TP 5 - PROGRAMMATION ROBOTIQUE - INFO2

YANN CHEVALEYRE & PIERRE BOUDES

Aujourd'hui, nous aborderons le problème de navigation dans un cadre ou l'on ne connait si le plan, ni la position. L'une des solutions serait de faire de la **localisation**, ce que nous ferons au prochain TP. Dans ce TP, nous mettrons en place une navigation très basique, basée sur l'évitement d'obstacle et le controle de la direction.

1. Introduction

Dans ce TP, on commencera les programmes par `from robosimnxt import *`

Ce nouveau package definit des commandes de navigation fonctionnant à la fois sur NxT et sur le simulateur.

Il reprend aussi la plupart des commandes du package précédent `robosim`.

Les commandes principales de ce nouveau package sont:

```
const_unicycle_nxt(degrees) # fait avancer de 18cm le NxT
#et le fait tourner de degrees
const_unicycle_sim(degrees) # fait avancer de 18cm le robot simulé
# et le fait tourner de degrees
# renvoie aussi le nombre de degrees de rotation reels
const_unicycle(degrees) # commande les deux à la fois
```

- Lancez le simulateur, et essayez la commande `const_unicycle_sim` avec differents paramètres

2. Controle d'angle - PID

Dans le simulateur et le NxT, les mouvements sont bruités. Par exemple, `const_unicycle_sim` ne tourne pas le robot exactement du nombre de degrés passés en paramètre. En revanche, il renvoie une bonne estimation du nombre de degrés parcourus.

En classe, on vous expliquera le principe du controleur PID, et vous devrez implémenter ce controleur pour que le robot se dirige correctement vers le coin en bas à droite du tableau.

Vous testerez votre algo sur un tableau vide, chargé avec `init('robot_vide')`

3. Evitement d'obstacle

Dans cette partie, le robot utilisera son télémètre pour éviter les obstacles.

- A l'aide des fonctions `telemetre, oriente, orientation,` écrivez une fonction qui lance 5 rayons, relativement à la direction du robot au nord, est, ouest, nord-est, nord ouest. Il s'agit ici de simuler un robot équipé de 5 télémètres ou d'un télémètre rotatif. La fonction doit renvoyer une liste de distances

- Ecrivez une fonction qui prend la liste renvoyée par la fonction précédente et qui renvoie les coordonnées (x,y) (relativement au repère du robot) de chaque obstacle repéré par le télémètre. La fonction doit renvoyer une liste du type `[(23,-2),(55,38),...]`
- Vous avez maintenant deux listes: les coordonnées des obstacles et les distances.
- calculez la moyenne des 5 coordonnées. Le point obtenu est-il une bonne destination pour le robot ? expliquez
- Pour chacun de ces 5 points, on calculera des coefficients, décroissants en fonction de la distance. L'idée est qu'un obstacle est important (donc coefficient fort) s'il est proche. Proposez des façons de calculer ces coefficients. Ensuite, calculez la moyenne pondérée de ces points.
- Calculez l'angle de direction pour le robot, en fonction des coordonnées du point obtenu aux questions précédentes.

4. Evitement et controle

- Intégrez l'évitement et le déplacement (proposez des idées pour cela)
- Testez cela avec `init('robot_obstacle2')`