Rapport final de Projet Ingénierie Entreprise

Développement de contrôleurs de marche pour le robot bipède Bolt basés sur de l'apprentissage par renforcement

Auteurs :
CC Guillaume BERTHELOT
Rafael CELENTE
Alexandre GOUILLER
Lucie MOUILLE
Maxime SEGURA

Encadrants :
Thomas FLAYOLS (LAAS)
Jean BOUSQUET
(AIRBUS)

Certificat d'authenticité

Nous soussignés, BERTHELOT Guillaume, CELENTE Rafael, GOUILLER Alexandre, MOUILLE Lucie, SEGURA Maxime, certifions qu'il s'agit d'un travail original et que toutes les sources utilisées ont été indiquées dans leur totalité. Nous certifions, de surcroît, que nous n'avons ni recopié ni utilisé des idées ou des formulations tirées d'un ouvrage, article ou mémoire, en version imprimée ou électronique, sans mentionner précisément leur origine et que les citations intégrales sont signalées entre guillemets.

Fait à Toulouse, le 18 mars 2024

Résumé du projet

Le projet s'articule autour du robot BOLT conçu par OPEN DYNAMIC ROBOT INITIATIVE. Ce robot bipède est conçu pour faire avancer la recherche en matière de contrôle de robot en fournissant les plans et le mode opératoire pour la construction du robot de manière open source et à faible coût. Le robot se meut autour de 6 articulations. La mission qui est confiée au groupe est de concevoir une politique de commande pour le robot, en utilisant préférentiellement une technique d'apprentissage par renforcement. Après une phase initiale de recherche bibliographique, nous nous concentrons sur l'implémentation d'une méthode d'apprentissage par renforcement de type Proximal Gradient Policy dans un environnement simulé. L'essentiel du travail consiste ensuite à déterminer une fonction de récompense qui permettra à l'algorithme de converger vers un politique de marche robuste. Nous procédons par itérations successives, en nous inspirant des travaux mentionnés en bibliographie. La politique déterminée pourra être ensuite implémentée directement sur le robot. La principale contrainte du projet réside dans le fait que chacun des entraînements nécessitent plusieurs heures. Afin de mener à bien ce projet, nous utilisons les techniques de gestion de projets.

Abstract

The project revolves around the BOLT robot designed by the OPEN DYNAMIC ROBOT INITIATIVE. This bipedal robot is intended to advance research in robot control by providing open-source plans and procedures for constructing the robot at a low cost. The robot operates with 6 joints. The assigned mission for the group is to design a control policy for the robot, preferably using reinforcement learning techniques. After an initial phase of literature review, the focus shifts to implementing a Proximal Gradient Policy reinforcement learning method in a simulated environment. The core of the work then involves determining a reward function to guide the algorithm towards a robust walking policy. We proceed through successive iterations, drawing inspiration from the referenced literature. The determined policy can subsequently be directly implemented on the robot. The main constraint of the project lies in the fact that each training session requires several hours. To successfully carry out this project, we employ project management techniques.

Remerciements

Ce projet, auquel nous avons consacré de nombreuses heures durant environ six mois, nous a permis de rencontrer et échanger avec de nombreuses personnes qui nous ont aiguillé dans notre travail. Sans eux, le projet n'aurait pu aboutir, ainsi nous voulions au nom du groupe de PIE travaillant sur Bolt, les remercier vivement.

Nous tenons particulièrement à remercier M. Thomas Flayols, notre client, pour sa disponibilité et sa pédagogie. La clarté de son discours nous a permis d'avancer de façon efficace sur ce projet malgré sa complexité indéniable.

Nous tenons également à remercier Pierre-Alexandre Paleziart pour sa disponibilité et les explications fournies pour l'utilisation du code.

Nous tenons enfin à remercier M. Jean Bousquet, notre tuteur pour ce projet. Il fût d'une aide précieuse pour orienter notre stratégie et le succès de ce projet n'aurait sans doute pas vu le jour sans ses conseils avisés.

Sommaire

Introduction

Dans le domaine en constante évolution de la robotique, la conception de robots bipèdes capables de marcher de manière autonome demeure un défi majeur. L'équilibre, la coordination des mouvements et l'adaptation à des environnements variables sont autant d'éléments complexes nécessitant une compréhension approfondie et une ingénierie précise. Dans cette perspective, le présent projet s'engage à explorer les potentialités prometteuses de l'apprentissage par renforcement pour enseigner à un robot bipède baptisé Bolt l'art subtil de la marche autonome. Supervisé par le Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS), cette initiative s'inscrit dans une démarche multidisciplinaire, conjuguant les principes de l'intelligence artificielle et de la mécanique robotique.

Au-delà de l'aspect technique, ce projet représente également une exploration des possibilités offertes par la collaboration entre l'homme et la machine. En effet, la programmation de Bolt nécessitera une compréhension approfondie des mécanismes de la marche humaine, ainsi qu'une capacité à traduire ces connaissances en algorithmes et en comportements robotiques. De ce fait, ce projet incarne une symbiose entre l'expertise humaine et les capacités computationnelles des robots.

1 Organisation des activités

1.1 Présentation générale du projet

1.1.1 Présentation du contexte et des parties prenantes

L'équipe Gepetto du LAAS CNRS est spécialisée dans l'étude des mouvements et la commande des robots anthropomorphes. L'équipe développe des plateformes robotique open source comme le quadrupède Solo12 ou le bipède Bolt. Les méthodes d'implémentation de contrôleurs de marche exploitant le modèle physique du robot ou utilisant de l'apprentissage profond par renforcement sont déjà matures pour ce qui est du robot quadrupède Solo12. L'objectif de la mission, proposée par l'équipe Gepetto, est d'étendre et adapter les méthodes utilisées sur Solo12 au robot bipède Bolt. Puisqu'il s'agit d'un projet de recherche, il y aura un importante phase exploratoire par recherche bibliographique sur les méthodes d'apprentissages par renforcement.

Notre client est l'équipe Gepetto du LAAS CNRS et plus particulièrement Thomas Flayols qui est notre principal contact. Du point de vue gestion de projet, notre tuteur est Jean Bousquet qui est chef de projet chez Airbus. Il encadre le projet et pourra nous transmettre ses compétences en matière de gestion de projet.

1.1.2 Description du projet

Ce projet est un projet open source impliquant plusieurs entités. Il est donc important de définir nos objectifs et les périmètres de notre travail.

Notre objectif est d'implémenter un contrôleur de marche par apprentissage profond par renforcement pour le robot bipède Bolt. Cela implique de :

- Se mettre à niveau sur le procédé d'apprentissage par renforcement et les principes de fonctionnement des algorithmes d'entraînement
- Capitaliser sur les résultats et apporter des solutions aux problèmes rencontrés par l'équipe de PIE précédente ayant travailler sur le même projet
- Faire un état de l'art sur les méthodes d'apprentissage par renforcement de la marche bipède
- Entraîner une politique faisant marcher le robot en simulation
- Entraîner une politique suffisamment robuste pour pouvoir être transférée sur le robot Bolt du LAAS

Pour décrire correctement notre périmètre de travail, il est important de dé-

tailler :

1) ce que l'on ne fait pas (Construction du robot, Transfert de la politique sur le robot réel (code C++))

2) ce qu'on peut faire si le temps et les ressources le permettent (apprentissage d'une politique de marche permettant une trajectoire non rectiligne, sur un sol irrégulier, de monter des escaliers ou de s'équilibrer à vitesse nulle en minimisant l'énergie)

3) et, évidemment, les résultats attendus (un état de l'art concernant la problématique (utilisation de l'apprentissage par renforcement massif), une politique de marche du robot établie par apprentissage par renforcement massif, un détail des critères de récompenses)

1.1.3 Identification des contraintes et hypothèses

Pour correctement cerner le projet, il est important de lister l'ensemble des contraintes et hypothèses de travail.

Les contraintes temporelles :
— Fin de projet mi-mars (délivrance des livrables le 6 mars et soutenance le 13 mars)
— Temps limité consacré au projet (103h par étudiant dans l'emploi du temps)

Les contraintes matérielles :
— Nécessité d'utiliser des ordinateurs équipés de carte graphiques RTX pour l'entraînement de la politique (ordinateur disponible au LAAS)

Les contraintes logicielle :
— Prise en main du logiciel de simulation Isaac Gym avec lequel Gepetto souhaite travailler

Nos hypothèses de travail :
— Le fichier URDF décrivant le robot Bolt est suffisamment fidèle à la réalité pour pouvoir effectuer les simulations et être utilisé lors de l'apprentissage. C'est un risque toujours d'actualité, tant que nous n'aurons pas validé le passage sur le robot réel.

1.2 Les livrables du projet

Plus précisement, les objectifs de ce projets nous ont été détaillés par le LAAS et par l'ISAE-Supaero au travers d'un ensemble de livrables.

La liste des produits livrables au client est :

— État de l'art bibliographique
— Méthode de marche robuste en simulation (fonctions récompenses choisies et paramètres associés)
— Vidéo de démonstration sur robot réel et document résumant les adaptations nécessaires pour que la méthode établie en simulation fonctionne en réel
— Rapport technique du projet et code documenté

La liste des livrables demandés par l'école est :

— Plan de développement du projet
— Rapport technique du projet
— Executive Summary
— Planches de la présentation de la soutenance

1.3 Définition des tâches et des jalons du projet

En ayant en tête d'ensemble des livrables demandés pour ce projet, nous pouvons l'organiser en un certain nombre de tâches et de jalons dont la description est donnée ci-dessous.

Une fois la définition des tâches réalisée, nous avons aussi du prendre en compte l'ensemble des jalons imposés par l'école et le client pour mener à bien le projet.

Les jalons clés dans ce projet seront ceux qui démontrent de l'aboutissement du projet. Côté client, il s'agit du rendu de la politique de marche robuste (JC2) qui représente la concrétisation de nos recherches et dont la restitution est clé pour la finalité du projet.

Titre	Description	Activités principales	Entrées nécessaires	Durée
Lecture des articles scientifiques	Cette tâche constitue l'étape de recherche de la réalisation de l'état de l'art	Seclection d'un ensemble d'articles, prise de note sur les articles, rédaction d'un document préliminaire	Un accès commun à la bibliographie (ZOTERO)	1 mois et 1 semaine jours
Rédaction de l'état de l'art	Rédiger la synthèse des documents lus	Activités de rédaction	Lecture des articles scientifiques	1 mois
Prise en main de l'environnement de simulation	La réussite du projet repose notamment sur la capacité à prendre en main un environnement de simulation. Celui-ci est indispensable pour développer des stratégies de marche	Benchmarking et recherche d'environnement cloud, test environnement Nvidia Isaac sur machines	Documentations Nvidia	2 jours semaines
Convergence vers une méthode en simulation	C'est un jalon essentiel avant le déploiement sur le robot. Il s'agit de trouver une politique convergente pour la marche du robot	Boucle (simulation - tuning de la fonction de récompense.). Recherche des stratégies efficaces.	Environnement de simulation, données physiques du robot	3 mois et 2 semaines

Intégration sur le robot en réel	Cette tâche nous permettera de valider notre méthode sur le robot	Validation de la politique sur un autre simulateur (PyBullet). Implémentation de la politique dans le robot via un code de transfert C++. Notons que le laboratoire est responsable du transfert de la politique dans le robot.	Politique validée	1 mois
Rédaction des rapports (rapport technique & summary)	Le rapport est un rendu exigé pour la validation du PIE.	Rédaction.	Résultats de l'entrainement en simulation	1 mois et 1 semaine
Préparation de la soutenance	La soutenance est un rendu nécessaire pour le PIE	Préparation du support de présentation, entraînements de groupe à la soutenance	Environnement de simulation, données physiques du robot	2 semaines

TABLE 1 – Tableau des tâches

Type de jalon	Identification	Descriptif du jalon
Client	JC1	Le rendu d'un état de l'art détaillé sur le sujet du développement de contrôleurs de marche pour les robots bipèdes, basés sur de l'apprentissage par renforcement.
Client	JC2	Le rendu d'une politique de marche robuste en simulation et sur le robot réel.
Client	JC3	Le rendu d'un document détaillant le périmètre établi pour la performance de la politique de marche.
École	JE1	Une première revue de projet au cours de laquelle les avancées et le plan de développement mis à jour seront présentés.
École	JE2	Une seconde revue de projet au cours de laquelle les avancées et le plan de développement mis à jour seront présentés.
École	JE3	La soutenance orale de présentation du projet.
École	JE4	La remise des livrables imposés par l'école : Rapport, Support de Soutenance, Plan de Développement et Executive Summary.

TABLE 2 – Tableau des jalons

Côté école, les jalons clés sont la soutenance orale et le rendu final des livrables (JE3 et JC4) d'une grande importance pour l'évaluation finale de ce projet.

1.4 Planning

Afin de mener à bien toutes les tâches que nous avons détaillés ci-dessous et de respecter les jalons imposés par le client et par l'école, nous les avons organisés dans un planning. Le planning renseigné ci-dessous est la version que nous avons utilisée tout au long du projet. En effet, dès le début du projet, nous avons vu assez large pour la durée des tâches et surtout, nous avons anticipé le fait que certaines tâches dureraient jusqu'à la fin du projet et les avons planifiées ainsi.

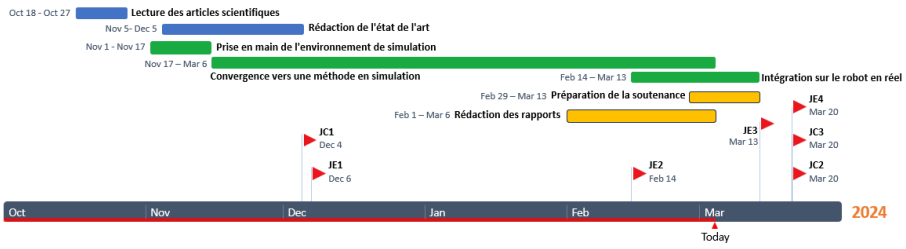

FIGURE 1 – Diagramme de Gantt

Comme représenté dans le diagramme de Gantt ci dessus, nos tâches se séparent en 3 groupes :

— l'état de l'art
— la détermination d'une politique de marche en simulation et la validation de cette politique en réel
— la préparation de la soutenance et des rendus "Ecole"

Ces trois groupes sont assez indépendants et leurs tâches peuvent être réalisées en parallèle. Cependant, nous remarquons qu'il existe 2 couples de tâches ayant un fort lien de précédence :

— la lecture des articles scientifiques doit être terminée avant de pouvoir commencer la rédaction de l'état de l'art
— l'environnement doit être correctement pris en main avant de commencer à travailler sur la détermination d'une politique de marche robuste.

Au moment de planifier le projet, nous avions noté qu'il faudrait accorder une attention particulière aux deux premières tâches de ces couples étant donné qu'un retard sur celles-ci pourraient engendrer un retard sur les deuxièmes. C'est d'ailleurs ce que nous avons fait et cela nous a permis de ne pas créer de retard dans notre projet.

Par ailleurs, il n'existe pas de chemin allant d'un bout à l'autre du projet et ne liant que des tâches avec des liens de précédence. Ceci est un avantage de notre découpage, qui permet d'éviter un maximum l'influence du retard accumulé sur les différentes tâches au cours du projet sur la date de fin globale du projet.

1.5 Organisation de l'équipe

1.5.1 Répartition des rôles au sein de l'équipe

Au sein de notre équipe, nous bénéficions d'une diversité de profils enrichissante. Certains membres possèdent une solide expertise en Machine Learning avec des connaissances préalables en Reinforcement Learning, tandis que d'autres sont néophytes dans ces domaines. Souhaitant tous approfondir nos connaissances dans ces sujets, nous avons opté dès le départ pour une approche collaborative.

La lecture des articles et la rédaction de l'état de l'art ont été réparties équitablement entre tous les membres. Pendant les phases d'appropriation de l'environnement et d'entraînement en simulation, chacun a pris tour à tour la main, derrière l'ordinateur, et nous avons réfléchi collectivement, durant chacune des phases de l'entraînement en simulation.

Cette démarche de réaliser toutes les tâches ensemble a permis aux membres les moins expérimentés de bénéficier des connaissances des plus aguerris, favo-

risant ainsi une montée en compétence homogène au sein de l'équipe. Aujourd'hui, chacun se sent à l'aise tant sur les aspects techniques que sur la mise en œuvre de notre politique.

Bien que cette approche de travail en équipe puisse sembler moins efficace sur le papier, le rythme de réunions que nous nous sommes fixé a garanti une progression en accord avec le planning établi tout en veillant à ce que chaque membre soit pleinement investi dans le projet. Le planning de nos réunions est présenté en détail ci-dessous.

1.5.2 Réunions/ drumbeat de l'avancement et revues

Pour assurer une progression efficace et une coordination au sein de notre équipe, nous avons établi une approche structurée pour organiser notre travail et valider nos développements en cours.

Nous organiserons des réunions hebdomadaires avec l'équipe et nos clients au LAAS. Ces sessions sont cruciales pour suivre la progression du projet et recevoir des retours immédiats sur notre travail, en particulier sur le développement de techniques d'apprentissage par renforcement profond. Cette interaction régulière nous permettra d'aligner nos efforts avec les attentes du client et d'apporter les ajustements nécessaires rapidement.

De plus, à deux reprises durant le projet, nous avons planifié des réunions de gestion de projet. Celles-ci seront plus complètes, se concentrant sur les aspects plus larges du projet, y compris l'allocation des ressources, les ajustements de calendrier et la planification stratégique.

1.5.3 Communication

Nous avons utilisé la plateforme Matrix pour les communications hebdomadaires avec le client. Cette plateforme facilite les discussions en temps réel, le partage rapide de fichiers et une collaboration efficace. Elle est la méthode de communication principale entre les chercheurs du laboratoire Gepetto, facilitant leur portée. De plus, pour la communication interne de l'équipe, un canal Slack a été mis en place où l'équipe envoie activement des mises à jour du projet, des questions et des suggestions.

Concernant le partage et le stockage des connaissances, nous avons utilisé Google Drive pour stocker et partager tous nos documents relatifs au projet, ainsi qu'un repository Github partagé et public pour tout le code et les données produites. Étant donné la nature de recherche du projet, un dépôt Zotero a également été créé pour partager et accéder à toute la littérature pertinente utilisée pour baser notre recherche.

1.6 Risques et opportunités

Tout au long du projet, nous n'avons cessé d'évaluer les risques et opportunités afin de parer à d'éventuels problèmes qui nous auraient fait perdre du temps, nous éloignant ainsi des deadlines prévues. En voici un tableau récapitulatif mis à jour.

Id	Catégorie	Cause	Risque	Conséquence	Proba	Réponse	État
1	Technique	aucune méthode testée ne fonctionne	non-convergence de la solution en simulation	non aboutissement du projet	neutre	étude approfondie avant implémentation	Non avéré
2	Technique	panne	plus d'accès à un ordinateur pouvant entraîner le robot	mise en pause des recherches jusqu'à réparation	très faible	travailler sous AWS, ou paperspace	Non avéré
3	Technique	imperfection du simulateur	le robot a été entraîné grâce à un biais du simulateur	difficultés lors du passage sur le vrai robot	neutre	comparer à ce qui a été fait sur Solo12, analyse des différences	Actif
4	Personnel	maladie, raisons personnelles	absences forcées de certains d'entre nous	ralentissement du projet	faible	prévoir de la marge dans les deadlines	Actif
5	Technique	choix d'une méthode erronée	perte de temps sur une méthode ne fonctionnant pas	perte de temps et risque de ne pas aboutir	neutre	état de l'art approfondi, faire beaucoup de simulations	Actif (mais la probabilité diminue)
6	Technique	mauvaise manipulation	casser le robot	impossibilité d'utilisation jusqu'à réparation	faible	travailler avec soin	Actif

TABLE 3 – Tableau des risques

probailité \ criticité	faible	moyen	haut	très haut
très haut				
haut				
moyen		3	5	
faible	4	6		

TABLE 4 – Échelle de Risque

2 Rapport technique

2.1 Présentation de l'environnement de travail

Le Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS) est un établissement renommé, reconnu pour ses avancées significatives dans le domaine de la robotique et des systèmes autonomes. Fort de son expertise multidisciplinaire et de ses infrastructures de pointe, le LAAS offre un cadre propice à l'innovation et à la recherche. Dans ce contexte, le projet visant à apprendre à Bolt à marcher s'inscrit comme une continuation logique de l'exploration des frontières de la robotique. Pour nous guider et nous permettre d'avancer dans ce projet, il nous a été conseillé par le LAAS de travailler avec le simulateur d'Isaac Gym, au vu de leur expérience avec ce logiciel et donc, en dehors de son efficacité, de l'aide qu'ils pouvaient nous apporter.

L'utilisation d'Isaac Gym revêt une importance cruciale dans le cadre du projet visant à enseigner à Bolt la marche autonome. Cette plateforme de simulation offre une gamme de fonctionnalités avancées qui facilitent le développement et l'optimisation des politiques de contrôle pour le robot bipède. Voici quelques-unes des caractéristiques d'Isaac Gym qui rendent son utilisation particulièrement intéressante dans ce projet.

Tout d'abord, Isaac Gym prend en charge l'importation de fichiers URDF et MJCF, ce qui permet d'intégrer facilement des modèles de robots complexes dans l'environnement de simulation. De plus, la décomposition automatique des maillages 3D en convexes facilite la représentation physique réaliste des objets, garantissant ainsi des interactions précises entre Bolt et son environnement.

L'API tensorielle accélérée par GPU d'Isaac Gym constitue un avantage majeur pour le projet, offrant une rapidité et une efficacité accrues dans l'évaluation de l'environnement et l'application d'actions. Cette fonctionnalité tire parti de la puissance de calcul parallèle offerte par les unités de traitement graphique (GPU), ce qui permet d'accélérer considérablement les opérations nécessaires à la simulation et à l'apprentissage par renforcement.

Tout d'abord, l'utilisation du GPU pour l'évaluation de l'état de l'environnement permet une exécution parallèle des calculs, ce qui permet de traiter rapidement de vastes ensembles de données et de générer des informations sur l'environnement en temps réel. Cela est essentiel pour des simulations en temps réel où Bolt doit interagir avec son environnement de manière dynamique, recevoir des données sensorielles et prendre des décisions rapidement. De plus, l'application des actions par le biais de l'API tensorielle accélérée par GPU permet une exécution parallèle des calculs, ce qui réduit considérablement le temps nécessaire pour effectuer des opérations telles que le calcul des déplacements et

des changements d'état de Bolt en réponse à ces actions. Cette rapidité d'exécution est cruciale pour assurer des expériences d'apprentissage par renforcement efficaces, où Bolt doit recevoir des rétroactions en temps opportun et ajuster ses actions en conséquence pour améliorer sa performance. L'API tensorielle accélérée par GPU d'Isaac Gym offre donc une solution efficace pour évaluer rapidement l'état de l'environnement et appliquer des actions dans le cadre de simulations en temps réel et d'expériences d'apprentissage par renforcement. En exploitant la puissance de calcul parallèle des GPU, cette fonctionnalité contribue à accélérer le processus d'apprentissage de Bolt et à garantir des interactions fluides et réalistes avec son environnement virtuel.

De plus, Isaac Gym offre une prise en charge de divers capteurs environnementaux, tels que la position, la vitesse, la force et le couple, ce qui permet à Bolt de percevoir son environnement et d'adapter ses actions en conséquence. Cette capacité à intégrer des données sensorielles complexes est essentielle pour l'apprentissage de la marche autonome, car Bolt doit réagir de manière adaptative aux informations provenant de son environnement.

Isaac Gym permet également la randomisation des paramètres physiques à l'exécution, ce qui signifie que les conditions de simulation peuvent être modifiées dynamiquement pour encourager l'apprentissage robuste et la généralisation des politiques de contrôle. Cette fonctionnalité est précieuse pour exposer Bolt à une variété de situations et de défis, ce qui contribue à renforcer sa capacité à marcher de manière autonome dans des environnements divers.

De plus, la prise en charge de la cinématique inverse et du calcul du jacobien permet d'explorer des stratégies de contrôle avancées pour Bolt, en lui permettant de planifier et d'exécuter des mouvements complexes de manière efficace.

Enfin, bien que la version complète d'Isaac Gym soit actuellement en cours de développement dans le cadre de NVIDIA Omniverse Isaac Sim 2022.1, une version préliminaire autonome est disponible pour les chercheurs et les universitaires qui souhaitent explorer le potentiel de l'apprentissage par renforcement accéléré par GPU. Cette version autonome reste accessible pour faciliter les travaux de recherche dans le domaine de la robotique autonome et de l'apprentissage par renforcement jusqu'à ce que la version complète soit disponible.

Isaac Gym se révèle non seulement être une plateforme de simulation avancée, mais elle propose également une gamme d'algorithmes d'apprentissage par renforcement prêts à l'emploi, parmi lesquels se trouve le Proximal Policy Optimization (PPO). Dans le cadre du projet visant à enseigner à Bolt, le robot bipède, l'art de la marche autonome sous la tutelle du Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS), l'utilisation de l'algorithme PPO intégré à Isaac Gym revêt une importance cruciale.

Le choix de l'algorithme PPO est motivé par sa capacité à gérer efficacement

les politiques de contrôle dans des environnements complexes et dynamiques, tout en garantissant la stabilité de l'apprentissage. PPO est réputé pour sa robustesse, sa capacité à gérer les grands espaces d'actions et sa facilité d'implémentation, ce qui en fait un choix idéal pour l'apprentissage de la locomotion chez Bolt.

2.1.1 Objectif de PPO

L'objectif de PPO est d'optimiser la politique d'un agent tout en évitant des mises à jour drastiques qui pourraient conduire à une divergence. Cela est accompli en limitant les changements dans la politique à chaque itération.

2.1.2 Formulation mathématique

Soit π_θ la politique paramétrée par θ, où θ sont les paramètres de la politique. L'objectif de PPO peut être formulé comme suit :

$$L(\theta) = \mathbb{E}\left[\min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} A^{\pi_{\theta_{\text{old}}}}(s,a), \text{clip}\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)}, 1-\varepsilon, 1+\varepsilon\right) A^{\pi_{\theta_{\text{old}}}}(s,a)\right)\right]$$

où $A^{\pi_{\theta_{\text{old}}}}(s,a)$ est l'avantage estimé de l'action a dans l'état s sous la politique $\pi_{\theta_{\text{old}}}$. La fonction de perte $L(\theta)$ est ensuite utilisée pour mettre à jour les paramètres de la politique.

2.1.3 Algorithme PPO

Algorithm 1 Proximal Policy Optimization (PPO)

Initialiser la politique π_θ et le réseau de valeur V_ϕ
for itération $= 1, 2, \ldots,$ nombre d'itérations **do**
    Collecter des trajectoires en interagissant avec l'environnement en utilisant π_θ
    Calculer les avantages $A^{\pi_{\theta_{\text{old}}}}(s,a)$
    for chaque epoch d'optimisation **do**
        Mettre à jour la valeur V_ϕ en minimisant la perte quadratique
        Calculer la perte de politique $L(\theta)$ et mettre à jour π_θ avec un optimiseur de votre choix (Adam,SGD,..)
    end for
    Mettre à jour $\theta_{\text{old}} \leftarrow \theta$
end for

Dans ce contexte, Isaac Gym offre un cadre complet pour l'entraînement de Bolt en utilisant l'algorithme PPO. La plateforme fournit une simulation réaliste où Bolt peut interagir avec son environnement virtuel, recevoir des rétroactions sensorielles et ajuster ses mouvements en conséquence.

2.1.4 Environnement de simulation

L'environnement de simulation produit par Nvidia implémente directement la simulation physique de l'environnement (ISAAC SIM) ainsi que différents algorithmes de reinforcement learning. La modélisation du robot est réalisée au travers d'un fichier au format urdf, pour Unifieds Robotics Description Format. Ce fichier au standard XML comporte les informations concernant la structure du robot, les masses et inerties de ses membres, ses articulations et les volumes de collision.

Les paramètres de la simulation et de l'entraînement sont implémentés dans un fichier .yaml . C'est ici que l'on renseigne en particulier l'information de taille du réseau de neurones, les informations concernant les plages d'explorations de la simulation, les différentes valeurs pour les paramètres de gains K_p, K_d ainsi que différentes informations utiles aux fonction de récompense.

Enfin, l'essentiel de la routine de simulation est implémentée dans un fichier python bolt.py. Il comporte les entrées sorties pour la simulation ainsi que le calcul des fonctions de récompenses. C'est ici que se concentre une grosse partie du travail de conception de la fonction de récompense.

2.2 Entraînement en simulation vers une politique de marche robuste

2.2.1 Description du modèle numérique

Afin de décrire le modèle numérique du robot, nous avons utilisé le format de fichier URDF. Ce format permet de spécifier l'ensemble des solides appelés "link" constituant le robot avec leurs caractéristiques physiques associées telles que leur masse ou leur moment d'inertie. Chaque solide est aussi associé à un repère qui lui est attaché, permettant de suivre sa trajectoire en simulation. L'URDF permet également de spécifier les liaisons appelées "joint" entre un solide parent et un solide enfant. Dans notre cas, l'ensemble des liaisons sont des pivots, sauf les liaisons entre les pieds et le bas des jambes qui sont des liaisons encastrement. Nous avons modélisé les pieds et les bas des jambes par des solides distincts, même s'ils sont encastrés, car nous souhaitions conserver les repères attachés aux centres des pieds afin de pouvoir spécifier les récompenses "foot slip" et "foot clearance" qui utilisent la vitesse des pieds ainsi que la condition de réinitialisation des épisodes. Un dernier élément que le format URDF permet de spécifier est les volumes de collision des solides. Ces volumes sont essentiels pour modéliser l'interaction du robot avec le sol et entre les éléments du robot. Dans notre modélisation, les pieds possèdent des volumes de collision sphériques, le bas et le haut des jambes sont modélisés par des capsules, et la base par une boîte. Ces formes simples facilitent le calcul des collisions lors de l'entraînement. Nous avons choisi de ne pas modéliser de volume de collision pour les hanches, car la collision des hanches avec un autre élément n'est physiquement pas possible sur le système réel et que l'interaction entre elles et le sol n'est pas pertinente lors de l'apprentissage.

Pour observer le modèle de collision du robot, cliquer sur **cette vidéo**

2.2.2 Architecture de contrôle

Notre architecture de contrôle contient un réseau de neurone à action directe ("feedforward") avec une structure de couche [512x256x128] et des fonctions d'activation "unité exponentielle linéaire" (ELU). Les entrées de notre réseau de neurones sont notées $\mathbf{S_t} = \{{}^{\mathbf{b}}\mathbf{v}, {}^{\mathbf{b}}\boldsymbol{\omega}, {}^{\mathbf{b}}\mathbf{g}, {}^{\mathbf{b}}\mathbf{cmd}, \mathbf{q_t} - \mathbf{q_0}, \dot{\mathbf{q}}_\mathbf{t}, \mathbf{a_{t-1}}\}$ et sont définies ci-dessous :

- ${}^{\mathbf{b}}\mathbf{v} \in \mathbb{R}^3$: La vitesse linéaire de la base qui est perçue grâce par la centrale inertielle.
- ${}^{\mathbf{b}}\boldsymbol{\omega} \in \mathbb{R}^3$: La vitesse de rotation de la base qui est perçue grâce par la centrale inertielle.
- ${}^{\mathbf{b}}\mathbf{g} \in \mathbb{R}^3$: Le vecteur de l'accélération gravitationnelle projeté dans le repère de la base qui est perçue par la centrale inertielle.

— ${}^{b}\mathbf{cmd} \in \mathbb{R}^3$: Le vecteur de la commande de vitesse dans le repère de la base composé des vitesses de translation cmd_x, cmd_y et de la vitesse de rotation cmd_ω.
— $\mathbf{q_t} - \mathbf{q_0} \in \mathbb{R}^6$: Le vecteur des positions des articulations centré sur les positions par défaut $\mathbf{q_0}$.
— $\dot{\mathbf{q}}_\mathbf{t} \in \mathbb{R}^6$: Le vecteur des vitesses des articulations.
— $\mathbf{a_{t-1}} \in \mathbb{R}^6$: Le vecteur des actions précédentes.

Le réseaux de neurone contrôle le robot dans l'espace des positions. On note $\mathbf{a_t}$ les actions du réseaux au temps t et on définit les positions désirées comme ceci :

$$\hat{\mathbf{q}}_\mathbf{t} = \mathbf{q_0} + \sigma \mathbf{a_t}$$

avec $\sigma = 0.5$. Cette position désirée sert de référence pour le contrôleur proportionnel dérivé servant à calculer le couple des articulations. Le vecteur des couples des articulations est noté τ et est calculé comme ceci :

$$\tau = K_p(\hat{\mathbf{q}}_\mathbf{t} - \mathbf{q_t}) + K_d(\hat{\dot{\mathbf{q}}}_\mathbf{t} - \dot{\mathbf{q}}_\mathbf{t})$$

avec $K_p = 4N.m.rad^{-1}$ et $K_p = 0.3N.m.s.rad^{-1}$

2.2.3 Vers une première politique

TABLE 5 – Reward functions

Effect	Expression
Command tracking	$r_v = \exp\left(-4\left(\|\|{}^{b}\mathbf{cmd}_{xy} - {}^{b}\mathbf{v}_{xy}\|\|^2\right)\right)$
	$r_\omega = 0.5\exp\left(-4\left(\|\|{}^{b}\mathbf{cmd}_{\omega} - {}^{b}\omega_z\|\|^2\right)\right)$
Keep balance	$r_{v_z} = -2{}^{b}v_z^2$
	$r_{\omega_{xy}} = -0.015({}^{b}\omega_x^2 + {}^{b}\omega_y^2)$
Smooth motions	$r_\tau = -0.00001\|\|\dot{\tau}\|\|^2$
	$r_{\dot{q}} = -0.0002\|\|\dot{\mathbf{q}}_t\|\|^2$
	$r_{\text{acc}} = -2e^{-7}\left\|\left\|\frac{\dot{\mathbf{q}}_t - \dot{\mathbf{q}}_{t-1}}{dt}\right\|\right\|^2$
	$r_a = -0.01\|\|\mathbf{a}_t - \mathbf{a}_{t-1}\|\|^2$
Joint limit	$r_q = 0.4\exp\left(-\|\|\mathbf{q}_t - \mathbf{q}_d\|\|\right)$

```
# Velocity tracking reward
lin_vel_error = torch.sum(torch.square(self.commands[:, :2] -
    base_lin_vel[:, :2]), dim=1)
ang_vel_error = torch.square(self.commands[:, 2] -
    base_ang_vel[:, 2])
rew_lin_vel_xy = torch.exp(-lin_vel_error/0.25) *
    self.rew_scales["lin_vel_xy"]
```

```
rew_ang_vel_z = torch.exp(-ang_vel_error/0.25) *
    self.rew_scales["ang_vel_z"]
```

$$r_v = \exp\left(-4\left(||^b\mathbf{cmd}_{xy} - {}^b\mathbf{v}_{xy}||^2\right)\right)$$
$$r_\omega = 0.5\exp\left(-4\left(||^b\mathbf{cmd}_{\omega} - {}^b\omega_z||^2\right)\right)$$

En considérant uniquement ces deux fonctions de récompense qui permettent au robot de suivre une direction et une vitesse, le robot réussi à suivre la trajectoire mais avec une démarche peu naturelle. Pour observer le résultat, cliquer sur **cette vidéo**

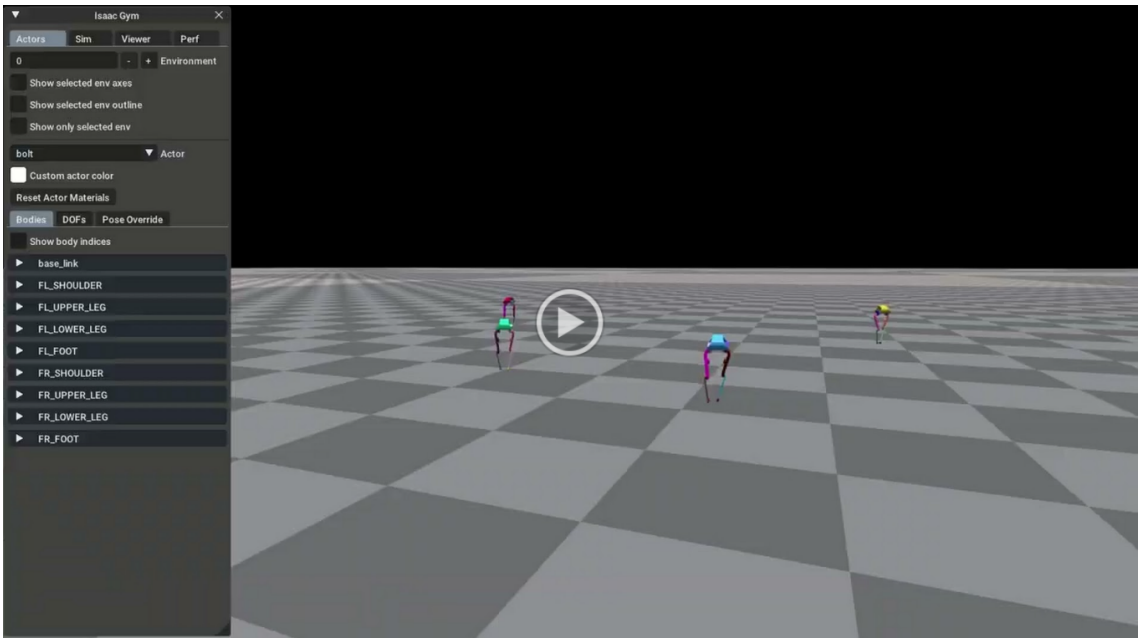

FIGURE 2 – Miniature de la vidéo du robot entraîné, entraînement 1

On obtient les courbes suivantes :

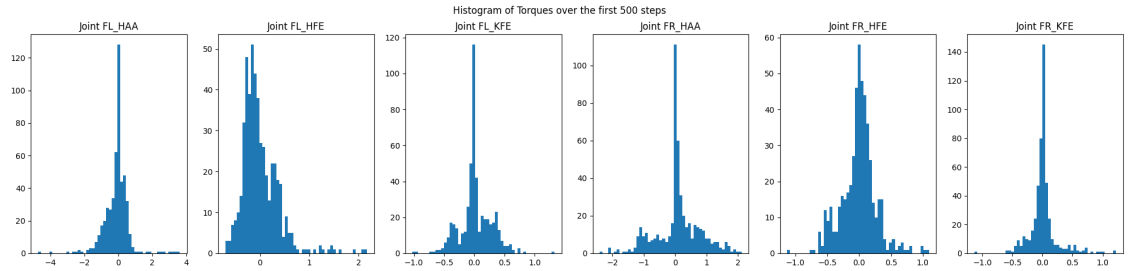

FIGURE 3 – Histogrammes des couples appliqués à chaque articulation (observations.$(N.m)^{-1}$)

```python
# foot slip penalty (solo 12 article)
contact = torch.norm(self.contact_forces[:, self.feet_indices, :], dim=2) > 1
rew_slip = torch.sum(contact * torch.square(torch.norm(self.foot_velocities[:, :, :2], dim = 2)), dim = 1) * self.rew_scales["slip"]
```

```python
# foot clearance penalty (solo 12 article)
rew_clearance = torch.sum(torch.square(self.foot_positions[:, :, 2] - self.max_height) * torch.sqrt(torch.norm(self.foot_velocities[:, :, :2], dim = 2)), dim = 1) * self.rew_scales["clearance"]
```

On ajoute ensuite ces deux fonctions de récompense pour éviter que le robot glisse, et pour le forcer à lever les pieds. Pour observer le résultat, cliquer sur **cette vidéo**

FIGURE 4 – Miniature de la vidéo du robot entraîné, entraînement 2

On obtient les courbes suivantes :

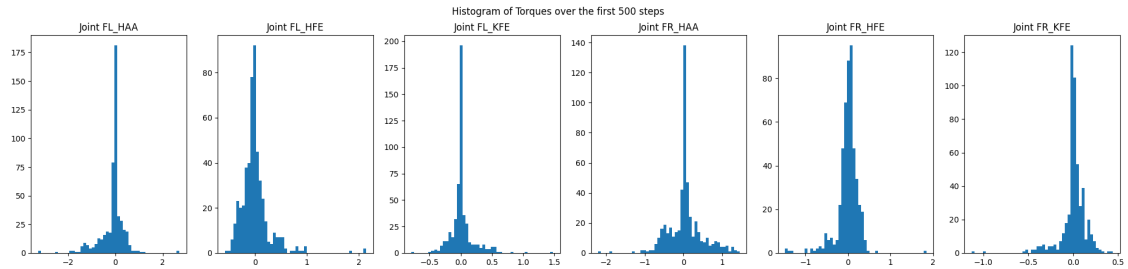

FIGURE 5 – Histogrammes des couples appliqués à chaque articulation (observations.$(N.m)^-1$)

2.2.4 Premiers pas

```
# torque limitation
rew_torque = torch.sum(torch.square(self.torques), dim=1) *
    self.rew_scales["torque"]
```

$$r_\tau = -0.00001||\dot{\tau}||^2$$

```
# acceleration limitation
rew_dof_acc = torch.sum(torch.square(self.dof_vel -
    self.last_dof_vel), dim=1) * self.rew_scales["dof_acc"] /
    (self.dt ** 2)
```

$$r_{\text{acc}} = -2e^{-7}\left|\left|\frac{\dot{\mathbf{q}}_t - \dot{\mathbf{q}}_{t-1}}{dt}\right|\right|^2$$

```
# action variation limitation
rew_action_change = torch.sum(torch.square(self.actions -
    self.last_actions), dim=1) * self.rew_scales["action_change"]
```

$$r_a = -0.01||\mathbf{a}_t - \mathbf{a}_{t-1}||^2$$

```
# velocity limitation
rew_dof_vel = torch.sum(torch.square(self.dof_vel), dim=1) *
    self.rew_scales["dof_vel"]
```

$$r_{\dot{q}} = -0.0002||\dot{\mathbf{q}}_t||^2$$

On ajoute ensuite ces quatre fonctions de récompense pour limiter les mouvements erratiques qui ne sont pas réalistes comme par exemple le fait que les robot avance avec des très petits pas extrêmement rapides. Pour observer le résultat, cliquer sur **cette vidéo**

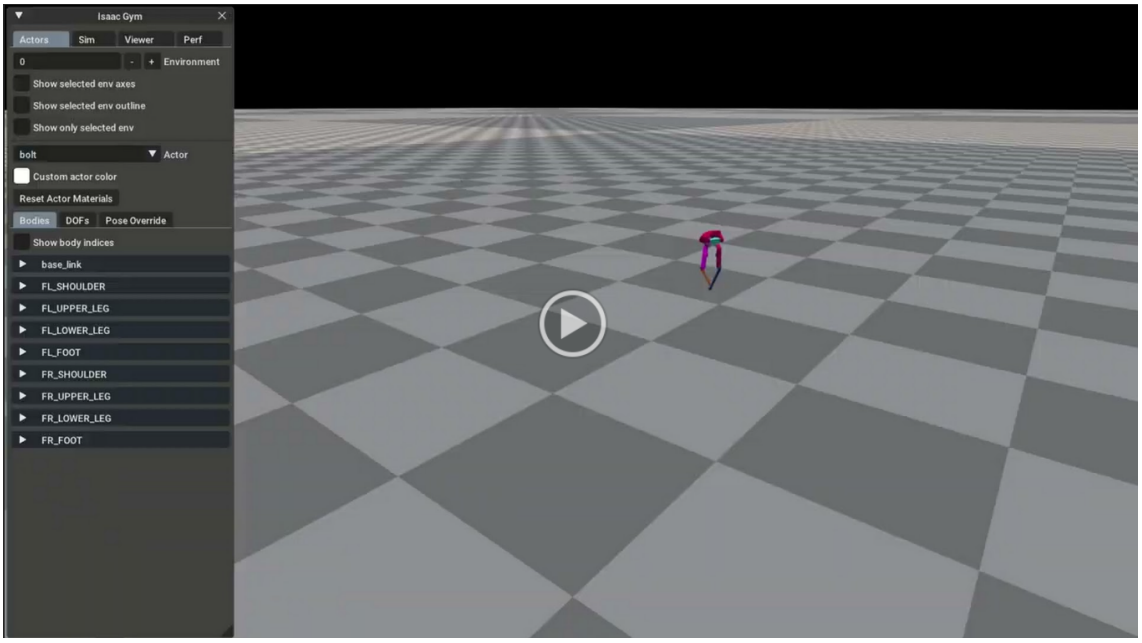

FIGURE 6 – Miniature de la vidéo du robot entraîné, entraînement 3

On obtient les courbes suivantes :

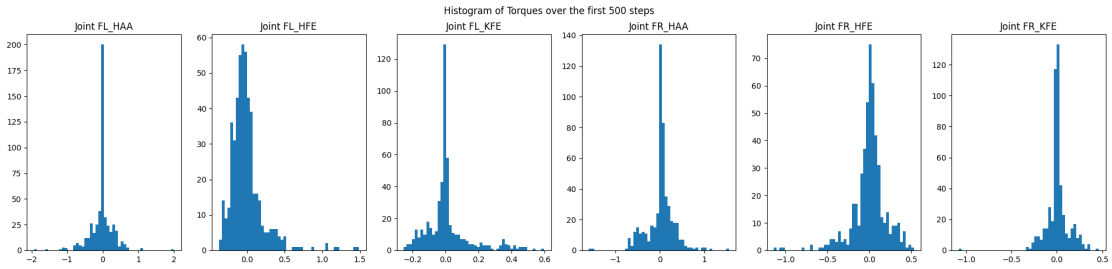

FIGURE 7 – Histogrammes des couples appliqués à chaque articulation (observations.$(N.m)^{-1}$)

2.2.5 Perfectionnement de la méthode

```
# keep balance
rew_balance_rotation = torch.sum(torch.square(base_ang_vel[:,
    :2]), dim=1) * self.rew_scales["balance_rotation"]
```

$$r_{\omega_{xy}} = -0.015({}^{b}\omega_x^2 + {}^{b}\omega_y^2)$$

```
# Penalty on difference between current position and initial
    position
rew_joint_limit = torch.exp(-torch.norm(self.dof_pos -
    self.default_dof_pos, dim=1)) *
    self.rew_scales["joint_limit"]
```

$$r_q = 0.4 \exp\left(-||\mathbf{q}_t - \mathbf{q}_d||\right)$$

```
# Keep balance
rew_balance_speed = torch.square(base_lin_vel[:, 2]) *
    self.rew_scales["balance_speed"]
```

$$r_{v_z} = -2^b v_z^2$$

On ajoute enfin ces trois fonctions récompense afin de stabiliser la base, ce qui permet d'éviter des mouvements étranges au niveau des genoux. Pour observer le résultat, cliquer sur **cette vidéo**

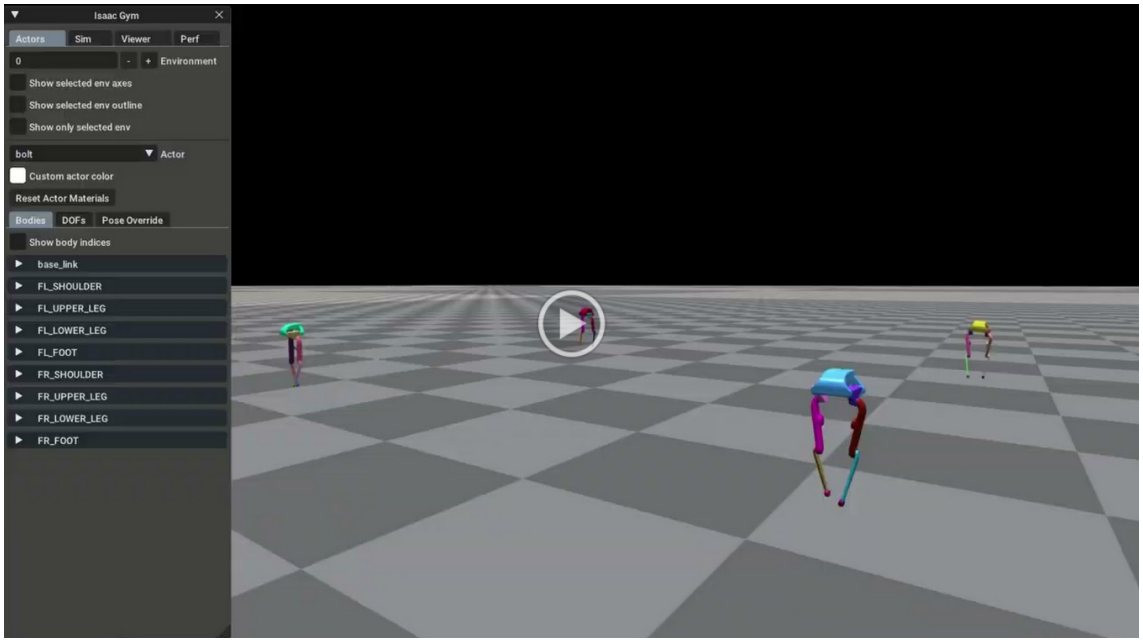

FIGURE 8 – Miniature de la vidéo du robot entraîné, entraînement 4

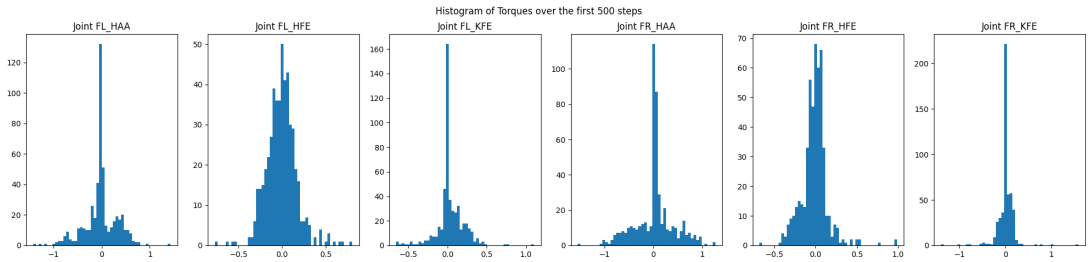

FIGURE 9 – Histogrammes des couples appliqués à chaque articulation (observations.$(N.m)^{-1}$)

2.2.6 Amélioration de la robustesse de notre politique, en vue d'un passage sur le robot réel

Grâce aux entraînements et à l'ajout successif de fonctions récompenses adéquates, détaillées ci-dessus, nous avons pu aboutir à une politique décrivant une marche fluide et réaliste. Cependant, en vue de transférer cette politique sur le robot réel, cette politique doit être rendue plus robuste afin d'être entraînée pour toutes les perturbations du monde extérieur que le robot pourra percevoir en marchant.

Pour le moment, notre robot n'a été entraîné qu'avec une vitesse de consigne rectiligne de 1m/s selon x. Nous souhaitons donc qu'il apprenne à tourner pour palier à toute commande du joystick en réel. Nous l'entraînons donc avec une vitesse de commande qui, pour chaque environnement d'entraînement, sera une composition aléatoirement choisie des intervalles suivants : [-1 m/s, 1 m/s] sur x, [-1 m/s, 1 m/s] sur y et [-0.2 rad, 0.2 rad] en lacet. Cet entraînement est un succès, nous choisissons donc de continuer dans cette démarche.

Par ailleurs, nous savons que, en simulation les capteurs sont considérés comme idéaux alors qu'en réalité, il existe des bruits dans les mesure. Pour palier ce problème, nous décidons d'ajouter un bruit Gaussien aux observations (entrées du réseau). Plus précisément, nous calculons un vecteur de bruit de la taille du vecteur des observations qui vaut, pour chaque observation, le produit d'un niveau de bruit (choisi à 1 ici), d'un facteur multiplicateur propre à chaque observation et du facteur d'échelle de l'observation (s'il existe). Ensuite, nous ajoutons aux observations, à chaque étape, un vecteur de valeurs prises entre 0 et 1 multiplié par le vecteur de bruit détaillé ci-dessus. Nous lançons un entraînement avec l'implémentation de ce bruit, cequi nous donne de bons résultats.

Observation	Facteur de bruit
Position des actionneurs	0.01
Vitesse des actionneurs	1.5
Vitesse linéaire	0.1
Vitesse angulaire	0.2
Vecteur de gravité	0.05
Commandes	0
Actions précédentes	0

TABLE 6 – Tableau des facteurs de bruits de mesure

Nous notons que les commandes et les actions précédentes ne sont pas brui-

tées parce que ce sont des données qui ne sont pas acquises par un capteur.

Finalement, toujours dans l'idée de rendre notre politique plus robuste, nous souhaitons entraîner notre réseau avec des paramètres physiques du robot changeants (masse des membres, Kp, Kd, coefficient de frottement, etc.). Pour ce faire, nous définissons pour chaque paramètre physique que nous souhaitons modifier au cours de la simulation, un intervalle (dépendant du paramètre physique), un nombre de pas (9000 ici), un type d'interpolation ("linéaire" ici) et une distribution ("uniforme" ici). Ainsi, avec cette implémentation, pour chaque environnement d'entraînement, un facteur multiplicateur choisi uniformément dans l'intervalle est appliqué au paramètre physique en question. Ce facteur multiplicateur est interpolé linéairement sur les 9000 premiers pas puis reste constant durant la fin de l'entraînement. Ainsi, cette modification du paramètre physique est appliqué progressivement au cours de l'entraînement. Cette technique d'apprentissage est appelée le "curriculum learning". L'entraînement avec la modification aléatoire des paramètres physiques du robot est un succès.

Paramètre physique	Intervalle de bruit
Masse	[0.8, 1.2]
Coefficient de frottement	[0.8, 1.2]
Kd	[0.9, 1.1]
Kp	[0.9, 1.1]

TABLE 7 – Tableau des intervalles de bruits sur les paramètres physiques

Pour chacune des implémentations détaillées ci-dessus, nous lançons un entraînement spécifique afin d'analyser leurs impacts.

Les résultats de l'entraînement du dernier réseau, comprenant toutes ces implémentions sont donnés ci-dessous.

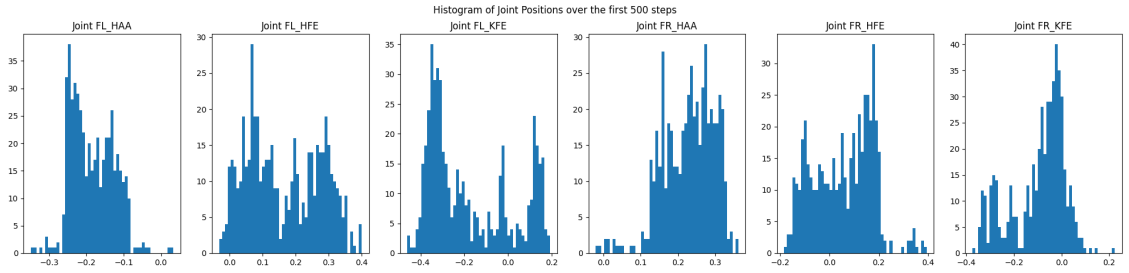

FIGURE 10 – Histogrammes des positions à chaque articulation (observations.rad^{-1})

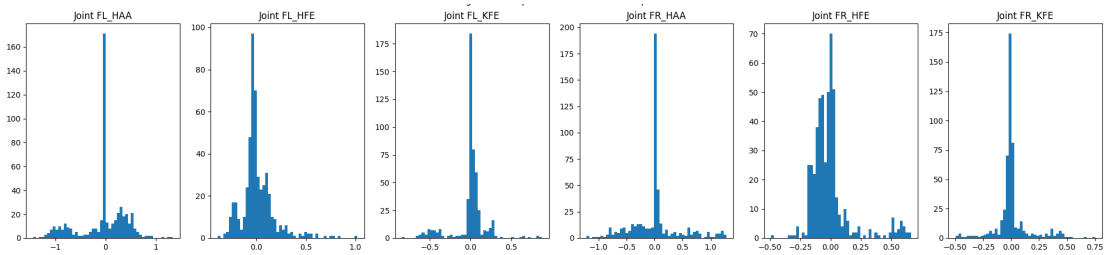

FIGURE 11 – Histogrammes des couples appliqués à chaque articulation(observations.$(N.m)^{-1}$)

En vue du passage sur le robot réel, nous avons encore besoin d'améliorer la robustesse du robot, notamment en l'entraînant avec des commandes en vitesses changeantes au cours de l'entraînement et en lui imposant des force perturbatrices ponctuellement pour le déséquilibrer.

Si le planning le permet, ces deux implémentations seront réalisées d'ici à la soutenance pendant laquelle nous pourrons présenter les résultats associés.

2.3 Intégration sur le robot en réel

Une étape importante a été franchie en implémentant un entraînement capable de générer une politique qui semble visuellement cohérente avec la démarche attendue pour le robot. Jusqu'ici, notre principale méthode d'analyse résidait dans la visualisation en temps réel de la simulation. Or le fait que le robot puisse marcher dans l'environnement virtuel n'est pas une condition suffisante pour la validation de la politique. Encore faut-il que les consignes fournies par le réseau et lues par le robot soient des consignes réalistes. La simulation n'implémente pas de valeurs de saturations pour les consignes de couple ou de vitesse angulaire. Le robot simulé peux en effet répondre sans limite aux consignes de couples ou de vitesses angulaires. Il est donc essentiel, avant d'initier toute démarche de transfert vers le robot réel, de tester les valeurs du couple lues au niveau des actionneurs, ainsi que celle des vitesses angulaires. Le tableau 8 rappelle ces valeurs limites.

Paramètre	**Valeur Limite**
Couple maximal	3 N.m
Vitesse angulaire maximale	16 rad.s^{-1}

TABLE 8 – Valeurs limites des actionneurs

Nous analysons en conséquence les données issues des la simulation, la figure 15 montre les valeurs de couple pour la démarche du robot et on remarque que l'on ne dépasse jamais les valeurs limites. Il en va de même pour les vitesses angulaires qui ne dépassent pas la valeur seuil. (on note une marge significative avec cette dernière).

Nous avons ensuite comparé les valeurs des couples de forces et leur vitesse angulaire maximale avec d'autres essais de simulation dotés de structures de récompense différentes. Dans l'image 12, sont montrés les résultats pour les couples de forces lors d'une session d'entraînement où uniquement les récompenses liées à la vitesse et à l'angle ont été utilisées. Pour l'image 13, les mêmes résultats sont présentés mais pour l'essai d'entraînement au cours duquel les récompenses liées à la vitesse, à l'angle, au "slip" et à la "clearence" ont été utilisées.

Ce que nous pouvons observer est que, bien que les récompenses de glissement et de dégagement aient aidé à améliorer la politique de marche, ce n'est qu'avec l'introduction des récompenses limitant le couple que (sans détériorer la politique de marche) les couples se sont trouvés dans une plage considérée comme sûre pour une introduction dans le robot réel. Avec cela à l'esprit, il était sécuritaire d'introduire davantage de récompenses pour améliorer encore la politique tout en maintenant les couples articulaires limités.

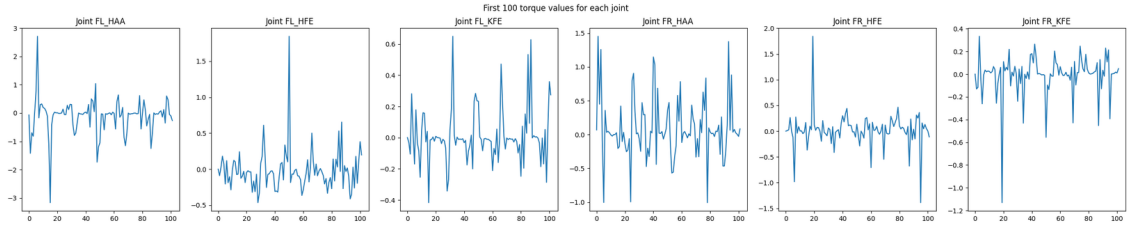

FIGURE 12 – Mesure du couple des actionneur (*velocity, angle, slip and clearence N.m*).

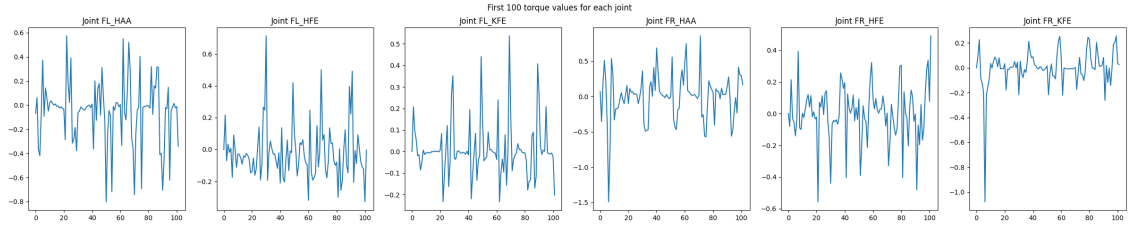

FIGURE 13 – Mesure du couple des actionneur (*velocity, angle, slip, clearence and torque limits*) *N.m*.

Pour définitivement valider la loi de commande, nous devons prendre une dernière précaution. L'ensemble des résultats a été obtenu au travers d'un simulateur, qui peut lui-même comporter des biais ou des phénomènes de bords. Nous devons donc vérifier nos résultats au travers d'un autre simulateur, dont on suppose qu'il comportera des biais et phénomènes de bord différents. Nous avons essayé d'intégrer notre robot sur Pybullet, un environnement open source. Nous utilisons pour cela un code fourni par le laboratoire.

Au moment du lancement d'une visualisation de notre robot entraîné sur le simulateur Pybullet, nous nous rendons compte que la simulation ne fonctionne pas pour différentes raisons : l'arrangement des vecteurs d'observations et d'actions n'est pas le même dans Isaac et dans Pybullet, les données de vitesses et de positions sont inversées. Après avoir compris d'où venait le problème, nous relançons la simulation et nous rendons compte qu'elle n'est pas stable. Cette fois-ci le problème est moins évident. En effet, nous avons lancé la simulation

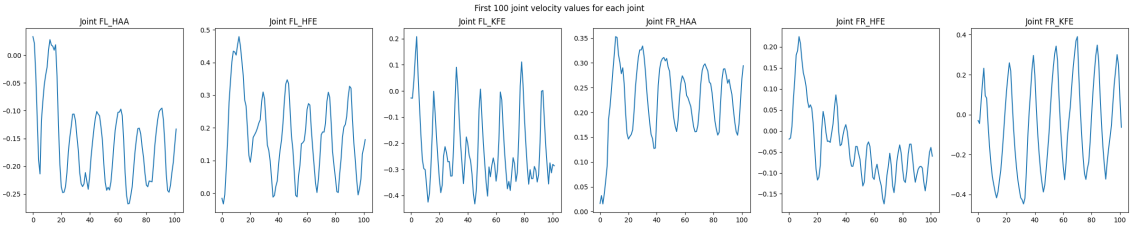

FIGURE 14 – Mesure de la vitesse angulaire au niveau des articulations ($rad.s^{-1}$)

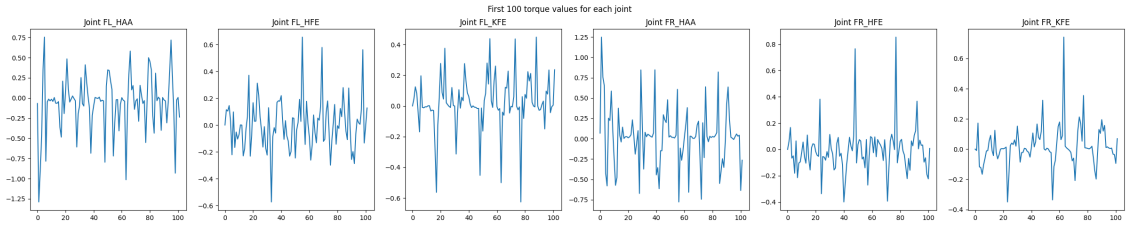

FIGURE 15 – Mesure du couple des actionneurs $(N.m)$

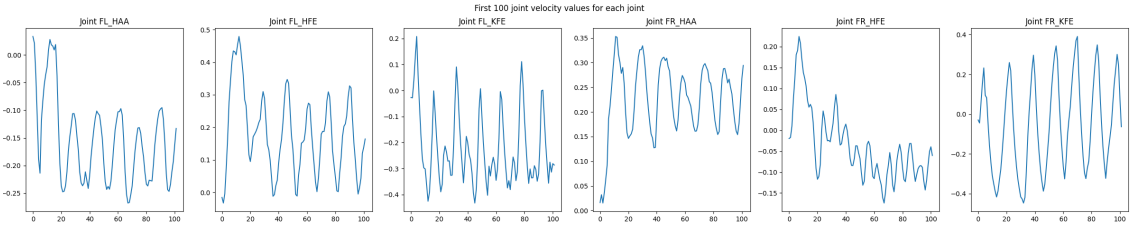

FIGURE 16 – Mesure de la vitesse angulaire au niveau des articulations $(rad.s^{-1})$

Pybullet avec la même fréquence pour le réseau de neurones et pour la simulation que lors de l'entraînement. Dans notre code, la fréquence du réseau de neurones et du contrôle / de la simulation sont corrélées et égales respectivement à 50Hz et 100 Hz. Ces fréquences sont assez basses mais la simulation est toujours apparue comme stable sur Isaac. Nous ne sommes pas tout à fait sûrs de la raison pour laquelle celle-ci est toujours apparue comme stable sur Isaac puisque lorsque nous passons sur Pybullet la simulation n'est pas du tout stable. Pour réussir le passage sim2sim et donc le passage en réel, il faudrait donc modifier la fonction pre-physics -steps du fichier bolt.py de façon à décorréler la fréquence du réseau de neurones et la fréquence du contrôle / de la simulation et pouvoir les configurer à 50 Hz et 1000 Hz, respectivement. Cette modification consiste à remplacer la fonction set-dof-position-target-tensor par un contrôle fait "à la main", comme décrit ci-dessous.

```
def pre_physics_step(self, actions):
    self.last_actions[:] = self.actions[:]
    self.actions = actions.clone().to(self.device)
    targets = self.action_scale * self.actions + self.default_dof_pos
    self.gym.set_dof_position_target_tensor(self.sim, gymtorch.unwrap_tensor(targets))
```

FIGURE 17 – Fonction pre-physics-step qui corrèle les fréquences du réseau et de la simulation

```python
def pre_physics_step(self, actions):
    self.actions = actions.clone().to(self.device)
    for i in range(self.decimation):
        torques = torch.clip(self.Kp*(self.action_scale*self.actions + self.default_dof_pos - self.dof_pos)
                             - self.Kd*self.dof_vel,
                             -80., 80.)
        self.gym.set_dof_actuation_force_tensor(self.sim, gymtorch.unwrap_tensor(torques))
        self.torques = torques.view(self.torques.shape)
        self.gym.simulate(self.sim)
        if self.device == 'cpu':
            self.gym.fetch_results(self.sim, True)
        self.gym.refresh_dof_state_tensor(self.sim)
```

FIGURE 18 – Fonction pre-physics-step qui décorrèle les fréquences du réseau et de la simulation

Cette étape de décorrelation nous semble clé pour la réussite du sim2sim. Cependant, n'ayant pas eu le temps de comprendre en profondeur tous les paramètres des deux simulateurs, nous craignons qu'il reste des soucis de "traduction" entre les deux simulateurs.

La dernière étape avant d'intégrer la loi de commande sur le robot est la traduction des poids du réseau en un langage compilé en C++. La routine de traduction est elle aussi fournie par le laboratoire. Cette routine pourra être employée dès lors que le problème décrit ci-dessus aura été réglé et que la robustesse du réseau face au changement de simulateur aura pu être vérifiée.

3 Conclusion

Le but de ce projet PIE était de déterminer une politique robuste pour la commande du robot bipède BOLT. Les principaux livrables étaient un rapport bibliographique sur le sujet et une politique entraînée dans une environnement de simulation. Notre approche pour répondre aux attendus s'est articulée autour de méthodes de gestion de projet éprouvées. Ainsi, nous avons pu rapidement décomposer le projet en tâches, identifier les tâches critiques mais aussi répartir le travail de façon efficace. Notre organisation nous a permis de fournir les livrables dans les délais impartis. Après un premier travail commun de recherche bibliographique, nous avons rédigé l'état de l'art. Cette première étape, indispensable dans ce cadre particulier de projet comportant une forte dimension de recherche, nous a permis de tracer les jalons suivants. Le premier jalon, critique, était d'être en mesure d'initialiser l'environnement de simulation. Nous avions bénéficié du retour d'expérience du groupe de l'an dernier car ceux-ci avaient rencontré des difficultés importante lors de cette étape. Nous avons choisi un environnement de simulation éprouvé ISAAC SIM, ayant fait ses preuves car déjà employé par le laboratoire avec succès sur un autre projet. Une fois l'environnement de simulation établi, nous avons pu mettre en place notre stratégie de recherche de fonction de récompense. Nous avons alors affiné, au travers de quatre phases de simulation, nos fonctions de récompenses. Nos principales sources d'inspiration pour le design de celles-ci sont issues de notre travail de recherche bibliographique. Nous avons finalement pu déterminer une politique capable de diriger le robot et qualifiée de robuste. Il est à noter que malgré notre limitation en capacité de calcul, notre approche a été relativement efficace. Nous rappellerons ici que des démarches similaires ont nécessité l'emploi de fermes de calcul GPU, alors que nous ne disposions que d'un unique GPU, certes puissant. La durée moyenne d'un entraînement étant de 8 heures, et ayant réalisé pas plus d'une vingtaine d'entraînement, nous sommes satisfait des résultats obtenus. Nous retiendrons de ce projet les avantages que procurent le dessin d'une stratégie et d'un plan tactique pour atteindre les objectifs fixés. La définition des jalons nous a permis de conserver la confiance en milieu de projet, alors que les résultats satisfaisant tardaient à venir. Nous savions que nous étions sur le trait. Nous espérons que ces résultats permettront de faire avancer le projet BOLT, nous sommes convaincus que le code réalisé pourra être réemployé pour des futurs projets. A l'instant de la rédaction de ce mémoire, les politiques calculées n'ont pas été implémentées sur le robot en réel. Bien qu'il ne s'agisse pas d'un livrable, nous espérons vivement pouvoir mener des essais d'ici la soutenance.

Références

[1] PPO Hyperparameters and Ranges. Proximal Policy Optimization (PPO) is... | by AurelianTactics | aureliantactics | Medium. https ://medium.com/aureliantactics/ppo-hyperparameters-and-ranges-6fc2d29bccbe.

[2] Proximal Policy Optimization. https ://openai.com/research/openai-baselines-ppo.

[3] Twin Delayed DDPG — Spinning Up documentation. https ://spinningup.openai.com/en/latest/algorithms/td3.html.

[4] Kübra Akbaş, Carlotta Mummolo, and Xianlian Zhou. Characterization of Human Balance through a Reinforcement Learning-based Muscle Controller, August 2023. Lucie.

[5] Michel Aractingi, Pierre-Alexandre Léziart, Thomas Flayols, Julien Perez, Tomi Silander, and Philippe Souères. Controlling the Solo12 quadruped robot with deep reinforcement learning. *Scientific Reports*, 13(1) :11945, 2023. Guillaume.

[6] AurelianTactics. PPO Hyperparameters and Ranges, July 2018.

[7] Gaurav Bhardwaj, Soham Dasgupta, N. Sukavanam, and R. Balasubramanian. Soft Soil Gait Planning and Control for Biped Robot using Deep Deterministic Policy Gradient Approach, June 2023. Maxime

Comment : Advances in Robotics (AIR) 2023 IIT Ropar.

[8] Yeonghun Chun, Junghun Choi, Injoon Min, Minsung Ahn, and Jeakweon Han. DDPG Reinforcement Learning Experiment for Improving the Stability of Bipedal Walking of Humanoid Robots. In *2023 IEEE/SICE International Symposium on System Integration (SII)*, pages 1–7. IEEE, 2023. Maxime.

[9] Jared Di Carlo, Patrick M. Wensing, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9, Madrid, October 2018. IEEE.

[10] Helei Duan, Bikram Pandit, Mohitvishnu S. Gadde, Bart Jaap van Marum, Jeremy Dao, Chanho Kim, and Alan Fern. Learning Vision-Based Bipedal Locomotion for Challenging Terrain, September 2023. Lucie.

[11] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing Function Approximation Error in Actor-Critic Methods, October 2018. Comment : Accepted at ICML 2018.

[12] Tuomas Haarnoja, Ben Moran, Guy Lever, Sandy H. Huang, Dhruva Tirumala, Markus Wulfmeier, Jan Humplik, Saran Tunyasuvunakool, Noah Y. Siegel, Roland Hafner, Michael Bloesch, Kristian Hartikainen, Arunkumar Byravan, Leonard Hasenclever, Yuval Tassa, Fereshteh Sadeghi, Nathan Batchelor, Federico Casarini, Stefano Saliceti, Charles Game, Neil Sreendra, Kushal Patel, Marlon Gwira, Andrea Huber, Nicole Hurley, Francesco Nori, Raia Hadsell, and Nicolas Heess. Learning Agile Soccer Skills for a Bipedal Robot with Deep Reinforcement Learning, April 2023. Guillaume Comment : Project website : https ://sites.google.com/view/op3-soccer.

[13] Chenghao Hu, Sicheng Xie, Liang Gao, Shengyu Lu, and Jingyuan Li. An overview on bipedal gait control methods. *IET Collab Intel Manufact*, 5(3) :e12080, September 2023. Rafael.

[14] Donghyun Kim, Jared Di Carlo, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. Highly Dynamic Quadruped Locomotion via Whole-Body Impulse Control and Model Predictive Control, September 2019.

[15] Yunho Kim, Hyunsik Oh, Jeonghyun Lee, Jinhyeok Choi, Gwanghyeon Ji, Moonkyu Jung, Donghoon Youm, and Jemin Hwangbo. Not Only Rewards But Also Constraints : Applications on Legged Robot Locomotion. *arXiv preprint arXiv :2308.12517*, 2023. Alexandre.

[16] Shunsuke Koseki, Kyo Kutsuzawa, Dai Owaki, and Mitsuhiro Hayashibe. Multimodal bipedal locomotion generation with passive dynamics via deep reinforcement learning. *Frontiers in Neurorobotics*, 16 :1054239, 2023. Rafael.

[17] Cheng-Yu Kuo, Hirofumi Shin, and Takamitsu Matsubara. Reinforcement Learning with Energy-Exchange Dynamics for Spring-loaded Biped Robot Walking. *IEEE Robotics and Automation Letters*, 2023. Alexandre.

[18] Ping-Huan Kuo, Jun Hu, Kuan-Lin Chen, Wei-Hsin Chang, Xin-Yu Chen, and Chiou-Jye Huang. *Advanced Engineering Informatics*, 57 :102067, 2023.

[19] Pierre-Alexandre Leziart, Thomas Flayols, Felix Grimminger, Nicolas Mansard, and Philippe Soueres. Implementation of a Reactive Walking Controller for the New Open-Hardware Quadruped Solo-12. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5007–5013, Xi'an, China, May 2021. IEEE.

[20] Chunguang Li, Mengru Li, and Chongben Tao. A parallel heterogeneous policy deep reinforcement learning algorithm for bipedal walking motion design. *Frontiers in Neurorobotics*, 17, 2023. Maxime.

[21] Zhongyu Li, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, and Koushil Sreenath. Robust and Versatile Bipedal Jumping Control through Reinforcement Learning, May 2023. Guillaume

Comment : Accepted in Robotics : Science and Systems 2023 (RSS 2023). The accompanying video is at https ://youtu.be/aAPSZ2QFB-E.

[22] Zebang Pan, Shan Yin, Guilin Wen, and Zhao Tan. Reinforcement learning control for a three-link biped robot with energy-efficient periodic gaits. *Acta Mechanica Sinica*, 39(2) :522304, 2023. Guillaume.

[23] Daoling Qin, Guoteng Zhang, Zhengguo Zhu, Teng Chen, Weiliang Zhu, Xuewen Rong, Anhuan Xie, and Yibin Li. A Heuristics-Based Reinforcement Learning Method to Control Bipedal Robots. *International Journal of Humanoid Robotics*, 2350013 :22, 2023. Alexandre.

[24] Nikita Rudin. Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning.

[25] Nikita Rudin, David Hoeller, Marco Hutter, and Philipp Reist. Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning.

[26] Nikita Rudin, David Hoeller, Marco Hutter, and Philipp Reist. Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning.

[27] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, August 2017.

[28] Pierre Schumacher, Thomas Geijtenbeek, Vittorio Caggiano, Vikash Kumar, Syn Schmitt, Georg Martius, and Daniel F. B. Haeufle. Natural and Robust Walking using Reinforcement Learning without Demonstrations in High-Dimensional Musculoskeletal Models, September 2023. Lucie.

[29] Rohan P. Singh, Zhaoming Xie, Pierre Gergondet, and Fumio Kanehiro. Learning bipedal walking for humanoids with current feedback. *IEEE Access*, 2023. Alexandre.

[30] José Mauricio Sànchez Soto and Christofer Alex Diaz Arapa. Lower Limb Actuation of a Humanoid Robot for Locomotion Using Reinforcement Learning. In *2023 17th International Conference on Engineering of Modern Electric Systems (EMES)*, pages 1–4. IEEE, 2023. Maxime.

[31] Jing Zhang, Biao Lu, Zhongyan Liu, Yonghui Xie, Wankai Chen, and Weijie Jiang. A Balance Control Method for Wheeled Bipedal Robot Based on Reinforcement Learning. In *2023 42nd Chinese Control Conference (CCC)*, pages 2288–2293. IEEE, 2023. Lucie.

Annexe

Etat de l'Art

L'évolution rapide de la robotique et des technologies d'intelligence artificielle a engendré un intérêt croissant pour le développement de contrôleurs de marche basés sur l'apprentissage par renforcement (RL) pour les robots bipèdes. Cette approche offre la possibilité d'optimiser la locomotion en permettant au robot d'apprendre et de s'adapter à des environnements variés et imprévisibles. Dans ce contexte, notre étude se focalise sur la problématique suivante : à l'heure actuelle, quelles sont les méthodes existantes pour optimiser l'adaptation du mouvement locomoteur d'un robot bipède tout en assurant sa stabilité et son efficacité ?

Les algorithmes d'entraînement de la marche pour les robots bipèdes représentent une composante cruciale dans le développement de systèmes de locomotion robustes et stables. Ces algorithmes visent à enseigner au robot bipède comment coordonner ses mouvements pour assurer une marche efficace et adaptative. Il existe différentes méthodes permettant d'aboutir à la marche d'un robot bipède. La première méthode utilisée a été la commande prédictive par modèle (MPC). C'est une approche de contrôle qui vise à générer des mouvements de marche stables et naturels en ajustant continuellement les entrées de contrôle du robot en fonction des informations sensorielles en temps réel. La MPC fonctionne en prévoyant les mouvements futurs du robot sur un horizon temporel défini. Elle optimise ensuite les commandes de manière itérative pour minimiser une fonction de coût, prenant en compte des objectifs tels que la stabilité, l'efficacité énergétique et la conformité aux contraintes physiques. Aractingi et al [5] utilise cette méthode en se basant sur les approches proposées par DiCarlo et al[9]. ainsi que Kime et al[3]. .La première est basée sur un modèle centroïdal pour planifier la trajectoire de base et les forces de réaction au sol des pieds en contact pour le Mini-cheetah. La seconde est également une formulation de contrôle de l'ensemble du corps qui génère le contrôle de bas niveau nécessaire pour suivre la trajectoire de base sur un horizon temporel plus court. Ces deux approches ont été simplifiées dans le cas du Solo12 afin de pouvoir effectuer ces calculs pour l'entièreté du corps du robot qui est quadrupède. Néanmoins, bien que toutes ces méthodes produisent des contrôleurs dynamiques robustes, elles sont lourdes du point de vue computationnel lors de l'exécution et nécessitent souvent un paramétrage très précis pour la configuration initiale. De nouvelles solutions ont alors été utilisées : l'apprentissage machine, et en particulier l'apprentissage par renforcement. Favorisée par l'augmentation de la capacité de calcul ainsi que le développement du deep learning sur des réseaux de neurones

toujours plus grands, de nombreuses méthodes d'apprentissage par renforcement permettant d'obtenir un contrôleur de marche pour un robot bipède ou quadrupède ont vu le jour.

Un algorithme largement utilisé est DDPG (Deep Deterministic Policy Gradient). C'est un algorithme d'apprentissage par renforcement adapté aux espaces d'action continus comme le contrôle de robots. Il utilise deux réseaux neuronaux, acteur et critique, avec une mémoire de replay pour la stabilité. L'entraînement se fait en minimisant l'erreur entre la valeur prédite par le critique et la récompense réelle, tandis que l'acteur est formé pour maximiser cette valeur. DDPG utilise une mise à jour lente des cibles pour stabiliser l'apprentissage, et combine l'apprentissage de politique et l'évaluation de la valeur pour traiter efficacement des environnements complexes avec des actions continues. Cet algorithme a par exemple été utilisé par Chun et al.[8] et Li et al., 2023[21] et présente des résultats prometteurs.

Bien que le DDPG puisse présenter de bonnes performances, il s'avère souvent fragile en ce qui concerne le réglage des hyperparamètres. Un défi courant avec le DDPG est le risque de surestimation significative des valeurs Q par la fonction Q apprise, entraînant une instabilité de la politique à mesure qu'elle exploite les erreurs dans la fonction Q. Pour résoudre ce problème, le Twin Delayed DDPG (TD3) intègre trois stratégies clés : un apprentissage à double Q limité (utilisation de deux fonctions), une mise à jour de politique "retardées", ainsi qu'un lissage de la politique cible. Dans l'ensemble, ces trois stratégies contribuent à des performances améliorées par rapport au DDPG [8]. Cet algorithme a été testé en simulation par Pan et al[22] sur un modèle basique de robot bipède et a montré de très bon résultats quant à l'efficience énergétique de la démarche du robot. Il n'a cependant pas été testé en condition réelle sur un robot bipède. Un autre algorithme largement utilisé est celui de PPO (Proximal Policy Optimization) présenté par OpenAI[2]. Ce dernier est appliqué pour entraîner la politique de mouvement du robot. Pendant la collecte de données, le robot interagit avec son environnement pour enregistrer des informations sur les états, les actions et les récompenses associées à la qualité de la marche. PPO utilise ensuite une estimation d'avantage pour évaluer la performance des actions prises par le robot pendant la marche, en comparant la récompense réelle avec une estimation basée sur la valeur de l'état actuel. L'objectif de PPO est d'optimiser la politique de marche du robot en ajustant les paramètres de manière à maximiser la récompense cumulative. Les mises à jour de la politique sont effectuées itérativement sur plusieurs cycles d'entraînement, permettant à l'algorithme d'apprendre progressivement des stratégies de marche plus efficaces. En utilisant des mini-lots d'expériences, PPO renforce la stabilité de l'apprentissage et évite des modifications trop brusques dans la politique de marche du

robot. l'équipe de M. Aractingi et al.[5] et D.Qin et al.[23] ont montré à travers leurs travaux l'efficacité de cette méthode, en particulier la stabilité de l'apprentissage qu'elle permet.

Une limite de cet algorithme est la complexité de réglage de la fonction de récompense et l'impossibilité d'expliciter les contraintes pour entraîner le modèle. Cette difficulté impose un fastidieux réglage des fonctions de récompense pour chaque type de robot. Une approche de PPO avec intégration des contraintes est proposée par Kim et al [15] permettant de généraliser plus facilement le pipeline d'apprentissage. Inspirés par les méthodes d'apprentissage par imitation, Daoling et al ont mis en place une méthode d'apprentissage guidée par des heuristiques basées sur des trajectoires de référence. Cette approche implique la définition de trajectoires de référence pour les mouvements des pieds du robot pendant leurs phases de balancement (swing) et d'appui (stance). Les trajectoires de référence sont définies à l'aide de courbes de Bézier [23]. L'application de l'heuristique prend la forme d'un terme de récompense qui encourage le robot à suivre les trajectoires de références. Afin de garantir une meilleure résilience sur des terrains complexes, et éviter que l'apprentissage soit contraint par les trajectoires de références, des fonctions reward d'exploration sont aussi implémentés. Ces récompenses visent à encourager le soulèvement des pieds et à favoriser des mouvements fluides. Bien que les trajectoires de référence ne spécifient aucun mouvement de rotation, cette méthode a produit de bons résultats pour des mouvements omnidirectionnels. Malgré l'absence de directives explicites pour les rotations, l'apprentissage guidé par ces trajectoires, combiné aux autres récompenses ont permis un apprentissage efficace en simulation et de générer une politique fonctionnelle pour le robot bipède BRAVER de 36 cm de haut, et équipé de "point feet". Une technique permettant d'améliorer la stabilité des robots avec des pieds plats consiste à utiliser le point de moment nul (Zero Moment Point - ZMP) [29]. Le point de moment nul est un concept issu de la biomécanique et de la robotique humanoïde. Il représente le point sur le sol où le moment résultant généré par les forces appliquées aux pieds du robot est nul. En d'autres termes, c'est le point où la somme des moments autour de ce point est égale à zéro. L'utilisation du ZMP dans le contexte de l'amélioration de la récompense signifie que l'on cherche à ajuster la politique de contrôle du robot de manière à maintenir ou optimiser la position du ZMP. Cela peut être accompli en ajustant les commandes de mouvement du robot de manière à influencer la répartition du poids sur ses pieds, contribuant ainsi à une stabilité accrue. L'objectif est souvent d'éviter que le ZMP ne sorte de la zone de support définie par les pieds du robot. Si le ZMP sort de cette zone, le robot risque de perdre l'équilibre et de tomber. En maintenant le ZMP à l'intérieur de la zone de support, on cherche à assurer la stabilité du robot.

Les méthodes d'apprentissage renforcé pour la locomotion des robots font généralement appel à des simulations de la physique du robot pour recueillir une grande quantité de données de manière sécurisée. Cependant, ces simulations peuvent comporter des erreurs ou ne pas saisir pleinement la complexité de la dynamique réelle. L'écart entre la simulation et la réalité, souvent appelé "écart sim2real", peut entraîner une performance médiocre lorsque la politique générée est déployée sur le système réel. Une approche courante pour atténuer ce problème consiste à rendre aléatoires certains paramètres tels que la masse, l'inertie, les coefficients de friction ou encore la position des centres de masse [29]. Ces paramètres sont alors sélectionnés dans des intervalles définis avant chaque simulation d'entraînement. De plus, des bruits aléatoires peuvent être ajoutés aux mesures de la simulation, comme la position ou la vitesse des articulations. Un point crucial pour la transition de la simulation au robot réel est l'espace d'action choisi pour le réseau de neurones. Dans [5], Aractingi et al soutiennent qu'il est plus complexe d'apprendre une politique qui contrôle le couple plutôt que la position des articulations. En effet, stabiliser le système à travers les couples, qui est un système d'ordre 2 est plus difficile qu'à partir des positions. De plus, le contrôle en couple peut entraîner des mouvements divergents dans les phases de balancement des jambes en raison de leurs faibles inertie. Pour ces raisons, le contrôle en position est préféré, couplé à un contrôleur PD intrinsèquement stable pour générer les couples envoyés aux actionneurs. Le contrôleur doit également être simulé dans la phase d'entraînement. Cette méthode a pour effet de générer des mouvements souples, puisque le contrôleur PD reproduit un système de ressort amorti sur chacune des articulations.

Il existe différentes approches pour traiter les terrains non-triviaux. Ces différentes approches mettent en évidence les méthodes existantes pour entraîner des contrôleurs de marche sur des terrains variés et ainsi relever les défis posés par différentes conditions environnementales. Une première approche, comme démontré par Helei Duan et al dans [10], utilise l'apprentissage par renforcement (RL) pour entraîner deux composants en utilisant un environnement simulé avec une carte de hauteur exprimée dans le repère local du robot. Le système comprend un prédicteur de carte de hauteur entraîné sur des images et des états du robot, qui alimente la politique de contrôle du robot, elle aussi entraînée. Cette approche permet ainsi une adaptation au terrain basée sur la vision, évitant ainsi le besoin d'une estimation explicite de l'état du robot.

Une autre approche pour l'entraînement de contrôleurs de marche vis-à-vis de terrains variés consiste à exposer progressivement le robot à des terrains difficiles au cours de son apprentissage, ce qui améliore sa capacité à s'adapter aux surfaces inégales. Cette méthode, baptisée "Terrain Curriculum' fait partie

des méthodes classiques de “Curriculum Learning” et est utilisée par Aractingi et al[5]. Dans le même esprit que l’approche précédemment décrite, il existe une démarche, détaillée par Nikita Rudin et al [12], qui consiste à faire évoluer l’agent dans un espace découpé en terrain plus ou moins escarpés avec une probabilité croissante de se retrouver dans un terrain escarpé au fur et à mesure que l’entraînement progresse. L’idée est, de façon similaire à l’article précédent, d’augmenter l’irrégularité du terrain au fur et à mesure que l’entraînement progresse. Cependant, dans le cadre de cet article, l’espace d’entraînement reste le même d’un épisode à l’autre et c’est l’agent qui se déplace en direction d’une partie de l’espace plus escarpée. Par ailleurs, cette méthode explore une configuration d’entraînement massivement parallèle, diminuant ainsi le temps d’entraînement.

Une autre étude, menée par Gaurav Bhardwaj et al [7], s’intéresse à l’entraînement d’un contrôleur de marche sur terrain déformable. L’approche proposée implique un robot bipède, l’algorithme de RL “Deep Deterministic Policy Gradient” (DDPG) et utilise un maillage triangulaire pour modéliser un sol souple. En utilisant le logiciel de simulation PyChrono, qui fournit en environnement “sol déformable” très réaliste, l’étude explique avoir convergé vers une politique de marche robuste à ce type de sol.

Un défi clé de l’apprentissage d’une démarche locomotrice spécifique via l’apprentissage par renforcement (RL) est de communiquer le comportement de la démarche à travers la fonction de récompense. La récompense doit être suffisamment spécifique pour produire la caractéristique de démarche désirée lorsqu’elle est optimisée, mais en même temps, elle doit tenir compte du fait qu’il y a une incertitude sur les détails exacts d’une démarche dans le contexte d’un terrain spécifique et de conditions dynamiques, ou en d’autres termes, ne devrait pas être trop contraignante. [16]

Par conséquent, les définitions des fonctions de récompense pour le but du comportement de démarche sont fondamentales pour la convergence de la politique non seulement vers une tâche spécifique, mais aussi vers l’extension de la politique à des tâches potentiellement modifiées et plus complexes. Ces fonctions peuvent encapsuler des interprétations de haut niveau du comportement souhaité, qui peuvent ne pas bien se généraliser à leur traduction dans l’environnement simulé ou physique.

Cependant, la littérature fournit une pléthore de fonctions de récompense communes qui sont utilisées dans la définition de la plupart des tâches spécifiques à la démarche, qui tentent d’abstraire l’idée d’une démarche optimale en fonctions représentant la "bonne qualité" de la démarche. Bien que ces fonctions puissent varier légèrement dans leur mise en œuvre, elles suivent toutes le

même objectif [25]. Quelques exemples de fonctions de récompense courantes trouvées dans la littérature sont :

— Correspondance de Vitesse : Récompense le robot pour correspondre aux vitesses commandées, s'assurant qu'il se déplace aux vitesses souhaitées.
— Pénalisation du Couple : Pénalise l'utilisation excessive des couples articulaires, favorisant des mouvements économes en énergie.
— Pénalisation de l'Accélération : Décourage les accélérations ou décélérations rapides pour maintenir un mouvement fluide.
— Pénalisation du Changement de Cible Articulaire : Évite des changements fréquents ou significatifs dans les cibles articulaires pour assurer un mouvement fluide.
— Pénalisation des Collisions : Pénalise le robot en cas de collision avec des obstacles, encourageant l'évitement des obstacles.
— Récompense de la Longueur de Pas : Encourage des pas plus longs pour des modèles de marche plus dynamiques et attrayants.
— Pénalisation de l'Atteinte de Contrainte : Peut inclure le maintien de l'équilibre, la pénalisation de la consommation d'énergie et l'assurance de la stabilité articulaire.

Selon la complexité de l'environnement, cependant, en raison de leur simplicité, ces termes peuvent ne pas induire la convergence ou le comportement de démarche désiré par eux-mêmes. Par conséquent, des fonctions plus spécifiques à l'agent ont été proposées dans la littérature, qui font l'hypothèse préalable de la physique et du comportement de l'agent, ainsi que des conditions de l'environnement. [16]

Koseki et al. [16] proposent une fonction de support pour les agents à démarche bipède qui encourage la politique à balancer les jambes et mène à une exploration efficace, tout en introduisant un terme pour encourager la symétrie des mouvements, produisant des mouvements qui ressemblent plus à ceux des humains. Si un motif de démarche spécifique est souhaité (comme la marche, le trot, etc.), une récompense peut être incluse pour encourager le robot à maintenir un cycle de démarche régulier.

Rudin et al. [25] présentent une approche d'apprentissage par curriculum qui récompense les démarches en fonction de leur niveau de difficulté. Le régime d'entraînement comprend un curriculum adaptatif inspiré des jeux qui met au défi le robot avec des tâches de plus en plus difficiles en fonction de ses performances. Cette approche accélère non seulement le processus de formation, mais améliore également la polyvalence et l'adaptabilité du robot.