Constant velocity, Constant acceleration, Process noise

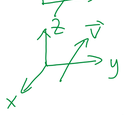
Wednesday, July 1, 2020 7:03 AM

- Definition: the motion model in which the target or object moves in a constant velocity

1D: $\vec{v}$

2D:

3D:



- state space representation:
  - Configuration space v.s. state space
    - ↳ d.o.f of the space : number of state you want to use to represent target motion.

1D → x, 1 Dof.
dim of config space = 1

1D config space

$\vec{x} = \begin{bmatrix} x \\ v \end{bmatrix}$ 2D state space, $\vec{x} = \begin{bmatrix} x \\ v \\ a \end{bmatrix}$ 3D state space.

  - Motion Model: flow of the state

$\dot{\vec{x}} = \vec{f}(\vec{x}, \vec{b})$ linear → $A\vec{x} + \vec{b}$

↳ vector field ↳ external vector.

  - 1D constant velocity model:

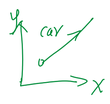
$\vec{x} = \begin{bmatrix} x \\ v \end{bmatrix}$, $\dot{\vec{x}} = \begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v \\ 0 \end{bmatrix}$, $\dot{\vec{x}} = \underbrace{\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}}_{=A} \vec{x}$ $\quad \dot{\vec{x}} = A\vec{x}$

  - 2D and 3D cases

2D: $\vec{x} = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix}$ $A_{2D} = \begin{bmatrix} A & 0_2 \\ 0_2 & A \end{bmatrix}$ 3D: $\vec{x} = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \\ z \\ \dot{z} \end{bmatrix}$ $A_{3D} = \begin{bmatrix} A & & \\ & A & \\ & & A \end{bmatrix}$

- Process noise: compensate uncertainties in estimation system.

e.g.



$\dot{\vec{x}} = A\vec{x} + B\xi$. $\quad \sigma_\xi$: standard deviation.

1D: $\vec{x} = \begin{bmatrix} x \\ v \end{bmatrix}$ $\quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}_{2\times 1}$, $\xi \sim N(0, \sigma_\xi^2)$

↳ scalar ↳ variance

- Continuous vs. discrete system

↳ theoretical ↳ practical, engineering

linear: $\dot{\vec{x}} = A\vec{x} + B\xi$ ⇢ $x_k = \Phi_{k,k-1} x_{k-1} + u_{k-1}$

↳ state transition matrix ↳ discrete process noise

  - Solution of linear system:

$$x(t) = \Phi(t, t_0)\, x(t_0) + \int_{t_0}^{t} \Phi(t, \tau)\, B(\tau)\, \xi(\tau)\, d\xi.$$

    - $\Phi(t, t_0)$

For linear time invariant system, $\Phi(t, t_0) = e^{A(t, t_0)}$

  - Process Noise:

$u_{k-1} \sim N(0, Q_{k-1})$, Q_{k-1}?

It can be proved:

$$Q_{k-1} = \int_{t_0}^{t} \Phi(t, \tau)\, B(\tau)\, \Sigma(\tau)\, B^T(\tau)\, \Phi^T(t, \tau)\, d\tau$$

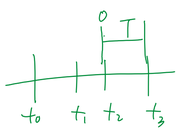
↳ $E[(u - \bar{u})(u - \bar{u})^T]$ ↳ mean

- Process noise for constant velocity model:

1D: $A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$ $\quad \Phi(t, 0) = e^{A(t-0)} = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix}$

Q.

- Assumption:
  - Initial time $t_0 = 0$
  - Step T



$$Q = \int_0^T \begin{bmatrix} 1 & T-\tau \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \sigma_\xi^2 \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ T-\tau & 1 \end{bmatrix} d\tau$$

$$= \begin{bmatrix} \frac{1}{3}T^3 & \frac{1}{2}T^2 \\ \frac{1}{2}T^2 & 0 \end{bmatrix} \sigma_\xi^2$$

$u_{k-1} \sim N(0, Q)$ sampling from it.

- 2D: $Q_{2D} = \begin{bmatrix} Q & 0_2 \\ 0_2 & Q \end{bmatrix}$ - 3D: $Q_{3D} = \begin{bmatrix} Q & & \\ & Q & \\ & & Q \end{bmatrix}$

Finished the C.V. model ∎

- Constant acceleration Model:

- 1D:

$\vec{x} = \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix}$ $\quad \dot{\vec{x}} = \begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dddot{x} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}}_{A} \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \xi$, $\xi \sim N(0, \sigma_\xi^2)$

$$\Phi(T, 0) = \begin{bmatrix} 1 & T & 0 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix}$$

$$Q = \begin{bmatrix} \frac{1}{20}T^5 & \frac{1}{8}T^4 & \frac{1}{6}T^3 \\ \frac{1}{8}T^4 & \frac{1}{3}T^3 & \frac{1}{2}T^2 \\ \frac{1}{6}T^3 & \frac{1}{2}T^2 & T \end{bmatrix} \sigma_\xi^2$$

- 2D

$\vec{x} = \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \\ y \\ \dot{y} \\ \ddot{y} \end{bmatrix}$ $\quad A_{2D} = \begin{bmatrix} A & 0_3 \\ 0_3 & A \end{bmatrix}$ $\quad \Phi_{2D} = \begin{bmatrix} \Phi & 0_3 \\ 0_3 & \Phi \end{bmatrix}$ $\quad Q_{2D} = \begin{bmatrix} Q & 0_3 \\ 0_3 & Q \end{bmatrix}$

- 3D

Get by yourself.

Next video: Matlab implementation!! ☺