Université de Bretagne Occidentale

Faculté des Sciences & Techniques

Projet de méthode computationnelle

Système d'inférence flou
Approximation d'une fonction de contrôle

ROBOT

Présenté par:

Boulmaali Linda Imene

Miszczuk Ivan

Introduction

L'objectif du TP est de se familiariser avec la commande floue, en mettant l'accent sur la construction de règles décrivant le comportement réactif d'un robot évoluant dans un environnement inconnu qu'il perçoit localement

Problème

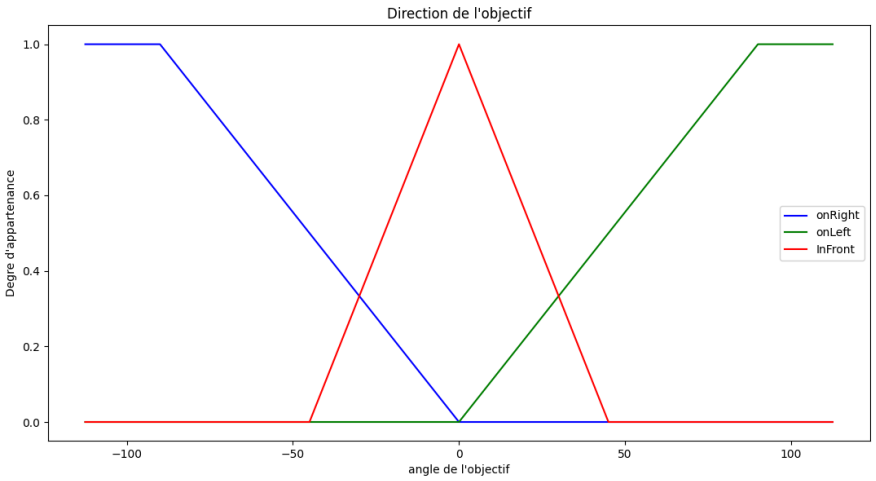
Le problème qui nous est posé consiste à résoudre le problème du pilotage d'un robot mobile dans un environnement inconnu. L'objectif du robot est de rejoindre un ensemble de cibles placées aléatoirement tout en évitant les obstacles présents sur son chemin.

Conception

Variables Linguistiques et Leurs Valeurs

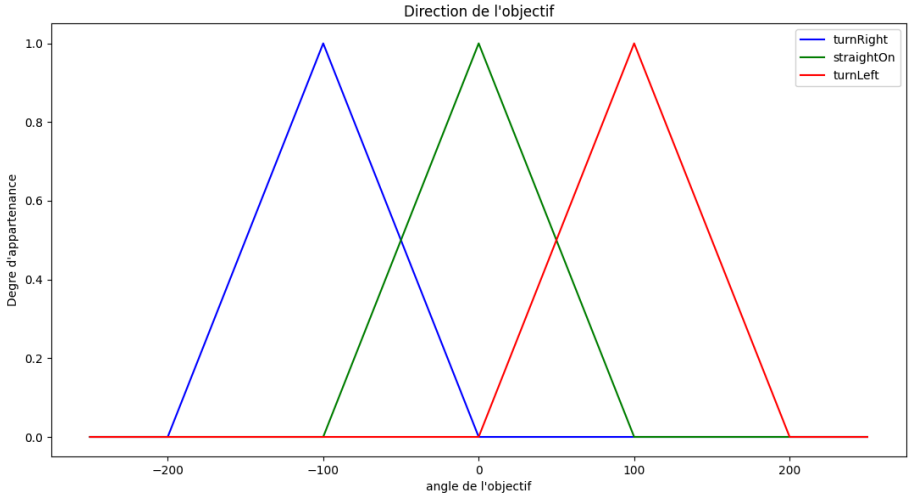
Pour la direction vers l'objectif (DirecGoal) :

- **onRight** : L'objectif est sur la droite.
- **inFront** : L'objectif est droit devant.
- **onLeft** : L'objectif est sur la gauche.



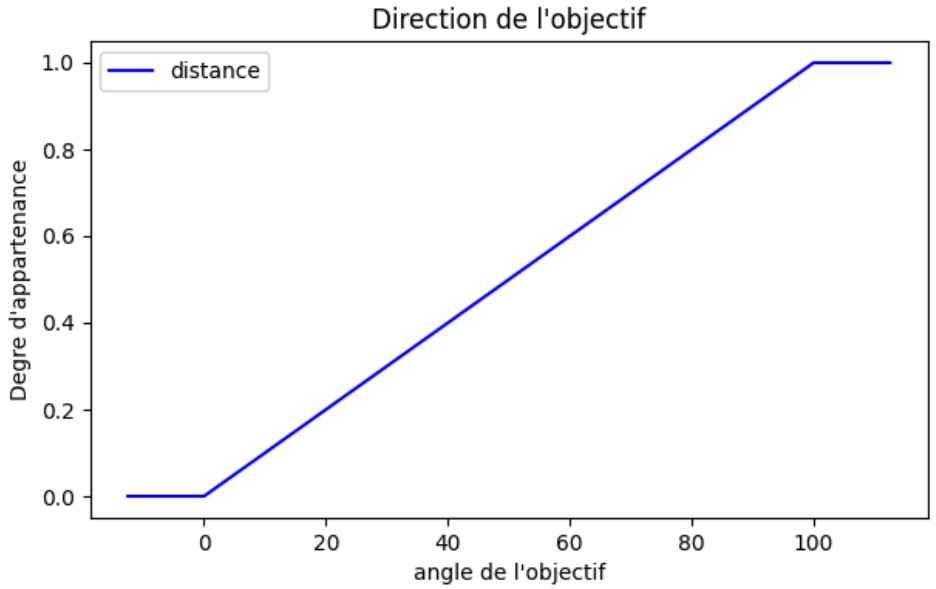
Pour la gestion de la direction (Sang) :

- **turnRight** : Pivoter à droite pour changer de direction.
- **straightOn** : Continuer tout droit, sans changer de direction.
- **turnLeft** : Pivoter à gauche pour changer de direction.



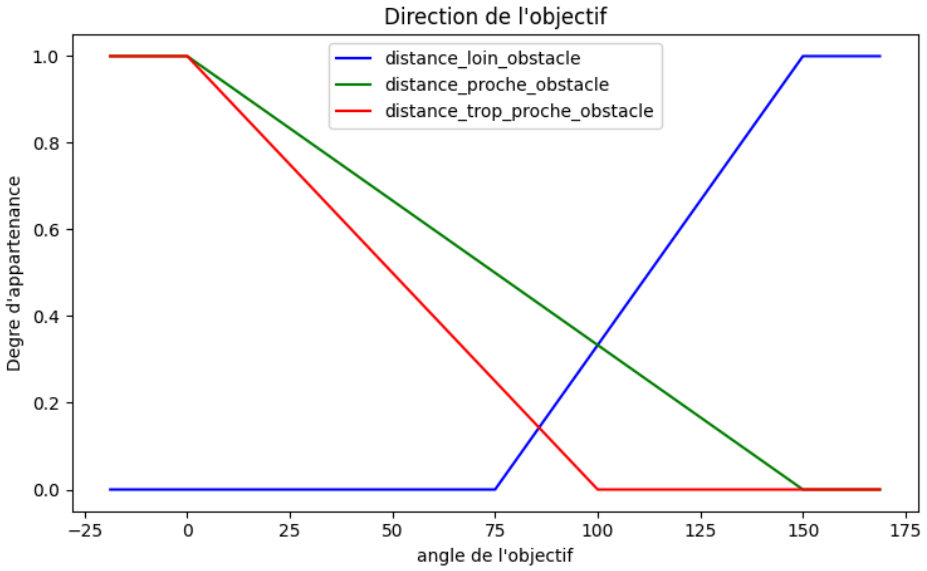
Pour l'évaluation de la distance avec l'objectif :

- **distance** : Détermine la présence et l'ampleur de la distance séparant l'objectif du robot.



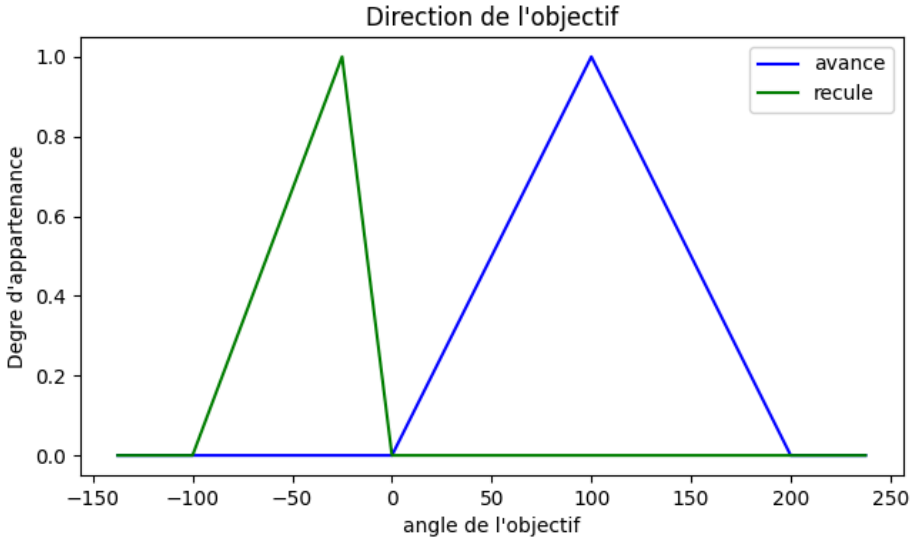
Pour la perception des obstacles :

- **distance_loin_obstacle** : Indique une grande distance entre l'obstacle et le robot.
- **distance_proche_obstacle** : Signifie que l'obstacle est relativement proche du robot.
- **distance_trop_proche_obstacle** : Signale une très faible distance avec l'obstacle, nécessitant une action immédiate.



Pour le contrôle de la vitesse (Slin) :

- **avance** : Le robot se déplace vers l'avant.
- **recule** : Le robot se déplace vers l'arrière.



Calibration des Paramètres Flous

Les paramètres des fonctions d'appartenance, telles que les triangles, ont été finement ajustés à travers de multiples itérations de test pour s'aligner au plus près avec le comportement désiré du robot.

Règles de Déduction Floue

Pour la gestion de la direction (Sang) :

1. Le robot pivote à gauche si l'objectif est à sa gauche et qu'aucun obstacle n'entrave cette direction.
2. Le robot pivote à droite si l'objectif est à sa droite et qu'aucun obstacle n'entrave cette direction.
3. En présence d'un obstacle directement devant, le robot choisit de contourner par la gauche.
4. Le robot continue en direction de l'objectif si aucune proximité d'obstacle n'est détectée.
5. Si un obstacle est détecté proche sur la droite, le robot privilégie la progression tout droit.

Pour la gestion de la vitesse (Slin) :

1. Le robot avance tant qu'une distance sécuritaire le sépare de tout obstacle en face et qu'il se dirige vers son objectif.
2. En cas de détection d'un obstacle trop proche, le robot active une marche arrière pour éviter la collision.

Méthodes adopté,

Nous avons conservé la configuration par défaut et construit nos règles en nous basant sur ces méthodes :

et := min; ou := max; decodage := centre; implication := larsen; ainsi_que := max;

Exemple d'application :

Si ObstFront est distance_trop_proche_obstacle alors Slin est recule;

Nous prenons ObstFront comme x, distance_trop_proche_obstacle comme A, Slin comme y et recule comme B, puis nous appliquons les mêmes règles.

Nous obtenons:

Si x est A alors y est B;

pour A implique B : on utilise larsen qui représente le produit, on aura donc le degré d'appartenance de A multiplier par le degré d'appartenance de B.
Pour cette condition nous obtenons la formule suivante:

$$u_{(A \text{ implique } B)}(y) = u_A(x) * u_B(y)$$

Conclusion

Nous avons mis en œuvre une solution pour le problème de pilotage d'un robot, et dans l'ensemble, le robot parvient généralement à éviter les obstacles et à atteindre les objectifs sur les cartes 0, 1 et 2. Cependant, nous avons rencontré quelques difficultés que nous résumons comme suit :

- Le traitement des cas particuliers affecte le comportement de base du robot. De nouvelles règles peuvent annuler d'autres règles potentiellement plus intéressantes, notamment lorsque nous ne sommes pas dans un cas particulier donné.
- Lorsque le robot se trouve près d'un obstacle et que l'objectif est situé à droite, il tente de tourner à gauche pour éviter l'obstacle et à droite pour atteindre l'objectif, ce qui le bloque dans une boucle entre les actions turnLeft et turnRight.
- Lorsque le robot tourne brusquement vers un obstacle, il ne dispose pas du temps nécessaire pour ralentir, ce qui l'empêche de l'éviter.

En conclusion, ce projet nous a permis de comprendre l'importance de la logique floue dans la résolution de certains problèmes. La sélection minutieuse des règles floues et des paramètres reflète un équilibre entre précision et flexibilité, essentiel pour le succès de l'application robotique.

En tant que perspectives futures pour ce projet, nous envisageons :

- Trouver des règles pour traiter les cas particuliers sans pour autant compromettre les règles prioritaires.
- Gérer efficacement le comportement du robot sur la carte 3.