TD 4 : Principes des Architectures des Systèmes Autonomes Intelligents

Miléna Kostov
Ekaterina Kostrykina

Question 2 :
Vr = 0
Vt = 0

Question 3 :
```
const double khepera3_matrix[9][2] = {
  {-5000, -5000},
  {-20000, 40000},
  {-30000, 50000},
  {-70000, 70000},
  {70000, -60000},
  {50000, -40000},
  {40000, -20000},
  {-5000, -5000},
  {-10000, -10000}
};
```
La colonne gauche correspond aux poids du moteur gauche, la colonne droite aux poids du moteur droit.
k correspond a const double khepera3_speed_unit = 0.00053429

Question 4 :
Lors de l'utilisation d'AlphaBot2 avec des capteurs binaires pour détecter les obstacles, l'adaptation de l'approche de Braintenberg nécessite une configuration simple du réseau neuronal : chaque capteur binaire devient une entrée, indiquant 1 pour obstacle, 0 pour rien. Un simple perceptron à une seule couche convient en raison de la nature binaire des capteurs. Il est possible d'initialiser les poids soit de manière aléatoire soit de manière arbitraire.On calcule la sortie du réseau en multipliant les lectures des capteurs par les poids correspondants, puis en additionnant ces produits. La sortie est utilisée afin de guider les actions du robot, comme tourner pour éviter les obstacles. Il est nécessaire d'ajuster les poids en fonction du comportement observé du robot pour minimiser les erreurs.

Question 5 :
Un algorithme d'évitement d'obstacle pour l'Alphabot2 sous forme d'un automate à états finis pourrait être décrit avec 4 états :

- Suivi de ligne : L'Alphabot2 suit la ligne en ajustant la vitesse des moteurs pour maintenir la ligne au centre des capteurs.
- Détection d'obstacle : Lorsqu'un ou plusieurs capteurs détectent un obstacle (valeur supérieure à un seuil prédéfini), le système passe à cet état.
- Évitement d'obstacle : L'Alphabot2 ajuste la trajectoire pour contourner l'obstacle. Cela peut impliquer une rotation ou un ajustement de la vitesse des moteurs.

- Recherche de la ligne : Après avoir évité l'obstacle, l'Alphabot2 entre dans cet état pour retrouver la ligne. Il peut effectuer une rotation jusqu'à ce qu'il détecte à nouveau la ligne.

Cet automate possède comme transitions possibles :

- De Suivi de ligne à Détection d'obstacle : Lorsqu'un capteur détecte un obstacle au cours du suivi de ligne.
- De Détection d'obstacle à Évitement d'obstacle : Lorsqu'un obstacle est détecté, le système passe à l'état d'évitement d'obstacle.
- De Évitement d'obstacle à Recherche de la ligne : Une fois que l'obstacle est contourné, l'Alphabot2 passe à l'état de recherche de la ligne.
- De Recherche de la ligne à Suivi de ligne : Lorsque la ligne est à nouveau détectée, le système revient à l'état de suivi de ligne.

Question 6 :

Pour le suivi des contours d'obstacles par la droite, il est possible de détecter si le robot s'approche d'un obstacle. Dans ce cas, la puissance du moteur gauche est réduite par rapport au moteur droit, permettant ainsi de contourner l'obstacle du côté droit. Le Khepera3 est équipé de 9 capteurs. Par conséquent, plusieurs capteurs à l'avant du robot sont utilisés pour détecter la présence d'obstacles de manière optimale. Il est également nécessaire de déterminer une valeur seuil. Si un ou plusieurs capteurs ont des valeurs inférieures à ce seuil, cela indique que le robot se rapproche trop de l'obstacle, et il doit donc contourner par la droite.

Les poids utilisés sont les suivants :

```
const double khepera3_matrix[9][2] =
{{-5000, -5000},
{-20000, 40000},
{-30000, 50000},
{-70000, 70000},
{70000, -60000},
{50000, -40000},
{40000, -20000},
{-5000, -5000},
{-10000, -10000}};
```

k correspond a const double khepera3_speed_unit = 0.00053429

La valeur de SEUIL_OBSTACLE vaut 75.

Lorsque le robot rencontre un obstacle, on change la vitesse des moteurs avec les équations suivantes afin de contourner par la droite :

```
if (sensors_value[5] > SEUIL_OBSTACLE || sensors_value[2] > SEUIL_OBSTACLE ||
    sensors_value[3] > SEUIL_OBSTACLE || sensors_value[4] > SEUIL_OBSTACLE ) {
    wb_motor_set_velocity(left_motor, max_speed * 0.5);
    wb_motor_set_velocity(right_motor, -max_speed * 0.01);    }
```

Question 7 :
Pour mettre en place un suivi de ligne avec les capteurs de l'Alphabot2, il est nécessaire que ces capteurs soient capables de détecter la présence ou l'absence de la ligne. Lorsque la ligne est détectée, les moteurs doivent être ajustés de manière à maintenir constamment la ligne au centre des deux capteurs. En revanche, si la ligne n'est pas détectée, le robot effectuera une rotation jusqu'à ce qu'il retrouve la ligne.

Question 8 :
L'architecture de subsumption, également connue sous le nom de Subsumption Architecture, est une approche de coordination basée sur des couches de comportements hiérarchiques. Chaque couche représente un comportement spécifique, et les couches supérieures ont un pouvoir de "subsumption" sur les couches inférieures. Si une couche supérieure devient active, elle inhibe les couches inférieures, permettant ainsi de prioriser les comportements.

```
# Comportement 1 : Suivi de ligne
def comportement_suivi_ligne():
    # Logique pour suivre la ligne

# Comportement 2 : Évitement d'obstacles
def comportement_evitement_obstacles():
    # Logique pour éviter les obstacles

# Coordination selon la Subsumption Architecture
while True:
    if ligne_detectee():
        comportement_suivi_ligne()
    elif obstacle_detecte():
        comportement_evitement_obstacles()
```

L'architecture DAMN est basée sur un mécanisme de votes où chaque comportement contribue à une décision collective. Chaque comportement attribue un vote, et la décision finale est prise en fonction du vote majoritaire.

```
# Comportement 1 : Suivi de ligne
def comportement_suivi_ligne():
    # Logique pour suivre la ligne
    vote_suivi_ligne = calculer_vote_suivi_ligne()

# Comportement 2 : Évitement d'obstacles
def comportement_evitement_obstacles():
    # Logique pour éviter les obstacles

    vote_evitement_obstacles = calculer_vote_evitement_obstacles()

# Coordination selon DAMN Architecture
while True:
    votes = [vote_suivi_ligne, vote_evitement_obstacles]
    action_finale = decide_action(votes)

    executer_action(action_finale)
```