ENSTA Bretagne

Suivi de waypoints par un robot buggy autonome

Fabrice LE BARS

Introduction

But

- Faire un robot buggy capable de suivre une trajectoire définie par des points GPS

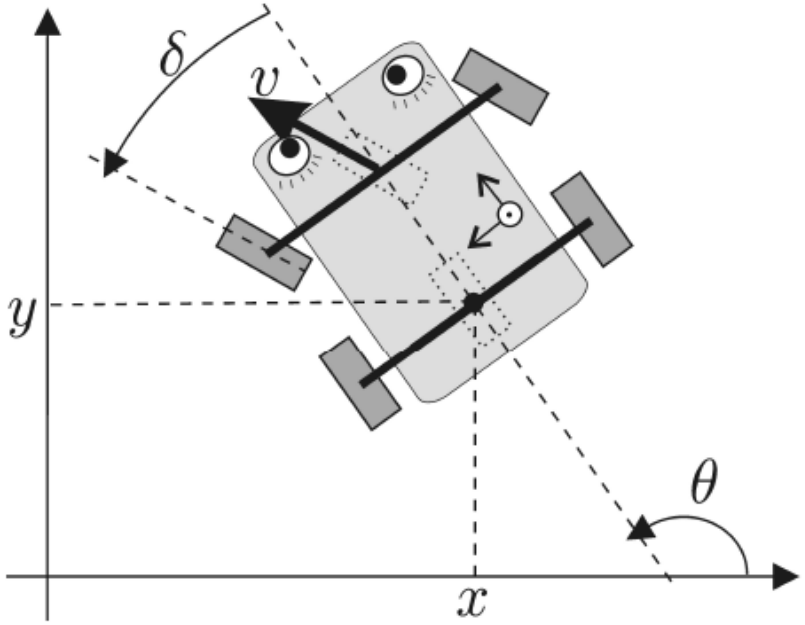
Modèle, observateurs et mesures

Schéma du système

Modèle d'état du buggy et équations géométriques

Buggy : modèle de type voiture

$$\begin{cases} \dot{x} &= v\cos\delta\cos\theta \\ \dot{y} &= v\cos\delta\sin\theta \\ \dot{\theta} &= \frac{v\sin\delta}{L} \end{cases}$$



$$\begin{aligned} v &= \alpha u_2 \\ \delta &= \beta u_1 \end{aligned}$$

L Distance entre les trains avant et arrière

Observateurs, mesures

Scénario : le buggy est dehors, capte le GPS précisément et a une boussole correcte

$$\begin{cases} y_1 = x \\ y_2 = y \\ y_3 = \theta \end{cases}$$

Remarques sur la boussole

- Sensible aux perturbations magnétiques dues aux objets métalliques de l'environnement proche (difficile à corriger mais on pourrait cartographier le champ magnétique)
- Sensible aux perturbations dues aux éléments constituant le robot (peut varier selon la vitesse des moteurs…). Les perturbations constantes peuvent cependant être facilement prises en compte

Remarques sur le GPS

- Ne fonctionne en général pas à l'intérieur (il faut qu'il ait une bonne « vue » des satellites dans le ciel)
- Il se peut qu'il donne des positions aberrantes lorsqu'il est à la limite de ne plus capter
- Temps de démarrage (« fix ») de plusieurs minutes variable selon les conditions

Régulation

Régulation

- Si on suppose que d'une manière ou d'une autre on a une estimation de x,y,theta, on peut maintenant réfléchir à la commande pour suivre un cap ou aller à une position particulière…

PID

- Commande proportionnelle à l'erreur, à son intégrale ou à sa dérivée
- Censée marcher assez bien dans beaucoup de cas
- Voir Wikipedia PID (page en Anglais) pour un exemple simple de pseudo-code de régulation par PID et de méthode pour trouver les coefficients (Ziegler–Nichols method…)

PID

```
previous_error = setpoint - actual_position
integral = 0
start:
  error = setpoint - actual_position
  integral = integral + (error*dt)
  derivative = (error - previous_error)/dt
  output = (Kp*error) + (Ki*integral) + (Kd*derivative)
  previous_error = error
  wait(dt)
  goto start
```

Régulation à une orientation voulue grâce à la boussole, à une vitesse arbitraire

- La boussole nous donne un angle au Nord en degrés θ
- Principe d'une régulation à un cap voulu θ_w :
  - Commande bang-bang : on fait tourner le robot à la vitesse de rotation maximale lorsqu'il est tourné dans le mauvais sens par rapport au cap voulu
  - Proportionnelle à l'erreur autrement :

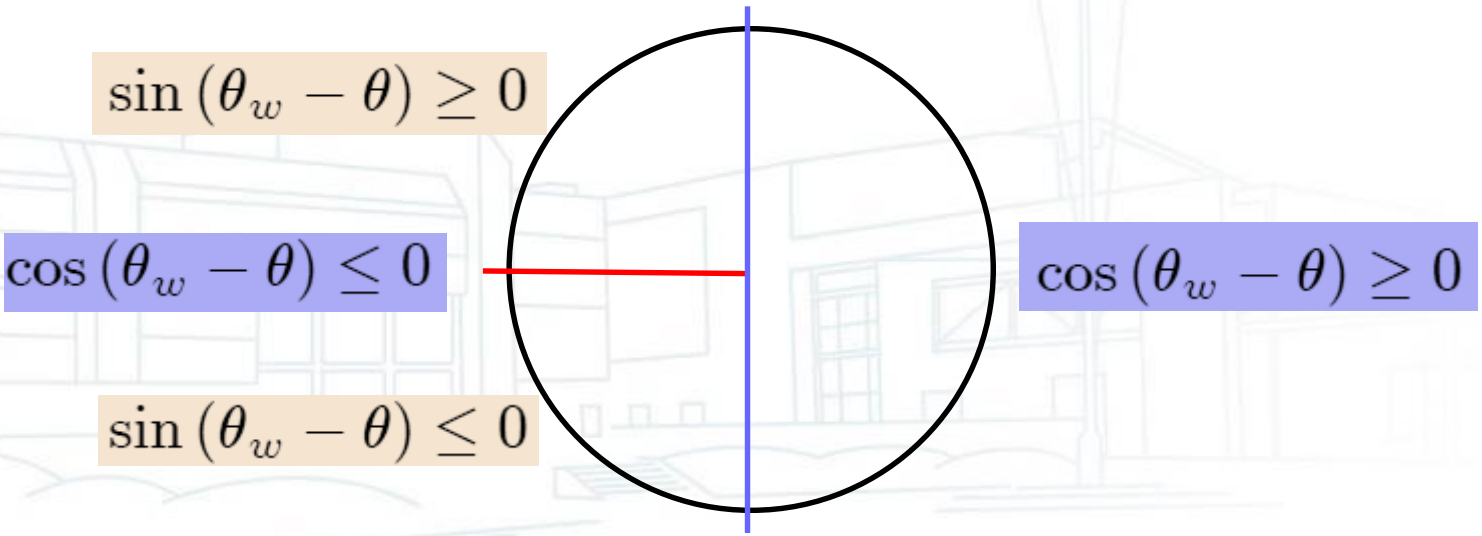
$$u_1 = K_p(\theta_w - \theta)$$
$$u_2 = u_w$$

- Attention aux problèmes de modulo 2π :
  - Utiliser des sin et cos par exemple
  - Voir aussi http://www.ensta-bretagne.fr/lebars/Share/fmod_360.zip)

Régulation à une orientation voulue grâce à la boussole, à une vitesse arbitraire

- Exemple : si l'erreur de cap est $(\theta_w - \theta)$, une commande possible sans problèmes de modulo 2π peut être :

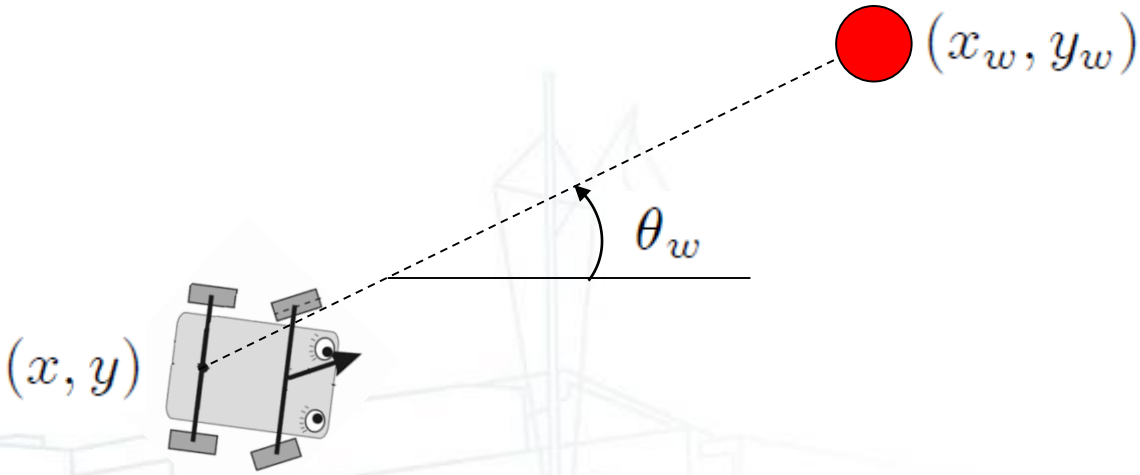
$$\delta = \begin{cases} \delta^{\max}.\sin(\theta_w - \theta) & \text{if } \cos(\theta_w - \theta) \geq 0 \\ \delta^{\max}.\text{sign}(\sin(\theta_w - \theta)) & \text{otherwise} \end{cases}$$



Suivi de waypoints GPS

- On peut prendre pour cap voulu :

$$\theta_w = \arctan_2 \left(y_w - y, x_w - x \right)$$



- atan2 est une fonction MATLAB comme arctan, mais qui retourne un angle dans $[-\pi, \pi]$ au lieu de $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$

Schéma du système pour le suivi de waypoints GPS