Loi de commande dynamique d'un robot parallèle par apprentissage par renforcement

Quentin Dechambre, Vincent Gagnol, Chedli Bouzgarrou, Thierry Chateau, Jochen Triesch

Loi de commande dynamique d'un robot parallèle par apprentissage par renforcement

Q. DECHAMBRE[a], V. GAGNOL[b], C. BOUZGARROU[c], , T. CHATEAU[d], , J. TRIESCH[e]

a. Université Clermont Auvergne, Clermont Auvergne INP, CNRS, Institut Pascal, F-63000 Clermont-Ferrand, France, quentin.dechambre@sigma-clermont.fr
b. Université Clermont Auvergne, Clermont Auvergne INP, CNRS, Institut Pascal, F-63000 Clermont-Ferrand, France, vincent.gagnol@sigma-clermont.fr
c. Université Clermont Auvergne, Clermont Auvergne INP, CNRS, Institut Pascal, F-63000 Clermont-Ferrand, France, belhassen-chedli.bouzgarrou@sigma-clermont.fr
d. Université Clermont Auvergne, Clermont Auvergne INP, CNRS, Institut Pascal, F-63000 Clermont-Ferrand, France, thierry.chateau@uca.fr
e. Frankfurt Institute for Advanced Studies, 60438 Frankfurt am Main, Germany, triesch@fias.uni-frankfurt.de

Résumé :

La modélisation dynamique des robots industriels est depuis plusieurs décennies soutenue par des modèles théoriques. Malgré les efforts déployés, ces modèles sont toujours limités quant à leur représentation de certains comportements physiques tels que le frottement des liaisons, l'amortissement du système ou ses interactions avec l'environnement. Néanmoins, ces paramètres doivent tout de même être identifiés, et ce, en utilisant des méthodes complexes et des capteurs coûteux intégrés aux le systèmes réels. Cet article développe une méthode basée sur l'Intelligence Artificielle (IA) permettant le contrôle dynamique d'un robot lors d'une tâche dédiée, dans un environnement restreint. L'intérêt de l'utilisation de l'IA est d'éviter par exemple l'identification complexe des paramètres dynamiques du robot et de générer directement une loi de commande en couple pour satisfaire la tâche demandée. Dans le cas présenté ici, le système utilisé pour développer et valider la méthodologie est un jeu de table de Air Hockey. La technologie utilisée ici est basée sur l'apprentissage par renforcement sur des modèles numériques simulés par ordinateur. La loi de contrôle déduite de cette approche de l'IA sera ensuite transférée au système réel composé d'une table d'Air Hockey équipée d'un robot parallèle à 5 barres. Des recherches sur ce jeu ont déjà été menées et proposent une commande d'un robot à l'aide d'algorithmes de commande traditionnels permettant ainsi de comparer les méthodes d'IA et les approches classiques.

Abstract :

The dynamic modelling of industrial robots has been supported by supported by theoretical models for decades. However, these models do not allow a precise representation of reality in the link friction, the damping of a system or its interactions with the environment. Nevertheless, these parameters have to be identified by using complex methods and expensive sensors integrated into the real system. This paper

develops a method based on Artificial Intelligence (AI) allowing the dynamic control of a robot during a dedicated task within a restricted environment. The interest of using AI method is to avoid the complex robot dynamic parameters identification. In this case, the test bench used to develop and validate our methodology is an Air Hockey table game. The artificial intelligence technology used here is based on reinforcement learning of computer- simulated numerical models. The control law deduced from this AI approach will then be transferred to the real system composed by an Air Hockey Table equipped with a 5-bar parallel robot. Research on this game was already conducted and propose a classical vision of robot control using traditional command algorithms allowing comparing AI methods and classical ones.

Mots clefs : Robotique, Intelligence Artificelle, Commande, Dynamique, Apprentissage par Renforcement, Unity, Machine Learning

1 Introduction

Les robots ont été conçus pour être des machines capables d'accomplir un ensemble de tâches dans un environnement avec une certaine autonomie et une robustesse face aux fluctuations des tâches et de l'environnement. On accorde aux robots, entre autres, les capacités suivantes : déplacements, préhension, perception, communication et prise de décision. Une difficulté majeure en robotique est la variabilité des environnements et des tâches auxquelles un robot doit être confronté. Et c'est face à ces problématiques que des recherches ont déjà été menées pour permettre aux robots d'intégrer des algorithmes basés sur l'intelligence artificielle et ses différentes technologies. Ces dernières années, la robotique basée sur les réseaux de neurones a attiré beaucoup d'attention en raison de leur forte capacité de généralisation, de la disponibilité d'une grande quantité de données d'entraînement et de la puissance de calcul des ordinateurs. C'est ainsi que des problèmes de suivi de trajectoires peuvent être comblés par l'utilisation de réseaux de neurones denses entraînés par méthode itérative [1]. Il est à noter aussi que la résolution de problèmes d'identification de pièces, de leur préhension et de leur assemblage par un robot manipulateur peut être réglée par l'utilisation de l'apprentissage par renforcement [2]. La robotique mobile est aussi un médium de l'utilisation de l'apprentissage par renforcement avec par exemple le traitement du *pathfinding* [3] ou la navigation de robot mobile à roues [4]. L'utilisation de modèle numérique pour un apprentissage par renforcement permet également le contrôle de robot quadrupède [5]. Il s'agit là d'une liste non exhaustive des différentes applications de l'intelligence artificielle et de l'apprentissage par renforcement dans le domaine de la robotique.

Ainsi au travers de ce document, une méthodologie est développée, permettant l'intégration d'une loi de commande dynamique basée sur des modèles numériques au sein d'un système réel, devant effectuer une tâche précise dans un environnement particulier. L'intérêt principal de notre approche est de générer une loi de commande sans se soucier des problèmes habituels de la commande de robot, comme le passage de singularités, exposée par Koessler dans [6], ou l'identification des paramètres du modèle dynamique [7][8].L'application choisie pour illustrer cette approche est le jeu de l'Air Hockey, offrant un environnement alliant mouvement rapide, commande dynamique, vision et stratégie. Nous aborderons les différentes spécificités de ce système afin de construire le cahier des charges à remplir par notre loi de commande. C'est ce cahier des charges et son environnement qui ont influencé le dimensionnement et la conception du robot utilisé. Ce robot a été construit pour répondre aux exigences du jeu Air Hockey.

Des recherches sur ce jeu ont déjà été menées et proposent une vision classique du contrôle de robot à l'aide d'algorithmes de commande traditionnels tels que ceux proposés dans [9] ou [10].
Afin de générer une loi de commande pour le robot, le système a été simulé numériquement à l'aide d'un moteur physique intégrant une compatibilité avec l'apprentissage par renforcement. La simulation est basée uniquement sur la modélisation numérique du système, sans prendre en compte de modélisations analytiques habituelles telles qu'on peut en trouver dans des modèles classiques comme celui présenté par Briot dans [11]. L'objectif principal de notre approche est d'éviter l'identification complexe des paramètres dynamiques, pour générer directement les couples de commandes à partir de prise de vue. Elle permet également d'éviter la prise en compte de beaucoup de variables d'environnement, telles que la luminosité, les couleurs et les contrastes.
Cet article présente tout d'abord le système choisi ainsi que le cahier des charges de l'application au jeu de Air Hockey. Dans une deuxième partie, les spécifications qui ont orienté le choix et le dimensionnement du robot seront présentées. Dans un troisième temps, la méthode complète proposée pour la simulation et l'intégration du comportement dynamique du robot par des techniques d'apprentissage est détaillée, ainsi que le processus d'entraînement. Enfin les premiers résultats d'apprentissage par renforcement seront présentés. La méthodologie complète de simulation est ainsi représentée dans la figure 1 et se verra dans de futurs travaux, transférée sur le système réel.

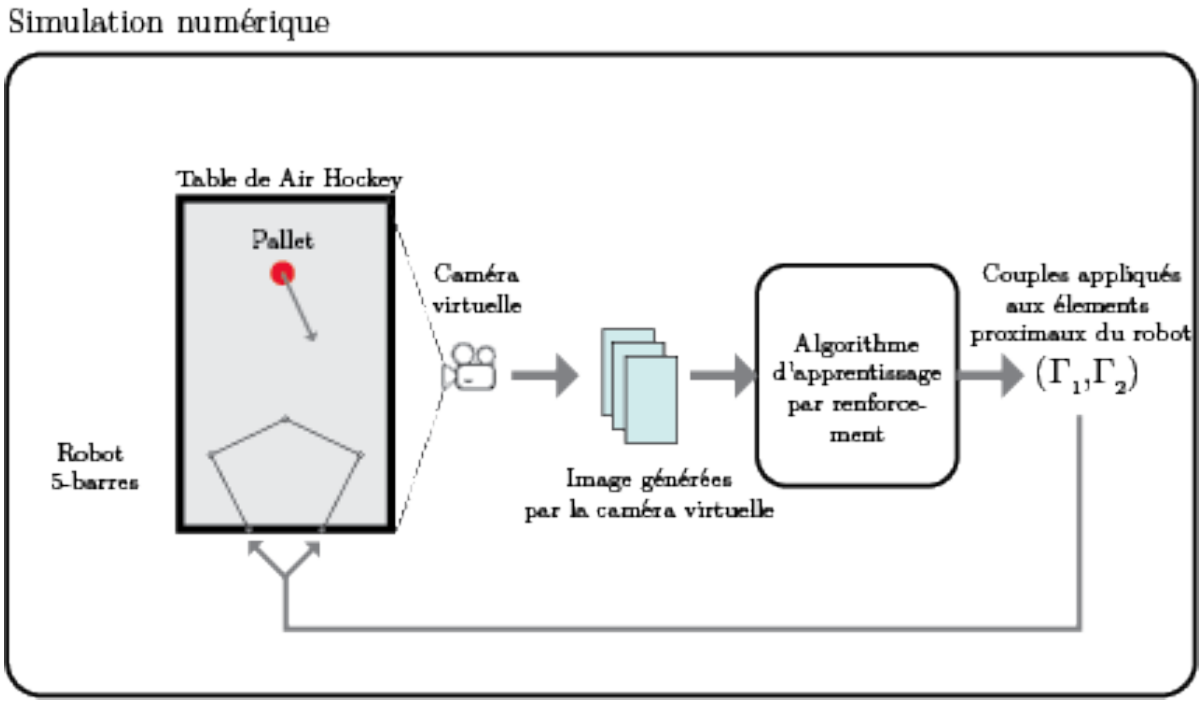

FIGURE 1 – Représentation schématique du système simulé complet

2 Présentation du système et approche proposée

2.1 Système considéré

Le système mis en place est constitué d'une table de Air Hockey, d'un robot équipé d'un butoir en guise d'effecteur. Ce robot est placé à l'extrémité de la table comme pourrait l'être un joueur, le robot ainsi que son architecture sont décrits dans 2.1.1.

Le jeu Air Hockey oppose deux adversaires de part et d'autre d'une table. Un palet se déplace sur la surface de la table et chaque joueur doit le pousser à l'aide d'un butoir et marquer un point en l'envoyant dans le but de l'adversaire. La table est équipée d'un système de soufflerie qui crée un coussin d'air sur la surface de la table afin de limiter la friction entre la table et le palet.
La table utilisée mesure 1,2m sur 2m offrant ainsi à chaque joueur une demi-zone dans laquelle il peut frapper le palet. L'application choisie réduit le degré de mobilité de notre système à 2 translations sur la surface de la table. De plus les règles du jeu indiquent que le palet doit toujours demeurer sur la surface de la table. Une dernière règle indique que le joueur a 7 secondes pour jouer son tir une fois que le palet est entré dans sa demi-zone.
Un modèle numérique du système est présenté en figure 2

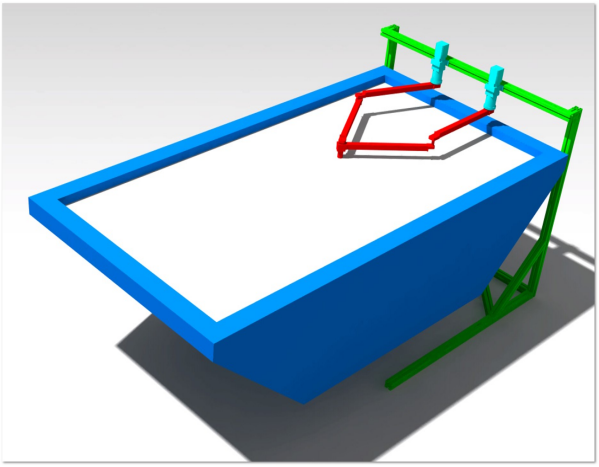

FIGURE 2 – Modèle numérique du système

2.1.1 Choix de l'architecture du robot et dimensionnement

Le choix du robot s'est porté sur un robot plan à structure parallèle. Cette architecture permet notamment d'obtenir de meilleures accélérations de l'effecteur par rapport à une architecture sérielle et une structure plus robuste face aux chocs avec le palet, comme montré dans [12]. Ce robot à 2 degrés de liberté possède 5 liaisons pivot dont deux seront pilotés : A_1 et A_2. La modélisation géométrique de ce robot est proposée en figure 3 et permet de définir les dimensions du robot vis-à-vis de l'environnement de jeu et de la zone de travail utile.

Afin de dimensionner le robot, le modèle vectoriel géométrique de Joubair [13] a été choisi :

$$\mathbf{r_{OE}} = \mathbf{r_{OB_1}} + \frac{b_1}{d_{B_1B_2}} + \gamma \frac{h}{d_{B_1B_2}} \mathbf{R}\left(\frac{\pi}{2}\right) \mathbf{r_{B_1B_2}} \tag{1}$$

Où $R(\theta)$ est la matrice de rotation d'ordre 2 caractérisant la rotation d'angle θ ainsi que :

$$d_{B_1B_2} = \sqrt{\mathbf{r_{B_1B_2}}^T \mathbf{r_{B_1B_2}}} \text{ , la distance entre } B_1 \text{ et } B_2 \tag{2}$$

$$b_1 = \frac{l_{12}^2 - l_{22}^2 + d_{B_1B_2}^2}{2d_{B_1B_2}} \text{ , la demi-distance entre } B_1 \text{ et } B_2 \tag{3}$$

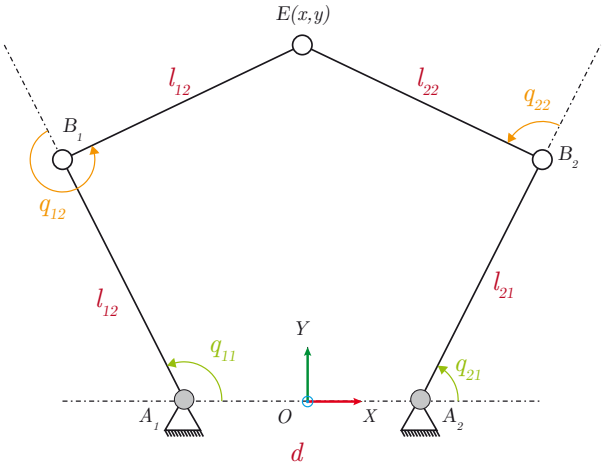

FIGURE 3 – Modélisation géométrique du robot

$$h = \sqrt{l_{12}^2 - b_1^2}\text{ , la hauteur du triangle } EB_1B_2 \tag{4}$$

La modélisation du robot plan à 5 barres utilise les notations suivantes :
$\mathbf{r_{ij}}$ est le vecteur $\vec{ij}$, l_{11}, l_{21} les longueurs des éléments proximaux, l_{12}, l_{22} les longueurs des éléments distaux, d la longueur entre A_1 et A_2, q_1, q_2 les angles formés par les éléments proximaux et le repère du robot, et γ le mode d'assemblage. L'effecteur du robot, situé en E, doit tout comme un joueur humain pouvoir accéder à la plus grande surface possible sur sa demi-zone de jeu, cette consigne a permis de déterminer les dimensions des éléments proximaux et distaux du robot, ainsi que la distance entre les moteurs. Les dimensions des éléments proximaux et distaux sont ainsi : $l_{11} = l_{21} = 569mm$ et $l_{21} = l_{22} = 563mm$.

2.2 Approche proposée

2.2.1 Environnement de simulation

L'ensemble de la simulation est réalisé à l'aide de la suite logicielle Unity, intégrant un environnement de simulation 3D et un moteur physique (Physics) pour gérer l'ensemble du comportement physique et dynamique de notre système. Ce moteur physique permet de gérer les mouvements du robot ainsi que les interactions entre les différents objets de la simulation. Ainsi, l'intégralité des composants physiques de notre système est simulée par des objets (appelés `GameObject`) proposé par Unity. Ils sont construits avec trois composants principaux : un `Rigidbody`, un `Collider`, un `Physic Material`.
L'ajout d'un composant `Rigidbody` à un objet place son mouvement sous le contrôle du moteur physique d'Unity. Même sans ajouter de spécification, un objet `Rigidbody` sera tiré vers le bas par la gravité et mais ne réagira pas aux collisions avec les objets entrants en contact avec lui. L'ajout d'un composant `Collider`, applique une masse à l'objet et définie la forme de l'objet pour les collisions physiques entre les `GameObject`. Un `Collider`, qui est invisible, n'a pas besoin d'avoir exactement la même forme que le modèle de l'objet ; une approximation est souvent plus efficace et indiscernable en simulation. Ces principales propriétés sont les composants `Drag` et `Angulardrag`, associés aux frottements avec le milieu, ici l'air.
Dans la simulation Air Hockey, par exemple, le palet et le maillet sont équipés de `Colliders` en forme de cylindre, tandis que les bords de la table et du plateau sont équipés de `Colliders` en forme de pa-

rallélépipède rectangle.
Enfin, nous appliquons à ces objets un `Physic Material`, qui leur donne une couleur, ainsi que des propriétés de friction et de rebond. La propriété de friction est la quantité qui limite le glissement des surfaces l'une sur l'autre. La friction se présente sous deux formes, dynamique et statique. La friction statique est utilisée lorsque l'objet est immobile, elle fixe un seuil de force pour déplacer l'objet. Si une force suffisamment importante est appliquée à l'objet, il commencera à bouger. C'est alors que la friction dynamique entre en jeu. La friction dynamique tente de ralentir l'objet lorsque son `Collider` est en contact avec un autre `Collider`. La même logique s'applique au modèle de rebond.
Il a été ajouté Unity, permettant de simuler visuellement et physiquement ce système, la gestion de l'entraînement supervisé avec le plug-in `ML-Agent` (Machine Learning Agent), offrant la possibilité de définir les différents éléments nécessaires à un apprentissage par renforcement. ML-Agents est un projet open source qui permet aux jeux et aux simulations de servir d'environnements pour l'entraînement d'agents intelligents.
L'unité de mesure par défaut dans la suite logicielle est le mètre, et tous les objets qui s'y trouvent sont mis à l'échelle à l'aide d'un facteur d'échelle.

2.2.2 Images simulées

Les entrées du réseau de neurones de l'apprentissage par renforcement (détaillées dans la partie 3) sont des images RVB acquises par une caméra. Dans la simulation numérique, ces images sont simulées par un `GameObject` particulier, une Caméra permettant de générer des images de notre système, un exemple de cette image est montrée dans la figure 4. Les paramètres de la caméra virtuelle ont été choisis pour imiter le comportement d'une caméra classique du commerce avec une fréquence d'acquisition de 24 images par seconde pour une dimension d'image de 480×270 pixels.

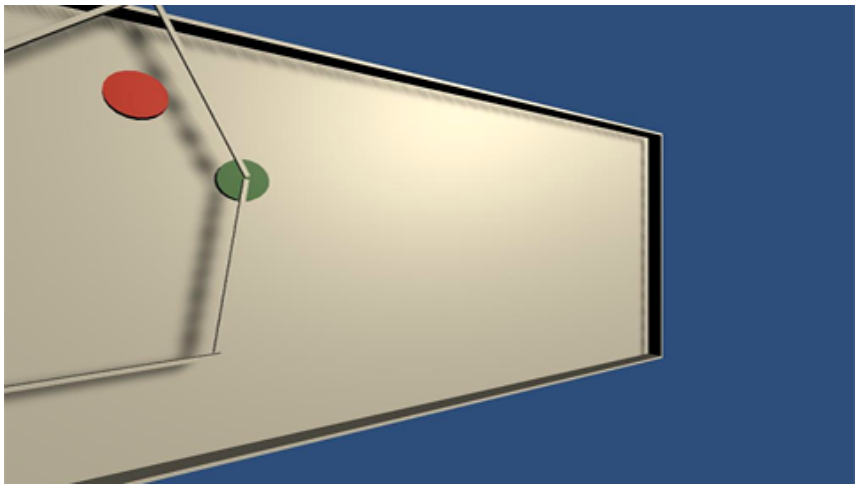

FIGURE 4 – Image générée par la caméra virtuelle

2.2.3 Modèle de la table, du robot et des éléments de la modélisation

La table est simplement modélisée par 5 cubes dont les facteurs d'échelle respectifs ont été modifiés pour être le plus proche de la réalité. Tous ces cubes sont définis par les composants nécessaires aux

fonctionnement physique du moteur (`Rigidbody`, `Collider`,`Physic Material`. Une attention particulière est portée sur les paramètres de friction afin d'obtenir un comportement avec le palet identique à celui que l'on peut avoir sur le système réel et mimer le fonctionnement sous coussin d'air. Tout ces objets sont fixés dans la simulation et ne possèdent aucun degré de liberté.
Le robot 5-barres est construit par 4 cubes dont les facteurs d'échelle respectifs ont été modifiés pour être le plus conformes à la réalité. Les cubes sont liés par les liaisons définies par le modèle présenté en 2.1.1.
Le palet et le butoir sont modélisés chacun, par 1 cylindre dont les facteurs d'échelle respectifs et les masses ont été modifiés pour être le plus conformes à la réalité. Ces cylindres sont définis par les composants nécessaires au fonctionnement physique du moteur (`Rigidbody`, `Collider`,`Physic Material`). Une attention particulière est portée sur les paramètres de friction afin d'obtenir un comportement avec la table identique à celui que l'on peut avoir sur le système réel.
Le butoir est fixé à l'extrémité du robot conformément par un encastrement à l'extrémité de l'un éléments distaux du robot.

3 Apprentissage par renforcement

3.1 Processus de Décision de Markov

Pour optimiser les actions d'un agent à la réalisation d'une tâche, un Processus de Décision de Markov (PDM) peut être utilisé. Comme décrit dans [14] ou [15], le PDM suivant est utilisé et représenté par un tuple $(\mathcal{S}, \mathcal{A}, \mathcal{F}, R)$, où $\mathcal{S}$ est l'espace d'état, $\mathcal{A}$ est l'espace des actions, R est la fonction de récompense, et $\mathcal{F}(s'|s, a)$ est le modèle de transition d'état ($s, s' \in \mathcal{S}$, où s est l'état au temps t et s' l'état au prochain pas de temps, $a \in \mathcal{A}$). Le comportement est déterminé par une politique de contrôle computationnelle $\pi_\Phi^{comp}(a|s)$, où Φ représente les paramètres de la politique. π_Φ^{comp} est représenté comme un réseau de neurones profond, avec Φ constitué des poids et des biais du réseau. L'objectif de l'apprentissage est de trouver les valeurs des paramètres de la politique qui maximisent la récompense cumulée : $\mathbb{E}\left[\sum_{t=0}^{T} R(s_t, a_t)\right]$, où T est la longueur de l'épisode.
Pour notre application :

- $\mathcal{S}$ est constitué des images simulées par la caméra décrite dans 2.2.2 avec un exemple illustré dans la figure 4.
- $\mathcal{A}$ consiste en l'incrémentation ou la décrémentation des couples Γ_1 et Γ_2 appliqués sur les éléments proximaux du robot.
- $\mathcal{R}$ la fonction de récompense définie par les états ci-dessous :
  - perte de 1 si le palet demeure dans la demi-zone allouée au robot
  - gain de $5\times(vitessedupalet)$ si celui-ci est dans la demi-zone adverse
  - gain de 0,0025 si le palet se trouve à la frontière entre les deux zones de jeu

C'est ce choix du système de récompense qui indique le comportement que doit adopter le robot. Ainsi avec le choix précédent et ces quantités empiriques de récompense, la notion de marquer des points n'est pas prises en compte, le robot doit uniquement renvoyer le palet avec une vitesse maximum dans la zone adverse.

3.2 Réseau de neurones

Comme précisé précédemment, π_Φ^{comp}, le réseau de neurones est construit de la manière suivante et illustrée dans la figure 5 :

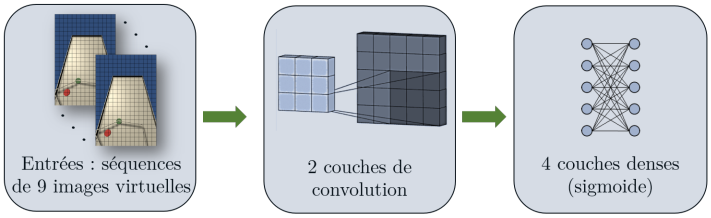

FIGURE 5 – Représentation schématique du réseau de neurone

Les entrées du réseau sont 9 images générées par la caméra virtuelle avec une fréquence de 24 images par seconde.
Les deux premières couches sont des couches de convolutions permettant au réseaux d'extraire les *features* en appliquant des opérations de filtrage par convolution et d'alimenter les couches suivantes, une explication du fonctionnement des couches de convolutions est proposée par Krizhesky et al. dans [16]
Les 4 couches suivantes sont des couches denses ayant pour fonction d'activation une sigmoïde. Cette architecture a été choisie, car les couches de convolution permettent le traitement efficace d'image [17]

3.3 Agent d'apprentissage

Avec la fonctionnalité *Ml-Agent*, un agent est défini dans la simulation. C'est celui-ci qui va suivre la politique π_{Φ}^{comp} et évoluer au sein de l'environnement. En l'occurrence il s'agit ici du robot 5 barres, dont les éléments proximaux sont soumis aux couples Γ_1 et Γ_2. Ces couples sont pilotés par l'algorithme d'apprentissage par renforcement.

3.4 Épisode d'entraînement

Ainsi une fois l'environnement totalement simulé et l'agent mis en place en son sein, l'épisode d'entraînement commence. Au début de l'épisode, la position initiale du robot est choisie aléatoirement dans son espace de travail. Une instance du palet est générée à la surface de la zone adverse avec une vitesse et une direction aléatoire (en direction néanmoins du robot). Cet épisode de simulations cesse au bout de 7s de simulation. Un visuel de cette simulation à un instant t est donné en figure 6

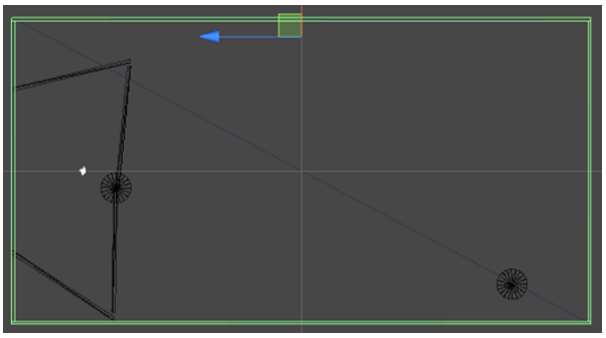

FIGURE 6 – Extrait d'un épisode de simulation

4 Résultats

Après de nombreuses instances de simulation, et par le *monitoring* de la fonction de récompense, l'agent optimise la politique mise en place et les résultats de simulation montrent que le robot renvoie le palet

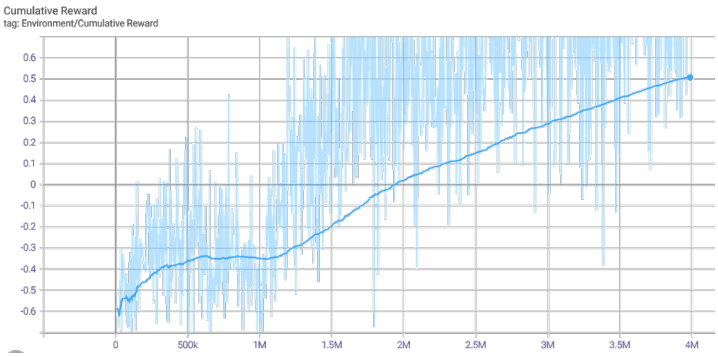

FIGURE 7 – Récompense cumulée de l'agent au cours des épisodes d'entraînement

dans la majorité des cas avec une vitesse suffisante pour atteindre la zone adverse. Sur la figure 7, malgré une stagnation et des résultats peu satisfaisants jusqu'à 1 million d'épisodes, la fonction récompense croît, montrant que l'agent s'applique correctement à la politique choisie. Les visualisations numériques montrent un robot renvoyant convenablement le palet dans la plupart des cas. La méthodologie ainsi proposée permet de générer une loi de commande en couple sur un système à partir de prise de vue. Une autre difficulté apparue lors du développement de cette méthode est le réglage des hyperparamètres non entraînables du réseau tel que la taille des noyaux des filtres de convolutions, le nombre de couches de neurones, le nombre de neurones par couche ou des paramètres d'entraînement comme le *learning rate* ou le paramètre β de la loi de distribution bêta. Dans cette application, des choix ont été faits de manière empirique jusqu'à l'obtention de résultats satisfaisants, par exemple différents choix de paramètres β sont illustrés en figure 8.

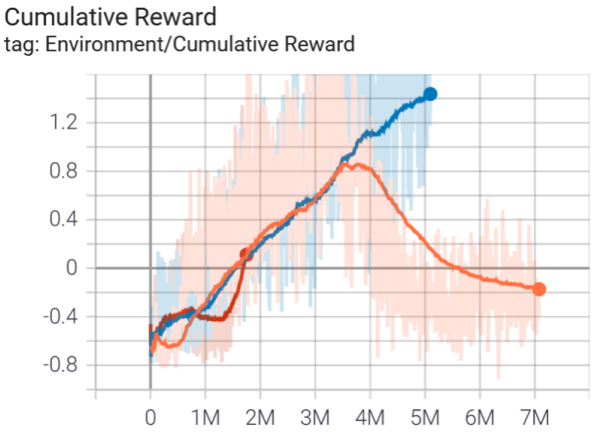

FIGURE 8 – Récompense cumulée de l'agent pour différents Beta de la loi de distribution

5 Conclusion et Perspective

Les prochaines étapes de validation de cette méthodologie sont l'implémentation sur le système réel et la validation de la loi de commande au point de vue du fonctionnement. Les principaux problèmes qui peuvent apparaître sont tous inclus dans le *reality gap* et mettent en exergue les difficultés de modélisations ainsi que leurs limites. Afin de palier à ces problèmes, il viendra une étape de *randomization* de la simulation, comme présenté dans [18]. Cela consiste en la parallélisation de l'entraînement : différents systèmes numériques sont instanciés (*i.e.* : plusieurs agents travaillent à l'optimisation d'un seul réseau),

chacune de ces instances étant légèrement différentes les unes des autres au point de vue des paramètres de simulation.
En faisant varier autour de la valeur originelle des facteurs, comme le poids des pièces, les paramètres de friction où les paramètres environnementaux de luminosité, de couleur et de contraste, on obtiendra une loi de commande plus robuste. Cette randomisation d'environnement de simulation permettra également de prendre en compte les problèmes d'*overfitting* inhérents à l'apprentissage par renforcement [19].

Remerciements

Ce travail a bénéficié d'une aide de l'État gérée par l'Agence Nationale de la Recherche au titre du programme "Investissements d'Avenir" dans le cadre du Laboratoire d'Excellence IMobS3 (ANR-10-LABX-0016) et de l'Initiative d'Excellence IDEX-ISITE CAP 20-25 (ANR-16-IDEX-0001).

Références

[1] Shuyang Chen and John T. Wen. Industrial robot trajectory tracking using multi-layer neural networks trained by iterative learning control. *arXiv*, 2019.

[2] Youngwoon Lee, Edward S. Hu, Zhengyu Yang, Alex Yin, and Joseph J. Lim. IKEA Furniture Assembly Environment for Long-Horizon Complex Manipulation Tasks. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–11, nov 2019.

[3] Subhrajit Bhattacharya and Siddharth Talapatra. Robot Motion Planning Using Neural Networks : A Modified Theory. *International Journal of Lateral Computing*, 2(1) :9–13, 2005.

[4] Hartmut Surmann, Christian Jestel, Robin Marchel, Franziska Musberg, Houssem Elhadj, and Mahbube Ardani. Deep Reinforcement learning for real autonomous mobile robot navigation in indoor environments. *arXiv*, 2020.

[5] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-Real : Learning Agile Locomotion For Quadruped Robots. *CoRR Computing Research Repository*, 2018.

[6] Adrien Koessler, Alexandre Goldsztejn, Sebastien Briot, and Nicolas Bouton. Certified detection of parallel robot assembly mode under Type 2 singularity crossing trajectories. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 6073–6079, 2017.

[7] M. Gautier and Wisama Khalil. Communications : Direct Calculation of Minimum Set of Inertial Parameters of Serial Robots. *IEEE Transactions on Robotics and Automation*, 6(3) :368–373, 1990.

[8] Mohsen Afrough and Ahmed Abu Hanieh. Identification of Dynamic Parameters and Friction Coefficients : of a Robot with Planar Serial Kinemtic Linkage. *Journal of Intelligent and Robotic Systems : Theory and Applications*, 94(1) :3–13, 2019.

[9] Bradley E. Bishop and Mark W. Spong. Vision-Based Control of an Air Hockey Playing Robot. *IEEE Control Systems*, 19(3) :23–32, 1999.

[10] Akio Namiki, Sakyo Matsushita, Takahiro Ozeki, and Kenzo Nonami. Hierarchical processing architecture for an air-hockey robot system. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1187–1192, 2013.

[11] Sébastien Briot and Wisama Khalil. *Dynamics of Parallel Robots*. Springer, 2015.

[12] Sebastien Briot and Ilian A. Bonev. Are parallel robots more accurate than serial robots ? *Transactions of the Canadian Society for Mechanical Engineering*, 31(4) :445–455, 2007.

[13] Ahmed Joubair, Mohamed Slamani, and Ilian A. Bonev. Kinematic calibration of a five-bar planar parallel robot using all working modes. *Robotics and Computer-Integrated Manufacturing*, 29(4) :15–25, 2013.

[14] Tianjian Chen, Zhanpeng He, and Matei Ciocarlie. Hardware as Policy : Mechanical and Computational Co-Optimization using Deep Reinforcement Learning. *Conference on Robot Learning (CoRL) 2020*, (CoRL), 2020.

[15] Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. Epopt : Learning robust neural network policies using model ensembles. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, (October), 2019.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Proceedings of the 25th International Conference on Neural Information Processing Systems*, pages 1–1432, 2012.

[17] Aseem Patil and Milind Rane. Convolutional Neural Networks : An Overview and Its Applications in Pattern Recognition. *Smart Innovation, Systems and Technologies*, 195 :21–30, 2021.

[18] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles MacKlin, Jan Issac, Nathan Ratliff, and DIeter Fox. Closing the sim-to-real Loop : Adapting simulation randomization with real world experience. *Proceedings - IEEE International Conference on Robotics and Automation*, 2019-May :8973–8979, 2019.

[19] Reda Bahi Slaoui, William R. Clements, Jakob N. Foerster, and Sébastien Toth. Robust Visual Domain Randomization for Reinforcement Learning. *International Conference on Learning Representation (ICLR)*, (2019) :1–14, 2019.