

République Algérienne Démocratique et Populaire  
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique



Université des Sciences et de la Technologie Houari Boumediene  
Faculté d'Informatique

## Mémoire de fin d'études

Pour l'obtention du diplôme Master

Option : Systèmes Informatiques Intelligents

---

# Digitalisation des entrepôts / drônes : Simulation des activités intralogistiques

---

Réalisé par :

Radhia DJEBROUNI  
USTHB

Encadré par :

Pr. Abdelghani BEKRAR, UPHF , France  
Pr. Kamel BOUKHALFA, USTHB , Algérie

Membres de jurys :

Pr. SELMOUNE Nazih, Président, USTHB

Pr. AMANI Ferhat, Membre, USTHB

N° : SII-E-21 / Promotion : 2021/2022

## *Dédicaces*

*To you dear dad*

*To all my dear family members...to my beloved friends*

*To you dear reader, enjoy !*

*Finally..To the little of me, little Radia, I did it, and you will !*

*Radia*

## ***Remerciements***

*Ce travail est l'aboutissement d'un semestre d'efforts acharnés et d'agréables défis vécus. Il n'aurait sans doute pu être concluant sans l'inestimable sollicitude de nombreuses personnes pour lesquelles je veux exprimer ma sincère gratitude.*

*Avant tout remerciement, louange à Dieu. Je remercie ALLAH, gloire à lui pour m'avoir accordé lucidité, vigueur et constance durant mon projet. De m'avoir donné le courage, la volonté et la patience pour mener à terme le présent travail.*

*Au premier rang, j'adresse mes plus sincères remerciements pour mon équipe de travail, mes encadrants Pr. Abdelghani BEKRAR, Pr. Abdecamed Ait El Cadi et Pr. Kamel BOUKHALFA, pour avoir accepté de m'encadrer, de m'avoir proposé un sujet aussi riche et passionnant, de leur aide et leurs précieux conseils tout au long du travail. Je remercie mon binôme monomiel Aicha, et j'exprime ma gratitude à Yasmine et Halima, Je vous remercie infiniment pour tout ce que vous avez fait pour nous pendant cette année pleine de défis.*

*J'exprime ma reconnaissance envers les membres du jury pour avoir accepté d'examiner mon travail et de l'enrichir par leurs remarques et propositions.*

*Je ne pourrais omettre d'exprimer ma gratitude infinie envers ma famille et mes amies qui ont été une source de soutien et de courage durant mon travail, envers le Micro club, qui a rendu mon année exceptionnelle, pleine d'aventures, de travail d'équipe, de succès, et de joie.*

*Et pour finir, un grand merci à toutes les personnes qui m'ont soutenue, de près ou de loin, et qui ont participé à la réalisation de ce mémoire.*

## مُلَحَّص

يَتَطَلَّب مُسْتَقْبِل إِدَارَة الْمُسْتَوْدِعَات وَمَرَاكِز التَّوزِيع مُرْوَنَة قُصُوبَى ، وَالصَّنَاعَة ٤ ، ٠ تَدُورُ حَوْلَ اسْتِخْدَامِ التَّكْنُولُوْجِيَا لِتَلْبِيَة هَذِهِ الْاِحْتِيَاجَات ، كَمَا تُعَتَّبِرُ الْمُرَكَّبَات الْجَوِيَّة بِدُون طَيَّار (UAVs) تِقْنِيَّةً أَسَاسِيَّةً لِلْمُسْتَوْدِعَات الْذِيَّيَّة . مِنْ بَيْنِ التَّحْديَات الَّتِي يَتَمُّ مُواجِهَتَهَا عِنْدَ اسْتِخْدَامِ الطَّائِرَات بِدُون طَيَّار لِأَتْمَتَةِ الْمُسْتَوْدِعَات تَحْطِيطِ الْمَسَار . تَمَّ إِجْرَاءِ الْعَدِيدِ مِنَ الْأَبْحَاثِ لِحَلِّ هَذِهِ الْمُشَكِّلَة ، وَالْهَدْفُ الْمُشَتَّرُك لِجَمِيعِ الْمَقَالَات هُوَ الْوُصُولُ إِلَى حَلٌّ بِأَقْلَى تَكْلُفٍ وَأَقْلَى قَدْرٍ مِنَ التَّعْقِيْدِ وَأَقْلَى وَقْتٍ تَدْرِيْبٍ وَبِأَقْصَى قَدْرٍ مِنَ الرِّبَح . يُخَاطِلُ عَمَلَنَا إِلْجَاهَةَ عَلَى سُؤَالٍ رَئِيْسِيٍّ وَهُوَ : كَيْفِيَّة اسْتِخْدَامِ التَّعْلُم التَّعْزيْيِيِّيِّ الْعَمِيقِ لِتَحْقِيقِ مُخَطَّطِ مَسَارِ دُونِ الْحَاجَةِ إِلَى خَرِيْطةِ عَالَمِيَّةِ لِلْبَيْنَة ، وَذَلِكَ مِنْ جَلَّ نَمْذَجَةِ الْمُشَكِّلَةِ فِي شَكَلِ نَمْوَذَج MDP ، ثُمَّ تَطْبِيقِ RL ظُرُقَ . وَمَعَ ذَلِكَ ، لِتَقْدِيمِ الْرُوْبُوتَاتِ إِلَى السُّوقِ ، فَإِنَّ الْإِنْتَاجِ الْضَّخِمِ يَتَكَلَّفُهُ مُنْخَفِضَةٌ يُمَثِّلُ تَحْدِيَاً أَيْضًا . لِذَلِكَ ، فِي هَذَا الْعَمَلِ ، نُخَاطِلُ بِنَاءَ نِظَامِ الْكَشْفِ عَنِ الْلَّيْزَرِ الزَّائِفِ بِنَاءَ عَلَى تَنْبُوِ الْعُمَقِ الْمُبَاشِرِ مِنْ صُورِ الْكَامِيرَا الفَرَدِيَّةِ مَعَ الْحُفَاظِ عَلَى الْأَدَاءِ الْمُسْتَقِرَّ . تَمَّتِ مُحاَكَاهَةُ الْعَمَلِ بِاسْتِخْدَامِ جَهَازِ مُحاَكَاهَةِ ثَلَاثِيِّ الْإِبْعَادِ . أَظَهَرَتِ النَّتَائِجِ التَّجَرِيْبِيَّةِ مِنْ نَاحِيَّةِ كَفَاءَةِ DDPG مُقاَرَنَةً بِخَوارِزمِيَّةِ PPO ، وَإِنَّ الْمُخَطَّطَ الْمُدْرَبُ يُمْكِنُ اسْتِخْدَامِهِ مُبَاشِرَةً فِي بَيْنَهُ افتراضِيَّةٌ غَيْرُ مَعْرُوفَةٌ ، مِنْ نَاحِيَّةِ أُخْرَى ، أَظَهَرَتِ التَّقيِيمَاتُ أَنَّ أَسْلُوبَ Monocular يُمْكِنُ أَنْ يَتَفَوَّقَ عَلَى الْلَّيْزَرِ . اسْتِخْدَامُ ، عَلَى الرَّغْمِ مِنْ أَنَّهُ يَسْتَغْرِقُ وَقْتًا أَطْوَلَ .

**الكلمات المفتاحية :** الطَّائِرَات بِدُون طَيَّار ، مُخَطَّطُ الْمَسَار ، التَّعْلُمِ الْمَعَزَّزِ الْعَمِيق ، عَمَلِيَّةِ اِتَّخَادِ الْقَرَازِ مَارِكُوف ، مُحاَكَاهَةِ ثَلَاثِيَّةِ الْإِبْعَاد ، الْكَامِيرَا الْأَحَادِيَّة .

# Abstract

The future of warehouse and distribution center management requires maximum flexibility, Industry 4.0 is all about using technology to meet these needs. Unmanned Aerial Vehicles (UAVs) are also considered a key technology for smart warehouses. Among the challenges encountered when using drones for warehouse automation is path planning. Several researches have been made in order to solve this problem, the common goal of all the articles is to achieve a solution with a minimum of cost, minimum of complexity, minimum of training time and with a maximum of gain. Our work attempts to answer a main question which is : How to use deep reinforcement learning to achieve a trajectory planner without the need for a global map of the environment, and this by modeling the problem in the form of a MDP model, then applying RL methods. However, to introduce robots to the market, mass production at low cost is also a challenge. Therefore, in this work, we attempt to build a pseudo-laser detection system based on direct depth matrix prediction from single camera images while maintaining stable performance. The work was simulated using a 3D simulator. The experimental results show on the one hand, the efficiency of the DDPG compared to the PPO algorithm, and that the trained planner can be used directly in an unknown virtual environment, on the other hand, the evaluations show that the approach Monocular camera can replace laser use, although it takes longer.

**Index Terms** :UAV, path planner, deep reinforcement learning, markov decision process, 3d simulator, monocular camera.

## Résumé

L'avenir de la gestion des entrepôts et des centres de distribution nécessite une flexibilité maximale, l'industrie 4.0 consiste à utiliser la technologie pour répondre à ces besoins. Les véhicules aériens sans pilote (UAV) sont également considérés comme une technologie clé pour les entrepôts intelligents. Parmi les défis rencontrés lors de l'utilisation des drones pour l'automatisation des entrepôts, la planification de trajectoire. Plusieurs recherches ont été faites afin de résoudre cette problématique, le but commun de tous les articles est de réaliser une solution ayant un minimum de coût, minimum de complexité, minimum de temps d'entraînement et avec un maximum de gain. Notre travail tente de répondre à une question principale qui est : Comment utiliser l'apprentissage par renforcement profond pour réaliser un planificateur de trajectoire sans avoir besoin d'une carte globale de l'environnement, et ceci en modélisant la problématique sous forme d'un modèle MDP, puis en appliquant les méthodes du RL. Cependant, pour introduire des robots sur le marché, la production de masse à faible coût est également un défi. Par conséquent, dans ce travail, nous tentons de construire un système de détection pseudo-laser basé sur la prédiction directe de la matrice de profondeur à partir d'images de caméra unique tout en maintenant des performances stables. Les travaux ont été simulés à l'aide d'un simulateur 3D. Les résultats expérimentaux montrent d'un côté, l'efficacité du DDPG par rapport à l'algorithme PPO, et que le planificateur formé peut être utilisé directement dans un environnement virtuel inconnu, d'un autre côté, les évaluations montrent que L'approche Caméra Monoculaire peut faire le travail d'un laser, bien que cela prenne plus de temps.

**Termes de l'index :** drone, planificateur de chemin sans carte, apprentissage par renforcement approfondi, processus de décision de Markov, simulateur 3D, Caméra Monoculaire

## TABLE DES MATIÈRES

<b>Introduction générale</b>	<b>1</b>
<b>I État de l'art</b>	<b>3</b>
<b>1 Généralités</b>	<b>4</b>
1.1 Introduction . . . . .	4
1.2 Industrie 4.0 . . . . .	4
1.2.1 Entrepôts . . . . .	4
1.2.2 Automatisation des entrepôts . . . . .	5
1.2.3 UAVs (Drones) . . . . .	5
1.2.4 Fonctionnement des drones dans les entrepôts . . . . .	6
1.2.5 Technologie de L'intralogistique 4.0 . . . . .	6
1.3 Intelligence Artificielle . . . . .	7
1.3.1 Apprentissage automatique . . . . .	7
1.3.2 Apprentissage profond . . . . .	8
1.3.3 Apprentissage par renforcement . . . . .	9
1.3.3.a Définition . . . . .	9
1.3.3.b Markov Decision Process . . . . .	9
1.3.3.c Éléments clés du MDP /RL . . . . .	9
1.3.3.d Résoudre le MDP mathématiquement [14] . . . . .	11
1.3.3.e Fonctionnement du RL . . . . .	13
1.3.3.f Défis . . . . .	13
1.3.3.g Off / On policy RL . . . . .	14
1.3.3.h Apprentissage par renforcement profond. . . . .	14
1.4 Conclusion . . . . .	14
<b>2 Déigitalisation des entrepôts/ drones dans l'état de l'art</b>	<b>16</b>
2.1 Introduction . . . . .	16
2.2 Défis . . . . .	16
2.3 UAV Path Planning . . . . .	17

2.3.1	Description de la problématique . . . . .	17
2.3.2	Travaux connexes . . . . .	18
2.4	Conclusion . . . . .	20
<b>II</b>	<b>Conception, implémentation et évaluation de la solution</b>	<b>21</b>
<b>1</b>	<b>Conception de la solution</b>	<b>22</b>
1.1	Introduction . . . . .	22
1.2	Vue globale sur la solution . . . . .	22
1.3	Modélisation de la problématique . . . . .	23
1.3.1	Agent : . . . . .	23
1.3.2	État (input) : . . . . .	23
1.3.3	Action(Output) : . . . . .	23
1.3.4	Environnement : . . . . .	24
1.3.5	Calcul du Reward : . . . . .	25
1.4	Apprentissage par renforcement profond . . . . .	25
1.4.1	Le réseaux Actor-Critic . . . . .	26
1.4.2	La méthode On-policy (PPO) . . . . .	27
1.4.3	La méthode Off-policy (DDPG) . . . . .	29
1.4.3.a	Deterministic Policy Gradient(DPG) . . . . .	29
1.4.3.b	Q-learning . . . . .	29
1.4.3.c	Deep Q-Network . . . . .	30
1.4.3.d	Deep Deterministic Policy Gradient(DDPG) . . . . .	31
1.4.4	Détection des obstacles . . . . .	34
1.4.4.a	Détection des obstacles en utilisant un Laser . . . . .	34
1.4.4.b	Détection des obstacles en utilisant l'estimation de profondeur monoculaire auto-supervisée . . . . .	35
1.5	Conclusion . . . . .	36
<b>2</b>	<b>Implémentation de la solution</b>	<b>37</b>
2.1	Introduction . . . . .	37
2.2	Outils d'Implémentation . . . . .	37
2.2.1	Matériel . . . . .	37
2.2.2	Logiciels et librairies . . . . .	38
2.2.2.a	Robot operating system (ROS) . . . . .	38
2.2.2.b	Gazebo . . . . .	39
2.2.2.c	Vs Code . . . . .	39
2.2.2.d	Langages . . . . .	40
2.2.2.e	Librairies . . . . .	40
2.3	Conclusion . . . . .	41
<b>3</b>	<b>Expériences et évaluation de la solution</b>	<b>42</b>
3.1	Introduction . . . . .	42
3.2	Environnements du test . . . . .	42

3.2.1	Environnement simple . . . . .	42
3.2.2	Environnement complexe . . . . .	43
3.2.3	Environnement du test final . . . . .	44
3.3	Paramètres utilisés . . . . .	46
3.4	Résultats des tests . . . . .	46
3.4.1	Environnement Simple . . . . .	47
3.4.1.a	PPO vs DDPG . . . . .	47
3.4.1.b	Monodepth camera vs Laser . . . . .	49
3.4.2	Environnement Complexe . . . . .	50
3.4.3	laser vs monodepth . . . . .	50
3.5	Discussion . . . . .	51
	<b>Conclusion générale</b>	<b>52</b>
	<b>Bibliographie</b>	<b>57</b>

## LISTE DES FIGURES

Figure 1	Entrepôts [3]	5
Figure 2	Drone (UAV)	6
Figure 3	DRL / ML / DL [9]	7
Figure 4	Fonctionnement du RL[13]	11
Figure 5	Diagramme de sauvegarde de fonction de valeur [14]	12
Figure 6	Architecture de l'environnement RL	25
Figure 7	Actor-Critic	26
Figure 8	Actor-Critic Architecture	27
Figure 9	Architecture du PPO [33]	28
Figure 10	Pseudo code du PPO [32]	29
Figure 11	Q-learning pseudo code[35]	30
Figure 12	DQN state et action [36]	30
Figure 13	Deep Q Network pseudo code[35]	31
Figure 14	Architecture du DDPG[37]	32
Figure 15	Pseudo code du DDPG [38]	33
Figure 16	Explication du DDPG [38]	34
Figure 17	Architecture du Ros	38
Figure 18	Structure du Gazebo	39
Figure 19	Structure du Gazebo/ ROS packages	40
Figure 20	Environnement simple du training	43
Figure 21	Environnement complexe du training	43
Figure 22	Environnement complexe du training	44
Figure 23	Environnement du test : entrepôt	45
Figure 24	Environnement du test : entrepôt image 2	45
Figure 25	Résultat de l'entraînement du DDPG dans un environnement simple (Moyenne du récompense)	47
Figure 26	Résultat de l'entraînement du PPO dans un environnement simple (Moyenne du récompense)	48

Figure 27 Comparaison entre les résultats du Laser vs Monocular Caméra dans un environnement simple . . . . .	49
Figure 28 Comparaison entre les résultats du Laser vs Monocular Caméra dans un environnement simple . . . . .	50

## LISTE DES TABLEAUX

Tableau I	Travaux connexes . . . . .	20
Tableau II	Angular vs Linear velocity [29] . . . . .	24
Tableau III	Retour d'une camera normale vs Monocular Camera . . . . .	36
Tableau IV	Machine utilisée . . . . .	37
Tableau V	Paramètres utilisés pour le Actor-critic Network . . . . .	46
Tableau VI	Paramètres utilisés pour le DRL . . . . .	46
Tableau VII	PPO vs DDPG - environnement simple . . . . .	48
Tableau VIII	Monocular Camera vs Laser - environnement simple . . . . .	49
Tableau IX	Monocular Camera vs Laser - environnement complexe . . . . .	50

## ACRONYMES

**CPU** Central Processing Unit.

**DDPG** Deep Deterministic Policy Gradient.

**DL** Deep Learning.

**DPG** Deterministic Policy Gradient.

**DQN** Deep Q Network.

**DRL** Deep reinforcement learning.

**GPU** Graphics Processing Unit.

**IA** Intelligent Agent.

**LASER** Unité de Calcul de Tenseurs.

**MDP** Markov decision process.

**ML** Machine Learning.

**PDDPG** Parallel Deep Deterministic Policy Gradient.

**PPO** Proximal Policy Optimization.

**PRM** Probabilistic Roadmap-Reinforcement Learning.

**RL** Reinforcement Learning.

**ROS** Robot Operating System.

**SLAM** Simultaneous Localization And Mapping.

**TD3** Twin-Delayed Deep Deterministic Policy Gradient Agents.

**UAV** Unmanned Aerial Vehicle.

## INTRODUCTION GÉNÉRALE

### Contexte

L'avenir de la gestion des entrepôts et des centres de distribution nécessite une flexibilité maximale, une automatisation complète et des traitements intelligents. L'intralogistique évolue de grands systèmes rigides vers des solutions logicielles modulaires et flexibles, assistées par robot et auto-optimisées.

Nous entrons aujourd'hui dans la quatrième révolution industrielle. L'industrie 4.0 modifie la façon dont nous interagissons avec les machines et dont ces machines interagissent entre elles. Cela aura un impact significatif sur l'entrepôt. Avec une intelligence décentralisée et en réseau, une robotique avancée et des processus auto-organisés, l'entrepôt du futur fera un pas de géant en termes de productivité, de flexibilité et d'efficacité.

L'industrie 4.0 consiste à utiliser la technologie numérique pour améliorer les processus fondamentaux impliqués dans les opérations de logistique et d'entrepôt, tels que le chargement et le déchargement, la préparation des commandes et la gestion des stocks. Cependant, il ne s'agit pas seulement d'utiliser des robots pour entreprendre des tâches ardues ou répétitives ; il s'agit également d'utiliser des ordinateurs pour collecter de grandes quantités de données et de les utiliser pour révéler des informations sur vos processus et apporter d'autres améliorations.

Les véhicules aériens sans pilote (UAV) sont également considérés comme une technologie clé pour les entrepôts intelligents, car ils peuvent effectuer des tâches répétitives et dangereuses avec peu ou pas d'intervention humaine ou de supervision. Ces dernières années, les drones se sont révélés très utiles dans des domaines tels que la télédétection (comme l'exploitation minière), la surveillance en temps réel, la gestion des catastrophes, la surveillance des frontières et de masse, les applications militaires, la livraison de marchandises, l'agriculture de précision, l'inspection des infrastructures, et les médias et le divertissement. Dans bon nombre de ces domaines, les drones sont chargés de collecter dynamiquement autant de données que possible à partir de plusieurs emplacements, l'une des pierres angulaires de l'Industrie 4.0. De plus, les drones peuvent non seulement collecter des données, mais aussi stocker, traiter et échanger des informations avec des appareils utilisés chez les fournisseurs et les usines[1].

---

Parmi les défis rencontrés lors de l'utilisation des drones pour l'automatisation des entrepôts, La planification de trajectoire. La planification de trajectoire est défini par trouver un chemin géométrique de la position actuelle du véhicule à la position cible en évitant tous les obstacles. Plusieurs recherches ont été faites afin de résoudre cette problématique, le but commun de tous les articles est de réaliser une solution ayant un minimum de coût, minimum de complexité, minimum de temps d'entraînement et avec un maximum de gain.

### **Objectifs et questions de recherche**

Notre travail tente de répondre à une question principale qui est : Comment utiliser l'apprentissage par renforcement profond pour réaliser un planificateur de trajectoire sans avoir besoin d'une carte globale de l'environnement ? Cela peut en fait être développé en trois sous-questions :

- Compte tenu des contraintes, comment formuler un tel problème de drone dans un cadre d'apprentissage par renforcement ?
- Comment remplacer l'utilisation d'une carte globale, et rendre la solution dynamique ?
- Quels sont les composants techniques nécessaires pour mettre en œuvre de telles solutions et évaluer leurs performances sans les tester dans un environnement réel ?

### **Contribution**

Une solution proposée dans ce projet est de réaliser un système appliquant différents algorithmes d'apprentissage par renforcement profond visant à guider un agent (drone) vers une position cible, en évitant tous les obstacles qu'il rencontre lors de ses déplacements, sans utiliser une carte globale, et en modélisant la problématique sous forme d'un MDP. Nous proposons aussi l'évaluation des algorithmes de politique par rapport à des algorithmes hors politique et identifions le meilleur. De plus, nous présentons une étude sur l'efficacité de l'utilisation des caméras monoculaires dans les drones et leur capacité à remplacer les lasers dans ces derniers. Finalement, Nous simulons le processus dans des environnements 3D en utilisant Ros et le logiciel Gazebo.

### **Plan**

Ce travail se décline en deux parties. La première intitulée "État de l'art" est une revue de la littérature sur les différents éléments du sujet. Nous commençons dans le premier chapitre de la partie par donner un aperçu général sur l'industrie 4.0, l'intralogistique, les entrepôts et les drones dans le premier chapitre, leur suivent une étude sur les défis et les travaux connexes dans un deuxième chapitre.

La deuxième partie intitulée "Conception, implémentation et évaluation de la solution" commence par introduire la problématique, la solution et ses composants dans le premiers chapitre, dans le deuxième chapitre nous citant les différents outils matériels et logiciels utilisés. Dans le dernier chapitre, nous présentons l'environnement de test, les expériences et résultats atteints de la solution, et ceci en comparant entre les différents concepts suivis.

# Première partie

## État de l'art

# CHAPITRE 1

## GÉNÉRALITÉS

### 1.1 Introduction

Ce premier chapitre est un bref aperçu de trois domaines auxquels notre étude s'intéresse : l'industrie 4.0, l'intelligence artificielle, et la simulation. Les sections suivantes rassemblent quelques définitions et éléments préliminaires qui sont nécessaires pour la bonne compréhension du contenu du mémoire. Dans la première section on énonce quelques généralités sur l'industrie 4.0, ses composants, entre entrepôts et drones, et la gestion de ses composants à l'aide de l'intralogistique 4.0. Ensuite, on explore le domaine de l'intelligence artificielle et ses différentes branches qui nous intéressent, en particulier l'apprentissage profond et l'apprentissage par renforcement.

### 1.2 Industrie 4.0

L'industrie 4.0 est un terme utilisé pour décrire la quatrième révolution industrielle qui se produit dans l'industrie manufacturière, et elle est en train de tout changer ! Ce terme a été créé par le gouvernement allemand en 2011 pour décrire la tendance à l'automatisation et à l'échange de données dans le secteur manufacturier. L'industrie 4.0 repose sur quatre piliers essentiels : les systèmes cyber-physiques, l'internet des objets, le big data et le cloud computing. Ces technologies se rejoignent pour créer une usine connectée où les machines peuvent communiquer entre elles et prendre des décisions de manière autonome. Cela permet une plus grande efficacité et une plus grande flexibilité dans le processus de fabrication, ainsi qu'un meilleur contrôle de la qualité et une réduction des déchets.

#### 1.2.1 Entrepôts

Un entrepôt est une installation qui, à l'aide d'équipements de stockage, d'engins de manutention, de ressources humaines et de moyens de gestion, contrôle les écarts entre les flux d'entrée de marchandises (provenant des fournisseurs, des sites de fabrication, etc.) et de sortie (marchandise destinée à la production, à la vente etc.). Ces flux ne sont généralement

pas coordonnés, ce qui implique la nécessité de mise en place d'une logistique de stockage optimale. [2]



FIGURE 1 – *Entrepôts* [3]

### 1.2.2 Automatisation des entrepôts

L'automatisation des entrepôts est le processus d'automatisation du mouvement des produits vers, à travers, et hors d'un entrepôt ou d'un centre de distribution avec moins d'assistance des travailleurs. Lorsqu'elle est correctement mise en œuvre, une solution d'automatisation d'entrepôt peut éliminer de nombreux processus manuels et à forte intensité de main-d'œuvre. Plus précisément, l'automatisation des entrepôts augmente la productivité des employés, augmente la précision, réduit les coûts, améliore la sécurité[4] Pour cette raison, il est essentiel de travailler avec un fournisseur intralogistique disposant d'une large gamme de solutions couvrant tous les niveaux d'automatisation, de la mise en œuvre d'un logiciel de gestion d'entrepôt dans une installation manuelle à la robotisation de tous les flux de marchandises.[5]

### 1.2.3 UAVs (Drones)

Un drone est un avion sans pilote. Les drones sont plus formellement connus sous le nom de véhicules aériens sans pilote (Unmanned Aerial Vehicles), d'où l'acronyme UAV, ou systèmes d'aéronefs sans pilote. Essentiellement, un drone est un robot volant qui peut être contrôlé à distance ou voler de manière autonome à l'aide de plans de vol contrôlés par logiciel dans ses systèmes embarqués, qui fonctionnent en conjonction avec des capteurs embarqués et un système de positionnement global (GPS).[6]



FIGURE 2 – *Drone (UAV)*

#### 1.2.4 Fonctionnement des drones dans les entrepôts

Afin d'utiliser des drones d'intérieur pour la gestion des stocks des entrepôts, les drones doivent pouvoir fonctionner correctement et en toute sécurité. Pour cette fin, ils doivent être dotés des éléments suivants :

- **Une puissance de traitement embarquée**, pour permettre au drone d'exécuter des algorithmes localement (à bord), même avec une mauvaise connexion réseau qui limiterait la capacité d'obtenir des instructions du système de gestion central pendant le vol.
- **Un système de vision**, ou de caméra qui doit être capable de récupérer les données de l'inventaire<sup>1</sup> et doit être capable d'identifier les racks, les allées, les voies de vrac pour se localiser dans l'entrepôt.
- **Un système d'évitement des collisions**, le drone a besoin de ce système pour assurer la sécurité de son vol dans l'entrepôt.
- **Une SDK/API**. En effet, les données collectées par le drone ne seront pas stockées sur le cerveau ou le logiciel du drone. Ces informations doivent être intégrées dans le système de gestion des stocks existant, de sorte que l'interface SDK/API applicable pour échanger des données est essentielle pour fournir aux gestionnaires des stocks un aperçu de l'entrepôt.

Avec ces technologies disponibles, les drones peuvent aider à améliorer la gestion des stocks avec trois modules : collecte de données, traitement des données, livraison d'informations

#### 1.2.5 Technologie de L'intralogistique 4.0

L'intralogistique 4.0 représente l'implantation de l'automatisation, l'intelligence artificielle et les systèmes d'intercommunication dans les différents processus de l'entrepôt avec des objectifs très clairs : exécuter les mouvements avec plus de précision, de rapidité et à moindre coût.

L'intralogistique 4.0 est née des progrès suivants :

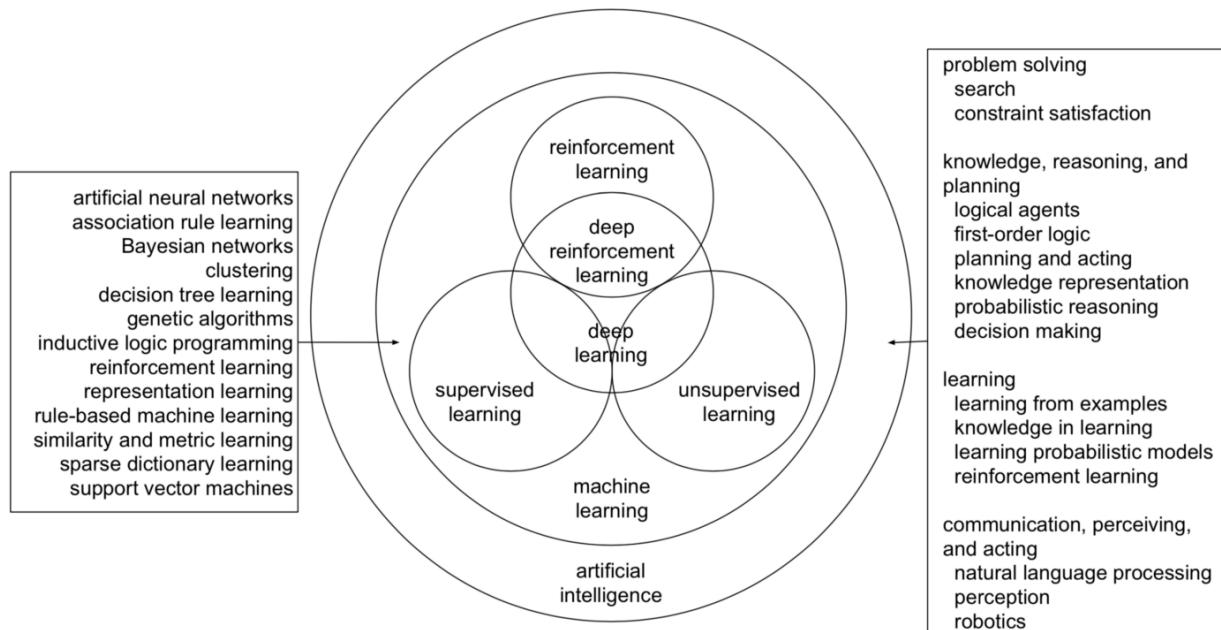
---

1. Le terme inventaire fait référence aux matières premières utilisées dans la production ainsi qu'aux biens produits qui sont disponibles à la vente.<sup>[7]</sup>

- **Automatisation.** Par l'automatisation, on fait références au développement de systèmes de stockage entièrement automatiques et équipements de manutention qui tirent le meilleur parti de l'espace et du volume de construction, tout en offrant une meilleure productivité et une grande disponibilité.
- **Robotique collaborative.** Cette technologie a apporté de l'efficacité à tous les processus de la chaîne d'approvisionnement, du transport des marchandises à la préparation des commandes.
- **Intelligence des données** Le big data est un outil très efficace pour comprendre et analyser les performances de l'entrepôt en temps réel. De cette manière, on peut prédire les scénarios et les comportements et, par conséquent, effectuer des améliorations. Cette technologie simplifie les processus et, grâce à elle, les entreprises sont beaucoup plus agiles et flexibles pour s'adapter aux changements du marché et aux nouveaux besoins des consommateurs.

### 1.3 Intelligence Artificielle

L'intelligence artificielle s'agit de l'ensemble de théories et de techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence humaine.[\[8\]](#)



Yuxi Li, Deep Reinforcement Learning, <https://arxiv.org/abs/1810.06339>, 2018

FIGURE 3 – DRL / ML / DL [\[9\]](#)

#### 1.3.1 Apprentissage automatique

Le Machine Learning (ML), ou apprentissage automatique, ou apprentissage statistique est un sous-ensemble de l'intelligence artificielle. Il s'agit d'un ensemble d'algorithmes d'IA développés pour imiter l'intelligence humaine – l'autre type étant l'IA symbolique, dite «

intelligence artificielle à l'ancienne », ou GOFAI (Good Old-Fashioned AI), qui désigne les moteurs de règles de type if-then (si... alors...). [10]

Les racines de l'apprentissage automatique sont les statistiques, qui peuvent également être considérées comme l'art d'extraire des connaissances des données. En particulier, des méthodes telles que la régression linéaire et les statistiques bayésiennes, qui datent déjà de plus de deux siècles.

Le domaine de l'apprentissage automatique est souvent divisé en sous-domaines selon les types de problèmes abordés. On peut les classer approximativement comme suit :

- **Apprentissage supervisé** : à partir d'une donnée d'entrée, par exemple une photographie avec un panneau de signalisation, la tâche consiste à prédire le résultat ou l'étiquetage correct, par exemple le panneau de signalisation qui se trouve dans l'image (limitation de vitesse, panneau stop, etc.). Dans les cas les plus simples, les réponses se présentent sous la forme de oui/non (on les qualifie alors de problèmes de classification binaire).
- **Apprentissage non supervisé** : il n'y a pas d'étiquetage ou de résultats corrects. La tâche consiste à découvrir la structure des données : par exemple, regrouper des éléments similaires pour former des groupes. La visualisation des données peut aussi être considérée comme un apprentissage non supervisé.
- **Apprentissage par renforcement** : communément utilisé dans les situations où un agent d'IA, comme une voiture autonome, doit fonctionner dans un environnement et où un retour d'information sur les bons ou mauvais choix est disponible avec un certain retard. Également utilisé dans les jeux dont le résultat ne peut être décidé qu'à la fin de la partie.

Les catégories se chevauchent et sont vraiment floues, de sorte qu'il est parfois difficile de ranger une méthode particulière dans une catégorie donnée. Par exemple, ce qu'on appelle l'apprentissage semi-supervisé est, comme son nom l'indique, en partie supervisé et en partie non supervisé. [11]

### 1.3.2 Apprentissage profond

L'Apprentissage Profond (Le deep learning) est souvent décrit comme un réseau neuronal multicouche, mais il s'agit en réalité d'utiliser plusieurs couches pour traiter différents niveaux de caractéristiques<sup>2</sup>, des caractéristiques de niveau inférieur aux caractéristiques de niveau supérieur, un peu comme une conception pyramidale de la connaissance.

Le Deep Learning extrait des informations d'une couche et les transmet à la couche suivante, ce qui lui permet d'affiner et d'atteindre une meilleure compréhension de ce qui est nécessaire. Le deep learning est également lié aux importantes capacités de calcul des processeurs « modernes » qui permettent de faire fonctionner des réseaux neuronaux de la taille de plusieurs millions de neurones.[12]

---

2. Dans le domaine de l'intelligence artificielle, une caractéristique fait référence à un attribut ou une colonne dans les données.

### 1.3.3 Apprentissage par renforcement

#### 1.3.3.a Définition

Le Reinforcement Learning ou apprentissage par renforcement est une méthode de Machine Learning de plus en plus utilisée. Elle consiste à laisser les ordinateurs apprendre de leurs expériences grâce à un système de récompense ou de pénalité. Il pourrait même s'agir de la clé permettant l'avènement d'une intelligence artificielle générale comparable à celle de l'humain.

#### 1.3.3.b Markov Decision Process

L'apprentissage par renforcement utilise le cadre formel des processus décisionnels de Markov pour définir l'interaction entre un agent d'apprentissage et son environnement en termes d'états, d'actions et de récompenses. Ce cadre est destiné à être un moyen simple de représenter les caractéristiques essentielles du problème de l'intelligence artificielle. Ces caractéristiques comprennent un sens de cause à effet, un sentiment d'incertitude et de non-déterminisme, et l'existence d'objectifs explicites [13]

#### 1.3.3.c Éléments clés du MDP /RL

- **Agent et Environnement.** Dans l'apprentissage par renforcement, il y a deux composants principaux :  
Un agent, qui représente la « solution », qui est un programme informatique ayant pour seul rôle de prendre des décisions (actions) pour résoudre des problèmes complexes de prise de décision dans l'incertitude. Un Environnement, c'est-à-dire la représentation d'un « problème », c'est-à-dire tout ce qui vient après la décision de l'Agent. L'environnement réagit avec les conséquences de ces actions, qui sont des observations ou des états, et des récompenses, aussi parfois appelées coûts.
- **State.** L'environnement est représenté par un ensemble de variables liées au problème. Un état est une instanciation de l'espace d'état  $S = s_1 \dots s_n$ , un ensemble de valeurs que prennent les variables.
- **Observation.** Étant donné que nous considérons que l'agent n'a pas accès à l'état complet réel de l'environnement, il est généralement appelé observation, la partie de l'état que l'agent peut observer.
- **L'action et la fonction de transition.** A chaque état, l'environnement met à disposition un ensemble d'actions  $A$ , parmi lesquelles l'Agent choisira une action  $a \in A$ . L'Agent influence l'environnement à travers ces actions, et l'environnement peut changer d'état en réponse à l'action de l'agent. La fonction de transition est une distribution de probabilité sur toutes les transitions possibles, autrement dit, en appliquant l'action  $a \in A$  dans un état  $s \in S$ ,  $P(s, a, s')$  est la probabilité de se retrouver dans l'état  $s' \in S$ . Ainsi nous avons pour un espace de temps discret  $t = 1, 2, \dots$ , où  $s_t$  désigne l'état au temps  $t$  et  $s_{t+1}$  désigne l'état au temps  $t + 1$ , l'équation suivante :

$$P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1} \dots) = P(s_{t+1}|s_t, a_t) \quad (1.1)$$

$$= P(s_t, a_t, s_{t+1})/P : S \times A \times S \rightarrow [0, 1] \quad (1.2)$$

- **Reward.** L'environnement a généralement une tâche bien définie et peut fournir à l'agent un signal de récompense en réponse directe aux actions de l'agent. Cette récompense est une rétroaction sur la façon dont la dernière action contribue à la réalisation de la tâche à accomplir par l'environnement. L'objectif de l'agent est de maximiser la récompense globale qu'il reçoit, et donc les récompenses sont la motivation dont l'agent a besoin pour agir dans le comportement souhaité.

$$R : S \times A \times S \rightarrow \mathbb{R}. R(s, a, s') = \mathbb{E}[r_t | s_t = s, a_t = a, s_{t+1} = s'] \quad (1.3)$$

Où  $\mathbb{E}$  désigne "la valeur attendue"

- **Episode.** La séquence d'étapes de temps du début à la fin d'une tâche épisodique est appelée un épisode.

L'apprentissage par renforcement présente quelques concepts supplémentaires au MDP, le transformant en un cadre complet pour aborder les problèmes de décision séquentielle, et ceci principalement en définissant ce qu'il faut optimiser et comment l'optimiser, comme nous le verrons dans ce qui suit :

- **Policy (politique).** Une policy est une stratégie qu'un agent utilise dans la poursuite d'objectifs. La policy dicte les actions que l'agent entreprend en fonction de l'état de l'agent et de l'environnement.
- **Return (rendement).** Les agents peuvent prendre plusieurs pas de temps et épisodes pour apprendre à résoudre une tâche. La somme des récompenses collectées dans un seul épisode s'appelle un retour  $G_t$ . Les agents sont souvent conçus pour maximiser le rendement.

$$G_t = r_{t+1} + r_{t+2} + r_{t+3} + \dots + r_T \quad (1.4)$$

- **Exploration vs. Exploitation.** Une autre caractéristique importante, et un défi dans l'apprentissage par renforcement, est le compromis entre « exploration » et « exploitation ». En essayant d'obtenir de nombreuses récompenses, un agent doit préférer les actions qu'il a essayées dans le passé et sait qu'elles seront des actions efficaces pour produire une récompense. Mais pour découvrir de telles actions, paradoxalement, il doit essayer des actions qu'il n'a jamais sélectionnées auparavant.
- **Value function.** les valeurs indiquent la désirabilité à long terme des états après avoir pris en compte les états susceptibles de suivre et les récompenses (reward) disponibles dans ces états. La fonction de valeur d'un état  $s$  sous une policy  $\pi$ , notée  $v_\pi(s)$ , est le rendement (return) attendu lors du démarrage en  $s$  et par la suite. On peut définir  $v_\pi(s)$  par :

$$v_\pi(s) = \mathbb{E}_\pi[G_t | s_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right] \quad (1.5)$$

Où  $\mathbb{E}_\pi[:]$  désigne la valeur attendue d'une variable aléatoire étant donné que l'agent suit la politique, et  $t$  est n'importe quel pas de temps. De même, une fonction de valeur d'action, également connue sous le nom de fonction  $Q$ , peut être définie par la valeur attendue revenir en partant d'un état  $s$  et en effectuant une certaine action  $a$ , tout en suivant . La fonction valeur d'action,  $q_\pi[:](s, a)$  est donnée par :

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t | s_t = s, a_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\right] \quad (1.6)$$

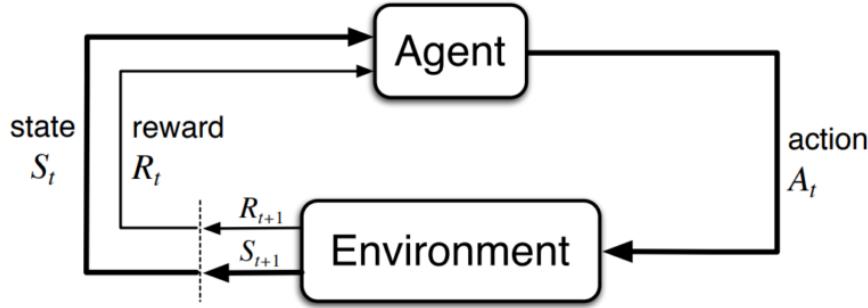


FIGURE 4 – Fonctionnement du RL [13]

#### 1.3.3.d Résoudre le MDP mathématiquement [14]

Afin de résoudre une tâche d'apprentissage par renforcement, l'agent cherche à maximiser le rendement attendu, ce qui se traduit par la maximisation des récompenses reçues sur le long terme. La définition du rendement comme la sommation des récompenses successives permet une relation récursive importante pour la théorie et les algorithmes de l'apprentissage par renforcement. Le rendement peut également être défini par l'équation suivante :

$$\begin{aligned} G_t &= r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots \\ &= r_{t+1} + \gamma(r_{t+2} + \gamma r_{t+3} + \dots) \\ &= r_{t+1} + \gamma G_{t+1} \end{aligned}$$

Semblable au retour, une propriété fondamentale des fonctions de valeur utilisées par les algorithmes du RL est qu'elles satisfont des relations récursives. Pour toute politique  $\pi$  et tout état  $s$ , la condition de cohérence suivante est vérifiée entre la valeur de  $s$  et la valeur de ses éventuels états successeurs :

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi[G_t | s_t = s] \\ &= \mathbb{E}_\pi[r_{t+1} + \dots | s_t = s] \\ &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r | s, a) [r + \gamma \mathbb{E}_\pi[G_{t+1} | s_{t+1} = s']] \\ &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r | s, a) [r + \gamma V(s')] \forall s \in S \end{aligned}$$

L'équation précédente est l'équation de Bellman<sup>3</sup> pour la fonction valeur. Il exprime une relation entre la valeur d'un État et les valeurs de ses États successeurs. Le schéma de la figure suivante explique cette relation où chaque cercle vide représente un état et chaque cercle plein représente une paire état-action.

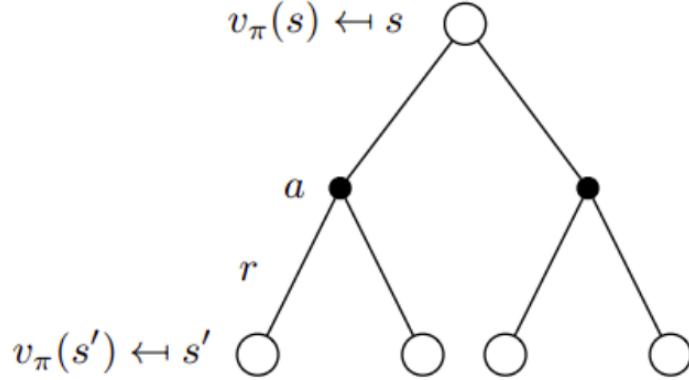


FIGURE 5 – Diagramme de sauvegarde de fonction de valeur [14]

De plus, les fonctions de valeur définissent un ordre partiel sur les politiques. Une politique  $\pi$  est meilleure ou égale à une politique  $\pi'$  si son rendement espéré est supérieur ou égal à celui de  $\pi'$  pour tous les états. En d'autres termes,  $\pi > \pi'$  si et seulement si  $v_\pi > v'_{\pi'}$  pour tous les  $s$  dans  $S$ . Il y a toujours au moins une politique qui est meilleure ou égale à toutes les autres politiques. On note toutes les politiques optimales par  $\pi^*$ . Ils partagent la même fonction de valeur, appelée fonction de valeur optimale définie comme suit :

$$v^*(s) = \max_{\pi} v_{\pi}(s) \quad \forall s \in S \quad (1.7)$$

De même, la fonction de valeur d'action optimale,  $q(s, a)$  peut être définie comme suit :

$$q^*(s, a) = \max_{\pi} q_{\pi}(s) \quad \forall s \in S \quad (1.8)$$

Avec les équations de Bellman, le problème d'apprentissage par renforcement est formulé en termes de relation récursive d'une fonction de valeur. Ainsi, l'équation d'optimalité de Bellman pour le  $v^*$  sera :

---

3. L'équation de Bellman est utilisée pour déterminer la valeur d'un état particulier et en déduire à quel point il est bon d'être dans/prendre cet état. L'état optimal nous donnera la valeur optimale la plus élevée.

$$\begin{aligned}
 v^*(s) &= \max_{\pi} q_{\pi^*}(s) \\
 &= \max_a \mathbb{E}_{\pi^*}[G_t | s_t = s, a_t = a] \\
 &= \max_a \mathbb{E}_{\pi^*}[r_{t+1} + \gamma G_{t+1} | s_t = s, a_t = a] \\
 &= \max_a \mathbb{E}_{\pi^*}[r_{t+1} + \gamma v^*(s') | s_t = s, a_t = a] \\
 &= \max_a \sum_{s', r} p(s', r | s, a) [r + \gamma V(s')]
 \end{aligned}$$

De même, l'équation d'optimalité de Bellman pour  $q^*$  est :

$$\begin{aligned}
 q^*(s, a) &= \mathbb{E}[r_{t+1} + \gamma \max_{a'} q^*(s_{t+1}, a') | s_t = s, a_t = a] \\
 &= \max_a \sum_{s', r} p(s', r | s, a) [r + \gamma \max_{a'} q^*(s', a')]
 \end{aligned}$$

### 1.3.3.e Fonctionnement du RL

L'agent IA doit apprendre à atteindre un objectif au sein d'un environnement incertain et potentiellement complexe. Pour y parvenir, l'ordinateur essaye toutes les façons possibles et apprend de ses erreurs.

À chaque tentative, l'IA reçoit une récompense ou une punition en fonction des actions effectuées. Elle est programmée pour maximiser sa récompense, et tentera donc de trouver la méthode le lui permettant.

Le programmeur se charge de mettre en place les conditions de récompenses. Il se charge donc de fixer les » règles du jeu » .

En revanche, aucune instruction, aucun indice n'est donné à l'agent IA pour lui suggérer comment accomplir la tâche demandée. C'est à lui de découvrir comment maximiser sa récompense, en commençant par des tentatives totalement aléatoires pour terminer par des tactiques extrêmement sophistiquées.

À l'heure actuelle, l'apprentissage par renforcement se révèle comme la façon la plus efficace de faire appel à la créativité des machines. Contrairement à un humain, une intelligence artificielle peut effectuer des milliers de tentatives en simultané. Il suffit pour ce faire de lancer le même algorithme en parallèle sur une puissante infrastructure informatique.

### 1.3.3.f Défis

L'un des défis qui se posent dans l'apprentissage par renforcement, et pas dans d'autres types d'apprentissage, est le compromis entre l'exploration et l'exploitation. Pour obtenir beaucoup de récompense, un agent d'apprentissage par renforcement doit préférer les actions qu'il a essayées dans le passé et jugées efficaces pour produire une récompense. Mais pour découvrir de telles actions, il doit essayer des actions qu'il n'a pas sélectionnées auparavant. L'agent doit exploiter ce qu'il sait déjà afin d'obtenir une récompense, mais il doit également explorer

afin de faire de meilleures sélections d'actions à l'avenir. Le dilemme est que ni l'exploration ni l'exploitation ne peuvent être poursuivies exclusivement sans échouer à la tâche. L'agent doit essayer une variété de actions et privilégier progressivement celles qui paraissent les meilleures. Sur une tâche stochastique, chaque action doit être tentée plusieurs fois pour obtenir une estimation fiable de sa récompense attendue.[13]

### 1.3.3.g Off / On policy RL

Il existe de type d'apprentissage par renforcement, on policy RL, et off policy RL

**Algorithmes On policy.** Les algorithmes d'apprentissage On-Policy sont les algorithmes qui évaluent et améliorent la même policy (politique) qui est utilisée pour sélectionner des actions. Elles résolvent le dilemme exploration vs exploitation en incluant le caractère aléatoire. Ces politiques sont appelées politiques epsilon-greedy<sup>4</sup> car elles sélectionnent des actions aléatoires avec une probabilité epsilon et suivent l'action optimale avec une probabilité de 1-epsilon. Quelques exemples d'algorithmes On-Policy sont Proximal policy optimisation (PPO), Monte Carlo pour On-Policy, Sarsa, etc.

**Algorithmes Off Policy.** Les algorithmes d'apprentissage Off Policy évaluent et améliorent une politique différente de la policy utilisée pour la sélection des actions. Elles offrent une solution différente au problème de l'exploration par rapport à l'exploitation. Alors que les algorithmes on-Policy essaient d'améliorer la même policy epsilon-greedy qui est utilisée pour l'exploration, les approches off policy ont deux types de policy : a behavior policy et target policy. Behavior policy est utilisée pour l'exploration et la génération d'épisodes, et target policy est utilisée pour l'estimation et l'amélioration de la fonction. Quelques exemples d'algorithmes d'apprentissage hors politique sont l'apprentissage Q, la sarsa attendue (peut agir dans les deux sens), etc. [15][16]

### 1.3.3.h Apprentissage par renforcement profond.

L'apprentissage par renforcement (deep reinforcement learning) profond est un apprentissage par renforcement appliqué à l'aide de réseaux de neurones profonds. Ce type d'apprentissage implique que les ordinateurs agissent sur des modèles sophistiqués et examinent un grand nombre d'entrées afin de déterminer un chemin ou une action optimisée.

## 1.4 Conclusion

Dans ce chapitre, nous avons présenté les différents concepts qui concerne la technologie de l'intralogistique et l'automatisation des entrepôts. De plus ,nous avons présenté les véhicules aériens sans pilote (UAV) et leur fonctionnement, et comment ces derniers peuvent aider à l'automatisation des entrepôts. Dans la 2eme section, nous avons exploré les différentes

---

4. Epsilon-greedy :est une méthode simple pour équilibrer l'exploration et l'exploitation en choisissant entre l'exploration et l'exploitation au hasard.

## PARTIE I

---

branches de l'intelligence artificielle, le machine learning, le deep learning, et finalement le deep reinforcement learning.

## CHAPITRE 2

### DÉGITALISATION DES ENTREPÔTS/ DRONES DANS L'ÉTAT DE L'ART

#### 2.1 Introduction

Dans ce chapitre, nous allons voir les différentes réalisations et défis rencontrés dans la recherche en dégitalisation des drone/entrepots, les méthodes utilisées pour la réalisation et simulation des solutions de cet problématique, plus précisément dans la problématique du path planning pour les drones

#### 2.2 Défis

Les défis clés de la mise en œuvre de la technologie des drones dans les entrepôts sont :

**Systèmes de gestion d'entrepôt (WMS) incompatibles** : le WMS existant peut ne pas prendre en charge cette nouvelle technologie, ou certains peuvent avoir des limites de personnalisation pour offrir tous les avantages. Par conséquent, l'adéquation doit être vérifiée avant d'identifier le bon partenaire technologique de drone

**Précision de l'étiquetage** : en raison des différentes tailles de boîtes des unités de gestion des stocks (SKU), les étiquettes ne sont pas uniformément réparties dans une baie de rayonnage ou de rayonnage donnée. Il y a des incohérences dans le collage de ces étiquettes sur chaque palette/boîte/bac, c'est-à-dire qu'elles peuvent ne pas toujours être dans la même direction. Il peut y avoir des problèmes d'impression ou une accumulation de poussière sur ces étiquettes. De tels problèmes peuvent créer des obstacles lors de la numérisation. Cependant, avec les dernières technologies en matière de numérisation, de caméras et de programmes basés sur l'IA/ML, ces défis sont largement relevés, et les drones doivent apprendre à lire ces événements et à les convertir au résultat souhaité.

**Vue de l'inventaire :** la technologie actuelle limite notre vision pour sélectionner la face de la palette/boîte/bac. Par conséquent, cela ne fonctionne bien que pour un stockage profond unique

**Stockage désorganisé :** le stockage désorganisé peut être un défi majeur dans le déploiement de drones. Si le protocole correct de conservation des étiquettes pour faire face aux drones n'est pas maintenu, le drone ne capturera pas les SKU, ce qui entraînera des inexactitudes. Il a également été observé qu'en raison de problèmes d'empilement, les boîtes peuvent se froisser et les étiquettes sont partiellement ou complètement invisibles, ce qui réduit la précision des numérisations. S'il y a des boîtes qui dépassent trop loin dans l'allée, cela peut entraîner des problèmes de sécurité. Avoir une expérience de vol peut aider à réduire de tels cas

**Coût et temps de vol des drones :** les composants nécessaires dans un drone, comme les capteurs et les lasers, coûtent généralement très cher, cependant, d'après les dernières études, il existe des approches reliées à l'intelligence artificielle, qui peuvent réduire les coûts des tâches importantes des drones comme l'utilisation des lasers et la construction en temps réel des cartes des entrepôts, lors de la planification du trajet. De plus, la technologie actuelle des batteries limite le temps de vol des drones. Ce défi peut être relevé par plusieurs drones déployés en séquence, c'est-à-dire que lorsque la batterie est faible, le drone revient pour se recharger et un deuxième drone s'envole pour reprendre l'activité. Une autre alternative consiste à connecter le drone via un câble au véhicule autonome mère, qui se déplace au sol. Cette disposition augmente le temps de déploiement du drone car des véhicules plus grands avec des batteries sont disponibles pour soutenir le drone pour l'alimentation. Il existe également une option pour une plate-forme de chargement sans fil vers laquelle le drone sait revenir. Cela permet un autre niveau d'autonomie qui supprime l'étape de sortir et de remplacer les piles déchargées par une nouvelle.

Avec l'évolution des technologies et des drones de plus en plus intelligents, et en utilisant les techniques de l'intelligence artificielle et machine learning, le cas d'utilisation pourrait devenir une réalité dans un avenir très proche, et chaque grand entrepôt sera vu avec des drones volant à l'intérieur.[17]

## 2.3 UAV Path Planning

### 2.3.1 Description de la problématique

La planification des trajets est le problème le plus important dans l'automatisation des drones dans les entrepôts. Il est défini comme la recherche d'un chemin géométrique de l'emplacement actuel du véhicule à un emplacement cible de sorte qu'il évite tous les obstacles. La planification des trajets est divisée en deux catégories principales basées sur des hypothèses :

1. Les méthodes de planification globale sont des méthodes dans lesquelles le milieu environnant est globalement connu, en supposant la disponibilité d'une carte. Ces méthodes

incluent l'algorithme de feuille de route, la décomposition cellulaire, les diagrammes de Voronoi, les moutures d'occupation et de nouvelles techniques de champ potentiel. Dans la plupart des cas, la dernière étape de la génération de trajectoire consiste à appliquer une courbe de Bézier.

2.Les méthodes de planification locale sont des méthodes dans lesquelles l'environnement est connu localement et peut être reconstruit sur la base de méthodes réactives utilisant des capteurs, tels que des capteurs infrarouges et à ultrasons, et des caméras vidéo locales, sans avoir besoin d'une carte globale. [18]

### 2.3.2 Travaux connexes

Dans cette section nous allons présenter les travaux déjà réalisés reliés à la problématique du path finding pour les robots et les différentes méthodes de simulations des solutions proposées.

Dans cet article [19] les auteurs proposent un nouveau mode d'entraînement incrémental pour résoudre le problème de planification de trajectoire basé sur l'apprentissage par renforcement profond (DRL) pour un robot mobile. Premièrement, ils évaluent des différents algorithmes du DRL dans un environnement 2D en le simulant avec la librairie Gym. transférons l'algorithme conçu à un environnement 3D simple et complexe ensuite,et en utilisant le simulateur 3D Gazebo en utilisant des capteurs, et sans l'aide d'une carte globale, pour se recycler afin d'obtenir les paramètres du réseau convergé. La méthode proposée et la combinaison entre l'algorithme DRL TD3 avec l'algorithme de planification de chemin global traditionnel PRM. Les résultats expérimentaux montrent que Le mode d'entraînement peut notamment améliorer l'efficacité du développement. De plus, le planificateur de trajectoire PRM+TD3 peut effectivement améliorer la généralisation du modèle. Cependant, Les résultats de chaque planification faite sont différents et peuvent ne pas être optimaux.

Les auteurs dans l'article [20], avec un même objectif que l'article précédent, présentent un planificateur de mouvement sans carte basé sur le DRL prenant les résultats des capteurs de plage 10 dimensions clairsemés et la position cible par rapport au cadre de coordonnées du robot mobile comme entrée et les commandes de direction continues comme sortie,et ceci en utilisant l'algorithme DDPG.Les expériences montrent que le planificateur de mouvement sans carte proposé peut diriger le robot mobile vers les cibles souhaitées sans entrer en collision avec des obstacles.

Dans ce travail [21], les auteurs montrent la faisabilité d'appliquer des méthodes d'apprentissage par renforcement, plus précisément, l'algorithme Proximal Policy Optimization (PPO), pour optimiser une politique de contrôle stochastique, afin d'effectuer le contrôle de position du quadrirotor<sup>1</sup>.Les expériences résultant ont été réalisées à l'aide du simulateur V-REP et du moteur physique Vortex, et ils ont présenté une bonne efficacité d'échantillonnage.Cependant, les auteurs visent à explorer de nouveaux signaux de récompense et à transférer des idées de la théorie classique du contrôle pour réduire l'erreur d'état stable

---

1. quadrirotor : Un quadrirotor est un aéronef à voilure tournante comportant quatre rotors pour sa sustentation [22]

observée. De même, Wang et Li [23], ont appliquer le PPO pour le problème du path finding pour les drones en utilisant un algorithme déjà entraîné, et en simulant l'environnement dans une plate-forme générale de simulation de combat.

De même que l'article précédent,l'article [24] décrive un algorithme de navigation générique qui utilise les données des capteurs à bord du drone pour guider le drone vers le site du problème dans les situations dangereuses avec précision et rapidité. La méthode utilisée est l'algorithme d'apprentissage par renforcement profond PPO, l'apprentissage incrémental du curriculum et aux réseaux de neurones à mémoire à court terme, et en déterminant le meilleur modèle a appliquer pour les environnements simples et complexes.L'outil de simulation utilisé et le Framework Unity 3D ML-agents. Parmis les limitations de cet article, l'utilisation d'un espace d'actions discret, qui limite les actions à faire par l'agent.

L'article suivant [25] propose une méthode qui améliore l'algorithme DDPG classique pour répondre à la tâche de navigation sans carte d'un seul robot. En étendons également l'algorithme à robot unique au système multi-robot et obtenons le Parallel Deep Deterministic Policy Gradient (PDDPG). En utilisant un capteur lidar<sup>2</sup>D. Les résultats de l'expérience dans la plate-forme de simulation 3D Gazebo illustrent que la méthode est capable de guider des robots mobiles pour construire la formation et conserver la formation pendant la navigation de groupe, directement via des entrées de données lidar brutes.

Cette étude[27] développe une politique de mobilité des robots basée sur le RL (DDPG). L'état de vue est obtenu par les capteurs d'entrée, le but est de naviguer vers l'objectif sans aucune collision, ce travail tente de construire un système de détection de pseudo-laser basé sur la prédiction directe de matrice de profondeur à partir d'une seule image de caméra tout en conservant des performances stables. Les résultats des expériences montrent qu'ils sont directement comparables à d'autres utilisant des capteurs coûteux.

Ce projet [28] aussi montre qu'un système peut naviguer en utilisant uniquement les données du capteur et la position du système par rapport à une cible. La méthode repose sur le système d'exploitation du robot,l'algorithme PPO, la librairie ROS et Gazebo pour construire un véhicule d'essai, tester la méthode proposée et la comparer à une méthode de navigation de pointe dans un scénario réel. L'évaluation a montré que la méthode proposée est surpassée par les solutions basées sur SLAM, en termes de précision de pose, de précision de localisation et de fiabilité pour atteindre la cible souhaitée. Cependant, la procédure développée offre un potentiel dans les cas où la cartographie préalable n'est pas possible et la capacité d'exploration d'un véhicule est requise.

Étant donné que les méthodes traditionnelles de navigation robotique conventionnelle dépendent de la reproduction précise des cartes et nécessitent des capteurs haut de gamme, les

---

2. Lidar :Light Detection And Ranging. Il s'agit d'une méthode de calcul qui permet de déterminer la distance entre le capteur et l'obstacle visé en utilisant la vitesse de la lumière sortant du capteur[26]

## PARTIE I

---

méthodes basées sur l'apprentissage sont des tendances positives, en particulier l'apprentissage par renforcement en profondeur ( notamment les algorithmes PPO et DDPG). Dans ce projet, nous voulant comparer entre ces deux algorithmes en réalisant un simulateur 3d avec Gazebo, d'un autre coté, et d'après l'article [27], nous allons étudier l'efficacité de la caméra monoculaire et la construction synchrone des images en profondeur par rapport à un laser, dans notre propre environnement d'entrepôt, en essayons de faire remplacer les cartes et les capteurs.

Article	Problématique	Méthode utilisé	Défi
Article [19]	Plannification de trajectoire sans carte	TD3+ PRM	entraînement incrémental, Le design du trajectoire
Article [20]	Plannification de trajectoire sans carte	DDPG + ADDPG	Espace d'actions continus + Teste dans le monde réel
Article [21]	Plannification de trajectoire sans carte	PPO + V-REP simulator	Utilisation de DRL model-free
Article [24]	Plannification de trajectoire sans carte	PPO + Unity 3D ML-agents	entraînement incrémental
Article [25]	Plannification de trajectoire sans carte	PDDGP + 3D Gazebo	système multi-robot
Article [27]	Plannification de trajectoire sans carte	DDPG + Caméra monoculaire	Ne pas utiliser un laser
Article [28]	Plannification de trajectoire sans carte	PPO + Gazebo	Réalisation d'un scénario réel

TABLE I – *Travaux connexes*

## 2.4 Conclusion

Nous avons présenté dans ce chapitre une étude de l'existant sur la problématique de planification du trajectoire pour les drone. Dans la partie suivante, nous allons modéliser et formuler la problématique, en définissant les méthodes qui seront utilisé pour la réalisation.

## Deuxième partie

# Conception, implémentation et évaluation de la solution

# CHAPITRE 1

## CONCEPTION DE LA SOLUTION

### 1.1 Introduction

Plusieurs limitations et défis dans l'automatisation des entrepôts peuvent être résolus en intégrant l'apprentissage par renforcement profond. Un cas d'utilisation important est l'utilisation des drones pour la gestion d'inventaire, et ceci ne peut pas être réalisé sans avoir passer par la problématique de la planification des trajectoires des robots mobiles, et les drones en particulier. Dans ce chapitre, nous allons présenter et expliquer l'architecture de notre solution de cette problématique, et proposer une formulation MDP qui permet ensuite de la résoudre avec des algorithmes du DRL.

### 1.2 Vue globale sur la solution

Notre solution consiste à réaliser un système DRL qui a pour but de guider l'agent (le drone) vers la position cible en évitant tout les obstacles auxquels il sera confronté pendant le trajet sans l'utilisation d'une carte globale, et ceci en choisissant un algorithme on-policy et un autre off-policy, et en faisant une étude comparative pour sélectionner le meilleur algorithme pour cette problématique, d'un autre côté, étudier l'efficacité de la Monocular camera, et sa capacité de pouvoir remplacer les lasers dans les drones. L'apprentissage par renforcement supprime la restriction qu'un ensemble de données doit être fourni, contrairement à d'autres techniques du ML bien connues. Cela fait de l'apprentissage par renforcement la solution idéale pour les problèmes où les ensembles de données sont soit indisponibles, soit coûteux. Malgré cela, les agents RL ont encore besoin d'interagir avec leur environnement et de recueillir des expériences (sous forme de données) afin de découvrir le plan d'action optimal. Une stratégie alternative consiste à créer un environnement virtuel qui répond aux actions de l'agent en lui donnant la prochaine observation et récompense en conséquence, permettant à l'agent d'effectuer autant d'itérations dans l'environnement que la ressource de calcul le permet. Interagir avec un environnement réel peut être lent, dangereux et très coûteux. Cela augmente l'efficacité, la sécurité et l'efficacité de divers prototypes de système RL.

## 1.3 Modélisation de la problématique

L'objectif de cette étude est la réalisation d'un modèle de simulation 3D du Mapless path planning, i.e la problématique de la planification de la trajectoire sans l'utilisation d'une map globale, mais d'un algorithme de détection local, qui pourra être testé sur un environnement dynamique, avec des obstacles et des positions cibles dynamiques, pour cela, on modélise le problème sous forme d'un MDP défini comme suie :

### 1.3.1 Agent :

L'agent représente le drone (un robot mobile)

### 1.3.2 État (input) :

Chaque état est composé de trois facteurs, à savoir l'observation obtenue du capteur d'entrée, les valeurs en coordonnées polaires représentent la position de la cible, et l'action effectuée précédemment. Donc on peut représenter l'état comme un espace de 16 dimensions contenant les informations suivantes :

- Le retour du Laser (10 Dimensions), les résultats clairsemés de la portée laser sont échantillonnés à partir des résultats bruts du laser entre -90 et 90 degrés dans une distribution triviale et à angle fixe. Les informations de plage sont normalisées à (0,1)
- Dernière action effectuée (Vitesse linéaire et vitesse angulaire) (2 Dimensions)
- La Position de la cible au niveau du robot (2 Dimensions) : distance relative et Angle relatif (Coordonnées polaires)
- Robot yaw angular (1 Dimensions)<sup>1</sup>
- Les degrés pour faire face à la cible i.e. |Yaw - Angle relatif| (1 Dimensions)

### 1.3.3 Action(Output) :

- Linear velocity (0 1 m/s) (1 Dimensions)
- Angular velocity (-0.5 0.5 rad/s) (1 Dimensions)

---

1. Yaw : vitesse de variation de l'angle de cap lorsque le robot est à l'horizontale. Il est généralement mesuré en degrés par seconde ou en radians par seconde.

Angular velocity	Linear velocity
La vitesse angulaire est définie comme le taux de changement de la position angulaire d'un corps en rotation.	La vitesse linéaire est définie comme le taux de changement de déplacement par rapport au temps lorsque l'objet se déplace le long d'une trajectoire rectiligne.
Lorsqu'un objet subit un mouvement circulaire, la vitesse angulaire d'une particule est le long de l'axe d'un cercle. La vitesse angulaire reste la même.	Lorsqu'un objet subit un mouvement circulaire, la vitesse linéaire d'une particule est le long de la circonférence d'un cercle. La vitesse linéaire varie en chaque point du cercle.
La vitesse angulaire est mesurée en degrés et en radians.	La vitesse linéaire est mesurée en m/s.
La vitesse angulaire est représentée par	La vitesse linéaire est représentée par v.

TABLE II – *Angular vs Linear velocity [29]*

### 1.3.4 Environnement :

La procédure d'apprentissage de notre modèle a été implémentée dans des environnements virtuels simulés par le simulateur 3d Gazebo [30]. Nous avons construit deux environnements pour montrer l'influence de l'environnement d'entraînement sur les résultats du modèle.. Les deux modèles de ces deux environnements ont été appris à partir de zéro. Un bot est utilisé comme plate-forme robotique. La cible est représentée par un objet carré. Dans chaque épisode, la position cible a été initialisée de manière aléatoire dans toute la zone de sorte qu'elle ne sera pas équivalente à une position d'obstacle.

Les fonctions principales de l'environnement RL sont les fonctions : init(), step(), reset() :

- **init()** : Dans le constructeur init() , nous définissons nos espaces d'état et d'action selon notre modèle MDP, l'initialisation des positions et odométries<sup>2</sup> du robot et les positions cibles, l'initialisation des services du ROS.
- **step()** : La méthode step() peut être considérée comme le cœur de tout environnement RL, c'est là que l'action de l'agent est reçue et appliquée à la simulation à cette instance de l'épisode. Enfin, le nouvel état de l'environnement est renvoyé à l'agent avec une récompense scalaire. Cela signifie la fin d'une étape et d'une expérience en cours d'acquisition.
- **reset()** : Une méthode reset() est nécessaire pour réinitialiser l'environnement à la fin de chaque épisode à l'état initial pour que l'agent recommence. En outre, dans de nombreuses implémentations RL , les retours de l'agent ne sont calculés qu'après l'exécution de cette fonction, comme il l'indique la fin d'un épisode.

2. L'odométrie est l'utilisation des données des capteurs de mouvement pour estimer le changement de position dans le temps. Il est utilisé en robotique par certains robots à pattes ou à roues pour estimer leur position par rapport à un emplacement de départ. Cette méthode est sensible aux erreurs dues à l'intégration des mesures de vitesse dans le temps pour donner des estimations de position.

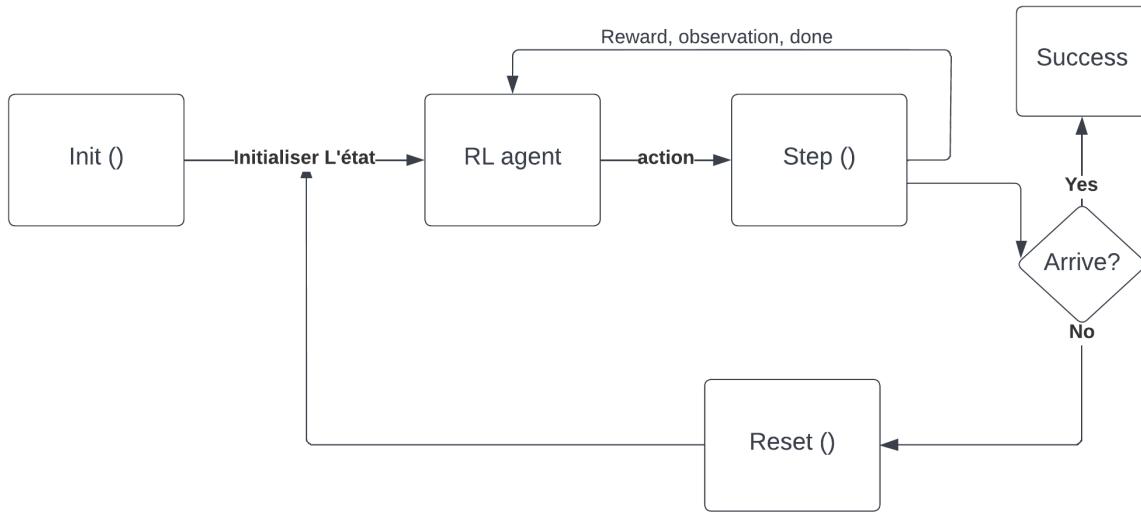


FIGURE 6 – Architecture de l’environnement RL

### 1.3.5 Calcul du Reward :

Afin de renforcer l'action, à chaque pas de temps, une récompense  $r$  est définie, en définissant la fonction du reward comme suit :

$$\text{reward} = \begin{cases} r_{\text{success}} & \text{if } d_{t+1} \leq \text{seuil} \\ r_{\text{done}} & \text{if } \text{done} \\ c_r(d_t - dt + 1), & \text{otherwise} \end{cases} \quad (1.1)$$

$r_{\text{success}}$  est la récompense obtenue lorsque la distance du robot vers la cible à  $t+1$  est inférieure au seuil donné  $\text{seuil}$ . En revanche, la collision est la rétroaction négative à recevoir lorsque le robot est entré en collision avec un obstacle  $e$ . Si aucun des cas ci-dessus est concerné, la récompense sera la soustraction de la distance jusqu'au point de destination le plus tôt et celui ensuite multiplié par un hyper-paramètre à affiner  $c_r$ . Cela motive le robot à se rapprocher de la position cible durant la planification.

## 1.4 Apprentissage par renforcement profond

Après avoir fait une étude de l'existant dans la première partie, beaucoup d'articles ont été intéressés par le nouveau algorithme PPO qui est un algorithme On-policy, à cause de la simplicité de son implémentation d'une part, et nécessitant, d'autre part, moins de données que les algorithmes Off-policy, ayant une complexité meilleure. D'un autre côté, plusieurs chercheurs ont opté pour l'algorithme Off-policy DDPG ayant des résultats meilleurs en moins de temps. Dans cette section, nous allons présenter les deux algorithmes en expliquant

le fonctionnement de chacun d'eux, et en jetons d'abord un coup d'oeil sur la méthode actor-critic qui sera utilisée dans les deux algorithmes.

### 1.4.1 Le réseaux Actor-Critic

L'idée principale est de diviser le modèle du DRL en deux : un pour calculer une action basée sur un état et un autre pour produire les valeurs Q de l'action.

L'acteur prend en entrée l'état et produit la meilleure action. Il contrôle essentiellement le comportement de l'agent en apprenant la politique optimale (basée sur la politique). Le critique, quant à lui, évalue l'action en calculant la fonction de valeur (basée sur la valeur). Ces deux modèles participent à un jeu où ils s'améliorent tous les deux dans leur propre rôle au fil du temps. Le résultat est que l'architecture globale apprendra à jouer le jeu plus efficacement que les deux méthodes séparément.

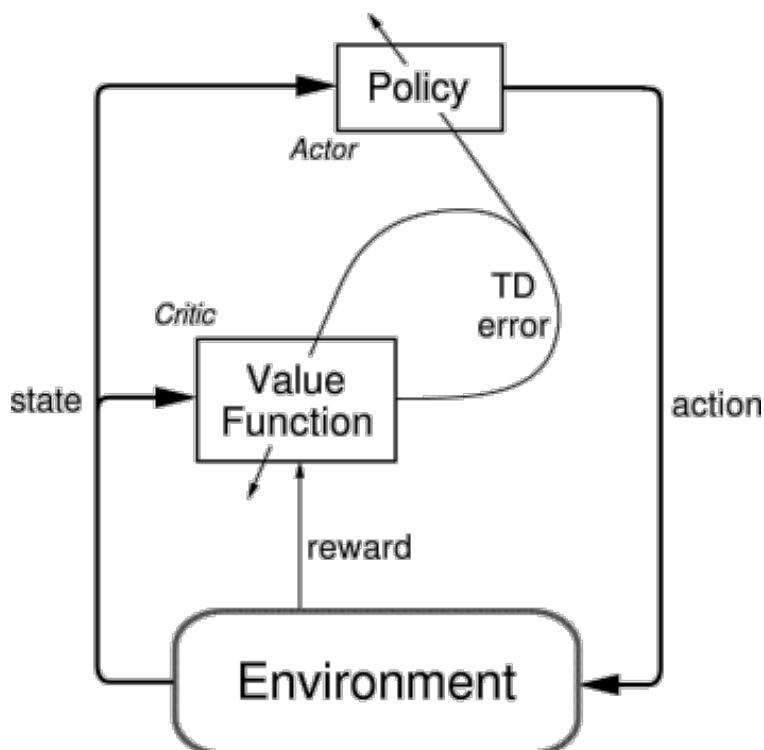


FIGURE 7 – Actor-Critic

L'acteur peut être un approximateur de fonction comme un réseau de neurones et sa tâche est de produire la meilleure action pour un état donné. Le critique est un autre approximateur de fonction, qui reçoit en entrée l'environnement et l'action de l'acteur, les concatène et produit la valeur d'action (valeur Q) pour la paire donnée.

La formation des deux réseaux est effectuée séparément et utilise le gradient ascent<sup>3</sup> (pour

3. Gradient ascent : est basée sur le principe de la localisation du plus grand point sur une fonction, puis du déplacement dans la direction du gradient.

trouver le maximum global et non le minimum) pour mettre à jour leurs deux poids. Au fil du temps, l'acteur apprend à produire des actions de mieux en mieux (il commence à apprendre la politique) et le critique évalue de mieux en mieux ces actions. Il est important de noter que la mise à jour des pondérations se fait à chaque étape (TD Learning) et non en fin d'épisode, contrairement aux gradients politiques.

Les actor-critic se sont avérés capables d'apprendre des environnements vastes et complexes et ils les ont utilisés dans de nombreux jeux 2D et 3D célèbres, tels que Doom, Super Mario et d'autres[31].

#### Architecture :

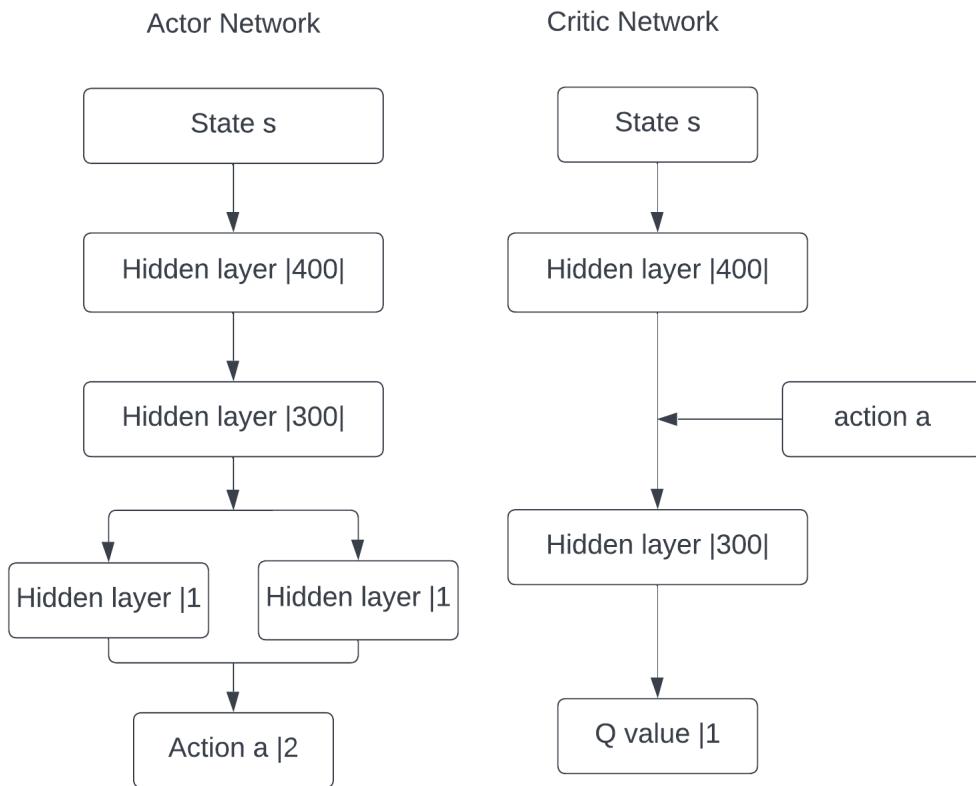


FIGURE 8 – *Actor-Critic Architecture*

#### 1.4.2 La méthode On-policy (PPO)

PPO est une méthode de gradient de politique<sup>4</sup> et peut être utilisée pour des environnements avec des espaces d'action discrets ou continus. Il forme une politique stochastique

4. Les méthodes de gradient de politique sont un type de techniques d'apprentissage par renforcement qui reposent sur l'optimisation de politiques paramétrées par rapport au rendement attendu par descente de gradient.

d'une manière on-policy. En outre, il utilise la méthode actor-critic. L'acteur fait correspondre l'observation à une action et le critique donne une attente des récompenses de l'agent pour l'obervation donnée. Premièrement, il collecte un ensemble de trajectoires pour chaque époque en échantillonnant à partir de la dernière version de la politique stochastique. Ensuite, les récompenses à emporter et les estimations des avantages sont calculées afin de mettre à jour la politique et d'ajuster la fonction de valeur. La politique est mise à jour via un optimiseur de montée de gradient stochastique, tandis que la fonction de valeur est ajustée via un algorithme de descente de gradient. Cette procédure est appliquée à de nombreuses époques jusqu'à ce que l'environnement soit résolu. [32]

**Architecture :**

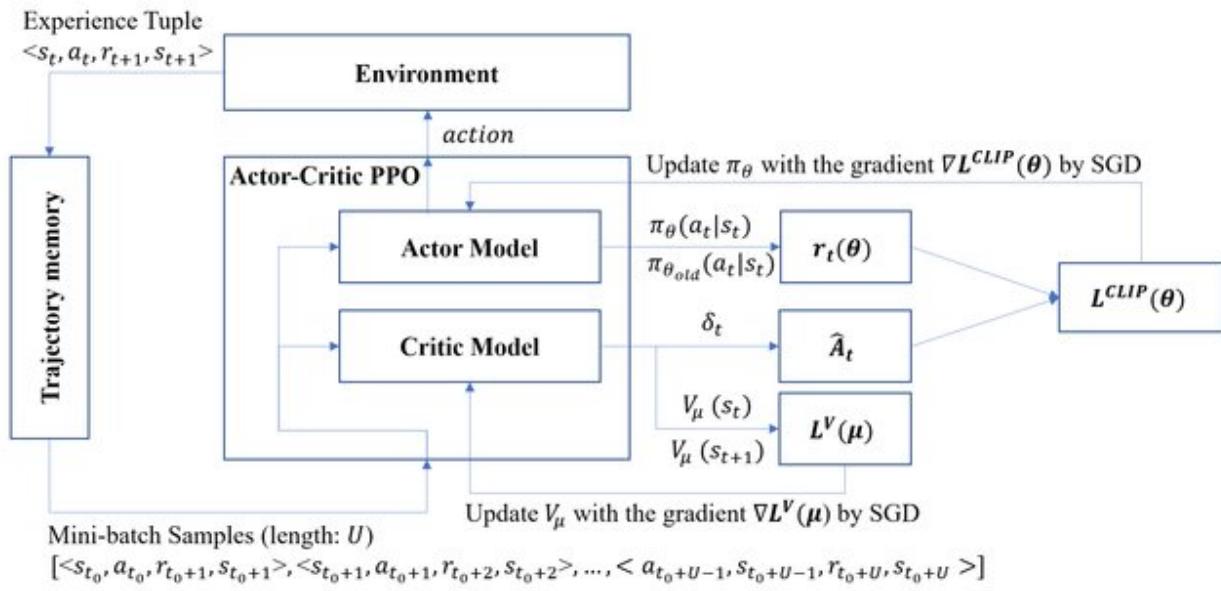


FIGURE 9 – Architecture du PPO [33]

**Pseudo-Code :**

---

**Algorithm 1** PPO-Clip

---

- 1: Input: initial policy parameters  $\theta_0$ , initial value function parameters  $\phi_0$
- 2: **for**  $k = 0, 1, 2, \dots$  **do**
- 3:   Collect set of trajectories  $\mathcal{D}_k = \{\tau_i\}$  by running policy  $\pi_k = \pi(\theta_k)$  in the environment.
- 4:   Compute rewards-to-go  $\hat{R}_t$ .
- 5:   Compute advantage estimates,  $\hat{A}_t$  (using any method of advantage estimation) based on the current value function  $V_{\phi_k}$ .
- 6:   Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg \max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \min \left( \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

typically via stochastic gradient ascent with Adam.

- 7:   Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg \min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \left( V_{\phi}(s_t) - \hat{R}_t \right)^2,$$

typically via some gradient descent algorithm.

- 8: **end for**
- 

FIGURE 10 – *Pseudo code du PPO* [32]

### 1.4.3 La méthode Off-policy (DDPG)

DDPG est un algorithme off-policy, utilisé pour l'apprentissage d'actions continues. Il combine les idées de DPG (Deterministic Policy Gradient) et DQN (Deep Q-Network). Il utilise Experience Replay<sup>5</sup> et les réseaux cibles à apprentissage lent de DQN, et il est basé sur DPG, qui peut fonctionner sur des espaces d'action continus.

#### 1.4.3.a Deterministic Policy Gradient(DPG)

Traditionnellement, les algorithmes de gradient de politique sont utilisés avec la fonction de politique stochastique . Cela signifie que la fonction de politique est représentée comme une distribution sur les actions. Pour un état donné, il y aura une distribution de probabilité pour chaque action dans l'espace d'action. Dans DPG, au lieu de la politique stochastique, , la politique déterministe  $\mu$  est suivie. Pour un état donné,  $s$ , il y aura une décision déterministe  $a=\mu(s)$  au lieu de répartition sur les actions.

#### 1.4.3.b Q-learning

Le Q-learning est un algorithme d'apprentissage basé sur la valeur et se concentre sur l'optimisation de la fonction de valeur en fonction de l'environnement ou du problème. Le Q

---

5. Experience Replay est une technique de mémoire de relecture utilisée dans le RL où nous stockons les expériences de l'agent à chaque pas de temps (state, reward, action, next state), dans un ensemble de données, regroupées sur de nombreux épisodes dans une mémoire de relecture.

dans le Q-learning représente la qualité avec laquelle le modèle trouve sa prochaine action améliorant la qualité. Le processus peut être automatique et simple. Le modèle stocke toutes les valeurs dans un tableau, qui est le tableau Q. En d'autres termes, il utilise la méthode d'apprentissage pour trouver la meilleure solution passant par les étapes suivantes : Initialisation, Exploration ou exploitation, Mesurer la récompense, Mise à jour du tableau Q, . [34]

---

**Algorithm 1:** Q-Learning

---

**Result:** Off-policy control for estimating  $\pi \simeq \pi_*$   
 Parameters: step size  $\alpha \in (0, 1]$ , small  $\epsilon > 0$   
 Initialize  $Q(s, a) \forall s \in \mathcal{S}^+, a \in \mathcal{A}(s)$  arbitrarily.  
 Set  $Q(\text{terminal}, \cdot) = 0$   
**for** each episode **do**  
     Initialize  $S$ ;  
     **for** step = 0 **to**  $T$  **do**  
         Choose  $A$  from  $S$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy);  
         Take action  $A$ , observe  $R, S'$ ;  
          $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$ ;  
          $S \leftarrow S'$ ;  
     **end**  
**end**

---

FIGURE 11 – *Q-learning pseudo code*[35]

#### 1.4.3.c Deep Q-Network

Dans Q-Learning, nous représentons la valeur Q sous forme de tableau. Cependant, dans de nombreux problèmes du monde réel, il existe d'énormes espaces d'état et/ou d'action et la représentation tabulaire est insuffisante. Par exemple, Computer Go a  $10^{170}$  états et des jeux comme Mario Bro ont un espace d'état continu. Lorsqu'il est impossible de stocker toutes les combinaisons possibles de valeurs de paires d'état et d'action dans le tableau 2-D ou la table Q, nous devons utiliser Deep Q-Network (DQN) au lieu de l'algorithme Q-Learning.

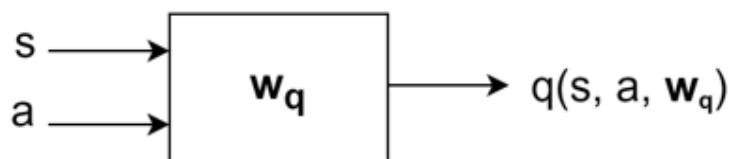


FIGURE 12 – *DQN state et action* [36]

DQN est également un algorithme RL sans modèle où la technique moderne d'apprentissage en profondeur est utilisée. Les algorithmes DQN utilisent le Q-learning pour apprendre la

meilleure action à entreprendre dans l'état donné et un réseau neuronal profond ou un réseau neuronal convolutif pour estimer la fonction de valeur Q. [36]

---

**Algorithm 1 Deep Q-learning with Experience Replay**

---

```

Initialize replay memory  $\mathcal{D}$  to capacity  $N$ 
Initialize action-value function  $Q$  with random weights
for episode = 1,  $M$  do
    Initialise sequence  $s_1 = \{x_1\}$  and preprocessed sequenced  $\phi_1 = \phi(s_1)$ 
    for  $t = 1, T$  do
        With probability  $\epsilon$  select a random action  $a_t$ 
        otherwise select  $a_t = \max_a Q^*(\phi(s_t), a; \theta)$ 
        Execute action  $a_t$  in emulator and observe reward  $r_t$  and image  $x_{t+1}$ 
        Set  $s_{t+1} = s_t, a_t, x_{t+1}$  and preprocess  $\phi_{t+1} = \phi(s_{t+1})$ 
        Store transition  $(\phi_t, a_t, r_t, \phi_{t+1})$  in  $\mathcal{D}$ 
        Sample random minibatch of transitions  $(\phi_j, a_j, r_j, \phi_{j+1})$  from  $\mathcal{D}$ 
        Set  $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$ 
        Perform a gradient descent step on  $(y_j - Q(\phi_j, a_j; \theta))^2$  according to equation 3
    end for
end for

```

---

FIGURE 13 – Deep Q Network pseudo code[35]

#### 1.4.3.d Deep Deterministic Policy Gradient(DDPG)

DDPG est un algorithme actor critic off policy free model qui combine le Deep Q Learning (DQN) et le DPG. DQN original fonctionne dans un espace d'action discret et DPG l'étend à l'espace d'action continu tout en apprenant une politique déterministe.

Comme il s'agit d'un algorithme hors politique, il utilise deux politiques distinctes pour l'exploration et les mises à jour. Il utilise une politique de comportement stochastique pour l'exploration et une politique déterministe pour la mise à jour de la cible.

DDPG a deux réseaux : acteur et critique. Techniquement, l'acteur produit l'action à explorer. Pendant le processus de mise à jour de l'acteur, l'erreur TD d'un critique est utilisée. Le réseau critique est mis à jour en fonction de l'erreur TD similaire à la règle de mise à jour Q-learning.

Nous avons appris le problème d'instabilité qui peut survenir dans le Q-learning avec le réseau de neurones profonds comme approximateur fonctionnel. Pour résoudre ce problème, la relecture de l'expérience et les réseaux cibles sont utilisés.

**Architcture :**

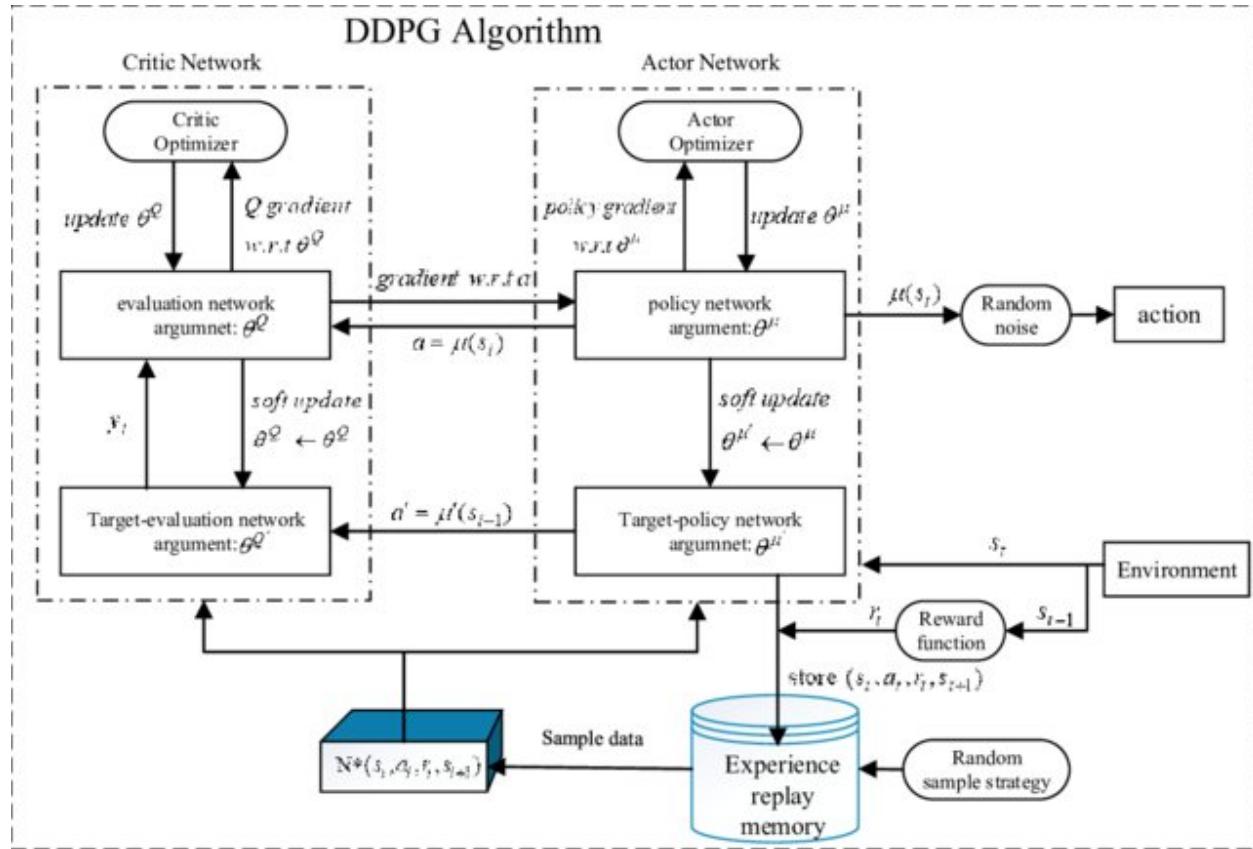


FIGURE 14 – Architecture du DDPG[37]

Pseudo-Code :

---

**Algorithm 1** DDPG algorithm

---

Randomly initialize critic network  $Q(s, a|\theta^Q)$  and actor  $\mu(s|\theta^\mu)$  with weights  $\theta^Q$  and  $\theta^\mu$ .  
 Initialize target network  $Q'$  and  $\mu'$  with weights  $\theta^{Q'} \leftarrow \theta^Q, \theta^{\mu'} \leftarrow \theta^\mu$   
 Initialize replay buffer  $R$   
**for** episode = 1, M **do**  
 Initialize a random process  $\mathcal{N}$  for action exploration  
 Receive initial observation state  $s_1$   
**for** t = 1, T **do**  
 Select action  $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$  according to the current policy and exploration noise  
 Execute action  $a_t$  and observe reward  $r_t$  and observe new state  $s_{t+1}$   
 Store transition  $(s_t, a_t, r_t, s_{t+1})$  in  $R$   
 Sample a random minibatch of  $N$  transitions  $(s_i, a_i, r_i, s_{i+1})$  from  $R$   
 Set  $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$   
 Update critic by minimizing the loss:  $L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$   
 Update the actor policy using the sampled policy gradient:  

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$
  
 Update the target networks:  

$$\theta^{Q'} \leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q'}$$
  

$$\theta^{\mu'} \leftarrow \tau \theta^\mu + (1 - \tau) \theta^{\mu'}$$
  
**end for**  
**end for**

---

FIGURE 15 – Pseudo code du DDPG [38]

Explication :

---

**Algorithm 1** DDPG algorithm

```

Randomly initialize critic network  $Q(s, a|\theta^Q)$  and actor  $\mu(s|\theta^\mu)$  with weights  $\theta^Q$  and  $\theta^\mu$ .
Initialize target network  $Q'$  and  $\mu'$  with weights  $\theta^{Q'} \leftarrow \theta^Q$ ,  $\theta^{\mu'} \leftarrow \theta^\mu$  Use target networks to do off-policy updates
Initialize replay buffer  $R$ 
for episode = 1, M do
    Initialize a random process  $\mathcal{N}$  for action exploration
    Receive initial observation state  $s_1$  Action selected using Deterministic Actor as explained before
    for t = 1, T do
        Select action  $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$  according to the current policy and exploration noise
        Execute action  $a_t$  and observe reward  $r_t$  and observe new state  $s_{t+1}$ 
        Store transition  $(s_t, a_t, r_t, s_{t+1})$  in  $R$ 
        Sample a random minibatch of  $N$  transitions  $(s_i, a_i, r_i, s_{i+1})$  from  $R$  Experience Replay
        Set  $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$ 
        Update critic by minimizing the loss:  $L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$ 
        Update the actor policy using the sampled policy gradient:
            
$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$
 Policy Gradient
        Update the target networks:
            
$$\theta^{Q'} \leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q'}$$

            
$$\theta^{\mu'} \leftarrow \tau \theta^\mu + (1 - \tau) \theta^{\mu'}$$
  $\tau \ll 1$   
Slow updates,  
but highly stable
    end for
end for

```

---

FIGURE 16 – *Explication du DDPG [38]*

#### 1.4.4 Détection des obstacles

Nous utilisons deux méthodes afin de détecter la collision entre l'agent et l'obstacle, et nous comparant entre les résultats dans la partie évaluation de la solution :

- L'utilisation d'un Laser 2D de 10 gammes
- L'utilisation d'une Monocular Depth Camera

##### 1.4.4.a Détection des obstacles en utilisant un Laser

laser, "light amplification by the stimulated emission of radiation" un dispositif qui stimule les atomes ou les molécules pour émettre de la lumière à des longueurs d'onde particulières et amplifie cette lumière, produisant généralement un faisceau de rayonnement très étroit. L'émission couvre généralement une gamme extrêmement limitée de longueurs d'onde visibles, infrarouges ou ultraviolettes.

Les scanners à distance laser sont utilisés pour la détection d'obstacles depuis un certain nombre d'années, en particulier pour la navigation en terrain accidenté. Les scanners à distance laser fonctionnent en balayant un laser sur une région d'intérêt et en mesurant, à chaque pixel, le temps nécessaire au laser pour quitter et revenir au capteur. Puisque la

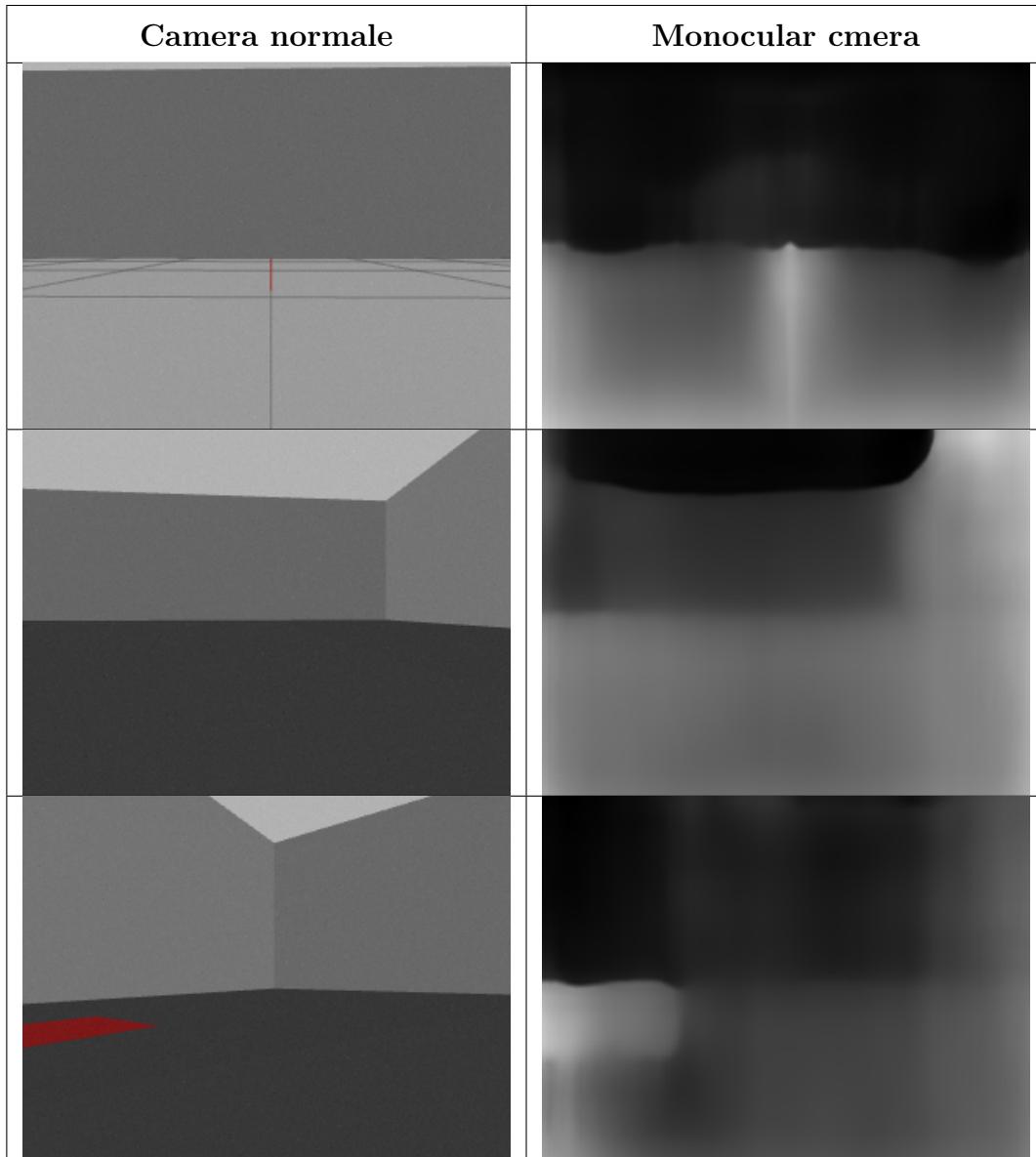
vitesse de la lumière est connue, la distance à chaque pixel peut être calculée. La plupart des scanners à distance laser fournissent également l'intensité du signal renvoyé à chaque pixel. Cependant, cette deuxième information, souvent appelée réflectance, a été essentiellement ignorée par la plupart des chercheurs.[39]

### 1.4.4.b Détection des obstacles en utilisant l'estimation de profondeur monoculaire auto-supervisée

L'estimation de la profondeur monoculaire est une autre méthode utilisée dans ce projet pour détecter les obstacles, elle consiste à estimer la valeur de la profondeur (distance par rapport à la caméra) de chaque pixel pour une seule image RVB (monoculaire). Cette tâche difficile est une condition préalable essentielle pour déterminer la compréhension de la scène pour des applications telles que la reconstruction de scène 3D, la conduite autonome et la réalité augmentée. Les méthodes de pointe appartiennent généralement à l'une des deux catégories suivantes : concevoir un réseau complexe suffisamment puissant pour régresser directement la carte de profondeur, ou diviser l'entrée en bacs ou fenêtres pour réduire la complexité de calcul. Les benchmarks les plus populaires sont les jeux de données KITTI et NYUv2. Les modèles sont généralement évalués à l'aide de la RMSE ou de l'erreur relative absolue.[40]

Nous utilisons le modèle UNet dans [41] pour prédire directement les images de profondeur à partir d'images monocularies. Cet UNet fournit des fonctionnalités approfondies ainsi que des informations locales sur les images. En fait, l'encodeur utilise le modèle ResNet-18 composé de 11 millions de paramètres tandis que le décodeur inclut une convolution de suréchantillonnage avec fonction d'activation ELU dans chaque couche et fonction sigmoïde à la dernière couche pour obtenir la carte de disparité D. L'ensemble UNet est également pré-formés sur le jeu de données KITTI.

Le tableau suivant présente des images exemplaires sur le retour d'une caméra normale vs le retour d'une monocular caméra testées dans l'environnement de simulation de notre projet :

TABLE III – *Retour d'une camera normale vs Monocular Camera*

## 1.5 Conclusion

Dans ce chapitre, nous avons modéliser la problématique sous forme d'un MDP, présenter les algorithmes qu'on va utiliser pour la réalisation de la solution et leurs architecture, et enfin parler sur l'utilisation des laser et l'utilisation du monocular camera pour la détection des obstacles.Dans le chapitre suivant, discuter l'implémentation de la solution.

## CHAPITRE 2

### IMPLÉMENTATION DE LA SOLUTION

#### 2.1 Introduction

Dans ce chapitre, nous allons présenter l'environnement du travail, définissant les différents logiciels, librairies et matériel utilisé pour la réalisation de la solution modélisée dans les chapitres précédents.

#### 2.2 Outils d'Implémentation

##### 2.2.1 Matériel

Le tableau suivant présente les informations nécessaire sur la machine utilisée pour le travail :

Nom	Description
RAM	16.0 GiB
Processeur	Intel® Core™ i7-8650U CPU @ 1.90GHz × 8
Graphics	Mesa Intel® NVIDIA Corporation GM108M [GeForce MX130] / Mesa Intel® UHD Graphics 620 (KBL GT2)
Système d'exploitation	Ubuntu 29.04.1 LTS 64-bit

TABLE IV – *Machine utilisée*

## 2.2.2 Logiciels et librairies

### 2.2.2.a Robot operating system (ROS)

Le Robot Operating System (ROS) est un framework open source qui aide les chercheurs et les développeurs à créer et à réutiliser du code entre les applications robotiques. ROS est également une communauté open source mondiale d'ingénieurs, de développeurs et d'amateurs qui contribuent à rendre les robots meilleurs, plus accessibles et disponibles pour tous.

**Architecture du ROS :** ROS fournit une architecture de communication flexible entre les processus et les machines. Les processus ROS sont appelés nœuds et chaque nœud peut communiquer avec d'autres nœuds via des rubriques. Les connexions entre les nœuds sont gérées par le maître et suivent ce processus :

Le premier nœud informe le maître qu'il a des données à partager. Le deuxième nœud informe le maître qu'il a besoin d'accéder aux données. Une connexion est créée entre deux nœuds. Le premier nœud peut envoyer des données au deuxième nœud. Les nœuds qui publient des données sont appelés Publishers et les nœuds qui s'abonnent aux données sont appelés Subscribers. Les nœuds peuvent être à la fois des éditeurs et des abonnés. Les messages envoyés aux sujets sont pour la plupart standardisés, ce qui rend le système très flexible. ROS permet la communication entre les machines où les nœuds s'exécutent sur des machines distinctes, mais peut communiquer de manière transparente pour l'utilisateur s'il connaît le même maître.

**Version utilisée :** ROS Noetic (la version compatible avec la version du système ubuntu utilisée)

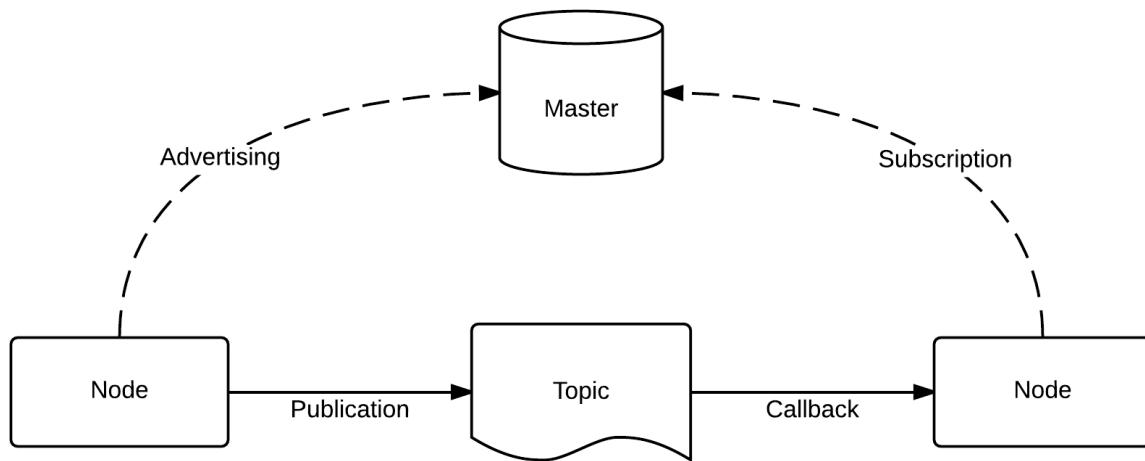


FIGURE 17 – Architecture du Ros

### 2.2.2.b Gazebo

Gazebo est environnement de simulation 3D puissant pour robots autonomes, particulièrement adapté aux tests d'évitement d'objets et de vision par ordinateur. Gazebo est le simulateur par défaut pour Robot Operating System (ROS). Il fournit des simulations dynamiques que vous pouvez choisir parmi différents moteurs physiques, des graphiques 3D avancés, des capteurs et des bruits, des plugins pour robots et des modèles de robots prêts à l'emploi. De plus, il s'exécute sur la couche de transport TCP/IP, ce qui permet une intégration réseau facile, comme le fait ROS.[30]

Gazebo se compose du World qui contient des informations sur tous les modèles, le moteur physique, les plugins et les capteurs. Le modèle contient des liens et des articulations. Les liens sont connectés les uns aux autres à l'aide d'un joint approprié. La première étape de la simulation du système dans consiste à en créer un modèle.

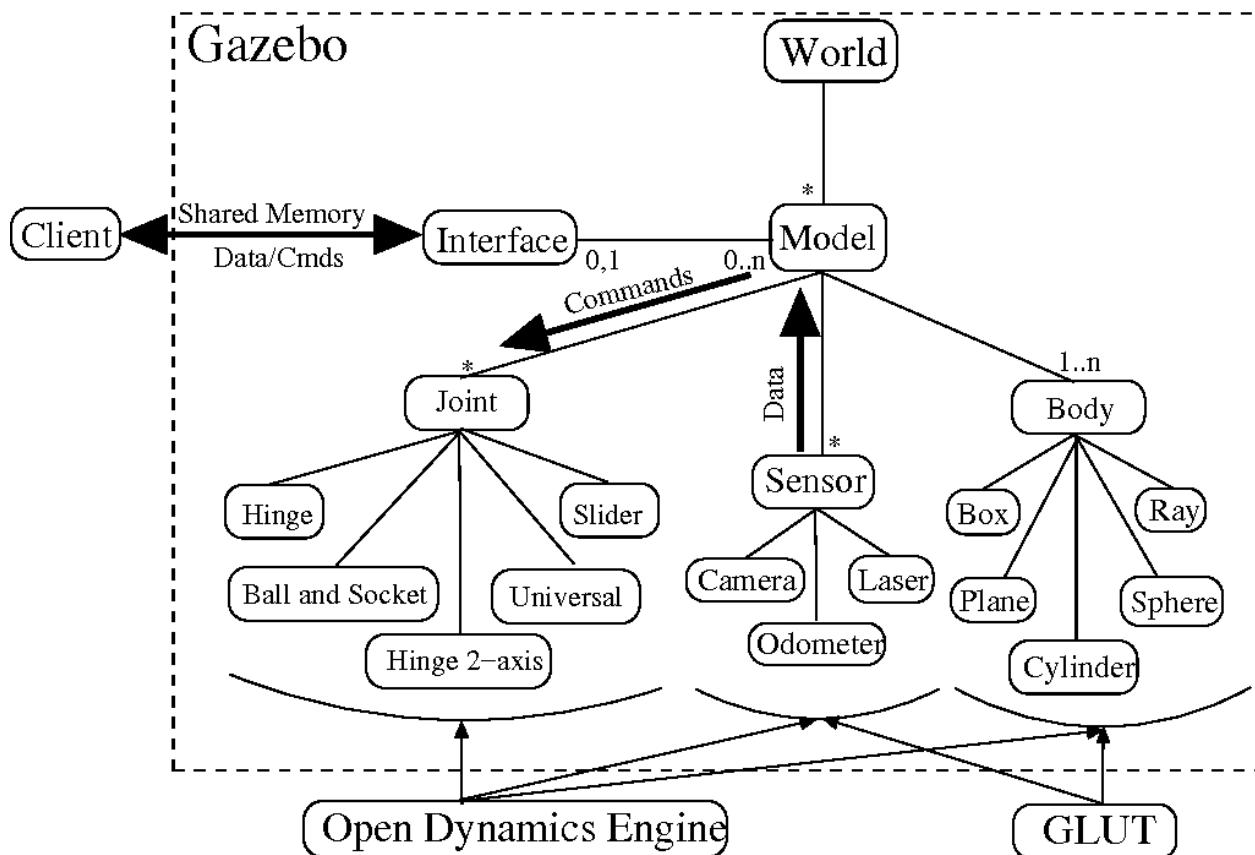


FIGURE 18 – Structure du Gazebo

### 2.2.2.c Vs Code

Visual Studio Code est un éditeur de code redéfini et optimisé pour créer et déboguer des applications Web et cloud modernes.

### 2.2.2.d Langages

Python3 pour la création de la solution, xml pour la création des modèles de simulation.

### 2.2.2.e Librairies

-Rospy : rospy est une bibliothèque client Python pure pour ROS. L'API client rospy permet aux programmeurs Python de s'interfacer rapidement avec les sujets, services et paramètres ROS

-OpenCV : OpenCV est une bibliothèque de fonctions de programmation principalement destinées à la vision par ordinateur en temps réel. Elle est utilisée pour extraire les informations à partir des images, lors de l'utilisation de la méthode Monodepth camera.

-CV Bridge : une librairie qui est utilisée pour convertir entre les messages d'image ROS et les images OpenCV.

-Autre librairies : Numpy, tensorflow, pandas, matplotlib.

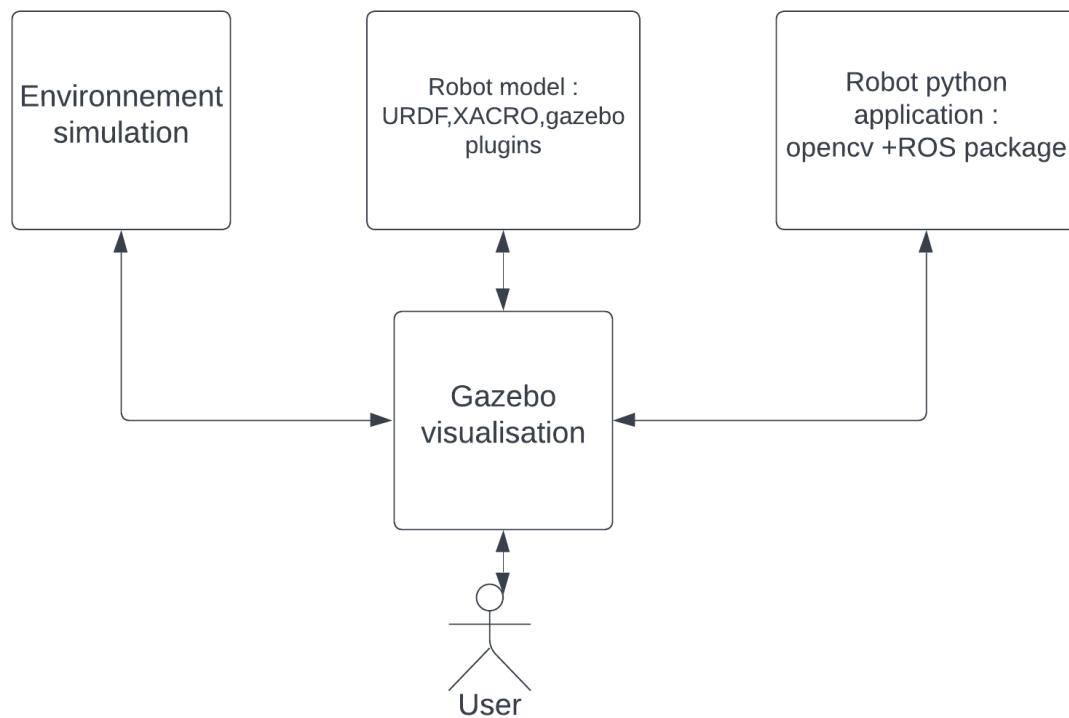


FIGURE 19 – Structure du Gazebo/ ROS packages

## 2.3 Conclusion

Dans ce chapitre, nous avons présenter l'environnement matériel et logiciel du projet, dans le chapitre suivant, nous allons afficher les environnements du test et discuter les résultats trouvés.

## CHAPITRE 3

### EXPÉRIENCES ET ÉVALUATION DE LA SOLUTION

#### 3.1 Introduction

Dans ce chapitre, nous présentons les expériences et les tests que nous avons tentés afin d'évaluer et de valider notre système. Tout d'abord, nous donnons un aperçu de nos environnements de test. Ensuite, nous introduisons la configuration de test où nous décrivons les mesures de performance ainsi que les variables de simulation utilisées. Enfin, nous passons en revue les expériences réalisées à la fois sur les méthodes utilisés et concluons par une discussion sur leurs résultats.

#### 3.2 Environnements du test

Afin de bien structurer le training, nous utilisons la méthode d'apprentissage progressif, l'agent apprend et applique le modèle du DRL dans un environnement simple, visualise les résultats, et enregistre les meilleurs modèles trouvés, ensuite, lancer le training dans un environnement plus compliqué i.e contient plus d'obstacles à éviter, en utilisant l'état des modèles enregistrés précédemment comme état initial, et ceci à fin d'éviter de commencer le training avec des poids aléatoires, et avoir un bon démarrage du training. Le robot est représenté par un simple modèle existant contenant un Laser et une Camera comme composants principaux.

##### 3.2.1 Environnement simple

Comme montre la figure suivante, l'environnement simple est un espace de 64m<sup>2</sup> contenant 4 murs, des positions aléatoires du target représentées par un carré rouge (restants dans l'espace) pour chaque épisode de training.

Le robot dans cette figure utilise un Laser

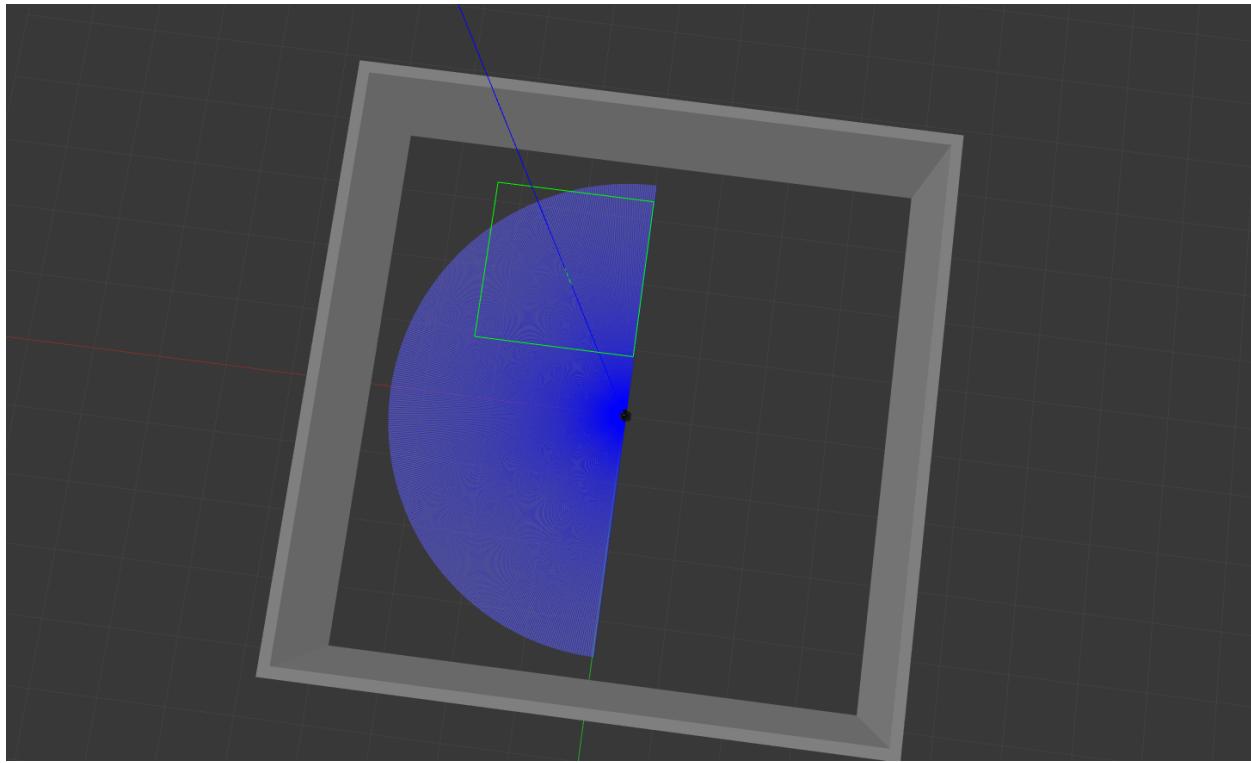


FIGURE 20 – *Environnement simple du training*

### 3.2.2 Environnement complexe

Un environnement simple, avec 5 obstacles diffusés dans de différentes positions dans l'espace entouré par les murs.

Le robot dans cette figure utilise la méthode du monodepth camera sans laser.

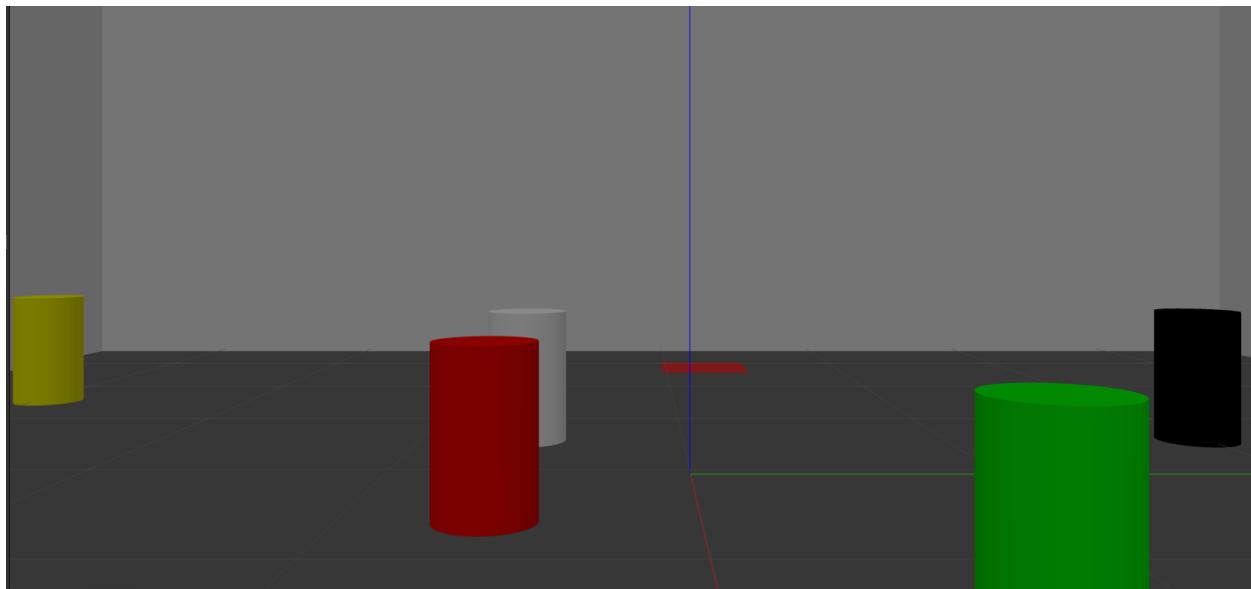


FIGURE 21 – *Environnement complexe du training*

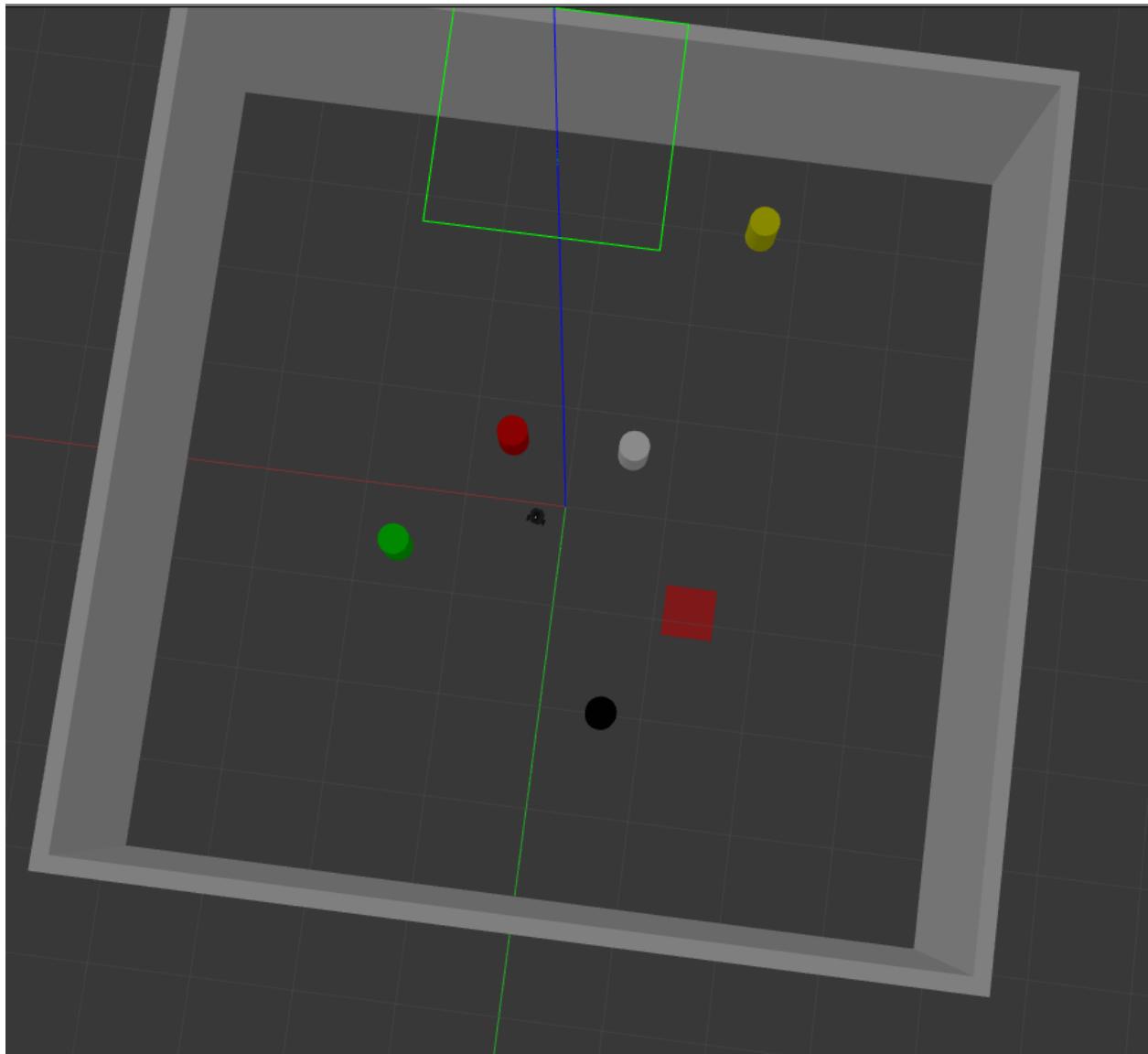


FIGURE 22 – *Environnement complexe du training*

### 3.2.3 Environnement du test final

L'environnement du test est un environnement open source présentant un entrepôt qui contient des modèles de stock personnalisés qui remplacent les différents obstacles rencontrés lors du training

Le robot dans cette figure utilise la méthode du monodepth camera sans laser.

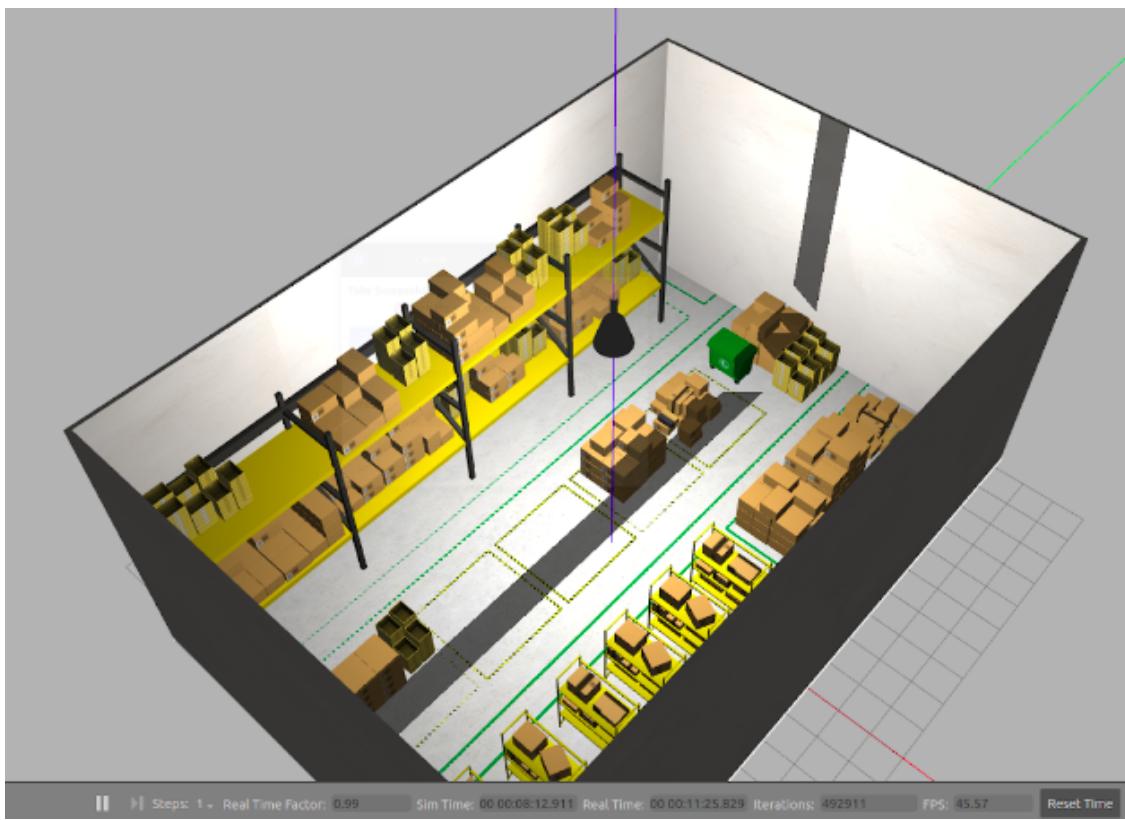


FIGURE 23 – *Environnement du test : entrepôt*

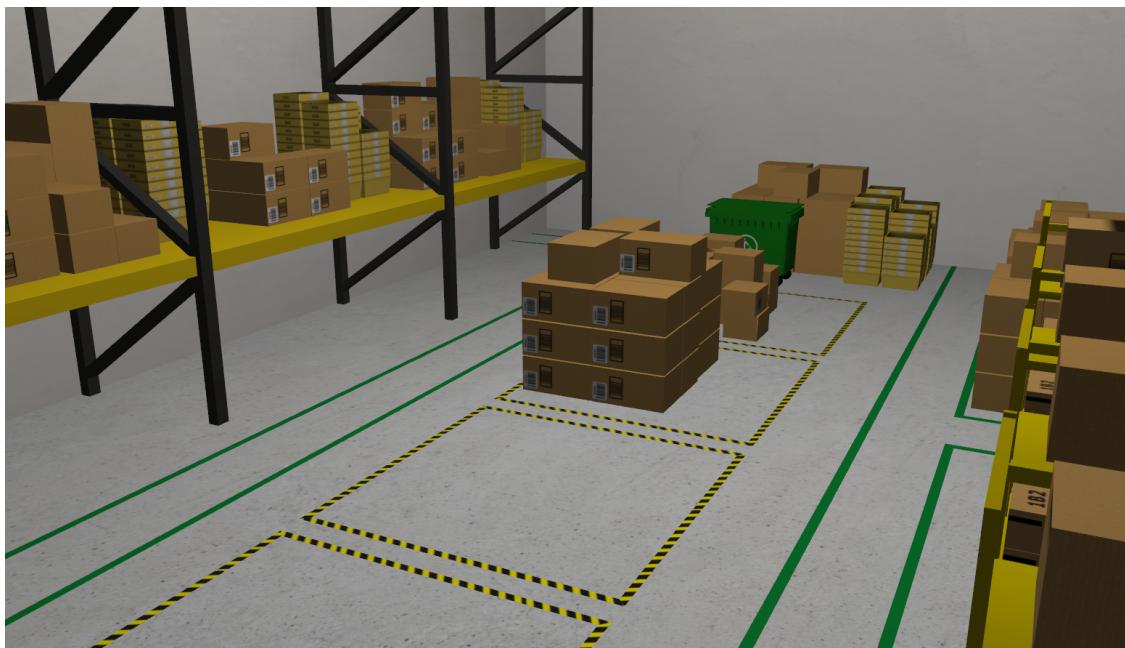


FIGURE 24 – *Environnement du test : entrepôt image 2*

### 3.3 Paramètres utilisés

Ci dessous un tableau présentant les valeurs des paramètres utilisés.

Paramètre	Valeur
Learning Rate	0.0001
Nombre de couches cachées	2
Nombre de neurones : 1ere cc	400
Nombre de neurones : 2ere cc	300
Optimiser	ADAM

TABLE V – *Paramètres utilisés pour le Actor-critic Network*

Ci dessous un tableau présentant les valeurs des paramètres utilisés pour le DRL.

Paramètre	Valeur
Gamma	2
Reward done	-120
Reward arrive	+100
Cr	500
Replay start size	10000
Replay buffer size	100000
Clipping loss ratio	0.1
Entropy loss ratio	0.2
Epsilon greedy variation	0.99

TABLE VI – *Paramètres utilisés pour le DRL*

### 3.4 Résultats des tests

Les résultats sauvegardés représentent la variation de la valeur du récompense pour chaque 5000 time step

### 3.4.1 Environnement Simple

#### 3.4.1.a PPO vs DDPG

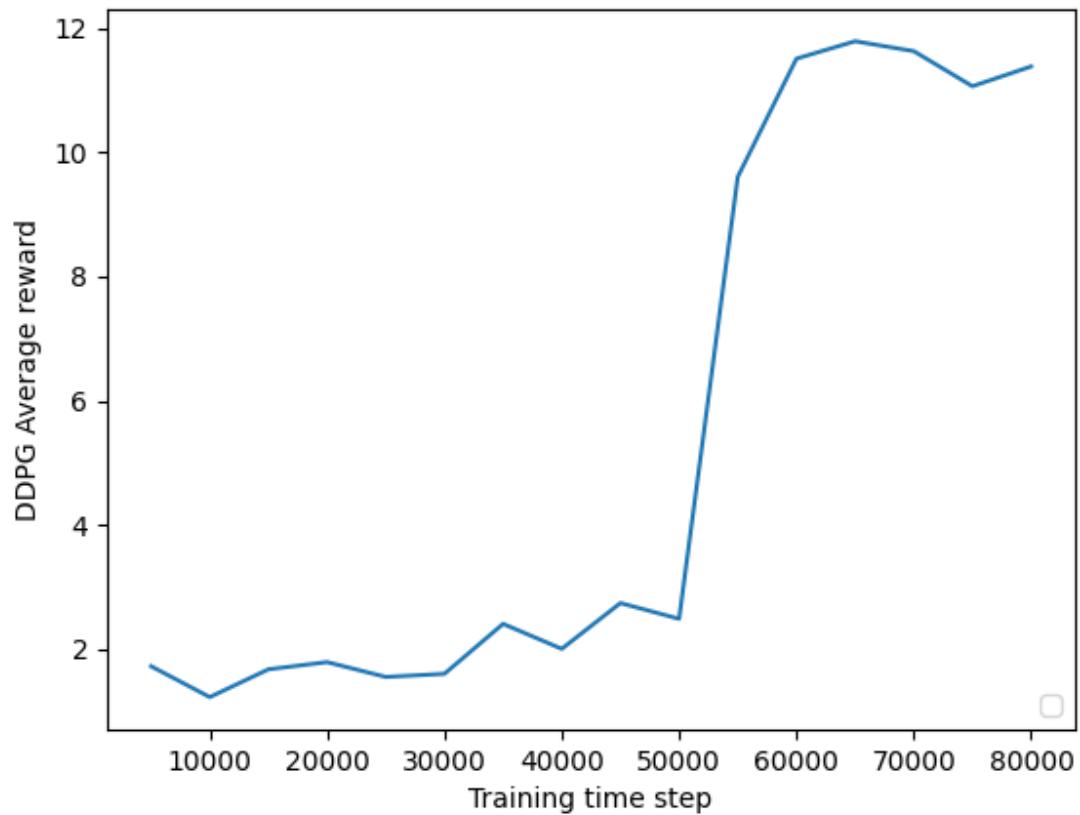


FIGURE 25 – Résultat de l’entraînement du DDPG dans un environnement simple (Moyenne du récompense)

Comme le montre le graphique, le modèle DDPG a convergé à des performances adéquates et stables après les premiers 50000 times steps , i.e le début de l’exploitation et la fin de la phase d’exploration de l’agent.

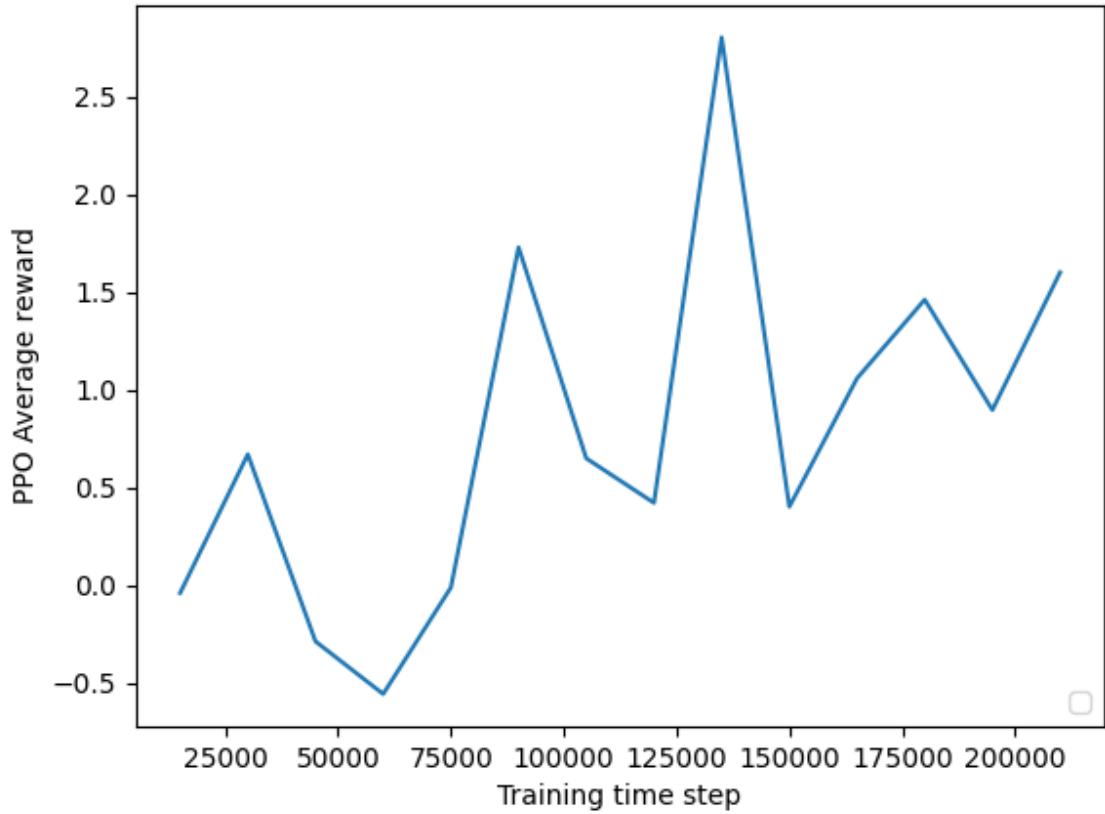


FIGURE 26 – Résultat de l’entraînement du PPO dans un environnement simple (Moyenne du récompense)

En plus de 200000 time step, comme le montre le graphique, le modèle PPO si dessus présente les valeurs augmentant des moyennes des récompenses collectés. contrairement à DDPG, et selon les articles scientifiques, l’algorithme PPO a besoin des millions de time step pour pouvoir converger a des performances stable, car il n’enregistre pas les données déjà collectées, ce qui nécessite un environnement matériel très puissant.

	PPO (en 200000 ts)	DDPG en 80000ts
<b>Min avg reward</b>	-0.65	1.71
<b>Max avg reward</b>	3.25	11.89
<b>Temps d’exécution de 5000 ts</b>	20 min.	20 min

TABLE VII – PPO vs DDPG - environnement simple

### 3.4.1.b Monodepth camera vs Laser

Après avoir effectuer le test sur un environnement simple avec PPO et DDPG, nous avons pris l'algorithme qui a produit des résultats meilleurs ( DDPG ) pour faire le test entre l'utilisation du Laser vs Monodepth camera dans le même environnement simple. La figure suivante montre les moyenne des récompense gagnés pour chaque 5000 ts pour les deux méthodes.

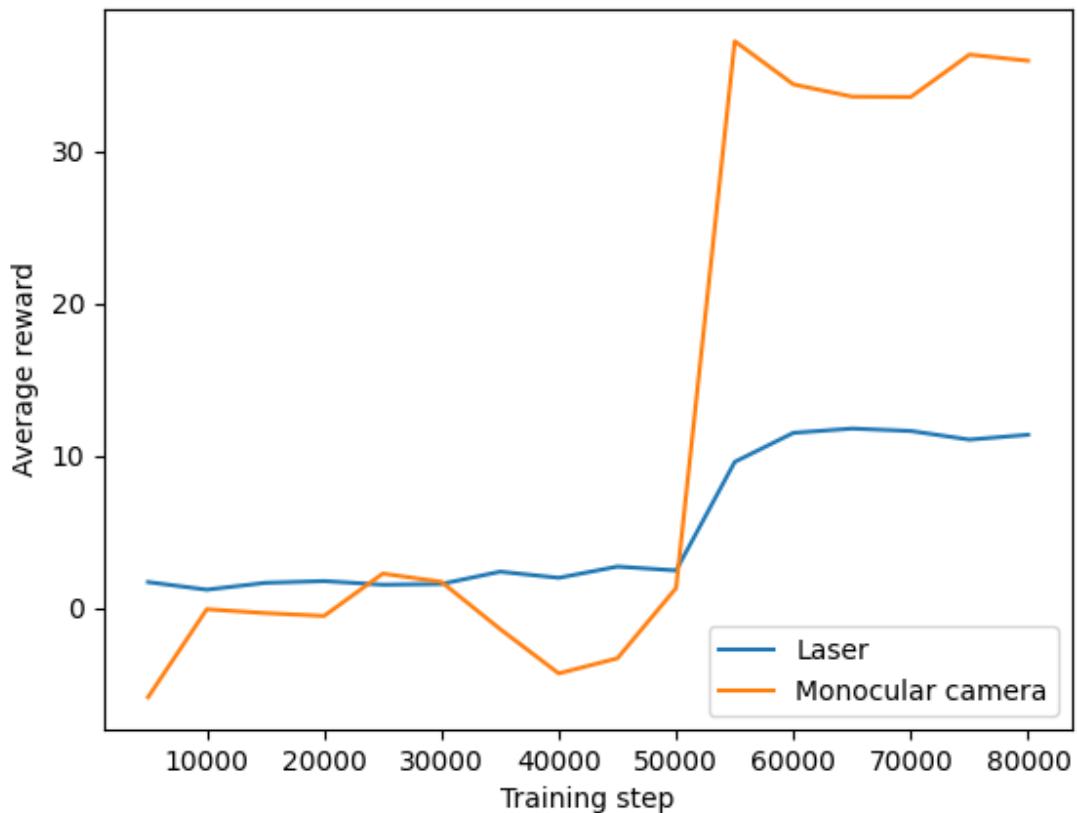


FIGURE 27 – Comparaison entre les résultats du Laser vs Monocular Caméra dans un environnement simple

	Monocular Camera	Laser
Min avg reward	-5.83	1.71
Max avg reward	36.35	11.89
Temps d'exécution de 5000 ts	90 min.	20 min

TABLE VIII – Monocular Camera vs Laser - environnement simple

### 3.4.2 Environnement Complexé

#### 3.4.3 laser vs monodepth

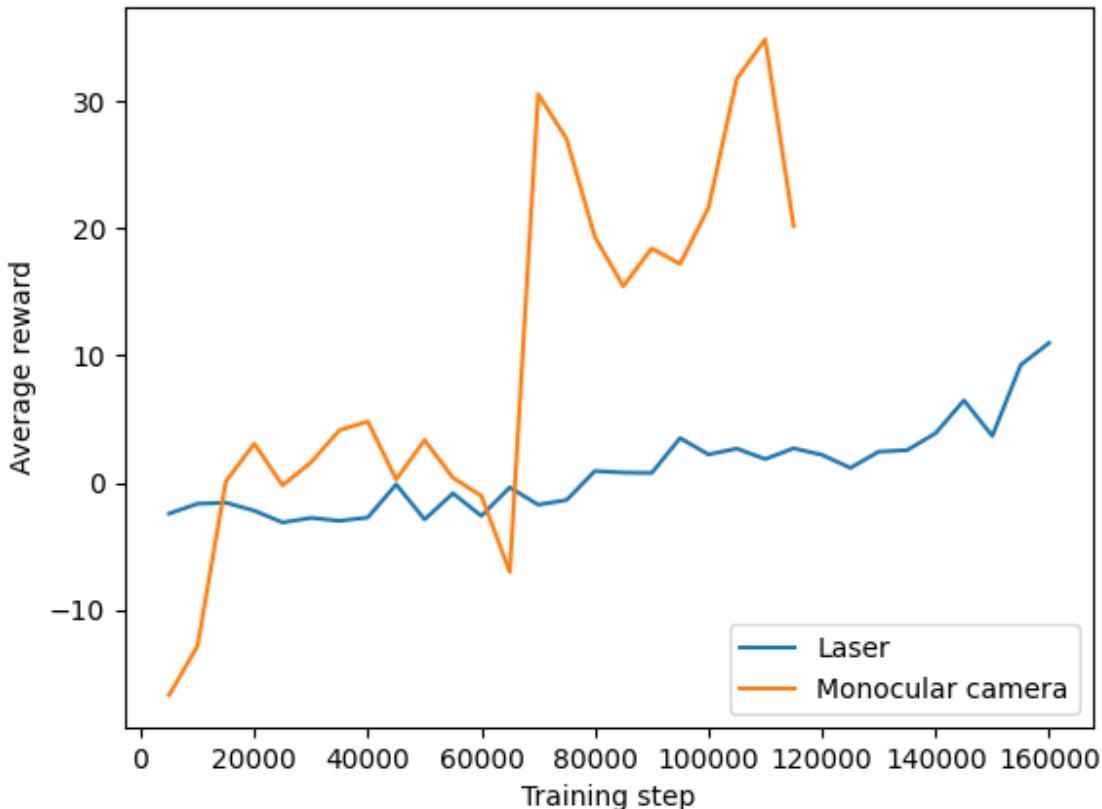


FIGURE 28 – Comparaison entre les résultats du Laser vs Monocular Caméra dans un environnement simple

	Monocular Camera	Laser
Min avg reward	-16.67	-3.11
Max avg reward	34.85	10.99
Temps d'exécution de 5000 ts	90 min.	20 min

TABLE IX – Monocular Camera vs Laser - environnement complexe

### 3.5 Discussion

On peut voir que les rendements finaux de la méthode utilisant l'algorithme DDPG est meilleur que l'algorithme PPO en terme de temps et de récompense. En effet, le maximum de récompense collecté par PPO se situe autour de 3, tandis que celui du DDPG arrive jusqu'à 11.89 pour un environnement simple, en 80000 ts et en utilisant un laser. Pour un environnement complexe, les récompenses trouvées par DDPG se situent entre -3.11 et 10.99 en utilisant un laser. Cependant, lors du remplacement du laser par la monocular camera, nous remarquons que la meilleure récompense moyenne trouvée pour un environnement simple en 80000 ts est de 36, ce qui est beaucoup meilleur que celle trouvée lors de l'utilisation d'un laser, mais avec un minimum de -6, qui est un peu loin du minimum avec laser. De même pour un environnement complexe, avec une valeur maximale autour de 34 au lieu de 11, et valeur minimale de -16 au lieu de -3.

L'agent avec laser arrive à apprendre un modèle stable et plus efficace dans le temps que l'agent avec caméra monocular. L'entraînement montre que monocular diverge et ce que pour 120000 ts, par contre l'agent avec laser est plus stable et s'améliore dans le temps. Faute de temps et de ressources, nous avons effectuer les tests avec les modèles issues du 120000 ts pour le monocular et 160000 ts pour le laser dans un environnement complexe, et les résultats du test s'accordent avec les observations de l'entraînement, concernant monocular, en effet, monocular est moins performante, le robot se heurte plus d'obstacles que lors de l'utilisation d'un laser, le taux de réussite se situe entre 4/10 et 7/10, pendant que le taux de réussite lors de l'utilisation d'un laser est compris entre 7/10 et 10/10 pour la plupart des temps. (taux de réussite = nombre d'obstacles évités/nombre totale d'obstacles) Nos observations nous mènent à déduire que la méthode proposée pour le monocular caméra est moins meilleure, et le DDPG est meilleur par rapport à PPO pour un tel environnement, et de tels paramètres d'environnement. La notion que nous voulons souligner ici est que l'approche du monocular peut bien remplacer le laser avec du matériel à faible coût, cependant, elle nécessite plus d'entraînement et du temps pour pouvoir y arriver au même niveau du laser.

## CONCLUSION GÉNÉRALE

L'entreposage a évolué au cours des dernières années en devenant plus stratégique et complexe, accessible et en poussant à l'automatisation. Il continuera d'évoluer au cours de la prochaine décennie, car il dépend de variables qui peuvent perturber la majorité des lieux de travail dans de nombreuses industries. L'entreposage continuera d'être poussé à s'adapter au monde en constante évolution.

Dans ce travail, nous avons étudié les progrès prometteurs vers l'industrie 4.0 et l'entreposage en intégrant les UAV en tant que technologie habilitante et l'apprentissage par renforcement en tant que paradigme mathématique robuste pour un meilleur contrôle et pour des performances supérieures.

La première partie de ce rapport est un rappel théorique des principaux éléments du sujet et des travaux de recherche menés dans chacun d'eux. Ainsi, nous avons commencé par introduire le domaine de l'industrie 4.0, et le domaine de l'intelligence artificielle en général. Nous éclairons ensuite spécifiquement l'automatisation des entrepôts dans l'industrie 4.0 en utilisant les drones. Dans le chapitre 2, nous présentons la problématique du path planning et les travaux connexes, les méthodes qu'ils utilisent et leurs résultats atteints. La deuxième partie commence par présenter une modélisation de la problématique sous forme d'un MDP, une vue globale sur la solution et sur les méthodes utilisés, notamment l'algorithme PPO, l'algorithme DDPG, et la monocular depth camera. Le 2ème chapitre de cette partie présente les outils utilisés pour réaliser le projet et simuler le travail dans un environnement 3d en citant les logiciels et bibliothèques utilisées. Le dernier chapitre présente les résultats obtenus dans des environnements de test, des expériences et des solutions en comparant les différents concepts poursuivis.

Pour Conclure, nous concluant d'un coté, que le Laser est plus stable par rapport à la Caméra monoculaire qui a besoin de beaucoup plus d'entraînement afin d'atteindre le niveau du laser. D'un autre coté, l'application de l'algorithme DDPG sur notre problématique a donné des résultats meilleurs que les résultats obtenus par l'utilisation de l'algorithme PPO sur la même problématique.

### Perspectives

## PARTIE II

---

L'apprentissage par renforcement profond est en hausse, des études récentes ont proposé de nouvelles idées et stratégies pour résoudre les problèmes clés de l'apprentissage par renforcement tels que l'efficacité de l'échantillonnage ou le compromis exploration-exploitation. Ces algorithmes ont été largement appliqués dans les environnements de jeux vidéo et autres émulateurs physiques, mais moins pour la navigation robotique et l'évitement d'obstacles. Nous espérons les utiliser pour résoudre des problèmes existants et améliorer le comportement du robot, ainsi que pour obtenir des performances plus impressionnantes en réduisant l'espace de recherche. Nous voulons aussi étudier plusieurs algorithmes de DRL, appliqués sur des environnement plus compliqués, en ajoutant un système multi-agent et des concepts de IOT et de la 5G afin de travailler sur un simulateur contenant plusieurs drones qui traillent au même temps. De plus, nous espérons travailler sur un espace d'action 3d, et ceci en considérant les trois axes x, y et z, et enfin, faire plus de tests afin de prouver l'efficacité de la monocular camera.

# Bibliographie

## BIBLIOGRAPHIE

## BIBLIOGRAPHIE

- [1] Yuxi LI. "Introducing Deep Reinforcement Learning". In : *Conférence CoRR* (2018).
- [2] George D. GREENWADE. "The Comprehensive Tex Archive Network (CTAN)". In : *TUGBoat* 14.3 (1993), p. 342-351.
- [3] Reliable plant NORIA. *17 Practical Tips to Optimize Your Warehouse Space*. 2022. URL : <https://www.reliableplant.com/Read/31449/optimize-warehouse-space>.
- [4] Advanced Mobile GROUP. *Overcoming Common Warehouse Automation Challenges*. URL : <https://www.advancedmobilegroup.com>. (2022).
- [5] Donald KNUTH. *Guide de l'automatisation des entrepôts*. URL : <https://www.mecalux.fr/blog/automatisation-entrepot>. (2021).
- [6] Drone (UAV). *Guide de l'automatisation des entrepôts*. URL : <https://www.techtarget.com/iotagenda/definition/drone>. (2021).
- [7] WILL KENTON. *What Is Inventory ?* URL : <https://www.investopedia.com/terms/i/inventory.asp>. (2022).
- [8] *intelligence artificielle - Larousse*. URL : <https://www.lemagit.fr/conseil/IA-machine-learning-deep-learning-quelles-differences>. (2022).
- [9] Yuxi LI. "Introducing Deep Reinforcement Learning". In : *Conférence CoRR* (2018).
- [10] David PETERSSON. *IA, machine learning, deep learning : quelles différences ?* URL : [https://www.larousse.fr/encyclopedie/divers/intelligence\\_artificielle/187257](https://www.larousse.fr/encyclopedie/divers/intelligence_artificielle/187257). (2020).
- [11] MINNALEARN. *Les types d'apprentissage automatique*. URL : <https://course.elementsofai.com/fr/4/1>. (2022).
- [12] MOBISKILL. *Quels sont les algorithmes de deep learning ?* URL : <https://mobiskill.fr/blog/conseils-emploi-tech/quels-sont-les-algorithmes-de-deep-learning/>. (2021).
- [13] WieringMarco A. Wiering MARTIJN VAN OTTERLOMARCO. "Reinforcement Learning and Markov Decision Processes". In : *Reinforcement learning* (2012).

- [14] R. S. SUTTON et A. G. BARTO. “Reinforcement learning : An introduction”. In : *MIT Press* (2018).
- [15] Abhishek SURAN. *On-Policy v/s Off-Policy Learning*. URL : <https://towardsdatascience.com/on-policy-v-s-off-policy-learning-75089916bc2f>. (2020).
- [16] by HRISTO HRISTOV. *Off-policy vs. On-policy Reinforcement Learning*. URL : <https://www.baeldung.com/cs/off-policy-vs-on-policy>. (2022).
- [17] Avinash POTDAR. *Drones for warehouse inventory management*. URL : <https://www.argonandco.com/en/news-insights/articles/drones-for-warehouse-inventory-management/>. (2021).
- [18] Reza Kazemi SHAHRAM AZADI et Hamidreza Rezaei NEDAMANI. *Vehicle Dynamics and Control*. 2021.
- [19] Jing Guo JUNLI GAO Weijie Ye et Zhongjuan LI. “Deep Reinforcement Learning for Indoor Mobile Robot Path Planning”. In : *sensors* (2020).
- [20] Giuseppe Paolo LEI TAI et Ming LIU. “Virtual-to-real Deep Reinforcement Learning : Continuous Control of Mobile Robots for Mapless Navigation”. In : *Latin American Robotic Symposium, Brazilian Symposium on Robotics (SBR) and Workshop on Robotics in Education (WRE)* (2018).
- [21] Alexandre da Silva Simoes GUILHERME CANO LOPES Murillo Ferreira et Esther Luna COLOMBINI. “Intelligent Control of a Quadrotor with Proximal Policy Optimization Reinforcement Learning”. In : *sensors* (2017).
- [22] FJIANG. *Knuth : Computers and Typesetting*. URL : <https://wiki. centrale-marseille.fr/egab/drone:quadrirotor>. (2020).
- [23] “A pretrained proximal policy optimization algorithm with reward shaping for aircraft guidance to a moving destination in three-dimensional continuous space”. In : *International Journal of Advanced Robotic Systems* (2021).
- [24] “Deep reinforcement learning for drone navigation using sensor data”. In : *Neural Computing and Applications* (2021).
- [25] “Mapless Collaborative Navigation for a Multi-Robot System Based on the Deep Reinforcement Learning”. In : *Applied Sciences* (2019).
- [26] FJIANG. *Génération robots*. URL : <https://www.generationrobots.com/blog/fr/qu-est-ce-que-la-technologie-lidar/>. (2020).
- [27] “A pretrained proximal policy optimization algorithm with reward shaping for aircraft guidance to a moving destination in three-dimensional continuous space”. In : *VNUHCM-University of Science* (2020).
- [28] “Mapless Navigation : A reinforcement learning approach”. In : *Eastern Switzerland University of Applied Sciences* (2021).
- [29] BYJUS. *Angular Velocity And Its Relation With Linear Velocity*. URL : <https://byjus.com/physics/angular-velocity-linear-velocity/>.
- [30] *Gazebo Simulation*. URL : <https://docs.px4.io/main/en/simulation/gazebo.html>. (2022).

- [31] Sergios KARAGIANNAKOS. *The idea behind Actor-Critics*. URL : [https://theaisummer.com/Actor\\_critics/](https://theaisummer.com/Actor_critics/). (2018).
- [32] Ilias CHRYSOVERGIS. *Proximal Policy Optimization*. URL : [https://keras.io/examples/rl/ppo\\_cartpole/](https://keras.io/examples/rl/ppo_cartpole/). (2021).
- [33] “Federated Reinforcement Learning for Training Control Policies on Multiple IoT Devices”. In : *sensors* (2020).
- [34] Data science TEAM. *Q Learning*. URL : <https://datascience.eu/fr/apprentissage-automatique/q-learning/>. (2020).
- [35] “Q-Learning”. In : *Carnegie Mellon University* (2020).
- [36] “Deep Q-Network, with PyTorch”. In : *Towards Data Science* (2021).
- [37] “An Autonomous Path Planning Model for Unmanned Ships Based on Deep Reinforcement Learning”. In : *sensors* (2020).
- [38] “Deep Deterministic Policy Gradients Explained”. In : *Towards Data Science* (2019).
- [39] Martial Hebert JOHN HANCOCK et Chuck THORPE. “Laser Intensity-Based Obstacle Detection”. In : *IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications* (1998).
- [40] “Defocus Deblurring Using Dual-Pixel Data”. In : *York University, Samsung AI Center, Toronto, Canada* (2020).
- [41] “Digging Into Self-Supervised Monocular Depth Estimation”. In : *CoRR* (2019).