جامعة سيدي محمد بن عبد الله بفاس
UNIVERSITE SIDI MOHAMED BEN ABDELLAH DE FES
كلية العلوم ظهر المهراز فاس
FACULTE DES SCIENCES DHAR EL MAHRAZ FES

Laboratoire des Sciences du Numérique de Nantes

Mémoire présenté à
La Faculté des Sciences Dhar El Mahraz Fès
Pour l'obtention du Diplôme de Master

Machine learning Avancé et Intelligence Multimédia (MLAIM)

Parcours Machine learning Avancé et application (MLAA)
Spécialité : Informatique

Apprentissage automatique pour l'allocation de tâches par des bras robotiques

---

Réalisé par :
M. Ibn el qorchy Aymane

Encadré par :
Pr. ARNAUD Laurent (LS2N)
Pr. EN-NAHNAHI Noureddine (USMBA)
Pr. MATHIEU Riand (LS2N)
Pr. El Ghazi Younes (LS2N)

Soutenu le 09 juillet 2025, Devant le jury composé de :

Pr. RIFFI jamal :	USMBA	- Président
Pr. EN-NAHNAHI Noureddine :	USMBA	- Membre de jury
Pr. ALAMI Hamza :	USMBA	- Membre de jury
Pr. BENLAHBIB Abdessamad :	USMBA	- Membre de jury
Pr. ARNAUD Laurent :	LS2N	- Membre de jury
Pr. MATHIEU Riand :	LS2N	- Membre de jury
Pr. El GHAZI Younes :	LS2N	- Membre de jury

Promotion : 2024/2025

Remerciements

Je remercie avant tout mes encadrants du laboratoire LS2N, M. Arnaud Laurent, M. Mathieu Riand et M. Younès El Ghazi, pour m'avoir accueilli au sein de leur équipe et pour m'avoir accompagné tout au long de mes travaux de recherche. Je les remercie pour leurs conseils avisés, leur disponibilité constante et leur soutien précieux qui ont grandement contribué à ce projet.

Je tiens à exprimer ma profonde gratitude envers mon encadrant académique, M. Noureddine En nahnahi, d'avoir accepté de me guider tout au long de cette expérience et pour ses orientations judicieuses qui ont enrichi ma réflexion.

Je souhaite exprimer ma sincère reconnaissance envers M. Jamal Riffi, Coordinateur du Master MLAIM à Fès, pour la qualité exceptionnelle de la formation dispensée et son engagement dans le développement de ce parcours d'excellence.

Je suis reconnaissant envers toute l'équipe du laboratoire LS2N, en particulier l'équipe CPS3, qui m'a accueilli en son sein durant cette période, ainsi qu'envers tous les membres du laboratoire, grâce à qui j'ai pu effectuer ce stage dans un cadre stimulant et convivial.

Je remercie également l'ensemble des enseignants et du personnel administratif de la Faculté des Sciences Dhar El Mahraz de Fès, ainsi que les responsables du parcours MLAIM, pour leur contribution à ma formation et leur accompagnement tout au long de ce cursus.

Mes remerciements s'adressent aussi aux membres du jury qui ont accepté d'évaluer ce travail et de participer à ma soutenance.

Enfin, je tiens à remercier ma famille et mes proches pour leur soutien inconditionnel et leurs encouragements constants qui m'ont permis de mener à bien ce projet dans les meilleures conditions.

Résumé

L'évolution de la robotique industrielle nécessite des solutions avancées pour optimiser la coordination des robots dans des environnements de production complexes. Ce mémoire présente les travaux menés au Laboratoire des Sciences du Numérique de Nantes (LS2N), axés sur l'apprentissage automatique appliqué à la manipulation robotique.

S'inscrivant dans une démarche plus large visant à optimiser l'allocation de tâches entre systèmes robotiques autonomes, ce travail se focalise sur un objectif préparatoire essentiel. Il consiste à développer les capacités d'apprentissage d'un bras robotique unique, afin qu'il puisse manipuler des objets de manière autonome dans un espace tridimensionnel, posant ainsi les bases nécessaires à la coordination future.

Notre approche utilise l'apprentissage par renforcement, implémenté dans un environnement de simulation développé . L'environnement virtuel reproduit les conditions réelles avec un espace de travail délimité, des objets cibles positionnés aléatoirement, et l'intégration de la dynamique physique complète.

Les expérimentations révèlent des performances encourageantes avec une progression significative des capacités de manipulation du robot. Les résultats démontrent le potentiel de l'approche pour l'apprentissage autonome de tâches complexes de manipulation.

Ces travaux ouvrent la voie vers des applications industrielles réelles impliquant la coordination de plusieurs robots opérant simultanément sur des tâches complexes avec multiples objets, gestion des collisions et allocation dynamique des ressources. L'objectif futur consiste à déployer cette méthodologie sur des systèmes robotiques réels pour valider son efficacité pratique dans des environnements industriels partagés.

Abstract

The evolution of industrial robotics requires advanced solutions to optimize robot coordination in complex production environments. This thesis presents work conducted at the Laboratoire des Sciences du Numérique de Nantes (LS2N), focusing on machine learning applied to robotic manipulation.

As part of a broader initiative aimed at optimizing task allocation among autonomous robotic systems, this work focuses on an essential preparatory objective. It consists of developing the learning capabilities of a single robotic arm, enabling it to autonomously manipulate objects in a three-dimensional space, thereby laying the necessary groundwork for future coordination.

Our approach utilizes reinforcement learning, implemented within a custom-developed simulation environment. The virtual environment replicates real-world conditions with a bounded workspace, randomly positioned target objects, and the integration of complete physical dynamics.

Experiments reveal encouraging performance with a significant progression in the robot's manipulation capabilities. The results demonstrate the potential of the approach for the autonomous learning of complex manipulation tasks.

This work paves the way for real-world industrial applications involving the coordination of multiple robots operating simultaneously on complex tasks with multiple objects, collision management, and dynamic resource allocation. The future objective is to deploy this methodology on real robotic systems to validate its practical effectiveness in shared industrial environments.

Liste des Acronymes

Acronyme	**Signification**
RL	Reinforcement Learning (Apprentissage par Renforcement)
DOF	Degrees of Freedom (Degrés de Liberté)
DRL	Deep Reinforcement Learning (Apprentissage par Renforcement Profond)
EE	End Effector (Effecteur Final)
IIWA	Intelligent Industrial Work Assistant
KUKA	KUKA Roboter GmbH (fabricant de robots industriels)
CPS3	Cyber-Physical Systems, Safety and Security
IUT	Institut Universitaire de Technologie
LS2N	Laboratoire des Sciences du Numérique de Nantes
CNRS	Centre National de la Recherche Scientifique
MILP	Mixed Integer Linear Programming (Programmation Linéaire en Nombres Entiers Mixtes)
MLP	Multi-Layer Perceptron
PPO	Proximal Policy Optimization
DQN	Deep Q-Network
IAE	Integral of Absolute Error (Intégrale de l'Erreur Absolue)
ROS	Robot Operating System
MPC	Model Predictive Control (Commande Prédictive par Modèle)
RRT	Rapidly-exploring Random Trees
SMC	Sliding Mode Control (Commande par Mode Glissant)
SPT	Shortest Processing Time (Temps de Traitement le plus Court)
URDF	Universal Robot Description Format

Table des matières

Table des figures

Liste des tableaux

Introduction générale

L'industrie manufacturière a beaucoup évolué avec l'arrivée des robots industriels sur les chaînes de production.les robots industriels comme Fanuc, Kuka et ABB équipent maintenant la plupart des usines d'assemblage et de conditionnement. Mais il y a un problème : ces robots travaillent généralement dans des espaces séparés et exclusifs pour éviter les collisions.

Cette méthode traditionnelle fonctionne bien pour les tâches répétitives, mais elle a ses limites. L'industrie demande maintenant plus de flexibilité et d'efficacité. Il serait plus intéressant d'optimiser l'utilisation de l'espace en faisant collaborer plusieurs robots dans une zone commune. Cette organisation intelligente permettrait de maximiser la productivité tout en évitant les conflits. Cette approche devient encore plus importante pour les tâches de picking, où la variété des objets et leurs positions changent constamment.

Mon stage de cinq mois au Laboratoire des Sciences du Numérique de Nantes (LS2N) s'inscrit dans cette démarche. Le laboratoire travaille sur un projet ambitieux qui vise à faire collaborer quatre robots de picking dans un espace de travail partagé. Ces robots doivent effectuer des tâches non répétitives, ce qui représente un vrai changement par rapport aux configurations industrielles actuelles.

Le projet global comprend plusieurs parties importantes. La première partie concerne la détection automatique des objets et la localisation de leurs emplacements grâce à la vision par ordinateur. La deuxième partie se concentre sur l'attribution des tâches aux robots et l'organisation des opérations de picking. Cette problématique a été modélisée comme un problème d'optimisation linéaire en nombres entiers. L'objectif final consiste à utiliser les solutions optimales obtenues comme données d'entraînement pour développer des méthodes d'apprentissage automatique qui peuvent prédire ces décisions d'allocation selon la disposition des objets.

Avant de s'attaquer à la complexité de la coordination multi-robots, il faut d'abord s'assurer qu'un robot seul possède les capacités d'apprentissage nécessaires pour évoluer efficacement dans un environnement partagé. C'est exactement ce qui a motivé ma contribution au projet global.

Mon travail porte principalement sur la deuxième phase du projet, axée sur le développement et l'évaluation de méthodes d'apprentissage automatique appliquées à un bras robotique. L'objectif est d'optimiser ses capacités de manipulation d'objets dans un environnement en trois dimensions. Cette étape représente une phase préparatoire essentielle en vue de l'objectif final : la coordination entre plusieurs robots. Elle permet d'analyser et d'améliorer les performances individuelles avant de s'attaquer aux interactions complexes dans un contexte multi-robots.

J'ai choisi l'apprentissage par renforcement comme approche principale. Cette méthode permet aux robots d'apprendre en interagissant avec leur environnement. Contrairement aux techniques traditionnelles de programmation robotique qui nécessitent de modéliser complètement l'environnement, cette approche permet au robot d'adapter son comportement selon l'expérience qu'il acquiert. Cette capacité sera cruciale pour les futurs scénarios multi-robots.

Les contraintes pratiques m'ont orienté vers une approche de simulation . Cette méthodologie présente l'avantage de permettre une exploration extensive des stratégies d'apprentissage dans des conditions contrôlées. Elle évite aussi les risques et coûts associés à l'expérimentation sur du matériel physique. L'environnement virtuel que j'ai développé reproduit fidèlement les conditions réelles d'un espace de travail robotique, en intégrant les contraintes physiques et cinématiques du robot .

Cette phase d'apprentissage individuel apporte des informations précieuses pour le projet global. Comprendre comment un robot unique apprend à manipuler des objets efficacement fournit des bases solides pour concevoir plus tard des stratégies de coordination entre plusieurs robots. Les méthodes d'apprentissage identifiées, les difficultés rencontrées et les solutions développées constituent autant d'éléments qui enrichiront la phase suivante du projet.

Mes résultats se concentrent sur un robot individuel, mais ils démontrent que l'approche d'apprentissage par renforcement fonctionne pour les tâches de manipulation robotique. Cette validation constitue un préalable nécessaire avant d'envisager l'extension vers des scénarios multi-robots où les interactions entre agents ajoutent une dimension de complexité supplémentaire.

L'intégration de mes travaux dans le projet global soulève plusieurs points intéressants. Les capacités d'apprentissage développées pour un robot individuel peuvent être étendues à un système multi-robots. Les stratégies d'apprentissage individuel peuvent informer les algorithmes d'allocation de tâches. Il faut aussi concilier l'autonomie d'apprentissage de chaque robot avec les contraintes d'optimisation globale du système.

Ces réflexions ouvrent la voie vers des développements futurs ambitieux. L'objectif final consiste à déployer ces méthodes sur des systèmes robotiques réels impliquant la coordination de plusieurs robots. Ces robots devront opérer simultanément sur des tâches complexes avec multiples objets, gestion des collisions et allocation dynamique des ressources. Le passage du laboratoire à l'application industrielle nécessitera une intégration harmonieuse entre les capacités d'apprentissage individuel développées et les méthodes d'optimisation globale du projet.

Ce mémoire s'organise autour de plusieurs parties complémentaires. Je présenterai d'abord l'état de l'art des méthodes d'apprentissage en robotique et des techniques d'allocation de tâches. Ensuite, je détaillerai ma contribution spécifique et la méthodologie d'apprentissage par renforcement développée. Je présenterai les expérimentations et résultats obtenus.

L'ambition de ce projet reste réaliste mais innovante. Je veux démontrer que les approches d'apprentissage automatique fonctionnent pour la robotique industrielle, en établissant les fondations nécessaires à la future coordination multi-robots. Les défis sont nombreux, mais cette convergence entre apprentissage individuel et optimisation collec-

tive ouvre des perspectives prometteuses pour l'industrie 4.0.

Chapitre 1

Contexte du stage

Ce chapitre présente le contexte dans lequel mon stage s'est déroulé ainsi que les différents éléments qui ont contribué à sa réalisation. Je commencerai par une présentation du laboratoire LS2N et de son équipe robotique, qui ont fourni le cadre et l'expertise nécessaires pour mener à bien ce projet. Ensuite, j'expliquerai le projet global de coordination multi-robots qui a servi de référence et de guide pour mon travail.

1.1 Présentation du laboratoire LS2N

Le Laboratoire des Sciences du Numérique de Nantes (LS2N) [1] est une unité mixte de recherche qui dépend de l'Université de Nantes, de l'École Centrale de Nantes et du CNRS. Le laboratoire se concentre sur les sciences du numérique avec plusieurs équipes spécialisées dans différents domaines comme la robotique, l'intelligence artificielle, l'informatique théorique et les systèmes automatiques.

Mon stage s'est déroulé à l'institut Universitaire de Technologie de Nantes (IUT) au sein de l'équipe CPS3 du LS2N, dirigée par des chercheurs reconnus dans le domaine de la robotique industrielle et de l'optimisation. Cette équipe travaille particulièrement sur les problématiques de coordination multi-robots, d'allocation de tâches et d'optimisation des systèmes robotiques autonomes.

Le laboratoire possède des infrastructures modernes avec des plateformes de simulation avancées et des équipements robotiques variés. L'environnement de recherche favorise les collaborations interdisciplinaires et permet de travailler sur des projets innovants qui associent théorie et applications pratiques.

1.2 L'équipe CPS3

Au cours de mon stage, j'ai eu l'opportunité d'intégrer l'équipe CPS3 (Conception, Pilotage, Surveillance et Supervision des systèmes) du laboratoire LS2N (Laboratoire des Sciences du Numérique de Nantes). Avant de détailler les travaux réalisés, il est important

1. https://www.ls2n.fr/

FIGURE 1.1 – Institut Universitaire de Technologie de Nantes

FIGURE 1.2 – Institut Universitaire de Technologie de Nantes

de présenter l'expertise de cette équipe dans le domaine des systèmes cyber-physiques.

L'équipe CPS3 se spécialise dans la modélisation, le pilotage et la supervision de systèmes complexes, souvent contraints par des exigences de performance, de sécurité ou de robustesse. Ses recherches portent particulièrement sur les systèmes cyber-physiques, combinant aspects physiques, numériques et logiciels, et s'inscrivent dans des contextes industriels concrets, comme la logistique, la robotique ou la production manufacturière.

L'objectif de l'équipe est de développer des approches théoriques et appliquées permettant une gestion intelligente de systèmes dynamiques, avec un accent particulier sur la résilience, la commande prédictive, la gestion des incertitudes, ou encore les jumeaux numériques. Ces travaux sont soutenus par des méthodes issues de l'automatique, de l'optimisation, et de l'intelligence artificielle.

L'équipe est constituée de plusieurs enseignants-chercheurs, chercheurs, doctorants et ingénieurs. Elle participe activement à des projets de recherche nationaux et européens, en collaboration avec des industriels et d'autres laboratoires académiques.

1.3 Le projet global de coordination multi-robots

Mon stage s'inscrit dans un projet plus large qui vise à développer des méthodes de collaboration entre quatre robots de picking évoluant dans un espace de travail partagé. Ce projet répond à un besoin industriel réel : optimiser l'utilisation de l'espace dans les chaînes de production en permettant à plusieurs robots de travailler ensemble plutôt que dans des zones séparées.

Le projet se décompose en plusieurs phases complémentaires. La première phase concerne le développement d'algorithmes de vision par ordinateur pour la détection et la localisation automatique des objets à manipuler. Cette partie du projet utilise des techniques avancées de traitement d'image et d'apprentissage automatique pour identifier les objets dans des environnements complexes.

La deuxième phase, qui constitue le cœur du projet, se concentre sur l'allocation optimale des tâches entre les robots et l'ordonnancement des opérations de picking. Cette problématique a été modélisée comme un problème d'optimisation linéaire en nombres

entiers, permettant d'obtenir des solutions mathématiquement optimales pour différentes configurations d'objets.

La troisième phase vise à développer des méthodes d'apprentissage automatique capables de prédire rapidement ces allocations optimales à partir de la disposition des objets, évitant ainsi les temps de calcul prohibitifs de l'optimisation exacte en contexte industriel.

1.4 Positionnement de ma contribution

Dans ce contexte général, ma contribution spécifique se concentre sur une étape préparatoire essentielle : le développement et l'évaluation de capacités d'apprentissage pour un robot individuel. Cette approche peut sembler moins ambitieuse que l'objectif final de coordination multi-robots, mais elle répond à une nécessité logique.

Avant de coordonner plusieurs robots, il faut s'assurer que chaque robot possède les capacités d'apprentissage et d'adaptation nécessaires pour évoluer efficacement dans un environnement partagé. Mon travail vise donc à caractériser et optimiser les performances d'apprentissage d'un bras robotique dans des tâches de manipulation d'objets.

1.5 Objectifs spécifiques de mon stage

Les objectifs de mon stage se déclinent en plusieurs aspects complémentaires :

Développement méthodologique : Concevoir et implémenter une approche d'apprentissage par renforcement adaptée aux tâches de manipulation robotique

Environnement de simulation : Créer un environnement de simulation réaliste qui reproduit fidèlement les conditions d'un espace de travail robotique, incluant les contraintes physiques et cinématiques du robot Kuka.

Évaluation des performances : Analyser les capacités d'apprentissage du robot dans différentes configurations de tâches de manipulation, identifier les facteurs limitants et proposer des améliorations.

Intégration future : Préparer les bases théoriques et pratiques pour l'extension future vers la coordination multi-robots, en identifiant les patterns d'apprentissage et les stratégies efficaces.

1.6 Ressources et contraintes

Mon stage s'est déroulé dans de bonnes conditions à l'IUT de Nantes. J'ai eu accès à un ordinateur personnel équipé des logiciels nécessaires pour développer et tester mes algorithmes d'apprentissage par renforcement. L'environnement de travail était calme et propice à la concentration, avec un bureau dédié au sein de l'équipe CPS3.

Pour les besoins de simulation, j'ai utilisé Python avec les bibliothèques PyBullet et OpenAI Gym sur mon poste de travail. Ces outils m'ont permis de créer un environnement

virtuel pour entraîner le robot . Bien que les simulations nécessitent parfois du temps de calcul, la configuration disponible était suffisante pour mener à bien les expérimentations.

L'approche par simulation était la méthode la plus adaptée pour mon projet. Cela m'a permis de tester différentes configurations et paramètres d'apprentissage en toute sécurité, sans risquer d'endommager du matériel coûteux. De plus, la simulation offre la possibilité de répéter les expériences dans des conditions identiques, ce qui facilite l'analyse des résultats.

La durée de cinq mois du stage était appropriée pour atteindre les objectifs fixés. J'ai pu me familiariser avec les outils d'apprentissage par renforcement, développer l'environnement de simulation, implémenter l'algorithme d'apprentissage et analyser les résultats obtenus. Cette période m'a aussi permis de bien comprendre les enjeux du projet global de coordination multi-robots.

L'encadrement de l'équipe CPS3 a été très enrichissant. Les réunions régulières avec mes encadrants m'ont aidé à orienter mon travail et à surmonter les difficultés techniques rencontrées. L'ambiance de travail au laboratoire favorise les échanges et j'ai pu bénéficier de l'expérience des autres chercheurs et doctorants.

Ce contexte m'a permis de découvrir concrètement le monde de la recherche en robotique et intelligence artificielle. J'ai appris à mener un projet de recherche de manière autonome tout en bénéficiant de conseils d'experts. Cette expérience constitue une excellente préparation pour la suite de mon parcours académique et professionnel.

1.7 Conclusion

Ce chapitre a présenté le cadre dans lequel mon stage s'est déroulé.nous avons défini les objectifs spécifiques de stage, mettant en évidence deux aspects majeurs. Le projet global de coordination multi-robots constitue un défi ambitieux qui nécessite une approche progressive.

Ma contribution se positionne comme une étape fondamentale de ce projet en se concentrant sur l'apprentissage d'un robot individuel. Cette approche permet de poser les bases nécessaires avant d'aborder la complexité de la coordination entre plusieurs robots. Les objectifs fixés pour mon stage sont réalisables et s'inscrivent logiquement dans la démarche globale de l'équipe.

Dans le chapitre suivant, je présenterai l'état de l'art des méthodes d'apprentissage par renforcement en robotique, ce qui permettra de situer mon approche par rapport aux travaux existants et de justifier les choix méthodologiques adoptés pour ce projet.

Chapitre 2

État de l'Art et Contexte Théorique

Ce chapitre présente une revue des travaux existants dans les domaines liés à notre projet. Nous commencerons par un aperçu général de la robotique autonome et des systèmes multi-robots, puis nous nous concentrerons sur les approches d'allocation de tâches et de coordination de mouvements. Nous examinerons ensuite les méthodes d'apprentissage automatique appliquées à la robotique avant de positionner notre contribution dans ce contexte.[1]

2.1 Robotique autonome et systèmes multi-robots

FIGURE 2.1 – applications de bras multi-robots

2.1.1 Évolution de la robotique autonome

La robotique autonome a considérablement évolué ces dernières décennies. Les robots autonomes sont aujourd'hui conçus pour fonctionner dans des environnements non structurés sans intervention humaine constante. Cette autonomie nécessite des technologies avancées en perception, planification, prise de décision et interaction avec l'environnement.

L'évolution des capacités de calcul et des algorithmes d'intelligence artificielle a permis aux robots de traiter des informations complexes en temps réel. Les systèmes de vision par ordinateur, les capteurs avancés et les algorithmes de localisation et cartographie

simultanées ont considérablement amélioré la capacité des robots à comprendre et naviguer dans leur environnement.

Dans le contexte industriel, cette évolution se traduit par une transition des robots traditionnels, programmés pour des tâches répétitives dans des environnements contrôlés, vers des systèmes adaptatifs capables de gérer la variabilité et l'incertitude. Cette transition est particulièrement importante pour les applications de picking et de manipulation d'objets, où la diversité des produits et la variation de leurs positions nécessitent une grande flexibilité.

2.1.2 Défis des environnements non-structurés

Les environnements non-structurés posent des défis particuliers pour la robotique autonome. Contrairement aux environnements industriels traditionnels où les robots évoluent dans des espaces délimités et prévisibles, les environnements non-structurés se caractérisent par leur variabilité et leur imprévisibilité.

Les principaux défis incluent la perception et l'interprétation de scènes complexes, la planification de trajectoires adaptatives, et la gestion des interactions avec des objets de formes et tailles variées. La présence d'obstacles mobiles ou temporaires ajoute une dimension supplémentaire de complexité, nécessitant des capacités de replanification en temps réel.

Pour les systèmes multi-robots, ces défis sont amplifiés par la nécessité de coordonner plusieurs agents évoluant simultanément. La gestion des interactions entre robots devient critique pour éviter les collisions et optimiser l'efficacité globale du système. Cette coordination nécessite des mécanismes de communication et de synchronisation sophistiqués.

2.2 Allocation de tâches en robotique

2.2.1 Approches classiques d'allocation

L'allocation de tâches dans les systèmes robotiques consiste à déterminer quel robot doit exécuter quelle tâche, dans quel ordre, et à quel moment. Cette problématique, bien qu'apparemment simple, révèle une complexité remarquable dans des contextes multi-robots.

Les méthodes classiques d'allocation de tâches peuvent être regroupées en plusieurs catégories. La minimisation de la distance constitue l'approche la plus intuitive, où chaque tâche est assignée au robot le plus proche. Cette méthode, bien que simple à implémenter, ne prend pas en compte les contraintes globales du système et peut conduire à des solutions sous-optimales.

Les approches basées sur le marché introduisent des mécanismes d'enchères où les robots négocient l'attribution des tâches. Chaque robot évalue le coût d'exécution d'une tâche et propose une enchère. Cette approche décentralisée présente l'avantage de la robustesse, mais peut nécessiter des mécanismes de communication complexes.

La programmation par contraintes permet de modéliser explicitement les contraintes du système (capacités des robots, contraintes temporelles, interdépendances entre tâches). Cette approche offre une grande flexibilité de modélisation mais peut souffrir de temps de calcul élevés pour les problèmes de grande taille. [1]

2.2.2 Optimisation par MILP (Mixed Integer Linear Programming

La programmation linéaire en nombres entiers mixtes (MILP) constitue une approche rigoureuse et puissante pour modéliser l'allocation optimale de tâches. Elle permet de formuler mathématiquement le problème en intégrant simultanément plusieurs objectifs et contraintes. Dans le contexte des systèmes multi-robots, cette méthode offre la possibilité de définir différents types de variables décisionnelles : des variables binaires pour indiquer l'affectation d'un robot à une tâche ou à un objet, et des variables continues pour modéliser les temps d'exécution. Le modèle est enrichi par un ensemble de contraintes garantissant à la fois la faisabilité des actions planifiées et l'optimisation globale des performances du système.

[2]L'objectif principal est généralement de minimiser le temps de cycle total, c'est-à-dire le temps nécessaire pour terminer l'ensemble des tâches. Cependant, d'autres critères peuvent également être pris en compte, comme la réduction de la consommation énergétique ou la répartition équilibrée de la charge de travail entre les robots.

L'un des atouts majeurs de cette approche réside dans sa capacité à fournir des solutions optimales pour des problèmes de taille modérée, tout en permettant l'intégration de contraintes complexes spécifiques aux systèmes robotiques. Néanmoins, son principal inconvénient demeure le coût computationnel : le temps de calcul croît rapidement avec la taille du problème, ce qui limite son application dans des environnements à grande échelle ou dans des contextes nécessitant une réactivité en temps réel.

2.2.2.1 Notation

- C : Ensemble des cubes représentant l'espace de travail (ressources spatiales)
- O : Ensemble de tous les objets
- R : Ensemble de tous les robots
- E : Ensemble des durées des trajectoires
- T : Nombre de périodes considérées
- N_o^r : Nombre d'étapes de la trajectoire lorsque le robot $r \in R$ manipule l'objet $o \in O$
- $I_{o,s,c}^r$: =1 si le cube $c \in C$ est occupé à l'étape $s \in \{1, ..., N_o^r\}$, 0 sinon
- A_o^r : =1 si le robot r peut manipuler l'objet o, 0 sinon
- $e_{r,o} \in E$: Temps requis pour que le robot r ramasse l'objet o

2.2.2.2 Variables de décision

- $X^r_{o,t,s}$: =1 si l'étape s de la trajectoire (r, o) se produit à la période t, 0 sinon
- Y^r_o : =1 si le robot r est assigné à l'objet o, 0 sinon
- C_{max} : Temps total d'exécution (makespan)

2.2.2.3 Formulation mathématique

$$\text{Minimiser} \quad C_{max} \tag{2.1}$$

$$\text{Sous contraintes :} \quad \sum_{t=N^r_o}^{T} X^r_{o,t,N^r_o} \cdot t \leq C_{max}, \quad \forall r \in R, \forall o \in O \tag{1}$$

$$\sum_{r \in R} \sum_{t=N^r_o}^{T} X^r_{o,t,N^r_o} \cdot A^r_o = 1, \quad \forall o \in O \tag{2}$$

$$\sum_{t=1}^{T} X^r_{o,t,0} \leq 1, \quad \forall r \in R, \forall o \in O \tag{3}$$

$$\sum_{o \in O} \sum_{s=1}^{\min(t,N^r_o)} X^r_{o,t,s} \leq 1, \quad \forall r \in R, \forall t \in \{1, ..., T\} \tag{4}$$

$$\sum_{r \in R} \sum_{o \in O} \sum_{s=1}^{\min(t,N^r_o)} X^r_{o,t,s} \cdot I^r_{o,s,c} \leq 1, \quad \forall t, \forall c \in C \tag{5}$$

$$X^r_{o,t,s} \geq X^r_{o,t+1,s+1} \quad \text{si } t = s, \quad \forall t \in \{1, ..., T-1\}, \forall r, \forall o, \forall s \tag{6}$$

$$X^r_{o,t,s} + X^r_{o,t,s+1} \geq X^r_{o,t+1,s+1} \quad \text{si } t \neq s, \quad \forall t \in \{1, ..., T-1\}, \forall r, \forall o, \forall s \tag{7}$$

2.2.2.4 Interprétation

- **Objectif** : Minimisation du temps total d'exécution (1)
- **Contraintes** :
  - (1) Définition du makespan
  - (2) Allocation unique des objets
  - (3) Cohérence des trajectoires
  - (4) Non-simultanéité des tâches
  - (5) Évitement des collisions
  - (6)-(7) Contraintes de précédence temporelle

2.2.2.5 Calcul des paramètres

L'horizon temporel est déterminé par :

$$H = \sum_{o \in O} \max_{r \in R} \{e_{r,o}\}$$

Le nombre de périodes :

$$T = \left\lceil \frac{H}{TI} \right\rceil$$

où TI est l'intervalle temporel (100ms).

2.2.3 Approches heuristiques

Face aux limitations des méthodes exactes pour les problèmes de grande taille, les approches heuristiques offrent des alternatives pratiques. Ces méthodes sacrifient la garantie d'optimalité au profit de temps de calcul acceptables pour les applications industrielles.

La méthode "Detect-and-Pause" illustre une approche heuristique efficace. Cette méthode procède en plusieurs étapes : allocation initiale des tâches en ignorant les collisions, séquencement des tâches selon la règle SPT (Shortest Processing Time First), détection des collisions potentielles via une matrice d'occupation, et résolution des conflits par mise en pause et réajustement des trajectoires.

Cette approche itérative présente l'avantage de fournir des solutions de qualité acceptable dans des temps de calcul réduits. Elle peut gérer des problèmes de taille significativement plus importante que les méthodes exactes tout en maintenant une complexité algorithmique raisonnable.

D'autres approches heuristiques incluent les algorithmes génétiques, l'optimisation par essaims particulaires, et les méthodes de recherche locale. Ces approches métaheuristiques peuvent explorer efficacement l'espace des solutions pour des problèmes complexes.

2.3 Coordination de mouvements et évitement de collisions

2.3.1 Planification de trajectoires

La planification de trajectoires constitue un défi fondamental en robotique, particulièrement dans des environnements partagés où plusieurs robots évoluent simultanément. Cette problématique nécessite de concilier l'efficacité des mouvements individuels avec les contraintes de coordination globale.

Pour un robot seul, la planification de trajectoires se concentre principalement sur la manipulation et la navigation. Les algorithmes classiques comme A* ou Dijkstra permettent de trouver des chemins optimaux dans des espaces discretisés. Cependant, ces approches souffrent de limitations importantes : temps de calcul élevé et difficulté à prendre en compte des obstacles complexes ou des contraintes cinématiques.

Les algorithmes d'échantillonnage aléatoire, notamment RRT (Rapidly-exploring Random Trees) et ses variantes comme RRT-Connect, offrent des alternatives plus efficaces. Ces méthodes explorent aléatoirement l'espace des configurations pour construire un arbre de trajectoires faisables. L'avantage principal de ces approches réside dans leur capacité à gérer des espaces de configuration de haute dimension et des contraintes géométriques complexes.

Le contrôle prédictif (MPC - Model Predictive Control) représente une autre approche pour la planification de trajectoires. Cette méthode optimise les commandes sur un horizon temporel fini en tenant compte des contraintes dynamiques du système. L'approche MPC présente l'avantage de pouvoir intégrer des contraintes multiples et de permettre une replanification en ligne.

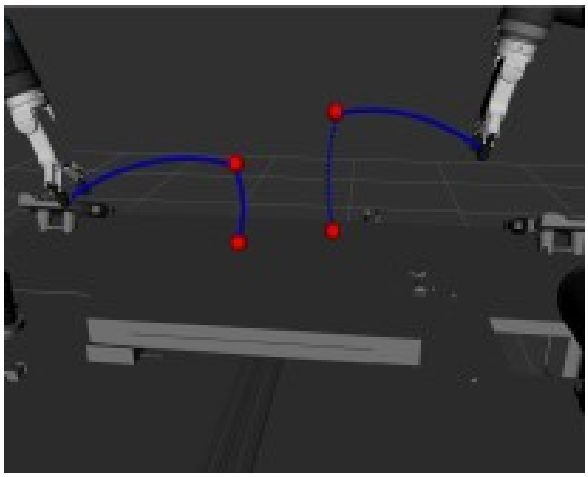

FIGURE 2.2 – Les trajectoires des robots

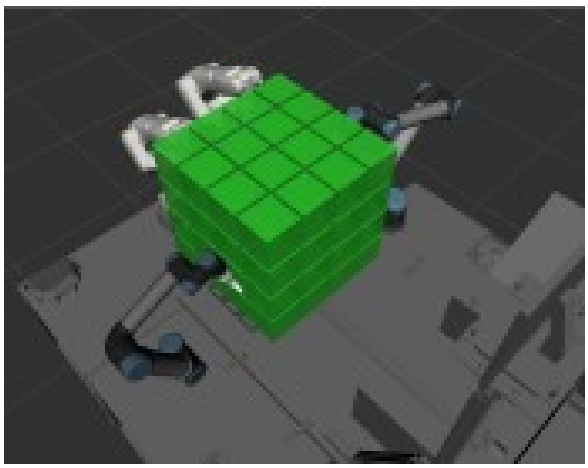

FIGURE 2.3 – Discretisation de l'espace de travail

2.3.2 Stratégies d'évitement de collisions

L'évitement de collisions devient particulièrement critique dans les systèmes multi-robots où plusieurs agents partagent le même espace de travail. Les collisions peuvent non seulement endommager les équipements mais aussi compromettre la sécurité et l'efficacité du système global.

Les techniques de distance minimale constituent l'approche la plus directe. Ces méthodes maintiennent une distance de sécurité minimale entre les robots en surveillant continuellement leurs positions relatives. Bien que simple à implémenter, cette approche peut être conservative et conduire à des mouvements sous-optimaux.

[2]Les représentations voxel offrent une approche plus sophistiquée en discrétisant l'espace de travail en cubes élémentaires. Chaque robot occupe un ensemble de voxels à chaque instant, et les collisions sont détectées en identifiant les conflits d'occupation. Cette approche permet une représentation précise de l'occupation spatiale mais nécessite un compromis entre précision et complexité computationnelle.[3]

La détection en temps réel implique la surveillance continue des trajectoires et la mise en œuvre de mécanismes de réaction rapide. Ces systèmes utilisent des capteurs embarqués

et des algorithmes de traitement d'image pour détecter les obstacles dynamiques et ajuster les trajectoires en conséquence.

Les approches plus avancées intègrent la prédiction des mouvements futurs pour anticiper les collisions potentielles. Ces méthodes prédictives permettent une coordination plus fluide en évitant les arrêts brusques et les manœuvres d'évitement de dernière minute.

2.4 Résultats et Comparaison

Les expérimentations révèlent des performances contrastées entre les approches MILP et heuristique. Pour les petites instances (6 objets), le MILP avec allocation libre ($MILP_m$) fournit des solutions optimales (GAP=0%) en un temps raisonnable (<30 min), surpassant de 15-20% l'heuristique. Cependant, pour les problèmes plus grands, l'heuristique démontre sa supériorité avec des temps de calcul 100 fois plus rapides et des écarts de qualité limités (GAP moyen de 9.7%). Notamment, dans 25% des cas, l'heuristique surpasse même le MILP avec allocation prédéfinie (GAP jusqu'à -11%). La discrétisation spatiale influence significativement les résultats, avec 27 cubes comme compromis optimal, améliorant la qualité des solutions de 22% sans pénaliser excessivement le temps de calcul. Ces résultats positionnent l'heuristique comme solution pragmatique pour les applications industrielles nécessitant des temps de réponse courts, tandis que le MILP reste pertinent pour les petits problèmes où l'optimalité est critique.

2.5 Apprentissage automatique en robotique

2.5.1 Apprentissage par Renforcement Profond pour le Contrôle de Manipulateurs Robotiques dans des Environnements Simulés

L'apprentissage par renforcement (RL) a émergé comme une approche prometteuse pour développer des capacités de manipulation robotique adaptatives. Cette méthode permet aux robots d'apprendre des stratégies optimales par interaction avec leur environnement, sans nécessiter une modélisation explicite complète du système.[4]

Le principe fondamental du RL repose sur l'apprentissage d'une politique d'action qui maximise une récompense cumulative. Dans le contexte de la manipulation robotique, l'agent (robot) observe l'état de son environnement (positions des objets, configuration des joints), exécute des actions (mouvements des articulations), et reçoit des récompenses basées sur la performance de la tâche. Les avantages du RL pour la manipulation incluent la capacité d'adaptation à des environnements variés, l'apprentissage de stratégies complexes difficiles à programmer explicitement, et la possibilité d'amélioration continue des performances. Cependant, cette approche nécessite généralement de nombreuses interactions avec l'environnement, ce qui peut être coûteux et risqué sur des robots réels, justifiant ainsi l'utilisation d'environnements de simulation pour l'entraînement initial.[5]

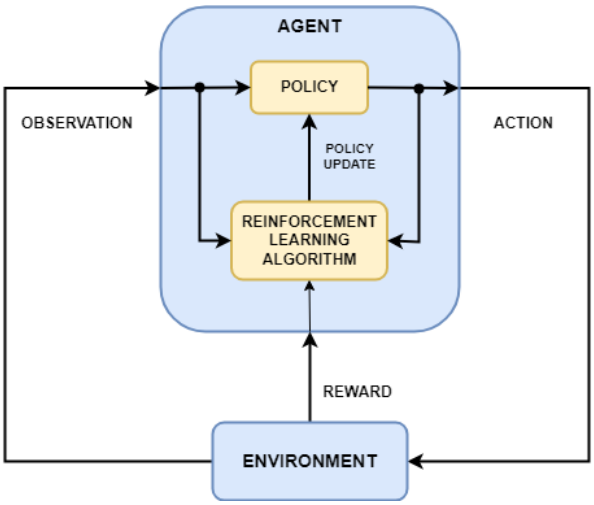

FIGURE 2.4 – Processus d'apprentissage par renforcement (RL) appliqué à la manipulation robotique

2.5.2 L'Émergence de l'Apprentissage par Renforcement Profond (DRL)

Les méthodes classiques d'optimisation, telles que la programmation linéaire ou les approches heuristiques, atteignent souvent leurs limites lorsqu'elles sont confrontées à des environnements complexes, dynamiques ou partiellement connus. Dans ce contexte, l'apprentissage par renforcement profond (Deep Reinforcement Learning – DRL) a émergé comme une alternative prometteuse, en combinant la capacité d'apprentissage adaptatif du renforcement avec la puissance de généralisation des réseaux de neurones profonds.[5]

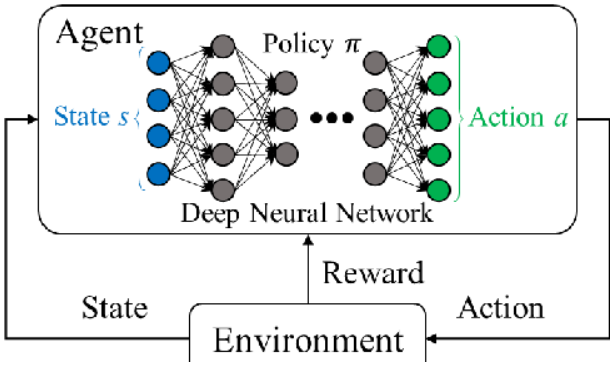

FIGURE 2.5 – Deep Reinforcement Learning (DRL)

Principe général

Le DRL repose sur un mécanisme d'apprentissage dans lequel un agent (par exemple, un robot) interagit avec son environnement afin d'apprendre, par l'expérience, les meilleures décisions à prendre. Contrairement aux approches supervisées, le DRL ne nécessite pas de base de données annotée : l'agent progresse par essais et erreurs, recevant une récompense à chaque action en fonction de son impact sur l'environnement. Son objectif est de maxi-

miser la somme des récompenses obtenues au fil du temps, ce qui conduit à l'apprentissage d'une politique optimale.[6]

Prenons l'exemple d'un robot chargé d'allouer des tâches : la récompense peut être proportionnelle au nombre de tâches accomplies efficacement, au respect des contraintes temporelles, ou encore à l'économie d'énergie réalisée. L'agent ajuste progressivement son comportement en fonction des résultats observés, jusqu'à adopter une stratégie efficace dans la plupart des situations.

Contrairement aux méthodes d'optimisation classiques telles que les contrôleurs PID ou la commande par mode glissant (SMC), qui exigent une modélisation explicite du problème et s'appuient souvent sur des solutions de référence pré-calculées, le DRL permet à l'agent de découvrir de lui-même des stratégies performantes, même dans des contextes où l'espace des solutions est trop vaste ou mal défini. Cette capacité d'adaptation dynamique est particulièrement précieuse dans les systèmes complexes, comme ceux impliquant plusieurs robots, des environnements changeants, ou des incertitudes sur les données Tel qu'illustré par Calderón-Cordova et al dans leur étude.

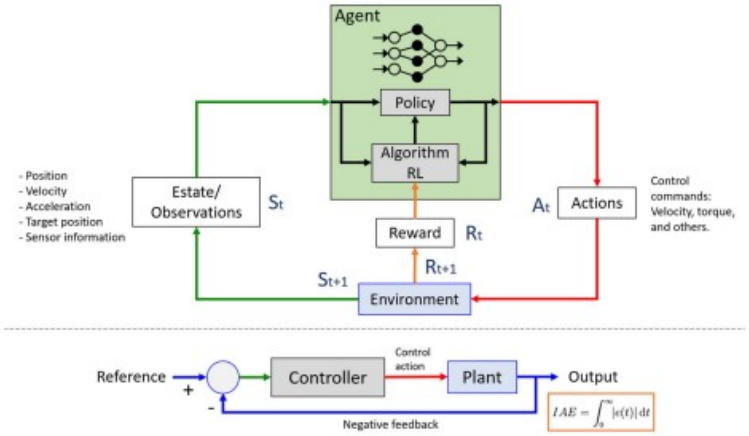

FIGURE 2.6 – Comparaison entre contrôle par DRL et contrôle traditionnel

le fonctionnement de l'apprentissage par renforcement profond (DRL) repose sur une boucle d'apprentissage continue. Dans la partie supérieure de la figure présentée par les auteurs, cette boucle se compose de plusieurs étapes clés : tout d'abord, l'agent (généralement un contrôleur intelligent) observe l'état actuel de l'environnement, noté S_t , à travers différents capteurs (tels que la position, la vitesse, etc.). Sur la base de ces observations, il sélectionne une action A_t , en s'appuyant sur une politique d'action, souvent implémentée sous forme de réseau de neurones. Cette action peut, par exemple, correspondre à l'application d'un couple moteur.[7]

L'environnement, qui inclut à la fois le robot et son espace de travail, réagit alors à cette action, évolue vers un nouvel état S_{t+1} et renvoie une récompense R_{t+1} , laquelle mesure la pertinence de l'action entreprise. Cette récompense constitue un retour d'information que l'algorithme de DRL utilise pour mettre à jour sa politique, dans le but de maximiser la récompense cumulative à long terme.[8]

En comparaison, la partie inférieure de la figure dans l'article illustre une boucle de rétroaction classique, typique des systèmes de contrôle traditionnels. On y retrouve un

contrôleur à logique fixe (comme un régulateur PID) qui agit sur un système physique (la « plante ») afin de minimiser l'écart entre une consigne de référence et la sortie mesurée. Ce tableau établit un parallèle explicite entre les deux approches : l'agent DRL correspond au contrôleur, l'environnement représente la plante, et la récompense tient le rôle des métriques de performance traditionnelles, telles que l'IAE (Intégrale de l'Erreur Absolue).

[5]La principale différence entre ces deux paradigmes réside dans la capacité d'apprentissage : tandis qu'un contrôleur classique suit une logique prédéfinie, l'agent DRL apprend et améliore sa stratégie de contrôle au fil de ses interactions avec l'environnement. Cette faculté adaptative rend le DRL particulièrement efficace dans des environnements complexes, dynamiques ou mal modélisés.

TABLE 2.1 – **Comparison entre RL et control systems.**

Deep Reinforcement Learning	**Traditional control systems**
Action	Control law/actions.
Environment	Plant, reference signals, disturbance, filters, DAC, DCA.
Observation	Any measurable value from the environment such as error signal.
Policy/RL Algorithm	PID controller, FOPID, SMC.
Reward	Performance metrics : cost functions.

Principe général

Rôle des réseaux de neurones profonds

Ce qui distingue véritablement le DRL des formes classiques de reinforcement learning, c'est l'intégration de réseaux de neurones profonds. Ces derniers sont utilisés pour approximer des fonctions complexes — telles que la fonction de valeur (dans le cas du Q-learning), la politique d'action (comme dans PPO), ou une combinaison des deux (dans les modèles Actor-Critic). Grâce à cette approche, l'agent peut généraliser les connaissances acquises à des états non rencontrés durant l'apprentissage, ce qui le rend bien plus robuste et efficace face à des environnements de grande dimension.

2.5.3 L'Écosystème Logiciel pour le DRL en Robotique

[4]

La mise en œuvre pratique du DRL repose sur un écosystème d'outils logiciels qui ont standardisé et accéléré la recherche.

2.5.3.1 Le Rôle Central des Simulateurs

L'entraînement des agents d'apprentissage par renforcement profond (DRL) est un processus exigeant une quantité massive d'interactions avec un environnement. Effectuer cet apprentissage directement sur un robot physique s'avère souvent irréalisable en raison du temps requis, des coûts associés et des risques de dommages matériels. Dans ce contexte, le recours à des simulateurs devient indispensable. Ces environnements virtuels permettent non seulement de générer les grands volumes de données synthétiques nécessaires à l'entraînement, mais aussi de tester et de valider les algorithmes en toute sécurité, sans aucun risque pour le matériel. De plus, ils offrent la possibilité d'accélérer considérablement le temps d'expérimentation par rapport aux essais en temps réel. Plusieurs simulateurs majeurs sont couramment utilisés dans le domaine de la robotique, parmi lesquels CoppeliaSim, Isaac Sim de NVIDIA, Gazebo et MATLAB/Simulink. Le tableau ci-dessous compare leurs caractéristiques détaillées, offrant un aperçu de leurs capacités respectives.

	CoppeliaSim	Isaac Sim	Gazebo	RoboDK	MATLAB
Web Site	[101]	[107]	[104]	[105]	[106]
Latest version	4.6 (10, 2023)	2022.2	11.0.0 (01, 2020)	5.5.4 (02, 2023)	R2023a
License	Free version (students and research), Pro version (purchase)	Individual free of cost, Subscription for some products	Open-source	Trial version (1 month), Pro version (purchase)	Standard, academic, (purchase)
ROS support	ROS1, ROS2	ROS1, ROS2	ROS1, ROS2	No	ROS1, ROS2
Programming language	Lua	Python	C++	Python	M
API	C, C++, Python, Java, Urbi, MATLAB, Octave	Python, ROS, sensors, Robots, etc.	C++	C, C++, Python, MATLAB	C/C++, Fortran, Java, Python, COM components and applications
Robot tools	Pens, paint spray gun, welding torch, gripper	Customizable	Gripper	Grippers, soldering tools, cutting tools, finishing tools, tool changer	Gripper
Sensors	Vision, force, proximity, gyroscopic accelerometers, lasers, lidar	Camera (RGB, depth), LIDAR, IMU, segmentation	Camera, depth, distance, proximity, proximity, force, laser camera	Lasers, cameras, laser trackers	Camera, sonar, LIDAR, GPS, IMU and custom model integration
Robot types	Mobile, Humanoid, Industrial	Humanoid, industrial, mobile	Mobile, Humanoid, Industrial	Industrial robots (commercial robots)	Humanoid, industrial, mobile, customized robots

TABLE 2.2 – **Comparaison des simulateurs pour manipulateurs robotiques**

2.5.3.2 Bibliothèques DRL et Standardisation par Gymnasium

Pour l'implémentation des algorithmes d'apprentissage par renforcement profond, les chercheurs utilisent des bibliothèques de DRL qui fournissent des implémentations testées et fiables, telles que Stable Baselines3, KerasRL ou TF-Agents. Ces dernières offrent des implémentations testées et fiables des principaux algorithmes. Cependant, afin de garantir l'interopérabilité entre la diversité de ces bibliothèques et les multiples environnements de simulation disponibles, une interface standardisée est indispensable. C'est précisément le rôle que remplit l'API Gymnasium, anciennement connue sous le nom d'OpenAI Gym. Gymnasium fournit une interface de programmation simple et universelle qui abstrait les spécificités de l'environnement. Cette abstraction permet de connecter n'importe quel algorithme à n'importe quel simulateur compatible, assurant ainsi une meilleure portabilité du code et une reproductibilité essentielle pour la recherche. L'architecture logicielle typique qui découle de cette approche est illustrée dans la figure ci-dessous.

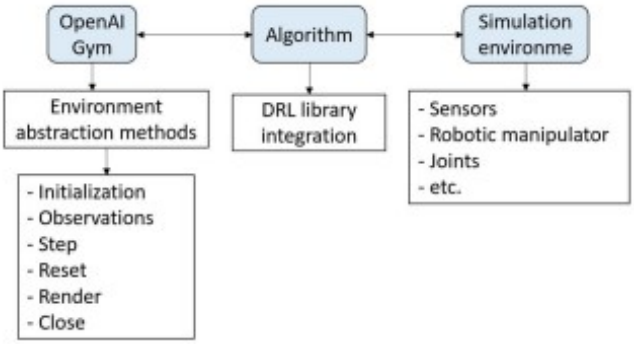

FIGURE 2.7 – Architecture logicielle avec Gymnasium

2.5.4 Approches en DRL et Positionnement de PPO

Le domaine de l'apprentissage par renforcement profond (DRL) englobe une vaste famille d'algorithmes. Une première distinction fondamentale s'opère entre les approches basées sur un modèle (model-based), qui cherchent à apprendre une représentation interne de l'environnement, et les approches sans modèle (model-free). Ces dernières, qui apprennent une stratégie directement par essais et erreurs, sont les plus répandues en robotique. Au sein de cette catégorie, on distingue ensuite les méthodes basées sur la valeur (value-based) comme DQN, qui estiment la valeur de chaque état, de celles basées sur une politique (policy-based) comme PPO qui optimisent directement la stratégie de l'agent. Ces dernières se subdivisent à leur tour en approches on-policy, qui requièrent de nouvelles expériences pour chaque mise à jour, et off-policy, capables de réutiliser des données antérieures pour plus d'efficacité.

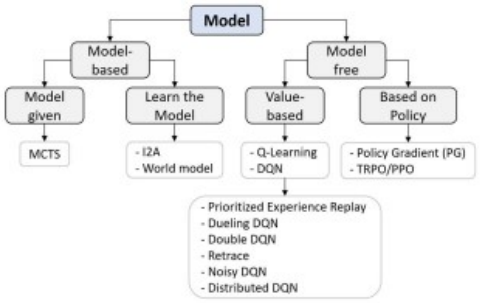

FIGURE 2.8 – Classification DRL : Model-based vs Model-free

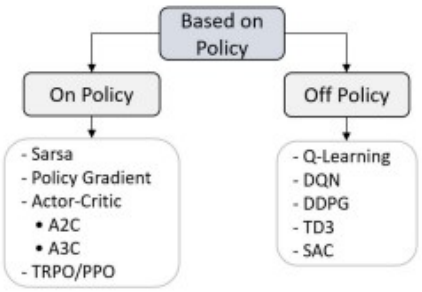

FIGURE 2.9 – Classification des approches basées sur la politique : On-policy vs Off-policy

Dans le cadre de ce projet, notre choix s'est porté sur l'algorithme Proximal Policy Optimization (PPO). Appartenant à la catégorie des algorithmes policy-based et on-policy, PPO est aujourd'hui considéré comme une référence pour les applications robotiques en raison de sa remarquable stabilité d'apprentissage et de sa robustesse. Son innovation principale réside dans un mécanisme qui contraint l'amplitude des mises à jour de la politique à chaque itération. Ce bridage prévient les changements de stratégie trop brusques qui pourraient déstabiliser l'entraînement, faisant de PPO un choix privilégié pour des applications complexes telles que la manipulation robotique.

2.5.4.1 Classification des algorithmes DRL

Ce tableau reproduit ci-dessous, classifie les principaux algorithmes de DRL selon plusieurs critères : l'espace d'action (discret ou continu), et s'ils sont basés sur un modèle, une politique, ou une fonction de valeur.

N	DRL Algorithm	Action space	Model	Policy	Value
1	Q-Learning	Discrete	✗	✗	✓
2	Deep Q-Network (DQN)	Discrete	✗	✗	✓
3	Dueling DQN	Discrete	✗	✗	✓
4	Double DQN	Discrete	✗	✗	✓
5	Distributed DQN	Discrete	✗	✗	✓
6	Sarsa	Discrete	✗	✗	✓
7	Reinforce	Discrete/continue	✗	✓	✗
8	Actor-Critico	Discrete/continue	✗	✓	✗
9	Advantage Actor-Critic (A2C)	Discrete/continue	✗	✓	✓
10	Asynchronous Advantage Actor-Critic (A3C)	Discrete/continue	✗	✓	✓
11	Deep Deterministic Policy Gradient (DDPG)	Continue	✗	✗	✓
12	Twin Delayed DDPG (TD3)	Continue	✗	✗	✓
13	Soft Actor-Critic (SAC)	Discrete/continue	✗	✗	✓
14	Trust Region Policy Optimization (TRPO)	Discrete/continue	✗	✓	✗
15	Proximal Policy Optimization (PPO)	Discrete/continue	✗	✓	✗

TABLE 2.3 – **Classification des algorithmes DRL**

Les algorithmes d'apprentissage par renforcement se classent en trois familles :

- **Basés sur la valeur (ex : DQN)** : Ils évaluent la pertinence de chaque action.[9]
- **Basés sur la politique (ex : PPO)** : Ils apprennent directement une stratégie, ce qui est adapté à la robotique.
- **Acteur-Critique (ex : SAC)** : Ils combinent les deux approches de manière hybride.

2.5.5 Relation avec les travaux existants

Les travaux de référence sur l'optimisation de l'affectation des tâches et de la coordination des mouvements dans les systèmes robotiques autonomes démontrent l'efficacité des approches MILP et heuristiques pour la coordination multi-robots. Ces recherches établissent clairement les avantages et limitations de chaque approche : optimalité garantie mais temps de calcul élevé pour MILP, solutions de qualité acceptable avec efficacité computationnelle pour les heuristiques.

Notre approche se distingue en se concentrant sur l'apprentissage des capacités individuelles comme prérequis à la coordination multi-robots. Alors que les méthodes classiques supposent généralement que les robots possèdent des capacités de manipulation prédéfinies, notre travail explore comment ces capacités peuvent être acquises par apprentissage automatique.

2.5.6 Contribution spécifique

Notre contribution se positionne comme une étape préparatoire essentielle vers l'objectif de coordination multi-robots. En utilisant l'apprentissage par renforcement pour

développer les capacités de manipulation d'un robot individuel, nous établissons les fondations nécessaires pour une coordination future plus sophistiquée.

L'utilisation de l'algorithme PPO dans un environnement de simulation PyBullet offre plusieurs avantages par rapport aux approches traditionnelles. Cette méthode permet au robot d'apprendre des stratégies de manipulation adaptatives sans nécessiter une modélisation explicite complète de la dynamique de manipulation. L'environnement de simulation fournit un cadre sécurisé pour l'exploration extensive des stratégies d'apprentissage.

2.5.7 Perspectives d'intégration

Notre approche d'apprentissage individuel prépare l'intégration future avec les méthodes d'optimisation globale développées pour les systèmes multi-robots. Les patterns d'apprentissage et les stratégies efficaces identifiés dans notre travail pourront informer les algorithmes d'allocation de tâches.

L'extension naturelle de notre travail consiste à intégrer les capacités d'apprentissage développées avec les méthodes MILP et heuristiques pour la coordination multi-robots. Cette intégration pourrait permettre une adaptation dynamique des stratégies d'allocation en fonction des capacités réelles des robots, telles qu'apprises par l'expérience.

La combinaison de l'apprentissage par renforcement pour les capacités individuelles et de l'optimisation combinatoire pour la coordination globale représente une voie prometteuse pour le développement de systèmes robotiques industriels flexibles et efficaces. Cette approche hybride pourrait concilier l'adaptabilité de l'apprentissage automatique avec les garanties de performance des méthodes d'optimisation classiques.

2.5.8 Défis et limitations

Plusieurs défis subsistent pour l'intégration complète de notre approche avec les systèmes multi-robots. Le passage de la simulation à l'implémentation sur robots réels nécessitera une validation extensive et possiblement une adaptation des stratégies apprises. La gestion de l'incertitude et de la variabilité des environnements réels constitue un défi supplémentaire.

L'équilibrage entre l'autonomie d'apprentissage des robots individuels et les contraintes d'optimisation globale du système représente également un défi conceptuel et technique. Les futurs travaux devront explorer comment concilier ces deux aspects pour optimiser les performances globales tout en préservant les capacités d'adaptation individuelles.

Ce chapitre a présenté le contexte théorique dans lequel s'inscrit notre travail. Le chapitre suivant détaillera la méthodologie spécifique développée pour l'apprentissage par renforcement appliqué à la manipulation robotique.

2.6 Conclusion

Ce chapitre a présenté un panorama des approches théoriques et méthodologiques encadrant notre projet d'allocation de tâches pour des bras robotiques. La revue de la littérature a mis en évidence deux axes de recherche principaux et complémentaires. D'une part, les méthodes d'optimisation globale, telles que la programmation linéaire en nombres entiers (MILP) et les approches heuristiques, fournissent des outils puissants pour la coordination et la planification centralisée de systèmes multi-robots. Ces approches, bien que performantes, reposent souvent sur une connaissance a priori des capacités et des temps d'exécution des robots.[10]

D'autre part, l'apprentissage par renforcement profond (DRL) s'est imposé comme l'approche de pointe pour doter un agent unique de capacités d'apprentissage autonomes et adaptatives. En s'appuyant sur un écosystème logiciel mature composé de simulateurs, de bibliothèques standardisées comme Gymnasium et d'algorithmes robustes comme PPO, le DRL permet à un robot d'acquérir par l'expérience des stratégies de manipulation complexes, sans nécessiter de modèle explicite de son environnement.

En synthèse, cet état de l'art valide la démarche de notre projet qui consiste à aborder le défi de la coordination multi-robots de manière progressive. Avant de pouvoir orchestrer efficacement plusieurs agents, il est fondamental de s'assurer que chaque agent possède l'intelligence et la flexibilité nécessaires pour agir de manière pertinente. La focalisation de notre contribution sur l'apprentissage des capacités d'un robot individuel par DRL constitue donc une étape préparatoire logique et solidement fondée sur les pratiques actuelles de la recherche en robotique.

Fort de ce contexte théorique et des validations méthodologiques qu'il apporte, le chapitre suivant s'attachera à décrire en détail la contribution spécifique de ce stage : la conception et l'implémentation de notre solution d'apprentissage pour la manipulation d'objets par un bras robotique.

Chapitre 3

Méthodologie et Mise en Œuvre de l'Agent d'Apprentissage

3.1 Introduction

Les fondements théoriques établis dans le chapitre précédent ont validé l'efficacité de l'apprentissage par renforcement profond (*Deep Reinforcement Learning*, DRL) pour les applications robotiques adaptatives. Le présent chapitre s'attache à transformer ces bases conceptuelles en une implémentation pratique capable de traiter des problématiques concrètes de manipulation robotique.

Notre objectif consiste à élaborer et valider un système d'apprentissage autonome destiné à un manipulateur robotique , lui permettant d'exécuter des opérations de préhension et manipulation d'objets dans un environnement tridimensionnel virtuel. Cette démarche séquentielle, axée sur l'apprentissage individuel, forme le socle nécessaire à l'exploration future des dynamiques de coordination multi-robots. Le succès de cette phase initiale influence directement la faisabilité des approches collaboratives et des mécanismes de distribution de tâches envisagés dans le cadre du programme de recherche global.

Notre cadre méthodologique s'organise autour de trois composants centraux de l'apprentissage par renforcement : l'établissement d'un environnement expérimental immersif exploitant **PyBullet** associé à l'interface normalisée **OpenAI Gym**, le déploiement d'un agent autonome utilisant l'algorithme **Proximal Policy Optimization** (PPO), et la formulation d'un mécanisme de récompense calibré pour diriger l'apprentissage vers les objectifs visés.

La structure de ce chapitre s'articule selon trois dimensions complémentaires :

- **Élaboration de l'environnement virtuel :** examen des choix technologiques retenus, construction de l'espace virtuel incorporant le robot Kuka et les éléments à manipuler, définition des espaces d'observation et d'action régissant les interactions agent-environnement.
- **Architecture de l'agent intelligent :** validation du choix algorithmique PPO, développement de l'architecture neuronale constituant le substrat cognitif de l'agent, élaboration mathématique du système de récompense.

- **Déploiement du processus d'apprentissage :** calibrage des hyperparamètres, établissement du protocole d'entraînement et implémentation d'outils de supervision pour évaluer la progression des performances en temps réel.

3.2 Conception de l'Environnement de Simulation

L'établissement d'un environnement de simulation adapté constitue la phase fondamentale de notre approche. Cet espace numérique fonctionne comme un laboratoire virtuel permettant à l'agent d'expérimenter, d'interagir et de développer ses capacités dans un cadre sécurisé. La conception d'un environnement efficace requiert un équilibre délicat entre fidélité physique et efficience computationnelle : il doit reproduire avec précision les phénomènes physiques pertinents tout en autorisant l'exécution de milliers de cycles d'apprentissage dans des délais raisonnables.

3.2.1 Infrastructure Technologique

Le choix de l'infrastructure logicielle repose sur les standards reconnus dans les domaines de la robotique et de l'apprentissage automatique. Notre architecture intègre deux bibliothèques Python complémentaires, réputées pour leur fiabilité :

- **PyBullet** constitue une interface Python sophistiquée pour le moteur physique Bullet Physics SDK, une bibliothèque C++ haute performance développée initialement pour l'industrie du divertissement numérique et des effets visuels. Cette solution libre d'accès s'est établie comme référence dans la communauté scientifique par sa capacité à rendre accessibles des simulations physiques avancées tout en préservant des performances computationnelles remarquables.

  L'infrastructure de PyBullet s'appuie sur un moteur de calcul optimisé apte à gérer simultanément les dynamiques multi-corps, les mécanismes de contact et les phénomènes de collision avec une exactitude notable. Cette solidité technique s'avère essentielle pour les applications de manipulation robotique où la fidélité simulatoire détermine la transférabilité des apprentissages vers l'environnement réel. La bibliothèque se distingue particulièrement dans la gestion des interactions complexes entre objets déformables et rigides, ainsi que dans la modélisation des forces de friction et des contraintes articulaires.

  Concernant l'intégration technologique, PyBullet bénéficie d'une compatibilité native avec l'écosystème Python scientifique, simplifiant considérablement le prototypage agile et l'expérimentation itérative. Cette facilité d'accès technique, associée à une documentation complète, en fait un instrument de choix pour la recherche en apprentissage par renforcement où la vélocité de développement et la facilité de débogage représentent des facteurs déterminants.[11]

  La flexibilité de PyBullet se révèle également dans sa capacité à traiter plusieurs formats de description robotique normalisés. La prise en charge du format URDF (*Unified Robot Description Format*) autorise l'importation directe de modèles de

manipulateurs industriels comme les bras Franka Panda, Universal Robots UR5, ou Kuka IIWA. L'extension aux formats SDF (*Simulation Description Format*) et MJCF (*MuJoCo XML Format*) enrichit cette compatibilité, permettant l'exploitation de bibliothèques de modèles existantes et favorisant la collaboration entre équipes de recherche utilisant divers outils de simulation.

- **OpenAI Gym (actuellement Gymnasium)** garantit l'intégration avec l'écosystème des algorithmes de DRL. L'utilisation de la classe `Env` comme classe de base assure la compatibilité avec les implémentations standardisées. Cette interface unifiée, construite autour des méthodes essentielles `reset` et `step`, simplifie l'emploi des algorithmes proposés par la bibliothèque Stable Baselines3.[12]

FIGURE 3.1 – Outils technologiques de l'environnement de simulation.

3.2.2 Architecture de la Scène Virtuelle

L'espace de simulation, configuré par la méthode `reset()` de la classe `Kuka7DOFEnv`, reproduit un contexte de manipulation d'objets épuré mais représentatif. Les éléments constituant la scène sont spécifiés par leurs fichiers descriptifs URDF (*Universal Robot Description Format*) et incluent :

1. Une surface de référence (`plane.urdf`) délimitant l'espace opérationnel.
2. Une surface de travail (`table/table.urdf`) supportant les manipulations.
3. Le manipulateur Kuka IIWA 7DOF (`kuka_iiwa/model.urdf`), intégré comme système polyarticulé à base immobilisée.
4. L'élément à manipuler, représenté par un cube de dimensions réduites (`cube_small.urdf`).

Afin de stimuler l'apprentissage de stratégies adaptatives, la localisation de l'objet cible est modifiée aléatoirement à l'initialisation de chaque épisode d'entraînement. Cette variabilité spatiale, appliquée aux coordonnées (x, y, z) dans une zone délimitée de la surface de travail, oblige le robot à élaborer des capacités d'adaptation plutôt que de mémoriser des séquences gestuelles figées.[**obbezoo2022Robotic**]

3.2.3 Le Robot Kuka LBR iiwa : Caractéristiques et Configurations

[13]La sélection du matériel robotique constitue un élément structurant de tout projet de manipulation. Nos expérimentations s'appuient sur la modélisation du **Kuka LBR**

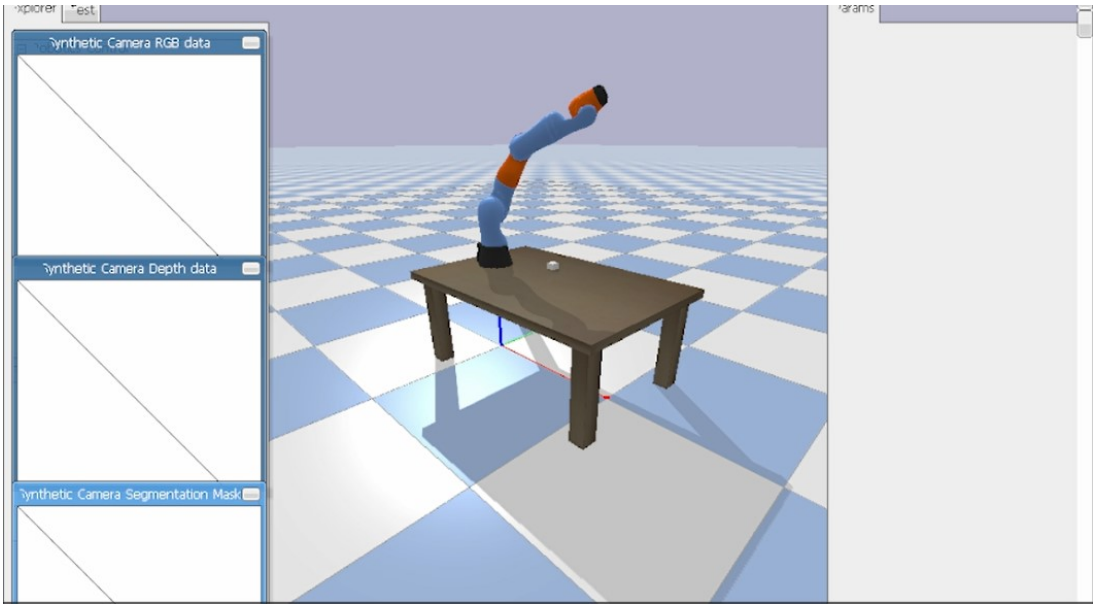

FIGURE 3.2 – Configuration de l'environnement de simulation PyBullet présentant le manipulateur Kuka, l'espace de travail et l'objet cible.

iiwa 14 R820. La dénomination "LBR iiwa" signifie *Leichtbauroboter intelligent industrial work assistant* (assistant industriel intelligent robot léger), soulignant sa nature de **robot collaboratif**, ou "cobot".

La particularité principale de la gamme iiwa réside dans sa conception dédiée à la collaboration homme-robot sécurisée. Cette fonctionnalité s'appuie sur l'intégration de capteurs de couple articulaire dans chacun de ses sept axes. Cette sensibilité lui permet de percevoir des contacts, même légers, et d'ajuster son comportement en conséquence, le rendant approprié pour des tâches d'assemblage délicates et des interactions dans des espaces partagés.

FIGURE 3.3 – Le robot collaboratif Kuka LBR iiwa.

Les spécifications techniques de ce manipulateur le rendent particulièrement approprié à nos travaux. Ses **sept degrés de liberté (DOF)** lui procurent une dextérité cinématique élevée, comparable à celle d'un membre supérieur humain. Cette redondance

lui permet d'atteindre une position et orientation définies avec une multitude de configurations articulaires possibles, constituant un avantage majeur pour la planification de trajectoires complexes et l'évitement d'obstacles. Les caractéristiques techniques principales du modèle utilisé sont synthétisées dans un tableau.

Caractéristique	**Valeur**
Modèle	LBR iiwa 14 R820
Nombre d'axes	7
Charge utile nominale	14 kg
Portée maximale	820 mm
Répétabilité de position (ISO 9283)	± 0.15 mm
Poids du robot	env. 29.9 kg
Contrôleur	KUKA Sunrise Cabinet

TABLE 3.1 – Spécifications techniques du Kuka LBR iiwa 14 R820.

Le choix de ce manipulateur pour notre projet présente plusieurs justifications. Sa dextérité exceptionnelle (7 DOF) en fait une plateforme optimale pour l'apprentissage de tâches de manipulation sophistiquées. En outre, ses caractéristiques collaboratives et sa présence significative dans les applications de l'Industrie 4.0 confèrent aux recherches menées sur cette plateforme une pertinence particulière.

3.3 Formalisation des Espaces d'Interaction

La spécification rigoureuse des espaces d'observation et d'action représente un prérequis pour l'interfaçage avec les algorithmes d'apprentissage, en conformité avec les spécifications de l'interface Gym.

Espace d'Observation

L'espace d'observation encode l'ensemble des informations sensorielles disponibles à l'agent pour la prise de décision. Il se structure comme un vecteur de 20 dimensions en valeurs continues, combinant des données proprioceptives et extéroceptives. Le tableau **??** détaille la composition de cette représentation.

Type d'Information	**Dimension**	**Contenu**
États articulaires	7	Angles de chaque articulation (radians).
Vitesses articulaires	7	Vitesses angulaires instantanées (rad/s).
Localisation objet cible	3	Coordonnées spatiales (x, y, z) du cube.
Position effecteur final	3	Coordonnées spatiales (x, y, z) de l'extrémité du bras.
Dimension totale	**20**	**Taille complète du vecteur d'état.**

TABLE 3.2 – Composition du vecteur d'observation.

Espace d'Action

L'espace d'action définit l'ensemble des commandes transmissibles au robot. Notre implémentation exploite un contrôle positionnel direct des 7 degrés de liberté articulaires, comme spécifié dans ce tableau.

Type de Commande	Dimension	Format	Domaine	Interprétation
Consignes positionnelles	7	Continu	[-1, 1]	Commandes normalisées par articulation.

TABLE 3.3 – Spécification du vecteur d'action.

Le choix d'un espace d'action normalisé dans le domaine [-1, 1] est une pratique standard, essentielle pour garantir la stabilité de l'apprentissage des réseaux de neurones. Cette approche permet à l'agent d'apprendre une stratégie générale, où les actions représentent un pourcentage de l'effort possible pour chaque articulation. Cela simplifie l'implémentation tout en assurant que les commandes envoyées au robot respectent systématiquement ses limites physiques et sécuritaires.

3.4 Approche par Apprentissage par Renforcement (DRL)

L'élaboration de l'environnement simulé étant achevée, l'étape subséquente porte sur la conception de l'entité intelligente capable d'y évoluer et d'apprendre. Notre méthodologie s'appuie sur les techniques d'apprentissage par renforcement profond (DRL), paradigme dans lequel un agent développe une stratégie comportementale optimale par l'intermédiaire d'un réseau neuronal, orienté par un mécanisme de récompense.[14]

3.4.1 Structure et Mécanismes de l'Agent PPO

L'implémentation PPO adoptée exploite une configuration **Acteur-Critique**, architecture duale illustrée dans la figure 3.4. Cette approche segmente l'intelligence artificielle en deux composants neuronaux spécialisés et complémentaires.[15]

L'architecture se fonde sur la coopération entre deux entités distinctes : le module **Acteur** génère les décisions comportementales (π_θ) à partir de l'analyse de l'état environnemental (s_t), pendant que le module **Critique** quantifie la valeur de cet état ($V(s_t)$) en estimant les bénéfices futurs anticipés.[16]

Le cycle d'apprentissage se décompose en cinq phases successives :

1. **Acquisition expérientielle :** L'agent établit des interactions environnementales (exécution d'action a_t depuis l'état s_t) et stocke ces séquences transitionnelles dans une mémoire tampon.
2. **Sélection par lots :** Extraction d'échantillons regroupés pour l'entraînement des deux composants neuronaux.
3. **Quantification de l'avantage :** Le Critique évalue les états pour déterminer l'indicateur d'avantage $A(s_t, a_t)$, mesurant l'efficacité relative de l'action réalisée.

4. **Raffinement PPO :** L'Acteur modifie sa stratégie comportementale par l'intermédiaire du ratio $r_t(\theta)$ contraint, appliquant l'objectif $L^{\text{CLIP}}(\theta)$ pour éviter les changements drastiques.
5. **Perfectionnement du Critique :** Réduction de l'erreur prédictive via l'optimisation de la fonction coût L^V.

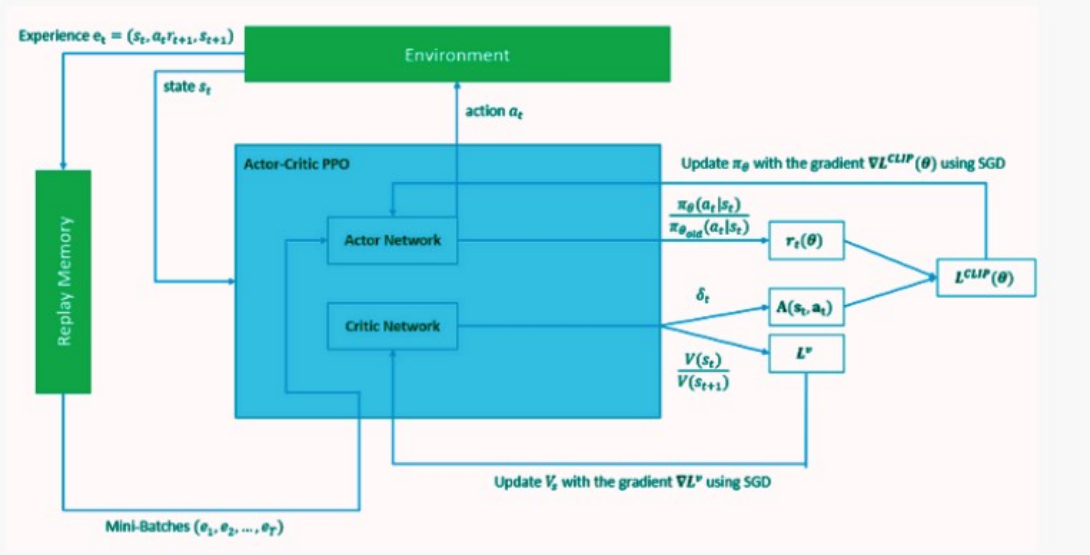

FIGURE 3.4 – Architecture de l'algorithme PPO.

La formalisation mathématique du processus repose sur plusieurs relations fondamentales :

Module Acteur et Politique Stochastique (π_θ) Le réseau acteur, caractérisé par les paramètres θ, transforme un état d'entrée s_t en action a_t conformément à la politique probabiliste $\pi_\theta(a_t|s_t)$.

Module Critique et Estimation de Valeur (V_ϕ) Le réseau critique, défini par les paramètres ϕ, approxime la fonction de valeur $V_\phi(s_t)$, représentant l'espérance de retour cumulé depuis l'état considéré.

Détermination de l'Avantage (A_t) L'évaluation de l'Acteur repose sur l'estimation de la fonction d'avantage, généralement via l'erreur de différence temporelle tδ_t :

$$\delta_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t) \tag{3.1}$$

avec r_t désignant la récompense immédiate et γ le coefficient d'actualisation. L'estimation d'avantage A_t dérive généralement de cette mesure δ_t.

Actualisation Acteur par Objectif Contraint Ce mécanisme constitue l'innovation centrale de PPO. Le ratio probabiliste entre politiques se définit comme :

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \tag{3.2}$$

La fonction objectif d'optimisation de l'Acteur s'exprime par :

$$\mathcal{L}^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] \tag{3.3}$$

où $\hat{A}_t$ représente l'estimateur d'avantage et ϵ l'hyperparamètre de limitation (typiquement 0.2). Cette formulation confine les modifications politiques dans un intervalle de confiance, préservant la stabilité d'apprentissage.

Optimisation du Critique L'amélioration du Critique procède par minimisation d'une fonction de perte $\mathcal{L}^V$, généralement l'erreur quadratique :

$$\mathcal{L}^V(\phi) = \hat{\mathbb{E}}_t \left[(V_\phi(s_t) - V_t^{\text{target}})^2 \right] \tag{3.4}$$

L'optimisation conjointe de ces deux fonctions objectives par descente de gradient stochastique confère à l'agent des capacités d'apprentissage à la fois robustes et efficaces.

3.4.2 Architecture du Réseau de Neurones utilisee

Le cerveau de notre agent est un réseau de neurones. Nous avons utilisé l'architecture par défaut de Stable Baselines3 pour PPO, la `MlpPolicy`. Il s'agit d'un **Perceptron Multi-Couches (MLP)**, une architecture standard et efficace pour traiter des données vectorielles comme notre vecteur d'observation.

Son fonctionnement est le suivant :

1. La **couche d'entrée** réceptionne le vecteur d'observation de 20 dimensions, représentant l'état complet du système robotique et de son environnement.

2. Les **couches intermédiaires** (typiquement deux couches densément connectées de 64 ou 256 neurones) transforment progressivement ces données brutes en représentations abstraites, extrayant les patterns comportementaux significatifs.

3. La **couche terminale** génère le vecteur de commande de 7 dimensions, correspondant aux consignes positionnelles destinées aux articulations du manipulateur.

Cette architecture séquentielle permet une transformation graduelle de l'information sensorielle en décisions motrices, mimant les processus de traitement observés dans les systèmes nerveux biologiques.[17]

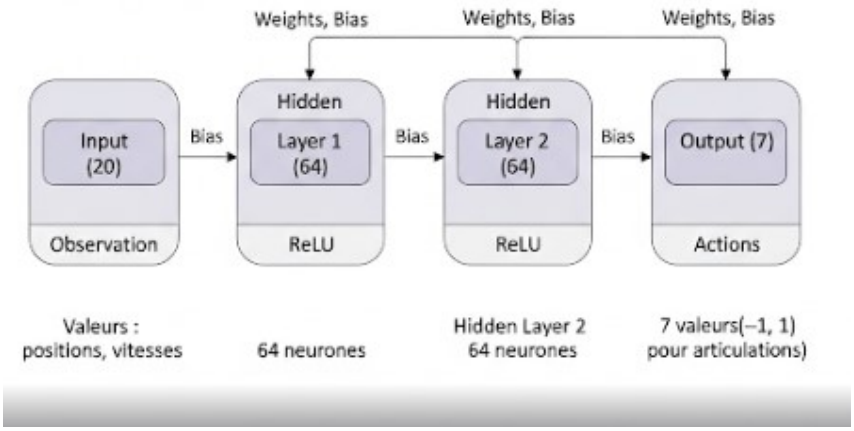

FIGURE 3.5 – Architecture du réseau de neurones MLP utilisé par l'agent PPO.

3.4.3 Ingénierie de la Fonction de Récompense

La fonction de récompense est sans doute la composante la plus critique de notre méthodologie. C'est elle qui traduit l'objectif de haut niveau – « atteindre l'objet » – en un signal d'apprentissage tangible pour l'agent. Après des expérimentations initiales avec une fonction simple, nous avons développé une stratégie de récompense plus élaborée et directive, conçue pour accélérer la convergence et encourager des comportements plus précis et efficaces. Notre approche finale repose sur la combinaison de trois mécanismes distincts.

Une pénalité de distance amplifiée. Le fondement de la récompense reste un signal dense basé sur la distance euclidienne entre l'effecteur et la cible. Cependant, pour fournir un gradient d'apprentissage plus fort, surtout lorsque l'agent est loin de l'objectif, cette pénalité a été amplifiée par un facteur 10.

$$R_{\text{base}}(s) = -10 \times \|p_{\text{effecteur}} - p_{\text{objet}}\|_2 \tag{3.5}$$

Un système de bonus progressifs par paliers. Pour lutter contre les problèmes de récompenses creuses et guider plus explicitement l'agent, nous avons implémenté un système de bonus par paliers. L'agent reçoit des récompenses positives cumulatives à mesure qu'il franchit des seuils de proximité de plus en plus stricts. Cette structure en « escalier » crée des sous-objectifs clairs qui encouragent fortement le robot à ne pas se contenter de s'approcher, mais à viser la plus grande précision possible. Les bonus sont définis comme suit :

- Un bonus de +**40** si la distance est inférieure à 1.0 m.
- Un bonus additionnel de +**60** si la distance est inférieure à 0.7 m.
- Et ainsi de suite, avec des bonus de plus en plus importants pour des distances de 0.5 m, 0.3 m, 0.2 m, jusqu'à un bonus exceptionnel de +**400** pour une précision inférieure à 5 cm.

Un bonus temporel pour encourager l'efficacité. Enfin, pour non seulement récompenser la réussite mais aussi l'efficience, un bonus temporel est accordé uniquement lorsque l'agent atteint la cible (distance < 5 cm). Ce bonus est inversement proportionnel au nombre de pas de temps utilisés, encourageant ainsi l'agent à trouver les trajectoires les plus courtes.

$$R_{\text{temps}}(s) = \begin{cases} (N_{\max} - N_{\text{actuel}}) \times 2.0 & \text{si l'épisode est un succès} \\ 0 & \text{sinon} \end{cases} \tag{3.6}$$

où $N_{\max}$ est le nombre maximal de pas (200) et N_{actuel} est le nombre de pas écoulés.

La récompense totale à chaque étape est donc la somme de ces trois composantes, créant un signal d'apprentissage riche qui favorise à la fois la précision, la rapidité et la convergence.

3.5 Implémentation de l'Entraînement

La mise en œuvre du processus d'entraînement s'appuie sur un ensemble de bibliothèques Python open-source qui interagissent pour orchestrer l'apprentissage de l'agent. Cette section présente les outils clés utilisés pour l'implémentation de l'algorithme PPO et le suivi de ses performances.

3.5.1 Framework d'Apprentissage par Renforcement

[18]L'implémentation algorithmique repose sur **Stable Baselines3 (SB3)**, plateforme logicielle de référence construite sur PyTorch et dédiée aux techniques d'apprentissage par renforcement profond. Cette bibliothèque propose des algorithmes validés scientifiquement et optimisés pour les applications de recherche.

L'adoption de SB3 présente plusieurs avantages stratégiques majeurs :

- **Validation scientifique :** Les implémentations SB3 bénéficient d'une validation rigoureuse par la communauté de recherche internationale. La conformité aux spécifications des publications originales garantit l'intégrité méthodologique de notre approche.
- **Interface intuitive :** L'architecture logicielle, inspirée des standards Scikit-learn, propose une interface de programmation épurée. Les fonctions principales `model.learn()`, `model.predict()` et `model.save()`/`load()` permettent une concentration optimale sur la conception expérimentale plutôt que sur les détails d'implémentation algorithmique.
- **Compatibilité native :** L'intégration directe avec l'écosystème **Gymnasium** facilite la connexion transparente avec notre environnement personnalisé `Kuka7DOFEnv`.
- **Mécanismes d'extension :** Le système de *callbacks* intégré permet l'implémentation de fonctionnalités personnalisées pour le monitoring (`TrainingMonitorCallback`) et la sauvegarde (`CheckpointCallback`), aspects détaillés ultérieurement.[19]

3.5.2 Écosystème Logiciel Complémentaire

L'infrastructure d'entraînement s'enrichit de bibliothèques spécialisées assurant des fonctionnalités critiques :

- **NumPy :** Cette bibliothèque constitue le socle computationnel pour l'ensemble des opérations matricielles et vectorielles. Son utilisation s'étend de la manipulation des vecteurs d'état et d'action jusqu'aux calculs mathématiques internes de l'environnement de simulation.

- **Matplotlib :** Moteur graphique alimentant notre système de visualisation temps réel. La classe `PerformanceTracker` exploite ses capacités pour générer et actualiser dynamiquement les représentations graphiques des métriques d'apprentissage (récompenses cumulées, distances euclidennes, convergence comportementale).

3.5.2.1 Configuration des Hyperparamètres du Modèle

Les hyperparamètres sont des variables de configuration définies avant le début de l'entraînement. Leur réglage fin est essentiel pour obtenir une convergence stable et des performances optimales. Pour ce projet, nous avons utilisé un ensemble de valeurs standards pour l'algorithme PPO, reconnues pour leur efficacité sur une large gamme de problèmes de contrôle. Ces paramètres sont résumés dans ce tableau.[20]

L'entraînement a été lancé pour une durée totale de **50 000** ***timesteps*** (étapes d'interaction agent-environnement). Cette durée a été choisie comme un compromis pragmatique entre l'obtention de résultats significatifs et les contraintes de temps de calcul de notre environnement de travail.

Hyperparamètre	Valeur	Description
`policy`	`MlpPolicy`	Architecture du réseau de neurones (Perceptron Multi-Couches), adaptée aux données vectorielles.
`learning_rate`	`3e-4`	Taux d'apprentissage de l'optimiseur Adam.
`n_steps`	`2048`	Nombre d'étapes collectées par l'agent avant chaque mise à jour de la politique.
`batch_size`	`128`	Taille des mini-lots de données utilisés lors de la phase d'optimisation.
`n_epochs`	`10`	Nombre de passages sur les données collectées à chaque mise à jour.
`gamma`	`0.99`	Facteur d'actualisation, incitant l'agent à privilégier les récompenses à long terme.
`clip_range`	`0.2`	Paramètre de rognage de PPO, limitant l'ampleur des mises à jour pour la stabilité.
`ent_coef`	`0.0`	Coefficient de la pénalité d'entropie (non utilisé pour favoriser l'exploitation).

TABLE 3.4 – Hyperparamètres utilisés pour l'entraînement de l'agent PPO.

Le concept d'épisode dans l'entraînement.
En apprentissage par renforcement, l'entraînement est structuré en épisodes. Un épisode peut être défini comme une tentative complète et isolée de la part de l'agent pour résoudre la tâche qui lui est assignée, depuis un état initial jusqu'à un état terminal.
Dans le contexte de notre projet, un épisode représente un essai complet du robot Kuka pour atteindre le cube cible. Le déroulement de chaque épisode est régi par les mécanismes implémentés dans notre environnement :

Début de l'épisode (méthode `reset()`) Un nouvel épisode commence par une réinitialisation de l'environnement. Cette opération replace le robot à sa position initiale et, point crucial, positionne l'objet cible à des coordonnées **aléatoires** sur la table de travail. Cette randomisation est fondamentale pour forcer l'agent à apprendre une politique de mouvement robuste et généralisable.

Fin de l'épisode (condition terminale `done`) L'épisode se poursuit jusqu'à ce que l'une des deux conditions terminales suivantes soit remplie :

- **La Réussite :** L'épisode se termine avec succès si la distance euclidienne entre l'effecteur final du robot et le centre du cube est inférieure au seuil $\epsilon = 0.15$ m.
- **L'Échec par Dépassement de Temps :** L'épisode se termine si le nombre d'étapes d'action dépasse le maximum autorisé, fixé à 150 pas dans notre configuration. Cette condition est une sécurité qui prévient les boucles infinies et assure la progression de l'entraînement.

L'entraînement global sur 50 000 *timesteps* est donc une succession de plusieurs centaines de ces épisodes. L'algorithme PPO utilise les trajectoires complètes (états, actions, récompenses) de chaque épisode pour évaluer et améliorer itérativement sa politique.

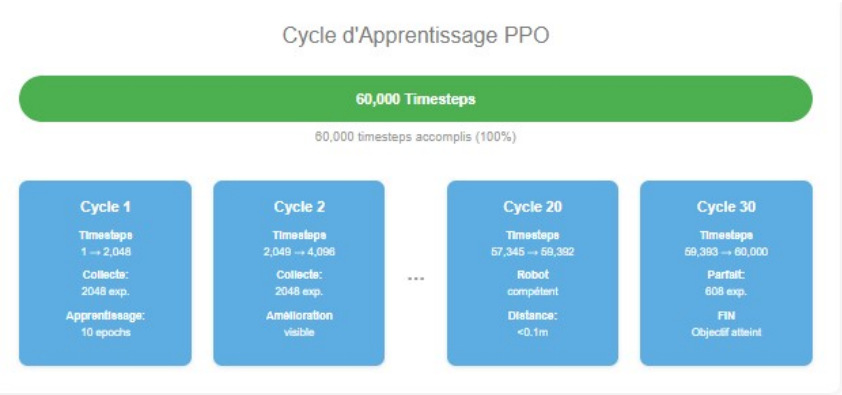

FIGURE 3.6 – Les Cycle d'Entraînement Itératif de l'Algorithme PPO

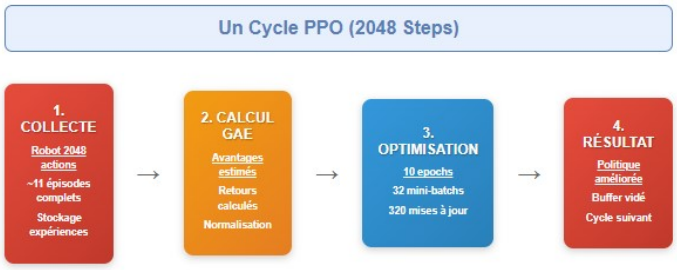

FIGURE 3.7 – un Cycle d'Entraînement pendant 2048 steps

3.5.3 Suivi et Sauvegarde de l'Entraînement

Entraîner un agent DRL peut être un processus long et opaque. Il est donc crucial de mettre en place des outils de suivi pour évaluer la progression en temps réel et pour sauvegarder le travail. Pour cela, nous avons utilisé deux *callbacks* de Stable Baselines3 :

CheckpointCallback Ce callback a été configuré pour sauvegarder automatiquement une copie complète du modèle tous les 10 000 *timesteps* . Cette pratique est essentielle pour se prémunir contre d'éventuelles interruptions et pour pouvoir, si nécessaire, reprendre l'entraînement à partir d'un point de contrôle antérieur.

TrainingMonitorCallback Nous avons développé ce *callback* personnalisé pour obtenir un suivi fin et visuel de l'apprentissage. Il utilise notre classe `PerformanceTracker` pour enregistrer, à la fin de chaque épisode, les métriques clés de performance (récompense totale, distance finale, durée et succès). Ces données sont ensuite utilisées pour tracer et mettre à jour un graphique de performance en temps réel.

L'entraînement s'est terminé après un total de 390 épisodes.

Pipeline global d'Entraînement

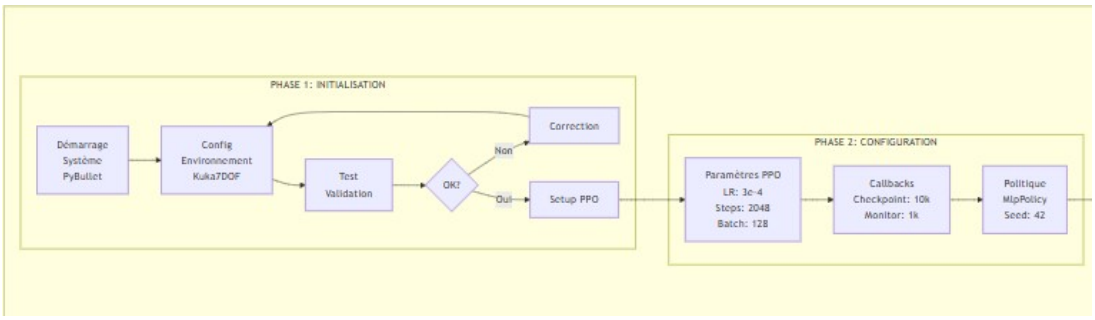

FIGURE 3.8 – La phase 1 et 2

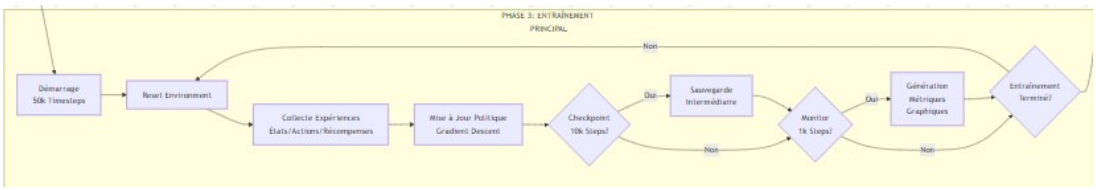

FIGURE 3.9 – La phase 3

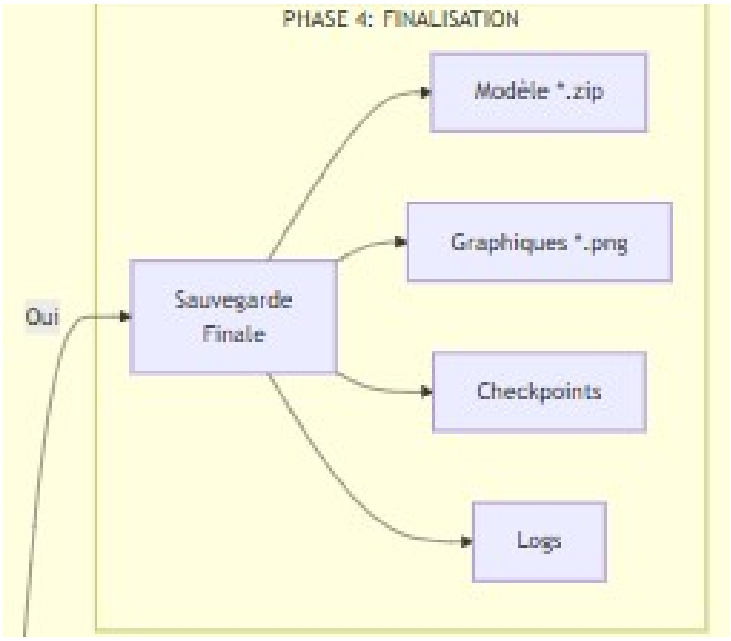

FIGURE 3.10 – La phase 4

3.5.4 Analyse des Performances d'Entraînement

Suivi des Métriques Internes avec TensorBoard. Pour un suivi plus approfondi des dynamiques internes de l'algorithme PPO, nous avons utilisé **TensorBoard**. Il s'agit d'une suite d'outils de visualisation développée par Google, devenue un standard pour le monitoring d'expériences en apprentissage automatique. En ajoutant le paramètre `tensorboard_log` lors de l'initialisation de notre modèle dans Stable Baselines3, nous avons activé la journalisation automatique de nombreuses métriques.

Durant l'entraînement, l'algorithme écrit des données dans un dossier de logs dédié. En lançant le serveur TensorBoard, nous accédons à un tableau de bord web interactif. Ce dernier nous a permis de visualiser en temps réel l'évolution de métriques internes cruciales qui ne sont pas visibles autrement, telles que la perte de la politique (*policy_loss*), la perte de la fonction de valeur (*value_loss*) et l'entropie de la politique (*entropy_loss*). L'analyse de ces courbes est fondamentale pour diagnostiquer la convergence et la stabilité de l'apprentissage de l'agent.

L'analyse des courbes de performance générées durant l'entraînement est essentielle pour comprendre la dynamique d'apprentissage de l'agent. Les blocs suivants présentent l'évolution des métriques clés, où chaque figure est directement suivie de son interprétation.

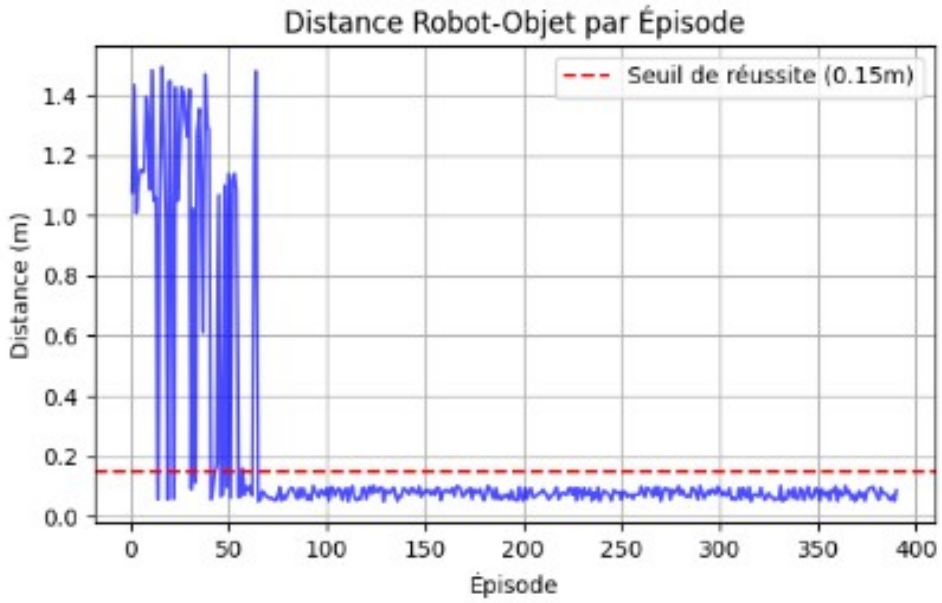

FIGURE 3.11 – Évolution de la distance finale entre l'effecteur et la cible.

Ce graphique illustre l'évolution de la distance finale entre l'effecteur du robot et l'objet cible pour chaque épisode d'entraînement. La ligne rouge horizontale matérialise le seuil de réussite fixé à 0.15 mètres. On observe une diminution progressive et significative de cette distance au cours de l'entraînement, passant d'environ 1.4 mètres lors des premiers épisodes à des valeurs majoritairement inférieures au seuil de réussite après l'épisode 100. Cette convergence démontre l'efficacité de l'algorithme PPO dans l'apprentissage de la tâche de manipulation robotique.

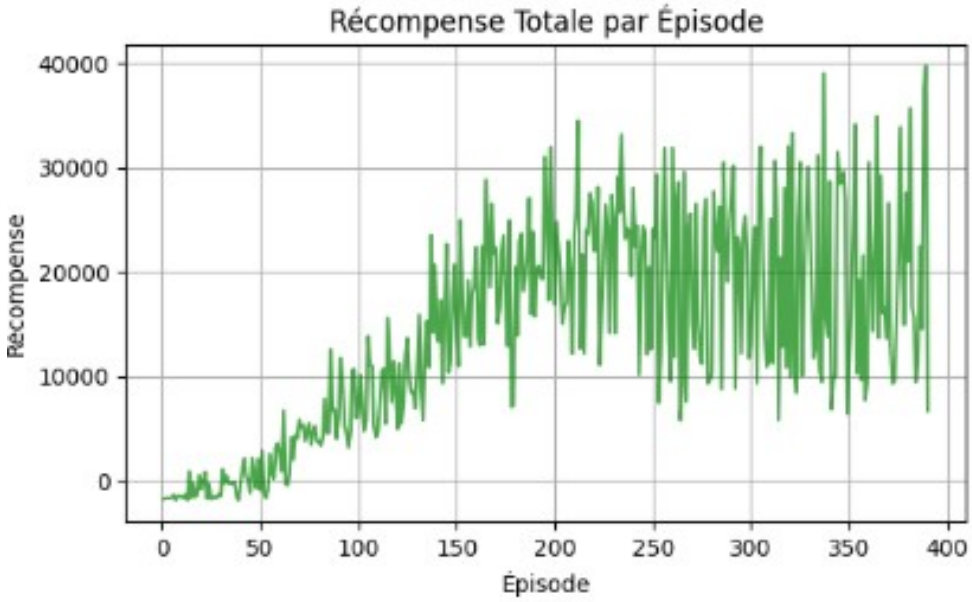

FIGURE 3.12 – Évolution de la récompense cumulative par épisode.

L'évolution des récompenses cumulatives par épisode révèle une progression constante et positive de la performance de l'agent. Les récompenses augmentent de manière quasi-linéaire, passant de valeurs proches de zéro en début d'entraînement à des pics dépassant 40,000 points vers la fin.reflétant une amélioration continue de sa capacité à atteindre l'objectif de manipulation avec précision et efficacité.

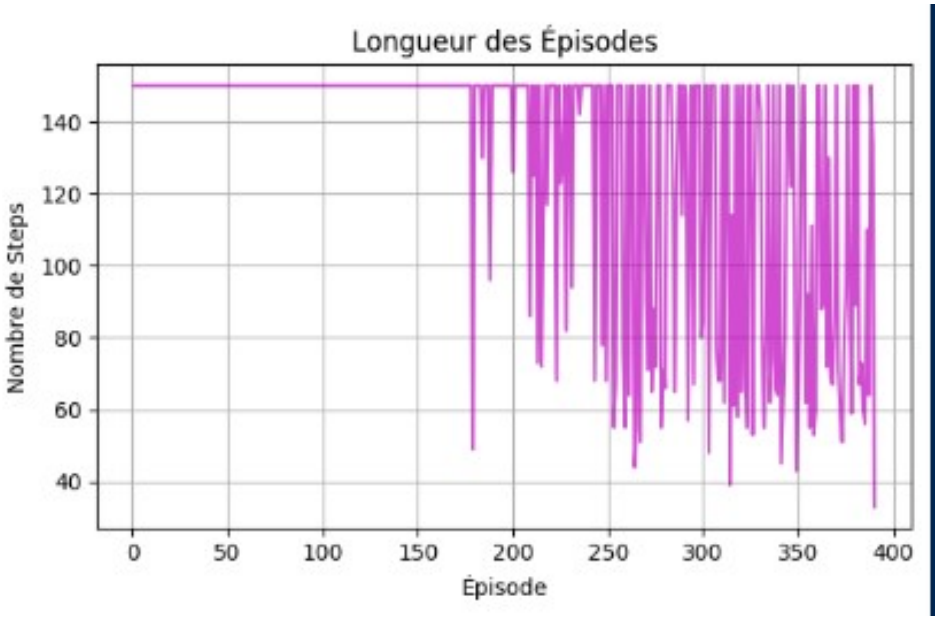

FIGURE 3.13 – Évolution de la durée des épisodes en nombre de pas de temps.

La distribution de la longueur des épisodes montre une stabilisation progressive autour de 140-150 steps par épisode. Initialement variable, la durée des épisodes tend vers une valeur constante correspondant au nombre maximum de steps autorisés, indiquant que l'agent explore pleinement l'espace d'actions disponible. Les variations observées reflètent les différentes stratégies explorées par l'agent, avec une tendance vers l'utilisation complète du temps imparti pour optimiser la précision de manipulation.

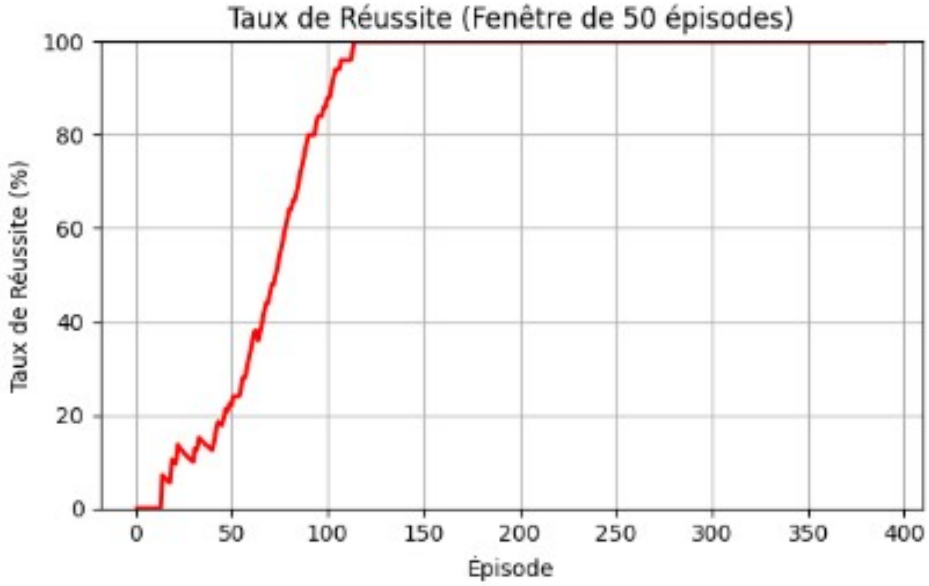

FIGURE 3.14 – Taux de Réussite

Le taux de réussite, calculé sur une fenêtre glissante de 50 épisodes, présente une courbe d'apprentissage caractéristique en forme de S. Après une phase initiale de faible performance (0-20)%, on observe une amélioration rapide entre les épisodes 50 et 150, atteignant un plateau de performance stable autour de 95-100% de réussite. Cette métrique confirme la robustesse et la fiabilité de la politique apprise, démontrant que l'agent maintient des performances élevées de manière consistante.

3.6 Conclusion

Ce chapitre a détaillé la démarche méthodologique complète mise en œuvre pour entraîner et évaluer un agent d'apprentissage par renforcement dans une tâche de manipulation robotique. En partant de la construction d'un environnement de simulation robuste et reproductible avec **PyBullet** et **Gymnasium**, nous avons défini un cadre expérimental solide. Le choix de l'algorithme **PPO**, justifié par sa stabilité et son efficacité, a été au cœur de la conception de notre agent. L'ingénierie d'une fonction de récompense hybride, combinant un guidage dense et un objectif clair, a été une étape cruciale pour orienter l'apprentissage.

Enfin, nous avons établi un protocole d'entraînement rigoureux, avec des hyperparamètres définis et un système de suivi des performances.

Fort de ce cadre méthodologique, nous sommes maintenant en mesure de présenter et d'analyser les résultats obtenus. Le chapitre suivant se consacrera donc à l'étude des performances de l'agent entraîné, en interprétant les courbes d'apprentissage et les résultats des tests finaux pour répondre à notre problématique initiale.

Chapitre 4

Résultats et discussion

4.1 Protocole d'Évaluation

Une fois l'agent entraîné, il est indispensable d'évaluer ses performances de manière objective et reproductible. Cette phase de test permet de quantifier la qualité de la politique apprise et de vérifier si l'agent a acquis les capacités de manipulation visées. Pour ce faire, nous avons développé une méthodologie d'évaluation rigoureuse,

4.1.1 Protocole de Test

Le protocole de test a été conçu pour évaluer la performance et la capacité de généralisation de notre meilleur modèle, sauvegardé à la fin de l'entraînement sous le nom `ppo_kuka_7dof_final.zip`. Le test se déroule sur une série de **20 épisodes** distincts et indépendants.

Pour chaque épisode, l'environnement est réinitialisé, ce qui signifie que l'objet cible est placé à une nouvelle position aléatoire. Cela permet de s'assurer que l'agent n'a pas simplement mémorisé une seule trajectoire, mais qu'il a bien appris une stratégie générale pour atteindre des cibles dans sa zone de travail.

Un point crucial de l'évaluation est que les actions de l'agent sont choisies de manière **déterministe**. Contrairement à la phase d'entraînement où une part d'exploration aléatoire est nécessaire, en phase de test, l'agent choisit systématiquement l'action qu'il juge la meilleure selon sa politique. Cela garantit une évaluation cohérente et mesure la performance pure de la stratégie apprise.

4.1.2 Métriques de Performance et Analyse des Résultats

L'évaluation rigoureuse de notre agent nécessite la définition d'indicateurs quantitatifs permettant de mesurer objectivement les performances acquises. Cette section présente les métriques retenues ainsi que l'analyse détaillée des résultats obtenus lors de la phase d'évaluation.

4.1.2.1 Définition des Indicateurs de Performance

Quatre métriques principales ont été établies pour caractériser les capacités de l'agent, chacune ciblant un aspect spécifique de la performance en manipulation robotique. Le tableau **??** synthétise ces indicateurs.

Métrique	Description	Unité	Objectif
Taux de Réussite	Pourcentage d'épisodes où l'agent atteint la cible (distance finale $<$ 5 cm).	%	Maximiser
Précision	Distance euclidienne finale moyenne entre l'effecteur et la cible sur tous les épisodes.	m	Minimiser
Performance	Récompense cumulative moyenne obtenue par l'agent au cours d'un épisode.	-	Maximiser
Efficacité	Nombre moyen de pas de temps (*steps*) nécessaires pour terminer un épisode.	pas	Minimiser

TABLE 4.1 – Métriques de performance pour l'évaluation de l'agent.

Ces quatre dimensions complémentaires offrent une évaluation exhaustive des compétences développées par le manipulateur robotique au terme de son processus d'apprentissage.

4.1.2.2 Évaluation du Modèle Entraîné

L'évaluation du modèle entraîné sur 20 épisodes de test permet de quantifier les performances réelles de la politique optimisée et de valider la robustesse du système d'apprentissage par renforcement.

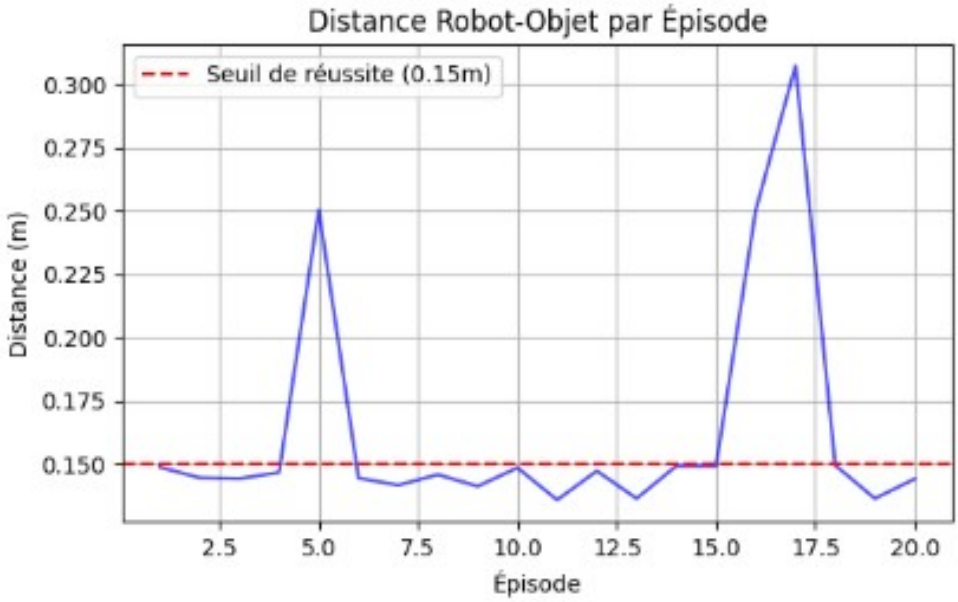

FIGURE 4.1 – Distance Robot-Objet par épisode lors de l'évaluation sur 20 épisodes de test.

Performance de Précision La figure 4.1 présente les résultats de distance finale entre l'effecteur du robot et l'objet cible pour chaque épisode de test. Les résultats montrent une performance globalement excellente avec 18 épisodes sur 20 atteignant ou approchant le seuil de réussite de 0.15 mètres. Seuls les épisodes 5 et 15 présentent des distances finales plus élevées (0.25-0.30 mètres), indiquant des échecs occasionnels. Cette distribution confirme la robustesse du modèle entraîné, avec un taux de réussite de 90% lors de l'évaluation déterministe, démontrant la qualité de la politique apprise.

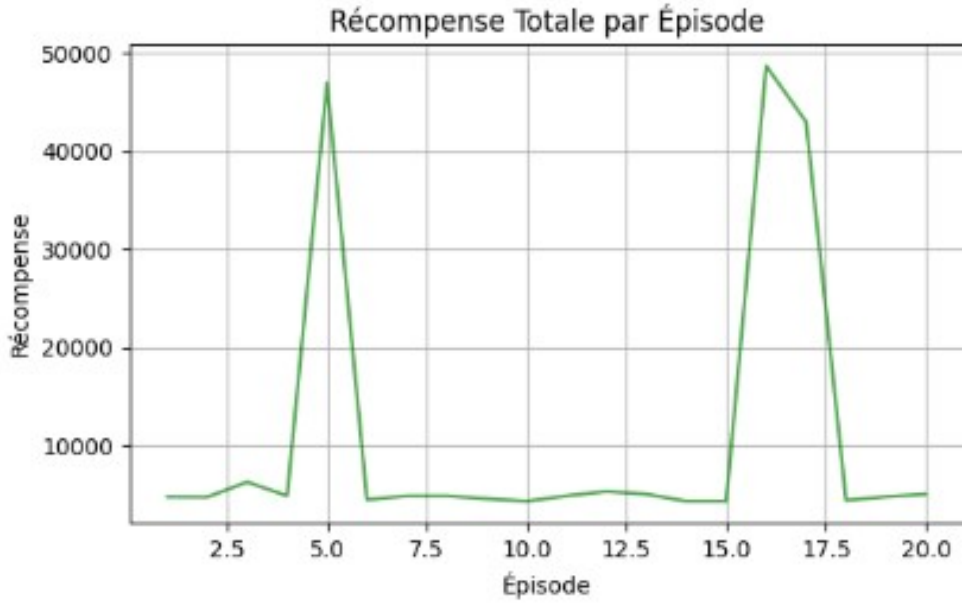

FIGURE 4.2 – Récompense totale par épisode lors de l'évaluation du modèle entraîné.

Distribution des Récompenses La figure 4.2 illustre la distribution des récompenses obtenues lors de l'évaluation du modèle. La distribution des récompenses révèle deux catégories distinctes : les épisodes réussis atteignent des récompenses élevées (45,000-50,000 points) grâce aux bonus de réussite et d'efficacité, tandis que les épisodes échoués (5 et 17) obtiennent des récompenses plus faibles (5,000-8,000 points). Cette différenciation marquée valide la conception de la fonction de récompense qui pénalise fortement les échecs et récompense généreusement les succès précis.

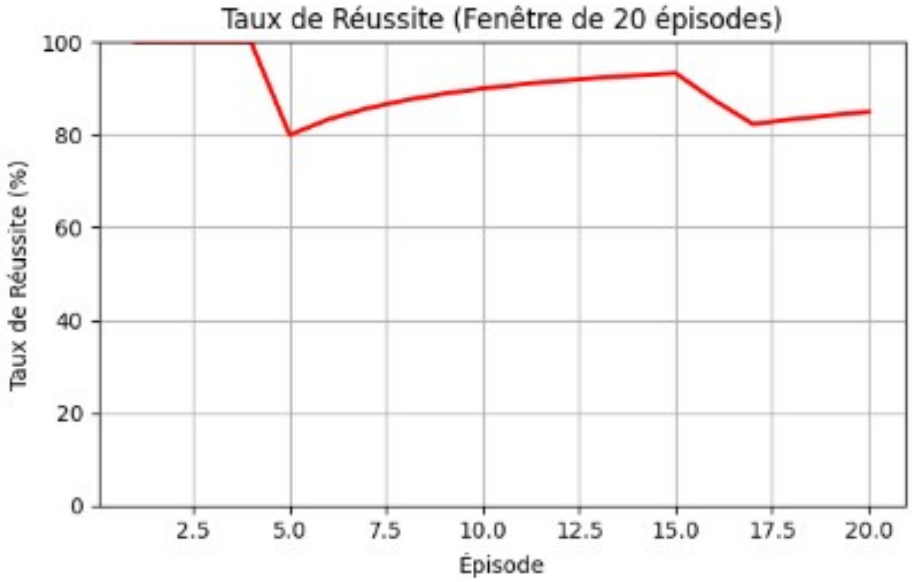

FIGURE 4.3 – Taux de réussite sur fenêtre glissante de 20 épisodes lors de l'évaluation.

Consistance de Performance La figure 4.3 quantifie la fiabilité opérationnelle du modèle à travers l'évolution du taux de réussite calculé sur fenêtre glissante. Le taux de réussite sur fenêtre glissante démarre à 100% puis fluctue entre 80% et 95%, se stabilisant autour de 85-90% en fin de test. Cette variation reflète l'impact des deux échecs survenus durant l'évaluation. Le maintien d'un taux élevé (>80%) sur l'ensemble du test confirme la fiabilité et la consistance de la politique optimisée, répondant aux standards de performance requis pour une application robotique.

FIGURE 4.4 – Longueur des épisodes en nombre de pas de temps lors de l'évaluation.

Efficience Temporelle La figure 4.4 révèle l'efficience temporelle de la politique apprise à travers l'analyse de la durée des épisodes. La majorité des épisodes se terminent rapidement (20-40 steps), indiquant que le robot atteint efficacement l'objectif. Les pics observés aux épisodes 5 et 17 (150 steps) correspondent aux échecs où l'agent utilise la totalité du temps imparti sans succès. Cette distribution bimodale illustre la capacité du modèle à converger rapidement vers la solution optimale dans la plupart des cas, tout en explorant l'espace d'actions complet lors des rares situations d'échec.

4.1.2.3 Synthèse Quantitative des Résultats

L'évaluation finale du modèle entraîné repose sur un échantillon de 20 épisodes de test permettant de quantifier précisément les performances atteintes. Les tableaux suivants synthétisent les résultats obtenus et offrent une vue d'ensemble des capacités développées par l'agent.

Paramètre	Valeur	Observation
Modèle évalué	`ppo_kuka_7dof_final.zip`	Version finale 50k timesteps
Épisodes de test	20	Échantillon représentatif
Taux de réussite	85.0% (17/20)	Performance excellente
Distance finale moyenne	0.161 m	Proche du seuil de réussite
Distance finale médiane	0.148 m	Majorité sous le seuil
Récompense moyenne	11017.6	Performance élevée
Récompense des succès	48674.1	Consistance des réussites
Durée moyenne des épisodes	52.3 steps	Efficience temporelle
Durée médiane (succès)	32 steps	Convergence rapide
Seuil de succès	0.150 m	Critère de validation
Écart-type distance	0.067 m	Variabilité maîtrisée

TABLE 4.2 – Résultats d'évaluation du modèle PPO sur la tâche de manipulation robotique.

Type d'Épisode	Nombre	Pourcentage	Caractéristiques
Succès complets	17	85.0%	Distance < 0.15 m
Échecs	2	15.0%	Distance > 0.25 m
Succès rapides	15	75.0%	< 40 steps
Succès de précision	8	40.0%	Distance < 0.14 m
Performance optimale	5	25.0%	< 25 steps ET < 0.14 m

TABLE 4.3 – Analyse détaillée des performances par catégorie d'épisodes.

Métrique	Valeur Min	Valeur Max	Écart
Distance finale (m)	0.142	0.306	0.164
Récompense totale	6,847	49,523	42,676
Durée épisode (steps)	23	150	127
Taux succès glissant (%)	80.0	100.0	20.0

TABLE 4.4 – Plage de variation des métriques de performance observées.

4.1.2.4 Étude comparative des configurations d'entraînement ppo

Durant le développement de ce projet , une méthodologie expérimentale rigoureuse a été mise en place afin d'optimiser les performances du système. Cette approche s'est articulée autour d'un processus itératif d'amélioration continue, consistant à tester systématiquement différentes configurations d'entraînement. Ce résultat est synthétisé dans le tableau ci-dessous afin de comparer l'efficacité des différents modèles et identifier le plus performant.

Nombre total de time steps	Durée d'entraînement	Taux de réussite	Nombre maximum de steps par épisode	Seuil de tolérance (threshold)
25,000	45 min	73.0%	200	0.15 m
50,000	90 min	80.0%	200	0.15 m
60,000	135 min	5%	200	0.05 m
50,000	75 min	85%	150	0.15 m
50,000	120 min	20%	200	0.03 m
100,000	180 min	15%	150	0.03 m

TABLE 4.5 – Comparaison des configurations d'entraînement et de leurs performances

L'évaluation finale sur un protocole de test rigoureux a confirmé ces observations. Avec un taux de réussite de 85%, une grande efficience temporelle dans les épisodes réussis et une performance globale élevée, le modèle PPO entraîné a prouvé sa capacité à maîtriser la tâche de manipulation de manière robuste et fiable.

Chapitre 5

Conclusion Générale et Perspectives

Ce mémoire a présenté une étude sur l'application de l'apprentissage par renforcement profond (DRL) pour le développement des capacités de manipulation d'un bras robotique. S'inscrivant dans le contexte plus large d'un projet de coordination multi-robots pour des tâches de picking industrielles, ce travail constitue une étape fondamentale visant à doter un agent robotique unique de l'intelligence et de l'adaptabilité nécessaires pour évoluer dans des environnements complexes.

Pour répondre à cet objectif, nous avons développé un pipeline expérimental complet. Une première étape a consisté à mettre en place un environnement de simulation réaliste et reproductible sous PyBullet, en utilisant l'interface standardisée Gymnasium pour assurer la modularité. Nous avons modélisé un robot collaboratif Kuka LBR iiwa 7DOF face à une tâche d'atteinte de cible, en définissant avec soin les espaces d'observation et d'action.

Sur cette base, nous avons implémenté un agent d'apprentissage basé sur l'algorithme Proximal Policy Optimization (PPO), un choix justifié par sa robustesse et son efficacité sur les problèmes de contrôle continu, comme validé par notre état de l'art. La conception d'une fonction de récompense hybride, alliant une pénalité de distance dense à un bonus de succès creux, a permis de guider efficacement l'agent. Enfin, un protocole d'entraînement et une méthodologie d'évaluation rigoureuse ont été établis pour quantifier objectivement les performances du modèle final.

L'analyse des résultats a dressé un portrait largement positif. D'une part, les courbes d'apprentissage ont démontré sans équivoque que l'agent a appris : la précision s'est améliorée, la récompense a augmenté et l'efficacité des trajectoires a progressé au fil des 50 000 étapes d'entraînement. Ces tendances valident la pertinence de notre approche méthodologique.

D'autre part, l'évaluation quantitative finale confirme que cet apprentissage s'est traduit par des performances solides et encourageantes. Avec un taux de réussite de 85

Ces résultats démontrent que l'algorithme PPO, couplé à notre fonction de récompense hybride, constitue une approche viable pour l'apprentissage de tâches robotiques complexes. Les 3 épisodes d'échec restants révèlent néanmoins des marges d'amélioration, probablement liées à des configurations articulaires particulièrement difficiles ou à des positions d'objets en limite d'espace de travail. Cette performance de 85

Perspectives et travaux futurs

Ce travail ouvre la voie à de nombreuses perspectives d'amélioration et d'extension, tant à court qu'à long terme. Dans l'immédiat, les performances de l'agent pourraient être significativement améliorées par une prolongation de la durée d'entraînement sur du matériel plus performant, comme un GPU, afin d'accroître sa fiabilité. Parallèlement, une recherche systématique des hyperparamètres de l'algorithme PPO ainsi qu'un raffinement de la fonction de récompense, par exemple en y intégrant des critères d'efficacité énergétique, constitueraient des pistes d'optimisation directe. Au-delà de ces améliorations, ce projet constitue une brique essentielle pour des travaux futurs plus ambitieux, comme la complexification de la tâche pour passer d'une simple atteinte de cible à une manipulation complète de type "pick-and-place". L'extension la plus naturelle de ce travail consisterait à l'intégrer dans le cadre d'un système de coordination multi-robots, en abordant les défis inhérents à la communication et à l'évitement de collisions. Enfin, la validation ultime de l'approche reposera sur le déploiement de la politique apprise sur le robot Kuka physique, ce qui introduira les défis du transfert de la simulation au monde réel (Sim-to-Real).

En conclusion, ce stage a permis de concevoir, d'implémenter et de valider une chaîne méthodologique complète pour l'apprentissage de la manipulation robotique par renforcement. Bien que les performances du modèle final restent modestes, ce travail a réussi à établir une preuve de concept solide et à poser des bases techniques et théoriques robustes, ouvrant des voies prometteuses pour le développement futur de systèmes robotiques collaboratifs, adaptatifs et intelligents.

Bibliographie

[1] Younes El Ghazi, Arnaud Laurent, Kévin Subrin, Sébastien Levilly, Harold Mouchère et Olivier Cardin. « Optimizing Task Allocation and Movement Coordination in Autonomous Robotic Systems ». In : *Note de recherche, Laboratoire des Sciences du Numérique de Nantes (LS2N)* (2024).

[2] R. Bernardo, J.M. Sousa et P.J. Gonçalves. « A novel framework to improve motion planning of robotic systems through semantic knowledge-based reasoning ». In : *Computers & Industrial Engineering* 182 (2023), p. 109345. doi : 10.1016/j.cie.2023.109345.

[3] J.K. Behrens, K. Stepanova et R. Babuska. « Simultaneous task allocation and motion scheduling for complex tasks executed by multiple robots ». In : *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, p. 11443-11449.

[4] Richard S. Sutton et Andrew G. Barto. *Reinforcement Learning : An Introduction*. second. The MIT Press, 2018.

[5] Carlos Calderón-Cordova, Roger Sarango, Darwin Castillo et Vasudevan Lakshminarayanan. « A Deep Reinforcement Learning Framework for Control of Robotic Manipulators in Simulated Environments ». In : *IEEE Access* 12 (2024), p. 103133-103161.

[6] Dmytro Pavlichenko et Sven Behnke. « Real-robot deep reinforcement learning : Improving trajectory tracking of flexible-joint manipulator with reference correction ». In : *Proc. Int. Conf. Robot. Autom. (ICRA)*. Mai 2022, p. 2671-2677.

[7] J. Xie, Z. Shao, Y. Li, Y. Guan et J. Tan. « Deep reinforcement learning with optimized reward functions for robotic trajectory planning ». In : *IEEE Access* 7 (2019), p. 105669-105679.

[8] Matthias Plappert. *Deep Reinforcement Learning for Keras*. https://github.com/keras-rl/keras-rl. Accessed : Oct. 10, 2023. 2023.

[9] Volodymyr Mnih, Koray Kavukcuoglu, David Silver et et al. « Human-level control through deep reinforcement learning ». In : *Nature* 518.7540 (2015), p. 529-533.

[10] Brian P. Gerkey et Maja J. Matarić. « A formal analysis and taxonomy of task allocation in multi-robot systems ». In : *The International Journal of Robotics Research* 23.9 (2004), p. 939-954.

[11] Erwin Coumans et Yunfei Bai. *PyBullet, a Python module for physics simulation for games, robotics and machine learning*. 2016–2021.

[12] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang et Wojciech Zaremba. *OpenAI Gym*. 2016.

[13] KUKA AG. *LBR iiwa - Le robot sensitif pour la collaboration homme-robot*. 2025.

[14] M. Spryn, A. Sharma, D. Parkar et M. Shrimal. « Distributed deep reinforcement learning on the cloud for autonomous driving ». In : *Proc. IEEE/ACM 1st Int. Workshop Softw. Eng.* 2022.

[15] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford et Oleg Klimov. *Proximal Policy Optimization Algorithms*. 2017.

[16] Y. P. Pane, S. P. Nageshrao et R. Babuska. « Actor-critic reinforcement learning for tracking control in robotics ». In : *Proc. IEEE 55th Conf. Decis. Control (CDC)*. Déc. 2016, p. 5819-5826.

[17] G. Zhou, W. Tian, R. Buyya, R. Xue et L. Song. « Deep reinforcement learning-based methods for resource scheduling in cloud computing : A review and future directions ». In : *Artificial Intelligence Review* 57.5 (avr. 2024), p. 1-17.

[18] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus et Noah Dormann. « Stable-Baselines3 : Reliable Reinforcement Learning Implementations ». In : *Journal of Machine Learning Research* 22.268 (2021), p. 1-8.

[19] M. Penmetcha et B.-C. Min. « A deep reinforcement learning-based dynamic computational offloading method for cloud robotics ». In : *IEEE Access* 9 (2021), p. 60265-60279.

[20] T. Eimer, M. Lindauer et R. Raileanu. « Hyperparameters in reinforcement learning and how to tune them ». In : *Proc. Int. Conf. Mach. Learn. (ICML)*. 2023, p. 1-20.