MIASHS
2025-2026

RAPPORT DE PROJET DE ROBOTIQUE

Réalisé par Adélie Bardagi, Alexandre Cholat & Yelli Coulibaly

Unité d'Enseignement : Introduction à l'Intelligence Artificielle

Encadrant : Mr Damien Pellier

Table des Matières

Introduction

Ce rapport présente le déroulement détaillé du projet mené dans le cadre du cours d'Introduction à l'Intelligence Artificielle, en explicitant le problème à résoudre, les choix techniques qui ont été faits, les obstacles et difficultés rencontrées, ainsi que l'organisation et la collaboration qui ont été mises en place durant la réalisation du projet.

Guide de Lecture

Ce rapport est organisé de manière progressive et structurée. Après l'introduction du projet, nous présentons l'analyse du problème puis les stratégies mises en place pour le contrôle, la navigation et la détection du robot. Une analyse critique permet ensuite de mettre en évidence les forces, les limites du travail de groupe et les améliorations possibles de notre projet avant de conclure sur le travail réalisé.

Analyse du problème

Cette tâche sera réalisée dans un environnement qui mesure 3 m × 2 m, avec des lignes colorées délimitant zones et en-buts, et des intersections marquant les positions initiales des palets (Figure 1: Diagramme du terrain de jeu). L'objectif de ce développement est de programmer un robot afin qu'il collecte et dépose efficacement des palets dans l'en-but adverse, de manière totalement autonome.

Le robot en question est un LEGO Mindstorms EV3, équipé de deux moteurs pour le déplacement et d'un moteur actionnant un gripper (aussi référence comme pinscher). Il dispose de capteurs tactiles, ultrason, et de couleurs.

Stratégies Adoptées

Sans adversaire, nous voulions concevoir une stratégie capable de se confronter au cas général. Il serait trop simple de se diriger vers les points de départ des palets, nous partirions donc de la supposition que le robot devrait interagir avec un environnement dynamique.

Après une phase de recherche initiale, il devient évident que la résolution des capteurs serait un défi majeur. Le capteur de couleur s'est révélé extrêmement sensible aux variations de luminosité. Nous avons donc décidé de privilégier les capteurs ultrason et tactile pour percevoir l'environnement. En plus de cela, les moteurs étaient assez imprécis, ce qui nous a poussé à devoir adapter nos stratégies et ajuster notre approche.

Stratégie de navigation

Un choix majeur de conception fut celui de l'estimation de la position du robot. Vu l'imprécision du robot, nous choisissons de minimiser la nécessité du savoir de la position. Grâce à la nature générale de notre algorithme, nous pouvions uniquement stocker l'angle de positionnement du robot par rapport à l'angle de départ.

L'avantage de l'algorithme est son adaptabilité. Il peut être renversé par un adversaire ou subir un imprévu dans sa direction, sans empêcher l'accomplissement de sa mission. Le robot peut aussi gérer des obstacles et des cas de bord plus facilement. Si un premier essai ne marche pas, il peut réessayer, se décaler, et répéter ses tentatives dans une situation complexe. L'inconvénient principal est une certaine redondance au cours de l'exécution et une perte de vitesse. Toutes les méthodes doivent pouvoir gérer les cas de bord, et le robot doit souvent vérifier son alignement par rapport à un mur pour se déplacer.

La méthode *align*

Pour naviguer l'environnement dynamique sous les contraintes d'imprécision et de variabilité physiques, le robot re-calibre sa position en s'alignant par rapport au mur. Le robot doit percevoir les distances autour de lui, s'assurer de la cohérence des distances pour inférer la position du mur, et s'orienter vers la distance la plus proche à lui, c'est-à-dire se centrer sur l'axe du mur. Plusieurs sous-méthodes ont été conçues pour le calcul de la descente du gradient pour identifier le centre, avec des performances variables selon les conditions. Une méta-méthode serait à envisager pour déléguer le choix de sous méthode.

Conception de l'automate de contrôle

La classe Automate

Le comportement du robot a été modélisé sous la forme d'un automate simple à états, permettant d'enchaîner les différentes étapes de la mission de manière structurée.
L'automate repose sur une boucle principale qui s'exécute tant que le nombre de palets ramassés est inférieur à l'objectif fixé.

Quatre états principaux ont été définis :

1. **Aller au centre du plateau** afin de se placer dans une position favorable.
2. **Recherche d'un palet** par rotation et analyse des capteurs.
3. **Approche et saisie du palet** en avançant vers l'objet détecté.
4. **Retour dans le camp adverse** pour déposer le palet.

La transition entre les états repose sur des conditions simples, principalement des booléens indiquant qu'un état a été correctement réalisé.

Stratégie de détection

La détection des palets est réalisée avec une rotation progressive du robot associée à des mesures successives du capteur à ultrasons. Une discontinuité importante entre deux distances mesurées indique la présence d'un objet. Le robot s'oriente alors vers cette discontinuité afin d'engager la phase d'approche. Avec les phases de tests, nous avons constaté que le robot capte une discontinuité avant même d'être en face du palet, nous l'avons donc fait décaler d'un angle de 15° de plus que l'angle où il capte la discontinuité, afin qu'il se positionne en face du palet.

D'autres stratégies plus compliquées de détection ont été tentées, notamment la méthode *angles_grab*, mais malgré de nombreux tests ces méthodes se sont montrées peu fiables. Il nous aurait fallu plus de temps pour les perfectionner.

Analyse critique et améliorations

Au cours de la réalisation du projet, une part importante du temps a été consacrée à la conception de la méthode d'approche du palet. Avec du recul la gestion de cas de bord est gérée par un changement d'état, simplifiant la méthode d'approche. Dans le cas d'approche sans succès, le robot peut simplement recommencer sa recherche de palet, au cas où le palet a été bougé par un autre agent ou simplement perdu par le robot.

La manière d'identifier des palets pourrait être améliorée en mobilisant l'information statique des dimensions du palet. En fonction de la distance connue d'un objet, et de l'identification de l'angle entre discontinuités, un simple calcul trigonométrique permettrait une identification plus fiable. La méthode actuelle se sert uniquement du calcul des discontinuités peu importe la distance de l'objet.

Ouvertures

Compte tenu de notre choix de stratégie, les ouvertures pour amélioration d'efficacité se reposent sur l'optimisation de l'algorithme pour les positions de palets originales. L'architecture du système, combinée à un état supplémentaire de 'début de partie', pourrait mobiliser l'avantage de savoir complet d'information pour une forte rapidité en début de match, en conservant la généralité et adaptativité au changement dans les états suivants.

Il aurait été possible de rendre notre algorithme encore plus robuste mettant en place certaines idées initialement envisagées par exemple, lors de la phase d'approche du palet gérée par la méthode *moveToGrabFacile*, le robot aurait pu reprendre automatiquement la recherche de l'objet lorsqu'il le perd de vue, grâce à l'appel de la méthode *trajectory*, permettant de corriger sa trajectoire ou encore la détection simultanée de plusieurs palets lors d'une rotation complète du robot avec la méthode *angles_grab*.

Dernièrement, une classe de positionnement qui estimait la localisation précise du robot serait bien plus compliqué à mettre en œuvre, mais elle aurait l'avantage de pouvoir cartographier l'environnement du robot, identifiant des zones déjà parcourues ou perçues pour augmenter l'efficacité de la recherche.

Collaboration et Organisation

Le développement de cette solution demandait un cycle d'itération constant et rapide. Nous travaillons sous des contraintes fortes de temps, et l'échéancier du projet était relativement court.

Pour cela nous avions décidé d'employer une méthodologie agile de développement informatique. Nous travaillions synchroniquement en présentiel une fois par semaine. Pendant ces réunions, nous testions les modifications de l'algorithme de la semaine en pratique, identifiant les nouveaux problèmes et les ouvertures qui apparaissent. En fin de réunion, nous nous fixions des objectifs pour la semaine en cours et répartissions les tâches à accomplir.

Il était primordial de pouvoir contribuer parallèlement au code et de manière asynchrone. Nous avons mobilisé GitHub pour cela, apprenant à gérer le répertoire et les méthodes push/pull. L'équipe devient rapidement experte de résolution de conflits de merge. A l'avenir, nous créerons des fichiers séparés dans le répertoire pour minimiser les conflits.

Forces et Faiblesses du Groupe

Forces

Nous notons que nous avons fait preuve d'une répartition plutôt équilibrée des tâches, avec une distribution claire chaque semaine des responsabilités de chacun, mais aussi une contribution égalitaire pour les décisions stratégiques. Chaque membre a accompli le travail assigné comme convenu, cela fait que nous avons pu au fil des semaines avancer de manière régulière.

Nous avons aussi réussi à avoir une bonne communication et beaucoup d'entraide, de sorte que lorsqu'un membre rencontrait un problème, il pouvait toujours trouver de l'aide auprès de l'un de nous.

De plus, nous avons été efficaces lors du travail synchronisé, avec une bonne coordination en temps réel, même si nous aurions pu améliorer certains points, comme nous le verrons dans la partie détaillant nos faiblesses.

Faiblesses

Le travail asynchrone a parfois pu se montrer plus difficile, car malgré la volonté des membres chaque semaine de réaliser le travail demandé, nous n'avions pas de validation régulière de nos avancées, étant souvent plus compliqué de tester notre robot sur le terrain des FABLAB en dehors du créneau de cours, ce qui a pu nous ralentir.

Nous avons eu une mauvaise gestion de temps de test, avec l'utilisation d'un seul ordinateur, et des itérations principalement synchrones, à la place d'un travail qui aurait sûrement été plus fluide de manière asynchrone, mais aussi parfois une perte de temps lorsque le robot se déchargeait complètement et que nous devions alors attendre qu'il soit de nouveau prêt à l'emploi.

Conclusion

Notre approche généraliste est particulièrement adaptée à une situation de jeu chaotique, caractérisée par un nombre limité d'informations disponibles ou par la présence de plusieurs agents en concurrence pour une même mission ou des ressources communes. Bien que l'algorithme ne traite pas l'ensemble des cas possibles, nous sommes satisfaits de notre travail collaboratif ainsi que de la gestion ingénieuse des capteurs et des moteurs malgré leurs capacités limitées.

Glossaire

Terme	Définition
Automate à états	Modèle de comportement du robot décrivant les séquences d'états (phases) par lesquelles le robot passe pour accomplir sa mission.
Gripper / Pincher	Composant robotique motorisé (pince) utilisé par le robot pour saisir et relâcher les palets.
Palet	L'objet (cylindre) que le robot doit collecter et déposer dans l'en-but adverse.
Capteur à ultrasons	Capteur utilisé pour mesurer la distance par rapport à un objet, crucial pour la stratégie de détection des palets.
Discontinuité	Changement brusque dans les mesures de distance du capteur à ultrasons, indiquant la présence d'un objet (un palet) dans le champ de vision.
En-but adverse	Zone désignée sur le terrain de jeu où le robot doit déposer les palets collectés.
GitHub	Plateforme de gestion de versions et de collaboration pour le code source, utilisée par l'équipe pour travailler de manière asynchrone.
Conflits de merge	Problème survenant lors de la fusion (merge) de branches de code dans GitHub lorsque des

Terme	Définition
	modifications concurrentes ont été apportées à la même partie d'un fichier.
Méthodologie Agile	Approche itérative et collaborative de gestion de projet de développement informatique.

Références

1. Pellier, D. (n.d.). *Projet de robotique*. Université Grenoble Alpes. Retrieved December 15, 2025, from https://pellierd.github.io/homepage/teaching/ai/project/
2. Hopcroft, J. E., Motwani, R., & Ullman, J. D. (2006). Introduction to Automata Theory, Languages, and Computation (3rd ed.). Pearson.
3. LeJOS EV3: LeJOS EV3 Development Team. *LeJOS EV3 Java Platform*. http://www.lejos.org/ev3.php

Index

Figure 1: Diagramme du terrain de jeu