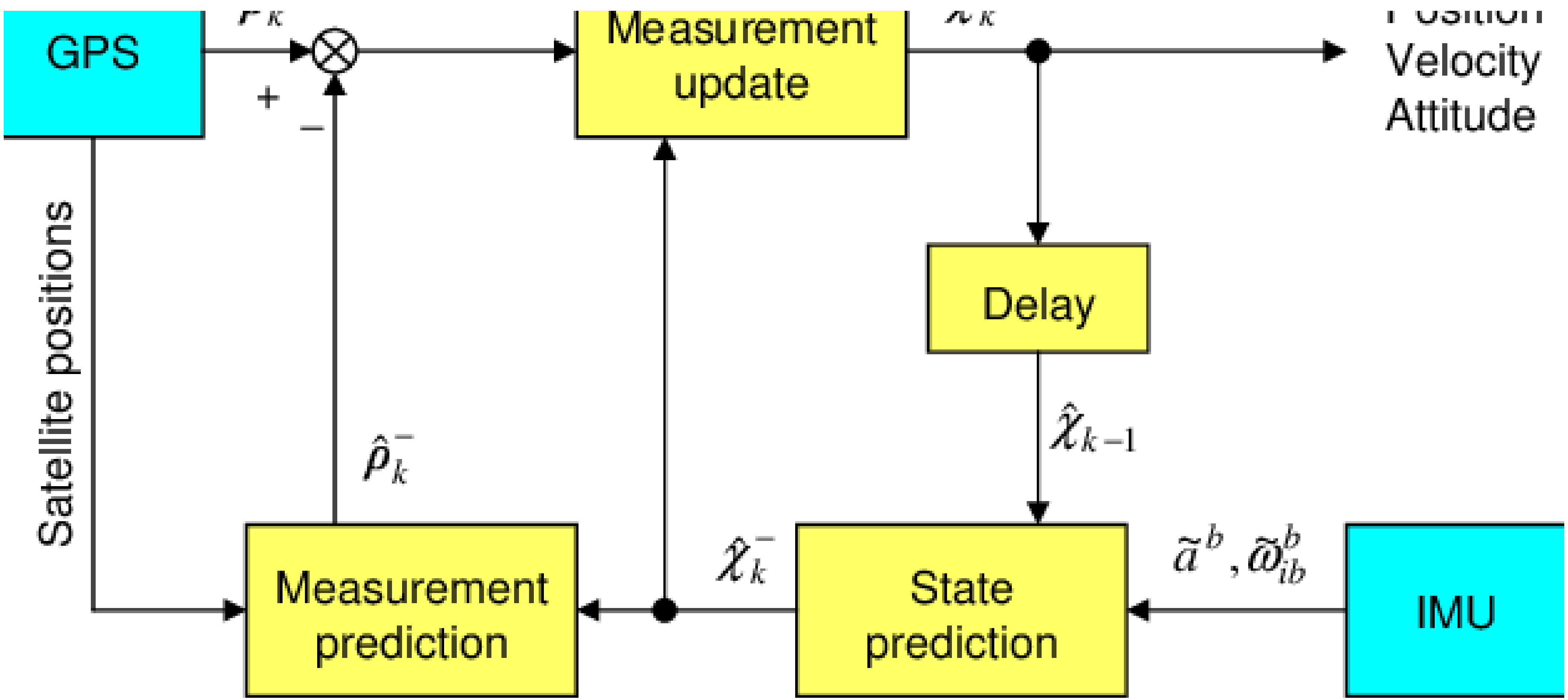
```
def Kalman_Filter() :
  for n in range(measurements):
     x = A*x+B*u[n]
     P = A*P*A.T + Q

 # Measurement Update (Correction)
 # ===============================
 # Compute the Kalman Gain
    S = H*P*H.T + R
    K = (P*H.T) * np.linalg.pinv(S)


# Update the estimate via z
    Z = mx[n]
    y = Z — (H*x) # Innovation or Residual
    x = x + (K*y)

 # Update the error covariance
    P = (I — (K*H))*P
```

Yaw Angle

- Now, there are four possible ways to update yaw information:
- From a model using inputs (throttle and steering angle).
- Magnetometer for angle relative to magnetic North.
- Gyro readings from IMU
- Heading required to move in a straight line from the previous GPS fix to the current GPS fix.



Robot localization



Motion equation

Velocity final equals the initial velocity plus acceleration multiplied by the time

$v = v_0 + at$ [1]

Final Position is calculated as the initial position plus initial velicity multiplied by the time plus half of the acceleration multiplied by the time square

$p = p_0 + v_0 t + ½at^2$ [2]

Extended Kalman filter

Extended Kalman Filter Overview

Initial state

$$X_0 \quad P_0$$

Initial state becomes previous

Previous state

$$X_{k-1} \quad P_{k-1}$$

Subscript k represents each state iteration cycle

Current state becomes previous

New state (predicted ['], based on physical model and previous state)

$$X'_k = F(X,U) + W_k$$
$$P'_k = AP_{k-1}A^T + Q_k$$

$F(X,U)$= Non-linear state transition matrix. A function of the state and control variables.

W= Predicted state noise matrix

Q= Process noise covariance matrix. Keeps the state covariance matrix from becoming too small or going to 0.

A,B,C= Adaptation matrices, to convert input state to process state

Output of updated state

$$X_k \quad P_k$$

X= State matrix

P= Process covariance matrix (represents error in the estimate)

State and covariance update

$$X_k = X'_k + K(Y_k - h(X'))$$
$$P_k = (I - KH_j)P'_k$$

I= Identity matrix

$h(X')$= Non-linear measurement matrix

Calculate Kalman gain

$$K = \frac{P'_k H_j}{H_j P'_k H_j^T + R}$$

K=Kalman gain

R= Sensor noise/measurement covariance matrix

H_j= Jacobian Matrix (also ensures size consistency)

Measurement from sensor

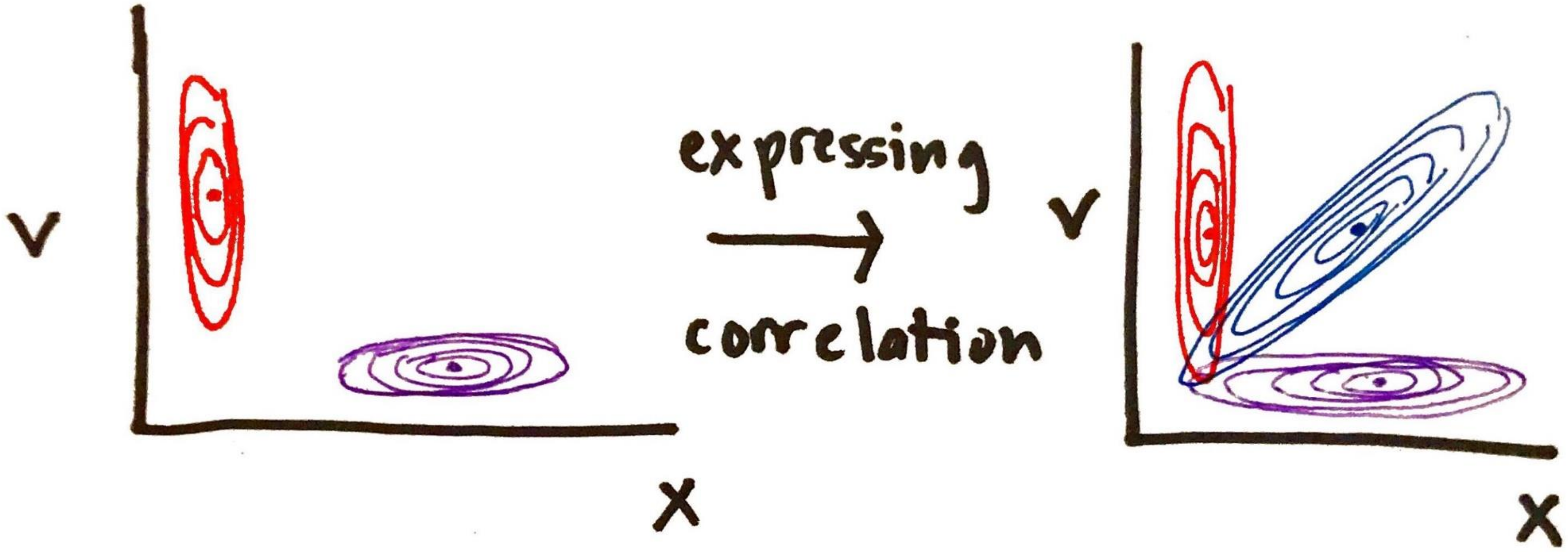
$$Y_t = CX^*_k + Z_k$$

Y= Measurement of state

Z = measurement noise

Estimated state

$$\hat{x} = \begin{bmatrix} x \\ v \end{bmatrix} = \begin{bmatrix} \text{position} \\ \text{velocity} \end{bmatrix}$$



Understanding Covariance Matrix

- *property*: symmetric
- diagonal elements: variances
- non-diagonal elements: covariances

$$C = \begin{bmatrix} V_x & C_{xy} \\ C_{yx} & V_y \end{bmatrix}$$

Parameters

with discrete time step k,

k-1: current state

k: next future state

Predictions

$\hat{x}_k$: Mean Vector (current best guess)

C_k: Covariance Matrix

P_k : Prediction Matrix, transforms data to give next future state

u_k : External Influence (Control) vector, takes into account effects of outside world

B_k : External Influence (Control) Matrix, takes into account effects of outside world

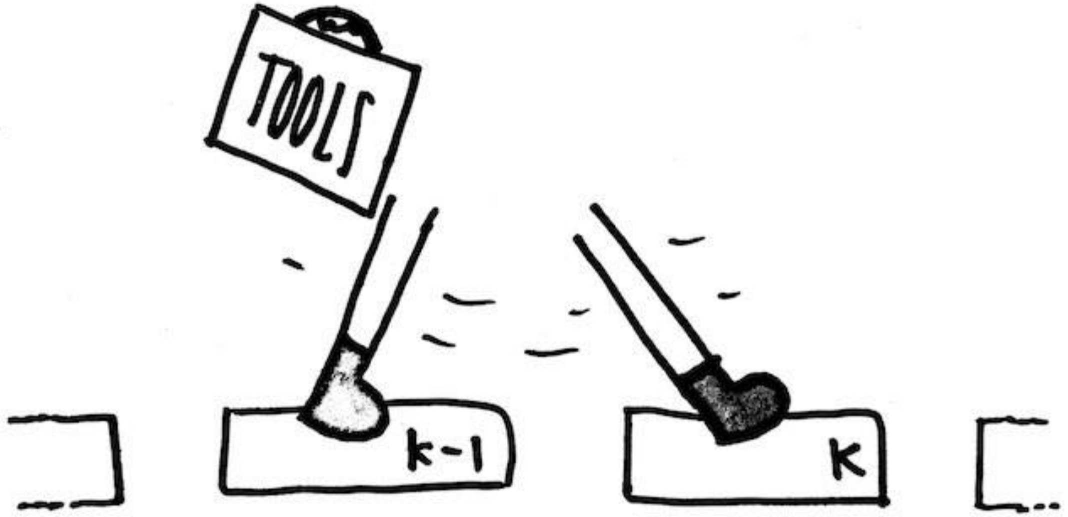
Q_k : Unaccounted External Influence Matrix, noise we may not know about

S_k : Sensor Matrix, scales predictions to units and scale of sensor readings

Sensor Data

r_k : Mean Vector of Sensor Reading

N_k : Covariance Matrix, contains sensor noise



$$\hat{X}_k = P_k \hat{X}_{k-1}$$

$$C_k = P_k C_{k-1} P^T$$

First step

predict

$$\hat{X}_k = P_k \hat{X}_{k-1} + \underbrace{B_k \vec{u}_k}_{\text{external influence}}$$

$$C_k = P_k C_{k-1} P^T + \underbrace{Q_k}_{\text{expanding covariance due to uncertainty}}$$

Second step

correct

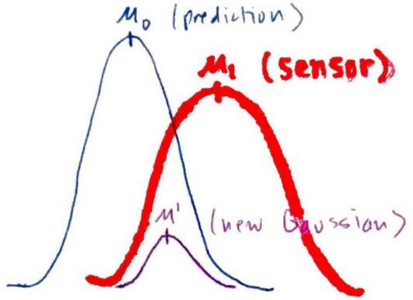
$$\mu_{expected} = S_k \hat{x}_k$$

$$\Sigma_{expected} = S_k C_k S_k^T$$

Third step

Transform Corrected Prediction (speak the same language as our sensors)

Fourth step



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

Read sensors

$$\underbrace{N(x, \mu_0, \sigma_0)}_{\text{prediction}} \cdot \underbrace{N(x, \mu_1, \sigma_1)}_{\text{sensor}} = \underbrace{N(x, \mu', \sigma')}_{\text{new Gaussian !!}}$$

Fifth step

$$\frac{1}{\sigma_0\sqrt{2\pi}} e^{\frac{-(x-\mu_0)^2}{2\sigma_0^2}} \cdot \frac{1}{\sigma_1\sqrt{2\pi}} e^{\frac{-(x-\mu_1)^2}{2\sigma_1^2}} = \frac{1}{\sigma'\sqrt{2\pi}} e^{\frac{-(x-\mu')^2}{2\sigma'^2}}$$

$$\Big\downarrow \text{fun algebra}$$

$$\mu' = \mu_0 + \frac{\sigma_0^2(\mu_1 - \mu_0)}{\sigma_0^2 + \sigma_1^2}$$

$$\sigma'^2 = \sigma_0^2 - \frac{\sigma_0^4}{\sigma_0^2 + \sigma_1^2}$$

factor out K

$$k = \frac{\sigma_0^2}{\sigma_0^2 + \sigma_1^2}$$

$$\mu' = \mu_0 + k(\mu_1 - \mu_0)$$

$$\sigma'^2 = \sigma_0^2 - k\sigma_0^2$$

Matrix Form

$$\underbrace{k = \frac{\Sigma_0}{\Sigma_0 + \Sigma_1}}_{\text{Kalman Gain}}$$

$$\mu' = \mu_0 + k(\mu_1 - \mu_0)$$

$$\Sigma' = \Sigma_0 - k\Sigma_0$$

Sixth step Find Kalman gain

Kalman filter simulation

- https://www.cs.utexas.edu/~teammco/misc/kalman_filter/
- Kalman filter tutorial 1
- https://www.youtube.com/watch?v=FkCT_LV9Syk
- Covariance
- https://www.youtube.com/watch?v=85Ilb-89sjk

- More tutorials
- https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python
- Other Localization Techniques MonteCarlo
- http://aslanfmh65.com/robotic-localization-kalman-filter-mcl/

Thank you