CENG4480

Lecture 08: Kalman Filter

Bei Yu

byu@cse.cuhk.edu.hk

(Latest update: October 31, 2018)

Fall 2018

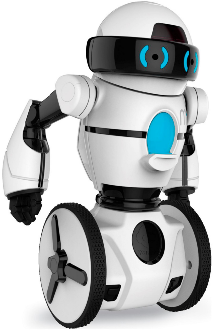
香港中文大學
The Chinese University of Hong Kong

Overview

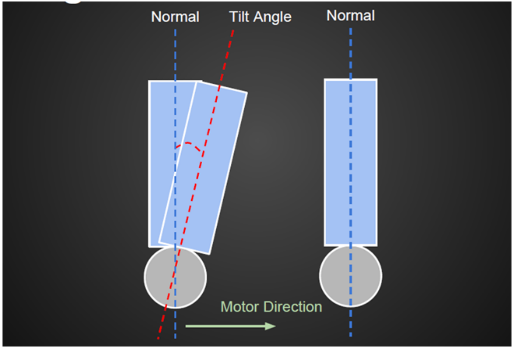
Overview

Self Balance Vehicle / Robot

- http://www.segway.com/
- http://wowwee.com/mip/

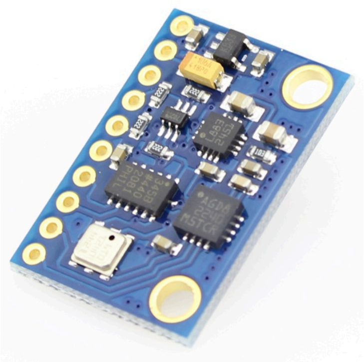


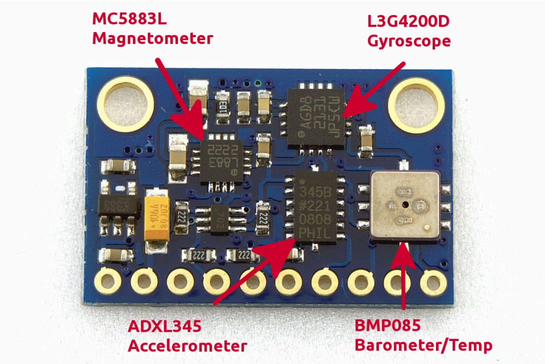
Basic Idea



Motion against the tilt angle, so it can stand upright.

IMU Board



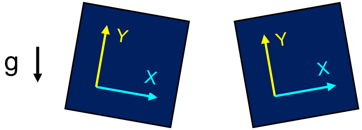


http://www.hotmcu.com/imu-10dof-l3g4200dadxl345hmc5883lbmp180-p-190.html

- L3G4200D: gyroscope, measure angular rate (relative value)
- ADXL345: accelerometer, measure acceleration

Overview

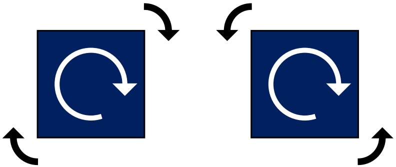
Complementary Filter



X now sees some gravity.

X reads slightly positive. X reads slightly negative

Accelerometer



Gyro reads positive. Gyro reads negative.

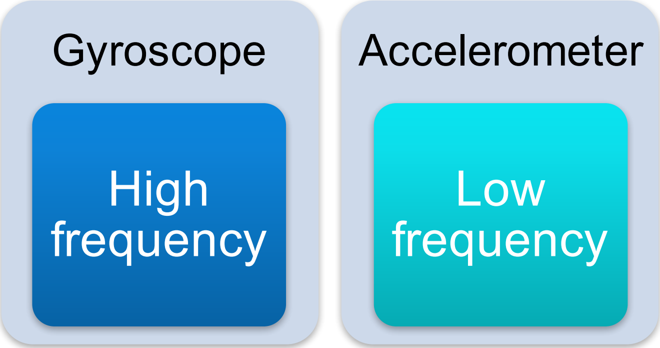
Gyroscope

- Give accurate reading of tilt angle
- Slower to respond than Gyro's
- prone to vibration/noise

- response faster
- but has drift over time

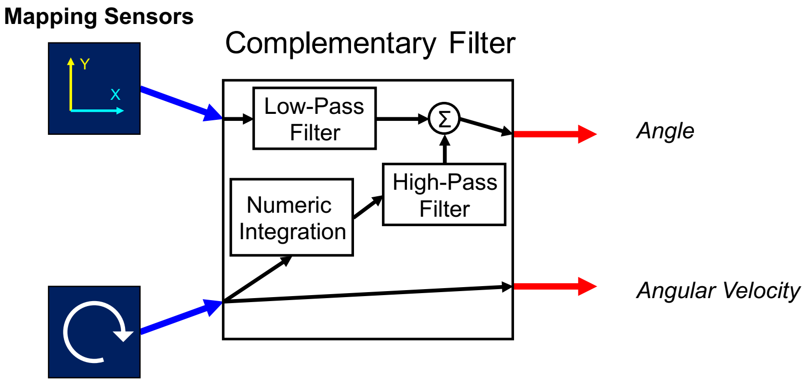
Complementary Filter (cont.)

- Since



- Combine two sensors to find output

Complementary Filter (cont.)




```
Read_acc();
Read_gyro();
Ayz=atan2(RwAcc[1],RwAcc[2])*180/PI;               //angle by accelerometer
Ayz-=offset;                                        //adjust to correct
Angy = 0.98*(Angy+GyroIN[0]*interval/1000)+0.02*Ayz; //complement filter
```

Overview

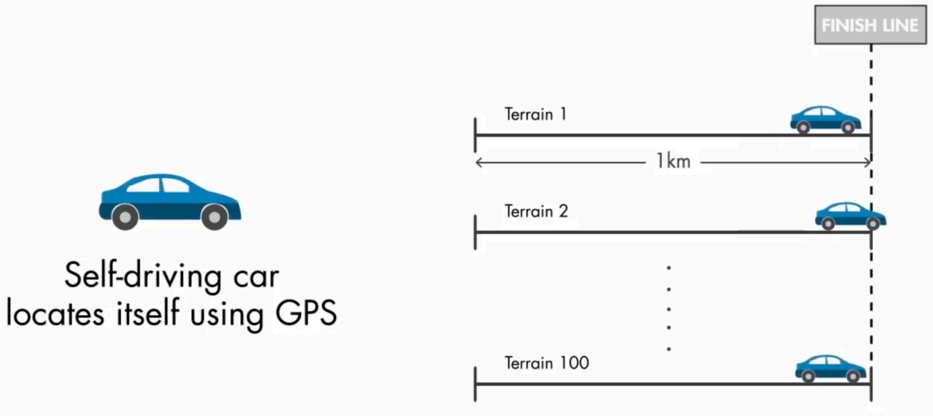
Rudolf Kalman (1930 – 2016)

- Born in Budapest, Hungary
- BS in 1953 and MS in 1954 from MIT electrical engineering
- PhD in 1957 from Columbia University.

- Famous for his co-invention of the Kalman filter – widely used in control systems to extract a signal from a series of incomplete and noisy measurements.
- Convince NASA Ames Research Center 1960
- Kalman filter was used during Apollo program

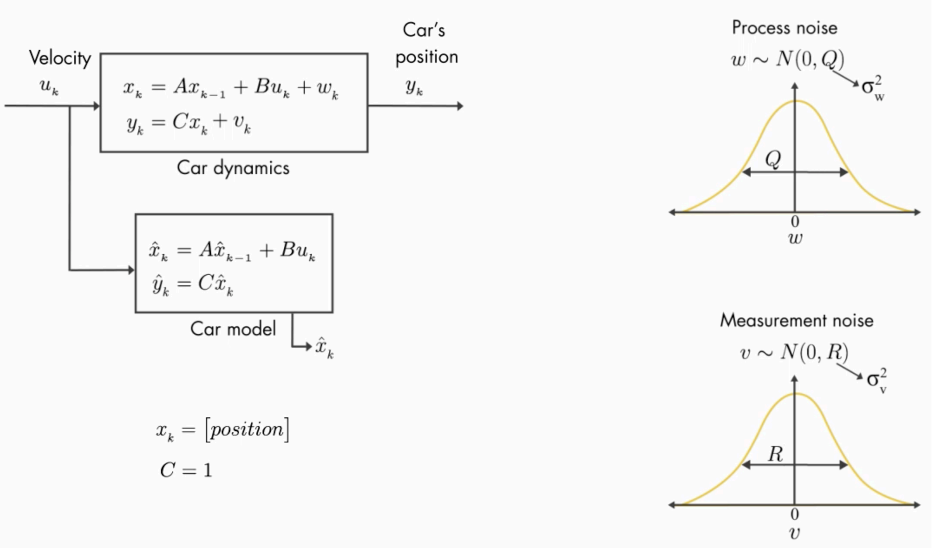
Problem Example 1

Self-Driving Car Location Problem



Problem Example 1

Self-Driving Car Location Problem



Problem Example 1

Self-Driving Car Location Problem

Prediction

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k$$

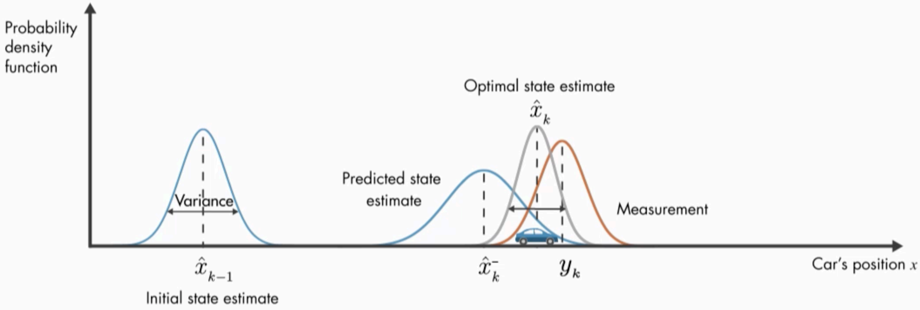
$$P_k^- = AP_{k-1}A^T + Q$$

Update

$$K_k = \frac{P_k^- C^T}{CP_k^- C^T + R}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(y_k - C\hat{x}_k^-)$$

$$P_k = (I - K_k C)P_k^-$$



Exercise: Analyse Kalman Gain

What is Kalman Gain $\boldsymbol{K}_k$, if measurement noise $\boldsymbol{R}$ is very small? What if $\boldsymbol{R}$ is very big?

Problem Example 2

Angle Measurement System

$$\boldsymbol{x}_t = \boldsymbol{A}_t \boldsymbol{x}_{t-1} + \boldsymbol{B}_t \boldsymbol{u}_t + \boldsymbol{w}_t$$

- $\boldsymbol{x}_t$: state in time t
- $\boldsymbol{A}_t$: state transition matrix from time $t-1$ to time t
- $\boldsymbol{u}_t$: input parameter vector at time t
- $\boldsymbol{B}_t$: control input matrix – apply the effort of $\boldsymbol{u}_t$
- $\boldsymbol{w}_t$: process noise, $\boldsymbol{w}_t \sim N(0, \boldsymbol{Q}_t)*$

$*\boldsymbol{w}_t$ assumes zero mean multivariate normal distribution, covariance matrix $\boldsymbol{Q}_t$

Problem Example 2 (Update on Oct. 29, 2018)

Angle Measurement System

$$\boldsymbol{x}_t = \boldsymbol{A}_t \boldsymbol{x}_{t-1} + \boldsymbol{B}_t \boldsymbol{u}_t + \boldsymbol{w}_t$$

- $\boldsymbol{x}_t = [x_t, \dot{x}_t]^\top$: x_t is current angle, while $\dot{x}_t$ is current rate
- $\boldsymbol{A}_t = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$
- $\boldsymbol{B}_t = [\frac{(\Delta t)^2}{2}, \Delta t]^\top$
- $\boldsymbol{u}_t = \Delta \dot{x}_t$

Problem Example 2

System Measurement

$$z_t = \boldsymbol{C}\boldsymbol{x}_t + \boldsymbol{v}_t$$

- z_t: measurement vector
- $\boldsymbol{C}$: transformation matrix mapping state vector to measurement
- $\boldsymbol{v}_t$: measurement noise, $\boldsymbol{v}_t \sim N(0, \boldsymbol{R}_t)$†

† $\boldsymbol{w}_t$ assumes zero mean multivariate normal distribution, covariance matrix $\boldsymbol{R}_t$

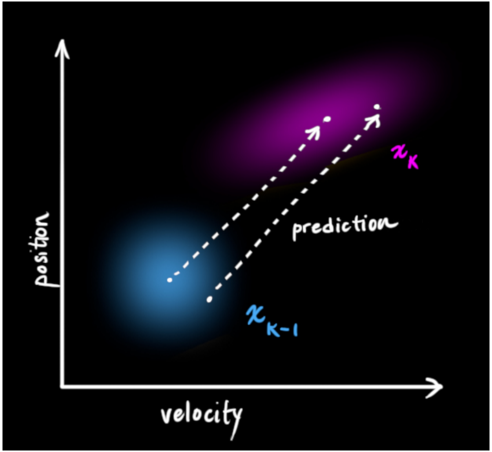
Exercise

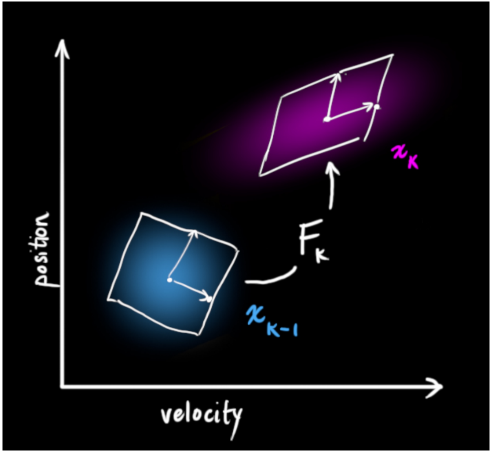
In angle measurement lab, what is the transformation matrix $\boldsymbol{C}$?

$$z_t = \boldsymbol{C}\boldsymbol{x}_t + \boldsymbol{v}_t$$

Model with Uncertainty

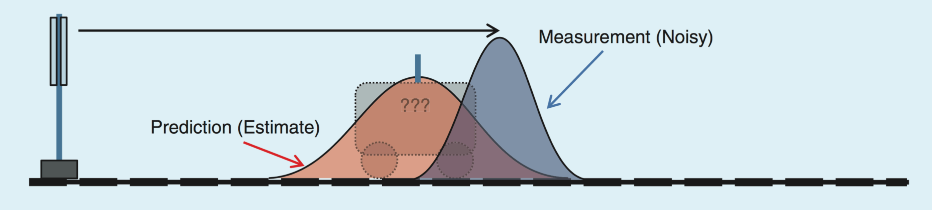
- Model the measurement w. uncertainty (due to noise $\boldsymbol{w}_t$)
- $\boldsymbol{P}_k$: covariance matrix of estimation $\boldsymbol{x}_t$
- On how much we trust our estimated value – the smaller the more we trust



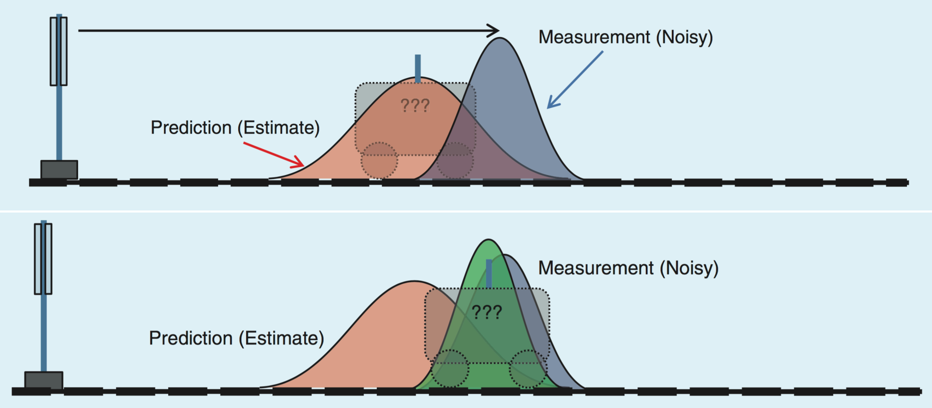


note: here $\boldsymbol{F}_k = \boldsymbol{A}_k$

Fuse Gaussian Distributions



Fuse Gaussian Distributions



Exercise

Given two Gaussian functions $y_1(r;\mu_1,\sigma_1)$ and $y_2(r;\mu_2,\sigma_2)$, prove the product of these two Gaussian functions are still Gaussian.

$$y_1(r;\mu_1,\sigma_1) = \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(r-\mu_1)^2}{2\sigma_1^2}} \qquad y_2(r;\mu_2,\sigma_2) = \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{(r-\mu_2)^2}{2\sigma_2^2}}$$

Step 1: Prediction

$$
\begin{aligned}
\boldsymbol{x}_t^- &= \boldsymbol{A}_t \boldsymbol{x}_{t-1} + \boldsymbol{B}_t \boldsymbol{u}_t &(1)\\
\boldsymbol{P}_t^- &= \boldsymbol{A}_t \boldsymbol{P}_{t-1} \boldsymbol{A}_t^\top + \boldsymbol{Q}_t &(2)
\end{aligned}
$$

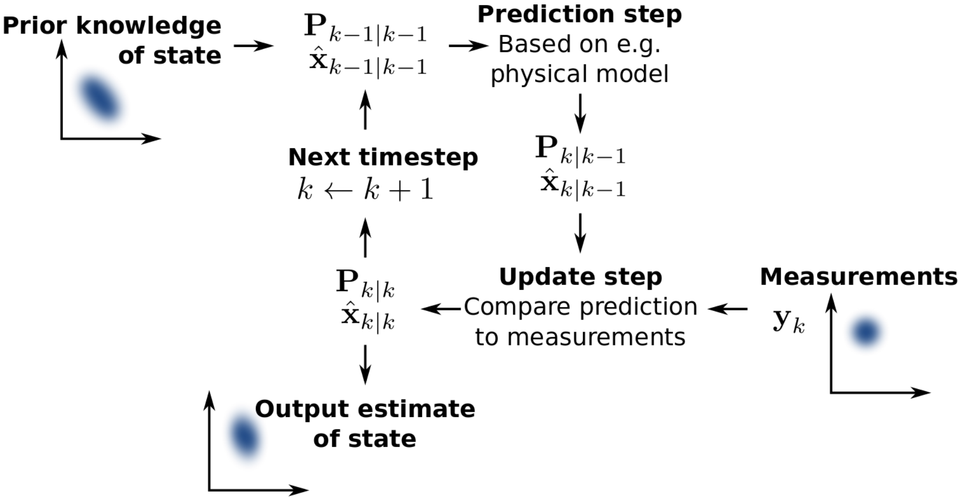
Step 1: Prediction

$$
\begin{aligned}
\boldsymbol{x}_t^- &= \boldsymbol{A}_t \boldsymbol{x}_{t-1} + \boldsymbol{B}_t \boldsymbol{u}_t && (1)\\
\boldsymbol{P}_t^- &= \boldsymbol{A}_t \boldsymbol{P}_{t-1} \boldsymbol{A}_t^\top + \boldsymbol{Q}_t && (2)
\end{aligned}
$$

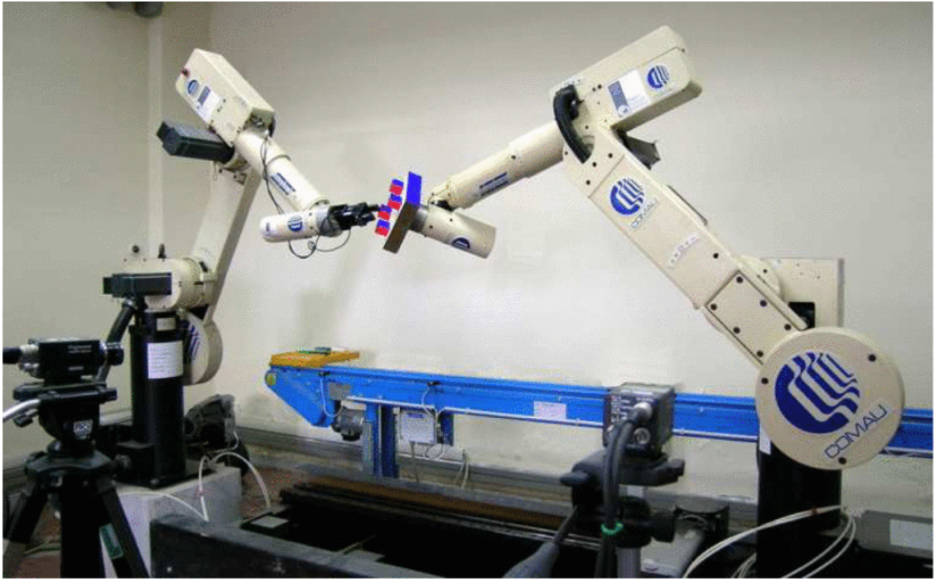
Step 2: Measurement Update

$$
\begin{aligned}
\boldsymbol{x}_t &= \boldsymbol{x}_t^- + \boldsymbol{K}_t(\boldsymbol{z}_t - \boldsymbol{C}\boldsymbol{x}_t^-) && (3)\\
\boldsymbol{P}_t &= \boldsymbol{P}_t^- - \boldsymbol{K}_t \boldsymbol{C} \boldsymbol{P}_t^- && (4)\\
\boldsymbol{K}_t &= \boldsymbol{P}_t^- \boldsymbol{C}^\top (\boldsymbol{C}\boldsymbol{P}_t^- \boldsymbol{C}^\top + \boldsymbol{R}_t)^{-1} && (5)
\end{aligned}
$$

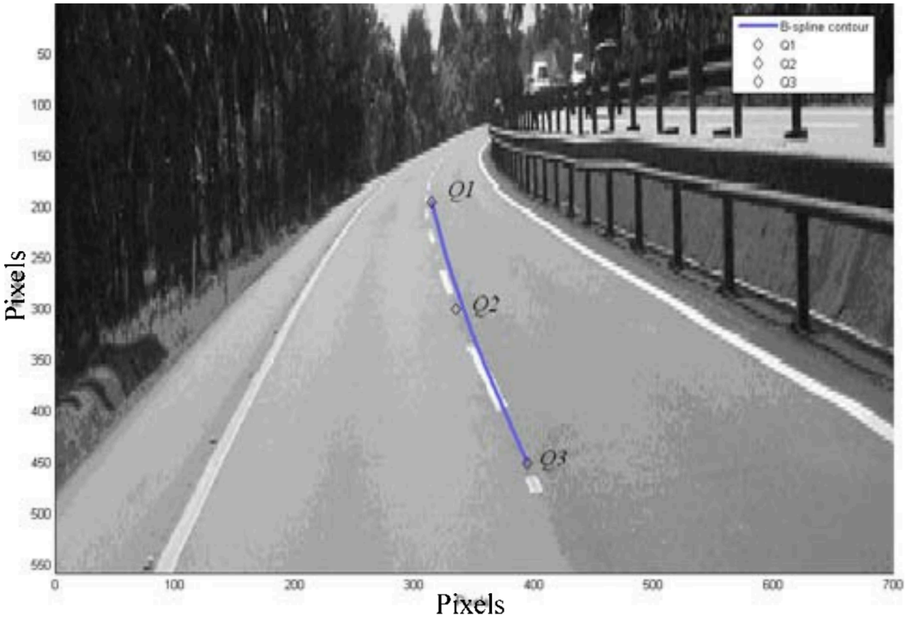
Basic Concepts



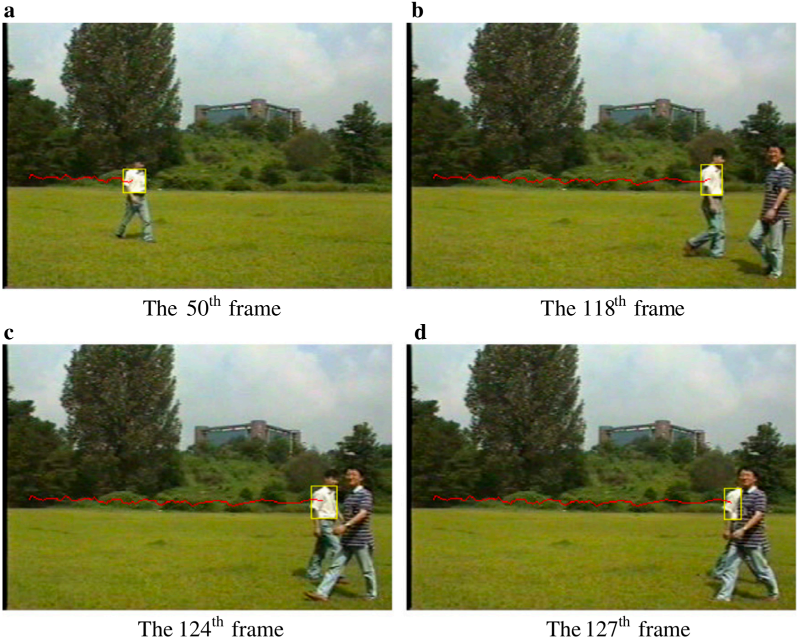
More Applications: Robot Localization



More Applications: Path Tracking



More Applications: Object Tracking



Overview

C Implementation

```c
// Kalman filter module
float Q_angle  =  0.001;
float Q_gyro   =  0.003;
float R_angle  =  0.03;

float x_angle = 0;
float x_bias = 0;
float P_00 = 0, P_01 = 0, P_10 = 0, P_11 = 0;
float dt, y, S;
float K_0, K_1;
```

- $\boldsymbol{Q}$:
- $\boldsymbol{R}$:
- $\boldsymbol{P}$:

C Implementation (cont.)

```
float kalmanCalculate(float newAngle, float newRate,int looptime)
{
    dt = float(looptime)/1000;
    x_angle += dt * (newRate - x_bias);
    P_00    += dt * (P_10 + P_01) + Q_angle * dt;
    P_01    += dt * P_11;
    P_10    += dt * P_11;
    P_11    += Q_gyro * dt;

    y   = newAngle - x_angle;
    S   = P_00 + R_angle;
    K_0 = P_00 / S;
    K_1 = P_10 / S;

    x_angle +=  K_0 * y;
    x_bias  +=  K_1 * y;
    P_00 -= K_0 * P_00;
    P_01 -= K_0 * P_01;
    P_10 -= K_1 * P_00;
    P_11 -= K_1 * P_01;

    return x_angle;
}
```

Summary

- Complementary Filter
- Kalman Filter