TP 2
Contrôle d'un robot mobile par logique floue

Ce TP utilise un simulateur de robot Khepera permettant de passer très simplement d'un mode simulation à un mode de contrôle du robot réel. Le but du TP est de mettre en oeuvre l'exemple de controleur flou vu en cours (suivi de mur à une distance donnée) et d'analyser son comportement (défauts, limites de fonctionnnement). La documentation du simulateur est incluse dans l'archive.

1) Prise en main du simulateur SIM2. Pour cela il faut récuperer le fichier SIM2.tar.gz sur le serveur louve.ucp, et l'installer dans votre répertoire d'acceuil. Un premier test consistera à compiler et exécuter l'exemple1. Attention, il faudra penser à adapter le makefile (pb de chemin d'acces à X11).

2) Ecrire sous forme de tableaux les règles floues dans la partie USER/user.c du programme. On supposera que l'on travaille avec un système capable de gérer 5 règles correspondant chacune à la conjonction de 4 variables linguistiques pour les conditions et une seule variable linguistique pour la partie déduction. Chaque variable linguistique sera définie par une fonction d'appartenance de forme trapézoidale (un seul trapèze pour simplifier).
   Travailler sur les capteurs de distance (IR)

3) Dans un premier temps, on suppose que le système flou n'a que 2 entrées : la distance à l'obstacle le plus proche à droite et la distance de l'obstacle le plus proche à gauche. Pour l'obtenir, on peut prendre le min des valeurs retournées par les capteurs IR du coté considéré (cette valeur est non linéaire par rapport à la distance mais on négligera ce problème). Pour les sorties, on suppose que l'on peut controler directement la direction du robot en relatif par rapport à son orientation courante. Une fonction sera utilisée pour convertir cet angle en une valeur de vitesse de rotation instantanée pour la roue droite et la roue gauche. Quelles sont les 3 règles adaptées à votre controleur ? Ecrire ces règles dans votre rapport (en langage courant en fonction des variables linguistiques que vous aurez définies).

4) Implanter et tester les règles proposées en simulation puis sur le robot réel.

5) Connaissant la complexité de l'environnement, à quelle vitesse faut il faire rouler le robot pour être sur d'éviter correctement les obstacles. Quel paramètre « caché » devez vous prendre en compte?

6) Rajouter des règles permettant de gerer la vitesse du robot

```
sudo apt-get install libx11-dev
```