

A3M33MKR

Kalmanův filtr

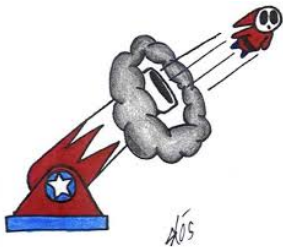
Ing. Karel Košnar PhD., RNDr. Miroslav Kulich, Ph.D.

Intelligent and Mobile Robotics Group
Decision Making and Control
Czech Technical University in Prague

29. října 2014

Zadání

- Budeme řešit známý fyzikální problém: šikmý vrh.
- Předpokládejme kanón, který vystřelí kouli pod neznámým úhlem a neznámou počáteční rychlostí.
- Máme kameru, která zaznamenává pozici koule ze strany. Měřené pozice jsou nicméně zatíženy nemalou chybou (s normálním rozložením).
- Cílem je odhadnout pozici koule jak nejpřesněji to půjde.



Matematický popis

Pozice koule $\mathbf{x} = (x, y)$ je určena vektorem počáteční rychlosti $\mathbf{v}_0 = (v_{0x}, v_{0y})$ a gravitačním zrychlením $g = 9.8$ (předpokládáme počáteční pozici $\mathbf{x} = (x_0, y_0)$)

$$x(t) = x_0 + v_{0x}t$$

$$y(t) = y_0 - v_{0y}t - \frac{1}{2}gt^2 \quad (1)$$

$$v_x(t) = v_{0x}$$

$$v_y(t) = v_{0y} - gt$$

Matematický popis

Pokud uvažujeme diskrétní případ, můžeme systém popsat

$$\begin{aligned}x_t &= x_{t-1} + v_{t-1x} \Delta t \\y_t &= y_{t-1} - v_{t-1y} \Delta t - \frac{1}{2} g \Delta t^2 \\v_{tx} &= v_{t-1x} \\v_{ty} &= v_{t-1y} - g \Delta t\end{aligned}\tag{2}$$

Popis stavu

$$x_t = \mathbf{A}x_{t-1} + \mathbf{B}u \quad (3)$$

u je řízení (v našem případě vliv gravitačního zrychlení)

Algoritmus Kalman_filter($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$)

Predikce (integrace akce)

$$\bar{\mu}_t = A\mu_{t-1} + Bu_t$$

$$\bar{\Sigma}_t = A_t\Sigma_{t-1}A^T + R$$

Korekce (integrace měření)

$$K_t = \bar{\Sigma}_t C^T (C\bar{\Sigma}_t C^T + Q)^{-1}$$

$$\mu_t = \bar{\mu}_t + K_t(z_t - C\bar{\mu}_t)$$

$$\Sigma_t = (E - K_t C)\bar{\Sigma}_t$$

return μ_t, Σ_t

A = Stavová matice přechodu.

B = Matice řízení.

C = Matice měření.

R = Šum procesu.

Q = Šum měření.

Kalmanův filtr

- Vstupy:
 - u = Řídicí vektor (v našem případě konstantní).
 - z_t = Vektor měření - pozice koule měřená v daném časovém okamžiku.
- Výstupy:
 - μ_t = Odhad aktuální pozice koule.
 - Σ_t = Nejistota v určení pozice.

Kalmanův filtr - jak ho počítat

Odhad aktuální pozice koule na základě řízení

$$\bar{\mu}_t = \mathbf{A}\mu_{t-1} + \mathbf{B}u$$

Odhad chyby

$$\bar{\Sigma}_t = \mathbf{A}\Sigma_{t-1}\mathbf{A}^T + \mathbf{R}$$

Porovnání reality s predikcí (inovace)

$$\tilde{y} = z_t - \mathbf{C}\bar{\mu}_t$$

Provnání chyby s predikcí

$$\mathbf{S} = \mathbf{C}\bar{\Sigma}_t\mathbf{C}^T + \mathbf{Q}$$

Kalmanův filtr - jak ho počítat

Kalmanovo zesílení

$$K = \bar{\Sigma}_t \mathbf{C}^T \mathbf{S}^{-1}$$

Aktualizace stavu

$$\mu_t = \bar{\mu}_t + \mathbf{K} \tilde{y}$$

a kovarianční matice

$$\Sigma_t = (\mathbf{E} - \mathbf{K}\mathbf{C})\bar{\Sigma}_t$$

- Zamyslete se na inicializací matic Σ , R , Q a vektoru μ .
- Jaké jsou rozměry matic a vektoru?

Vysvětlení kódu

Inicializace matic, velikost stavového vektoru je 4 a vektoru měření je 2.

```
Matrix4f A,B,\Sigma,R;  
Matrix2f Q;  
Matrix<float,??,??> C;  
Vector4f mu;  
float dt = 0.1;
```

Matice lze naplnit takto:

```
X << 1, 0, 0, 0,  
      0, 1, 0, 0,  
      0, 0, 1, 0,  
      0, 0, 0, 1;
```

Vysvětlení kódu - pokračování

Inicializace GUI a simulátoru

```
Gui gui;  
System system;
```

Point má složky x a y . Proměnná

measurement udržuje pozici koule změřenou kamerou

truth obsahuje skutečnou pozici koule (použitá pro kreslení)

kfPosition je pozice koule odhadnutá Kalmanovým filtrem

Vysvětlení kódu - pokračování

Hlavní smyčka

- simuluje jeden krok systému,
- počítá skutečnou pozici koule,
- generuje pozici měřenou kamerou a uchovává je v příslušných proměnných
- odhaduje pozici Kalmanovým filtrem
- a kreslí všechny pozice

```
for(int i=1; i<nSteps; i++) {  
    system.makeStep();  
    truth = system.getTruthPosition();  
    measurement = system.getMeasurement();  
    kfPosition = KalmanFilter(measurement);  
    gui.setPoints(truth, measurement, kfPosition);  
}  
gui.startInteractor();
```

Shrnutí

Vaším úkolem je implementovat Kalmanův filtr ve funkci *KalmanFilter*

```
Point KalmanFilter(const Point measuredPosition )  
{  
  
}
```

kteřá odhadne pozici koule a vrátí ji jako *Point*.