**TD Architectures des Systèmes Autonomes Intelligents**

**Question 2**

Implémentez le déplacement de votre robot en ligne droite et son arrêt devant le premier obstacle rencontré.

Dans le dossier Question 2 vous trouverez le code + vidéo sur le simulateur

**Une image contenant capture d’écran, échecs, carré, Jeux

Description générée automatiquement**

**1 Initialisation :**

Le programme commence par initialiser le robot et ses composants, tels que les moteurs et les capteurs de distance.

Il définit également des constantes telles que le nombre de capteurs (NUM\_SENSORS), le seuil d'obstacle (OBSTACLE\_THRESHOLD), la fenêtre de moyennage (AVERAGE\_WINDOW), et la vitesse maximale des moteurs (max\_speed).

**2 Boucle principale :**

Le programme entre ensuite dans une boucle principale (while (wb\_robot\_step(time\_step) != -1)), qui s'exécute tant que la simulation n'est pas arrêtée (wb\_robot\_step(time\_step) renvoie -1 lorsque la simulation est terminée).

**3 Lecture du capteur avant :**

À chaque itération de la boucle, le capteur de distance avant (sensors[3]) est lu, et sa valeur est stockée dans la variable front\_sensor\_value.

**4 Mise à jour de la moyenne des valeurs du capteur :**

La valeur lue est utilisée pour mettre à jour une moyenne pondérée des valeurs du capteur (average\_sensor\_value). Cela est réalisé en utilisant une moyenne mobile pondérée, où les lectures antérieures ont un poids moins important que la lecture actuelle. Le compteur average\_count est également mis à jour pour maintenir une fenêtre glissante de lectures.

**5 Affichage de la moyenne :**

La moyenne des valeurs du capteur avant est affichée à chaque itération de la boucle.

**6 Vérification de l'obstacle :**

Le programme vérifie si la moyenne des valeurs du capteur avant dépasse le seuil d'obstacle (OBSTACLE\_THRESHOLD).

**7 Arrêt ou poursuite du mouvement :**

Si un obstacle est détecté (la moyenne dépasse le seuil), les moteurs du robot sont arrêtés en fixant la vitesse à zéro. Sinon, le robot continue de se déplacer vers l'avant en maintenant une vitesse constante (max\_speed).

**8 Fin de la boucle :**

La boucle continue de s'exécuter jusqu'à ce que la simulation soit arrêtée.

**Question 3**

Avec le simulateur Webots, en utilisant le robot khepera III, implémentez cet algorithme en utilisant tous les proximétres du robot. Choisissez les bons poids pour que le robot est un comportement satisfaisant

Une image contenant texte, Police, blanc, typographie

Description générée automatiquement

On peut calculer les vitesses des moteurs 𝑉*R* et 𝑉*L* en fonction des valeurs renvoyées par les capteurs gauches 𝑋 1 · · · 𝑋 𝑛 et droits 𝑋 1 · · · 𝑋𝑛 avec n= 9

On considère la matrice des poids

W =( *WL* \* X1, *WL* \* X2 ,. . . , *WL* \* X9

*WR*​\* X1, *WR*​ \* X2 , . . ., *WR*​ \* X9 )   
ou X = (X1, X2, . . ., X9) est les mesures de chaque capteur

Avec Xmax est la valeur maximum d'un capteur si y pas un obstacle à détecter.

Pour mettre en évidence l'influence des poids des capteurs sur les deux moteurs, il est possible d'employer une approche expérimentale en mettant en œuvre l'algorithme de Braintenberg avec notre robot Khepera III.

**For i =1 ,9 ,1 do**

V *L* =V *L* +W *L* ∗ X[ i ]∗(1 −( X [ i ] / Xmax ) )

V =V +W ∗ X[ i ]∗(1 −( X [ i ] / Xmax ) )

**End**

La matrice des poids prédéfini :

khepera3\_matrix[9][2] = {{-5000, -5000}, {-20000, 40000}, {-30000, 50000}, {-70000, 70000}, {70000, -60000}, {50000, -40000}, {40000, -20000}, {-5000, -5000}, {-1000-10000}};

On suppose que les mesures du trois capteurs (X3, X5, X7) détecte l'obstacle, alors la matrice W va être :

W = (-70000 , -60000 , -30000 , -70000 , 70000 , -60000, 40000 , -70000 , -5000

-70000 , -60000 , -5000 , -70000 , 70000 , -60000 , 70000 , -70000 , -5000 )

On peut calculer et comme suit :

= \* X1 \* (1 - (X1/Xmax)) + \* X2 \* (1 -(X2/Xmax)) + .. + \*X9 \* (1 - (X9/Xmax))

= W \*X1 \* (1 - (X1/Xmax)) + W \* X2 \* (1 - (X2/Xmax)) + .. + W \*X9 \* (1 - (X9/Xmax))

On obtient

= -30000 \* (1 - (200/512)) + 70000 \* (1 - (500/512)) + 40000 \* (1 - (800/512)) + ...

= -30000 \* (1 - 0.3906) + 70000 \* (1 - 0.9766) + 40000 \* (1 - 1.5625) + ...

= -30000 \* 0.6094 + 70000 \* 0.0234 + 40000 \* -0.5625 + ...

= -18281.25 + 1640.63 - 22500 + ...

= -429.62

= -70000 \* (1 - (200/512)) - 60000 \* (1 - (500/512)) + 70000 \* (1 - (800/512)) + ...

= -70000 \* (1 - 0.3906) - 60000 \* (1 - 0.9766) + 70000 \* (1 - 1.5625) + ...

= -70000 \* 0.6094 - 60000 \* 0.0234 + 70000 \* -0.5625 + ...

= -42656.25 - 1406.25 - 39375 + ...

**Question 4**

Comment implémenteriez-vous un réseau de neurones pour l’évitement d’obstacles selon braintenberg dans le cas de l’alphabot2 ?

Pour mettre en œuvre l'évitement d'obstacles à l'aide d'un réseau neuronal selon l'algorithme de Braitenberg dans le contexte de l'Alphabot2, il faut remplacer l'algorithme de Braitenberg existant dans le code par une approche basée sur un réseau neuronal. Le réseau neuronal doit prendre les relevés des capteurs en entrée et produire des commandes de moteur en sortie.

Dans le dossier Question 4 vous trouverez le code

**Question 5**

Si vous deviez fournir un algorithme d’évitement d’obstacle pour l’Alphabot2, sous forme d’un automate à état fini, quel serait-il ? Implémentez et testez.

Un automate à états finis (AEF) pour l'évitement d'obstacles peut être conçu comme suit. Cet automate aura deux états principaux : un état d'évitement et un état de déplacement droit. La transition entre ces états sera basée sur la détection d'obstacles par les capteurs du robot.

**État d'évitement (Avoidance):**

Si un obstacle est détecté sur le côté gauche, tournez à droite.

Si un obstacle est détecté sur le côté droit, tournez à gauche.

Si des obstacles sont détectés des deux côtés, reculez et tournez à droite.

**État de déplacement droit (Straight):**

Si aucun obstacle n'est détecté, continuez à avancer tout droit.

Si un obstacle est détecté sur le côté gauche ou droit, passez à l'état d'évitement.

Dans le dossier Question 5 vous trouverez le code

**Question 6**

Quel algorithme mettriez-vous en place pour un suivi de contours d’obstacles par la droite sur le Khepera III ? Implémentez et testez.

Une image contenant texte, Police, blanc, typographie

Description générée automatiquement

Le suivi du contour d'obstacle en ajustant la vitesse des moteurs en fonction de la distance mesurée par le capteur. L'objectif est de maintenir une distance constante de TARGET\_DISTANCE par rapport à l'obstacle.

Dans le dossier Question 6 vous trouverez le code

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

**Question 7**

Si vous voulez implémenter un suivi de ligne grâce aux capteurs du robot Alphabot2, que proposeriez-vous comme algorithme ? Implémentez et testez.

Pour mettre en place un suivi de ligne avec les capteurs du robot Alphabot2, on va utiliser un algorithme relativement basé sur la détection des transitions entre la ligne et la surface adjacente.

Dans le dossier Question 7 vous trouverez le code

Dans le code :

Nous utilisons des capteurs de lumière (wb\_light\_sensor) pour détecter la ligne.

La variable error mesure la déviation du robot par rapport au centre de la ligne.

Les vitesses des moteurs sont ajustées en fonction de cette erreur.

Le terme proportionnel (Kp) contrôle la sensibilité du suivi de ligne.

**Question 8**

Mettez en place deux stratégies de coordination différentes et testez les différences

Nous allons adapter les 2 stratégies dans le contexte d'évitement d'obstacles

Dans le dossier Question 8 vous trouverez les codes et les vidéos

**Stratégie 1 : Somme pondérée des comportements**

Dans cette stratégie, chaque comportement génère une contribution à la commande motrice, pondérée par un facteur de priorité. La commande finale est obtenue en sommant ces contributions pondérées.

**Stratégie 2 : Architecture de Subsomption**

Dans cette stratégie, chaque comportement est associé à une couche, et ces couches sont empilées par priorité. Chaque couche inhibe les couches inférieures si elle produit une commande.