Úvod do mobilní robotiky — AIL028

Zbyněk Winkler a Martin Dlouhý

{zw|md} at robotika.cz

http://robotika.cz/guide/umor07/cs

8. listopadu 2007

Komunikace s hardwarem

I/O struktury

```
sync(status, command);
```

- synchronní, iniciovaná „shora“
- jednoduchá simulace
- snadné logování
- snadný přenost např. po RS-232
- kód může běžet na PC nebo na jednočipu

Status

```
int8_t timestamp;
int8_t enc_left;
int8_t enc_right;
```

Command

```
int8_t executeAt;
int8_t motor_left;
int8_t motor_right;
```

Jedeme rovně
if-metoda

```
while (true)
{
  sync(status,command);
  if (status.moc_vlevo())
    commnad.vic_vpravo();
  else if (status.moc_vpravo())
    command.vic_vlevo();
  else
    command.rovne();
}
```

Kombinace několika činností
FSM

```
while (true)
{
  sync(status, command);
  switch(state)
  {
    case INIT:           state = init(status); break;
    case GOING_TO_SUN:   state = goingToSun(status); break;
    case SUN_REACHED:    state = sunReached(status); break;
    case DOING_SUN:      state = doingSun(status); break;
    ...
  }
}
```

Refactoring, code reuse
implementace dovedností

- je celkem jedno, jestli robot jede rovně ke slunci, puku nebo brance
- každá „dovednost“ si potřebuje udržovat určité stavové informace
  - zbývající vzdálenost
  - kolik zbývá času
  - jaká byla historie odchylek od zvoleného směru
- nabízí se implementace jednotlivých dovedností jako objektů (vnořené FSM)
- zlepší se code reuse
- ale stále přetrvávají některé problémy
  - kdo zodpovídá za správnou inicializaci stavových proměnných?
  - kdo říká, kdy se přechází do jiného stavu?

Inside Out

Co to třeba otočit celé naruby?

```
state = INIT;
while (true)
{
   sync(status, command);
   switch(state)
   {
      case STRAIGHT:
         if (dist == 0) {
            state = TURN;
            angle = 90;
            command.turn();
            break;
         }
         dist--;
         break;
      case TURN: ...
         break;
   }
}
```

```
while(true)
{
   ahead(100);
   turnRight(90);
}


void ahead(dist)
{
   command.ahead();
   while(dist > 0)
   {
      sync(status, command);
      dist--;
   }
}
```

Rychlost otáčení kolečka

- K dispozici máme enkodér
- Jak zjistíme, jakou rychlostí se kolečko otáčí?

Vlastní řízení

Kvalita měření

Jaké máme rozlišení?

Filtrování

Průměr

- asi první věc, co každého napadne
- definice:

$$\overline{x} = \frac{1}{n} \sum_{i=0}^{n} x_i$$

Průměr

- asi první věc, co každého napadne
- definice:

$$\overline{x} = \frac{1}{n} \sum_{i=0}^{n} x_i$$

- co když nám data chodí postupně?
- co když si nemůžeme pamatovat úplně všechny hodnoty?
- co když vyžadujeme menší (konstantní) složitost?

Inkrementální výpočet průměru

- definice:

$$\begin{aligned} s_0 &= 0 \\ s_n &= s_{n-1} + x_n \\ \overline{x}_n &= s_n / n \end{aligned}$$

- ekvivalentně:

$$\begin{aligned} \overline{x}_n &= (\overline{x}_{n-1} \cdot (n-1) + x_n)/n \\ &\Downarrow \\ \overline{x}_n &= \overline{x}_{n-1} + \tfrac{1}{n}(x_n - \overline{x}_{n-1}) \end{aligned}$$

Inkrementální výpočet průměru

- definice:

$$\begin{aligned} s_0 &= 0 \\ s_n &= s_{n-1} + x_n \\ \overline{x}_n &= s_n / n \end{aligned}$$

- ekvivalentně:

$$\begin{aligned} \overline{x}_n &= (\overline{x}_{n-1} \cdot (n-1) + x_n)/n \\ &\Downarrow \\ \overline{x}_n &= \overline{x}_{n-1} + \tfrac{1}{n}(x_n - \overline{x}_{n-1}) \end{aligned}$$

- v jakých případech nám pomůže?
- co když se odhadovaná hodnota mění? (roste/klesá/osciluje)

Plovoucí průměr délky k

- definice:

$$\begin{aligned} x_i, s_0 &= 0 \qquad \text{pro } i < 0 \\ s_n &= s_{n-1} - x_{n-k} + x_n \\ \overline{x}_n &= s_n / k \end{aligned}$$

- ekvivalentně:

$$\begin{aligned} \overline{x}_n &= (\overline{x}_{n-1} \cdot k - x_{n-k} + x_n)/k \\ &\Downarrow \\ \overline{x}_n &= \overline{x}_{n-1} + \tfrac{1}{k}(x_n - x_{n-k}) \end{aligned}$$

Plovoucí průměr délky k

- definice:

$$\begin{aligned} x_i, s_0 &= 0 \qquad \text{pro } i < 0 \\ s_n &= s_{n-1} - x_{n-k} + x_n \\ \overline{x}_n &= s_n / k \end{aligned}$$

- ekvivalentně:

$$\begin{aligned} \overline{x}_n &= (\overline{x}_{n-1} \cdot k - x_{n-k} + x_n)/k \\ &\Downarrow \\ \overline{x}_n &= \overline{x}_{n-1} + \tfrac{1}{k}(x_n - x_{n-k}) \end{aligned}$$

- co když si nemůžeme/nechceme pamatovat k starých měření?

Odhad plovoucího průměru

Celkový průměr:
$\overline{x}_n = \overline{x}_{n-1} + \frac{1}{n}(x_n - \overline{x}_{n-1})$

Plovoucí průměr:
$\overline{x}_n = \overline{x}_{n-1} + \frac{1}{k}(x_n - x_{n-k})$

Nejlepší odhad x_{n-k}, co máme, je $\overline{x}_{n-1}$.

$$\overline{x}_n = \overline{x}_{n-1} + \frac{1}{k}(x_n - \overline{x}_{n-1})$$

Odhad plovoucího průměru

Celkový průměr:
$\overline{x}_n = \overline{x}_{n-1} + \frac{1}{n}(x_n - \overline{x}_{n-1})$

Plovoucí průměr:
$\overline{x}_n = \overline{x}_{n-1} + \frac{1}{k}(x_n - x_{n-k})$

Nejlepší odhad x_{n-k}, co máme, je $\overline{x}_{n-1}$.

$$\overline{x}_n = \overline{x}_{n-1} + \frac{1}{k}(x_n - \overline{x}_{n-1})$$

- jednoduché, rychlé, praktické

Odhad plovoucího průměru

Celkový průměr:
$\overline{x}_n = \overline{x}_{n-1} + \frac{1}{n}(x_n - \overline{x}_{n-1})$

Plovoucí průměr:
$\overline{x}_n = \overline{x}_{n-1} + \frac{1}{k}(x_n - x_{n-k})$

Nejlepší odhad x_{n-k}, co máme, je $\overline{x}_{n-1}$.

$$\overline{x}_n = \overline{x}_{n-1} + \frac{1}{k}(x_n - \overline{x}_{n-1})$$

- jednoduché, rychlé, praktické
- průměr se zpožďuje za aktuální hodnotou

Odhad plovoucího průměru

Celkový průměr:
$\overline{x}_n = \overline{x}_{n-1} + \frac{1}{n}(x_n - \overline{x}_{n-1})$

Plovoucí průměr:
$\overline{x}_n = \overline{x}_{n-1} + \frac{1}{k}(x_n - x_{n-k})$

Nejlepší odhad x_{n-k}, co máme, je $\overline{x}_{n-1}$.

$$\overline{x}_n = \overline{x}_{n-1} + \frac{1}{k}(x_n - \overline{x}_{n-1})$$

- jednoduché, rychlé, praktické
- průměr se zpožďuje za aktuální hodnotou
  - dá se s tím něco udělat?

Kalmanův filtr

- je to vlastně „vylepšený“ odhad plovoucího průměru
- vylepšení spočívá v rozdělení algoritmu na dva kroky
  - predikci nového stavu (a nového rozptylu/váhy)
  - korekci pomocí nového měření
- „průměr“ předpokládá, že odhadovaná veličina je konstanta

Algoritmus aktualizace

Predikce stavu a chyby — pomocí stavové rovnice

$$\begin{aligned} x_{k+1}^{-} &= Ax_k \\ P_{k+1}^{-} &= AP_kA^T + Q \end{aligned}$$

Korekce pomocí měření — pomocí rovnice měření

$$\begin{aligned} z_k &= Hx_k + v_k \\ K_k &= P_k^{-}H^T(HP_k^{-}H^T + R)^{-1} \\ x_k &= x_k^{-} + K_k(z_k - Hx_k^{-}) \\ P_k &= (I - K_kH)P_k^{-} \end{aligned}$$