APPRENTISSAGE SUPERVISÉ KNN CAMERA

PREREQUIS

CONNAISSANCES

Statistiques de bases et savoir ce qu'est un « apprentissage supervisé »

Connaître la distance euclidienne $d = \sqrt{\sum_i (x_i - y_i)^2}$

LOGICIEL

Avoir le logiciel AlphAi installé sur un ordinateur.

MATERIEL

Un ordinateur avec une carte Wi-Fi pour communiquer avec le robot

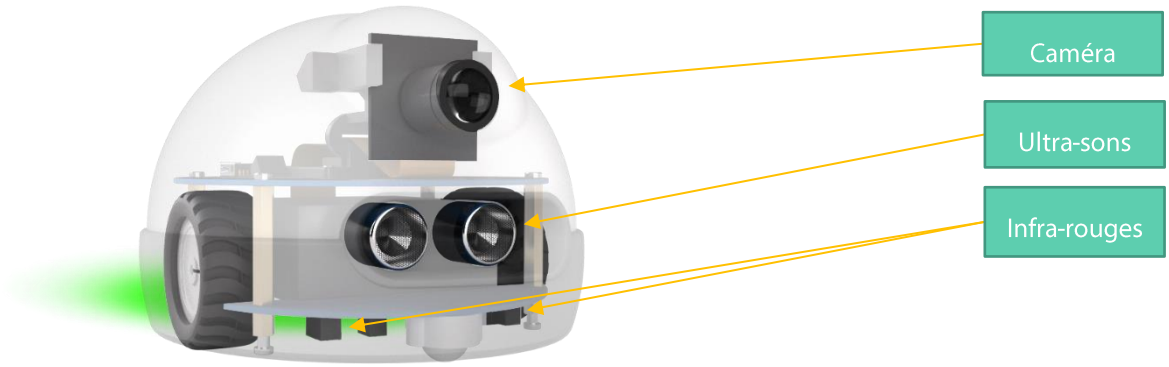
Avoir le robot AlphAi avec une caméra embarquée fonctionnelle.

PRESENTATION DU ROBOT ET LOGICIEL ALPHAI

LE ROBOT ALPHAI

Le robot vient équipé de plusieurs capteurs et actuateurs. Il possède :

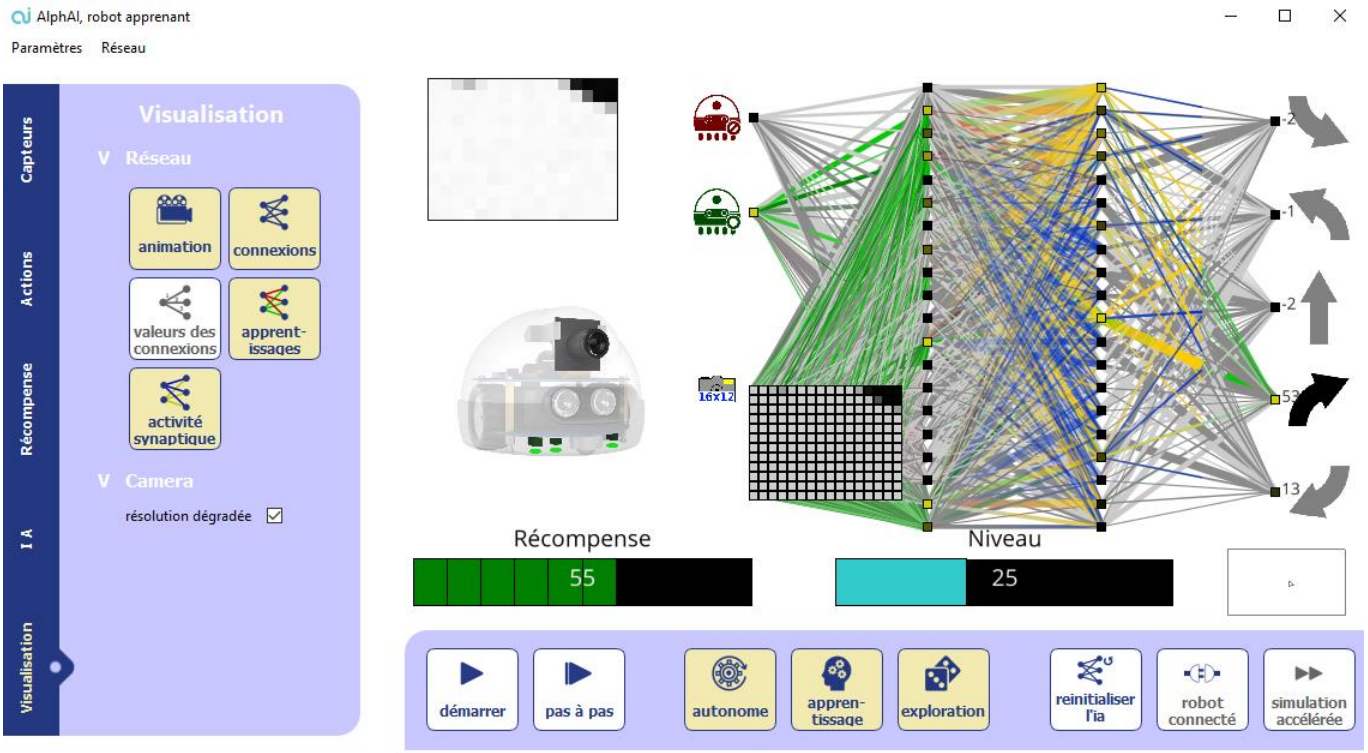
- un module à ultra-son pour mesurer une distance frontale
- cinq capteurs infrarouges orientés vers le bas qui permettent de détecter un mouvement ou un blocage comme le fait une souris d'ordinateur.
- une caméra pour les traitements de flux vidéos
- une paire de moteurs et de roues pour se déplacer.



Il est contrôlé par un micro-ordinateur, un Raspberry Pi, qui communique avec un PC en Wi-Fi. Les décisions sont prises par le logiciel AlphAi et envoie les ordres au robot, tels que « Tourne à droite » ou « quel est l'état du capteur à ultra-sons ». Le robot ne fait que les exécuter, la véritable intelligence se trouvant à distance.

LE LOGICIEL ALPHAI

Pour contrôler le robot, le logiciel AlphAi est utilisé.



Sur le bandeau de gauche se situent toutes les configurations nécessaires au robot et à son apprentissage. L'onglet *Capteurs* permet de choisir quels capteurs utiliser pour l'apprentissage, l'onglet *Actions* permet de choisir quelles seront les différentes actions réalisables par le robot après calcul par l'algorithme ; enfin l'onglet *Récompense* permet d'ajuster les récompenses et pénalités que le robot reçoit lors de son apprentissage. Dans la partie basse, l'onglet *IA* regroupe les différents algorithmes d'apprentissage disponible et leur paramètres ; et l'onglet *Visualisation* ajuste les différents paramètres visuels de l'interface graphique pour par exemple afficher la caméra en résolution dégradée – ou non.

INTRODUCTION

Nous allons étudier de près un algorithme très connu parmi les algorithmes de *machine learning* : l'algorithme des K plus proches voisins, ou KNN en anglais pour « *K nearest neighbours* ». Cet algorithme a l'avantage d'être très simple, et donc de fournir une bonne base pour comprendre l'apprentissage supervisé.

L'idée est d'utiliser un grand nombre de données afin "d'apprendre à la machine" à résoudre un certain type de problème. L'algorithme est particulièrement intuitif quand les données se présentent comme des points dans un espace de dimension 2, qui pourront donc être affichés à l'écran. Nous allons donc choisir une configuration avec seulement 2 entrées capteurs, à savoir 2 « super-pixels » de la caméra afin de permettre au robot de repérer les murs et de circuler dans l'arène sans blocage. Dans un premier temps ce sera à nous d'indiquer au robot l'action associée pour chaque nouvelle donnée (2 « super-pixels »), mais ensuite il sera capable de circuler par lui-même en choisissant l'action majoritaire prise par les K plus proches voisins parmi les données entrées pendant la phase d'entraînement.

L'objectif de ce TP est de se familiariser avec l'algorithme KNN et de trouver une valeur *k* qui minimise les erreurs dans la prise de décision pour optimiser la navigation du robot. On part du principe de base que l'objectif du robot est de parcourir la plus grande distance sans percuter un mur ou être arrêté par un obstacle.

L'ALGORITHME DES K PLUS PROCHES VOISINS, KNN

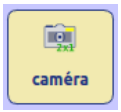
Cet algorithme d'apprentissage supervisé permet de classifier des données en fonction de leur proximité avec celles existantes constituant les données d'entraînement. Les nouvelles données considèrent *k* voisins, les *k* plus proches, pour choisir quelle classification appliquer. Avec des données discrètes, comme des points dans un plan ou dans l'espace, les plus proches voisins sont sélectionnés en utilisant la distance euclidienne, mais d'autres méthodes peuvent être utilisées dans d'autres situations, comme la distance Manhattan si les données se placent sur un quadrillage ou un échiquier. L'intérêt de cet algorithme réside dans le choix de *k* : trop petit et l'algorithme devient trop sensible aux erreurs ou aux données manquantes, trop grand et il considère des données qui ne devraient pas être considérées.

PHASE DE PREPARATION DE L'ENVIRONNEMENT DE TRAVAIL

- → Allumez le robot (l'interrupteur se trouve en dessous). Il effectue un petit mouvement lorsqu'il est prêt. Connectez-vous en wifi du robot (chercher le wifi qui commence par ALPHAI : le mot de passe est identique au nom du wifi).
- → Lancez le logiciel en cliquant sur l'icône **« AlphAI »** sur le Bureau ou depuis le menu démarrer.
- → Chargez la configuration « apprentissage supervisé – KNN caméra ».
- → Ou pour sélectionner vous-mêmes les paramètres, lancez le programme puis :

DANS L'ONGLET CAPTEURS :

1) Sélectionnez « camera »



2) Sélectionnez « pré-calcul »

3) Choisissez :

« **vert-rouge** » si l'arène est rouge

« **niveaux de gris** » si l'arène est d'autre couleur.

DANS L'ONGLET ACTION :

1) Sélectionnez

2) Désactivez

3) Cochez aussi l'option « **pause entre les actions** » pour un apprentissage plus fiable

DANS L'ONGLET RECOMPENSE

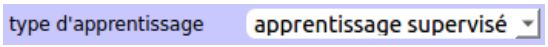
Choisissez « vitesse et blocage »

DANS L'ONGLET VISUALISATION (NON OBLIGATOIRE)

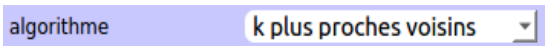
Décochez « **image dégradée** » pour voir que ces 2 pixels sont les valeurs moyennes dans 2 régions d'intérêt qui apparaissent

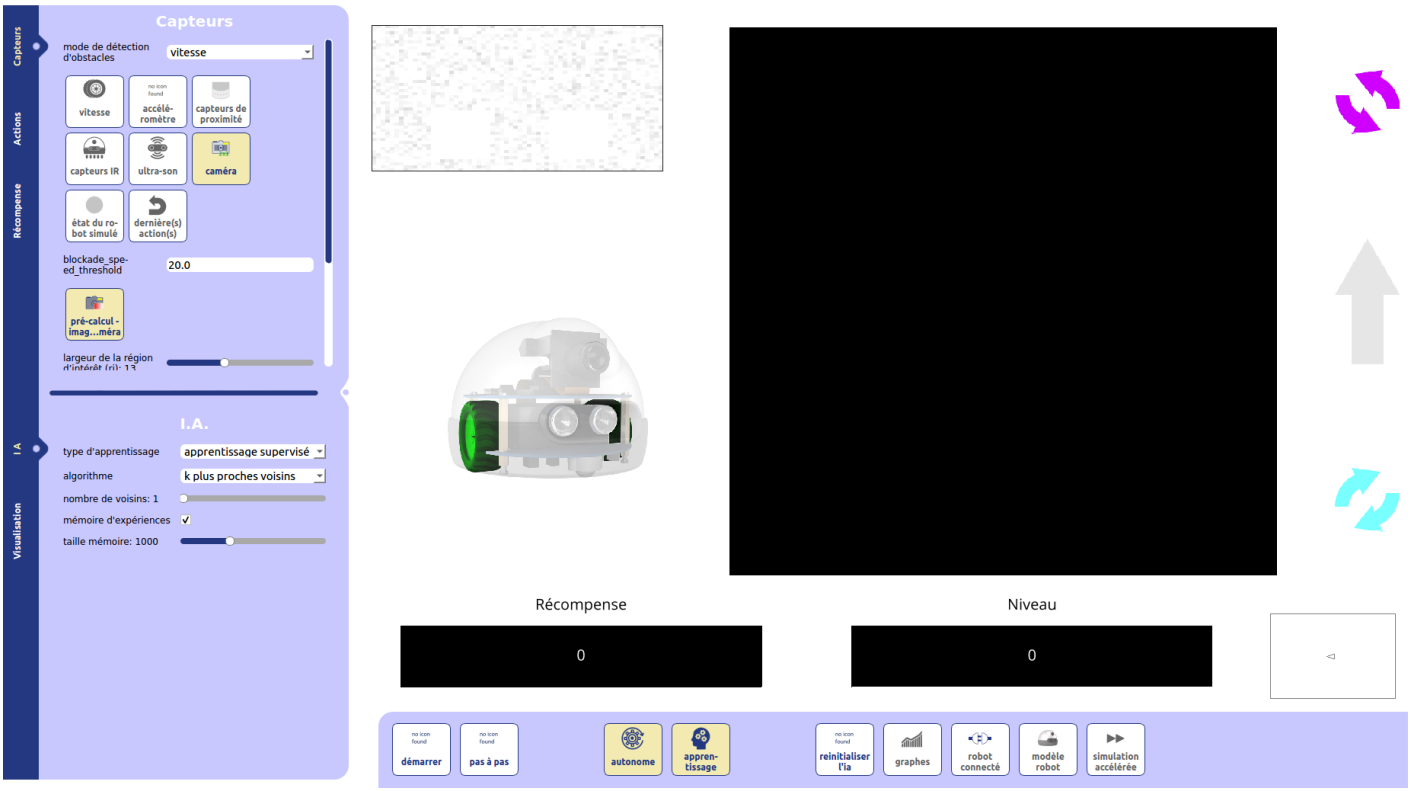
DANS L'ONGLET IA

1) Choisir un apprentissage supervisé



2) Choisir l'algorithme KNN





ATTENTION : *Pour ce scénario, afin d'obtenir le meilleur contraste possible, mettez une surface blanche pour l'arène, et limitez la lumière du jour, par exemple en vous éloignant des fenêtres.*

PHASE D'ENTRAINEMENT

- Connectez le robot 

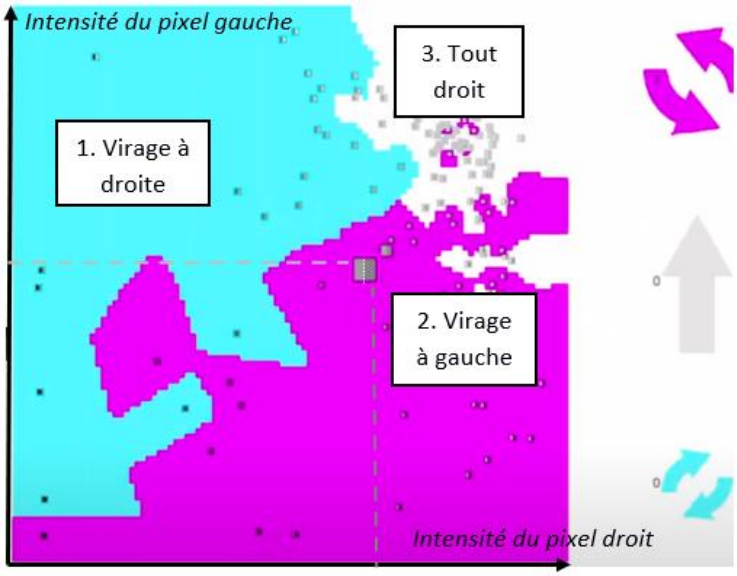
VOUS ETES MAINTENANT CONNECTE AU ROBOT. LE NIVEAU DE SA BATTERIE DOIT S'AFFICHER EN BAS A DROITE (VERIFIEZ QUE LE ROBOT EST BIEN CHARGE)

- Décochez autonome 

ensuite démarrer le robot 

- Tant que le robot n'est pas coincé, appuyez sur aller tout droit.
- Si le robot s'approche d'un mur, appuyez sur virage sur place à gauche ou à droite.

Le graphe se remplit avec un nouveau point à chaque nouvelle action. Ces points x/y représentent le pixel gauche/droite de la caméra et la couleur autour de chaque point est la couleur associée à l'action choisie (violet pour pivoter à droite, blanc pour en avant et bleu ciel pour pivoter à gauche). Ils sont appelés données d'entraînement.



Chaque point sur le graphique représente un donné d'entrée correspondant à 2 super pixels (gauche/droite) de la caméra.

Les 2 super pixels (gauche/droite) de la caméra

NOTEZ BIEN :

Les pixels deviennent plus sombres lorsque le robot s'approche des murs. Normalement si le pixel de gauche est le plus sombre, il faut tourner à droite, et si le pixel de droite est le plus sombre, il faut tourner à gauche.

Lorsque vous aurez trois régions bien claires sur le graphique, arrêtez la phase d'entraînement et réactiver l'autonomie pour passer à la phase de test.

PHASE DE TEST

Si le robot est bien entraîné, il saura éviter les bords par lui-même. Si au contraire il est mal entraîné ou la luminosité dans l'arène n'est pas homogène, cela va produire du bruit dans les régions créées et le robot fera plus d'actions inappropriées.

NOTEZ BIEN :

Afin d'obtenir le meilleur comportement du robot, il faut tester plusieurs valeurs de *k*. Pour changer cette valeur, allez dans l'onglet **« IA »** et changer la valeur du paramètre **« nombre de voisins »**.

Une bonne valeur de *k* réduit l'effet du bruit et permet au robot de mieux gérer les régions inexplorées.

Une fois le robot bien entraîné et que les tests se déroulent bien, vous pouvez passer aux activités.

ACTIVITÉS

ACTIVITÉ 1 :

Entrainez le robot pour 3 minutes et testez-le avec k=1, k=3, k=10 et k=30. Ensuite, d'après ce que vous constatez, essayer de déterminer la valeur de k qui minimise l'effet des erreurs et donne le meilleur comportement du robot.

ACTIVITÉ 2 :

Maintenant que vous avez bien compris l'algorithme, essayez d'entrainer le robot avec le minimum de points possibles qui lui permettent de tourner sans taper contre les murs.

Commencez par faire une version théorique en traçant la carte des mouvements. Quelles zones sont à dessiner ? Quels points sont à placer pour la réaliser ? Testez ces points avec le robot après les avoir rentrés dans le logiciel. Votre modèle théorique a-t-il été applicable à la réalité, ou avez-vous dû y faire des modifications ? Quelles sont les limitations de cette méthode pour un robot tel que celui-là ?

ASTUCE :Vous pouvez déplacer le robot à la main pour choisir les points que vous voulez lui apprendre.

REMARQUE :Nous avons réussi à apprendre au robot à circuler dans l'arène sans taper les murs avec 5 points seulement.

POUR ALLER PLUS LOIN :

Si il vous reste du temps et que vous voulez expérimenter, changez d'algorithme et regardez quelles différences le robot a dans son comportement par rapport au KNN. Quel autre algorithme proposé a un apprentissage plus rapide ? Et quel autre algorithme permettrait, selon vous, l'apprentissage de tâches plus complexes, comme taper dans un ballon vert ou reconnaître des ordres somatiques ?

CONCLUSION :

L'algorithme KNN utilise la distance euclidienne avec les données d'entraînement pour généraliser les décisions à de nouvelles données. Son avantage principal est qu'il est particulièrement simple, et dans le cas de la dimension 2 visualisable à l'écran. En revanche le temps de calcul explose lorsque les données deviennent plus volumineuses, et la distance euclidienne n'est pas toujours la plus appropriée pour certains types de données (par exemple en reconnaissance de visage).