Apprentissage non-supervisé pour la découverte de propriétés d'objets par découplage entre interaction et interprétation

Maxime Chareyre

Apprentissage non-supervisé pour la découverte de propriétés d'objets par découplage entre interaction et interprétation

Thèse de doctorat de l'Université Clermont Auvergne

Ecole doctorale N° 70, Sciences Pour l'Ingénieur (ED SPI)
Spécialité Électronique et Systèmes

Présenté par

Maxime CHAREYRE

Thèse soutenue à Palaiseau le 18 décembre 2023 devant le jury composé de :

Olivier SIGAUD Professeur, Sorbonne Université	Président du jury
Ouiddad LABBANI-IGBIDA Professeure, Université de Limoges	Examinatrice
David FILLIAT Professeur, ENSTA	Rapporteur
Alain DUTECH Chargé de recherche, Loria	Rapporteur
Jean-Marc BOURINET Professeur, Clermont Auvergne INP - SIGMA Clermont	Co-directeur de thèse
Youcef MEZOUAR Professeur, Clermont Auvergne INP - SIGMA Clermont	Co-directeur de thèse
Julien MORAS Ingénieur de recherche, ONERA	Co-encadrant
Pierre FOURNIER Ingénieur de recherche, ONERA	Co-encadrant

Remerciements

Je tiens à exprimer ma profonde gratitude envers toutes les personnes qui ont contribué à la réalisation de cette thèse.

J'aimerais tout d'abord remercier mes directeurs de thèse, Jean-Marc Bourinet et Youcef Mezouar, tous deux professeurs à Clermont Auvergne INP - SIGMA Clermont qui m'ont fait confiance en acceptant de diriger cette thèse même si ce domaine n'était pas entièrement de leur expertise. Pendant mes études en école d'ingénieurs, ils ont su me transmettre leur passion pour les probabilités statistiques et la robotique. En particulier, je tiens à remercier chaleureusement Jean-Marc qui a assuré un suivi méticuleux tout au long de mes études à SIGMA Clermont et qui m'a accompagné lors de tous mes stages, malgré le fait que je n'aie pas suivi le cursus qu'il m'avait proposé à plusieurs reprises. Sans la confiance et le soutien indéfectible de Jean-Marc, je n'aurais jamais exploré le domaine de l'apprentissage profond ni réalisé ma thèse à l'ONERA.

Je suis extrêmement reconnaissant envers Julien Moras, ingénieur de recherche à ONERA - DTIS et l'un de mes encadrants de cette thèse, qui m'a offert la possibilité de poursuivre en thèse après mon stage de fin d'études au sein de l'ONERA. Bien que je sois initialement novice dans le domaine de l'apprentissage par renforcement, il a eu une confiance totale en moi pour la réalisation du stage et de la thèse, même si nos échanges ont été initialement limités en raison de la pandémie mondiale. L'encadrement de cette thèse a également été assuré par Pierre Fournier, également ingénieur de recherche à l'ONERA - DTIS. Un très grand merci pour son implication sur ce sujet qui, je le sais, lui tenait à cœur. Il m'a beaucoup appris sur le travail de rédaction et a toujours apporté un point de vue externe permettant de présenter ce sujet complexe, suscitant de nombreuses interrogations en raison de son originalité. Un très grand merci à mes deux encadrants qui m'ont soutenu et apporté leur expertise au cours de ces trois années et sans qui je ne serais pas diplômé aujourd'hui. Leurs commentaires

et suggestions ont largement enrichi ces travaux, les rendant plus clairs et solides grâce à leurs contributions.

Merci à mes directeurs et encadrants pour leurs nombreuses relectures méticuleuses sur chacun des chapitres de ce manuscrit et sur l'ensemble des articles publiés.

Je tiens également à exprimer ma gratitude envers les rapporteurs de ma thèse, David Filliat et Alain Dutech, ainsi qu'envers les examinateurs Ouiddad Labbani-Igbida et Olivier Sigaud, qui ont généreusement accepté de consacrer leur temps et leur expertise à l'évaluation de ce travail. J'ai grandement apprécié vos retours constructifs et les discussions que nous avons eues lors de la soutenance. Vos suggestions représentent une suite logique pour ces travaux, et grâce à elles, il est possible d'améliorer la qualité de cette recherche.

Mes sincères remerciements s'adressent à tous les membres de l'unité IVA de l'ONERA-DTIS. L'ensemble des chercheurs et doctorants ont été des atouts inestimables dans le développement de mes réflexions, tout en constituant une source précieuse de soutien moral et émotionnel. Les discussions sur le rugby et la montagne avec Frédéric et Philippe, ainsi que les moments conviviaux partagés avec l'ensemble de l'unité, notamment Alexis, Adrien, Alexandre, Anthelme, Aurélien, Baptiste, Clément, Clara, Dao, David, Élise, Flora, Guy, les Guillaumes, Julien, Javiera, Kevin, Laura, Liam, Lucrezia, Louis, Mathias, Marie-Ange, Pol, Patrick, Pauline, Rémy, Romane, Salome, Swann, Stéphane, Valentin, Yann, ont contribué à enrichir mon expérience. Mention particulière à Adrien Chan Hon Tong, encadrant de mon stage de fin d'études, qui m'a prodigué des conseils et un soutien précieux, surtout dans les moments difficiles. Un grand merci à mon chef d'unité Martial pour sa bonne humeur, son efficacité et sa réactivité face à tout besoin éventuel. Bien sûr, je ne saurais oublier Marius, Nathan, Thomas et Quentin, ce groupe uni depuis le début de la thèse, toujours prêt à s'entraider dans les moments les plus difficiles. Enfin, un immense merci à Guillaume Vaudaux Ruth pour tout le soutien apporté durant cette thèse et le temps consacré à relire avec rigueur le manuscrit.

Un grand merci à mes amis pour tous les bons moments que nous avons partagés durant cette thèse.

Mes derniers remerciements vont bien sûr à ma famille, mes parents, ma copine Ludivine de m'avoir soutenu durant toute cette thèse, mais également tout le long de mes études. Un très grand merci à mes parents de m'avoir permis de réaliser les études que je souhaitais.

À tous ceux qui ont contribué de près ou de loin à cette thèse, je vous adresse mes plus sincères remerciements.

Résumé

Les robots sont de plus en plus utilisés pour réaliser des tâches dans des environnements contrôlés. Leur utilisation en milieu ouvert est cependant encore confrontée à des difficultés. L'agent robotique est en effet susceptible de rencontrer des objets dont il ignore le comportement et la fonction. Dans certains cas, il doit interagir avec ces éléments pour réaliser sa mission en les collectant ou en les déplaçant mais, sans la connaissance de leurs propriétés dynamiques il n'est pas possible de mettre en place une stratégie de résolution de la mission efficace.

Dans cette thèse, nous présentons une méthode visant à apprendre à un robot autonome une stratégie d'interaction physique avec des objets inconnus, sans aucune connaissance a priori, l'objectif étant d'extraire de l'information sur un maximum de propriétés physiques de l'objet à partir des interactions observées par ses capteurs. Les méthodes existantes pour la caractérisation d'objets par interactions physiques ne répondent pas entièrement à ces critères. En effet, les interactions établies ne permettent qu'une représentation implicite de la dynamique des objets, nécessitant une supervision pour identifier leurs propriétés. D'autre part, la solution proposée s'appuie sur des scénarios peu réalistes sans agent. Notre approche se distingue de l'état de l'art en proposant une méthode générique pour l'apprentissage de l'interaction, indépendante de l'objet et de ses propriétés, et pouvant donc être découplée de la phase de leurs prédictions. Cela permet notamment de mener à un pipeline global totalement non-supervisé.

Dans une première phase, nous proposons d'apprendre une stratégie d'interaction avec l'objet via une méthode d'apprentissage par renforcement non-supervisée, en utilisant un signal de motivation intrinsèque qui repose sur l'idée de maximisation des variations d'un vecteur d'état de l'objet. Le but est d'obtenir une série d'interactions contenant des informations fortement corrélées aux propriétés physiques de l'objet.

Cette méthode a été testée sur un robot simulé interagissant par poussée et a permis d'identifier avec précision des propriétés telles que la masse, la forme de l'objet et les frottements.

Dans une seconde phase, nous réalisons l'hypothèse que les vraies propriétés physiques définissent un espace latent explicatif des comportements de l'objet et que cet espace peut être identifié à partir des observations recueillies grâce aux interactions de l'agent. Nous mettons en place une tâche de prédiction auto-supervisée dans laquelle nous adaptons une architecture de l'état de l'art pour construire cet espace latent. Nos simulations confirment que la combinaison du modèle comportemental avec cette architecture permet de faire émerger une représentation des propriétés de l'objet dont les composantes principales s'avèrent fortement corrélées avec les propriétés physiques de l'objet.

Les propriétés des objets étant extraites, l'agent peut les exploiter pour améliorer son efficacité dans des tâches impliquant ces objets. Nous concluons cette étude par une mise en avant du gain de performance de l'agent au travers d'un entraînement via l'apprentissage par renforcement sur une tâche simplifiée de repositionnement d'objet où les propriétés sont parfaitement connues.

L'intégralité du travail effectué en simulation confirme l'efficacité d'une méthode novatrice visant à découvrir en autonomie les propriétés physiques d'un objet au travers d'interactions physiques d'un robot. Les perspectives d'extension de ces travaux concernent le transfert vers un robot réel en milieu encombré.

Abstract

Robots are increasingly used to achieve tasks in controlled environments. However, their use in open environments is still fraught with difficulties. Robotic agents are likely to encounter objects whose behaviour and function they are unaware of. In some cases, it must interact with these elements to carry out its mission by collecting or moving them, but without knowledge of their dynamic properties it is not possible to implement an effective strategy for resolving the mission.

In this thesis, we present a method for teaching an autonomous robot a physical interaction strategy with unknown objects, without any a priori knowledge, the aim being to extract information about as many of the object's physical properties as possible from the interactions observed by its sensors. Existing methods for characterising objects through physical interactions do not fully satisfy these criteria. Indeed, the interactions established only provide an implicit representation of the object's dynamics, requiring supervision to identify their properties. Furthermore, the proposed solution is based on unrealistic scenarios without an agent. Our approach differs from the state of the art by proposing a generic method for learning interaction that is independent of the object and its properties, and can therefore be decoupled from the prediction phase. In particular, this leads to a completely unsupervised global pipeline.

In the first phase, we propose to learn an interaction strategy with the object via an unsupervised reinforcement learning method, using an intrinsic motivation signal based on the idea of maximising variations in a state vector of the object. The aim is to obtain a set of interactions containing information that is highly correlated with the object's physical properties. This method has been tested on a simulated robot interacting by pushing and has enabled properties such as the object's mass, shape and friction to be accurately identified.

In a second phase, we make the assumption that the true physical properties de-

fine a latent space that explains the object's behaviours and that this space can be identified from observations collected through the agent's interactions. We set up a self-supervised prediction task in which we adapt a state-of-the-art architecture to create this latent space. Our simulations confirm that combining the behavioural model with this architecture leads to the emergence of a representation of the object's properties whose principal components are shown to be strongly correlated with the object's physical properties.

Once the properties of the objects have been extracted, the agent can use them to improve its efficiency in tasks involving these objects. We conclude this study by highlighting the performance gains achieved by the agent through training via reinforcement learning on a simplified object repositioning task where the properties are perfectly known.

All the work carried out in simulation confirms the effectiveness of an innovative method aimed at autonomously discovering the physical properties of an object through the physical interactions of a robot. The prospects for extending this work involve transferring it to a real robot in a cluttered environment.

Table des matières

Table des figures

Liste des tableaux

CHAPITRE I

Introduction

Sommaire

I–1 Contexte

Un robot est un système dynamique mêlant des techniques issues des branches de la mécanique, de l'électronique et de l'informatique. C'est une machine capable de résoudre des tâches dans le monde physique. Par l'intermédiaire de capteurs, le robot acquiert des données et les interprète pour décider d'une action à effectuer selon un mode opératoire conçu par un roboticien. Selon sa programmation, le système peut prendre un certain nombre de décisions de façon autonome. De nos jours, ils sont autant utilisés dans l'industrie que pour un usage personnel, dans le but de réaliser des tâches dangereuses ou pénibles.

(a) Handle. [1]

(b) Thermite RS. [2]

(c) Amazon Astro. [3]

FIGURE I.1 – Exemple de trois robots aux utilisations distinctes. (a) : un robot industriel spécialisé dans la manutention. (b) : un robot de sécurité civile conçu pour assister des équipes d'intervention en milieu hostile. (c) : un robot domestique apportant une assistance vocale au quotidien.

L'autonomie d'un tel système dépend de sa capacité à observer, comprendre, agir et s'adapter aux environnements souvent complexes dans lesquels il est amené à évoluer. Dans la majorité des cas industriels, l'ingénieur programme les actions du robot sur la base de fortes connaissances à priori, pour un environnement connu et souvent entièrement contrôlé. En milieu ouvert, comme dans le cas d'exploration d'espaces inconnus ou d'utilisation chez un particulier, ces hypothèses ne sont pas vérifiées à cause du manque de connaissances et de la potentielle présence de perturbations imprévisibles. Dans de telles conditions, la performance du robot dépend de deux points clefs : premièrement, sa capacité à traiter des données pour comprendre la situation, et deuxièmement, sa capacité à décider de l'action optimale à exécuter dans cette situation précise.

Ainsi, pour naviguer et fonctionner efficacement dans divers environnements, l'attention du robot portée aux objets environnants devient un point central et déterminant.

1. https://www.bostondynamics.com/legacy
2. https://www.howeandhowe.com/civil/thermite
3. https://www.amazon.com/Introducing-Amazon-Astro/dp/B078NSDFSB

I–1.1 Raisonner autour des objets

Un environnement est généralement constitué de nombreux objets avec lesquels le robot cherchera à interagir ou qu'il tentera d'éviter s'ils sont considérés comme des obstacles. Dans l'industrie, les objets sont souvent les éléments principaux de la tâche : collecte, manipulation mobile, assistance aux humains dans l'assemblage. Cela nécessite la compréhension de paramètres de l'objet, relatifs à la tâche et à l'environnement. Dans ce type de milieu, ces éléments sont généralement suffisamment bien définis pour permettre une résolution autonome du problème.

Ces environnements sont cependant assez différents des espaces réels, destinés aux humains, pour lesquels la communauté scientifique s'efforce activement de développer des robots autonomes, capables de s'adapter et de résoudre efficacement des demandes. La complexité s'accroît particulièrement dans des scénarios de monde ouvert où de nouveaux objets peuvent apparaître à tout moment et où le robot doit opérer dans un environnement non-structuré et dynamique. Une perspective pourrait être, par exemple, de concevoir une aide aux particuliers ou professionnels pour des tâches de nettoyage, rangement et collecte d'objets (Figure I.2). Que cela soit dans un magasin ou au sein d'un domicile, c'est un enjeu technique significatif, qui combine perception, navigation et compréhension. La prise de décisions amenant à la réussite de la tâche repose sur un savoir indispensable et acquis de multiples façons (modélisation de l'environnement à partir des observations du robot, informations transmises de l'ingénieur au robot, ...). Pour ranger un objet, le robot doit avoir connaissance du lieu et d'un moyen de stockage, mais il doit également savoir détecter puis manipuler l'objet. En généralisant à partir de cet exemple du quotidien, la découverte d'un élément inconnu présente un grand intérêt dans l'exploration autonome, comme l'évolution sur un terrain hostile avec des décombres (identification d'éléments mobiles) ou l'exploration de planètes (manipulation de roches), par exemple.

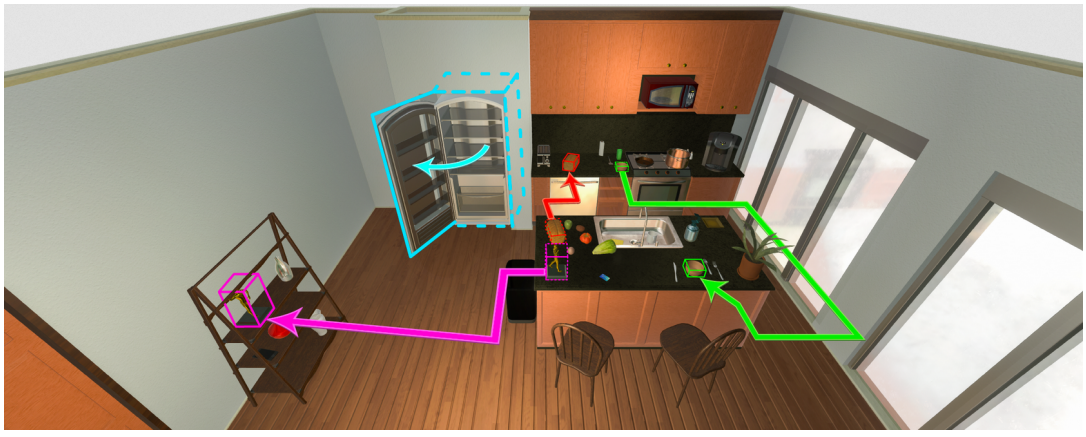


FIGURE I.2 – Scénario de réorganisation d'objets dans un environnement simulé, semblable à celui d'une cuisine. Crédit image : Weihs et al. [2021].

La diversité des formes, textures, propriétés, et modes d'utilisation rendant chaque objet unique, l'enjeu pour le robot est de disposer de connaissances sur chacun d'eux. Établir une base de connaissances contenant toutes les possibilités envisageables ne semble pas être une approche réaliste ou viable.

Naviguer et interagir efficacement avec les objets implique non seulement une compréhension approfondie des environnements et des objets, mais aussi une reconnaissance visuelle précise et un contrôle astucieux du robot. Cela souligne l'importance de la vision robotique et des techniques de contrôle intelligent pour optimiser les interactions du robot dans des milieux variés et dynamiques.

I–1.2 Vision et contrôle pour la robotique

Lorsque le sujet de la compréhension des objets est évoqué, l'apprentissage automatique apparaît comme un incontournable pour leur identification, notamment grâce aux avancées récentes dans le domaine de l'apprentissage profond (ou *Deep Learning - DL*) [Dong et al., 2021, Goodfellow et al., 2018]. Ce type d'apprentissage, initialement appliqué à des jeux de données d'images statiques, permet désormais d'identifier avec une grande précision des personnes, des objets du quotidien, des obstacles, des éléments mobiles, ou encore d'estimer des positions relatives [Arkin et al., 2023]. Les modèles ainsi formés ont démontré une capacité à généraliser à de nouvelles images tout en conservant leur précision.

FIGURE I.3 – Exemple de détection de zones de saisie estimée par apprentissage profond pour faciliter la préhension d'objets à l'aide d'un bras manipulateur. Crédit image : Lenz et al. [2015].

L'état de l'art de Károly et al. [2021] montre que le domaine de la robotique a su exploiter les avancées du domaine de l'apprentissage profond afin d'apprendre à percevoir une scène et à en identifier les objets, en exploitant les signaux issus des capteurs d'un robot, tels que des images issues d'une caméra, mais également des nuages

de points issus d'un LiDAR[4] ou d'une carte de profondeur[5]. Une fois « entraîné », le modèle de perception est généralement suffisamment rapide pour être utilisé en temps réel sur un robot facilitant ainsi la planification et la prise de décision grâce à une meilleure compréhension de la scène. Par exemple, la saisie par un bras articulé, illustrée sur la Figure I.3, peut être améliorée grâce à une estimation plus précise des distances, des positions et de l'orientation des objets [Lenz et al., 2015, Mahler et al., 2018, Yu et al., 2021]. En robotique mobile, le pré-traitement des données des capteurs via des réseaux de neurones se combine avantageusement avec des algorithmes de cartographie et localisation simultanées (SLAM) [Chaplot et al., 2020b, Gao and Zhang, 2017] facilitant ainsi la planification et la navigation vers des objets via des approches de robotique classique.

Bien qu'il soit possible de généraliser les capacités de perception à des environnements inconnus, de nombreux scénarios voient leur complexité décuplée sous l'effet de nombreux facteurs : encombrement, obstacles en mouvement, changements soudains au sein de l'environnement, etc. Le défi pour évoluer dans un environnement est alors de choisir les actions à effectuer tout en gérant une multitude de situations, impossibles à définir à l'avance. L'apprentissage par renforcement (ou *Reinforcement Learning - RL*) [Sutton and Barto, 2018] est l'une des solutions privilégiée pour un contrôle en milieu dynamique. Les algorithmes développés ont démontré tout leur potentiel d'adaptation et de contrôle optimal dans des contextes de jeux de plateaux [Pérolat et al., 2022, Silver et al., 2017] et jeux vidéos [Berner et al., 2019, Mnih et al., 2015, Vinyals et al., 2019], jusqu'à dépasser les performances humaines. Les laboratoires spécialisés en robotique et apprentissage sont aujourd'hui capables d'appliquer ces algorithmes à des fins de locomotion en milieux naturels [Miki et al., 2022].

En RL, l'objectif est d'apprendre à un système dynamique, appelé agent, à effectuer une tâche par une série d'essais et d'erreurs en expérimentant séquentiellement diverses actions au sein de son environnement. Comme illustré sur la Figure I.4, l'agent observe à pas de temps régulier son environnement, agit et reçoit une récompense positive ou négative en conséquence. Il ajuste ensuite sa stratégie de sorte à maximiser la somme des récompenses cumulées sur sa trajectoire. Depuis l'essor du DL, un réseau de neurones est entraîné pour décider du comportement du système à partir d'observations pré-traitées par des modèles de perception comme ceux présentés précédemment.

Les robots concernés par ce type d'approche sont les robots manipulateurs, robots mobiles ou encore les drones, avec des cas d'applications nombreux (Figure I.5) :

- Planification de trajectoire [Chiang et al., 2019, Faust et al., 2018],

4. Capteur laser qui mesure le « temps de vol » d'un faisceau laser infrarouge
5. Représentation de la profondeur de chaque pixel d'une image par des nuances de gris.

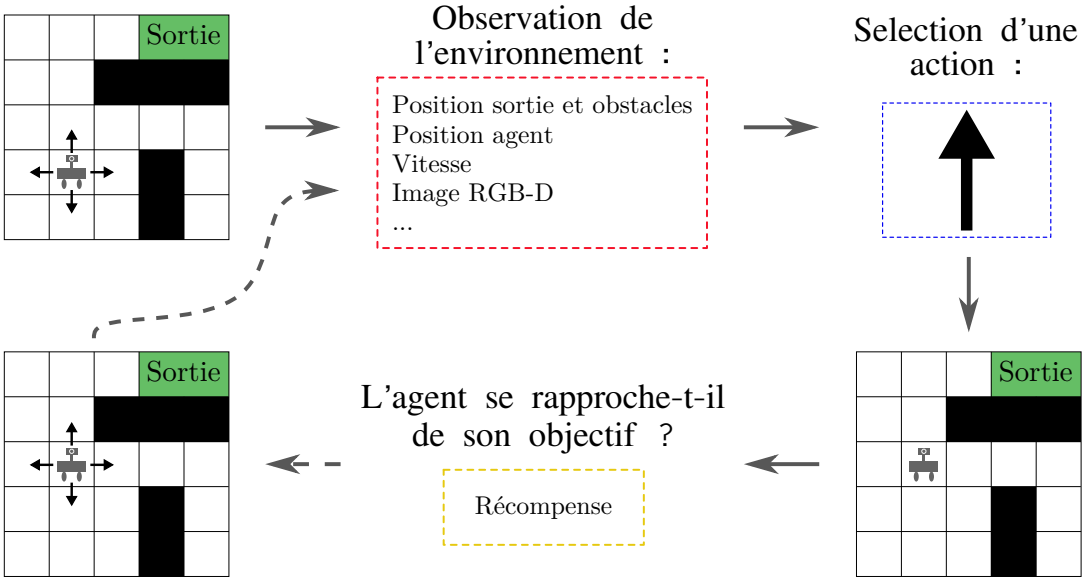
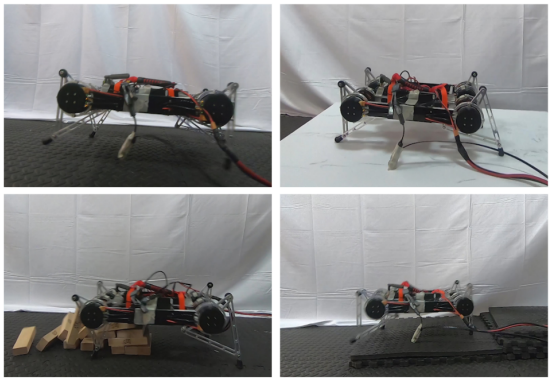


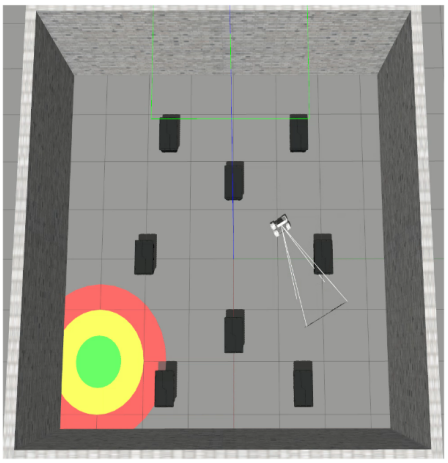
FIGURE I.4 – Illustration du processus itératif entre un agent et son environnement dans le cadre de l'apprentissage par renforcement.

- Navigation et évitement d'obstacles [Chaffre et al., 2020, Tai et al., 2017, Wang et al., 2019],
- Tracking [El-Fakdi and Carreras, 2013, Luo et al., 2020],
- Interactions homme-robot [Khamassi et al., 2018],
- Recherche d'objets ou de points de vue [Chaplot et al., 2020a, Schmid et al., 2019],
- Contrôle bas niveau pour apprendre à voler ou à marcher en terrain complexe [Becker-Ehmck et al., 2020, Haarnoja et al., 2019],

En RL, l'agent construit en temps réel sa propre base de données d'apprentissage, ce qui constitue un atout, comparé aux méthodes d'apprentissage supervisé et non-supervisé qui nécessitent un ensemble de données vaste et préétabli. Cette stratégie signifie cependant que l'agent enrichira sa base de connaissance uniquement avec les informations qu'il est capable d'observer grâce à son comportement actuel. L'agent doit donc généralement effectuer un très grand nombre d'expériences avant d'atteindre un comportement optimal, ce qui ne peut être réalisé directement sur un robot réel. Des simulateurs sont alors utilisés pour entraîner l'agent, dans le but d'accélérer le processus et de réduire les risques. En effet, l'avantage de l'utilisation de la simulation réside dans la capacité qu'elle offre à l'agent de réaliser rapidement un grand nombre d'expériences. Ceci est rendu possible grâce à la vitesse de calcul et à la parallélisation assurées par l'unité centrale de traitement (*Central Processing Unit - CPU*) et les unités de traitement graphique (*Graphics Processing Unit - GPU*), voir Figure I.6a. La diversité des simulateurs a connu une augmentation significative ces dernières années, permet-



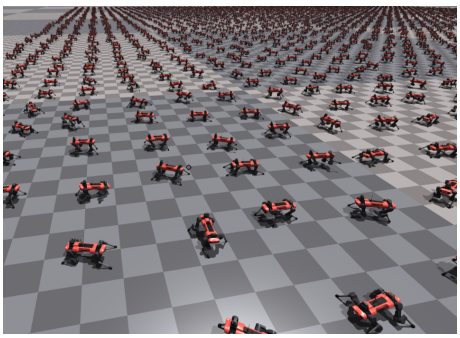
(a) Apprentissage de la marche sur un robot "Minitaur" généralisant à divers types de perturbations terrain. Crédit image : Haarnoja et al. [2019].



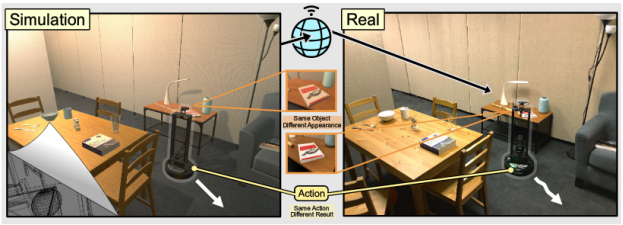
(b) Tâche de navigation vers cible avec évitement d'obstacles. Crédit image : Chaffre et al. [2020].

FIGURE I.5 – Exemples d'application de l'apprentissage par renforcement en robotique.

tant de former un agent sur des tâches aussi variées que complexes, le tout avec un réalisme pouvant se rapprocher du monde physique que nous connaissons. Par conséquent, de nombreuses études se focalisent sur les méthodes de transfert d'un modèle comportemental appris en simulation vers un système réel [Zhao et al., 2020]. Comme la simulation peut être adaptée pour la rendre proche de la réalité elle peut ainsi permettre à des robots d'apprendre à se déplacer sur des terrains accidentés ou encore à naviguer dans des environnements humains (Figure I.6b).



(a) Parallélisation d'agents pour augmenter le nombre d'expériences et accélérer le processus d'apprentissage. Crédit image : Makoviychuk et al. [2021].



(b) Simulateur RoboTHOR [6] dédié à l'apprentissage et transfert vers système réel. Crédit image : Deitke et al. [2020].

FIGURE I.6 – Simulateurs dédiés à l'apprentissage d'un agent robotique.

6. https://ai2thor.allenai.org/robothor

Ainsi, les travaux de Chaplot et al. [2020a,b], Ramakrishnan et al. [2020] montrent qu'en combinant RL et DL, un robot mobile est capable de naviguer en autonomie pour explorer un espace inconnu et cartographier la zone, en détectant puis en conservant la position et le type des objets rencontrés (Figure I.7). Les modèles de perception issus de méthodes d'apprentissage aident à la conception de cartes d'occupation égocentrique[7] élaborées à partir des observations d'une caméra RGB-D, laquelle fournit une image en couleur et une carte de profondeur. Ces représentations spatiales permettent ensuite de réaliser des sous-tâches avec les objets recensés.

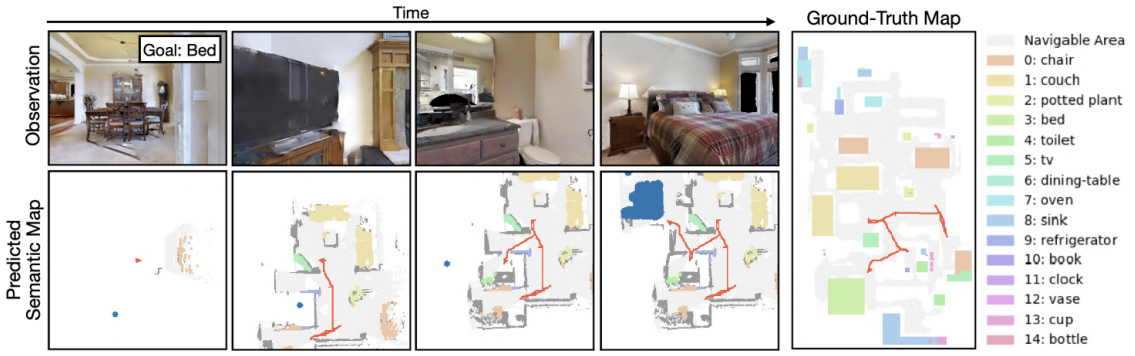


FIGURE I.7 – Exemple de trajectoire effectuée par l'agent pour explorer son environnement et trouver l'objet recherché (ici un lit). Crédit image : Chaplot et al. [2020a].

Alors que la combinaison de l'apprentissage profond et de l'apprentissage par renforcement a montré des avancées significatives dans la navigation et l'interaction robotiques, elle révèle également ses limites liées à une perception essentiellement passive de l'environnement. Face à ces limites, l'approche de la perception active émerge, transformant le robot d'un simple observateur à un acteur proactif, ajustant et enrichissant sa compréhension de l'environnement complexe qui l'entoure.

I–1.3 Perception active de l'environnement

La perception du robot est le point central pour la réussite d'une tâche car elle est non seulement nécessaire à la compréhension de l'environnement, qui peut être inconnu et changeant, mais aussi parce qu'elle oriente sa prise de décision. Toutefois, la quantité d'information collectée est généralement limitée à l'observation passive de l'environnement. Dans ce cas, le robot transmet une observation (ou une série d'observations) à un modèle de perception qui va détecter, classifier et segmenter les objets de la scène. Ensuite, ce flux de données unidirectionnel est transmis au modèle de

7. Carte indiquant la position d'obstacles, des zones libres et des zones non explorées. Dans le cas égocentrique, le repère de représentation est celui de l'agent.

contrôle relié aux actionneurs, pour guider le robot, dans le but d'atteindre son objectif. La perception de l'agent est donc limitée aux données capturées par les capteurs lorsque le robot effectue sa tâche. Il ne choisit pas comment percevoir la scène. Or, l'avantage pour un robot, est justement de pouvoir se mouvoir et/ou manipuler les objets. Ainsi, plutôt que de se restreindre à une observation passive, il peut se déplacer pour optimiser sa perception des éléments, ou des points de vue, qu'il interprète avec difficulté.

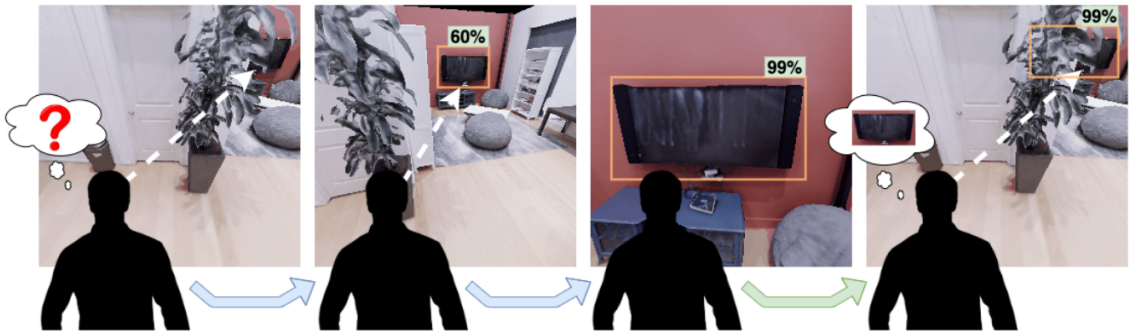


FIGURE I.8 – Amélioration de la compréhension de l'environnement par la perception active. Crédit image : Fang et al. [2021].

C'est dans ce contexte que s'inscrit la notion de perception active. Déjà étudiée à la fin du 20ème siècle [Bajcsy, 1988, Whitehead and Ballard, 1990], le principe est d'intégrer le mouvement de l'agent dans la boucle de perception. Les travaux de Roggeman [2017] visent par exemple à améliorer la capacité de localisation en adaptant la trajectoire de l'agent pour diminuer les incertitudes. Dans la mesure où des éléments d'une scène peuvent être facilement occultés par d'autres objets, en se déplaçant, l'agent s'affranchit des occultations et identifie les objets plus efficacement, comme l'illustre la Figure I.8. Cela reflète le comportement naturel de l'humain lorsqu'il cherche à comprendre et manipuler facilement des objets. Typiquement, en cas de doute sur une saisie, l'humain commencera par observer l'objet sous différents angles afin de choisir le meilleur moyen de le saisir.

Un comportement similaire peut être instauré chez un agent manipulateur afin de lui permettre de discerner davantage de caractéristiques d'un objet et ainsi en améliorer la représentation [Zaky et al., 2020]. Cependant, les systèmes en robotique traditionnelle étant complexes, le roboticien ne peut généralement pas coupler manuellement le module de perception et la boucle de contrôle. L'usage de l'apprentissage par renforcement pour contrôler la perception active peut alors permettre son automatisation. Le problème de contrôle peut être formulé comme un processus de décision markovien : l'agent observe son environnement puis sélectionne une action qui lui permet de se déplacer pour obtenir une nouvelle observation permettant d'affiner sa

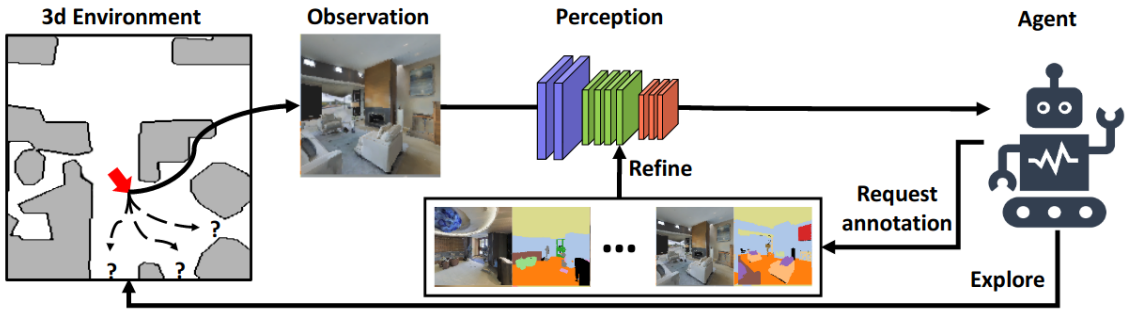


FIGURE I.9 – Amélioration d'un modèle de perception par combinaison de l'apprentissage par renforcement et perception active. Un agent explore, observe puis transmet certaines observations complexes à l'utilisateur pour annotation. Avec l'ajout de ces nouvelles données d'entraînement, la perception est améliorée. Crédit image : Nilsson et al. [2021].

perception de l'environnement. Après chaque action, l'agent est récompensé en fonction de son nouveau niveau de compréhension de l'environnement. Dans la littérature, différents objectifs sont expérimentés pour effectuer de la perception active par RL. La recherche d'objets ou d'emplacements en est un exemple [Han et al., 2019, Schmid et al., 2019, Yang et al., 2019]. Ici, le modèle de détection observe sa cible mais n'est pas confiant sur la correspondance avec son objectif. Les actions réalisées lui permettent alors de se déplacer autour de l'objet pour obtenir de meilleurs points de vue, plus proches, ou lui permettant d'éviter les occlusions visuelles qui rendent la détection impossible. Aussi, certains objets peuvent être d'apparence très similaire et leur différenciation nécessite l'observation de caractéristiques localisées sur une infime partie de leur structure. Une exploration active de l'objet peut ainsi aider à les détecter. L'ensemble des données collectées durant l'exploration active peut également servir à ré-entraîner les modèles de perception afin de les améliorer et de les re-transférer vers les agents (Figure I.9). C'est par exemple le cas dans la méthode mise en place par Fang et al. [2021] qui consiste à améliorer un détecteur d'objets du quotidien, en propageant les prédictions qui sont sûres à d'autres points de vue d'un même objet. Dans ce cas, l'agent apprend à se déplacer activement là où le détecteur échoue ou est incertain [Chaplot et al., 2020c, Nilsson et al., 2021].

Toutefois, par l'utilisation de la perception active, une perspective apparaît, poussant l'agent à non seulement observer, mais également interagir de manière intentionnelle avec son environnement. En effet, lorsque l'interaction physique se combine avec la perception, elle ouvre un champ d'applications plus vaste, dans lequel le robot peut non seulement comprendre, mais également influencer son environnement de manière concrète et constructive.

I–1.4 Agents interactifs

Les approches présentées précédemment utilisent exclusivement les signaux provenant des capteurs de perception du robot, sans exploiter sa capacité à agir physiquement avec l'objet. Néanmoins, l'interaction avec les objets est aujourd'hui au coeur d'une majorité de tâches de robotique, comme le montre la quantité de challenges proposés à "*Embodied AI CVPR workshop*". Listés dans le Tableau I.1, ces challenges proposent de réaliser diverses tâches nécessitant de l'interaction, dont des tâches de ré-arrangement de l'environnement, d'interaction multimodale combinant vision et langage, de navigation avec possibilité de déplacer les objets, ou encore de manipulation avec un bras robotique.

TABLE I.1 – Liste de challenges et des simulateurs associés portant sur l'interaction avec des objets. La plupart considèrent des robots mobiles.

Nom du challenge	Tâche	Simulateur
AI2-THOR Rearrangement	Rearrangement	AI2-THOR
ALFRED	Vision-and-Language Interaction	AI2-THOR
iGibson	Interactive Navigation	iGibson
iGibson	Social Navigation	iGibson
TDW-Transport	Rearrangement	TDW
TEACh	Vision-and-Dialogue Interaction	AI2-THOR
Language Interaction	Instruction Following and Dialogue	AI2-THOR
DialFRED	Vision-and-Dialogue Interaction	AI2-THOR
ManiSkill	Generalized Manipulation	SAPIEN
Habitat Rearrangement	Rearrangement	Habitat
HomeRobot : OVMM [8]	Vision-and-Language Interaction	Habitat

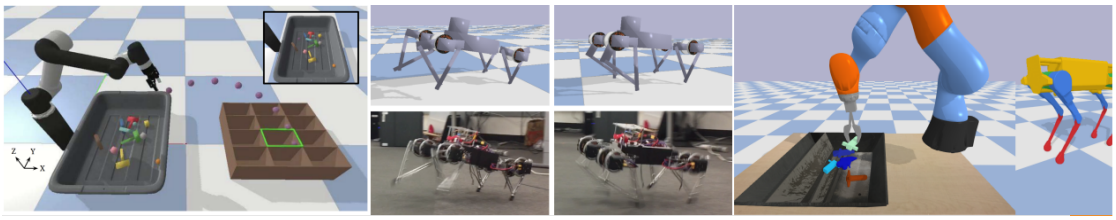
En parallèle, de nombreux simulateurs dédiés à l'apprentissage en environnement réaliste a vu le jour, chacun disposant de sa spécialité. Certains incluent la possibilité de manipuler la physique des objets tandis que d'autres permettent d'évaluer le transfert vers des systèmes réels. La Figure I.10 donne un aperçu de certains des simulateurs les plus classiques. Par exemple, Habitat et iGibson conçus par Li et al. [2022], Manolis Savva* et al. [2019] sont axés sur la navigation en simulation photoréaliste, comme des bureaux ou un domicile. Les recherches correspondantes visent avant tout un fonctionnement sur un robot réel. Les simulateurs AI2Thor et ThreeDWorld se focalisent plutôt sur l'interaction agent-objet et objet-objet sur des outils du quotidien [Deitke et al., 2020, Gan et al., 2021]. Enfin, il existe des simulateurs spécialisés en mécanique des corps rigides et déformables comme Pybullet [Coumans and Bai, 2016–2022], où l'utilisateur peut aisément faire varier les paramètres des objets comme la masse, le frottement, le glissement, etc.

Le développement des simulateurs et l'augmentation du nombre de challenges

(a) iGibson. Scènes interactives et photo-réalistes. Crédit image : Li et al. [2022].

(b) Habitat. Navigation en environnement photo-réaliste. Crédit image : Manolis Savva* et al. [2019].

(c) ThreeDWorld. Une plate-forme multimodale de haute fidélité pour la simulation physique interactive. Crédit image : Gan et al. [2021].



(d) Pybullet. Module Python pour la simulation physique des corps rigides et déformables, la robotique et l'apprentissage par renforcement profond. Crédit image : Coumans and Bai [2016–2022].

FIGURE I.10 – Illustration des simulateurs les plus courants utilisés pour l'apprentissage d'un agent interactif.

montre le besoin de concevoir des robots autonomes capables d'évoluer et d'interagir avec le monde réel. Même si les modèles de perception sont aujourd'hui très performants, la perception seule ne suffit pas à comprendre suffisamment la multitude de situations et d'éléments qui peuvent entourer l'agent. L'interaction physique avec les objets peut palier ce manque et permettre d'acquérir plus d'information sur les objets ou de découvrir ses propriétés non-visibles.

Un exemple, illustré par la Figure I.11, est de considérer la situation où un agent, équipé uniquement de capteurs, doit différencier deux objets visuellement identiques. Le matériau de chaque objet, masqué par une texture (peinture, revêtement, ...), peut

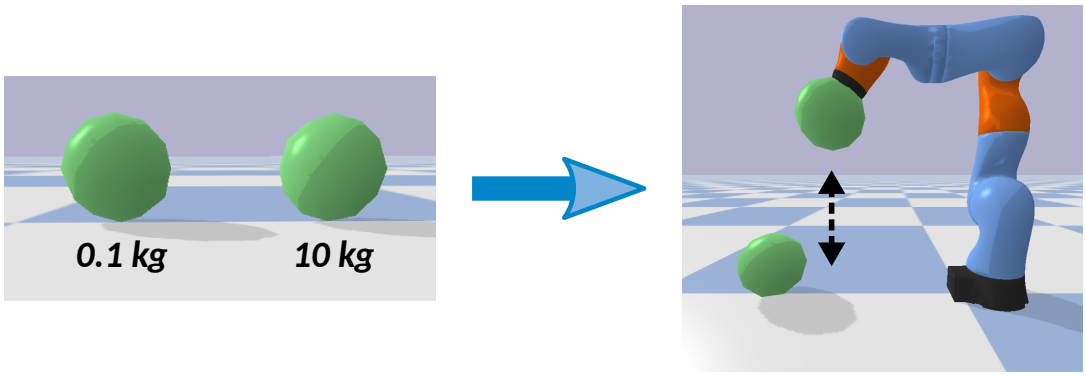


FIGURE I.11 – Exemple d'une propriété cachée pouvant uniquement être révélée par interaction. Les deux sphères sont visuellement identiques, mais leurs masses sont différentes. Elles peuvent être identifiées par la poussée ou une levée à l'aide d'un bras.

être difficilement identifiable et le contenu peut également être variable (plein ou creux par exemple). Ainsi, bien qu'ils soient visuellement identiques, leur masse diffère. Si la tâche de l'agent est de pousser ou d'attraper ces objets, la force nécessaire sera différente pour chacun d'eux, et, sans cette information, l'interaction pourra être dégradée ou même risquée pour l'agent.

Le besoin croissant d'interaction dans le domaine robotique, mis en évidence par les simulateurs et challenges précédents, met en lumière la nécessité d'une exploration plus approfondie des méthodes permettant aux agents d'acquérir des connaissances sur les objets à manipuler. L'exploration active se positionne alors comme un élément central pour approfondir cette compréhension.

I–1.5 Exploration active d'objets

Pour pouvoir interagir avec son environnement de manière complète et efficace, l'agent doit donc disposer d'informations sur les propriétés physiques des objets. Un modèle de perception se révèle toutefois insuffisant pour estimer précisément ces informations, compte tenu de l'immense variété d'objets et de contextes d'interaction existants.

Explorer une approche interactive devient essentielle pour s'approprier les propriétés de l'objet. La stratégie ici n'est pas simplement d'interagir avec l'objet par des moyens prédéfinis mais d'interagir activement avec lui dans le but d'enrichir et d'affiner les informations collectées. À la manière de la perception active, l'objectif est d'élaborer, au fil des interactions, une base de données en continu sur l'objet, révélant ainsi ses attributs cachés et enrichissant l'agent d'une compréhension élargie.

Cette forme d'apprentissage trouve des parallèles dans le comportement d'un nouveau-né qui présente un ensemble de comportements visuo-moteurs lui permettant d'acquérir des connaissances de manière spontanée et en l'absence de signaux

extérieurs [Baillargeon, 2004, Oakes and Baumgartner, 2012]. Les enfants en bas âge mènent des expérimentations autonomes avec les éléments de leur environnement et cherchent activement à générer et à observer des événements à la fois instructifs et stimulants pour satisfaire leur curiosité.

Dans ce contexte, plusieurs recherches récentes s'intéressent à la formalisation mathématique de cette approche d'apprentissage orienté par un comportement automotivé, (ou motivation intrinsèque [Oudeyer and Kaplan, 2007]) conduisant à l'observation de la nouveauté et de la surprise, éléments pertinents pour la robotique [Ivaldi et al., 2014, Rayyes, 2023]. En utilisant un signal de motivation intrinsèque, Ivaldi et al. [2014] orientent l'agent vers l'interaction avec les objets dont la compréhension est la plus complexe. De plus, en utilisant un signal de curiosité, ils incitent l'agent à choisir différentes stratégies d'interaction pour améliorer la compréhension d'un objet.

La direction choisie pour les travaux présentés dans ce document s'inscrit dans cette perspective. L'ambition est de rendre un agent capable de découvrir de manière autonome les propriétés d'un objet et de les formuler de manière explicite, facilitant ainsi la réutilisation de ces informations dans la résolution de tâches ultérieures.

I–2 Objectif de la thèse et démarche

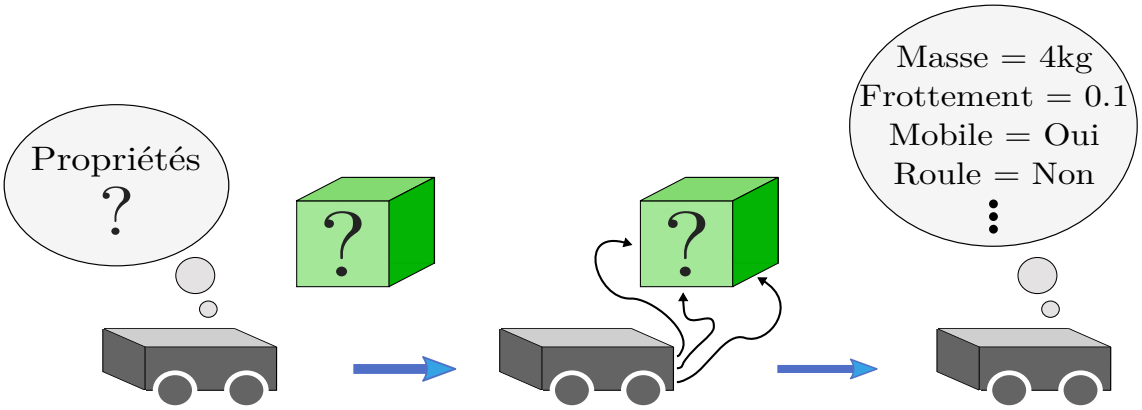


FIGURE I.12 – Illustration de l'objectif fixé durant cette thèse. L'agent fait face à un objet inconnu et interagit avec une série d'actions bien choisies pour estimer ensuite ses propriétés et caractéristiques. Il peut être capable de prédire des propriétés continues, telles que la masse et les frottements, ou d'identifier des caractéristiques telles que "L'objet est-il mobile ?" ou "Peut-il rouler ?".

Lors de son fonctionnement, un robot autonome peut être confronté à des objets qui lui sont inconnus. Dépourvu de connaissances a priori sur ces objets, il doit tout de même les manipuler pour accomplir sa tâche. L'agent peut alors utiliser ses moyens de préhension pour interagir avec eux, traiter les observations résultantes et en extraire

de l'information pertinente. Dans ces conditions, l'enjeu majeur réside dans l'élaboration d'une méthodologie permettant d'apprendre à un robot comment interagir avec un objet totalement inconnu. L'objectif pour lui est d'être capable d'identifier, via de ses propres capteurs et sans connaissances a priori, le plus grand nombre possible de propriétés physiques et dynamiques de l'objet.

L'intérêt principal de collecter un ensemble de propriétés relatives à l'objet est de pouvoir utiliser ultérieurement ces connaissances pour optimiser son efficacité et la faisabilité dans la réalisation de tâches diverses (manipulation, réarrangement, ...). La complexité, qui caractérise également l'originalité de ces travaux, se manifeste non seulement dans la nécessité de découvrir l'objet en totale autonomie, mais aussi d'agir en l'absence de connaissances a priori concernant l'objet lui-même, les propriétés recherchées et l'environnement dans lequel il se trouve. Dans le cadre de cette problématique, les défis majeurs sont :

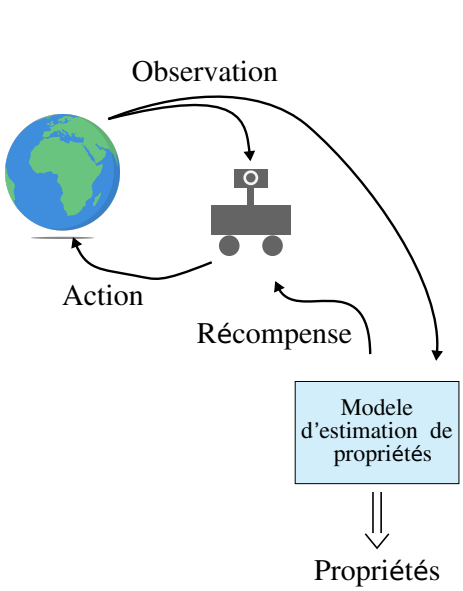
1. D'interagir avec l'objet dans cet environnement inconnu.
2. De découvrir et estimer de manière autonome et sans supervision les propriétés de l'objet, en se basant sur les observations de l'agent.

Concernant l'interaction, diverses méthodes de l'état de l'art, détaillées par la suite mettent en œuvre des actions aléatoires issues d'un ensemble prédéfini [Agrawal et al., 2016, Kandukuri et al., 2022, Li et al., 2018, Xu et al., 2019]. Nous identifions plusieurs faiblesses dans cette approche :

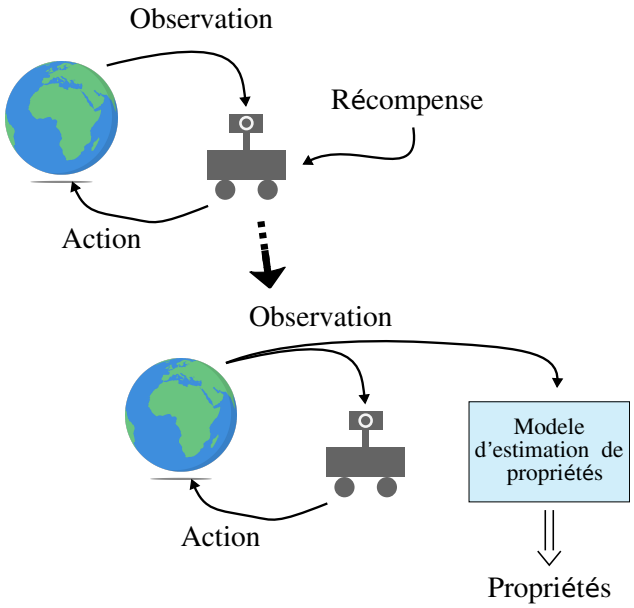
- Dans notre contexte, cela nécessite un algorithme capable de proposer un ensemble d'actions réalisables sur l'objet à l'apparence inconnue, pour pouvoir ensuite réaliser l'une d'elles aléatoirement. La stratégie d'interaction est donc défini à l'avance et ce pour l'ensemble des objets rencontrés. En l'absence de connaissances préalables sur les propriétés cherchées ou sur les objets que l'agent pourrait rencontrer, l'interaction ne pourra pas être optimale pour chacun d'eux.
- Cette stratégie d'interaction manque également de structure. Intuitivement, chaque observation effectuée par l'agent ne peut contenir la même quantité ou qualité d'information pour la compréhension de l'objet. Avec une politique aléatoire, certaines actions pourraient être répétées si l'espace d'actions est discret ou être très similaires si l'espace est continu. De plus, une action particulière pourrait être bien plus descriptive que de nombreuses autres actions et ne jamais être sélectionnée. Cette observation rappelle les principes de la perception active [Fang et al., 2021, Gouko et al., 2015, Zaky et al., 2020] dans laquelle une collecte d'observations plus stratégiques induit de meilleures performances.

Pour palier ces manques, notre objectif est d'élaborer une politique universelle dont les interactions structurées généreront des observations optimales facilitant ainsi

la compréhension de l'objet par un modèle non-supervisé à déterminer. Afin de développer cette stratégie d'interaction, nous avons choisi d'exploiter l'apprentissage par renforcement, dont l'utilisation a démontré des résultats prometteurs dans le domaine de la perception active en robotique.



(a) Le comportement de l'agent est appris grâce à un signal de récompense provenant du modèle d'estimation de propriétés.



(b) L'agent s'auto-récompense afin d'apprendre son comportement. Une fois l'apprentissage terminé, il interagit à nouveau pour générer les observations qui seront exploitées par le modèle d'estimation de propriétés.

FIGURE I.13 – Deux stratégies d'apprentissage d'un comportement, associant perception et interaction, visant à estimer les propriétés des objets.

Une stratégie est employée par Kannabiran et al. [2019], comme illustré sur la Figure I.13a, dans le cadre d'une problématique d'estimation des masses d'un objet articulé. Cette stratégie consiste à apprendre simultanément un prédicteur de propriétés et la politique de l'agent, en le récompensant en fonction l'erreur de prédiction induite. Ainsi, l'agent est incité en un minimum d'interactions à sélectionner les actions qui mènent à la plus grande précision d'estimation. Cependant, la méthode ne répond pas à la problématique posée puisqu'elle nécessite impérativement de disposer des vérités terrain pour entraîner de façon supervisée le modèle de prédiction. L'utilisation d'un jeu de données de vérité terrain induit cependant plusieurs limitations dans notre contexte :

- Tout d'abord, cela oblige à déterminer à l'avance quelles propriétés doivent être recherchées alors que nous souhaitons en découvrir un maximum sans nécessairement connaître leur nature au préalable. Par ailleurs, selon les objets, les propriétés les plus caractéristiques peuvent varier, soulignant l'importance d'adap-

ter dynamiquement les propriétés à estimer en fonction du contexte et de l'objet lui-même.

- La politique d'interaction et le prédicteur de caractéristiques risquent d'autre part de ne pas se généraliser efficacement à de nouveaux objets, dans des contextes jamais explorés auparavant, et pourraient donc même entrer en conflit.
- Enfin, cette approche nécessite d'entraîner simultanément l'apprentissage du module de prédiction et la politique. Au cours de cet apprentissage, l'erreur de prédiction peut résulter soit d'une mauvaise estimation des caractéristiques, soit d'une politique sous-optimale, ce qui peut entraîner une instabilité de l'apprentissage et potentiellement empêcher la convergence.

La majorité des méthodes essaient de résoudre la problématique de caractérisation d'un objet en couplant la stratégie d'interaction au modèle d'estimation des propriétés de l'objet. En alternative, nous proposons d'étudier l'impact d'un découplage de l'apprentissage de ces modèles en deux phases distinctes, comme illustré sur la Figure I.13b. Cette approche s'inspire d'une pratique nouvelle en apprentissage par renforcement, où l'apprentissage d'un agent est séparé en deux phases distinctes : l'une dédiée à l'exploration de l'environnement, et l'autre à la spécification de la politique d'action à la tâche. Cette approche est reconnue pour son potentiel à améliorer les performances et la vitesse d'apprentissage de l'agent dans la tâche finale [Colas et al., 2018] en utilisant un signal de motivation intrinsèque pour guider la phase initiale d'exploration [Laskin et al., 2021, Oudeyer and Kaplan, 2007].

Dans notre cas, la stratégie que nous mettons en place place consiste à apprendre tout d'abord à interagir efficacement avec les objets, puis à identifier les propriétés de l'objet en exploitant les observations découlant de la politique précédemment établie. Concernant la seconde phase, notre ambition est de développer une architecture auto-supervisée inspirée de l'état de l'art [Janner et al., 2019, Xu et al., 2019, Zheng et al., 2018]. L'objectif est de construire un espace de représentation dans lequel les propriétés de l'objet sont encodées. Il est essentiel de s'assurer que la structuration de cette espace est véritablement liée aux propriétés de l'objet. Le défi suivant est d'élaborer une méthode permettant à l'agent d'interagir avec l'objet sans en avoir de connaissances a priori. Sans information préalable, guider l'agent par un signal de récompense pour lui indiquer les actions pertinentes est une tâche ardue. Néanmoins, en l'absence d'un signal de récompense explicite [Oudeyer and Kaplan, 2007], les agents peuvent explorer leur environnement en étant guidés par des motivations intrinsèques dites de curiosité, nouveautés, incertitudes, etc. Nous proposons de focaliser l'apprentissage de l'agent sur des interactions avec l'objet permettant de le mettre en mouvement et d'en observer une large diversité. L'intuition est qu'en diversifiant les interactions et par conséquent les mouvements engendrés de l'objet, cela permet d'identifier de nou-

velles caractéristiques ou d'affiner les connaissances déjà acquises sur celui-ci. Cette approche vise finalement à utiliser ces propriétés identifiées pour réussir une tâche de manipulation secondaire qui, sans le succès des étapes précédentes, serait soit irréalisable soit sous-optimale.

I–3 Plan du manuscrit

Cette thèse propose de répondre à cette problématique au travers de 6 chapitres.

Après ce premier chapitre de mise en contexte, le Chapitre II rappelle des notions d'apprentissage profond et d'apprentissage par renforcement nécessaires à la compréhension des méthodes existantes et de la méthode que nous avons mise en place pour résoudre la problématique. Le Chapitre III présente ensuite les travaux existant sur l'acquisition de connaissances sur les objets et notamment des propriétés physiques. Nous montrons que les méthodes actuelles ne permettent pas de répondre en totalité aux objectifs fixés.

La méthode de résolution proposée dans la thèse se décompose en 3 phases : apprendre une stratégie d'interaction avec les objets produisant des observations permettant de distinguer leurs propriétés, utiliser cette stratégie pour identifier et évaluer ces propriétés sans supervision et enfin utiliser ces connaissances dans la résolution de sous-tâches.

- Dans une première phase présentée dans le Chapitre IV, nous allons chercher à apprendre à un agent à interagir avec un objet sans connaissance a priori sur celui-ci. Pour cela, nous faisons l'hypothèse qu'une stratégie possible est de motiver l'agent à produire de la diversité dans les mouvements de l'objet. De cette manière, nous pensons obtenir un ensemble d'acquisitions contenant des régularités corrélées à des propriétés. Ainsi, nous proposons une solution de contrôle basée sur l'apprentissage par renforcement où l'agent se récompense lui-même grâce à un signal de diversité traduit mathématiquement par une mesure d'entropie. Cette stratégie comportementale est ensuite évaluée par sa capacité à fournir des observations permettant de prédire des propriétés de l'objet par une approche supervisée.
- Dans le Chapitre V, nous cherchons à identifier les propriétés physiques de l'objet de manière non-supervisée à partir du comportement interactif développé dans le Chapitre IV. Nous souhaitons montrer que notre stratégie comportementale rend les propriétés identifiables et évaluables à partir d'un espace de représentation latent de l'objet obtenu par un modèle appris en auto-supervision. Nous présentons ainsi plusieurs modèles, d'architectures diverses, adaptés à la mé-

thode proposée pour faire émerger depuis un espace réduit les propriétés d'un objet.

- Le Chapitre VI est consacré à l'utilisation des connaissances obtenues, grâce à la combinaison du Chapitre IV et Chapitre V, pour la résolution d'une tâche de manipulation. Nous cherchons à savoir si l'ajout de ces propriétés dans l'espace d'observation de l'agent permet d'améliorer ses performances à la suite d'un apprentissage. Ainsi, deux agents sont comparés sur la capacité et la rapidité à résoudre une tâche de poussée vers une zone cible, l'un qui n'a pas de connaissances sur l'objet et l'autre qui l'a préalablement manipulé et représenté.

Le Chapitre VII conclut sur les contributions apportées par la méthode proposée et expose les limites et perspectives de ce travail.

I–4 Contributions

Ce travail, réalisé en trois ans, a permis les avancées suivantes :

— **La constitution d'une première étape d'un cycle d'action-perception orienté objet pour les robots autonomes.** Un robot qui se déplace et explore seul peut se retrouver dans une pièce en présence de plusieurs objets. Sans guide externe, il interagit avec, les comprends puis conserve les connaissances acquises pour de nouvelles tâches.

— **Un dispositif d'apprentissage autonome pour identifier des propriétés physiques inconnues, par la combinaison de l'apprentissage par renforcement non-supervisé et d'une tâche de prédiction auto-supervisée.** À ce jour, nous n'avons pas trouvé d'études qui permettent à un agent d'interagir puis d'estimer les propriétés d'objets sans supervision. L'agent utilise le concept de la motivation intrinsèque pour apprendre à interagir puis utilise les observations collectées pour construire seul une représentation d'un objet.

— **Une mise en pratique en simulation sur un système physique réaliste.** L'ensemble de la méthode est validée sur un robot mobile qui interagit avec des objets de formes simples et de propriétés physiques variables. À partir d'observations non-contraignantes pour un transfert en réel, l'agent est capable d'établir une représentation physiquement intuitive de propriétés continues et catégorielles. Nous constatons que les variables de cette représentation sont fortement corrélées avec les valeurs réelles des propriétés.

— **Une tentative d'intégration des propriétés acquises pour une résolution plus efficace de sous-tâches.** Nous avons imaginé une tâche de manipulation qui met en pratique les acquis de la boucle action-perception et qui nous permet d'étudier le transfert des connaissances acquises. D'apparence simple, cette tâche met en avant les nombreuses difficultés liées à l'exploration de l'environnement et au transfert d'informations.

I–5 Publications et communications

Les recherches réalisées ont donné lieu à plusieurs publications et communications scientifiques.

Publications publiées dans des conférences internationales avec comité de lecture ou dans une revue international :

— **M. Chareyre**, P. Fournier, J. Moras , Y. Mezouar et J.-M. Bourinet, *Unsupervised discovery and estimation of objects physical properties through maximum entropy reinforcement learning*, Pattern Recognition Letters, 2023. (Soumis)

— **M. Chareyre**, P. Fournier, J. Moras , Y. Mezouar et J.-M. Bourinet, *Towards generic object property estimation using unsupervised reinforcement learning*, Dans IROS 2022 Workshop on Mobile Manipulation and Embodied Intelligence (MOMA) : Challenges and Opportunities, Kyoto, Japan, October 23-27 (2022).

Présentation orale :

— **M. Chareyre**, P. Fournier, J. Moras , Y. Mezouar et J.-M. Bourinet, *Towards generic object property estimation using unsupervised reinforcement learning*, Dans IROS 2022 Workshop on Mobile Manipulation and Embodied Intelligence (MOMA) : Challenges and Opportunities, Kyoto, Japan, October 23-27 (2022).

CHAPITRE II

Notions d'apprentissage machine

Sommaire

Ce chapitre introduit les concepts de base des techniques d'apprentissage sur lesquels reposent l'ensemble des travaux de ce manuscrit. Notre méthode utilise différentes architectures et algorithmes de l'état de l'art provenant tant de l'apprentissage profond que de l'apprentissage par renforcement. Une première section présente l'apprentissage profond par réseaux de neurones. Une seconde introduit ensuite les bases et algorithmes classiques de l'apprentissage par renforcement.

II–1 Apprentissage profond

L'apprentissage automatique (ou *Machine Learning - ML*) est un domaine qui fascine mathématiciens et informaticiens depuis le milieu du 20^{eme} siècle. Le ML vise à définir des algorithmes capables d'apprendre à réaliser une tâche spécifique, à partir d'un ensemble d'exemples ou d'expériences. Avec cet ensemble de données, la tâche consiste à prédire une ou plusieurs quantités d'intérêt, ou encore, à déceler des régularités dans les exemples, le tout sans fournir explicitement les règles qui régissent le problème. Le programme doit apprendre par lui même ces règles pour pouvoir à posteriori réaliser cette tâche sur de nouvelles données, en général non vues durant l'apprentissage. Définissons par $\mathbf{x} \in \mathcal{X}$, le vecteur de caractéristiques à partir duquel l'algorithme doit déduire une valeur cible $y \in \mathcal{Y}$ aussi appelée étiquette ou vérité terrain. Une collection d'exemples $\{(\mathbf{x}_i, y_i), 1 \leq i \leq N\}$ forme un ensemble de données D (ou *Dataset*).

Ces algorithmes sont catégorisés selon le mode d'apprentissage effectué. Les classes de problèmes souvent rencontrés sont :

— **L'apprentissage supervisé** : le but est d'apprendre à réaliser une prédiction $\hat{y}$ en un point x du domaine d'étude à partir de $D = (\mathbf{x}_i, y_i), i = 1, ..., N$. Les vérités terrain sont utilisées pour établir un signal de supervision destiné à adapter le modèle d'apprentissage en comparant $\hat{y}$ à y pour le forcer à prédire la bonne valeur. La sortie de ce modèle peut prendre différentes formes : discret, continu, binaire, catégorique, chaînes de caractères, ...

— **L'apprentissage non-supervisé** : ici l'étiquette $y \in \mathcal{Y}$ n'est pas connue. L'objectif est d'apprendre une représentation pour déceler des propriétés de l'ensemble $\mathcal{X}$. Les utilisations standards ont pour but de permettre l'identification de groupes (par exemple via l'algorithme K-means) ou de réduire la dimension de $\mathbf{x} \in \mathcal{X}$ pour condenser l'information importante dans un nouvel espace.

— **L'apprentissage par renforcement** : cette méthode n'utilise pas d'ensemble D. A la place, un agent collecte en temps réel des états $\mathbf{s} \in \mathcal{S}$ en interagissant avec son environnement. Il reçoit des récompenses $r \in \mathcal{R}$ en retour de ses actions

$\mathbf{a} \in \mathcal{A}$ qui l'amènent vers un nouvel état $\mathbf{s}' \in \mathcal{S}$. Un modèle comportemental est appris à partir de multiples séquences (s, a, r, s').

Aujourd'hui, les réseaux de neurones profonds représentent la technique la plus répandue pour réaliser de tels apprentissages. Ceci est dû aux performances atteintes par les méthodes d'apprentissage profond (ou *Deep Learning - DL*) [Goodfellow et al., 2016, LeCun et al., 2015] qui consistent à assembler un grand nombre de couches de neurones pour résoudre des tâches de plus en plus complexes et dont l'espace $\mathcal{X}$ est de très grande dimension. Souvent, le réseau est un enchaînement de différent types de couches de neurones présentant chacun leurs spécificités : couches linéaires, couches convolutives, couches récurrentes, couches d'attention, ... Nous allons, dans la suite de cette section, détailler la méthode d'apprentissage de ces couches ainsi que le fonctionnement de celles utilisées dans cette thèse.

II–1.1 Perceptron multicouche

Rosenblatt [1958] a introduit le neurone, aussi appelé perceptron, utilisé aujourd'hui comme brique de base dans tous les réseaux de l'état de l'art. Le neurone est dans ce cas une fonction, qui à partir d'un vecteur d'entrée $\mathbf{x} \in \mathbb{R}^n$, retourne une prédiction $\hat{\mathbf{y}} \in \mathbb{R}^m$. Cette fonction est une somme pondérée des entrées sur laquelle une fonction d'activation non linéaire est appliquée (voir Figure II.1) :

$$g(\mathbf{x}) = \hat{y} = f(b + \mathbf{W}^{\mathbf{T}} \cdot \mathbf{x}) \tag{II.1}$$

où $\mathbf{x} = (x_1, x_2, ..., x_n) \in \mathbb{R}^n$ est le vecteur d'entrée du neurone, $\hat{y} \in \mathbb{R}$ sa sortie, $\mathbf{W} = (w_1, w_2, ..., w_n) \in \mathbb{R}^n$ et $b \in \mathbb{R}$ respectivement les poids et le biais qui sont les paramètres optimisés durant l'apprentissage, et $f : \mathbb{R} \to \mathbb{R}$ une fonction d'activation choisie.

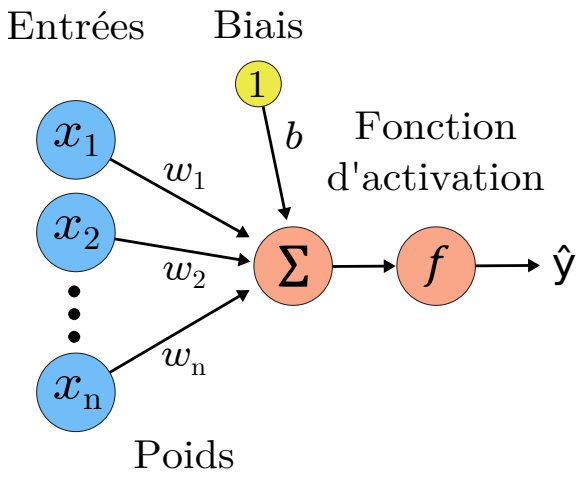


FIGURE II.1 – Illustration d'un perceptron.

Un seul neurone n'est pas suffisant pour approcher n'importe quelle fonction. Pour augmenter la capacité du modèle à approcher des relations entrées-sorties complexes, plusieurs neurones sont assemblés pour former une couche. Pour un même vecteur d'entrée, il y a autant de sorties $g(\mathbf{x})$ que de neurones dans la couche. Chaque liaison entre les entrée $\mathbf{x}$ et les perceptrons possède un biais et des poids indépendants. Le perceptron multicouche [1] (ou *Multilayer Perceptron - MLP*) représenté sur la Figure II.2 est alors défini comme une juxtaposition de k couches de perceptrons dont la prédiction finale $\hat{\mathbf{y}} \in \mathbb{R}^m$ est donnée à l'Eq. (II.2). Les $k-1$ premières couches du réseau sont appelées couches cachées. g_k est appelée couche de sortie. Chaque couche cachée est entièrement connectée avec la suivante et la précédente ce qui signifie que chaque neurone d'une couche est connecté à chaque neurone de la couche précédente et de la couche suivante.

$$\hat{\mathbf{y}} = g_k(g_{k-1}(...g_1(\mathbf{x}))) \tag{II.2}$$

où k défini le nombre de couches et par conséquent la profondeur du réseau.

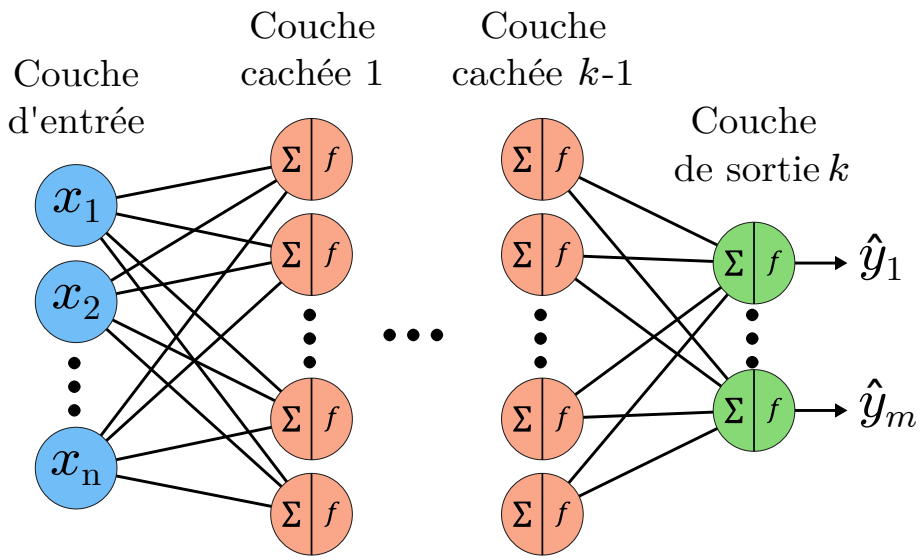


FIGURE II.2 – Le perceptron multicouche. Il est ici composé de $k-1$ couches cachées et de m sorties $\hat{\mathbf{y}} = (\hat{y}_1, \hat{y}_2, \hat{y}_m)$. Chaque liaison dispose d'un poids unique. Les biais ne sont pas représentés.

Le nombre total de paramètres $\boldsymbol{\theta} = (\mathbf{W}, b)$ (i.e. biais et poids) d'une telle architecture est donné par l'Eq. (II.3). Plus il y a de paramètres dans le réseau, plus l'apprentissage sera long et la solution optimale difficile à trouver.

$$N_{\boldsymbol{\theta}\text{MLP}} = \sum_{i=0}^{k-1} (n_i^{in} + 1) n_{i+1}^{out} \tag{II.3}$$

1. Le perceptron multicouche peut aussi être appelé réseau dense ou réseau entièrement connecté.

où i est le numéro de la couche, n_i^{in} la taille du vecteur de la couche entrante i, n_{i+1}^{out} le nombre de neurones de la couche sortante $i+1$.

Les fonctions d'activation f jouent un rôle primordial dans un réseau de neurones. Elles ajoutent de la non linéarité au modèle permettant des représentations plus complexes. En général, le type de fonction d'activation des couches cachées est différent de celui de la couche de sortie. La fonction ReLU (pour *Rectified Linear Unit*), ainsi que les propositions d'amélioration GeLU (*Gaussian Error Linear Unit*), ELU (*Exponential Linear Unit*), Leaky ReLU sont privilégiées pour les couches cachées. La fonction utilisée en couche de sortie dépend du type du problème : fonction Softmax pour une classification multi-classes, la fonction sigmoid pour une classification binaire, linéaire pour une régression.

II–1.2 Optimisation des paramètres

L'entraînement des paramètres $\boldsymbol{\theta}$ nécessite une mesure indiquant l'erreur entre la prédiction du réseau $\hat{\mathbf{y}}$ et la prédiction souhaitée $\mathbf{y}$. Cette mesure s'effectue à l'aide d'une fonction de coût J (ou *loss function*). Que ce soit dans un cas supervisé ou non-supervisé, l'ensemble des éléments de l'ensemble de données D est introduit (*forward pass*) dans le réseau par paquets, aussi appelés *batch*. Sur ce paquet est calculé l'erreur de prédiction à l'aide de la fonction de coût. Ici encore, cette fonction varie selon le problème traité mais les 2 fonctions les plus utilisées sont :

— L'erreur quadratique moyenne (ou *Mean Square Error - MSE*) utilisée pour les tâches de régression :

$$MSE = \frac{1}{N_{batch}} \sum_{i}^{N_{batch}} (\hat{\mathbf{y}}^i - \mathbf{y}^i)^2 \tag{II.4}$$

où N_{batch} est le nombre d'éléments du batch.

— L'erreur d'entropie croisée (ou *Cross-Entropy loss*) utilisée pour les tâches de classification :

$$CE = -\frac{1}{N_{batch}} \sum_{i}^{N_{batch}} \sum_{c=1}^{M} y_{x,c}^i \ln(p_{x,c}^i) \tag{II.5}$$

où N_{batch} est le nombre d'éléments du batch, M est le nombre total de classes, $y_{\mathbf{x},c}$ un indicateur binaire valant 1 si la véritable classe associé au vecteur $\mathbf{x}$ est c et 0 sinon, $p_{\mathbf{x},c}$ la probabilité du vecteur $\mathbf{x}$ d'appartenir à la classe c donnée par le réseau via la fonction d'activation softmax.

Lors de l'entraînement, l'objectif est de trouver la combinaison de paramètres $\boldsymbol{\theta}$ qui minimise la fonction de coût. Les algorithmes d'optimisation utilisés pour résoudre ce

problème sont principalement basés sur la méthode de descente de gradient stochastique. Les paramètres sont alors mis à jour de la manière suivante :

$$\boldsymbol{\theta}' = \boldsymbol{\theta} - \alpha \nabla J(\hat{y}, y; \boldsymbol{\theta}) \tag{II.6}$$

où α représente le taux d'apprentissage (ou *learning rate*) qui fixe l'amplitude des modifications effectuées sur les paramètres.

Le calcul de $\nabla J(\hat{y}, y; \boldsymbol{\theta})$ s'effectue à l'aide de la règle de dérivation en chaîne afin de simplifier la complexité du calcul par des dérivations plus simples et moins consommatrices en temps. Le fait de renvoyer l'information de la fonction de coût aux paramètres du réseau est appelé rétro-propagation du gradient (ou *backpropagation*). Nous appellerons époque (ou *epoch*), un cycle correspondant au passage unique de l'ensemble des exemples $(\mathbf{x_i}, y_i)$ de la base d'apprentissage. Lors de l'apprentissage, est fixé un nombre d'époques suffisamment grand pour permettre d'atteindre une solution proche des paramètres $\boldsymbol{\theta}$ optimaux.

II–1.3 Réseaux convolutifs

En robotique, comme dans de nombreux autres domaines, les observations $\mathbf{x}$ à traiter sont constituées de vecteurs de mesures diverses mais également d'images de grande dimensions (par exemple images RGB et cartes de profondeur en robotique). Pour traiter de tels vecteurs, la quantité de poids d'un MLP devient alors rapidement incontrôlable ce qui rend l'apprentissage quasi-impossible. De plus dans un MLP, chaque entrée du vecteur $\mathbf{x}$ est évaluée de manière indépendante sans prendre en compte les pixels des alentours alors qu'un ensemble de pixels peut fournir une information importante sur les caractéristiques d'un élément visuel recherché.

Pour pallier à l'ensemble de ces problèmes, LeCun and Bengio [1995] ont introduit les réseaux convolutifs (ou *Convolutional Neural Network - CNN*). Une convolution, illustrée sur la Figure II.3, consiste en l'application d'un ensemble de filtres qui parcoure l'image en partant des pixels placés en haut à gauche de l'image jusqu'à atteindre les pixels en bas à droite. Ce filtre, appelé également noyau de convolution (ou *kernel*), est en 3-dimension (largeur, longueur et profondeur) évaluant une portion de l'image (ou patch), pour obtenir une valeur issue du produit scalaire entre les pixels du patch et les paramètres du filtre donnée par l'Eq. (II.7). Ce filtre contient les paramètres entraînables et il est donc appris sur les données. En parcourant l'image, le filtre constitue une nouvelle image, appelée carte de caractéristiques (ou *features maps*), de plus petite dimension où l'information contenue dans cette carte est associée à une caractéristique de l'image (formes verticales, formes horizontales, bruits, ...). En ajoutant plusieurs filtres en parallèle, plusieurs cartes caractéristiques sont ob-

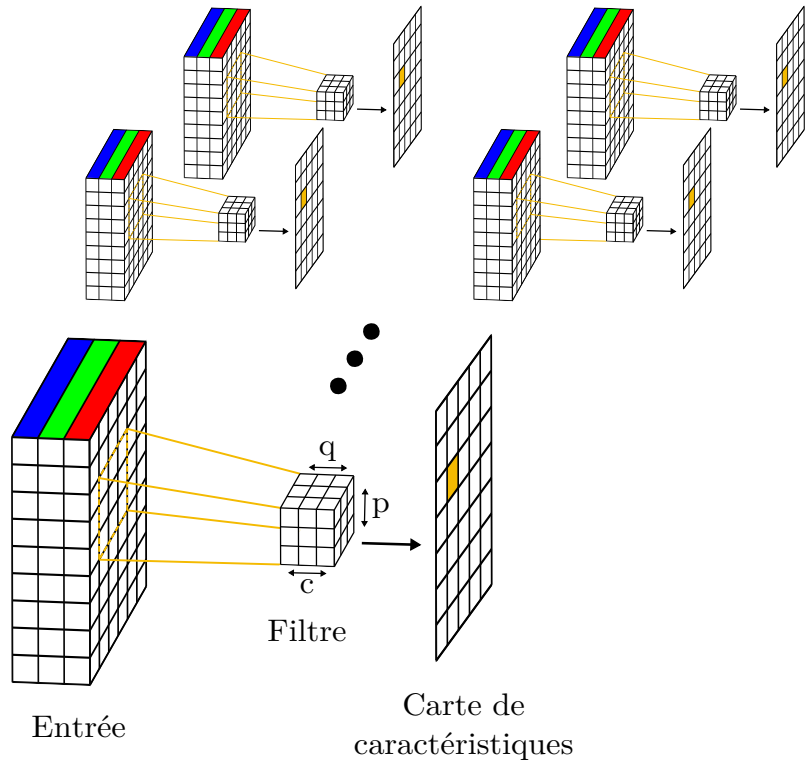


FIGURE II.3 – Fonctionnement d'une convolution sur une image d'entrée à trois canaux (RGB). Dans une même couche, plusieurs filtres sont utilisés en parallèle sur la même entrée.

tenues. Ces cartes ont pour objectif de retenir des informations différentes, cependant rien n'empêche que 2 filtres soient identiques.

$$I_{out} = B + \sum_{k=1}^{C} K_k I_{in,k} \tag{II.7}$$

où I_{in} est l'ensemble des pixels du patch, I_{out} est le résultat de la convolution, C est la profondeur de l'image, $B \in \mathbb{R}$ le biais du filtre, $K \in \mathbb{R}^{c \times p \times q}$ le noyau de convolution de taille $p \times q \times c$ (longueur, largeur, profondeur). K et B correspondent aux paramètres $\boldsymbol{\theta}$ à optimiser.

La manière dont le filtre se déplace sur l'image peut être ajustée par le *stride* qui indique le nombre de pixels par lesquels le filtre se déplace après chaque opération de convolution. Le stride ainsi que les propriétés du noyau jouent sur la dimension de l'image de sortie. Cependant le nombre de paramètres à entraîner, pour une couche de convolution, ne dépend que du nombre de noyaux et de sa taille selon l'équation :

$$N_{CNN} = (qpc + 1)M \tag{II.8}$$

où q est la largeur du filtre, p sa hauteur et c sa profondeur. M correspond au nombre de filtres mis en parallèle afin de créer M cartes caractéristiques. Notons qu'il n'y a

qu'un seul biais par filtre.

En enchaînant plusieurs couches de convolutions, l'information de l'image d'entrée est condensée dans un espace de bien plus petite dimension tout en limitant le nombre de paramètres à entraîner. Une architecture classique (voir la Figure II.4) consiste à ajouter un MLP en sortie des convolutions pour effectuer la tâche de régression ou de classification. Tout comme pour le MLP, des couches d'activation (ReLU, GeLU, ...) sont ajoutées entre chaque produit de convolution. De plus, pour réduire un peu plus la dimension de l'image et forcer le réseau à apprendre des caractéristiques de haut niveau sémantique plus condensées, sont utilisées des couches dites de *pooling*. Ces couches consistent en un filtre qui parcourt une image en conservant uniquement une moyenne des valeurs du patch ou la valeur maximale de ce patch. Le filtre de ce type de couche doit être de faible dimension afin d'éviter la perte d'un trop grand nombre d'informations.

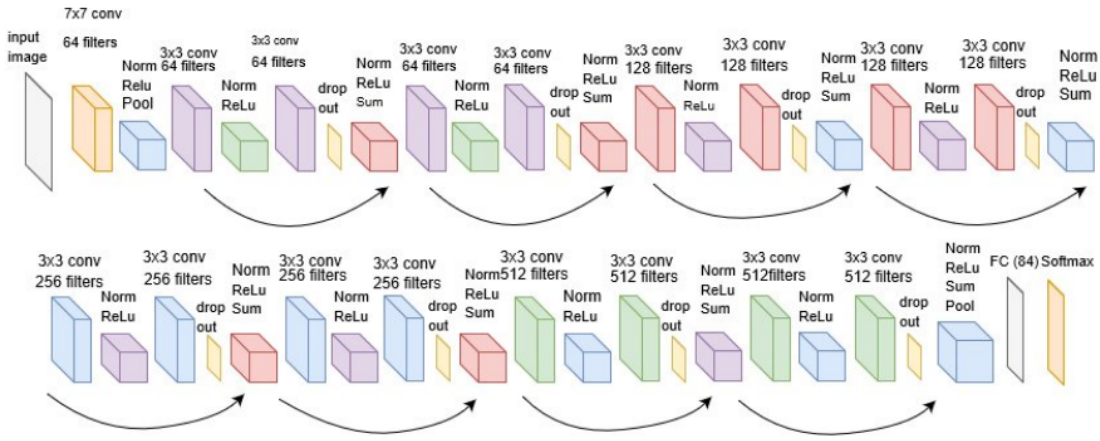


FIGURE II.4 – Représentation d'une version modifiée de l'architecture ResNet-18 utilisée pour la classification d'images de la base de données MNIST. Crédit image :[Al Rabbani Alif et al., 2017]

Les architectures modernes ne se contentent plus d'utiliser seulement des couches convolutives et des MLP. Au fur et à mesure des avancées, de nouvelles cellules ont vu le jour pour répondre à certaines problématiques ou améliorer les relations entre les données durant l'apprentissage. Les deux prochaines sous-sections introduisent brièvement les réseaux récurrents ainsi que le mécanisme d'attention.

II–1.4 Réseaux récurrents

Les réseaux présentés jusqu'ici sont particulièrement adaptés pour le traitement de données statiques, qu'elles soient des images ou de simples scalaires, en considérant un instant fixe. L'éventuelle distribution temporelle des données n'est pas prise en

compte. La prise de décision en robotique s'effectue à pas de temps régulier à partir des observations acquises. Or, pour prendre la meilleure décision, il peut être important de prendre en compte l'aspect dynamique de la scène et du robot.

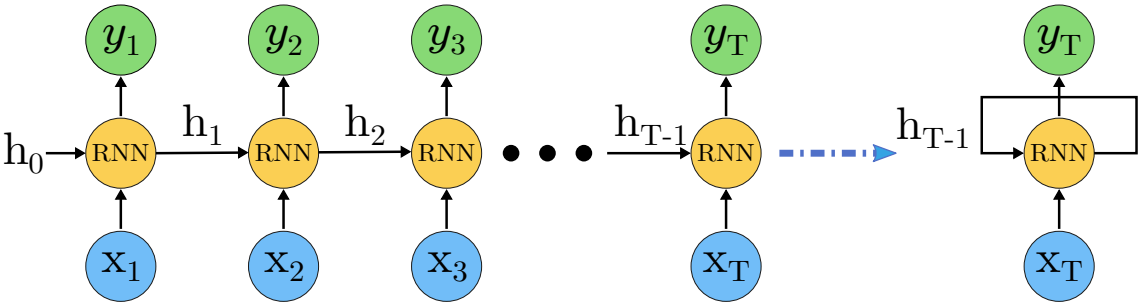
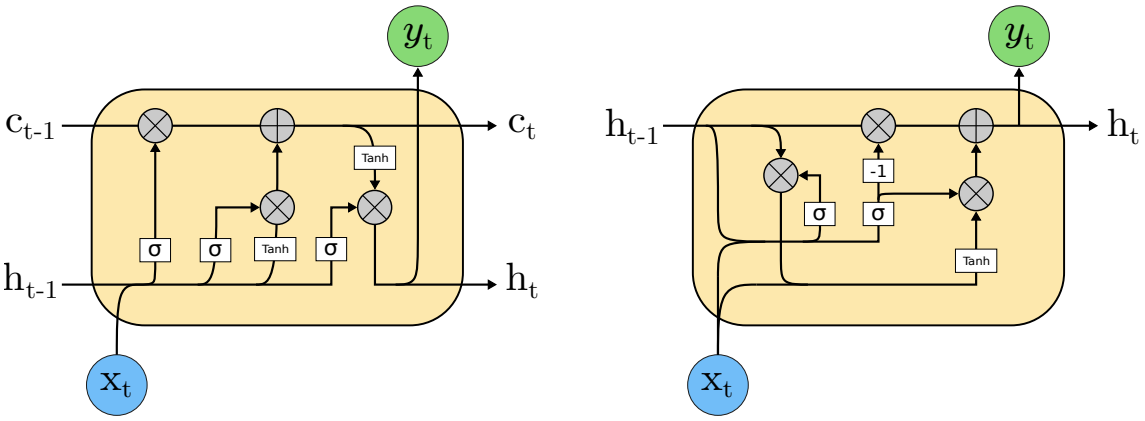


FIGURE II.5 – Illustration d'un réseau récurrent. Les données $\mathbf{x}_t, t = 1, 2, ..., T$ sont introduites séquentiellement dans le RNN en conservant les poids fixes durant tout le processus.

Pour traiter une série temporelle, il est possible d'empiler une série de mesures (images ou scalaires) et de les traiter avec une architecture standard. Seulement, cela demande de fixer la longueur de cette série. Les réseaux récurrents (ou *Recurrent Neural Network - RNN*) regroupent un ensemble d'architectures pour traiter spécifiquement cette problématique. Le principe de l'architecture la plus standard est illustré sur la Figure II.5. Une observation $\mathbf{x}_t$ effectuée à l'instant t est introduit dans un réseau récurrent qui fournit une sortie y_t à partir d'une combinaison de l'observation $\mathbf{x}_t$ avec un état caché $\mathbf{h}_t$, qui à pour objectif de représenter l'ensemble de la séquence de données jusqu'à l'instant t. Cette "mémoire" $\mathbf{h}$ est construite itérativement en parallèle des prédictions $\hat{y}$ effectuées. La cellule récurrente optimise ses paramètres pour apprendre à conserver les informations essentielles des séquences dans h_t et à oublier le reste tout en fournissant une prédiction $\hat{y}_t$ qui tient compte de la dynamique de la série. Plusieurs modes de fonctionnement sont possible : prédire une série de données à partir d'une seule entrée (*one to many*), prédire une sortie à partir d'une série d'entrées (*many to one*), prédire une série de données à partir d'une série d'entrées (*many to many*), ...

Les cellules récurrentes les plus populaires sont le *LSTM* [Hochreiter and Schmidhuber, 1997] et le *GRU* [Cho et al., 2014], toutes deux représentées sur la Figure II.6. Dans les travaux de cette thèse, nous utiliserons la cellule GRU. Ce choix est fait car, le GRU est une variante simplifiée du LSTM, moins longue à entraîner puisque il possède moins de paramètres et comporte une structure plus légère. La cellule comporte deux portes : la porte d'actualisation ($\mathbf{Z}_t$) et la porte de réinitialisation ($\mathbf{R}_t$). La porte d'actualisation décide des informations de la nouvelle observation à ajouter à la mémoire et de celles à jeter. La porte de réinitialisation décide de la quantité d'information de la mémoire à jeter. Pour une observation $\mathbf{x}_t$ introduite dans la cellule GRU, les fonctions suivantes sont calculées :



(a) Cellule LSTM représentée à l'instant t. **(b)** Cellule GRU représentée à l'instant t.

FIGURE II.6 – Architectures des cellules récurrentes LSTM et GRU.

$$\mathbf{r}_t = \sigma(\mathbf{W}_{ir}\mathbf{x}_t + b_{ir} + \mathbf{W}_{hr}\mathbf{h}_{t-1} + b_{hr}) \tag{II.9}$$

$$\mathbf{z}_t = \sigma(\mathbf{W}_{iz}\mathbf{x}_t + b_{iz} + \mathbf{W}_{hz}\mathbf{h}_{t-1} + b_{hz}) \tag{II.10}$$

$$\mathbf{n}_t = \tanh(\mathbf{W}_{in}\mathbf{x}_t + b_{in}) + \mathbf{r}_t \odot (\mathbf{W}_{hn}\mathbf{h}_{t-1} + b_{hn}) \tag{II.11}$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{n}_t + \mathbf{z}_t \odot \mathbf{h}_{t-1} \tag{II.12}$$

où $\boldsymbol{\theta} = (\mathbf{W}_*, b_*)$ sont les paramètres à optimiser, σ est la fonction sigmoïde, et $\odot$ est le produit de Hadamard (produit terme à terme).

Le nombre de paramètres de la cellule GRU est donné par l'équation suivante :

$$N_{GRU} = 3(n^2 + nm + 2n) \tag{II.13}$$

où n correspond à la dimension de l'état caché $\mathbf{h}_{t-1}$ et m à la dimension de l'entrée $\mathbf{x}_t$.

L'apprentissage d'un GRU n'est pas trivial (problèmes de convergence) et le gradient calculé pour mettre à jour les paramètres peut facilement exploser (*exploding gradients*) ou disparaître (*vanishing gradients*). Il est conseillé de limiter le nombre de paramètres à entraîner dans la cellule en limitant la dimension de la mémoire mais aussi de la donnée d'entrée. En présence d'images, un réseau CNN est d'abord utilisé pour réduire l'information avant de l'introduire dans un réseau récurrent.

II–1.5 Mécanisme d'attention

Les RNN sont une solution adaptée à des problèmes de traitement de données séquentielles comme la traduction de texte et la création de modèles conversationnels. Seulement, la performance des cellules comme les GRU et LSTM décroit fortement sur de très longue séquences tel qu'un texte. Pour traiter de très longues séquences, les

architectures modernes intègrent un mécanisme d'attention introduit par Bahdanau et al. [2014].

Le principe est de se concentrer sur les parties les plus importantes d'une séquence d'entrée en attribuant des poids à chaque élément de la séquence dans un contexte donné. Dans la même idée que les portes d'actualisation et de réinitialisation d'un GRU, une cellule d'attention cherche à ignorer certaines entrées et à se focaliser sur d'autres. Le concept d'attention est utilisé pour créer le *transformer*, un modèle combinant attention et diverses couches neuronales pour être utilisé et parallélisé dans des architectures de l'état de l'art sur des tâches linguistiques. Les modèles conversationnels récents tel que ChatGPT [Schulman et al., 2022], ou les modèles de langage développé par Google [Thoppilan et al., 2022] utilisent le transformer comme brique dans leurs architectures.

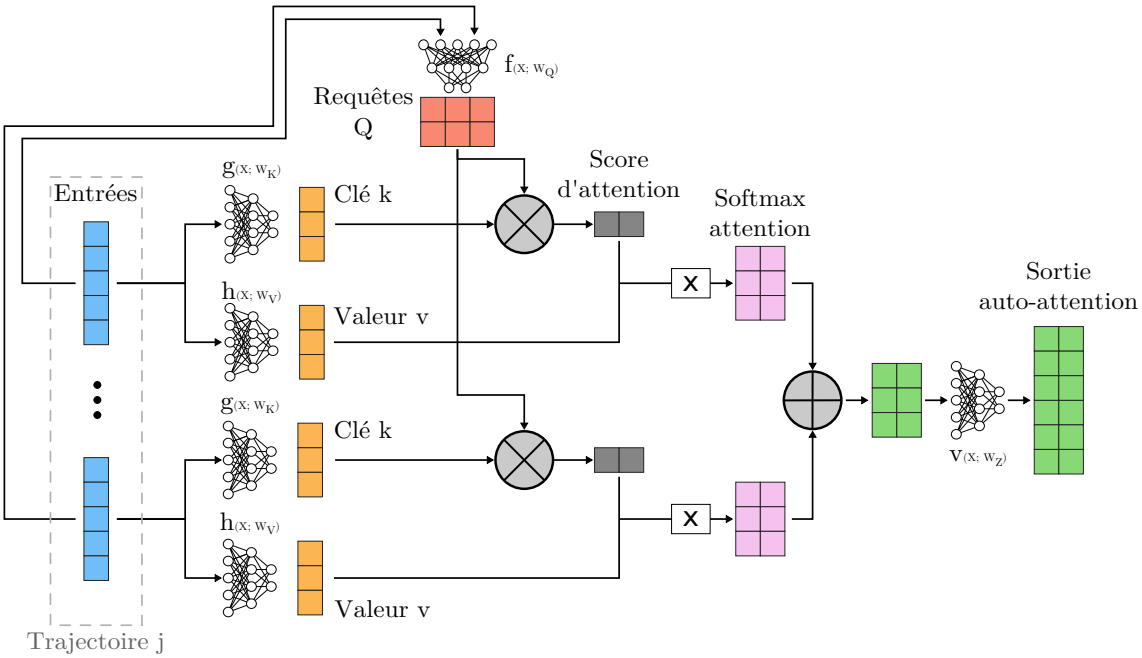


FIGURE II.7 – Le mécanisme d'attention appliqué à une série j d'observations. Dans ce cas, la sortie est l'auto-attention car les requêtes q sont issues de chaque observation de cette série j.

Le mécanisme d'attention, illustré sur la Figure II.7, utilise trois composantes appelées clé $\mathbf{K}$ (*key*), valeur $\mathbf{V}$ (*value*), requête $\mathbf{Q}$ (*query*). Au passage d'une séquence de données, les composantes vont former des matrices qui vont être utilisées pour calculer un score d'attention donné par l'équation :

$$\mathbf{Z} = \text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V} \tag{II.14}$$

où d_k correspond à la dimension de $\mathbf{K}$ ou $\mathbf{Q}$ (identique).

La requête $\mathbf{Q}$ représente le contexte permettant de sélectionner les valeurs $\mathbf{V}$ les

plus pertinentes. Les clés **Q** sont identiques aux valeurs **V**. Le processus effectue le calcul suivant : chaque vecteur de contexte **q** est multiplié aux vecteurs clés **k** pour attribuer un score aux éléments de **k** indiquant les plus reliés au contexte. L'ensemble des scores de la séquence somme à 1 grâce à l'utilisation de la fonction softmax pour calculer le score. Ces scores sont ensuite multipliés aux valeurs **v** pour produire la matrice de sortie. Le mécanisme d'attention n'a pas de paramètre à "apprendre". Par contre la sortie **Z** ainsi que les entrées **Q**, **K**, **V** sont préalablement introduits dans un MLP et sont donc associées aux paramètres $\boldsymbol{\theta}$ de ce MLP. La cellule d'attention telle que définie ici est alors un processus dans lequel l'opération suivante est réalisée :

$$\text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = (\text{Softmax}(\frac{\mathbf{Q}\mathbf{W}_Q\mathbf{K}^T\mathbf{W}_K}{\sqrt{d_k}})\mathbf{V}\mathbf{W}_V)\mathbf{W}_Z \tag{II.15}$$

où $(\mathbf{W}_Q, \mathbf{W}_V, \mathbf{W}_K, \mathbf{W}_Z)$ sont les paramètres d'apprentissage.

Il est appelé auto-attention le cas où **Q**, **K** et **V** sont identiques. Cela signifie que chaque élément de la séquence est évalué par rapport aux autres. En général, plusieurs cellules d'attention sont utilisées simultanément afin d'apprendre des relations différentes à partir de projections **Q**, **K**, **V** différentes, provenant de MLP mis en parallèles. Dans nos travaux, nous utilisons le mécanisme d'auto-attention pour encoder des séquences d'observations en utilisant une unique cellule.

II–1.6 Auto-encodeurs

Dans cette thèse, nous évoquerons régulièrement le terme *autoencoders* tant dans des méthodes de l'état de l'art que pour détailler le fonctionnement de notre méthode. Introduit par Rumelhart et al. [1986], c'est un type d'architecture régulièrement utilisé en apprentissage non-supervisé pour des tâches d'apprentissage de représentation. Pour être entraîné, ces architectures ne nécessitent pas d'étiquettes. A partir d'une observation **x** du set D, un modèle essaie de reconstruire cette même observation ou une autre observation du set. C'est un apprentissage auto-supervisé.

Le principe de cette architecture est de compresser les entrées en un espace de plus petite dimension, appelée représentation latente, puis d'apprendre à reconstruire l'entrée depuis cette représentation (Figure II.8). Il y a donc 3 composantes : un encodeur, un décodeur et l'espace latent. L'encodeur réduit la dimension de l'observation initial et renvoi l'espace latent. Le décodeur, à l'inverse, augmente la dimension de l'espace latent pour reconstruire l'observation initiale. En général, l'encodeur et le décodeur sont modélisés par des réseaux de neurones. Ces réseaux peuvent évidemment combiner les différentes cellules que nous avons présentées précédemment (MLP, CNN, RNN, Transformers,...). L'entraînement des paramètres $\boldsymbol{\theta}$ du décodeur et de l'encodeur nécessite de calculer un signal d'erreur de reconstruction qui est généralement

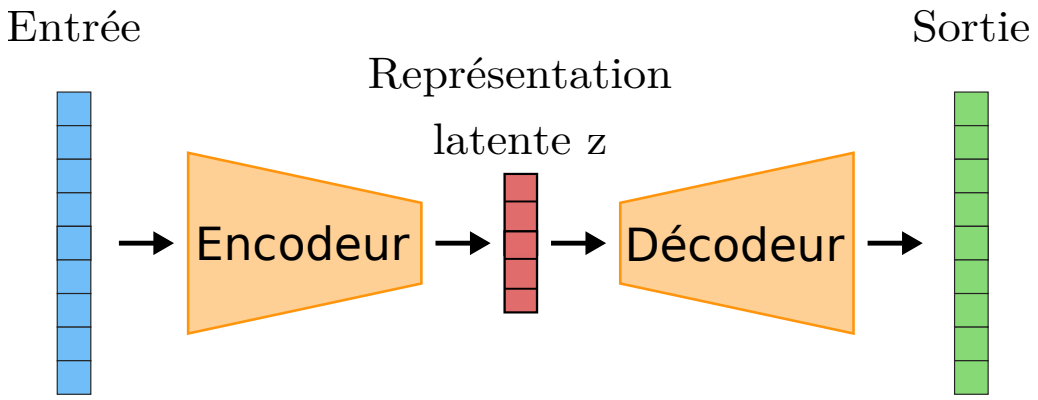


FIGURE II.8 – Utilisation d'un auto-encodeur pour modéliser une entrée dans un espace de représentation latent **z** de plus petite dimension. Cette représentation **z** sert d'entrée au décodeur chargé de reproduire une observation de l'ensemble de données.

l'erreur MSE. Les auto-encodeurs sont aujourd'hui utilisés pour de nombreuses tâches : réduction de dimension, compression d'images, débruitage, modèles génératifs, ...

II–2 Apprentissage par renforcement

L'apprentissage par renforcement [Sutton and Barto, 2018] (ou *Reinforcement Learning - RL*) est un type d'apprentissage dédié au contrôle de systèmes dynamiques qui, comme nous l'avons dit précédemment, n'impose pas la contrainte de disposer d'une base de données. Le RL utilise l'expérience acquise durant un processus d'essais et d'erreurs pour entraîner un agent à choisir une séquence d'actions qui permet de résoudre une tâche de manière optimale. Lors de ses interactions avec l'environnement, l'agent va recevoir une série de récompenses positives ou négatives qui va lui permettre d'adapter son comportement afin de cumuler davantage de récompenses sur ses prochaines interactions. L'apprentissage par renforcement profond (ou *Deep Reinforcement Learning*) qui combine l'apprentissage par renforcement avec l'apprentissage profond est devenu particulièrement intéressant pour résoudre des tâches avec un robot dans des environnements complexes et incertains.

Dans cette section, nous allons introduire les concepts et notations de base avant de présenter les principes des algorithmes de l'état de l'art dont certains sont utilisés pour nos expériences. Tout comme le DL, le RL est un domaine en constante évolution présentant de multiples branches de recherches et de nombreuses techniques pour faciliter la résolution d'un problème. Nous introduisons ici uniquement les notions importantes à la compréhension du déroulement d'un apprentissage. Nous reviendrons dans une dernière section sur l'apprentissage par renforcement non-supervisé (ou *Unsupervised Reinforcement Learning - URL*) et plus particulièrement sur la motivation intrinsèque. C'est un moyen pour l'agent d'apprendre de lui-même son comportement,

sans passer par une intervention humaine lui indiquant les bonnes et mauvaises actions. L'URL est une brique clé de notre méthode.

II–2.1 Processus de décision markovien

En apprentissage par renforcement, un problème doit se formaliser sous la forme d'un processus de décision markovien (ou *Markov Decision Process - MDP*), illustré sur la Figure II.9. Dans un MDP, une interaction se produit à chaque pas de temps indexé via $t \in \mathbb{N}$ entre un système décisionnel, appelé agent, et l'environnement dans lequel il évolue. A chaque instant t, l'agent reçoit une représentation de l'environnement appelé état $\mathbf{s}_t \in \mathcal{S}$ à partir de laquelle il sélectionne une action $\mathbf{a}_t \in \mathcal{A}$ suivant une politique π. En conséquence de cette action, l'agent se retrouve dans un nouvel état $\mathbf{s}_{t+1} \in \mathcal{S}$ et reçoit une récompense $r_t \in \mathcal{R} \subset \mathbb{R}$. Après plusieurs itérations, l'agent a effectué une trajectoire $\mathbf{H}_t \in \mathcal{H}$:

$$\mathbf{H}_t \doteq (\mathbf{s}_0, \mathbf{a}_0, r_0, \mathbf{s}_1, \mathbf{a}_1, r_1, ..., \mathbf{s}_{t-1}, \mathbf{a}_{t-1}, r_{t-1}, \mathbf{s}_t) \tag{II.16}$$

Il est désigné par épisode l'intervalle de temps $[0, T]$ durant lequel le processus est passé par différents états jusqu'à atteindre un état final $\mathbf{s}_T \in \mathcal{S}^*$

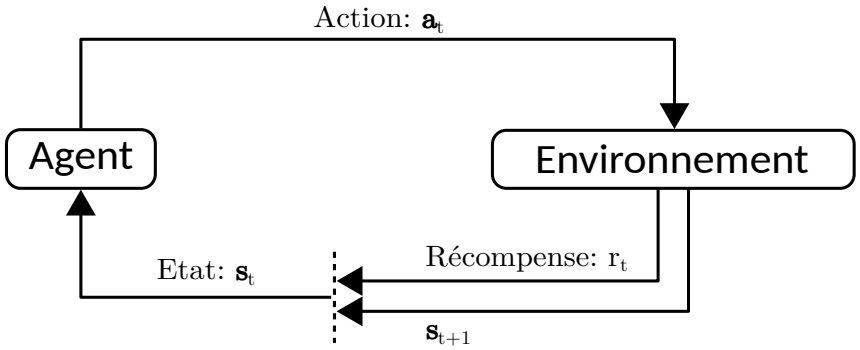


FIGURE II.9 – L'interaction agent-environnement dans un processus de décision markovien.

Un MDP doit respecter l'hypothèse de Markov qui considère que la distribution des états futurs $\mathbf{s}_{t+1}$ ne dépend que de l'état présent $\mathbf{s}_t$ et ne dépend pas des états antérieurs.

$$\mathbb{P}(\mathbf{s}_{t+1} \mid \mathbf{s}_t) = \mathbb{P}(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{s}_{t-1}, ..., \mathbf{s}_0) \tag{II.17}$$

Cette hypothèse suppose alors que l'agent a une connaissance parfaite de son environnement au travers de l'état $\mathbf{s}$. Or, dans une majorité des cas, l'agent n'a qu'une observation $\mathbf{o} \in \mathcal{O}$ partielle de son environnement. Le processus de décision markovien devient alors partiellement observable (ou *Partially Observable Markov Decision Process - POMDP*). Il existe divers moyens de se ramener au cas d'un MDP notamment par l'ajout d'une mémoire, mais ce n'est pas toujours le cas. La politique apprise pourra

tendre vers des solutions sous-optimales du problème. La suite de la théorie que nous présentons suppose être dans un MDP où $\mathbf{o} = \mathbf{s}$.

Pour résumé, un système dynamique régit par un MDP est défini par :

$$M = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma) \tag{II.18}$$

où

- ▹ $\mathcal{S}$ correspond à l'espace d'états,
- ▹ $\mathcal{A}$ correspond à l'espace d'actions. S'il est fini, l'espace d'action est dit discret. S'il est infini, l'espace d'action est dit continu.
- ▹ $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ correspond à la probabilité de transition vers l'état $\mathbf{s}' \in \mathcal{S}$ depuis un état $\mathbf{s} \in \mathcal{S}$ en prenant l'action $\mathbf{a} \in \mathcal{A}$,
- ▹ $\mathcal{R}(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ correspond à la fonction de récompense $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathrm{R}$,
- ▹ $\gamma \in [0, 1[$ un facteur de pondération qui permet d'ajuster à quel point l'agent doit tenir compte des récompenses sur le long terme ou non.

II–2.1.a Politique

Dans un MDP, la stratégie comportementale de l'agent est définie par la politique π. Cette fonction détermine la prochaine action $\mathbf{a}$ que l'agent doit entreprendre à partir d'un état $\mathbf{s}$ donné. π peut être déterministe ou stochastique :

- Politique déterministe : $\pi(\mathbf{s}) = \mathbf{a}$. La politique $\pi : \mathcal{S} \to \mathcal{A}$ associe un état $\mathbf{s} \in \mathcal{S}$ à une action $\mathbf{a} \in \mathcal{A}$.
- Politique stochastique : $\pi(\mathbf{a} \mid \mathbf{s}) = \mathbb{P}[A_t = \mathbf{a} \mid S_t = \mathbf{s}]$. La politique $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ associe une densité de probabilité sur l'espace des actions $\mathcal{A}$ à un état $\mathbf{s} \in \mathcal{S}$. L'action est alors échantillonnée sur cette distribution. Une politique déterministe peut être extraite d'une politique stochastique en prenant par exemple l'action ayant la plus grande probabilité.

II–2.1.b Fonction de valeur

L'agent doit être récompensé de manière à ce qu'en maximisant le cumul des récompenses r obtenues le long de sa trajectoire, il atteigne l'objectif souhaité. Le but est donc de trouver, par l'expérience, la politique optimale π^* qui maximise cette somme totale des récompenses.

Pour cela, l'agent utilise une fonction de gain G, indiquant la récompense qu'il obtient en suivant une trajectoire τ (indiquée par sa politique) depuis un pas de temps donné :

$$G^\tau = \sum_{t=0}^{\infty} \gamma^t r_t \tag{II.19}$$

Plus γ est proche de 1, plus l'agent s'intéresse aux récompenses sur le long terme. Plus γ est proche de 0, plus il s'intéresse aux récompenses du pas de temps suivant en ignorant le long terme. Enfin, γ évite que la somme G ne diverge, en étant strictement inférieur à 1.

Pour une politique π, la fonction état-valeur $V(\mathbf{s})$ indique le gain moyen qu'il est possible d'obtenir depuis l'état $\mathbf{s}$ en suivant la politique π :

$$V_\pi(\mathbf{s}) = \mathbb{E}_\pi[G_t \mid S_t = \mathbf{s}] = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t \mid S_t = \mathbf{s}] \tag{II.20}$$

De la même manière, la fonction action-valeur $Q(\mathbf{s})$ indique le gain moyen qu'il est possible d'obtenir depuis l'état $\mathbf{s}$ en prenant l'action $\mathbf{a}$ puis en suivant la politique π :

$$Q_\pi(\mathbf{s},\mathbf{a}) = \mathbb{E}_\pi[G_t \mid S_t = \mathbf{s}, A_t = \mathbf{a}] = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t \mid S_t = \mathbf{s}, A_t = \mathbf{a}] \tag{II.21}$$

Notons par ailleurs la relation entre V et Q où :

$$V_\pi(\mathbf{s}) = \sum_{\mathbf{a}\in\mathcal{A}} \pi(\mathbf{a} \mid \mathbf{s}) Q_\pi(\mathbf{s},\mathbf{a}) \tag{II.22}$$

L'agent peut prendre des trajectoires suivant une infinité de politiques différentes. Parmi ces politiques, il en existe au moins une telle que la fonction état-valeur $V(\mathbf{s})$ ou la fonction action-valeur $Q(\mathbf{s})$ est supérieure à toutes les autres. La fonction état-valeur optimale V^* est donnée par :

$$\forall \mathbf{s} \in \mathcal{S}, V^*(\mathbf{s}) = \max_\pi V_\pi(\mathbf{s}) \tag{II.23}$$

La fonction action-valeur optimale Q^* est donnée par :

$$\forall \mathbf{a} \in \mathcal{A}, \forall \mathbf{s} \in \mathcal{S}, Q^*(\mathbf{s},\mathbf{a}) = \max_\pi Q_\pi(\mathbf{s},\mathbf{a}) \tag{II.24}$$

L'objectif final du RL est donc de trouver cette politique optimale π^* qui maximise V (ou Q). Le choix de l'action à effectuer est celle qui amènera l'espérance de gain la plus forte :

$$\begin{aligned} \pi^* &\in \arg\max_{\mathbf{a}\in\mathcal{A}} V^*(\mathbf{s}) \\ \pi^* &\in \arg\max_{\mathbf{a}\in\mathcal{A}} Q^*(\mathbf{s},\mathbf{a}) \end{aligned} \tag{II.25}$$

II–2.1.c Équation de Bellman

Les équations de Bellman fournissent une solution exacte au problème de maximisation défini précédemment. Or les fonctions de valeur V et Q satisfont une relation de

récurrence illustrée sur la Figure II.10. La fonction de valeur d'un état **s** peut s'exprimer selon la récompense immédiate r et la fonction de valeur de l'état suivant $\mathbf{s}'$.

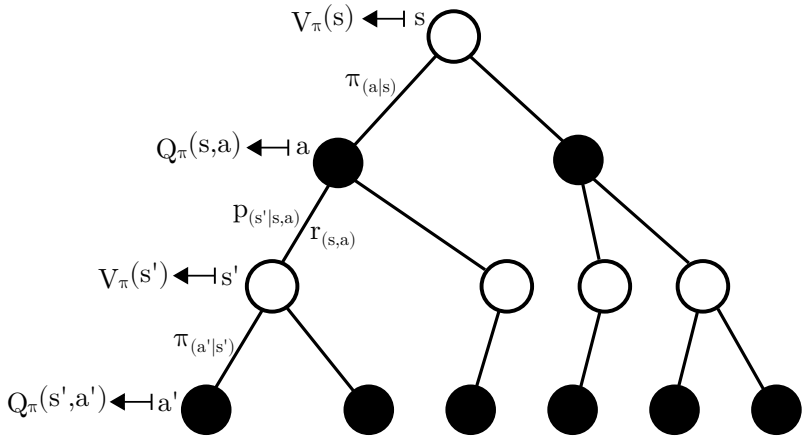


FIGURE II.10 – Récursivité des MDP. Un rond blanc correspond à un état, tandis qu'un rond noir correspond à une action. Dans cet exemple, l'espace d'action est discret de dimension 2.

Les équations de Bellman dans le contexte des MDP correspondent à la relation :

$$\begin{aligned} V^\pi(\mathbf{s}) &= \mathbb{E}_\pi\left[R_t + \gamma V^\pi(\mathbf{s}') \mid S_t = \mathbf{s}\right] \\ Q^\pi(\mathbf{s},\mathbf{a}) &= \mathbb{E}_\pi\left[R_t + \gamma \mathbb{E}_{\mathbf{a}'\sim\pi}[Q^\pi(\mathbf{s}',\mathbf{a}')] \mid S_t = \mathbf{s}, A_t = \mathbf{a}\right] \end{aligned} \tag{II.26}$$

Ces relations s'expriment également pour les fonctions de valeurs optimales :

$$V^*(\mathbf{s}) = \max_{\mathbf{a}} \mathbb{E}_\pi\left[R_t + \gamma V^*(\mathbf{s}') \mid S_t = \mathbf{s}\right] \tag{II.27}$$

où en tenant compte de la relation entre V et Q (Eq. (II.22)) :

$$\begin{aligned} V^*(\mathbf{s}) &= \max_{\mathbf{a}} \sum_{\mathbf{a}\in\mathcal{A}} \pi(\mathbf{a} \mid \mathbf{s}) Q^*(\mathbf{s},\mathbf{a}) \\ &= \max_{\mathbf{a}} Q^*(\mathbf{s},\mathbf{a}) \\ Q^*(\mathbf{s},\mathbf{a}) &= \mathbb{E}_\pi\left[R_t + \gamma \max_{\mathbf{a}'} Q^*(\mathbf{s}',\mathbf{a}') \mid S_t = \mathbf{s}, A_t = \mathbf{a}\right] \end{aligned} \tag{II.28}$$

décrivant ainsi les équations de Bellman optimales.

II–2.1.d Résolution par programmation dynamique

Les méthodes de base de l'apprentissage par renforcement cherchent donc à déterminer la fonction de valeur associée au MDP. Une fois connue, la politique est simplement de choisir l'action **a** qui mène à l'état **s** où l'espérance de gain est la plus forte, on parle de sélection gloutonne. La solution au problème peut se trouver selon deux cas d'études :

— Le MDP du problème est connu. Cela signifie que la fonction de transition p est connue. Dans ce cas, l'équation de Bellman peut-être résolue de manière analytique en exprimant le problème sous forme matricielle avec comme solution : $V^{\pi} = (I - \gamma \mathcal{T}^{\pi})^{-1} \mathcal{R}^{\pi}$. On parle de méthode basée modèle (ou *model-based RL*).

— Le MDP n'est pas connu. Dans ce cas la fonction de valeur est apprise de manière itérative grâce à l'expérience. On parle de méthode non basée modèle (ou *model-free RL*).

Dans la majorité des cas d'études, le MDP n'est pas connu et la résolution est effectuée par programmation dynamique. La politique optimale est alors déterminée de la manière suivante. La fonction de valeur V ou Q pour chaque état est initialisée arbitrairement. Pour une politique π fixe, la fonction de valeur peut être approchée en itérant suffisamment :

$$\begin{gathered} V(\mathbf{s})^{\pi}_{k+1} \leftarrow V(\mathbf{s})^{\pi}_{k} + \alpha[G(\mathbf{s})^{\pi} - V(\mathbf{s})^{\pi}_{k}] \\ \text{ou} \\ Q(\mathbf{s},\mathbf{a})^{\pi}_{k+1} \leftarrow Q(\mathbf{s},\mathbf{a}))^{\pi}_{k} + \alpha[G(\mathbf{s})^{\pi} - Q(\mathbf{s},\mathbf{a}))^{\pi}_{k}] \end{gathered} \tag{II.29}$$

où $G(\mathbf{s})$ est issu de l'expérience. Sa valeur peut être calculée à partir de l'ensemble des récompenses de la trajectoire $G(\mathbf{s}) = \sum_{t=0}^{\infty} \gamma^t r_t(\mathbf{s})$ aussi appelée méthode de Monte-Carlo (MC). $G(\mathbf{s})$ peut également se calculer en utilisant la relation de récursivité de Bellman $G(\mathbf{s}) = r + \gamma V(\mathbf{s}')$ aussi appelée différence temporelle (ou *TD learning*). L'avantage du TD learning est de ne pas avoir besoin de parcourir l'ensemble de la trajectoire (et donc de stocker états et récompenses) pour modifier la valeur de V/Q.

Ce processus itératif est nommé évaluation de la politique. La fonction de valeur résultante n'est valable que pour une politique π_0 initialisée arbitrairement. A partir de cette fonction de valeur, la politique est améliorée en suivant une sélection d'actions gloutonnes.

$$\pi_{k+1} = \max_{\mathbf{a}\in\mathcal{A}} V^{\pi_k} \quad ou \quad \pi_{k+1} = \max_{\mathbf{a}\in\mathcal{A}} Q^{\pi_k} \tag{II.30}$$

En réalisant plusieurs séquences d'évaluation de la politique puis d'amélioration de la politique telles qu'illustrées sur la Figure II.11, le processus converge vers la politique optimale si les espaces d'état et d'action sont suffisamment visités.

Bien que les méthodes MC et TD soient les deux premières techniques employées, il existe de nombreuses variantes pour calculer $G(\mathbf{s})^{\pi}$ de l'Eq. (II.29) et évaluer la politique. L'une des méthodes les plus connues est celle du Q-Learning qui intègre évaluation et amélioration dans la même itération de la fonction action-valeur Q :

$$Q(\mathbf{s},\mathbf{a}) \leftarrow Q(\mathbf{s},\mathbf{a}) + \alpha[r + \gamma \max_{\mathbf{a}'} Q(\mathbf{s}',\mathbf{a}') - Q(\mathbf{s},\mathbf{a})] \tag{II.31}$$

Bien que le Q-learning soit très efficace pour déterminer la politique optimale, elle

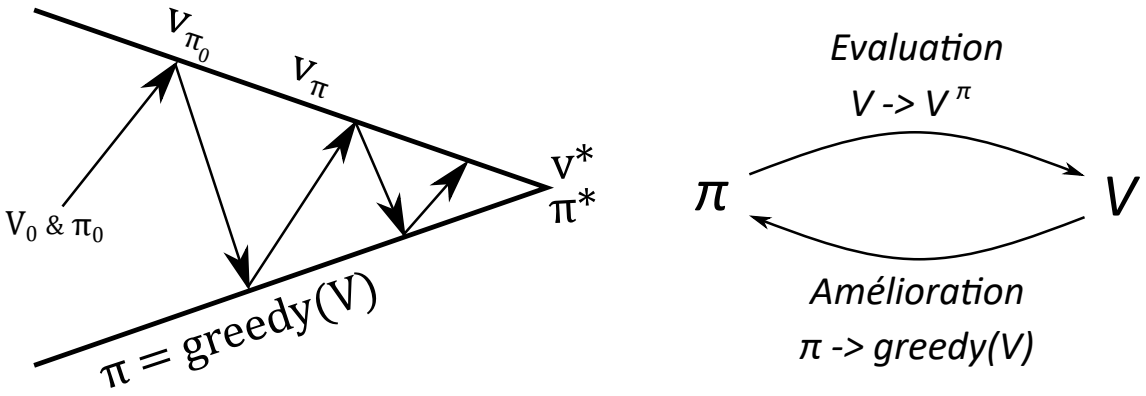


FIGURE II.11 – Processus itératif d'évaluation puis d'amélioration de la politique jusqu'à atteindre le comportement optimal π^*.

suppose que l'agent soit capable de visiter l'ensemble des états mais également de pouvoir stocker en mémoire l'ensemble des valeurs de $Q(\mathbf{s}, \mathbf{a})$ pour tous les $\mathbf{s}$ et $\mathbf{a}$. Dans des environnements réalistes, la dimension de l'espace d'état devient infiniment grande ce qui rend impossible le stockage de l'ensemble des valeurs. Pour cette raison, les méthodes actuelles d'apprentissage par renforcement modélisent les fonctions de valeur par un réseau de neurones. De cette combinaison est introduit l'apprentissage par renforcement profond.

II–2.2 Apprentissage par renforcement profond

L'approximation de la fonction de valeur à l'aide de réseaux de neurones a ouvert la porte à la résolution de problèmes complexes. Dans un premier temps au sein d'environnements virtuels, facilement répétables comme les jeux vidéo [Mnih et al., 2015], puis dans un second temps sur des systèmes plus réalistes évoluant au sein d'environnements remplis de perturbations incertaines.

La modélisation par réseaux de neurones de la fonction de valeur n'est pas l'unique moyen d'apprendre une politique. Certaines méthodes cherchent à modéliser directement la fonction π sans passer par l'approximation d'une fonction de valeur. Ces méthodes sont appelées *policy gradient*. D'autres combinent estimation de la politique et estimation de fonction de valeur en créant une architecture dite Acteur-Critique. Une classification non-exhaustive des algorithmes de RL est donnée sur la Figure II.12.

II–2.2.a Méthodes value-based

Les méthodes basées fonction de valeur cherchent à optimiser les paramètres $\boldsymbol{\theta}$ de la fonction paramétrique $V(\mathbf{s}; \boldsymbol{\theta})$ / $Q(\mathbf{s}, \mathbf{a}; \boldsymbol{\theta})$ de façon à approcher $V(\mathbf{s})$ / $Q(\mathbf{s}, \mathbf{a})$. Pour

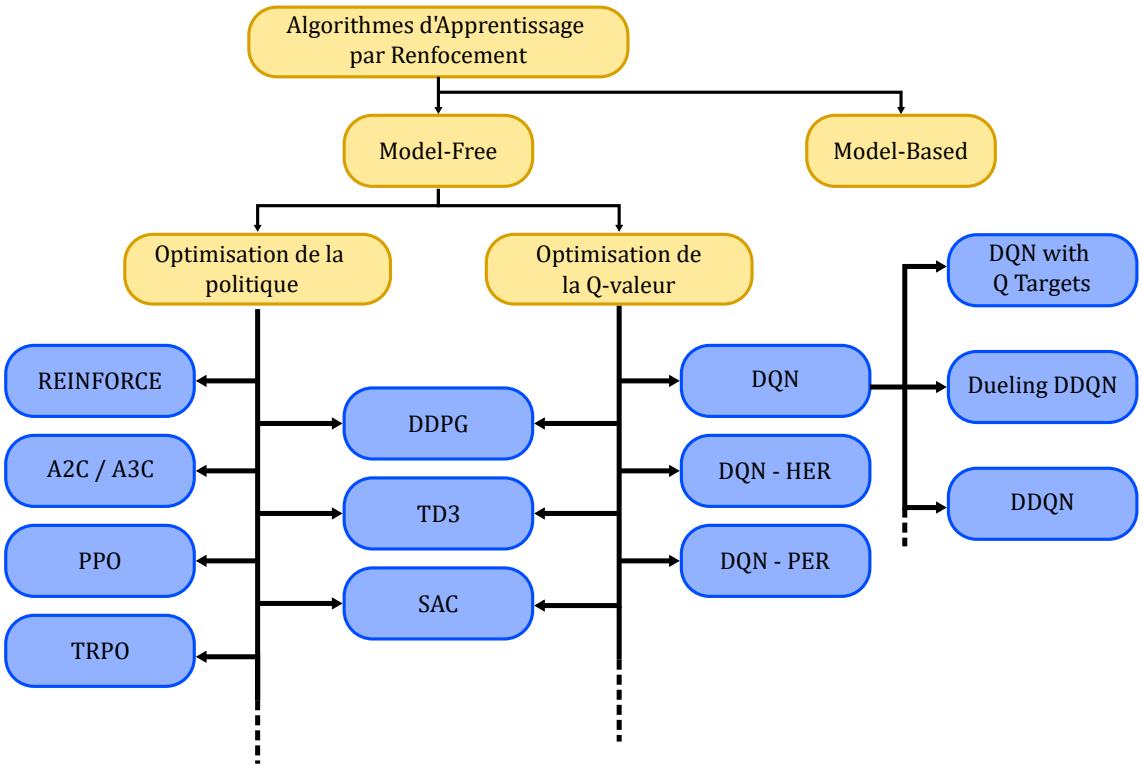


FIGURE II.12 – Classification des algorithmes d'apprentissage par renforcement établie par OpenAI [2023].

cela, la fonction objectif est définie comme suit :

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\pi}\left[(V^{\pi}(\mathbf{s}) - \hat{V}^{\pi}(\mathbf{s},\boldsymbol{\theta}))^2\right] \tag{II.32}$$

L'espérance est approximée par un échantillonnage d'expériences où les paramètres $\boldsymbol{\theta}$ sont mis à jour par descente de gradient stochastique sur la fonction objectif suivante :

$$\begin{aligned} &\boldsymbol{\theta}' = \boldsymbol{\theta} - \alpha\Delta\boldsymbol{\theta} \\ &\text{où} \\ &\Delta\boldsymbol{\theta} = -\frac{1}{2}\alpha\nabla_{\boldsymbol{\theta}}J(\boldsymbol{\theta}) \\ &\qquad = -\alpha(V^{\pi}(\mathbf{s}) - \hat{V}^{\pi}(\mathbf{s},\boldsymbol{\theta}))\nabla_{\boldsymbol{\theta}}\hat{V}^{\pi}(\mathbf{s},\boldsymbol{\theta}) \end{aligned} \tag{II.33}$$

Cependant, en RL, nous ne disposons pas de la valeur cible $V^{\pi}(\mathbf{s})$ pour superviser l'entraînement. A la place, $V^{\pi}(\mathbf{s})$ est substitué par des relations contenant les récompenses acquises lors de l'expérience. C'est ici qu'intervient de nouveau les équations de Bellman nous permettant de remplacer $V^{\pi}(\mathbf{s})$ par $r + \gamma\hat{V}^{\pi}(\mathbf{s}';\boldsymbol{\theta})$. Les paramètres sont donc appris à partir d'une valeur cible issue de la récompense immédiate et d'une estimation biaisée de l'état suivant $\mathbf{s}'$ provenant du modèle en cours d'apprentissage

(aussi appelé *bootstrapping*).

$$J(\boldsymbol{\theta}) = (\underbrace{(r + \gamma \hat{V}^{\pi}(\mathbf{s}'; w))}_{V(\mathbf{s})\text{Cible}} - \hat{V}^{\pi}(\mathbf{s}, \boldsymbol{\theta}))^2 \tag{II.34}$$

Il existe diverses combinaisons de substitution de $V^{\pi}(\mathbf{s})$ parmi lesquels les algorithmes classiques :

— Méthode de Monte Carlo : $J(\boldsymbol{\theta}) = (G_s - \hat{V}^{\pi}(\mathbf{s}, \boldsymbol{\theta}))^2$. Cette méthode impose d'attendre la fin de la trajectoire pour calculer G_s.

— Difference temporelle (TD(0)) : $J(\boldsymbol{\theta}) = (r + \gamma \hat{V}^{\pi}(\mathbf{s}'; \boldsymbol{\theta})) - \hat{V}^{\pi}(\mathbf{s}, \boldsymbol{\theta}))^2$

— Difference temporelle (TD(n)) : $J(\boldsymbol{\theta}) = (r + \gamma(r' + \gamma^2(...(r^n + \gamma^n \hat{V}^{\pi}(\mathbf{s}^n; \boldsymbol{\theta})) - \hat{V}^{\pi}(\mathbf{s}, \boldsymbol{\theta}))^2$

— Deep Q-learning (DQN) : $J(w) = (r + \gamma \max_{\mathbf{a}'} \hat{Q}(\mathbf{s}', \mathbf{a}', \boldsymbol{\theta}) - \hat{Q}(\mathbf{s}, \mathbf{a}, \boldsymbol{\theta}))^2$

L'algorithme DQN [Mnih et al., 2013] a connu de grands succès et de nombreuses améliorations pour gagner en efficacité et stabilité lors de l'apprentissage : DDQN, Dueling DDQN, ... Nous ne détaillerons cependant pas ces nombreux algorithmes qui sont aujourd'hui connus au sein de la communauté scientifique RL et dont les implémentations sont disponibles au sein de librairies Python telle que StableBaseline[Raffin et al., 2021].

II–2.2.b Méthodes policy-based

Les méthodes basées politique cherche à optimiser directement $\pi(\mathbf{s}; \boldsymbol{\theta})$ plutôt que les fonctions de valeurs V et Q pour maximiser la fonction de gain G. Dans ce cas, la fonction objectif est la suivante :

$$\begin{aligned} J(\boldsymbol{\theta}) &= \mathbb{E}_{\pi}[G(\tau)] \\ &= \sum_{\mathbf{s} \in \mathcal{S}} d^{\pi}(\mathbf{s}) V^{\pi}(\mathbf{s}) \end{aligned} \tag{II.35}$$

où $d(\mathbf{s})$ est la probabilité d'occurrence de l'état $\mathbf{s}$ en suivant π. D'où avec la relation entre V et Q donnée Eq. (II.22) :

$$J(\boldsymbol{\theta}) = \sum_{\mathbf{s} \in \mathcal{S}} d^{\pi}(\mathbf{s}) \sum_{\mathbf{a} \in \mathcal{A}} \pi(\mathbf{s}; \boldsymbol{\theta}) Q^{\pi}(\mathbf{s}, \mathbf{a}) \tag{II.36}$$

Le calcul du gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ est complexe puisque $J(\boldsymbol{\theta})$ dépend de la sélection de l'action donnée par la politique $\pi_{\boldsymbol{\theta}}$ mais aussi de la probabilité d'occurrence des états qui est indirectement déterminée par $\pi_{\boldsymbol{\theta}}$. Le théorème du *policy gradient* [Sutton

et al., 1999] permet cependant de se libérer de la contrainte et donne :

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \propto \sum_{\mathbf{s}\in\mathcal{S}} d^{\pi}(\mathbf{s}) \sum_{\mathbf{a}\in\mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(\mathbf{s};\boldsymbol{\theta}) Q^{\pi}(\mathbf{s},\mathbf{a}) \tag{II.37}$$

Grâce à l'astuce du rapport de vraisemblance où :

$$\nabla \log f(x) = \frac{\nabla f(x)}{f(x)} \tag{II.38}$$

l'Eq. (II.37) devient :

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \sum_{\mathbf{s}\in\mathcal{S}} d^{\pi}(\mathbf{s}) \sum_{\mathbf{a}\in\mathcal{A}} \pi(\mathbf{s};\boldsymbol{\theta}) \frac{\nabla_{\boldsymbol{\theta}} \pi(\mathbf{s};\boldsymbol{\theta})}{\pi(\mathbf{s};\boldsymbol{\theta})} Q^{\pi}(\mathbf{s},\mathbf{a}) \\ &= \sum_{\mathbf{s}\in\mathcal{S}} d^{\pi}(\mathbf{s}) \sum_{\mathbf{a}\in\mathcal{A}} \pi(\mathbf{s};\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log \pi(\mathbf{s};\boldsymbol{\theta}) Q^{\pi}(\mathbf{s},\mathbf{a}) \\ &= \mathbb{E}_{\pi}[\nabla_{\boldsymbol{\theta}} \log \pi(\mathbf{s};\boldsymbol{\theta}) Q^{\pi}(\mathbf{s},\mathbf{a})] \end{aligned} \tag{II.39}$$

Le calcul de l'espérance est remplacé par une estimation de la distribution basée sur un échantillonnage de trajectoires. La forme généralisée de la fonction objectif, introduite par Schulman et al. [2016] et sur laquelle se base l'ensemble des méthodes basées politiques, est :

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\pi}\Big[\sum_{t=0}^{\infty} \psi_t \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t \mid \mathbf{s}_t)\Big] \tag{II.40}$$

où ψ_t peut-être :

— $G(\tau)$ la récompense totale de la trajectoire. Cet algorithme est appelé *REINFORCE*.

— $Q_{\pi}(\mathbf{s},\mathbf{a})$, dénommé *Q Actor-Critic*. Ce type d'algorithmes combinant politique et fonction de valeur implique l'apprentissage d'un réseau de neurones pour l'acteur et d'un autre pour le critique.

— $A_{\pi}(\mathbf{s},\mathbf{a})$ la fonction Avantage où $A_{\pi}(\mathbf{s},\mathbf{a}) = Q_{\pi}(\mathbf{s},\mathbf{a}) - V_{\pi}(\mathbf{s})$. C'est la version de base des algorithmes *A2C*, et *A3C* très connu dans la littérature [Mnih et al., 2016].

— $r + V_{\pi}(\mathbf{s}) - V_{\pi}(\mathbf{s})$ version basée sur la différence temporelle.

Aujourd'hui, il existe un large choix d'algorithmes qui sont des améliorations des méthodes de base utilisant cette forme généralisée. Les plus connues d'entre elles sont notamment : TRPO [Schulman et al., 2015], PPO Schulman et al. [2017], TD3 Fujimoto et al. [2018], DDPG [Lillicrap et al., 2016], SAC Haarnoja et al. [2018]. Chacun de ces algorithmes propose une amélioration dans la manière d'échantillonner

les trajectoires et actions, mais également dans l'architecture permettant d'estimer la politique et les fonctions de valeur.

Il existe plusieurs avantages pour les méthodes basées politiques. Elles possèdent généralement des meilleures propriétés de convergence. De plus, elles sont bien plus efficaces sur des espaces d'état et d'action (continu et discret) de grande dimension. La modélisation de la politique peut s'effectuer par une fonction paramétrique stochastique impliquant une bien meilleure exploration des états que les méthodes value-based où l'action est choisie de manière gloutonne. Nous évoquerons le dilemme exploration/exploitation dans la prochaine sous-section. Les méthodes policy-gradient peuvent cependant converger fréquemment vers un minimum local et avoir un temps d'entrainement bien plus long.

II–2.2.c On-policy vs. Off-policy

Parmi les algorithmes cités précédemment, certains nécessitent que la trajectoire d'expériences utilisée pour apprendre soit issue de la même politique utilisée pour la générer. Ce type d'apprentissage est appelé *on-policy*. Au contraire, dans les apprentissages de type *off-policy*, il est possible d'utiliser n'importe quelles trajectoires issues de politiques différentes.

Les méthodes on-policy sont plus contraignantes puisqu'elles demandent de générer suffisamment d'expériences (pour des questions de stabilité) dès que les paramètres de la politique sont changés. Bien que cette approche ait tendance à converger plus vite vers une solution, la politique apprise peut rapidement converger vers un minimum local. L'avantage des méthodes off-policy est de pouvoir constamment collecter de l'expérience au sein d'un *replay-buffer* en utilisant n'importe quelle politique, notamment pour de l'apprentissage continu. Il est donc possible d'utiliser des politiques de démonstration mises en place par l'ingénieur pour faciliter la convergence vers une politique optimale. De plus, les données peuvent être massivement collectées en parallèle en suivant des politiques variées. Pour aider l'apprentissage de politiques optimales, des techniques d'échantillonnage depuis le replay-buffer peuvent être utilisées pour sélectionner les expériences les plus valorisantes pour l'agent. Cependant, l'utilisation d'un buffer ralentit l'apprentissage puisque l'expérience piochée peut être issue d'une politique aléatoire comme d'une politique bien avancée dans la résolution du problème.

II–2.3 L'exploration en apprentissage par renforcement

La convergence des algorithmes de RL vers la politique optimale est valide si et seulement si l'agent explore suffisamment l'ensemble des états de l'environnement

et de ses capacités d'actions. C'est l'une des problématiques critique de l'apprentissage par renforcement que nous présentons dans la prochaine sous-section. Bien qu'il existe des solutions simples à mettre en place, elles ne suffisent plus à explorer convenablement les environnements dans lesquels les agents récents sont amenés à évoluer. La récompense par motivation intrinsèque est une solution de plus en plus privilégiée pour apprendre sur des tâches très complexes, et ce sans avoir besoin de concevoir manuellement des systèmes de récompenses spécifiques à la tâche. Nous parcourons dans une seconde sous-section ces apprentissages à l'aide de récompenses intrinsèques.

II–2.3.a Exploration vs. exploitation

Dans les méthodes Value-based, les paramètres du modèle sont mis à jour dès que l'agent reçoit une récompense. Cependant, dans les cas où l'acquisition d'une telle récompense est rare, aussi appelée *sparse reward*, l'agent peut ne jamais trouver de solution et se retrouver bloqué sans jamais avoir atteint de récompense. Puis, s'il tombe sur l'une de ces récompenses, la fonction de valeur va de suite être modifiée et l'agent va donc, par sa politique gloutonne, se diriger dès que possible vers cet état de récompense. Cependant, il est probable qu'un autre chemin plus complexe lui amène encore plus de récompenses, mais qu'il ne découvrira pas car il n'a pas suffisamment exploré. La question est donc de savoir quand et comment l'agent doit explorer l'environnement, d'une part, et quand il doit exploiter la politique, d'autre part.

Une solution classique est de prendre de temps en temps des actions aléatoires, de manière à explorer l'environnement et le reste du temps à prendre des actions de façon gloutonne en suivant la fonction de valeur apprise. Cette technique est celle de l'*ϵ-greedy* [Mnih et al., 2013]. Le paramètre ϵ régule la fréquence à laquelle l'agent prend des actions aléatoires ou des actions gloutonnes. Le principe est de favoriser les actions aléatoires sur les premières trajectoires puis d'augmenter petit à petit (suivant différentes lois : linéaire, exponentielle, ...) la chance d'exploitation des connaissances au fur et à mesure de l'apprentissage.

Les méthodes policy-based sont moins impactées par ce problème puisque la sortie du modèle est souvent une distribution sur l'espace d'action ce qui est équivalent à l'*ϵ-greedy*. Ce type d'exploration fonctionne généralement bien sur des espaces de faible dimension mais il n'est plus suffisamment efficace dès lors que l'environnement se complexifie. D'autres stratégies ont donc vu le jour pour encourager l'agent à explorer comme l'ajout d'un terme d'entropie dans la fonction objectif pour favoriser la diversité des actions [Haarnoja et al., 2018, Schulman et al., 2017], ou encore l'ajout de bruit dans l'espace d'état entraînant un changement dans la distribution de l'action à prendre [Fortunato et al., 2018].

Une bonne exploration devient particulièrement difficile lorsque l'environnement

fournit rarement des récompenses en guise de retour d'information. Dans ce cas, la solution la plus intuitive est le *reward shaping*. Cette technique vise à compléter la récompense initiale par une seconde source de récompense qui doit conduire l'agent vers les récompenses primaires. Même si la technique est très efficace pour converger vers une solution, elle demande d'être élaborée à la main et nécessite souvent de nombreuses itérations avant de fonctionner. De plus, un biais humain est introduit dans les politiques que l'agent trouvera pour résoudre le problème. De plus, le reward shaping suppose tout de même une certaine connaissance de l'environnement pour être employé.

II–2.3.b RL non supervisé : motivation intrinsèque

Dans le même principe que le reward shaping, l'idée de la motivation intrinsèque est de récompenser régulièrement un agent pour l'encourager à découvrir son environnement et ses capacités grâce à une récompense intrinsèque r^i. Une tâche peut alors être apprise par la combinaison d'un signal de récompense associé au problème, c'est-à-dire la récompense extrinsèque r^e, et une récompense intrinsèque impliquant que $r_t = r_t^e + \beta r_t^i$ avec β un hyperparamètre pour ajuster l'influence de r^i (voir la Figure II.13). La motivation intrinsèque est inspirée du comportement de l'enfant où l'étude de Oudeyer and Kaplan [2007] traduit les mécanismes exploratoires d'un enfant, en un signal de récompense destiné à l'apprentissage par renforcement. Elle est définie comme l'accomplissement d'une activité pour sa propre satisfaction plutôt que par une pression externe.

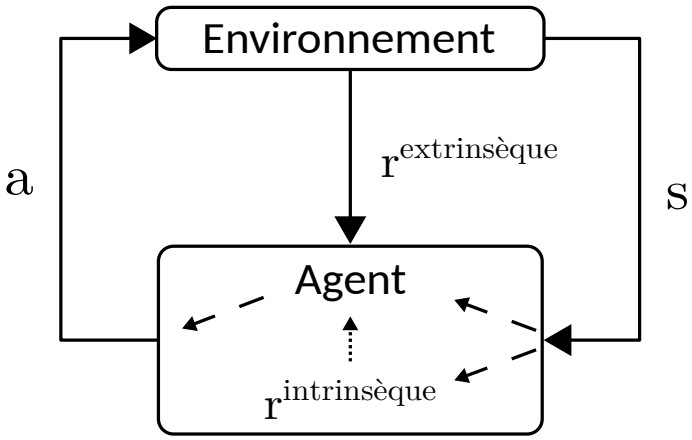


FIGURE II.13 – L'interaction agent-environnement en présence d'une récompense extrinsèque et intrinsèque.

La récompense intrinsèque peut provenir de différentes sources corrélées à la curiosité, la surprise ou encore la familiarité des observations reçues. Trois grandes familles de motivation intrinsèque se détachent dans la littérature [Laskin et al., 2021, Oudeyer and Kaplan, 2007] :

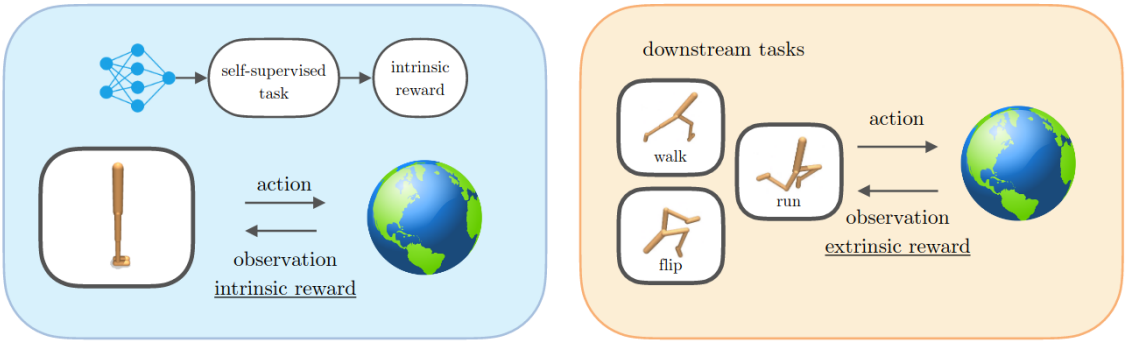
— *Data-based* : La motivation est liée à une modélisation de la distribution des observations et de la probabilité d'observer certains événements. Parmi ces approches, on trouve la mesure de la nouveauté au travers de méthodes d'exploration par comptage où l'agent compte la fréquence d'apparition $N(\mathbf{s})$ d'un état et est récompensé s'il visite avec le moins de redondance possible chacun des états. Cette idée de diversité est reprise par les méthodes de mesure de l'entropie $\mathcal{H}(\mathbf{s})$ sur l'espace d'état $\mathcal{S}$ pour fonctionner sur un espace continu en mesurant la fréquence de visite des états grâce à l'estimation d'un modèle de densité. La motivation par la surprise fait également partie de cette famille où l'agent se récompense selon la probabilité d'occurrence d'un événement $P(e)$ non attendu, $r^i(e) = \frac{1-P(e)}{P(e)}$.

— *Knowledge-based* : La motivation est liée à la notion du savoir de l'agent concernant son environnement. Cette notion de familiarité de son environnement passe par l'utilisation d'un modèle de prédiction appris en parallèle de la politique. L'idée est d'utiliser comme récompense de motivation l'erreur $r^i = |f(\mathbf{s},\mathbf{a}) - \mathbf{s}'|^2$ où $f(\mathbf{s},\mathbf{a}) = \hat{\mathbf{s}}'$ est le modèle de prédiction de l'état suivant à partir de l'état actuel $\mathbf{s}$ et de l'action $\mathbf{a}$ qui va être effectuée. L'agent est ainsi encouragé à visiter les états pour lesquels les prédictions sont les plus mauvaises. D'autres formes de récompense intrinsèque utilisent cette erreur notamment au travers de la mesure du progrès qui compare l'erreur moyenne du prédicteur $< f(\mathbf{s},\mathbf{a})_t >$ entre plusieurs pas de temps. Dans ce cas la récompense $r^i = < f(\mathbf{s},\mathbf{a})_t > - < f(\mathbf{s},\mathbf{a})_{t+k} >$ encourage l'agent à s'améliorer sur les états qu'il connaît le moins. Il existe aussi la mesure de surprise de prédiction où un second prédicteur estime l'erreur que va produire $f(\mathbf{s},\mathbf{a})$ motivant l'agent à se déplacer là où la surprise est la plus grande.

— *Competence-based* : La motivation est liée à la mesure des compétences qu'a l'agent pour atteindre son but. La récompense est issue d'une mesure d'incompétence $l = |\tilde{g} - g|$ comparant le but souhaité g à celui qui a réellement été atteint $\tilde{g}$. Si $r^i = l$, l'agent est récompensé pour les objectifs sur lesquels il performe le moins bien. À l'inverse, l'agent peut être récompensé pour le gain de compétence et donc du progrès qu'il fait au fur et à mesure de son apprentissage $r^i = l_t - l_{t+k}$.

Les signaux de motivation intrinsèque des trois familles sont aussi utilisés comme outil de pré-entraînement d'agents avant de les entraîner spécifiquement sur une tâche. Ce type d'apprentissage se retrouve sous le nom d'apprentissage par renforcement non-supervisé (ou *Unsupervised Reinforcement Learning - URL*) [Laskin et al., 2021]. La différence entre l'URL et le RL par motivation intrinsèque est que, dans le cas de l'URL, l'agent va se récompenser seul. Il va découvrir de lui-même ses capacités et

l'environnement afin d'apprendre dans un second apprentissage avec une récompense extrinsèque à se spécialiser sur une tâche donnée bien plus rapidement qu'en partant d'une politique aléatoire. Comme illustré sur la Figure II.14, l'apprentissage d'un agent par URL se déroule en deux phases : exploration de son environnement par motivation intrinsèque, puis spécialisation sur une tâche par RL classique.



Part 1: Unsupervised Pre-training with Intrinsic Rewards

Part 2: Supervised Finetuning to Downstream Tasks

FIGURE II.14 – Processus d'apprentissage d'une tâche par URL. La politique pré-entraînée se généralise et converge bien plus rapidement sur diverses tâches. Crédit image : Laskin et al. [2021].

II–2.3.c Motivation par l'entropie

Parmi l'ensemble des familles d'apprentissage par renforcement non-supervisé [Laskin et al., 2021], nous sommes particulièrement intéressés par les algorithmes dit "Data-based" visant à maximiser l'entropie d'une distribution sur un espace donné. Comme introduit dans le Chapitre I, en utilisant l'un de ces algorithmes, nous souhaitons faire varier les mouvements d'un objet en maximisant l'entropie d'un espace qui le représente. Une comparaison de ces méthodes basées sur l'entropie est donnée dans le Tableau II.1.

Après sélection d'un espace $\mathcal{S}^{\text{ent}} \subseteq \mathcal{S}$ l'agent est entraîné à maximiser l'entropie de la distribution d des observations $\mathbf{s}^{\text{ent}}$ acquises en suivant la politique π en cours d'apprentissage. Le calcul de cette entropie varie selon la méthode utilisée et nous détaillerons uniquement au Chapitre IV le fonctionnement de l'algorithme sélectionné. Ce choix peut s'effectuer en fonction des critères recensés dans le Tableau II.1. La colonne "Espace d'état maximisé" indique si l'espace $\mathcal{S}^{\text{ent}}$ doit être discret pour pouvoir calculer l'entropie ou s'il peut être continu. Il est important de faire la différence entre l'état $\mathbf{s}$ et l'observation $\mathbf{s}^{\text{ent}}$. Les contraintes du tableau sont données pour l'observation $\mathbf{s}^{\text{ent}}$ et non pas pour l'état $\mathbf{s}$. Les algorithmes ProtoRL, APT et RE3 permettent de maximiser l'entropie sur un espace $\mathcal{S}^{\text{ent}}$ de grande dimension (par exemple une

Algorithme	**Espace d'entropie maximisé**	**Visuel**	**Off-Policy**
MaxEnt [Hazan et al., 2019]	Discret	Non	Non
MEPOL [Mutti et al., 2020]	Continu	Non	Non
ProtoRL [Yarats et al., 2021]	Continu	Oui*	Oui
APT [Liu and Abbeel, 2021]	Continu	Oui*	Oui
RE3 [Seo et al., 2021]	Continu	Oui	Oui

TABLE II.1 – Algorithmes de pré-entrainement RL basés maximisation de l'entropie de l'espace d'état. Espace d'état maximisé : indique si l'entropie nécessite une discrétisation de l'espace $\mathcal{S}^{\text{ent}}$. Visuel : la méthode fonctionne bien sur la maximisation d'entropie mesurée sur un espace $\mathcal{S}^{\text{ent}}$ de grande dimension tel que des images. Off-Policy : compatible avec une optimisation off-policy et l'utilisation d'un Replay Buffer. $*$ utilisation de méthodes dites "representation learning" pour encoder l'espace sur lequel est mesuré l'entropie.

image). Même si $\mathcal{S}^{\text{ent}}$ ne peut être de grande taille pour MaxEnt et MEPOL, rien n'empêche a priori que l'état $\mathbf{s}$ soit une image. Nous justifierons le choix de l'algorithme au Chapitre IV.

CHAPITRE III

Représentation des caractéristiques physiques et dynamiques d'un objet

Sommaire

Pour un agent physique ou virtuel, comprendre le comportement dynamique d'un objet peut permettre de planifier et d'exécuter des actions efficacement dans le monde réel. Lorsqu'un humain perçoit un objet mis en mouvement par ses actions ou par toute autre perturbation externe, sa compréhension de la physique lui permet de prédire le comportement futur de cet objet, mais aussi de choisir l'action appropriée pour l'amener dans un état particulier McCloskey [1983]. Ce processus de raisonnement s'appuie sur les connaissances préalables de l'objet et l'expérience directe de sa mise en mouvement, et cela de plusieurs manières :

- Les propriétés internes de l'objet sont inconnues, mais des observations antérieures peuvent donner une idée de son comportement et permettre de prédire vers quelles positions l'objet est susceptible de se déplacer. Cela constitue une modélisation implicite du comportement de l'objet.
- La connaissance des propriétés physiques de l'objet, telles que sa masse, son matériau ou sa forme, permet d'approximer les vitesses qu'il peut atteindre et les distances qu'il peut parcourir, en se basant sur les lois physiques connues.
- L'affordance, un concept non abordé dans ce manuscrit, établit un lien entre une caractéristique dynamique de l'objet et un mode d'interaction spécifique, dépendant de la manière dont l'agent perçoit et interagit avec son environnement. Par exemple, l'orientation d'une poignée de tiroir peut indiquer qu'il est probable que le tiroir s'ouvre en coulissant. Équiper un agent de la capacité à détecter les affordances d'un objet peut jouer un rôle important dans son aptitude à interagir efficacement avec de nouveaux objets.

L'originalité de notre étude réside dans la capacité à identifier les propriétés d'un objet grâce à un agent physique, le tout sans connaissances a priori sur l'environnement ou les objets. L'objectif est donc de découvrir une méthode permettant de mettre en avant les propriétés physiques d'un objet sans supervision. De nombreuses études ont déjà tenté de résoudre ce problème en utilisant une tâche de prédiction des états futurs pour construire une représentation de la dynamique de l'objet. Dans cet état de l'art, nous allons montrer que les méthodes existantes ne parviennent pas à répondre pleinement au problème posé, faute de satisfaire à un ensemble de critères que nous présentons dans le Tableau III.1.

Nous décomposons ce chapitre en 3 sections. La première section est consacrée aux travaux qui établissent une modélisation implicite des propriétés dynamiques d'un objet via une tâche de prédiction de sa trajectoire. Toutefois, afin de faciliter l'interprétation des propriétés et le transfert des connaissances vers de nouvelles tâches, nous visons une formalisation explicite de ces propriétés. Comme nous le verrons dans la seconde section de ce chapitre, l'estimation directe de ces propriétés implique d'importantes contraintes sur l'environnement et en ce qui concerne les connaissances déjà

acquises sur l'objet et sur l'environnement, ce qui entre en contradiction avec notre objectif. Enfin, comprendre la physique d'un objet nécessite de l'observer en mouvement ce que nous cherchons à réaliser de manière autonome à l'aide d'un robot. La dernière section est donc consacrée à la stratégie comportementale mise en oeuvre dans les expérimentations existantes impliquant un agent physique.

TABLE III.1 – Comparaison des méthodes de l'état de l'art avec notre méthode sur plusieurs critères : représentation des propriétés (implicites ou explicites), mode d'apprentissage ((N)S : (non-)supervisé, AS : auto-supervisé, (U)RL : apprentissage par renforcement (non-supervisé)), contraintes a priori sur l'environnement et les propriétés, type de la politique (* pas d'interaction physique avec l'objet), utilisation d'un agent robotique, qualité de la simulation physique (physique réaliste / environnement réaliste).

Étude	Représentation des propriétés	Mode d'apprentissage	Contraintes a priori	Politique	Robot	Qualité du simulateur
Mottaghi et al. [2016a]	implicite	S	trajectoires labélisées	✗	✗	✗
Mottaghi et al. [2016b]	implicite	S	trajectoires labélisées	✗	✗	✗
Li et al. [2016]	implicite	S	tour d'objets + label de stabilité	✗	✗	++
Groth et al. [2018]	implicite	S	tour d'objets + label de stabilité	✗	✗	++
Lerer et al. [2016]	implicite	AS / S	tour d'objets + label de stabilité	✗	✗	++
Janner et al. [2019]	implicite	AS	tour d'objets + modèle physique	✗	✗	+
Baradel et al. [2020]	implicite / explicite	AS / S	tour d'objets + nombre d'objets	✗	✗	++
Ehrhardt et al. [2019]	implicite	AS	-	✗	✗	-
Watters et al. [2017]	implicite	AS	Sphères en mouvement + nombre d'objets + positions connues	✗	✗	-
Finn et al. [2016]	implicite	AS	Séquences vidéo d'interactions dans un espace clos	Déterministe	✓(bras)	✗

TABLE III.2 – Suite du tableau.

Étude	Représentation des propriétés	Mode d'apprentissage	Contraintes a priori	Politique d'interaction	Robot	Qualité du simulateur
Agrawal et al. [2016]	implicite	AS	Séquences vidéo d'interactions dans un espace clos	Déterministe	✓(bras)	✗
Li et al. [2018]	implicite	AS	Espace fermé à un objet	Déterministe	✓(bras)	++
Byravan and Fox [2017]	implicite	AS	Espace fermé + nombre d'objets	Déterministe	✓(bras)	++
Nematollahi et al. [2020]	implicite	AS	Espace fermé	Déterministe	✓(bras)	++
Wu et al. [2017]	implicite / explicite	AS	Sphères en mouvement + modèle physique + couleur de sphère selon propriété	✗	✗	-
Xu et al. [2019]	implicite / explicite	AS / S	Espace fermé + forme de l'environnement (plans inclinés) + environnement connu	Déterministe	✓(bras)	++
Fan et al. [2017]	explicite	s	label forme	✗	✗	✗
Standley et al. [2017]	explicite	S	label masse	✗	✗	✗
Bell et al. [2015]	explicite	S	label matériau	✗	✗	✗
Kannabiran et al. [2019]	explicite	RL / S	label masse + objet articulé + état objet	Appris	✓(bras)	++
Gouko et al. [2015]	explicite	RL / S	Objet unique + forme cube, triangle ou cylindre	Appris *	✓(mobile)	++
Gouko et al. [2017]	explicite	RL / NS	Objet unique + forme cube, triangle ou cylindre	Appris *	✓(mobile)	++
Haughton et al. [2022]	explicite	S	SVM préentrainés (matériau, rigidité, repartition de forces)	Déterministe	✓(bras)	✗
Denil et al. [2017]	explicite	RL	connaissance nombre d'objets, de leurs états et des propriétés associées (masse, nombre de corps)	Appris	✗	+

TABLE III.3 – Suite et fin du tableau.

Étude	Représentation des propriétés	Mode d'apprentissage	Contraintes a priori	Politique d'interaction	Robot	Qualité du simulateur
Wu et al. [2015]	explicite	AS	modèle physique + forme de l'environnement (plans incliné)	✗	✗	++
Wu et al. [2016]	explicite	AS	modèle physique + forme de l'environnement (plans incliné)	✗	✗	++
Link et al. [2022]	explicite	AS	modèle physique	✗	✗	-
Kandukuri et al. [2022]	explicite	AS	modèle physique + forme de l'environnement (plans incliné)	Déterministe	✗	++
Lohmann et al. [2020]	explicite	AS	relation entre la propriété et l'action + discrétisation de la masse en 3 catégories	Appris	✗	+++
Ye et al. [2018]	explicite	AS	ensembles de séquences où une seule propriété varie (masse, friction)	✗	✗	++
Zheng et al. [2018]	explicite	AS	sphères en mouvement + propriétés discrètes	✗	✗	-
Lu et al. [2019]	explicite	AS	série temporelle régit par une équation différentielle	✗	✗	✗
Nous	explicite	URL + AS	Objet unique	Appris	✓(mobile)	++

III–1 Modélisation implicite de la dynamique d'un objet

Pour permette à un agent d'apprendre à contrôler un objet, de nombreuses études s'appuient sur une méthode d'apprentissage où la trajectoire de l'objet est prédite par l'agent. L'objet peut être mis en mouvement soit par les actions de l'agent, ou soit par d'autres forces extérieures. La prédiction d'une trajectoire repose sur l'hypothèse que le modèle de prédiction de l'agent apprend la physique régissant les trajectoires de l'objet réalisant ainsi une modélisation implicite des propriétés de l'objet. L'utilisation de cette représentation permet à l'agent d'atteindre plus facilement ses objectifs [Agrawal et al., 2016, Li et al., 2018, Xu et al., 2019] et même d'en extraire explicitement les propriétés physiques [Baradel et al., 2020, Xu et al., 2019].

Après avoir implicitement appris la modélisation, le contrôle de l'objet peut s'effectuer par l'utilisation d'un modèle inverse qui détermine la meilleure action à effectuer

en se basant sur l'état courant de l'environnement et un état cible. Ce processus intègre dans le modèle l'état courant du système et un ensemble d'actions possibles depuis cet état, prévoyant l'état futur résultant de chaque action. Ensuite, une mesure de similarité entre l'état cible et l'état prédit permet de sélectionner la meilleure action. Le modèle de modélisation implicite (ou modèle direct) et le modèle inverse, illustrés sur la Figure III.1, sont généralement issus du même modèle et appris simultanément.

Dans les méthodes que nous allons évoquer, la modélisation est réalisée par des réseaux de neurones profonds dont l'architecture dépend de la nature des états à prédire. Ces modèles traitent des entrées pouvant être des images de la scène, ou des mesures telles que les positions, orientations et vitesses, avec pour tâche de prédire l'état subséquent de ces entrées.

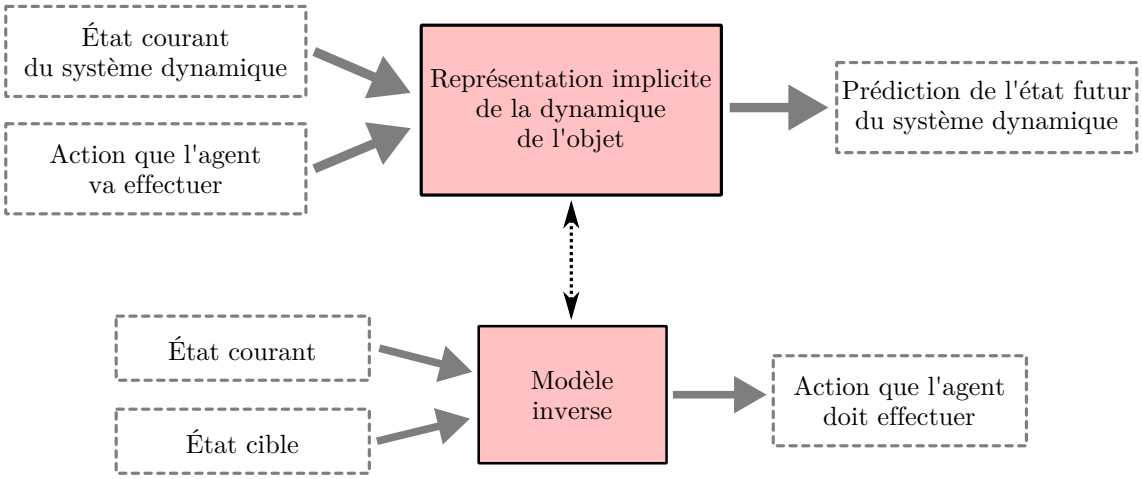


FIGURE III.1 – Apprentissage d'un modèle dynamique de l'objet avec pour objectif la planification et le contrôle des actions d'un agent.

III–1.1 Modélisation par apprentissage supervisé

L'apprentissage de ces modèles peut s'effectuer de façon supervisée. Dans [Mottaghi et al., 2016a,b] les auteurs utilisent des bases de données vidéos, où la trajectoire de l'objet est annotée, pour prédire la direction et la vitesse que prend l'objet au sein de scénarios newtoniens (chute d'un objet, trajectoire d'un projectile, poussée et frottement sur le sol). La tâche s'avère complexe puisque ces deux études se basent uniquement sur une image statique de la scène, ce qui ne permet pas de prendre en compte la dynamique de l'objet. Mottaghi et al. [2016b] incluent cependant dans l'entrée la force appliquée à l'objet.

Li et al. [2016] et Groth et al. [2018] ont utilisé la simulation pour prédire la stabilité d'une tour constituée de blocs de différentes formes. À partir de différents points de vue de la tour (voir Figure III.2), un modèle effectue une classification supervisée

binaire indiquant si elle est stable ou non. Il est ensuite possible à l'aide d'un modèle inverse de reconstruire une tour stable et de choisir les bons éléments pour éviter une perte de stabilité. La modélisation dynamique apprise grâce à la classification supervisée se limite au cas d'un ensemble d'objets dans une configuration particulière (tour de blocs).

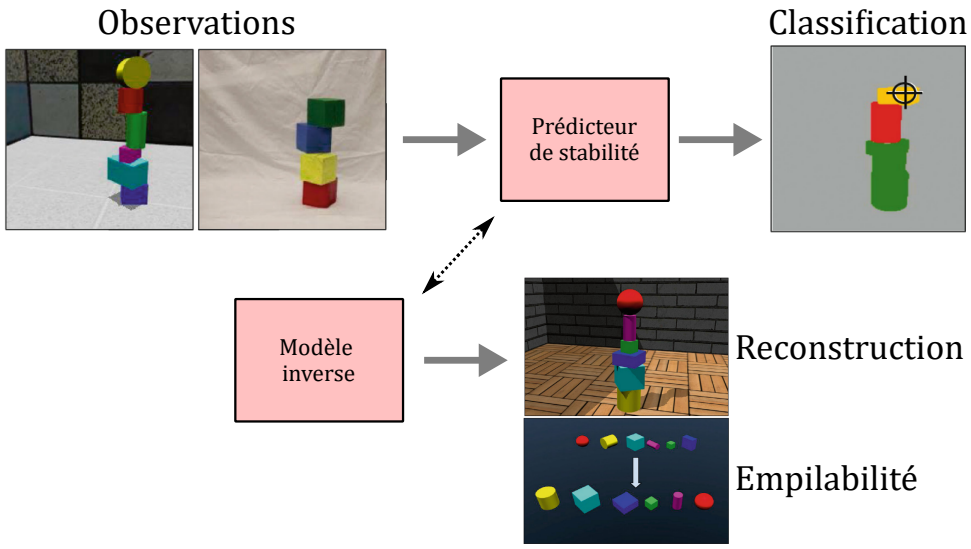


FIGURE III.2 – Illustration des expériences menées dans [Groth et al., 2018, Li et al., 2016] sur l'apprentissage d'un prédicteur de stabilité de tours. Crédit image : Groth et al. [2018].

La modélisation implicite apprise de manière supervisé n'est pas une solution compatible avec nos objectifs puisque cette modélisation nécessite une connaissance parfaite de l'environnement et de la dynamique de l'objet. Les travaux plus récents, que nous allons présenter dans la sous-section suivante, se sont focalisés sur la suppression de cette supervision.

III–1.2 Modélisation par apprentissage auto-supervisé

Un avantage majeur de la modélisation implicite dans le contexte de notre problématique est la capacité d'apprendre le modèle de manière auto-supervisée, réduisant ainsi le besoin de connaissances a priori (stabilité, direction du déplacement, vitesse, ...). Dans ce type d'apprentissage, la dynamique de l'objet est acquise uniquement avec des observations de la scène. Au sein d'un environnement simulé présentant une tour de blocs, les études menées par Baradel et al. [2020], Janner et al. [2019], Lerer et al. [2016], Wu et al. [2017] cherchent à prédire la trajectoire de chute d'une tour instable. À partir d'une image ou d'une série d'images de la tour, une architecture de type auto-encodeur est utilisée pour reconstruire l'image suivante en réalisant une représentation de plus faible dimension des observations d'entrée avant d'effectuer la prédiction. En comparant cette prédiction avec l'observation réelle issue de la simula-

tion (auto-supervision), le modèle est entraîné à produire des images plus précises de la trajectoire réelle.

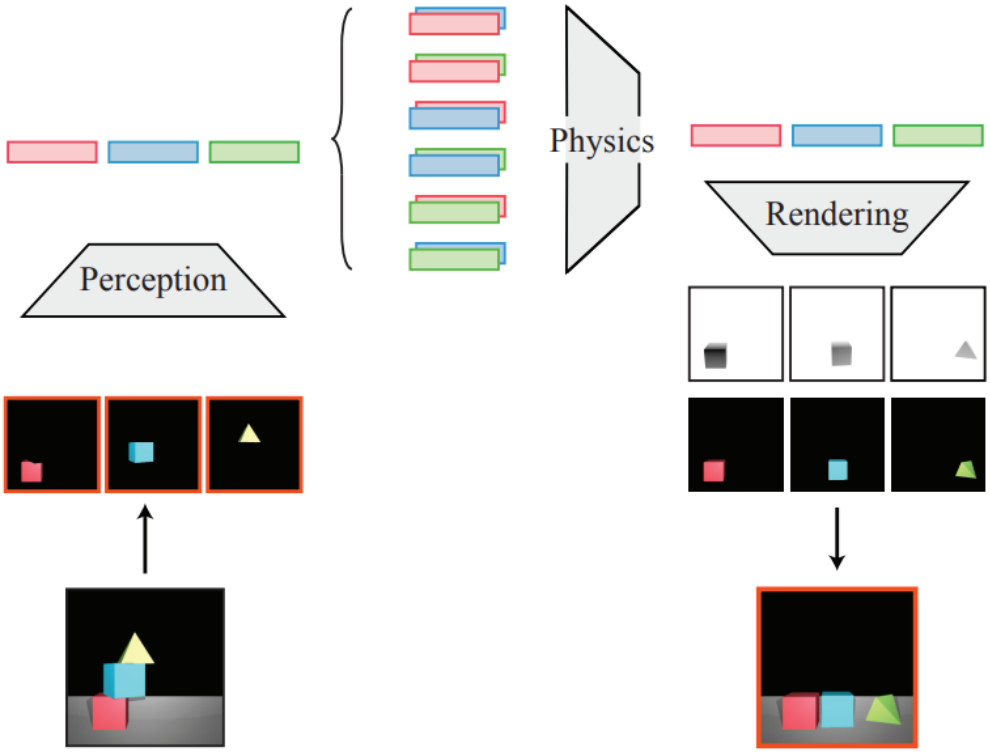


FIGURE III.3 – Architecture employée dans [Janner et al., 2019] pour établir un espace de représentation de la dynamique d'un objet. Crédit image : [Janner et al., 2019].

Cependant, la représentation obtenue est centrée sur l'ensemble de la dynamique de la scène et non pas sur un objet précis, comme nous le souhaitons dans notre étude. Pour capter la dynamique d'un objet précis, Janner et al. [2019] ont isolé chaque objet en les segmentant, avant de passer l'observation dans un module de perception. Les sorties obtenues pour chaque objet de la scène sont alors des représentations individuelles de la dynamique. Pour prédire l'état futur de chaque objet, il est nécessaire de tenir compte des interactions physiques qui peuvent se produire entre eux, notamment à l'aide de modèles dédiés [Battaglia et al., 2016, Chang et al., 2017]. Comme illustré sur la Figure III.3, un nouvel espace intégrant ces interactions est créé et devient l'entrée permettant la prédiction de l'état futur de la scène. Dans le cadre de notre étude, cette notion de relation entre objets (aussi mise en place dans [Baradel et al., 2020, Ehrhardt et al., 2019, Watters et al., 2017]) n'est pas traitée puisque nous considérons un agent évoluant face à un objet unique.

D'autres études ont réalisé une modélisation implicite d'objets, notamment dans des environnements 2D (voir Figure III.4) où des sphères se déplacent et peuvent rentrer en collision entre elles. La simulation permet de changer les propriétés intrin-

sèques de ces sphères comme la masse, les frottements ou le coefficient de rebondissement. Le principe est de prédire le déplacement d'une ou plusieurs sphères et, pour cela, la même technique d'apprentissage supervisé est utilisée pour prédire la prochaine image [Ehrhardt et al., 2019] ou alors directement les positions et vitesses des sphères dans un espace latent [Watters et al., 2017].

FIGURE III.4 – Simulation de trajectoires de sphères dont les propriétés physiques peuvent varier. Crédit image : [Zheng et al., 2018].

Bien que des propositions de solutions non-supervisées aient été proposées pour modéliser la dynamique des objets, les travaux évoqués précédemment n'intègrent pas d'agent dans la solution proposée pour interagir avec l'environnement. Dans notre contexte, si un agent se déplace vers un nouveau lieu dans lequel se trouvent un ou plusieurs objets inconnus, ces derniers ne seront pas en mouvement et ne le seront probablement pas naturellement. Par conséquent, l'agent n'obtiendra aucune information sur ces objets à partir de simples observations passives. De plus, comme nous l'avons exposé dans le Chapitre I, la maîtrise de l'interaction (section III–3 de ce chapitre) peut favoriser une représentation plus précise des propriétés physiques. Nous nous intéressons donc désormais à l'utilisation d'agents robotiques pour modéliser implicitement la dynamique des objets.

III–1.3 Utilisation d'un agent robotique

La stratégie d'apprentissage auto-supervisé mise en place dans des environnements simulés est également utilisée en robotique. Nous retrouvons ici des architectures où les modèles direct et inverse sont en général appris simultanément [Agrawal et al., 2016, Li et al., 2018, Nematollahi et al., 2020]. Des robots à bras utilisent leur capacité d'interaction pour déplacer un objet, initialement immobile, afin de comprendre sa dynamique et pouvoir plus tard l'emmener vers une position cible. La scène dans laquelle se déroule l'expérience est restreinte aux déplacements possibles de l'effecteur et est généralement composée d'un seul objet. Pour entraîner le modèle de prédiction, un ensemble de données constitué de séquences de courtes interactions du robot avec

l'objet est préalablement acquis. Dans une majorité de cas, les actions que l'agent effectue sur l'objet ne sont pas issues d'un modèle comportemental et sont choisies aléatoirement parmi un set d'actions prédéfinies nécessitant généralement de connaître parfaitement l'environnement. En effet l'exécution d'une action de ce set, par exemple "pousser l'objet dans telle direction", impose de connaître les positions et orientations de l'objet ainsi que celles l'agent.

Une fois ce set créé, Finn et al. [2016] entraînent un modèle direct composé de couches convolutives à prédire l'observation future à partir d'une seule image RGB et de l'action du robot. Dans [Agrawal et al., 2016, Li et al., 2018, Nematollahi et al., 2020], la prédiction de l'observation future s'effectue dans un espace de plus petite dimension (voir Figure III.5). L'image RGB passe dans un réseau convolutif afin d'obtenir une représentation latente des caractéristiques. Un second réseau entièrement connecté combine l'action effectuée par l'agent avec la représentation latente pour produire un nouveau vecteur qui doit s'apparenter au vecteur des caractéristiques de l'état futur. Via un réseau convolutif jumeau du premier, l'image réelle de l'état suivant est introduite pour obtenir la véritable représentation latente et la comparer à la prédiction. Ce processus constitue l'apprentissage du modèle direct.

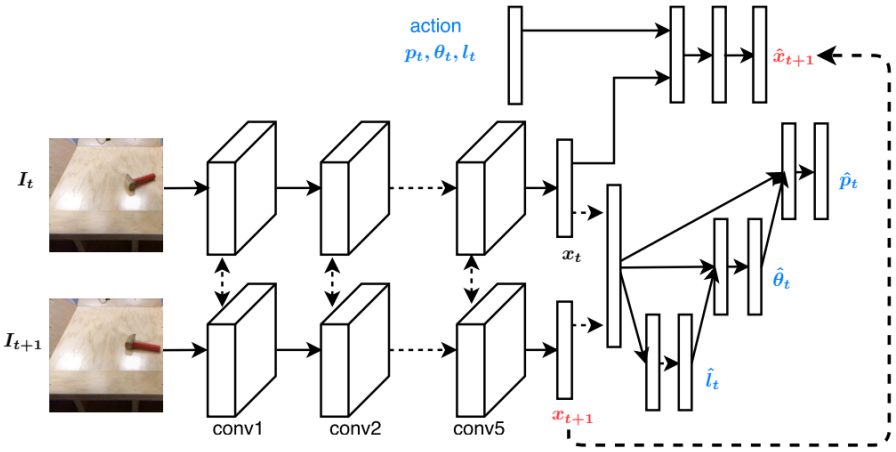


FIGURE III.5 – Architecture employée dans [Agrawal et al., 2016] visant à apprendre un modèle direct et inverse pour pousser un objet vers une position donnée. Crédit image : [Agrawal et al., 2016].

En même temps, le modèle inverse est appris par un réseau entièrement connecté qui combine la représentation de l'image à t et à $t+1$ pour prédire l'action à effectuer afin d'arriver dans l'état $t+1$ à partir de l'état t. L'action estimée est comparée à celle qui a réellement été effectuée. C'est grâce à la somme des deux fonctions de coût des modèles direct et inverse que les auteurs entraînent leur modèle. Contrairement à Agrawal et al. [2016], Li et al. [2018] construisent un espace de représentation à partir de plusieurs pas d'interactions en utilisant une cellule récurrente LSTM pour introduire un historique de la trajectoire et prendre en compte la dynamique du système. Les

auteurs de Byravan and Fox [2017], Nematollahi et al. [2020] utilisent à la place d'une image RGB un nuage de points issu d'une carte de profondeur, puis reconstruisent l'état futur sous la forme d'une modélisation $3D$ de l'environnement. Ils montrent que cela apporte une meilleure précision dans la représentation et ce, même avec des données bruitées ayant pour but une application sur un système réel.

Notre méthode se différencie de ces travaux sur plusieurs points. D'une part, il n'y a pas de recherche d'une politique pour mettre en mouvement l'objet. L'utilisation d'actions prédéfinies fait que la méthode de ces travaux n'est pas si différente des méthodes employées dans les environnements simulés de tours et de sphères mis en mouvement par des forces extérieures. Le robot n'est donc pas réellement au coeur de la méthode de modélisation de l'objet. D'autre part, la prédiction de l'état futur s'effectue à partir d'une séquence contenant une seule interaction, ce qui intuitivement, ne nous semble pas suffisant pour encoder des propriétés comme la masse ou les frottements dans les cartes de représentation latentes du modèle. Dans les études présentées, c'est en entraînant le modèle avec un grand nombre de séquences pour un même objet que la modélisation implicite de la dynamique sera établie, sans pour autant identifier les propriétés physiques. Dans notre cas, nous souhaitons construire un espace de représentation à partir d'une séquence contenant plusieurs interactions pour en extraire explicitement les propriétés physiques. Enfin, nous n'entraînons pas de modèle inverse en même temps que le modèle d'obtention de représentation des caractéristiques de l'objet. Nous souhaitons séparer le processus apprentissage en deux phases : dans un premier temps, il s'agit d'apprendre à établir la représentation puis dans un second temps de l'utiliser en données d'entrée d'une politique entraînée à résoudre une tâche spécifique, en fonction de l'état de l'environnement et des propriétés physiques des objets présents. Cette approche devrait permettre d'utiliser les connaissances acquises à plus de tâches, contrairement aux cas présentés où la modélisation de la dynamique obtenue est spécifique à la tâche.

Alors que [Agrawal et al., 2016, Li et al., 2018, Nematollahi et al., 2020] se focalisent sur des scènes contenant un seul objet et encodent l'image entière dans une représentation latente vectorielle, Xu et al. [2019] proposent une représentation de la scène de niveau pixellique, ce qui signifie que les prédictions des états futurs s'effectuent à partir d'un espace 2D. L'objectif de cette représentation est de généraliser à de nouvelles scènes contenant plusieurs objets. L'environnement est un plateau comportant des plans inclinés permettant d'effectuer 2 types d'interactions avec les objets à l'aide d'un bras robotique : la poussée et le glissement sur l'un des plans. Les actions sont choisies dans un espace prédéfini et demandent une connaissance parfaite de l'environnement (positions et orientations des éléments). L'architecture employée est détaillée sur la Figure III.5. A partir d'une carte de profondeur, les auteurs extraient une première représentation visuelle de l'image d'entrée. Cette représentation visuelle

R_v est intégrée à la représentation de l'objet R_a, qui encode l'effet de la dernière interaction sur les objets, pour former une représentation physique R_p. La représentation de l'objet après une interaction dépend des propriétés physiques de l'objet. La comparaison entre R_a et R_v doit donc mener à l'identification de la dynamique. R_a est obtenu à partir d'une cross-convolution entre le vecteur des actions et la représentation R_p. L'ensemble du modèle est entraîné en auto-supervision en comparant l'état prédit à $t+1$ à partir de la représentation R_a avec la véritable observation.

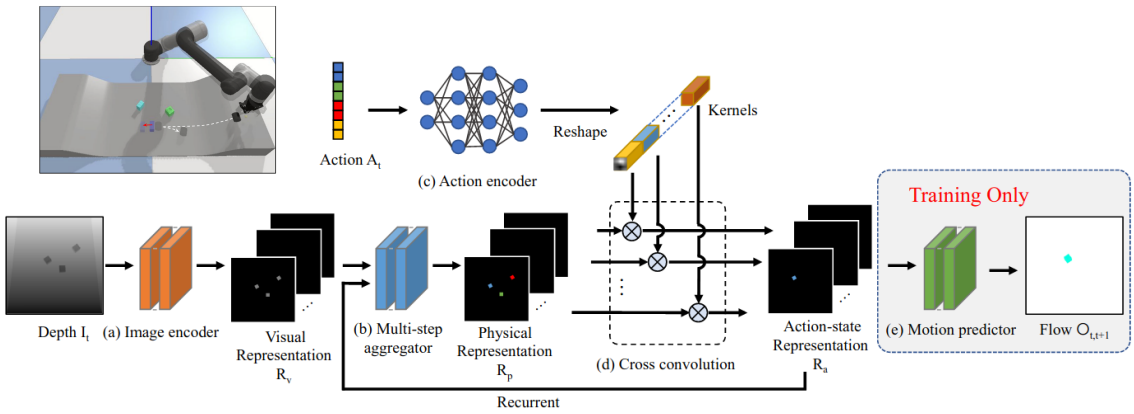


FIGURE III.6 – Environnement de simulation et architecture employée dans [Xu et al., 2019]. La prédiction de l'état futur de l'environnement s'effectue à partir d'une représentation latente pixel à pixel. Crédit image : [Xu et al., 2019].

Contrairement à [Agrawal et al., 2016, Li et al., 2018], la méthode proposée dans l'article [Xu et al., 2019] permet de construire une représentation de la dynamique d'un objet à partir de séquences d'interactions de natures variées. Via une approche supervisée, les auteurs montrent que la représentation implicite construite à partir de plusieurs séquences d'interactions caractérise de façon plus précise les propriétés de l'objet.

Les méthodes de représentation implicite décrites dans cette section présentent un intérêt, notamment pour leur capacité à entraîner les modèles en auto-supervision. La représentation ne se généralise cependant pas à de nouvelles tâches comme le montre l'étude de Xu et al. [2019], où les propriétés physiques de l'objet sont extraites de l'espace de représentations avant d'être utilisées pour une tâche différente. C'est pourquoi, dans notre méthode, nous souhaitons identifier explicitement les propriétés de l'objet. C'est cet objectif qui sera exploré dans la section suivante.

III–2 Modélisation explicite des propriétés physiques d'un objet

L'interprétabilité des propriétés physiques qu'il est possible de découvrir sur un objet constitue un aspect essentiel de nos travaux. Plusieurs études se sont consacrées sur la modélisation explicite des caractéristiques dynamiques d'un objet à partir de ses observations. Toutefois, comme nous allons le présenter dans cette section, l'estimation directe des propriétés impose bien souvent de contraindre fortement l'environnement et le mode d'apprentissage, rendant difficile la réalisation de notre objectif final.

III–2.1 Estimation par apprentissage supervisé

Les propriétés physiques d'un objet peuvent être obtenues par apprentissage supervisé à partir d'une base de données contenant un nombre important d'exemples (Figure III.7). À l'aide d'images statiques et sans interaction physique, il est possible de prédire la forme [Fan et al., 2017], la masse [Standley et al., 2017] ou encore le matériau [Bell et al., 2015] d'un objet. Comme exposé dans le chapitre introductif, le principal désavantage de ces approches est que, sans interaction physique, le modèle peut être visuellement induit en erreur par des variations de texture masquant le matériau ou la structure interne d'un objet. De plus, chaque modèle est conçu pour estimer une unique propriété à la fois.

FIGURE III.7 – Exemples d'objets contenus dans le set de données utilisé par [Standley et al., 2017] pour entraîner un réseau à estimer la masse d'un objet du quotidien. Crédit image : [Standley et al., 2017].

Plusieurs travaux ont cherché à estimer de manière supervisée les propriétés physiques d'un objet, en se basant soit sur des interactions émises par des forces virtuelles [Denil et al., 2017], soit sur des interactions menées par un véritable agent physique [Gouko et al., 2015, Haughton et al., 2022, Kannabiran et al., 2019, Xu et al., 2019].

Haughton et al. [2022] ont pré-entraîné à partir de SVM (*Support Vector Machines*) des classifieurs pour identifier le matériau, la rigidité ou encore la distribution sur l'objet de la force appliquée. Ces modèles pré-entraînés sont ensuite utilisés pour interpréter les observations effectuées par un robot à bras dans un environnement réel.

Denil et al. [2017] et Kannabiran et al. [2019] entraînent un estimateur de propriétés à partir des observations de l'agent qui interagit avec les objets. Nous évoquerons dans la dernière section de ce chapitre la particularité de ces deux approches visant à apprendre simultanément une politique d'interaction.

Détaillée dans la section précédente, la méthode de Xu et al. [2019] établit dans un espace latent R_p une représentation implicite de la dynamique. Les auteurs montrent que, dans cet espace, les propriétés de masse et de frottement, catégorisées chacune en 30, valeurs peuvent facilement être séparées avec un classifieur linéaire entraîné de manière supervisée (voir Figure III.8). Il est possible de déterminer le matériau de l'objet de manière non supervisée en utilisant un algorithme de clusturing tel que t-SNE. Cependant, cette approche suppose que le matériau soit la seule propriété discrète présente dans l'expérience.

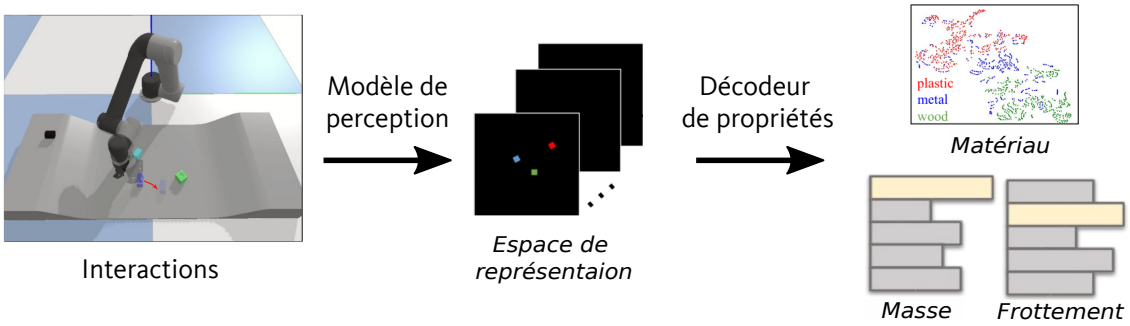


FIGURE III.8 – L'estimation des propriétés dans [Xu et al., 2019] provient d'une prédiction supervisée à partir d'un espace de représentation implicite de la dynamique. Crédit image : [Xu et al., 2019].

Comme nous venons de le voir, différentes méthodes permettent d'identifier les propriétés dynamiques d'un objet suite à des interactions physiques. Seulement, ces approches ne permettent pas de répondre à notre problème puisque la prédiction des propriétés est effectuée de façon supervisée. Ceci nécessite en effet la connaissance des propriétés durant l'entraînement et limite par conséquent les propriétés que l'agent est capable d'estimer. Notre ambition est de pouvoir détecter un ensemble de propriétés sans savoir à l'avance celles qui caractérisent l'objet.

III–2.2 Estimation par apprentissage auto-supervisé

Dans [Wu et al., 2015, 2016], les auteurs estiment de manière non supervisée la masse et le coefficient de frottement d'un objet en observant son glissement sur une pente et ses collisions avec un obstacle en bas de cette pente. Dans [Wu et al., 2016] la méthode est généralisée pour obtenir d'autres types de propriétés (matériau, élasticité au rebondissement, flottaison). Les observations fournies au modèle sont des images issues de séquences vidéos d'objets réels en mouvement dans divers scénarios

physiques (par exemple un objet en positionné en haut d'une rampe). Il n'y a pas d'agent pour créer l'interaction et l'observation acquise est centrée uniquement sur l'objet (voir des exemples d'acquisition sur la Figure III.9). Les auteurs utilisent un modèle physique pour simuler l'environnement et le comportement de l'objet, basé sur des prédictions des propriétés telles que la masse, le frottement, etc. La valeur des propriétés de l'objet est ensuite déterminée en comparant la position et la vitesse de l'objet en simulation avec celles observées dans la séquence réelle, grâce à l'utilisation d'un outil de suivi d'objet.

FIGURE III.9 – Exemples d'observations utilisées dans [Wu et al., 2016] issues de vidéos de divers scénarios : glissement, flottement, chute, étirement. Crédit image : [Wu et al., 2016]

Bien que les propriétés soient extraites sans supervision, la méthode suppose des connaissances a priori sur l'environnement puisque le simulateur intègre des lois physiques liées aux propriétés recherchées. Dans le cas où l'on dispose d'un modèle de l'environnement, de nature physique ou prédictive [de Avila Belbute-Peres et al., 2018, Jaques et al., 2020], capable d'utiliser des actions et des propriétés physiques pour prédire les états futurs des objets, alors il est possible d'apprendre la valeur des propriétés. Cette démarche mise en oeuvre par [Wu et al., 2015, 2016] repose sur la confrontation entre les prédictions et les évolutions des objets observées. Dans les travaux de Wu et al. [2017] et Link et al. [2022] le modèle physique est fourni par simulation. Les propriétés de masse, frottement, position et vitesse de sphères en mouvement dans un environnement 2D sont ainsi inférées. Dans [Kandukuri et al., 2022], les propriétés de masse et de frottement d'un cube sont inférées à partir des équations physiques les reliant à la vitesse et aux positions (estimées par un modèle de perception) atteintes par l'objet après application d'une une force extérieure.

Dans notre approche, nous ne voulons pas fournir une si forte connaissance à priori de la dynamique de l'environnement. A la place, nous essayons d'apprendre implicitement à l'aide de réseaux de neurones les lois physiques qui régissent le comportement de l'objet avec pour objectif de faire émerger leurs propriétés dans un espace de représentation.

Lohmann et al. [2020] se sont libérés des contraintes de connaissance a priori de lois physiques et ont réussi à découvrir la masse et la mobilité d'un objet en auto-supervision et dans un environnement photo-réaliste simulant l'intérieur d'une maison (voir Figure III.10). Cependant, la complexité du problème a été réduite par l'élimination de la présence d'un agent physique et par la discrétisation de la propriété étudiée, ce qui a conduit à l'hypothèse sous-jacente liant cette propriété aux capacités d'action de l'agent. En effet, les auteurs classifient les objets de l'environnement selon 3 catégories de masse en fonction de la réaction de l'objet suite à l'application d'une force virtuelle de faible, moyenne ou forte puissance. En appliquant une force sur un objet, ce dernier va bouger si l'amplitude de la force est suffisamment élevée. Un réseau convolutif (U-Net) apprend ainsi à prédire un indicateur de mobilité de l'objet, l'action à effectuer pour le déplacer et la masse des objets sous la forme d'une carte sémantique. Dans notre méthode, nous ne donnons pas ces connaissances a priori car notre objectif est que l'agent apprenne de manière autonome à sélectionner des actions qui facilitent la reconnaissance d'un ensemble diversifié de propriétés.

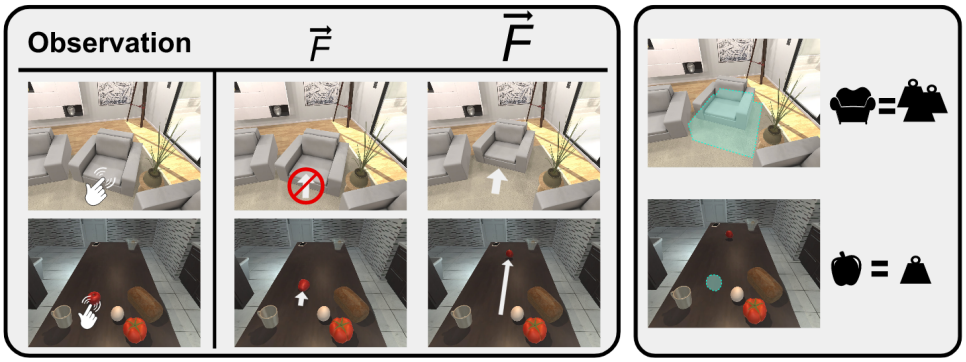


FIGURE III.10 – Détection de la mobilité et catégorie de masse d'objets par auto-supervision. Crédit image : [Lohmann et al., 2020].

Pour estimer un ensemble de propriétés physiques, certaines méthodes cherchent à capter ces caractéristiques dans les composantes d'un vecteur latent issu d'architectures de type auto-encodeur [Lu et al., 2019, Ye et al., 2018, Zheng et al., 2018].

Pour encoder les propriétés, Ye et al. [2018] partent du principe qu'ils peuvent obtenir et utiliser des ensembles de séquences vidéo d'un objet en mouvement où chaque ensemble contient la variation d'une seule propriété de l'objet. Un encodeur prend en entrée une série de 4 images puis condense l'information dans un vecteur à partir

duquel un décodeur doit prédire l'image de l'instant suivant. Le vecteur latent va être subdivisé en plusieurs composantes, chacune ayant pour objectif de contenir des informations spécifiques relatives à une propriété particulière (vitesse, masse, frottement). Pendant la phase d'apprentissage, le modèle fait des prédictions sur l'état futur de la scène en se basant uniquement sur les séquences issues d'un seul de ces ensembles, tout en optimisant uniquement les paramètres du réseau associés à la composante du vecteur latent correspondant à la propriété variant dans l'ensemble sélectionné. Cette méthode s'approche d'un apprentissage supervisé puisqu'il est nécessaire de connaître au préalable la propriété qui varie dans le set de séquences pour fixer certains des paramètres du modèle lors de l'apprentissage. Dans nos expériences, chaque séquence d'interaction est générée par l'interaction avec un objet dont les propriétés on été générées aléatoirement et cachées à l'agent. Par conséquent, nous ne pouvons pas mettre en œuvre cette méthode d'apprentissage spécifique.

Les études de Zheng et al. [2018] et Lu et al. [2019] proposent une méthode permettant de se passer des connaissances a priori sur les propriétés recherchées tout en apprenant à produire un vecteur de propriétés physiques. Dans un environnement à 2 dimensions où le modèle observe des sphères en mouvement, Zheng et al. [2018] ont créé une architecture dite de "perception-prédiction" (voir Figure III.11) pour forcer un vecteur latent à contenir les propriétés physiques des sphères.

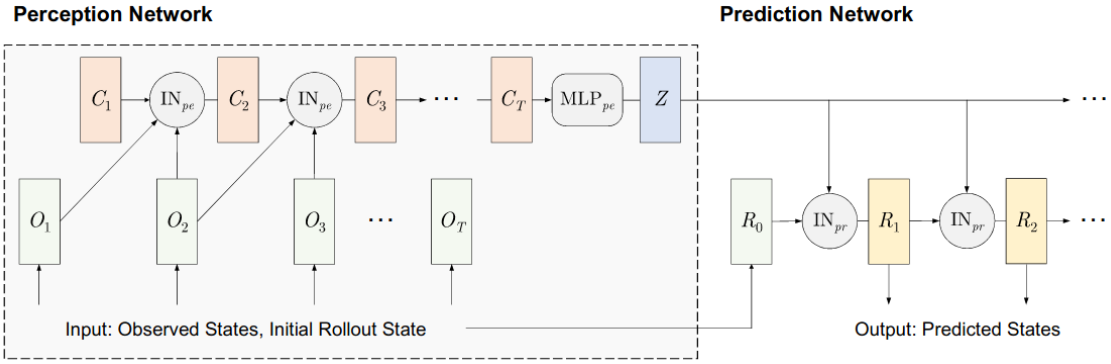


FIGURE III.11 – Architecture de perception-prédiction employée par Zheng et al. [2018] pour forcer un vecteur latent Z à contenir les propriétés dynamiques d'un objet. Crédit image : [Zheng et al., 2018].

La phase de perception consiste à observer, sur plusieurs pas de temps, les sphères ayant été initialisées avec des propriétés tirées aléatoirement dans un espace discret. À un pas de temps t donné, l'observation précédente O_{t-1}, l'observation actuelle O_t et un espace de représentation de la trajectoire C_{t-1} équivalent à une mémoire, sont introduits dans un réseau dense IN [1] pour obtenir un nouvel espace C_t. Une fois la

1. Étant donné que plusieurs sphères interagissent entre elles, IN est une architecture particulière [Battaglia et al., 2016] capable de prendre en compte des relations physiques inter-objets.

séquence de longueur T observée, l'état C_T est introduit dans un réseau entièrement connecté pour diminuer sa dimension et former un vecteur Z. La phase de prédiction consiste à estimer une séquence d'états futurs de l'environnement pour une nouvelle trajectoire avec les propriétés des sphères identiques à celles de la phase de perception. Pour cela, l'état initial R_0 et le vecteur Z sont introduits dans un réseau IN pour estimer l'état suivant R_1. Z et R_1 sont ensuite utilisés pour produire R_2 et ainsi de suite. À partir d'un état statique unique R, il semble a priori impossible de prédire l'état suivant de l'environnement puisqu'il ne contient aucune information sur les propriétés des sphères (vitesse, propriétés intrinsèques). En entraînant le modèle de perception et de prédiction à l'aide d'un même signal issu de la comparaison entre l'état prédit $\hat{R}$ et l'état réel R, Z va être contraint de mémoriser des informations liées aux propriétés des sphères.

Pour vérifier cette intuition, les auteurs réalisent, une fois l'apprentissage terminé, des mesures de corrélations entre les composantes du vecteur Z (auquel a été appliquée une analyse en composantes principales) et les propriétés réelles des sphères (masse, coefficient de restitution, friction). Ceci permet de relier les composantes de Z à des valeurs interprétables. La même approche a été réalisée dans [Lu et al., 2019] pour des observations de type image, provenant de systèmes dynamiques régis par des équations différentielles, et dont l'objectif est d'estimer certaines propriétés de ces équations. Leurs travaux se distinguent par l'ajout d'un terme de régularisation identique à celui utilisé pour l'entraînement d'un auto-encodeur variationnel (ou *VAE* pour *variational auto encoder*). L'intérêt est d'imposer des contraintes de régularisation sur l'espace latent.

La méthode semble particulièrement intéressante pour répondre à notre problématique, mais ne répond pas à l'ensemble de nos objectifs notamment sur le fait que l'environnement n'est pas réaliste et qu'il n'y a pas présence d'un agent pour créer les interactions. Dans le Chapitre V, nous proposerons une adaptation de [Zheng et al., 2018] pour apprendre à identifier les propriétés d'un objet dans le contexte propre de notre étude.

Nous pouvons conclure que l'estimation des propriétés d'un objet sans avoir recours à un apprentissage supervisé nécessite d'imposer de fortes contraintes sur l'environnement et l'agent (modèle physique connu, connaissance parfaite de l'environnement, relations entre les actions et les propriétés, séquences d'interactions sélectionnées, etc.). De plus, il n'existe pas de méthode permettant d'extraire des propriétés sans supervision tout en utilisant un agent physique pour interagir avec l'objet. Seuls Gouko et al. [2017] ont réussi à identifier la forme d'un objet, parmi 3 catégories (cube, triangle, cylindre), à l'aide d'un robot mobile percevant son environnement uniquement au travers de ses propres capteurs. Cependant, l'agent n'est pas doté de moyens d'interaction avec l'objet et se contente d'une perception provenant d'un signal

de 8 capteurs lasers positionnés autour de lui. Les observations acquises par l'agent permettent de constituer des clusters à l'aide d'un algorithme non-supervisé suffisant pour classer l'objet. La méthode ne peut identifier qu'une unique propriété discrète à la fois qui doit en plus être discrète. L'originalité de ces travaux réside dans le fait que l'agent est entraîné à se déplacer pour effectuer des mesures idéales pour classifier la forme. Cette technique de sélection des actions pour réaliser des interactions idéales avec les objets est le sujet abordé dans la prochaine section.

III–3 Méthode de sélection des actions

Jusqu'à présent, nous avons exploré diverses méthodes pour représenter les propriétés d'un objet, que ce soit de manière implicite ou explicite. Parmi les travaux qui mettent en place un agent physique, une majorité sélectionne l'action dans un espace prédéfini. Très peu cherchent à apprendre une stratégie comportementale adaptée à la découverte de propriétés. Comme nous l'avons souligné dans le chapitre introductif, cette capacité à apprendre une telle stratégie pourrait significativement améliorer la qualité de la représentation des objets.

III–3.1 Utilisation d'une politique déterministe

Dans les expériences d'Agrawal et al. [2016], Finn et al. [2016], la dynamique de l'objet est modélisée de manière implicite par des actions de poussée. Le choix s'effectue de manière aléatoire pour définir le point d'impact de l'effecteur, l'orientation de la poussée et la longueur de cette poussée. L'action choisie n'est donc pas forcément optimale pour découvrir un comportement de l'objet qui n'aurait pas encore été observé par l'agent. L'agent peut en effet manquer des points de contact qui changeraient totalement le comportement de l'objet s'il étaient utilisés pour une poussée. Dans [Li et al., 2018], le déplacement de l'effecteur de l'agent est choisi de sorte à rapprocher l'objet d'un état cible. Un ensemble de déplacements rectilignes est aléatoirement généré à partir de l'image de la scène. Chaque action est passée dans le modèle direct (en cours d'apprentissage) qui génère en retour une estimation de l'état de l'objet en suivant cette action. Le déplacement choisi est celui pour lequel l'état prédit est le plus proche de l'état cible souhaité.

Un des points forts des travaux de Xu et al. [2019] est l'utilisation de 2 types d'interactions qui sont la poussée avec un effecteur et la collision avec un obstacle, en positionnant l'objet au sommet d'une rampe comme illustré sur la Figure III.12. Le choix du type d'action est pseudo-aléatoire, favorisant le type d'action le moins fréquemment utilisé durant la séquence d'interactions. La manière dont la poussée est effectuée (force, direction, point de contact) ou le positionnement de l'objet sur la

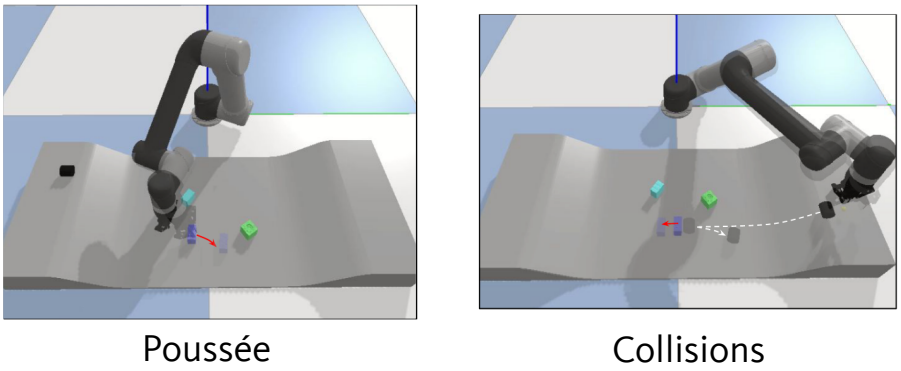


FIGURE III.12 – Les deux types d'interactions effectuées dans [Xu et al., 2019], la poussée et la collision après glissade sur un plan incliné. Crédit image : [Xu et al., 2019].

rampe sont choisis aléatoirement dans un ensemble de possibilités. Kandukuri et al. [2022] agissent également sur l'objet par la poussée et le glissement sur une rampe en appliquant des forces virtuelles dont la zone d'application et l'amplitude sont aléatoirement choisies.

La méthode de sélection d'une action dans un espace prédéfini présente 2 défauts majeurs, communs à l'ensemble des travaux que nous venons de détailler. D'une part, l'exécution de l'action demande une connaissance parfaite de l'environnement, et notamment la position et l'orientation de l'objet et de l'effecteur, mais aussi des paramètres des éléments particuliers comme l'emplacement et la hauteur de la rampe. L'agent n'est donc pas en mesure d'évoluer dans un environnement inconnu comme le veut notre objectif . Il n'est pas non plus en mesure de s'adapter à d'éventuelles variations de l'environnement et perturbations externes. D'autre part, l'agent ne sélectionne pas ses actions de manière stratégique pour recueillir des observations qui seraient particulièrement révélatrices des propriétés physiques de l'objet.

Haughton et al. [2022] montrent l'importance d'actions judicieusement placées, là où le modèle de prédiction est peu performant, pour améliorer la qualité d'une segmentation de la scène. Une mesure d'entropie croisée calculée sur la distribution sémantique renvoyée par le modèle indique à l'agent une zone sur laquelle l'incertitude est élevée et où une action devrait être effectuée. Dans [Lohmann et al., 2020], l'amplitude de la force à appliquer sur l'objet est indiquée par le modèle de perception chargé d'identifier la catégorie de masse de l'objet. Si l'objet est considéré comme lourd, alors une force d'amplitude moyenne est appliquée pour s'assurer que l'objet n'est pas plus léger que ce que le modèle suppose.

III–3.2 Apprentissage de la politique

L'apprentissage par renforcement offre une solution de contrôle efficace pour choisir les actions en vue d'obtenir des observations qui apportent plus d'information sur

les caractéristiques de l'objet. Gouko et al. [2017] réalisent ainsi une identification non-supervisée de la forme d'un objet grâce à une perception active de l'environnement issue d'un comportement appris par RL. Les observations acquises sur une trajectoire sont données en entrée d'un réseau de neurones qui retourne un espace latent dans lequel un algorithme de clustering est appliqué. Plus le vecteur de caractéristiques se distingue des autres clusters, plus l'agent recevra une récompense élevée. Ainsi, l'agent va apprendre à se déplacer pour obtenir des observations qui vont permettre de distinguer clairement la forme des objets. Cette méthode basée uniquement sur la vision (pas d'interaction physique avec l'objet) n'est valide que dans le cas où une seule propriété discrète différencie les objets. Dans nos travaux, notre objectif est de permettre à un agent d'identifier un ensemble de propriétés continues, ou discrètes, tout en s'assurant que les observations acquises ne se focalisent pas exclusivement sur l'une de ces propriétés.

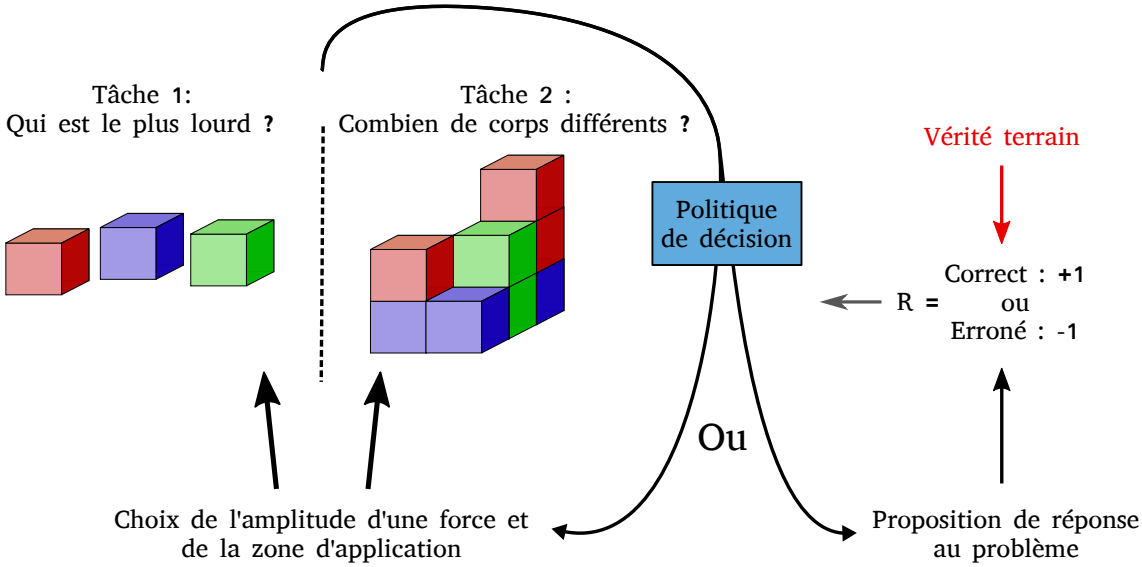


FIGURE III.13 – Illustration des tâches et du processus d'interaction mis en place dans [Denil et al., 2017].

La méthode d'apprentissage présentée dans [Denil et al., 2017, Kannabiran et al., 2019] pourrait être en mesure de faciliter l'identification de propriétés à la fois discrètes et continues. Dans divers scénarios physiques où l'objectif peut varier de l'identification du bloc le plus lourd à la détermination du nombre de corps rigides distincts, (voir Figure III.13), Denil et al. [2017] utilisent l'apprentissage par renforcement pour sélectionner l'amplitude d'une force et l'objet sur lequel l'appliquer pour acquérir ainsi de l'information permettant de répondre à la problématique. Pour cela, l'agent interagit autant qu'il veut avec les objets et, lorsque la politique estime avoir la réponse à la problématique, elle indique son choix au lieu d'une action. Il obtient une récompense positive si la réponse est correcte et une récompense négative s'il donne la mauvaise

réponse, ou si le temps de réponse est dépassé.

Dans [Kannabiran et al., 2019], l'estimation de la propriété de masse d'un objet articulé est directement couplée à la stratégie comportementale qui a généré les observations fournies à l'estimateur (voir Figure III.14). L'erreur de prédiction est utile de deux manières : elle guide l'optimisation des paramètres du modèle d'estimation et elle sert également de signal de récompense pour entraîner la politique. Cette dernière est ainsi encouragée à générer des observations qui contribuent à minimiser l'erreur de prédiction. Le principal inconvénient de ces deux travaux [Denil et al., 2017, Kannabiran et al., 2019] est que, pour générer le signal de récompense, il est nécessaire de disposer des valeurs réelles des propriétés physiques des objets. Le couplage entre la politique et l'estimateur de propriétés semble alors impossible pour répondre à notre objectif. C'est pourquoi, dans cette thèse, nous proposons une méthode où l'apprentissage de la politique comportementale est découplé de l'estimateur des propriétés. Ces deux aspects sont détaillés respectivement dans les Chapitres IV et V.

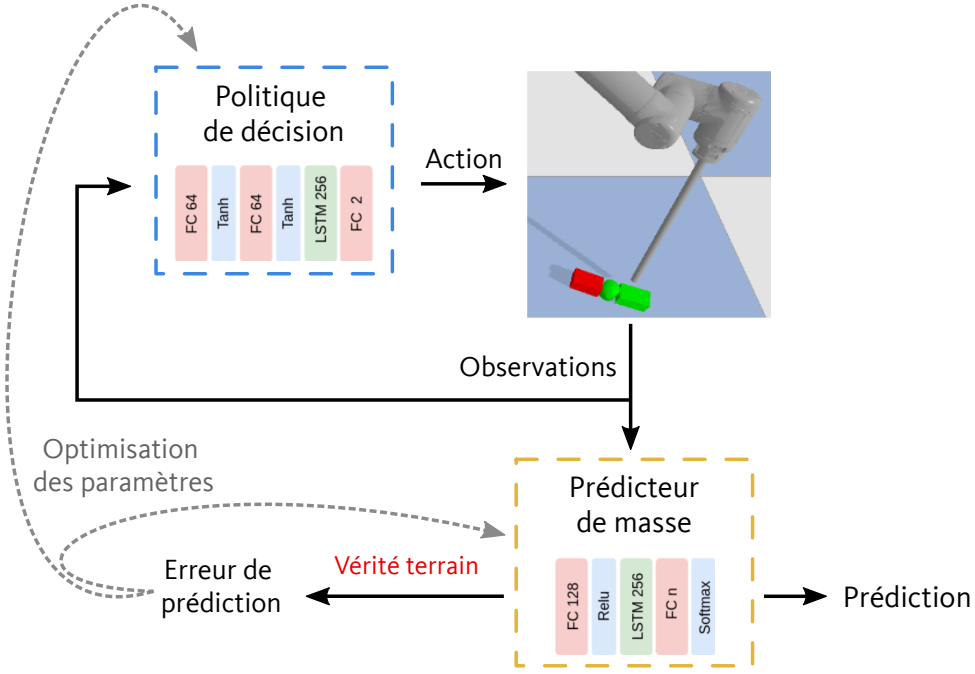


FIGURE III.14 – Couplage du prédicteur de propriétés à la politique pour l'apprentissage des interactions. Crédit images : [Kannabiran et al., 2019].

CHAPITRE IV

Apprentissage d'une politique d'interaction

Sommaire

IV–1 Introduction

Dans ce chapitre, nous cherchons à apprendre une stratégie d'interaction qui permette de recueillir des observations contenant l'information des caractéristiques physiques et dynamiques d'un objet. La principale contrainte est d'effectuer ces interactions sans connaissances ni sur l'objet ni même sur les propriétés qui sont recherchées. Une première question se pose : comment apprendre à interagir sans récompenser l'agent par rapport à un objectif précis ?

Nous réalisons l'hypothèse que, pour obtenir des interactions avec l'objet dont un ensemble de ses propriétés physiques peut être déduit, il suffit de susciter un maximum de variations dans ses réponses (voir Figure IV.1). En effet, le comportement des objets peut varier dû à de nombreux facteurs. Ils peuvent être très glissants, ou ne pas bouger à cause de leur poids. Leur forme ou surface peut faire qu'ils soient mobiles dans une direction, mais pas dans l'autre. C'est probablement en provoquant une diversité de réponses que l'agent pourra observer de telles caractéristiques. Nous proposons alors d'évaluer cette diversité au travers d'une mesure de l'entropie de la distribution des états traversés par l'objet, et d'utiliser cette mesure comme signal de motivation intrinsèque pour entraîner la politique sans supervision. Après une description de notre méthode générale pour répondre à la première problématique, nous l'appliquons au cas particulier d'un robot mobile pouvant interagir uniquement par la poussée. Cette expérience est réalisée en simulation dans un environnement que nous allons décrire dans ce chapitre et qui sera réutilisé pour toutes les expériences de cette thèse.

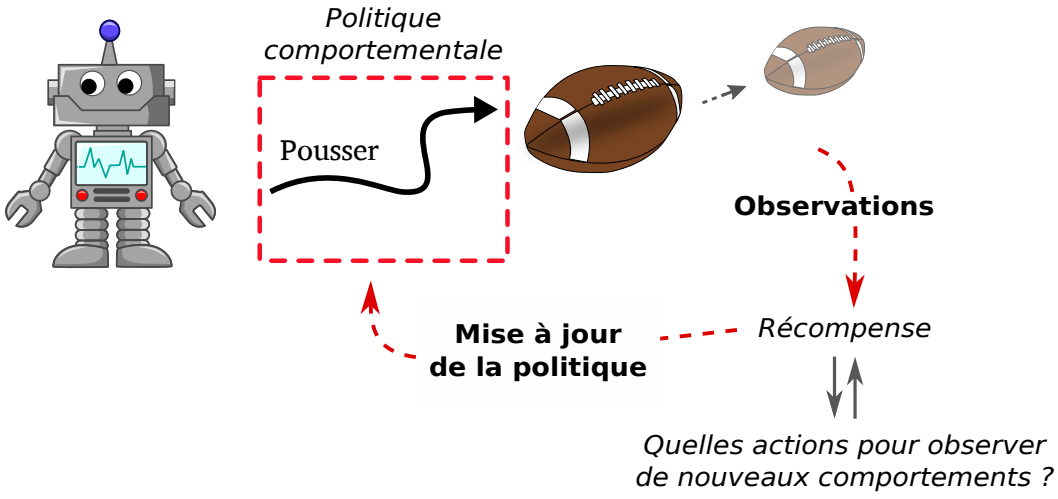


FIGURE IV.1 – Description de la phase d'apprentissage de la politique.

Une fois cet agent entraîné, une seconde question se pose : comment savoir si le comportement obtenu permet de constituer un ensemble d'observations suffisamment informatif pour établir une représentation des propriétés de l'objet lors de la seconde phase (Chapitre V) ?

Une possibilité est d'évaluer la totalité du processus qui consiste en l'exploration de l'objet suivie de l'extraction des propriétés à partir d'une représentation, en affirmant qu'une bonne extraction ne peut résulter que d'une bonne exploration. Un défaut de cette approche est qu'il est impossible de savoir si une mauvaise extraction des propriétés résulte d'un problème d'exploration ou d'un mauvais modèle pour l'extraction. Nous allons par conséquent séparer les deux phases : interaction puis extraction des propriétés. Dans cette partie seulement, nous proposons d'évaluer la politique par sa capacité à explorer et à générer un jeu de données tel que des propriétés préalablement choisies puissent être inférées à l'aide d'un modèle de prédiction entraîné par une approche supervisée. En comparant la performance de ce modèle supervisé selon différentes stratégies d'interaction, nous souhaitons montrer que la politique obtenue avec notre méthode offre des observations plus informatives pour l'identification des propriétés d'objet. Pour cela, nous testons l'agent en faisant varier divers paramètres comme l'ajout de bruit dans l'espace d'observation, le nombre d'exemples d'entraînement et le temps d'interaction avec l'objet.

IV–2 Apprentissage auto-supervisé de la politique par maximisation d'une mesure d'entropie

Considérons un environnement dans lequel se trouvent un objet à explorer et un agent capable d'interagir et de percevoir son environnement par divers moyens. L'objet est animé d'un comportement qui est lié directement à des propriétés comme sa forme, sa masse et la répartition de celle-ci, son matériau, les coefficients de frottement, etc. L'agent n'a cependant à cet instant aucune connaissance a priori de ces propriétés physiques. Son objectif est de susciter des mouvements de l'objet dont l'observation suffit à établir sans aide extérieure une compréhension de la dynamique de l'objet.

IV–2.1 Formalisation du problème

Étant donné un objet caractérisé par un ensemble $P = \{p_1, p_2, ..., p_l\}$ de propriétés p_i continues ou discrètes, il existe une fonction de transition $\mathcal{T}$ prenant en entrée l'état de l'objet et ses propriétés, et donnant en sortie la distribution d'états suivants de l'objet en fonction des actions de l'agent. Sur la base de ses capteurs, l'agent n'observe pas les propriétés de l'objet mais seulement son état. L'environnement ou l'objet ne fournissent d'autre part aucune récompense extrinsèque à l'agent. Par conséquent, nous modélisons le problème générique à résoudre comme l'apprentissage d'une politique

dans un POMDP sans récompense :

$$M = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{O}) \tag{IV.1}$$

où $\mathcal{O}$ correspond à l'espace des observations partielles $\mathbf{o}$ de l'état de l'environnement $\mathbf{s}$ que l'agent mesure au travers ses capteurs (lidar, mesure de distances et vitesses relative, information de contact, ...).

IV–2.2 Approche générale proposée

Pour résoudre ce POMDP sans récompense, nous utilisons un signal de récompense intrinsèque qui repose sur l'idée suivante : l'observation de comportements de l'objet les plus variés possibles est suffisante pour permettre l'identification de ses propriétés. Autrement dit, nous cherchons à ce que l'agent soit motivé intrinsèquement par le fait de rassembler des mesures relatives à l'objet suffisamment éloignées les une des autres après diverses série d'actions.

Suivant l'introduction de ce chapitre, les expériences que nous présenteront dans les sections suivantes s'organisent en deux phases : apprentissage d'une politique d'interactions sans accès aux propriétés des objets, puis validation des politiques apprises par leurs performances en constitution de jeux de données pour l'apprentissage supervisé de prédicteurs de ces propriétés.

Phase d'apprentissage (voir Figure IV.2) : Un épisode commence avec l'initialisation de l'agent et d'un objet aux propriétés P choisies aléatoirement et non observées. A chaque pas de temps $t \in [0, T]$ d'une trajectoire τ de longueur T, l'agent observe son environnement via les observations $\mathbf{o}_t \in \mathcal{O}$ puis effectue une action $\mathbf{a}_t \in \mathcal{A}$. Pour chaque pas de temps, l'agent va également effectuer des acquisitions $\mathbf{s}_t^{\text{ent}} \in \mathcal{S}^{\text{ent}}$ où $\mathcal{S}^{\text{ent}}$ est un sous-ensemble de $\mathcal{O}$. $\mathbf{s}_t^{\text{ent}}$ a pour but de contenir un ensemble de composantes relatives à la dynamique de l'objet et mesuré par les capteurs de l'agent, comme la position et la vitesse de l'objet. En suivant une politique $\pi_{\boldsymbol{\theta}}$ de paramètres $\boldsymbol{\theta}$, l'agent va rassembler un ensemble d'échantillons $\mathbf{S}^{\text{ent}} = \{\mathbf{s}_0^{\text{ent}}, \mathbf{s}_1^{\text{ent}}, ..., \mathbf{s}_T^{\text{ent}}\}$. Nous définissons ainsi $d_T^{\pi_{\boldsymbol{\theta}}}(\mathbf{S}^{\text{ent}})$ comme la distribution des observations $\mathbf{s}_t^{\text{ent}}$ induite par la politique $\pi_{\boldsymbol{\theta}}$.

L'objectif de l'agent est de maximiser l'entropie de Shannon $\mathcal{H}$ de la distribution $d_T^{\pi_{\boldsymbol{\theta}}}(\mathbf{S}^{\text{ent}})$ et faire ainsi en sorte d'observer le plus de différences dans l'ensemble d'acquisitions $\mathbf{S}^{\text{ent}}$. Dans ce contexte, la politique optimale π^* est obtenue en résolvant le problème d'optimisation suivant :

$$\max_{\boldsymbol{\theta}} \mathcal{H}(d_T^{\pi_{\boldsymbol{\theta}}}(\mathbf{S}^{\text{ent}})) \tag{IV.2}$$

Le résultat de cette entropie est utilisé comme récompense intrinsèque r^{int} pour mettre à jour les paramètres $\boldsymbol{\theta}$ de la fonction paramétrique $\pi_{\boldsymbol{\theta}}$ qui modélise la politique.

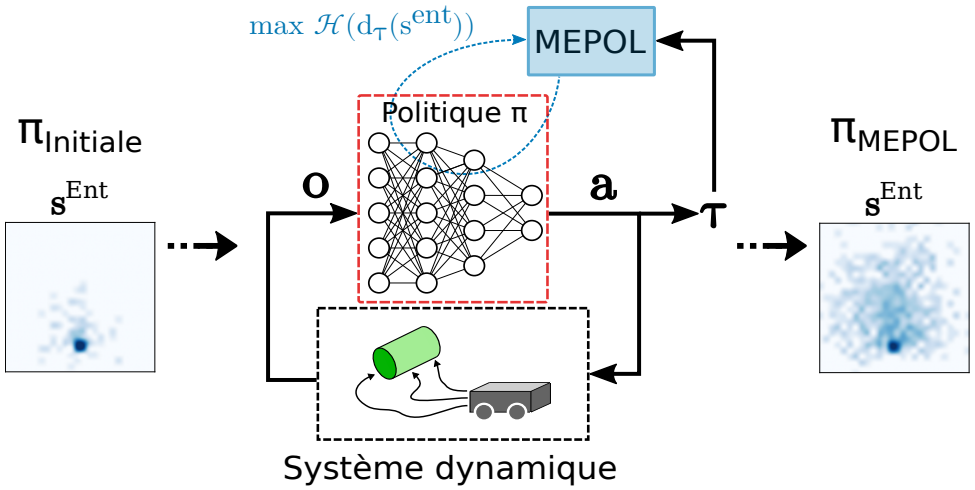


FIGURE IV.2 – Illustration de la méthode d'apprentissage de la politique d'interaction entrainée à l'aide de MEPOL chargé de maximiser l'entropie sur $\mathcal{S}^{\text{ent}}$ à partir d'une politique aléatoire π_{initiale}.

La résolution de l'Eq. (IV.2), est effectuée grâce à l'un des algorithmes de maximisation d'entropie présenté précédemment dans le Chapitre II–2.3.c. Ici, nous avons fait le choix d'utiliser MEPOL [Mutti et al., 2020], dont le fonctionnement est donné dans la prochaine sous-section. La politique obtenue après avoir réalisé l'entraînement est illustré sur la Figure IV.2 et est désigné par π_{MEPOL}.

Phase d'évaluation : Cette phase a pour but de valider que la politique apprise produit des observations qui contiennent les informations nécessaires à l'inférence des propriétés physiques de l'objet. L'agent, équipé de la politique π_{MEPOL}, effectue un nombre N de trajectoires face à des objets de propriétés p_i aléatoires pouvant avoir été rencontrés ou non durant l'entraînement. Il va ainsi constituer une base de données D où chaque trajectoire $\tau^k, k \in [1, K]$ est une succession d'observations $\mathbf{x}_t^k$ acquises à pas de temps régulier $t \in [0, T]$. Les observations $\mathbf{o}_t^k$ destinées à la politique peuvent être différentes des observations $\mathbf{x}_t^k$ destinées à l'identification des propriétés.

Un modèle de perception prend comme entrée l'ensemble des observations $X = \{\mathbf{x}_0^k, \mathbf{x}_1^k, ..., \mathbf{x}_T^k\}$ puis effectue une prédiction d'un ensemble de propriétés de P. Nous supposons alors à cet instant connaître la vérité terrain des propriétés p_i ce qui permet de calculer l'erreur de prédiction et d'entraîner le modèle. Cette phase d'évaluation a uniquement pour but de s'assurer qu'il est possible d'extraire de la trajectoire de l'agent une meilleure représentation de l'objet permettant la prédiction de propriétés. Si cette tâche supervisée montre de mauvaises performances, alors il ne semble pas pertinent de passer à la phase 2 dont l'objectif est de réaliser le même processus sans connaissance des vérités terrain. C'est aussi un moyen de s'assurer qu'une politique maximisant l'entropie est plus performante que d'autres politiques d'interaction.

IV–2.3 Résolution via MEPOL

Le choix d'utilisation de MEPOL pour résoudre l'Eq. (IV.2) est motivé par plusieurs facteurs. D'une part, les mesures de l'agent et ses actions doivent pouvoir être continues. D'autre part, nous souhaitons que $\mathbf{s}_t^{\text{ent}}$ soit de faible dimension pour ne pas complexifier d'avantage le problème. Il n'est pas utile de valider la méthode sur des mesures complexes comme celles des images naturelles si elle ne fonctionne pas sur un espace simplifié. En restant dans un espace de faible dimension, la puissance de calcul nécessaire et le temps de calcul ne sera que plus faible. De plus, en considérant une image, l'information n'est plus centrée sur un seul objet mais sur toute la scène observée (pouvant présenter d'autres objets, des obstacles, ...). Cela nécessiterait d'utiliser d'autres outils de perception pour isoler les objets d'intérêt et revenir à une étude individuelle. L'idée est donc de se baser sur des mesures $\mathbf{s}_t^{\text{ent}}$ simples uniquement effectuées par les capteurs de l'agent, comme des positions relatives ou des vitesses relatives, et ce même si les observations $\mathbf{o}_t$ de l'agent peuvent être de grande dimension comme des images. Nous avons donc moins d'intérêt à utiliser des modèles tels que ProtoRL, APT ou RE3 listés dans le tableau II.1, qui eux maximisent l'entropie sur des mesures $\mathbf{s}^{\text{ent}}$ de très grandes dimensions. Enfin, tous les algorithmes de maximisation d'entropie ne permettent pas d'être aisément modifié. Nous nous sommes donc assurés de pouvoir adapter l'algorithme sélectionné à notre problème. Ces raisons nous ont poussés à choisir MEPOL qui restera le seul algorithme utilisé durant ces travaux.

MEPOL ne calcule pas directement l'entropie de la distribution donnée par la politique π, ce qui nécessiterait de connaître le modèle de probabilité de transition $\mathcal{T}$ ou encore le modèle de densité $d_T^{\pi_\theta}$. A la place, il est effectué N épisodes de longueur T en suivant la politique π puis, calculé une estimation de l'entropie à partir des réalisations $\{\mathbf{s_i}^{\text{ent}}\}_{i=0}^{N'=T\times N}$ acquises lors de ces trajectoires. Cette estimation repose sur la formule de Singh et al. [2003] basée sur une approche particulaire des k_{nn} plus proches voisins (k-NN) qui, appliquée à notre cas, donne :

$$\hat{\mathcal{H}}_{k_{nn}}(d_{N'}^{\pi_\theta}) = -\frac{1}{N'}\sum_{i=1}^{N'} \ln\Big(\frac{k_{nn}}{T V_i^{k_{nn}}}\Big) + \ln k_{nn} - \Psi(k_{nn}) \tag{IV.3}$$

où Ψ est la fonction digamma[1], et $V_i^{k_{nn}}$ le volume de l'hypersphère de rayon $R_i = \max_{j\in[0,k_{nn}]}(||\mathbf{s}_i^{\text{ent}} - \mathbf{s}_{i,j}^{\text{ent}}||)$ correspondant à la distance euclidienne la plus grande entre la réalisation $\mathbf{s}_i^{\text{ent}}$ et ses k_{nn} plus proches voisins $\mathbf{s}_{i,j}^{\text{ent}}$, tel que :

1. Fonction disponible dans des librairies telle que scipy

$$V_i^{k_{nn}} = \frac{R_i^m \pi^{\frac{m}{2}}}{\Gamma(\frac{m}{2}+1)} \tag{IV.4}$$

où m est la dimension de $\mathbf{s}^{\text{ent}}$ et Γ la fonction gamma [2].

La formule de l'Eq. (IV.4) ne peut pas être utilisée pour entraîner la politique $\pi_{\boldsymbol{\theta}}$. En effet, dans cette estimation de l'entropie, les réalisations $\{\mathbf{s}_i^{\text{ent}}\}_{i=0}^{N'}$ ne sont pas reliées aux paramètres $\boldsymbol{\theta}$ de la politique $\pi_{\boldsymbol{\theta}}$ alors que les actions ont bien une influence sur les réalisations obtenues. Il n'est donc pas possible de mettre à jour la politique et de propager le gradient dans les paramètres $\boldsymbol{\theta}$ pour modifier le comportement de l'agent et maximiser l'entropie. À la place, il est utilisé une autre forme d'estimation de l'entropie introduite par Ajgl and Šimandl [2011] et qui utilise la distribution de 2 politiques pour effectuer un échantillonnage préférentiel (ou *importance sampling*) :

$$\hat{\mathcal{H}}_{k_{nn}}(d_{N'}^{\pi_{\boldsymbol{\theta}'}} | d_{N'}^{\pi_{\boldsymbol{\theta}}}) = -\sum_{i=1}^{N'} \frac{W_i}{k_{nn}} \ln\Big(\frac{W_i}{V_i^{k_{nn}}}\Big) + \ln k_{nn} - \Psi(k_{nn}) \tag{IV.5}$$

avec :

$$W_i = \sum_{j \in \mathcal{N}_i^{k_{nn}}} w_j \tag{IV.6}$$

où $\mathcal{N}_i^{k_{nn}}$ est le set d'indices des k-NN de $\mathbf{s}_i^{\text{ent}}$ et les w_j les poids d'importances normalisés des échantillons $\mathbf{s}_i^{\text{ent}}$ qui sont définis comme :

$$w_j = \frac{\overline{w}_j}{\sum_{n=0}^{N'} \overline{w}_n}, \qquad \overline{w}_j = \prod_{z=0}^{t} \frac{\pi_{\boldsymbol{\theta}'}(\mathbf{a}_j^z | \mathbf{o}_j^z)}{\pi_{\boldsymbol{\theta}}(\mathbf{a}_j^z | \mathbf{o}_j^z)} \tag{IV.7}$$

où $\prod_{z=0}^{t} \pi(\mathbf{a}_j^z | \mathbf{o}_j^z)$ est la portion de la trajectoire $\mathbf{o}_j^t = (\mathbf{o}_0, \mathbf{o}_1, ..., \mathbf{o}_t)$ qui a menée à $\mathbf{s}_j^{\text{ent}}$.

L'Eq. (IV.5) relie cette fois-ci directement les réalisations $\{\mathbf{s}_i^{\text{ent}}\}_{i=0}^{N'}$ aux actions de l'agent, ce qui permet de mettre à jour les paramètres $\boldsymbol{\theta}$ en suivant l'algorithme dont le pseudo-code est donné à l'Algorithme IV.1. L'exécution de cet algorithme, nécessite également le calcul d'une dissimilarité mesurée à partir d'une estimation de la divergence de Kullback-Leibler entre $d_{N'}^{\pi_{\boldsymbol{\theta}'}}$ et $d_{N'}^{\pi_{\boldsymbol{\theta}}}$ est donnée par :

$$\hat{\mathcal{D}}_{\text{KL}\ k_{nn}}(d_{N'}^{\pi_{\boldsymbol{\theta}'}} || d_{N'}^{\pi_{\boldsymbol{\theta}}}) = \frac{1}{N'} \sum_{i=1}^{N'} \ln\Big(\frac{\frac{k_{nn}}{N'}}{W_i}\Big) \tag{IV.8}$$

Le principe de MEPOL est d'effectuer un certain nombre d'epoch où, à chaque fois, il est effectué les itérations suivantes :

1. Génération d'un nombre N de trajectoires de longueur T avec $\pi_{\boldsymbol{\theta}}$,

2. Fonction également disponible dans des librairies telle que scipy

2. Initialisation d'une seconde politique $\pi_{\theta'}$ avec des paramètres $\boldsymbol{\theta}' = \boldsymbol{\theta}$,
3. Maximisation de l'Eq. (IV.5) par une descente de gradient sur les paramètres $\boldsymbol{\theta}'$ de la politique $\pi_{\theta'}$, tout en laissant fixes ceux de la politique $\pi_{\boldsymbol{\theta}}$,
4. Calcul de la dissimilarité entre les distributions de $\pi_{\theta'}$ et $\pi_{\boldsymbol{\theta}}$. Si la valeur résultante est inférieure à la valeur δ alors l'itération des gradients continue,
5. Une fois ce coefficient δ (ou un nombre limite fixé à l'avance) dépassé, $\pi_{\boldsymbol{\theta}}$ devient $\pi_{\theta'}$ puis un nouvel epoch débute.

Algorithme IV.1 Processus d'apprentissage de la politique mis en place dans MEPOL.

Pre-condition: Exploration d'horizon T, nombre de trajectoires d'échantillonnage N, seuil de confiance δ, learning rate α, nombre de plus proches voisins k_{nn}.
Initialiser une politique stochastique π avec des paramètres $\boldsymbol{\theta}$.
pour epoch $= 1, 2, \ldots$ jusqu'à convergence **faire**
Générer un ensemble de N trajectoires τ de longueur T avec $\pi_{\boldsymbol{\theta}}$ pour constituer un set d'échantillons $\mathbf{S}^{\text{ent}} = \{\mathbf{s}_i^{\text{ent}}\}_{i=0}^{N'}$.
initialiser $h = 0$ et $\boldsymbol{\theta}_h = \boldsymbol{\theta}$
tant que $\hat{\mathcal{D}}_{\text{KL}\, k_{nn}}(d_{N'}^{\pi_{\boldsymbol{\theta}_h}}(\mathbf{S}^{\text{ent}}) || d_{N'}^{\pi_{\boldsymbol{\theta}_0}}(\mathbf{S}^{\text{ent}})) \leq \delta$ **faire**
Effectuer la descente de gradient :
$\boldsymbol{\theta}_{h+1} = \boldsymbol{\theta}_h + \alpha \nabla_{\boldsymbol{\theta}_h} \hat{\mathcal{H}}_{k_{nn}}(d_{N'}^{\pi_{\boldsymbol{\theta}_h}}(\mathbf{S}^{\text{ent}}) | d_{N'}^{\pi_{\boldsymbol{\theta}_0}}(\mathbf{S}^{\text{ent}}))$
$h \leftarrow h + 1$
fin tant que
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_h$
fin pour
Sortie: Politique optimale π_{MEPOL}.

IV–2.4 Limites et modifications de MEPOL

Tel que proposé, l'algorithme original de MEPOL n'est pas adapté à notre formalisation. En effet, MEPOL mesure l'entropie sur un batch de trajectoires issues du même MDP. Or, dans notre cas, chaque trajectoire correspond à un nouveau MDP. L'entropie $\hat{\mathcal{H}}_k(d_{N'}^{\pi_{\boldsymbol{\theta}'}} | d_{N'}^{\pi_{\boldsymbol{\theta}}})$ estimée n'est donc plus valable dans nos conditions ce qui peut entraîner une instabilité lors de l'apprentissage, en plus de comportements qui ne mènent pas du tout à de la diversité selon l'objet.

Nous avons donc modifié l'algorithme pour que l'entropie soit estimée sur une seule trajectoire et non plus sur un batch. En prenant un horizon T suffisamment élevé, nous faisons l'hypothèse que l'estimation $\hat{\mathcal{H}}_k(d_T^{\pi_{\boldsymbol{\theta}'}} | d_T^{\pi_{\boldsymbol{\theta}}})$ approchera la distribution réelle de $\pi_{\boldsymbol{\theta}'}$. Chaque objet peut avoir un comportement bien différent selon ses propriétés p_i. Par exemple, un objet peut être très glissant le rendant peu maniable ou encore très lourd l'empêchant de bouger. Dans ces conditions la diversité peut-être difficile à obtenir ce qui rend l'apprentissage instable. Plutôt que de mettre à jour les

Algorithme IV.2 Variante de MEPOL adapté au cas d'un POMDP.

Pre-condition: Exploration d'horizon T, nombre d'échantillons N, seuil de confiance δ, learning rate α, plus proches voisins k_{nn}.
Initialiser une politique stochastique π avec des paramètres $\boldsymbol{\theta}$.
pour epoch $= 1, 2, \ldots$ jusqu'à convergence **faire**
Générer un ensemble de N trajectoires τ de longueur T avec $\pi_{\boldsymbol{\theta}}$ pour constituer un set d'échantillons $\mathbf{S}^{\text{ent}} = \{\boldsymbol{s}_i^{\text{ent}}\}_{i=0}^{T\times N}$.
initialiser $h = 0$ et $\boldsymbol{\theta}_h = \boldsymbol{\theta}$
tant que $\sum_n^N \frac{1}{N} \hat{\mathcal{D}}_{\text{KL}\, k_{nn}}(d_{T,n}^{\pi_{\boldsymbol{\theta}_h}}(\mathbf{S}_n^{\text{ent}}) || d_{T,n}^{\pi_{\boldsymbol{\theta}_0}}(\mathbf{S}_n^{\text{ent}})) \leq \delta$ **faire**
Calcule de la moyenne des entropies :
$\overline{\mathcal{H}_{k_{nn}}} \leftarrow \sum_n^N \frac{1}{N} \hat{\mathcal{H}}_{k_{nn}}(d_{T,n}^{\pi_{\boldsymbol{\theta}_h}}(\mathbf{S}_n^{\text{ent}}) | d_{T,n}^{\pi_{\boldsymbol{\theta}_0}}(\mathbf{S}_n^{\text{ent}}))$
Effectuer la descente de gradient :
$\boldsymbol{\theta}_{h+1} = \boldsymbol{\theta}_h + \alpha \nabla_{\boldsymbol{\theta}_h} \overline{\mathcal{H}_{k_{nn}}}$
$h \leftarrow h + 1$
fin tant que
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_h$
fin pour
Sortie: Une politique π_{MEPOL}.

paramètres $\boldsymbol{\theta}$ à partir d'une trajectoire unique, nous estimons l'entropie de plusieurs trajectoires avant d'en calculer la moyenne. C'est cette moyenne qui sera ensuite utilisée lors de la descente de gradient. La modification s'applique également à l'estimation de $\hat{\mathcal{D}}_{KL}(d_T^{\pi_{\boldsymbol{\theta}'}} | d_T^{\pi_{\boldsymbol{\theta}}})$. Le pseudo-code de cette version alternative de MEPOL est donné à l'Algorithme IV.2.

IV–3 Application de la méthode au cas d'un robot mobile interagissant uniquement par la poussée

Notre objectif est de valider la méthode proposée sur un dispositif expérimental pour aboutir à des résultats interprétables. Pour ce faire, nous proposons de valider chaque étape du processus d'apprentissage dans un environnement simulé avec PyBullet [Coumans and Bai, 2016–2022]. Nous exposons dans une première sous-section l'environnement de simulation conçu et comprenant l'agent, les objets ainsi que la scène dans laquelle l'agent va évoluer. La seconde sous-section détaille l'application de la méthode à ce cas particulier. Un objectif de valorisation à long terme de ce travail étant le transfert des expériences simulées sur robot réel, nous nous sommes restreints dans le cadre de cette thèse à un agent modélisé selon le robot Wifibot décrit ci-après, qui ne peut interagir avec des objets que via la poussée. Ce choix d'utiliser ce robot mobile et ce mode d'interaction particulier a été largement influencé par la disponibilité au laboratoire du modèle virtuel et réel du robot.

IV–3.1 Environnement de simulation

IV–3.1.a L'agent

L'agent est un robot mobile à 4 roues motrices appelé WifiBot [3] pouvant embarquer divers moyens de calculs (CPU et GPU) et capteurs (Camera RGB-D, Lidar, centrale inertielle). Nous disposons d'un modèle URDF [4] (Unified Robot Description Format), conçu par l'unité robotique de l'ONERA. La Figure IV.3 donne un aperçu de cet agent modélisé dans le simulateur PyBullet.

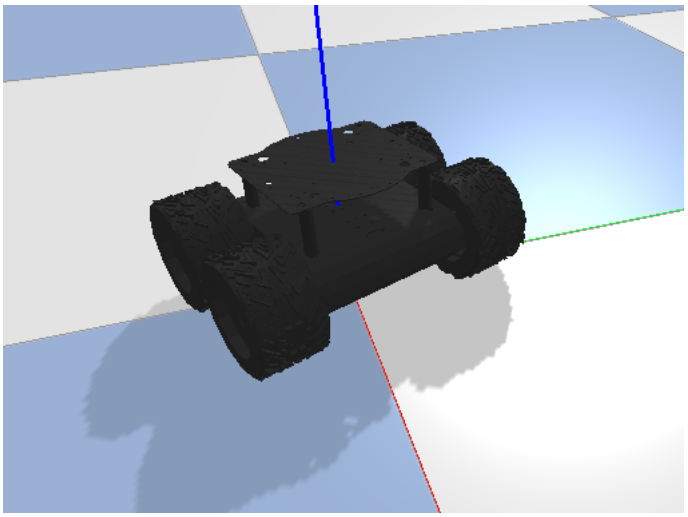


FIGURE IV.3 – Agent WifiBot au sein de la simulation PyBullet.

L'espace d'action de l'agent est continu et de dimension 2 avec une vitesse d'avance a_{av} et une vitesse de rotation a_{ro}. La vitesse des moteurs de chaque roue se contrôle à partir des consignes $(a_{av}, a_{ro}) \in [-1, 1]$:

$$\begin{cases} v_{moteurs\_gauche} = (a_{av} + a_{ro})\beta \\ v_{moteurs\_droit} = (a_{av} - a_{ro})\beta \end{cases} \tag{IV.9}$$

où β est une constante fixant la vitesse d'avance maximale de l'agent à environ 0.8 m/s.

Bien que nous considérons que l'agent ait une perception des objets qui l'entourent, nous n'utilisons pas de simulations de capteurs pour concevoir l'espace d'observation de l'agent. Les données souhaitées sont directement récupérées depuis l'interface PyBullet. Nous mesurons des positions, orientations, vitesses de chaque élément de l'environnement pour construire les différents espaces d'observation dont nous avons be-

3. https://www.wifibot.com/
4. Il s'agit d'un format permettant de décrire sous la forme d'un fichier standardisé un robot complet. L'ensemble des objets de l'environnement que nous avons conçus est créé suivant ce format.

soin. PyBullet nous fournit ces informations dans le repère monde R_{monde}. Or, la majorité des observations dont nous avons besoin sont des mesures exprimées dans le repère de l'agent (équivalentes à celles issues de l'un de ses capteurs). Pour exprimer la position d'un objet dans le repère mobile R_{agent} de l'agent, nous utilisons la matrice de transformation :

$$\begin{bmatrix} x^{R_{agent}} \\ y^{R_{agent}} \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x^{R_{monde}} \\ y^{R_{monde}} \\ 1 \end{bmatrix} \tag{IV.10}$$

où θ est la rotation de l'agent selon l'axe vertical monde $z^{R_{monde}}$ et où t_x, t_y sont les translations selon $x_{R_{monde}}, y_{R_{monde}}$ de l'agent dans le repère monde

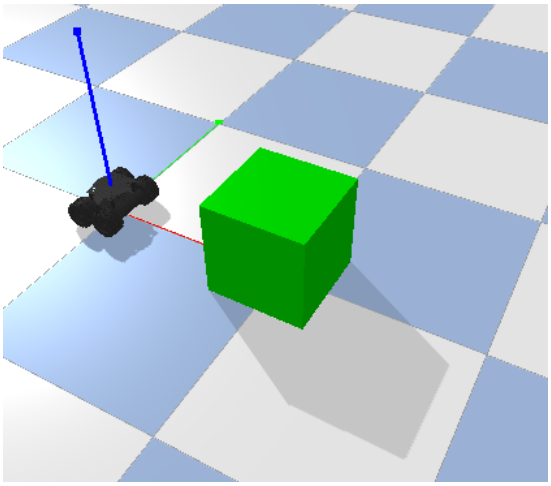
Le pas de temps de simulation est fixé par défaut dans PyBullet à 240 Hz. Chaque action est cependant répétée 72 fois de manière à réaliser des acquisitions seulement toutes les 0.3s. Le terme pas de temps utilisé en apprentissage par renforcement correspondra dans la suite de l'étude à 0.3s d'interactions.

IV–3.1.b Les objets

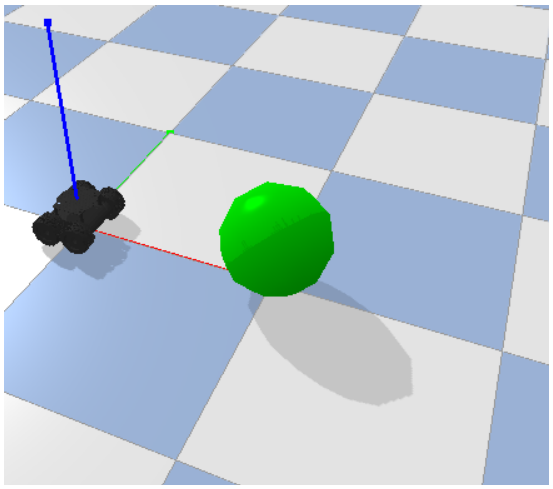
Pour tester la méthode, nous souhaitons que les objets présentent des comportements qui soient distinguables sans dépendre de la précision du simulateur, mais suffisamment divers pour nécessiter plusieurs interactions de la part de l'agent afin d'enlever toute ambiguïté entre les propriétés à l'origine de leurs comportements.

À cette fin, nous choisissons trois formes possibles : un cube (Figure. IV.4a), une sphère (Figure. IV.4b) et un cylindre reposant sur une de ses génératrices (Figure. IV.4c). Le cube et la sphère ont deux particularités bien distinctes : l'un roule et l'autre glisse. Le cylindre lui possède ces deux caractéristiques puisque selon la direction de poussée, il va rouler ou glisser. De ce fait, si l'agent interagit mal avec le cylindre, il peut confondre le cylindre avec l'un des deux autres objets.

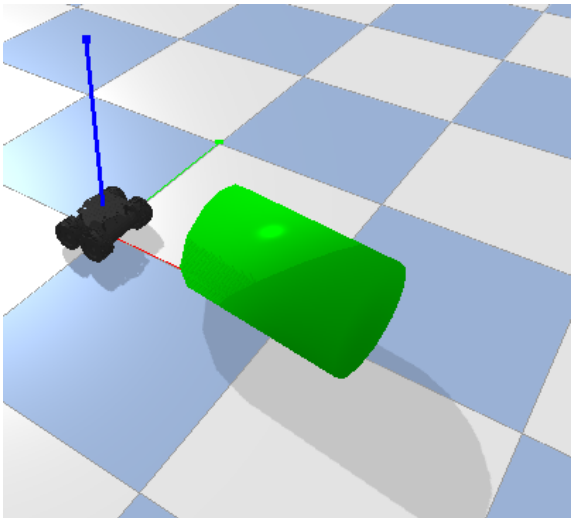
Tel que proposé, l'environnement reste trop simple et ne demande en théorie pas plus de 2 interactions pour comprendre quels objets peuvent rouler ou glisser. Nous apportons de la complexité au problème en faisant varier les propriétés de masse, et friction (équivalent à différents types de contacts entre matériaux). Concernant la friction, PyBullet nous permet de modifier deux propriétés : la résistance au roulement et le frottement latéral. La résistance au roulement impacte la sphère et le cylindre. Le frottement latéral correspond à un frottement modélisé par la loi de Coulomb. Par l'ajout de ces deux nouvelles propriétés, la confusion peut avoir plusieurs sources. Par exemple un objet très glissant avec frottement latéral faible peut être confondu avec un objet roulant mais aussi être considéré comme un objet léger alors que ce n'est pas forcement le cas. Sans une variété d'interactions l'agent ne sera pas en mesure de



(a) Cube de coté 20cm.



(b) Sphère de rayon 10cm.



(c) Cylindre de rayon 10cm et de longueur 30cm.

FIGURE IV.4 – Les formes d'objets, de taille fixe, que l'agent peut être amené à rencontrer durant ses expérimentations.

distinguer les propriétés internes de l'objet. Pour distinguer la masse du frottement, il va devoir par exemple effectuer des chocs courts mais aussi des contacts et poussées plus longs.

Chacune de ces propriétés est tirée aléatoirement dans un espace que nous avons défini empiriquement, les espaces étant définis dans le Tableau IV.1. Nous avons fait en sorte que l'objet puisse être très lourd et peu mobile comme très léger et donc très mobile, le rendant ainsi difficile à manier. De même les frottements et résistances sont choisies de sorte à ce que l'objet soit extrêmement glissant ou alors difficilement poussable. Ainsi le cylindre peut être facilement manoeuvrable dans le sens du glissement mais pas dans le sens du roulement car une forte résistance au roulement l'empêche quasiment d'avancer. Nous nous sommes assurés que, l'objet puisse toujours bouger,

TABLE IV.1 – Plages possibles des propriétés des objets.

Propriété	Valeur		
	Cylindre	Cube	Sphère
Dimension (cm)	$l = 30$ $r = 10$	$l = 20$	$r = 10$
Masse (kg)	$[0.1, 6]$	$[0.1, 6]$	$[0.1, 6]$
Frottement latéral	$[0.1, 0.5]$	$[0.01, 0.3]$	1
Résistance à la rotation	0.001	0.001	0.001
Résistance au roulement ($\times 10^2$)	$[0.4, 5]$	0.1	$[0.4, 5]$
Restitution	0.2	0.2	0.2

même très légèrement, quelle que soit la combinaison choisie. Si l'objet reste constamment immobile, il n'apportera aucune information à l'agent pouvant créer des gênes lors de l'entraînement des modèles. Ce n'est pas un cas que nous souhaitions traiter dans ces premières validations.

Comme nous avons décidé de travailler uniquement sur une surface plane et sans obstacles, nous ne modifions pas le coefficient de restitution puisqu'il ne peut pas être observé par les interactions.

IV–3.1.c Définition de la scène

L'agent est positionné au sein de l'environnement tel qu'illustré sur la Figure IV.5. Son orientation par rapport à l'axe vertical du repère monde est tirée aléatoirement dans $\theta_z^{R_{agent}} \in [-180°, 180°]$. L'objet quant à lui peut apparaître avec une orientation aléatoire $\theta_z^{R_{objet}} \in [-180°, 180°]$ dans une zone délimitée par un tracé carré, centré sur l'agent. Ses propriétés de forme, de masse, de frottement latéral et de résistance au roulement sont tirées aléatoirement de manière uniforme, à chaque épisode, dans leurs espaces de définition donnés dans le Tableau IV.1. Nous effectuons dans un premier temps l'entraînement sur les 3 formes d'objets qui sont : le cube, le cylindre couché, et la sphère.

A $t = 0$, les vitesses de l'objet et de l'agent sont nulles. Ensuite, pour chaque pas de temps, la politique de l'agent renvoie une consigne de vitesse d'avance et de rotation $\{a_{av}, a_{ro}\}$ normalisées entre $[-1, 1]$. Les mesures que l'agent peut effectuer sont les suivantes :

- $\{v_a, v_l\}$: sa vitesse d'avance et sa vitesse latérale exprimé dans son repère à l'instant courant,
- $\{r, \theta\}$: sa position et orientation relative par rapport au centre de masse de

l'objet,

- $\{d_x, d_y\}$: le déplacement de l'objet entre deux pas de temps t et $t+1$ exprimé dans le repère de l'agent à $t=0$,
- $\{v_x, v_y\}$: la vitesse instantanée de l'objet au pas de temps $t+1$ exprimé dans le repère de l'agent à $t=0$ [5],
- $\{c\}$: un booléen indiquant s'il y a eu contact avec l'objet entre deux pas de temps t et $t+1$,
- *Une carte d'occupation égocentrique*.

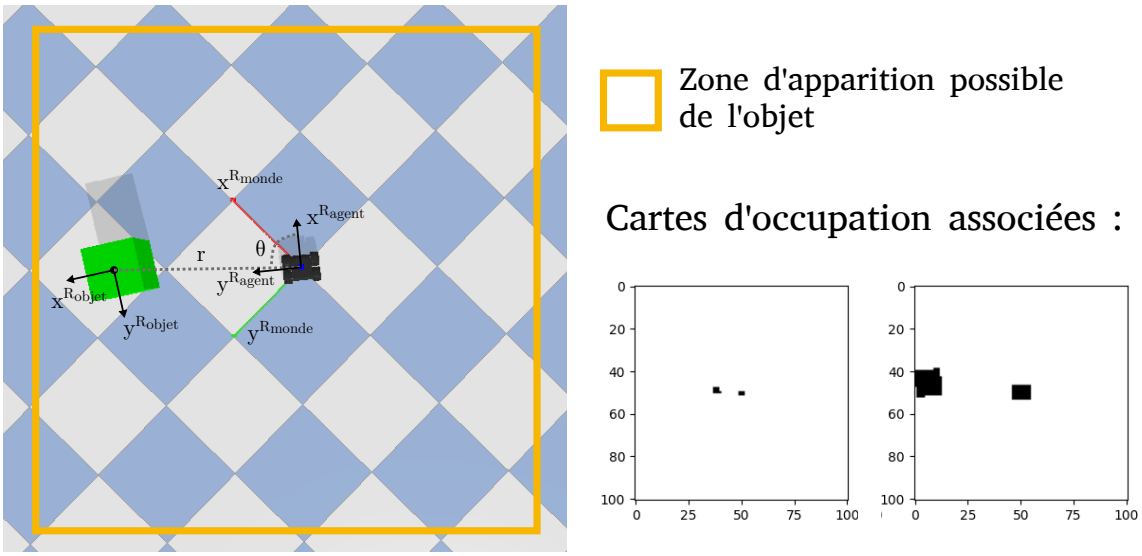


FIGURE IV.5 – Environnement aux conditions initiales avec les cartes d'occupation à 2 échelles observées par l'agent. Les deux cartes sont chacune de dimension 100 × 100 pixels. Le coté d'un pixel vaut une distance de 15cm pour la vue éloignée et 4cm pour la vue proche.

La façon de fournir la position et vitesse des objets à l'agent est une problématique en soit. Dans le cas simple où nous donnons les coordonnées de l'objet à l'agent, deux problèmes se posent lorsque nous passons au cas multi-objets. D'une part, il faut connaître à l'avance le nombre d'objets ce qui impose au vecteur d'observation de rester de taille fixe durant l'entraînement. D'autre part, cela pose des problèmes d'invariance dans les mesures du vecteur d'observation, puisque la même scène peut être représentée par toutes les permutations de ces mesures.

Nous proposons d'utiliser une grille d'occupation pour représenter la position des objets dans l'environnement proche du robot. Cette méthode nous semble plus souple pour prendre en compte le cas multi-objets et la présence d'obstacles dans de futurs travaux. La dimension et la résolution de cette grille discrète sont fixées selon la distance de vision et la qualité de résolution que nous souhaitons avoir. Pour savoir si

5. $\{v_x, v_y\}$ est obtenu simplement en divisant $\{d_x, d_y\}$ par le pas de temps de simulation.

une case de la grille est occupée ou non, nous testons si chaque case appartient au un polygone représentant l'objet vu de dessus à l'aide d'une fonction de la librairie Python "matplotlib". Pour pouvoir créer ce polygone, nous supposons connaître à l'avance les dimensions des objets utilisés (rayon, longueur, largeur). Ces hypothèses peuvent être supprimées en construisant la carte à partir d'algorithme moderne de cartographie en temps réel.

IV–3.2 Paramétrisation de la méthode

IV–3.2.a Choix de l'espace de maximisation

Pour entraîner la politique MEPOL, il est nécessaire de définir l'espace $\mathbf{s}^{\text{ent}}$ sur lequel l'entropie permettant l'optimisation de la politique va être estimée.

D'une part, cet espace doit être un sous-espace de l'état interne de l'objet, non relatif à l'agent, sans quoi des actions de l'agent sans impact sur l'objet, inutiles donc, deviennent sources d'entropie. D'autre part, cet espace doit refléter une notion de variabilité dans la dynamique de l'objet puisque nous avons fait l'hypothèse qu'elle seule pouvait mener à l'extraction des propriétés.

Nous proposons alors de maximiser l'entropie sur les vitesses instantanées $\{v_x, v_y\}$ observées par l'agent, que nous projetons en coordonnées polaires dans le repère de l'objet. L'intérêt de choisir $\mathbf{s}^{\text{ent}} = \{r^{objet}_{v_x,v_y}, \theta^{objet}_{v_x,v_y}\}$ est que la vitesse de l'objet est directement reliée à de nombreuses propriétés dynamiques. En la faisant varier, nous espérons pouvoir identifier avec précision des propriétés comme la masse, les frottements, et plus encore. Cet espace est d'autant plus intéressant qu'une majorité des objets sont anisotropes [6] concernant la direction de poussée. L'idée est donc d'encourager l'agent à interagir avec l'objet selon différents angles d'approche et différentes vitesses. Un grand avantage de cet espace est qu'il peut être appliqué à tous les objets et s'adapter à des environnements 3D en rajoutant la vitesse selon l'axe vertical. Pour projeter $\{v_x, v_y\}$ dans le repère de l'objet, l'agent doit être capable de suivre, par des points de repère, les différentes rotations de l'objet et ainsi pouvoir identifier à chaque pas de temps l'orientation du repère qui lui est associé. Étant en simulation, nous utilisons directement les informations du simulateur pour projeter dans l'espace souhaité sans passer par les capteurs de l'agent.

IV–3.2.b Détails d'implémentation de la politique

Le modèle utilisé dans MEPOL n'est pas pensé pour recevoir des observations de type images / cartes comme entrées de la politique. Il est simplement constitué de

6. Corps dont les propriétés varient selon la direction considérée

quelques couches denses entièrement connectées. Ainsi, nous nous sommes limités à un espace d'observation de faible dimension constitué de :

$$\mathbf{s} = [r, \theta, d_x, d_y, v_x, v_y, c]^\intercal$$

Mais comme indiqué précédemment, cet espace ne nous semble pas viable pour une implémentation dans un environnement inconnu potentiellement encombré.

De ce fait, nous utilisons comme espace d'observation deux cartes d'occupation égocentriques à 2 échelles comme représentées sur la Figure IV.5. La première carte est une vue lointaine de l'environnement ayant pour but de localiser l'agent par rapport à l'ensemble des éléments de la scène. La seconde carte est une vue plus proche et donc plus détaillée de ce qui se trouve autour de l'agent, lui permettant d'interagir avec précision sur l'objet. Sur la base de l'information donnée par ces cartes, l'agent ne peut pas connaître sa vitesse à l'instant t et celle de l'objet relativement à lui. Comme la carte est égocentrique, le seul moyen d'indiquer la vitesse de l'agent et de lui fournir cette mesure en plus des cartes. Par contre, la vitesse relative de l'objet peut être inférée par la politique en donnant une série de cartes acquises sur les derniers pas de temps,[7] ou alors en utilisant une architecture récurrente. C'est cette seconde approche qui a été privilégiée dans un but de permettre à la politique de construire une mémoire de sa trajectoire et d'effectuer des actions qui maximisent $\mathbf{s}^{\text{ent}}$.

Paramètres d'apprentissage : L'ensemble des hyper-paramètres importants relatifs à l'apprentissage et aux hyper-paramètres de MEPOL sont recensés dans le Tableau IV.2.

TABLE IV.2 – Paramètres liés à MEPOL et l'apprentissage.

Paramètre	Valeur
Learning rate (α)	10^{-5}
Optimiseur	Adam
Nombre d'epochs maximal	300
Horizon (T)	500
k-NN (k)	10
Seuil de confiance (δ)	15
Taille du batch de trajectoire (N)	20

Construction du modèle de la politique : Dans notre version alternative de MEPOL, nous avons modifié le modèle de la politique originale pour avoir la possibilité

7. Technique dite de frame-stacking couramment utilisée en RL pour inférer la dynamique d'une scène.

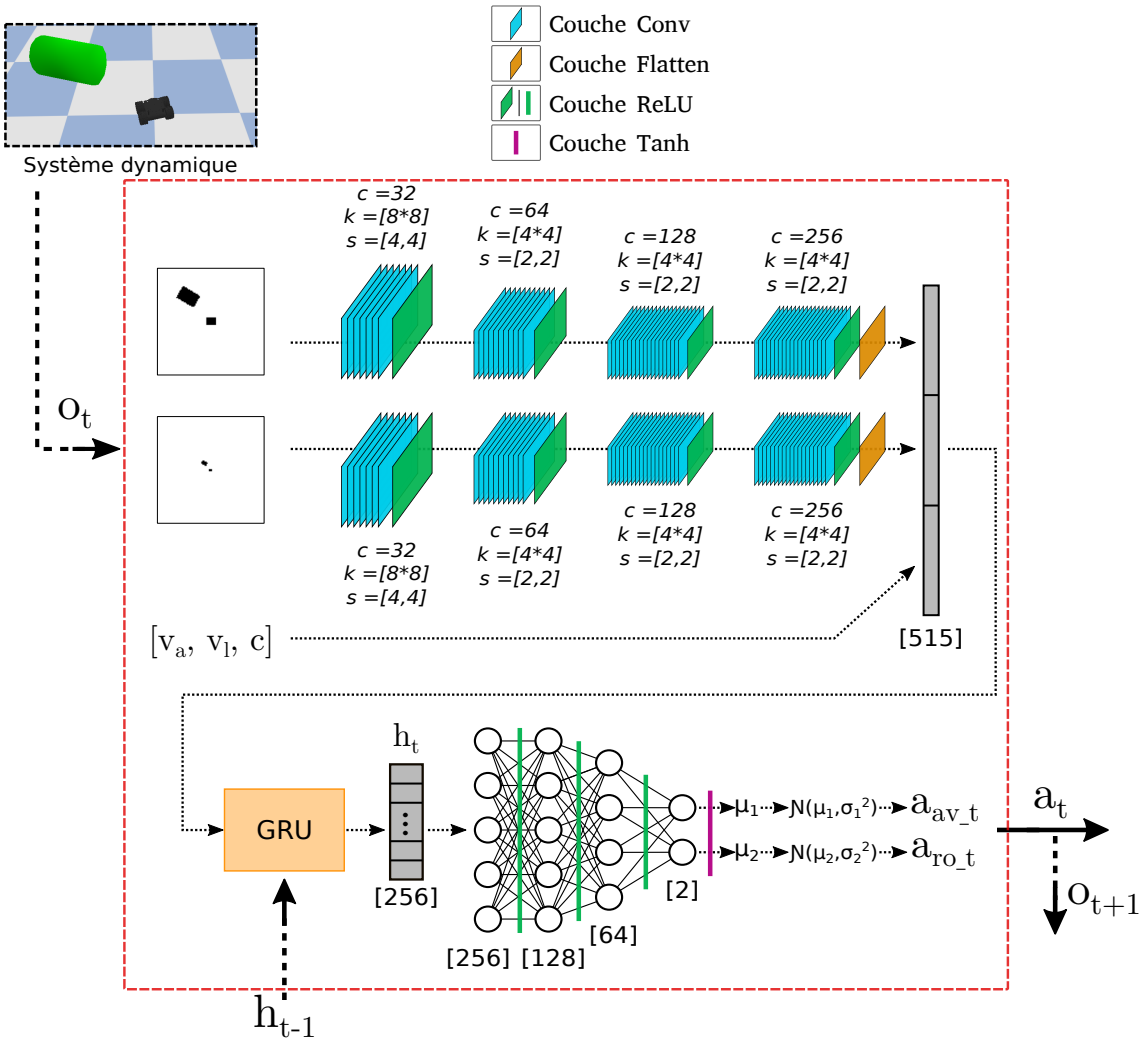


FIGURE IV.6 – Architecture du modèle de la politique utilisée dans notre version de MEPOL.

de donner en entrée des images et des mesures scalaires, mais aussi d'inclure un réseau récurent[8]. La Figure IV.6 donne le détail de cette architecture en indiquant le nombre de couches et les paramètres sélectionnés. Deux réseaux convolutionnels indépendants sont chargés d'encoder dans un vecteur de caractéristiques les deux cartes égocentriques. Cette indépendance est nécessaire étant donné que les cartes sont à deux échelles différentes et qu'il n'y a donc aucune raison de chercher des relations spatiales entre ces deux entrées. Ils sont concaténés aux vecteurs caractéristiques les observations $[v_a, v_l, c]$. Le nouveau vecteur constitué est ensuite donné en entrée à un réseau récurrent GRU qui produit un état caché h_t. Cet état est introduit dans un réseau dense multi-couche pour prédire les moyennes et les écart-types des gaussiennes

8. La méthode de MEPOL ne permet pas de travailler avec un espace $\mathbf{s}^{ent}$ de grande dimension. Cependant, elle ne semble pas incompatible avec l'utilisation d'un état $\mathbf{s}$ de plus grande taille. Il est important de noter cette séparation entre $\mathbf{s}$ et $\mathbf{s}^{ent}$.

sur lesquelles les actions $\{a_{av}, a_{ro}\}$ sont tirées.

IV–4 Évaluation de la politique

Dans cette section, nous évaluons dans un premier temps les trajectoires apprises par l'agent après avoir appliquée la méthode dans l'environnement que nous venons de présenter. Cette évaluation ne permettant pas d'avoir une indication de la qualité de la politique concernant l'objectif d'identification des propriétés de l'objet, nous avons dû, afin d'évaluer l'agent, passer par une tâche annexe de prédiction supervisée que nous présentons dans la seconde partie de cette section.

IV–4.1 Analyse des trajectoires de l'agent

Dans les conditions présentées, il faut environ 16h sur un GPU de type Nvidia RTX 2080 Ti pour converger vers une politique qui maximise correctement l'entropie et un peu plus de 48h pour achever les 300 epochs [9]. Cela correspond à 6000 objets aux propriétés différentes vus durant l'entraînement. Au delà des 300 epochs, les performances stagnent et donc il n'y a plus d'intérêt à continuer l'apprentissage. La courbe d'évolution de l'entropie en fonction du nombre d'epochs effectués est donnée sur la Figure IV.7.

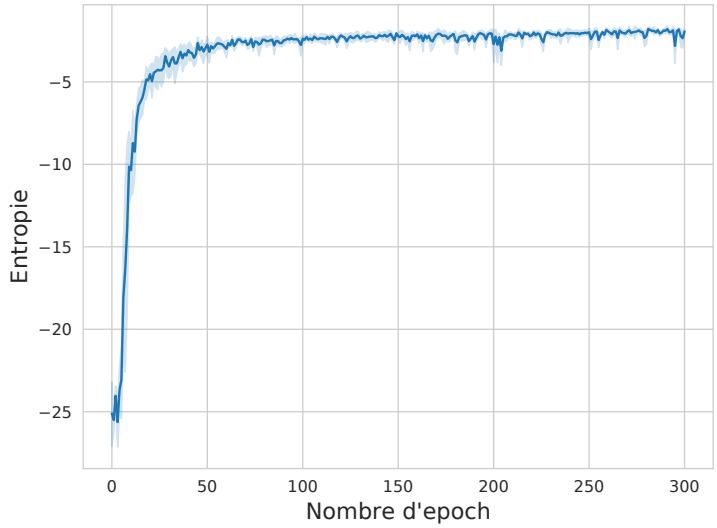


FIGURE IV.7 – Évolution de l'entropie en fonction du nombre d'epoch effectué.

Visuellement, les trajectoires que l'agent prend sont proches de ce que nous attendions. Il cherche à pousser l'objet en changeant régulièrement de point d'impact et en

9. Le simulateur PyBullet ne fait pas de multithreading lors de la simulation ce qui rend la génération de trajectoires assez lente.

faisant le tour de l'objet. La Figure IV.8 illustre ce comportement lorsque l'agent est confronté à chacune des 3 formes possibles du dispositif expérimental.

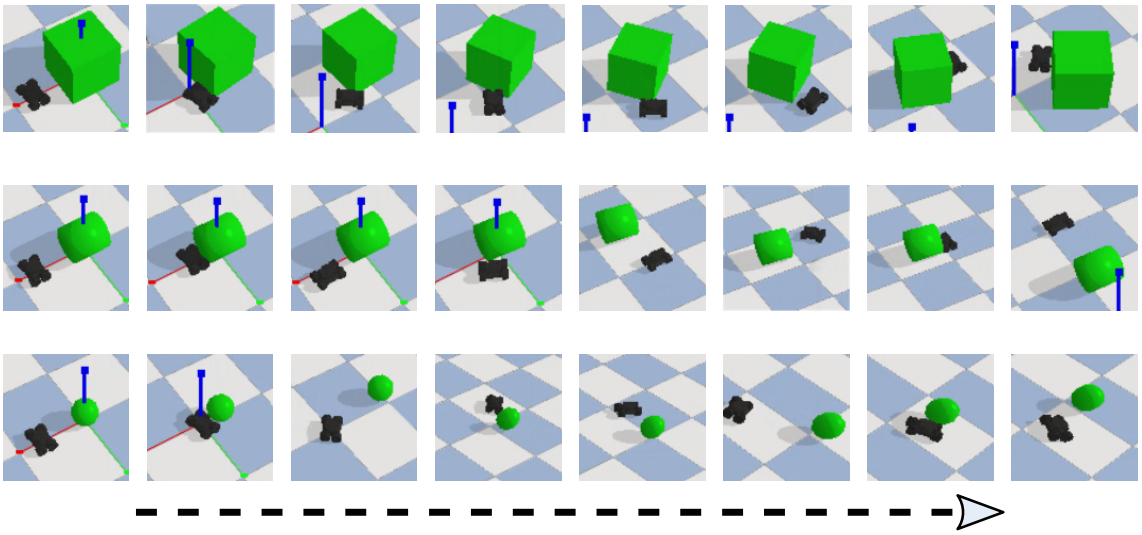
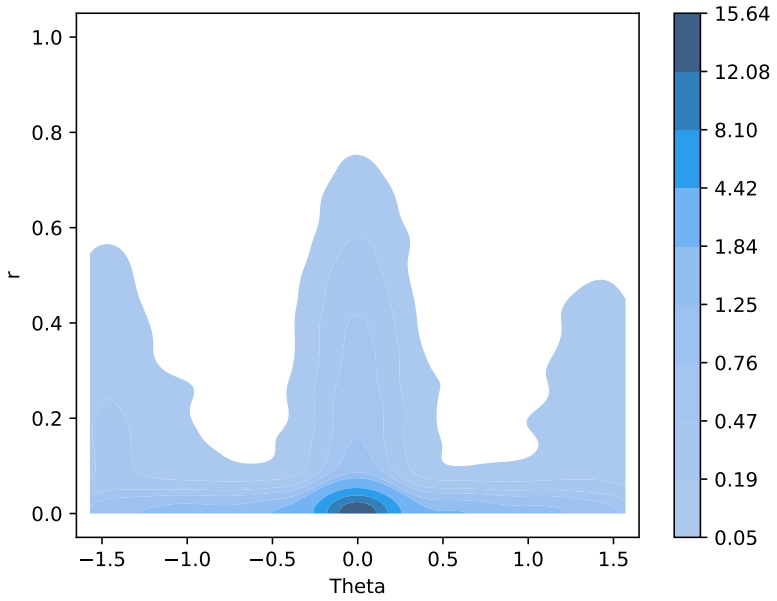


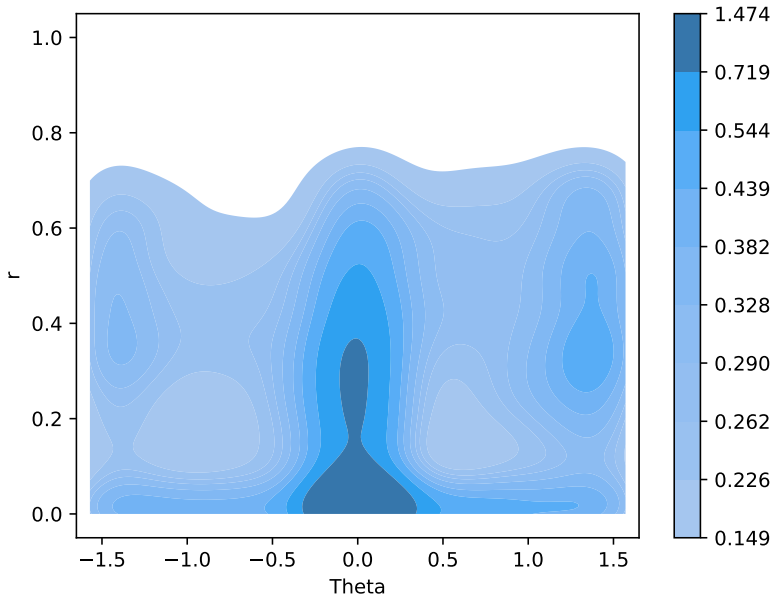
FIGURE IV.8 – Trajectoires de l'agent face à un objet de chaque forme.

Vérifions maintenant quantitativement ces affirmations. Nous sauvegardons les paramètres $\boldsymbol{\theta}$ pour différents niveaux d'entropie atteints. Nous utilisons ensuite cette politique π_{MEPOL} pour effectuer 20 trajectoires de 500 pas de temps et représenter avec une carte de fréquentation la distribution engendrée par la politique sur l'espace S^{ent}. Les cartes données sur la Figure IV.9 montrent la capacité de l'agent à explorer l'espace S^{ent} après 10 et 300 epochs. Nous pouvons observer une nette différence sur la capacité d'exploration de ces deux politiques. Après convergence de la politique, l'agent recouvre avec de fortes densités la carte, ce qui signifie qu'il est capable de prendre des actions qui vont varier la vitesse de l'objet selon différentes directions. Ainsi, l'agent a appris à changer son point d'impact sur l'objet en venant l'impacter sur l'ensemble de son contour (variations sur l'axe "Theta"). Les variations observées sur l'axe des ordonnées, indiquent que l'objet atteint des vitesses plus ou moins élevées. Notons une densité élevée lorsque r est proche de 0. Elle est due aux instants où l'agent n'est pas en contact avec l'objet comme des périodes de repositionnement de l'agent par rapport à l'objet et les temps de trajet pour le rejoindre. Pour réaliser ces variations, soit l'agent entre en contact avec l'objet avec une vitesse très élevée puis observe le ralentissement de l'objet, soit il pousse l'objet et reste en contact avec lui à différentes vitesses.

Pour vérifier cela, nous traçons sur la Figure IV.10 la distribution des actions de l'agent avant un contact ainsi que sa vitesse quelques millisecondes après ce contact pour une seule trajectoire de longueur $T = 500$. Nous pouvons voir sur le graphique de gauche que la politique a tendance à privilégier sur une trajectoire des actions menant à la vitesse maximale, bien que nous dénombrions une quantité uniforme



(a) Après 10 epochs.



(b) Après 300 epochs.

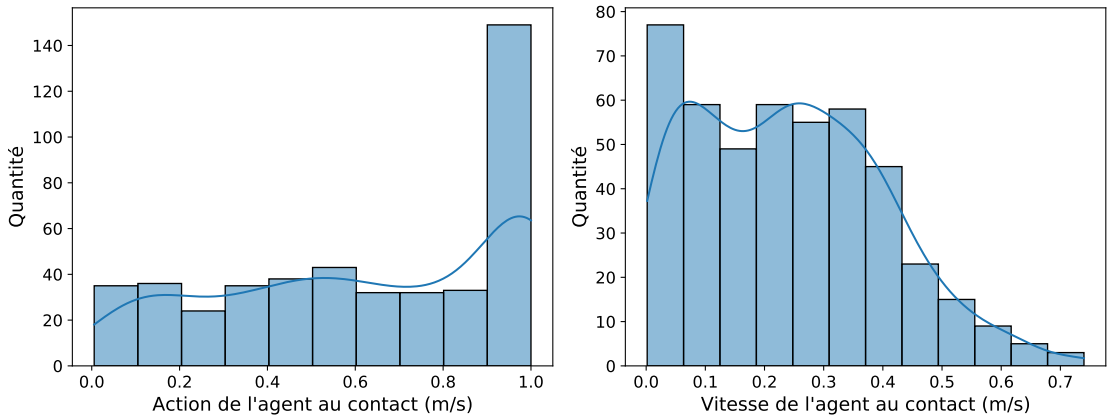
FIGURE IV.9 – Fréquence de visite de l'espace S^{ent} après 10 et 300 epochs.

d'actions pour des vitesses entre 0 et 0.9 m/s. Cette distribution peut être justifiée de différentes manières.

- L'agent peut chercher à frapper l'objet puis l'observer ralentir (méthode qui peut suffire à maximiser l'entropie si l'angle d'impact change),

- La vitesse de l'agent avant le contact est faible et il cherche donc à accélérer pour atteindre différentes vitesses au moment du contact,
- Lorsque la masse et le frottement/résistance de l'objet sont élevées, au moment du contact, l'objet va peu bouger et l'agent va atteindre une vitesse nulle. En restant en contact avec cet objet et en poussant à vitesse maximale, il va réussir à déplacer tout en l'accélérant (jusqu'à atteindre une limite).

Ce dernier point semble être la raison principale. En regardant le graphique de la vitesse de l'agent après contact, nous constatons que l'agent atteint peu de vitesses au-delà de 0.5 m/s. Cela signifie que l'objet est soit lourd soit peu glissant/roulant. Par contre, il existe une répartition uniforme pour des vitesses inférieures à 0.4 m/s, ce qui signifie que l'agent cherche bien à varier la vitesse de l'objet en le poussant et sans uniquement l'observer glisser. Il reste cependant difficile de comprendre la stratégie



(a) Distribution des actions de l'agent qui ont menées à un contact avec l'objet.

(b) Distribution des vitesses de l'agent dans le même pas de temps qui a suivi le contact.

FIGURE IV.10 – Distribution de la commande (à gauche) et de la vitesse observée (à droite) concernant l'agent, au moment d'un contact avec l'objet. Mesure effectuée pour une trajectoire de $T = 500$.

exacte de l'agent bien que nous ayons quelques pistes. La stratégie la plus probable est que l'agent alterne entre vitesse maximale et variations à plus faibles vitesses pour s'adapter à toutes les conditions (objet peu mobile et très mobile). La courbe de la distribution des actions est en moyenne similaire à celle affichée pour tous les objets. Celle de la distribution de la vitesse de l'agent varie selon les propriétés de l'objet (masses et frictions).

Plusieurs questions se posent désormais : la trajectoire issue de MEPOL permet-elle d'acquérir de l'information sur la dynamique de l'objet ? En quoi la politique MEPOL la rend meilleure qu'une autre ?

IV–4.2 Caractérisation de la politique permettant l'identification des propriétés physiques

La difficulté rencontrée lors de la conception de cette méthode est de savoir si cette politique, qui semble en apparence intéressante, est efficace pour permettre plus tard la construction d'une représentation de la dynamique de l'objet sans supervision. Cela nous semble possible uniquement si les observations issues des interactions contiennent des régularités corrélées à des propriétés. De plus, nous pouvons faire l'hypothèse que plus ces observations contiennent et exposent ces régularités, meilleures son les propriétés extraites. Or ces deux hypothèses peuvent être testées dans un premier temps hors du cadre délicat de l'apprentissage non-supervisé souhaité. C'est la solution que nous proposons dans cette sous-section : évaluer ces hypothèses dans un cadre d'une prédiction supervisée des propriétés à partir d'une série d'observations acquises grâce à la politique. Nous supposons donc, pour cette évaluation seulement, connaître certaines propriétés de l'objet tel que sa forme, sa masse et ses coefficients de frottement latéral et résistance au roulement.

IV–4.2.a Méthode d'évaluation de la politique

L'objectif est d'entraîner le modèle à prédire l'ensemble de ces propriétés à la fois, à partir d'une série d'observations issues d'une même trajectoire. Pour cela, nous constituons un ensemble de données D en suivant la méthode décrite pour la phase d'inférence. D est divisé en 3 sets de façon à obtenir un set d'entraînement D_{train}, un set de validation D_{val} et un set de test D_{test}. Chaque trajectoire est une série d'observations $\mathbf{x}_t$ composées des mesures :

$$\mathbf{x}_t = [e^{(-r^t)}, \cos(\theta^t), \sin(\theta^t), d_x^t, d_y^t, v_x^t, v_y^t, c^t, v_a^t, v_l^t, a_{av}^{t-1}, a_{ro}^{t-1}]^\intercal$$

où l'orientation relative θ est introduite sous la forme d'un sin et cos pour éviter les discontinuités aux bornes de l'espace de définition de $[-180°, 180°]$. Afin de faciliter la normalisation en environnement inconnu, nous prenons l'exponentielle négative de r. En se rapprochant de l'objet, cette valeur est proche de 1. D_{train}, D_{val}, D_{test} sont donc ici composés des paires $(\tau^k = [\mathbf{x}_0^k, \mathbf{x}_1^k, ..., \mathbf{x}_T^k], P^k = \{p_1, p_2, ..., p_l\})$.

Pour cette tâche de prédiction, nous avons imaginé un modèle composé de plusieurs têtes de prédictions, voir Figure IV.11. Chacune de ces têtes est chargée de prédire une propriété différente à partir d'un même espace latent qui a accumulé l'ensemble des observations $\mathbf{x}^k$ de la trajectoire τ^k. Pour cela, nous introduisons les observations $\mathbf{x}_t^k$ une par une dans une cellule GRU. L'état caché final $\mathbf{h}_T$, de la cellule GRU, devient alors une représentation latente de la trajectoire qui est donnée en entrée aux têtes de prédictions ρ_i dont l'architecture est un simple réseau de neurones

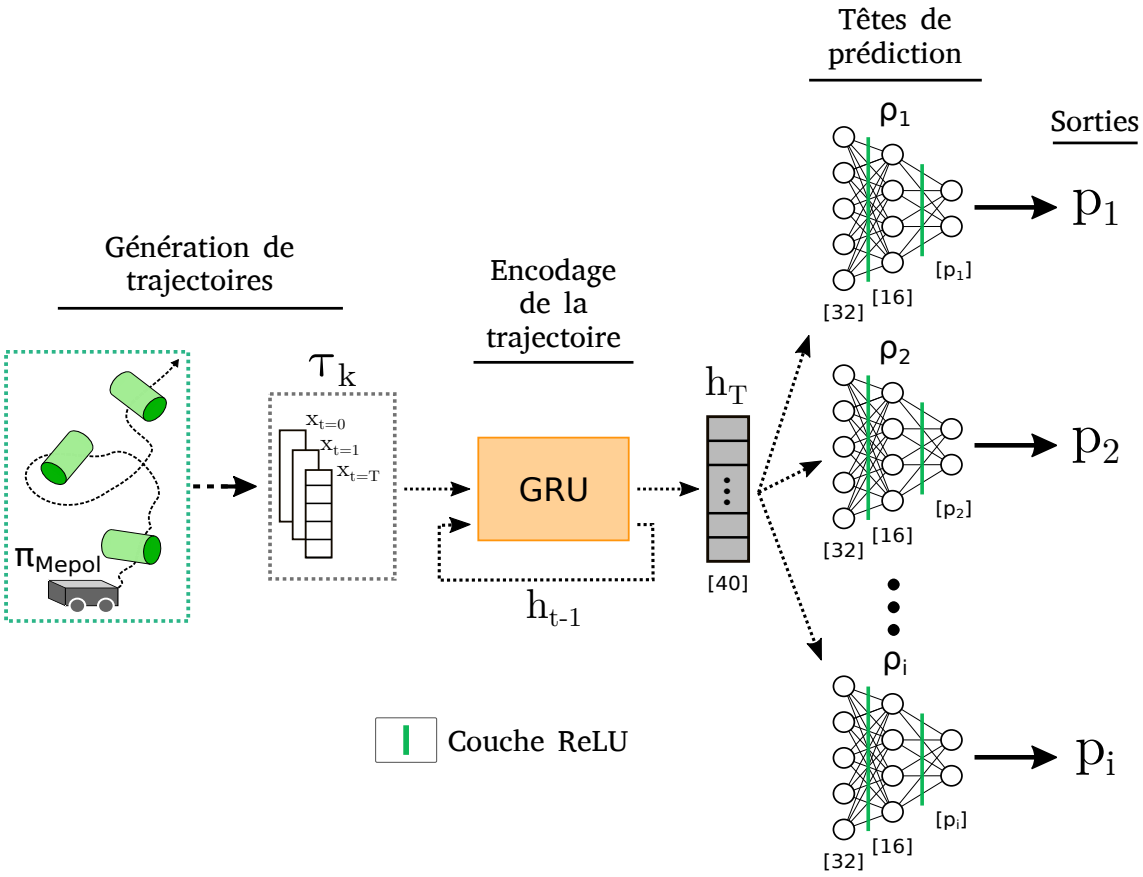


FIGURE IV.11 – Architecture du modèle de prédiction de propriétés.

entièrement connecté. L'ensemble des paramètres du modèle sont ensuite mis à jour avec la fonction de coût f_{Loss} calculée à partir de la somme des erreurs de prédiction :

$$f_{Loss} = \sum_{i}^{l} L_{p_i}$$

où

$$L_{p_i} = \begin{cases} \text{MSELoss}(\hat{p_i}, p_i) & \text{si } p_i \text{ est une propriété continue,} \\ \text{CrossEntLoss}(\hat{p_i}, p_i) & \text{si } p_i \text{ est une propriété discrète.} \end{cases} \tag{IV.11}$$

et l le nombre de propriétés.

5 modèles identiques sont entraînés de manière indépendant avec des set D_{train} différents (les hyper-paramètres d'apprentissages sont donnés dans le Tableau IV.11). Chaque modèle est évalué sur le même set D_{val} pour conserver les paramètres qui généralisent le mieux. Ces 5 modèles sont ensuite testés sur le même set D_{test}. Cela nous permet de calculer des moyennes et variances des prédictions indiquant s'il existe des observations d'entraînement de plus ou moins bonnes qualités. Pour évaluer les performances de prédiction, nous mesurons l'erreur absolue moyenne exprimée en

TABLE IV.3 – Paramètres d'apprentissage du modèle de prédiction supervisé.

Paramètre	Valeur
Learning rate	10^{-4}
Optimiseur	Adam
Nombre d'epochs	2000
Horizon (T)	200
Taille du batch	128

pourcentage (ou *Mean Average Percentage Error - MAPE*) pour les propriétés continues, et la précision de classification pour les propriétés discrètes.

IV–4.2.b Politique de comparaison

Le modèle qui vient d'être présenté nous permet d'évaluer la politique π_{MEPOL} à différents instants de son apprentissage, mais surtout celle qui mène à la plus grande diversité des mouvements de l'objet. Cependant, même si celle-ci donne de bonnes performances en prédiction, nous n'avons aucune élément permettant d'étayer que cette politique est la meilleure. Un comportement plus simple pourrait peut-être suffire à résoudre le problème de prédiction supervisé.

Ici encore la difficulté a été de trouver une politique à laquelle se comparer. Pour cela, il faut une politique entraînée également sans supervision et connaissances a priori sur l'environnement. Par conséquent, nous avons imaginé une politique naïve dont le but est simplement de pousser l'objet de manière répétée. Cette politique est entraînée avec un algorithme standard d'apprentissage par renforcement : l'algorithme *Proximal Policy Optimization* ou *PPO* [Schulman et al., 2017]. Nous noterons π_{PPO} cette politique. Pour l'entraîner, nous lui avons simplement donné comme récompense :

$$r_t = \begin{cases} 1 & \text{si l'agent a touché l'objet durant le pas de temps,} \\ 0 & \text{sinon.} \end{cases} \tag{IV.12}$$

Nous entraînons cette politique dans les mêmes conditions que celles mises en place pour obtenir π_{MEPOL}. L'espace d'observation et l'architecture de la politique sont identiques à ceux de la Figure IV.6. Comme pour π_{MEPOL}, nous générons avec π_{PPO} des sets D composés des paires $(\tau^k = [\mathbf{x}_0^k, \mathbf{x}_1^k, ..., \mathbf{x}_T^k], P^k = \{p_1, p_2, ..., p_l\})$.

IV–4.2.c Analyse des politiques apprises et variation d'hyperparamètres

Évaluons maintenant les performances de prédiction obtenues avec des modèles issus de π_{MEPOL} ou π_{PPO}. Dans un premier temps, nous validons l'hypothèse principale

de la méthode qui repose sur la maximisation de l'entropie, comme moyen d'obtenir des observations exposant plus facilement les caractéristiques d'un objet. Pour cela, nous évaluons les performances de modèles de prédiction de propriétés entraînés à partir d'observations provenant de politiques menant à différents niveaux d'entropie. Ensuite, nous comparons la politique π_{MEPOL} à la politique naïve π_{PPO} pour différentes variations d'hyperparamètres comme la taille de D_{train}, le bruit dans les observations recueillies, la longueur des trajectoires d'entraînement T. En utilisant l'architecture supervisée présentée sur la Figure IV.11, l'objectif est ici d'estimer la forme, la masse, le frottement latéral et la résistance au roulement. Seule la première propriété est discrète.

Le Tableau IV.4 rassemble l'ensemble des évaluations effectuées et indique quels sont les paramètres qui varient en fonction de l'expérience.

TABLE IV.4 – Ensembles des évaluations menées sur les performances du modèle de prédiction de propriétés. Les paramètres sont appliqués aux trajectoires de D_{train}, D_{val}, D_{test}. Les figures qui comparent la politique π_{MEPOL} avec π_{PPO} sont indiquées avec une $*$

Figure	**Entropie** π_{MEPOL}	**Taille** D_{train}	**Horizon** T	**Bruit**
Figure IV.12	Variable	2000	200	0
Figure IV.13 $*$	-1.733	Variable	200	0
Figure IV.14 $*$	-1.733	2000	Variable	0
Figure IV.15 $*$	-1.733	2000	200	Variable

Influence de l'entropie sur la richesse des observations : La Figure IV.12 montre la différence de performance entre la politique π_{MEPOL} et des politiques de plus faibles entropies : à gauche, le taux de classifications correctes de la forme (plus le taux est élevé, plus le modèle est performant), à droite, l'erreur MAPE pour chacune des propriétés étudiées (plus l'erreur est faible, plus le modèle est performant). Ces résultats sont calculés à partir du set D_{test} contenant 500 trajectoires de longueur $T = 200$ face à des objets avec des propriétés aléatoires.

La courbe révèle une augmentation significative de la précision de prédiction entre une politique d'action aléatoire et des politiques conduisant à une entropie plus élevée. En effet, le taux de classifications correctes passe de 40 à 98% et la MAPE atteint des valeurs entre 15 et 30% pour les propriétés continues. Ces résultats confirment la capacité de l'agent à acquérir des observations relatives aux caractéristiques de l'objet. Ils montrent aussi que cette capacité augmente, comme prévu, avec la maximisation de l'entropie sur S^{ent}. L'observation d'une plus grande MAPE sur les propriétés de frottement et résistance est logique puisque ce sont des propriétés bien plus complexes à estimer. Cette estimation est d'autant plus difficile que ces paramètres sont étroitement liés à la masse et que la variation des deux peut porter à confusion. Notons aussi le gain rapide de performances lors des premières augmentations de l'entropie.

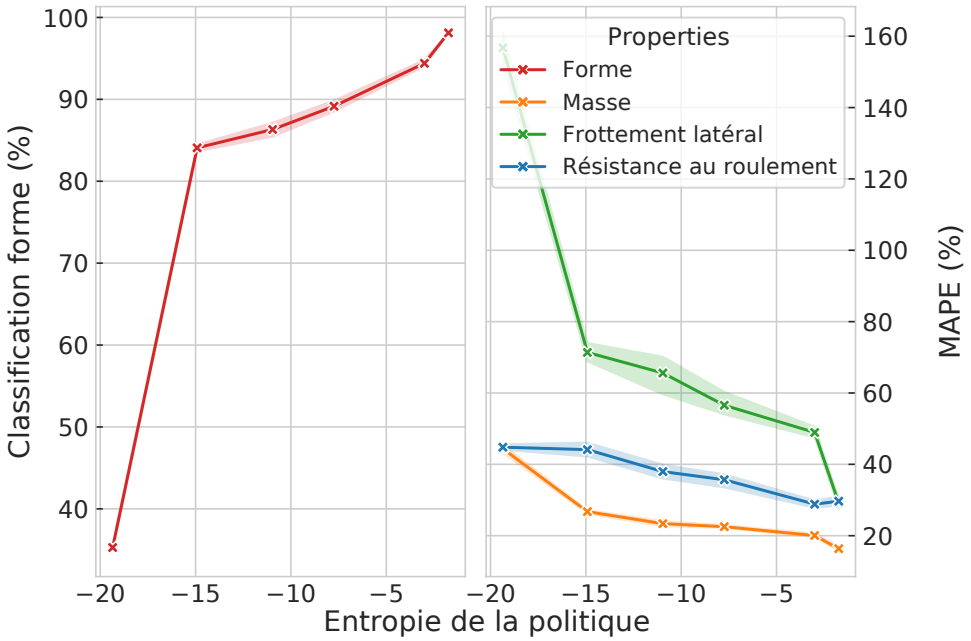


FIGURE IV.12 – Influence de l'entropie engendrée par la politique sur la précision des prédictions. A gauche, la classification de la propriété de forme. A droite, l'erreur de prédiction pour les propriétés continues.

Cette observation est logique sur la classification de forme puisqu'en théorie 2 actions bien choisies suffisent à déterminer quel objet peut rouler ou glisser : un contact selon un angle donné puis un second orienté de 90° par rapport au premier (distinction glissement/roulement). Dans les faits, ce n'est pas une solution aussi simple puisqu'il existe des combinaisons (frottement latéral, masse) pour lesquelles les mouvements résultants peuvent créer de la confusion.

Au vu des résultats obtenus, nous pouvons valider l'hypothèse que l'augmentation de l'entropie permet d'acquérir des observations plus riches en caractéristiques dynamiques de l'objet, ce qui engendre une classification et régression de ses propriétés plus précise.

Influence de la taille du set d'entraînement : Nous avons évalué sur la Figure IV.13 l'impact du nombre de trajectoire dans le set d'entraînement D_{train} sur les performances du modèle. Nous comparons cette fois-ci des modèles issus de la politique π_{MEPOL} avec des modèles issus de π_{PPO}.

La politique π_{PPO} permet déjà d'atteindre de bonnes performances. Ce n'est pas un résultat surprenant étant donné que la politique reste viable par rapport aux propriétés considérées. Pour observer la masse et le frottement/résistance, il suffit de frapper l'objet à une vitesse donnée puis d'observer la distance parcourue. Concernant la classification de la forme, le score élevé est au-delà de ce que nous espérions. L'agent

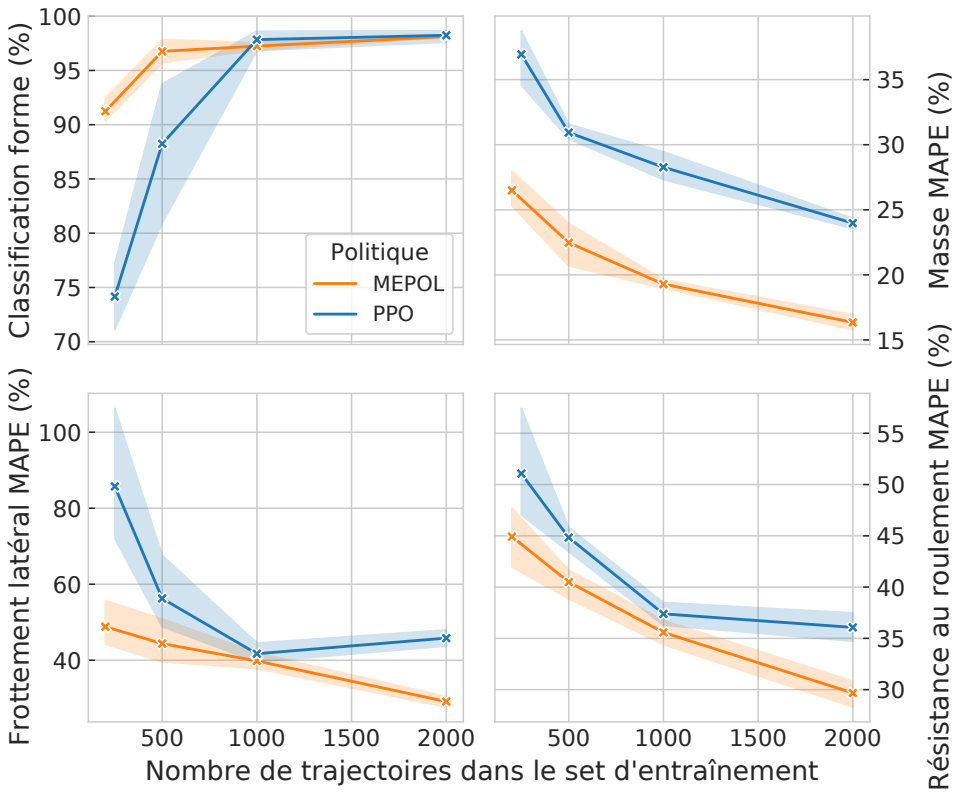


FIGURE IV.13 – Influence de la taille du set d'entraînement sur les performances des modèles lorsque entraînés avec des observations de π_{MEPOL} (en orange) ou π_{PPO} (en bleu).

semble, pour les sets de grande taille, arriver aisément à classifier l'objet. Il est probable que la tâche de classification soit évidente en s'appuyant sur la distribution de certaines mesures. Pour rappel, π_{PPO} est entraîné à taper le plus de fois possible l'objet. En observant son comportement, il semble avoir appris à entrer en contact avec l'objet selon un angle et position identique pour une même forme. Dans le cas du cube, il viendra toujours frapper une face alors que pour un cylindre, il restera en contact, soit avec un côté roulant, soit un côté latéral. Puis, peu importe l'objet, il le poussera continuellement avec la force maximale. Le problème provient en partie de l'observation r_{ao}. Comme l'agent se positionne quasiment de la même manière et reste en contact avec l'objet, la distance ne varie quasiment pas et la mesure effectuée permet de classifier l'objet. Environ toujours 0.32m pour la sphère, 0.35m ou 0.50m pour le cylindre, 0.42m pour le cube. Ce n'est probablement pas la seule observation qui pose ce type de problème. La performance de la politique π_{PPO} est par conséquent surestimée par rapport à ses réelles capacités à identifier des caractéristiques.

Malgré les performances de π_{PPO}, la politique π_{MEPOL} gagne en général 10% de précision sur les prédictions de propriétés continues et obtient des performances équivalentes sur la tâche de classification. Les résultats que nous souhaitons mettre en avant ici sont l'augmentation de la variance ainsi que la diminution de la précision

lorsque la taille de D_{train} diminue. Avec peu de données, les performances sont très vite dégradées pour π_{PPO} alors que π_{MEPOL} conserve de bonnes prédictions. Ces résultats nous permettent d'affirmer que les observations obtenues avec π_{MEPOL} sont de meilleure qualité en contenant plus d'informations sur les caractéristiques dynamiques de l'objet.

Influence de la longueur de trajectoire : Nous étudions ici l'influence du nombre d'interactions au travers d'une variation de la longueur de la trajectoire T. Les résultats sont présentés sur la Figure IV.14 en comparant une nouvelle fois π_{MEPOL} et π_{PPO}.

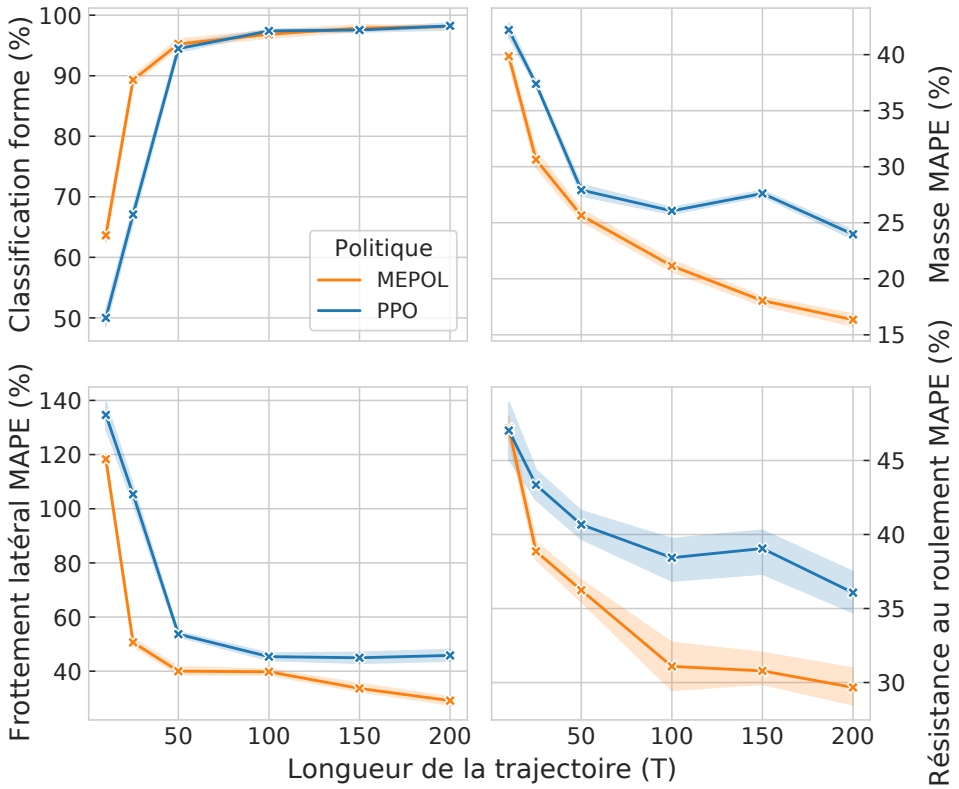


FIGURE IV.14 – Influence de la longueur de trajectoire sur les performances des modèles lorsque entraînés avec des observations de π_{MEPOL} (en orange) ou π_{PPO} (en bleu).

Pour la classification, nous observons des scores élevés pour les deux politiques lorsque la trajectoire dépasse 50 pas de temps. La politique π_{PPO} permet toujours de classifier correctement l'objet en s'appuyant sur la distribution de r_{ao}. Par contre, après 25 pas de temps correspondant à moins de 10 interactions, la politique π_{MEPOL} dépasse avec un écart de 23% les performances de π_{PPO} qui conserve 90% de classifications correctes. Étant donné que l'agent doit se rapprocher de l'objet, il n'est plus capable d'identifier la forme à l'aide de r_{ao} car il ne reste pas suffisamment contre, ce qui peut expliquer une telle chute des performances pour π_{PPO}.

L'erreur de prédiction des propriétés continues augmente avec la diminution de la longueur de la trajectoire. Pour $T = 50$, nous constatons que π_{MEPOL} permet d'atteindre

des performances de même qualité que celles de π_{PPO} pour $T = 200$. Cela confirme donc que la politique apprise avec notre méthode, qui effectue une exploration efficace de l'objet. Les résultats de cette étude valident l'idée pressentie qu'augmenter le nombre d'interaction conduit à un modèle plus précis pour identifier l'objet.

Influence du bruit sur les observations effectuées : Nous concluons ces expériences par l'ajout de bruit sur les mesures de l'agent permettant de se rapprocher de conditions réalistes et de tester la robustesse du modèle selon la politique employée. Nous simulons des erreurs sur la mesure de position du centre de l'objet en ajoutant un bruit gaussien à cette mesure. Cette erreur se propage donc aux observations $[e^{(-r^t)}, \cos(\theta^t), \sin(\theta^t), d_x^t, d_y^t, v_x^t, v_y^t]$. Les observations $[c^t, v_a^t, v_l^t, a_{av}^{t-1}, a_{ro}^{t-1}]$ de contact, de vitesse de l'agent et de commande ne sont pas bruitées. La Figure IV.15 compare les deux politiques pour un bruit d'estimation allant de 2cm à 20cm.

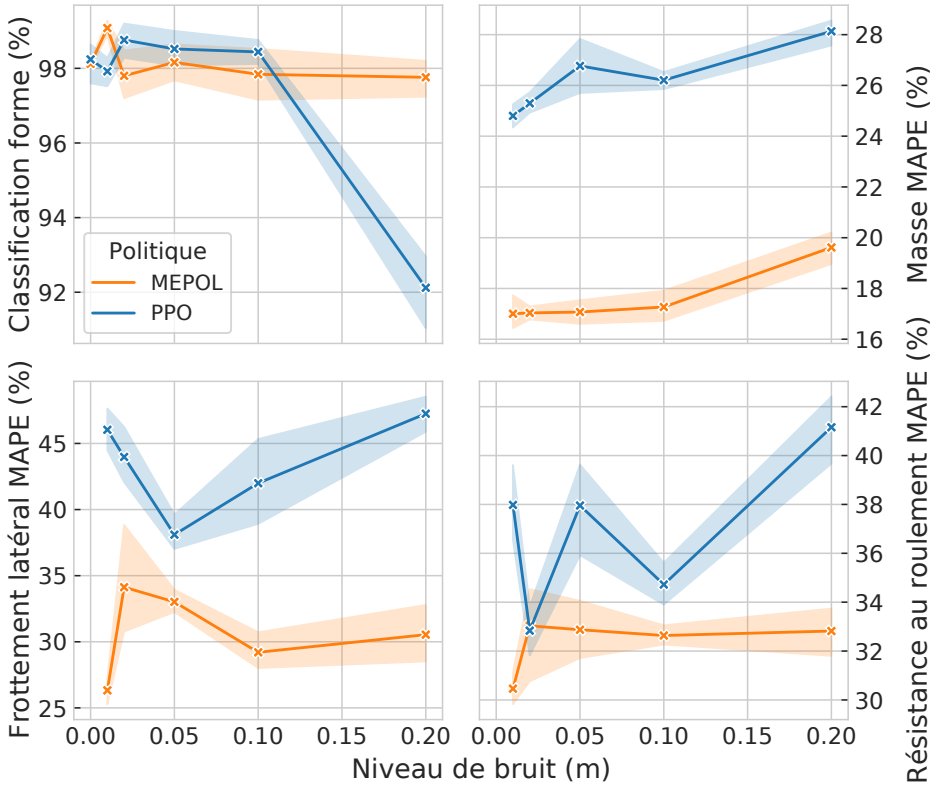


FIGURE IV.15 – Influence du bruit de mesure sur les performances des modèles lorsque entraînés avec des observations de π_{MEPOL} (en orange) ou π_{PPO} (en bleu).

Que ce soit pour une propriété discrète ou une propriété continue, les modèles obtenus avec π_{PPO} perdent en précision avec un bruit plus élevé. La politique π_{MEPOL} quant à elle reste relativement stable malgré des erreurs de mesure. Seule l'estimation du frottement latéral semble plus sensible au bruit avec de plus grandes variations. Notre méthode permet donc d'obtenir des observations permettant d'identifier des propriétés malgré la présence de bruit dans ces observations. Même avec un bruit

de 20cm, les prédictions restent plus précises que la politique de comparaison. Ces résultats nous permettent de rester optimiste sur l'obtention d'une représentation non-supervisé de l'objet même en conditions dégradées. Cela prouve aussi que l'exploration effectuée et par conséquent les observations acquises sont de bien meilleures qualités et plus informatives grâce à notre méthode d'apprentissage.

IV–5 Conclusion

Nous avons, dans ce chapitre, établi une méthode permettant à un agent d'apprendre à interagir avec un objet sans connaissances a priori sur celui-ci, grâce à un algorithme d'apprentissage par renforcement non-supervisé nommé MEPOL. Cet algorithme maximise l'entropie d'une distribution d'observations définies sur un espace de faible dimension que nous avons relié à des composantes de la dynamique de l'objet. Après avoir conçu un environnement dans lequel un robot mobile peut interagir par la poussée avec des objets de formes et de propriétés dynamiques aléatoires, nous avons appliqué la méthode proposée en réalisant un certain nombre de choix sur les observations de l'agent de sorte à éviter le besoin de connaissances a priori sur l'environnement. Après une évaluation qualitative des trajectoires d'interaction de l'agent, nous avons cherché à confirmer par une tâche de prédiction supervisée que la politique obtenue est particulièrement efficace pour l'identification de propriétés physiques sur l'objet. Pour cela nous avons conçu un modèle de prédiction composé d'une cellule GRU pour encoder les observations de l'agent sur un seul objet où l'espace latent résultant est introduit dans plusieurs têtes de prédiction chacune chargée de prédire une propriété. Suivant cette méthode d'évaluation, notre politique est comparée à une politique naïve entraînée à frapper l'objet par une approche RL classique.

Nous avons prouvé que la maximisation d'entropie que notre politique effectue, permet d'identifier les propriétés de masse, forme, frottement latéral et résistance au roulement en se basant uniquement sur des mesures relatives entre l'agent et l'objet. En comparant la politique avec un comportement naïf, nous avons montré que les observations acquises par notre politique sont plus riches pour une caractérisation des propriétés rendant les prédictions robustes à des variations comme du bruit dans les observations et un plus faible nombre d'interactions / exemples d'entraînement. Nous avons donc validé dans ce chapitre une politique intrinsèquement motivée qui, comme illustrée sur la Figure IV.16, agit sur l'objet pour observer une diversité de comportements qui pourraient permettre de modéliser la dynamique de l'objet. Nous allons dans le chapitre suivant remplacer l'identification supervisée des propriétés par une modélisation non-supervisée de l'objet pour ainsi proposer une boucle d'interaction-perception réalisable en totale autonomie et sans connaissance a priori.

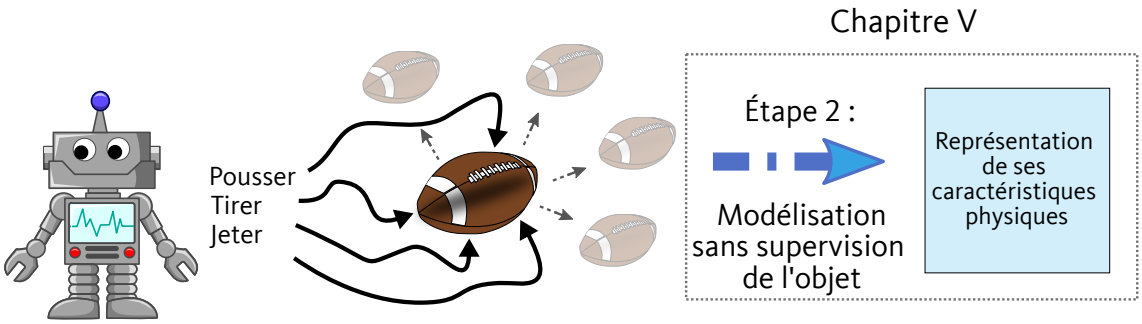


FIGURE IV.16 – Description de la phase d'inférence de la politique.

La solution que nous avons proposée peut encore évoluer, notamment pour prendre en compte l'encombrement potentiel de l'environnement. Il est par exemple possible d'indiquer ces éléments à l'agent par un ajout dans la carte égocentrique. Seulement, il sera probablement nécessaire de rajouter de nouvelles capacités d'action à l'agent pour lui permettre de débloquer un objet coincé contre un autre élément. Nous pourrions également évaluer d'autres signaux de motivation intrinsèque pour entraîner l'agent, en particulier des signaux basés sur la nouveauté ou encore de la famille "Knowledge-based" liés à la notion de savoir de l'agent sur son environnement. Nous avons préféré dans ces travaux fournir un cycle complet d'interaction, extraction puis utilisation plutôt que d'approfondir chacune de ces trois phases. L'avantage est de pouvoir désormais remplacer l'un de ces trois bloc par de nouveaux modèles et ré-évaluer le nouveau cycle ainsi obtenu. Concernant une application sur un système réel, nous avons montré que l'agent peut collecter des données dont la mesure est imparfaite sans trop impacter la performance du modèle d'identification. Le bruit pose cependant un problème sur l'entraînement de la politique avec le signal de motivation choisi. En effet, si les données sur lesquelles l'entropie est mesurée sont bruitées, alors l'agent est récompensé sans même avoir à bouger, ce qui peut impacter l'apprentissage et conduire à un comportement sous-optimal. La question du transfert et de l'encombrement sont des perspectives qui, une fois résolues, rendront la méthode fonctionnelle en milieu réaliste qui répondra parfaitement à la problématique initiale.

Une partie des résultats présentés dans le chapitre ainsi que la méthode qui a mené à π_{MEPOL} ont été valorisés par la publication d'un article au workshop "IROS 2022 Workshop on Mobile Manipulation and Embodied Intelligence (MOMA)" [Chareyre et al., 2022]. Une présentation orale de ces travaux a également été effectuée dans cette même conférence, suite à une sélection parmi les meilleures publications.

CHAPITRE V

Découverte et estimation de propriétés d'objets à partir de trajectoires d'interaction

Sommaire

V–1 Introduction

Dans le chapitre précédent, nous avons proposé d'utiliser de l'apprentissage par renforcement intrinsèquement motivé par une forme de maximisation d'entropie pour apprendre des comportements d'interactions avec des objets inconnus. Nous avons validé que les interactions obtenues contiennent alors plus de régularités corrélées aux propriétés des objets que ce que d'autres approches auraient permis. Plus précisément, un prédicteur des propriétés entraîné de manière supervisée à partir de ces interactions obtient de meilleures performances qu'à partir d'interactions suscitant moins d'entropie d'une part, ou cherchant simplement à entrer en contact avec les objets d'autre part. Cependant, ce premier travail, illustré sur la Figure V.1, montre seulement que les observations obtenues sur les objets contiennent l'information pour inférer leurs propriétés. Seulement, il ne permet pas d'extraire les propriétés sans supervision.

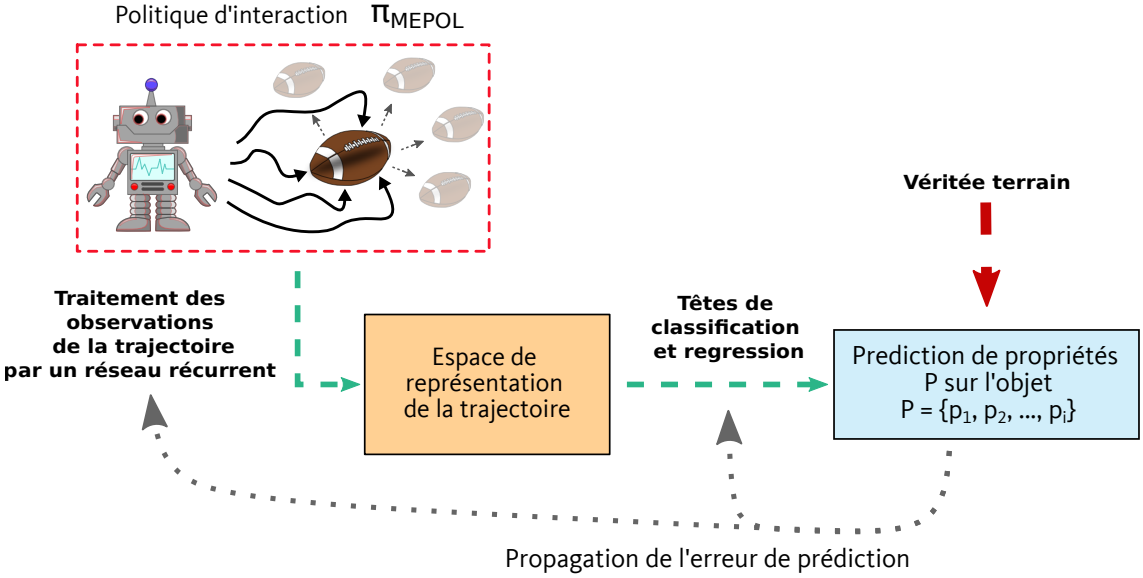


FIGURE V.1 – Méthode d'identification des propriétés mise en place dans le Chapitre IV pour valider la qualité des interactions.

Dans ce chapitre, nous traitons le problème initial d'extraction non-supervisée des propriétés de l'objet à partir des trajectoires de l'agent (voir Figure V.2). Nous proposons d'identifier les propriétés d'un objet comme les composantes principales d'un espace latent appris par un modèle auto-supervisé de prédiction des états futurs de l'objet le long des trajectoires générées au chapitre précédent. Ce chapitre s'intéresse à l'architecture la plus pertinente à mettre en place pour forcer l'espace latent à retenir les caractéristiques de l'objet.

Dans les sections qui suivent, nous décrivons tout d'abord la méthode de prédiction auto-supervisée élaborée pour créer un espace latent des caractéristiques dynamiques de l'objet. Les espaces obtenus lors de l'inférence ne sont cependant pas interprétables

tels quels, ce qui ne permet pas de vérifier si les propriétés sont présentes dans cet espace. Il est donc proposé, dans cette même section, une méthode d'évaluation du modèle par la mesure d'un score de corrélation entre les composantes du vecteur latent et les propriétés physiques de l'objet pouvant être continues ou discrètes. Dans une troisième section, nous appliquons la méthode à notre dispositif expérimental déjà présenté dans le Chapitre IV–3. Nous présentons plusieurs adaptations d'architectures proposées dans la littérature pouvant servir de base au modèle de notre méthode. Enfin, nous concluons ce chapitre par une comparaison des performances de chaque modèle sur leur capacité à créer cet espace pour fournir une représentation explicite des propriétés de l'objet. Nous cherchons dans cette dernière section à valider l'hypothèse que nous avons proposée lors du chapitre précédent : notre politique, qui suit un comportement maximisant l'entropie, doit permettre de faire émerger une meilleure représentation latente des propriétés des objets. Pour valider cette hypothèse, nous allons comme dans le Chapitre IV comparer la politique π_{MEPOL} à notre politique naïve π_{PPO}, mais aussi à des politiques menant à de plus faibles niveaux d'entropie tout en faisant varier certaines propriétés des trajectoires.

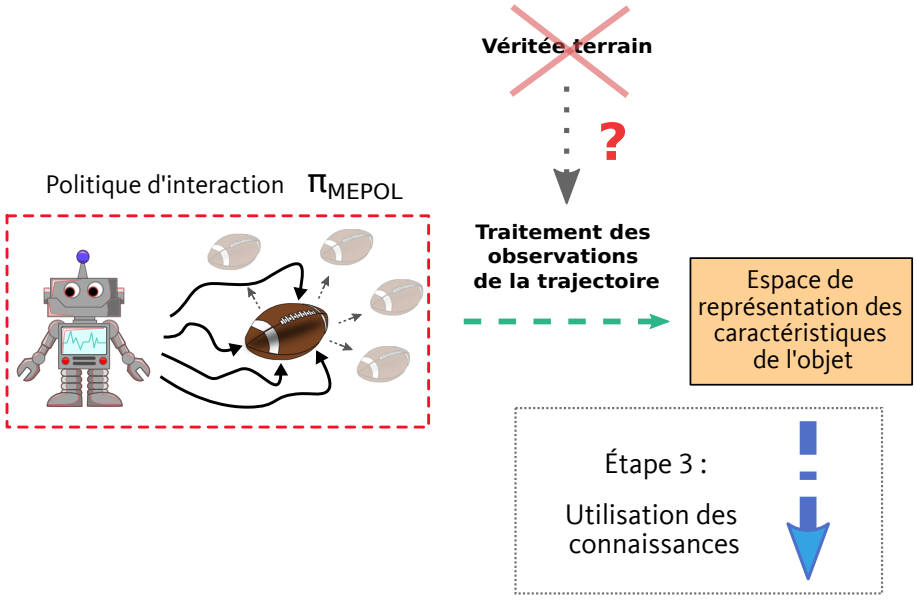


FIGURE V.2 – Objectif de la méthode développée dans ce chapitre. En utilisant les observations acquises lors des interactions de l'agent, nous cherchons à entraîner un modèle de perception capable de construire sans supervision un espace de représentation des caractéristiques de l'objet. Cet espace sera ensuite utilisé comme aide pour des tâches de manipulation.

V–2 Obtention d'un espace de représentation de l'objet à partir de trajectoires

Dans cette section, nous proposons une formulation générique du problème d'obtention d'une représentation latente des propriétés physiques de l'objet. Dans un premier temps, nous introduisons la méthode d'apprentissage permettant d'établir un espace de représentation puis, dans un second temps, la méthode d'évaluation permettant de vérifier si l'espace contient les caractéristiques physiques de l'objet.

V–2.1 Construction d'un espace de représentation par une tâche de prédiction auto-supervisée

Considérons un agent équipé de la politique π_{MEPOL}. Il se trouve à proximité d'un seul objet de propriétés physiques $P = \{p_1, p_2, ..., p_l\}$ inconnues, pouvant être continues ou discrètes. La politique est utilisée pour générer un nombre K de trajectoires face à ces objets aux propriétés aléatoires pouvant avoir été rencontrés ou non durant l'entraînement de la politique. Un ensemble de données D est ainsi obtenu où chaque trajectoire τ^k est une succession d'observations $\mathbf{x}_t^k$ acquises à pas de temps réguliers t. Pour chaque objet, l'agent va effectuer une seconde trajectoire $\tau^{k'}$ qui diffère de la première et former ainsi un ensemble D'. Nous cherchons ensuite à résoudre la tâche illustrée sur la Figure V.3.

Par chaque trajectoire τ^k de D, un encodeur extrait un vecteur $\mathbf{z}^k$ à partir de l'ensemble d'observations $\{\mathbf{x}_0^k, \mathbf{x}_1^k, ..., \mathbf{x}_T^k\}$ (phase 1). Puis en utilisant une trajectoire $\tau^{k'}$ de D' réalisée face au même objet, un modèle doit prédire l'ensemble des observations $\mathbf{x}_{t+1}^{k'}$ à partir de $\mathbf{x}_t^{k'}, t \in [0, T-1]$, l'action $\mathbf{a}_t^{k'}, t \in [0, T-1]$ qui va être effectuée et $\mathbf{z}^k$ (phase 2). Nous comparons ensuite la prédiction $\hat{\mathbf{x}}_{t+1}^{k'}$ avec l'observation réelle $\mathbf{x}_{t+1}^{k'}$ par le calcul de la somme des erreurs quadratiques moyennes (ou *Total Mean Squared Error - TMSE*), définie à l'Eq. (V.1). Les paramètres de l'encodeur et du modèle de prédiction, sont mis à jour de sorte à minimiser cette erreur de prédiction.

$$f_{Loss} = \frac{1}{N} \sum_{n}^{N} \sum_{t}^{T} \sum_{j}^{J} (\hat{\mathbf{x}}_{t,j}^{k'_n} - \mathbf{x}_{t,j}^{k'_n})^2 \tag{V.1}$$

où $\hat{\mathbf{x}}_{t,j}^{k'}$ est la $j^{\text{ème}}$ composante de l'observation $\mathbf{x}_t = [x_{t,0}, x_{t,1}, x_{t,J}]^\intercal$, T la longueur de la trajectoire k' et N correspond à la taille du batch.

L'idée d'effectuer la prédiction sur des observations qui ne sont pas issues de la trajectoire introduite dans l'encodeur est inspirée des travaux de Zheng et al. [2018]. De cette manière, nous forçons le vecteur $\mathbf{z}$ à contenir des informations liées à l'objet et sa

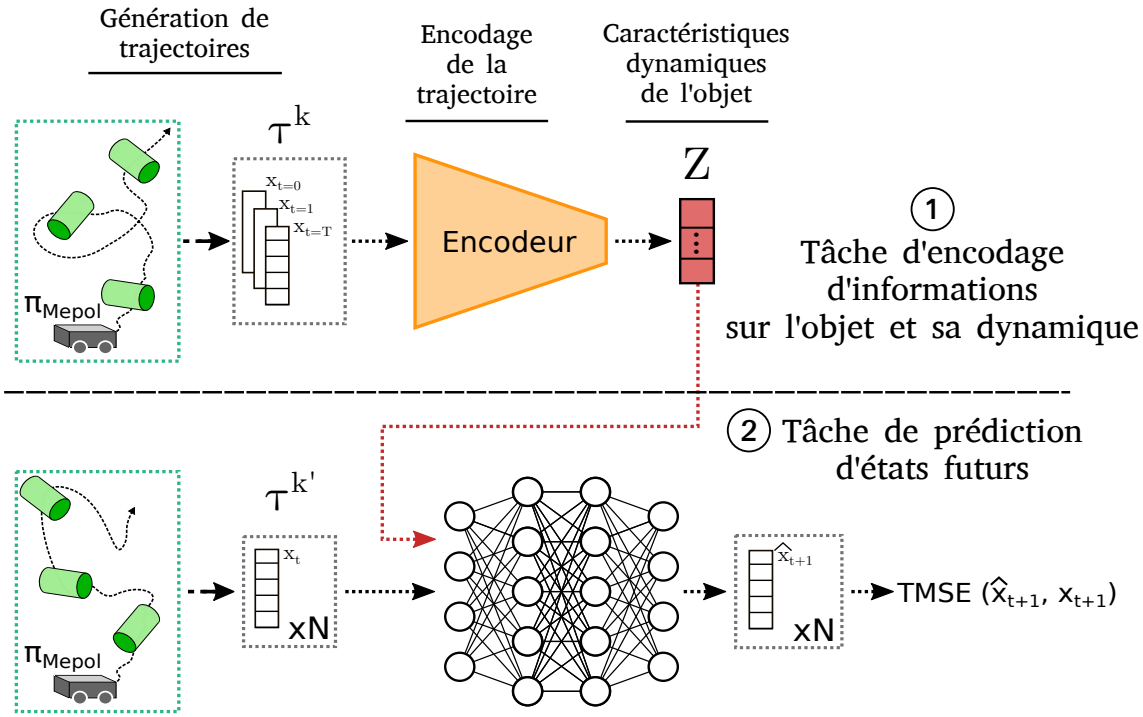


FIGURE V.3 – Architecture générale dédiée à l'apprentissage d'un espace latent $\mathbf{z}$ contenant les caractéristiques dynamiques d'un objet (phase 1). Pour forcer cet espace à contenir les propriétés de l'objet, un second modèle effectue une prédiction de l'état futur $\mathbf{x}_{t+1}^{k'}$ du même objet dans une toute nouvelle trajectoire $\tau^{k'}$ à partir de l'état présent $\mathbf{x}_t^{k'}$ et de $\mathbf{z}$ (phase 2).

dynamique plutôt que des caractéristiques de la trajectoire. Nous faisons l'hypothèse que si aucune connaissance sur les caractéristiques dynamiques de l'objet n'est donnée dans $\mathbf{x}_t^{k'}$, alors la prédiction des états suivants est possible seulement si $\mathbf{z}$ apporte ces informations manquantes. Contrairement à [Zheng et al., 2018], la prédiction est effectuée uniquement pour l'état suivant et non pas pour la trajectoire complète en partant de $\mathbf{x}_0^{k'}$ ce qui nécessite d'inclure un réseau récurrent.

Durant l'apprentissage, nous mesurons la TMSE sur un ensemble de prédictions issues de trajectoires de validation qui constituent les ensembles D_{val} et D'_{val}. Nous sauvegardons ainsi les paramètres optimaux avant sur-apprentissage du modèle et mettons fin à la phase d'apprentissage.

V–2.2 Caractérisation des propriétés d'un objet

Une fois le modèle entraîné, la caractérisation des propriétés d'un objet nécessite uniquement l'utilisation de l'encodeur (phase 1) et se passe de la tâche de prédiction et du prédicteur associé (Figure V.3). Dans cette phase, l'agent effectue sa trajectoire face à un objet puis fournit ses observations à l'encodeur qui renvoie un vecteur $\mathbf{z}$. Ce vecteur doit contenir des informations liées aux propriétés de l'objet que l'agent pourra utiliser comme connaissances pour d'autres tâches. Seulement, à cet instant,

rien n'indique que **z** contienne les propriétés et nous ne savons pas sous quelle forme elles sont représentées. Est-ce que chaque variable de **z** peut être associée à une propriété de l'objet ? Est-ce une représentation moins explicite où l'ensemble du vecteur est une modélisation de l'objet ? Il est aussi probable que plusieurs variables encodent la même information sur l'objet et que la taille de **z** joue un rôle dans les capacités à retenir de l'information. Si l'espace est trop petit, il ne pourra pas contenir toutes les propriétés pertinentes et, s'il est trop grand, les informations pourrait être redondantes.

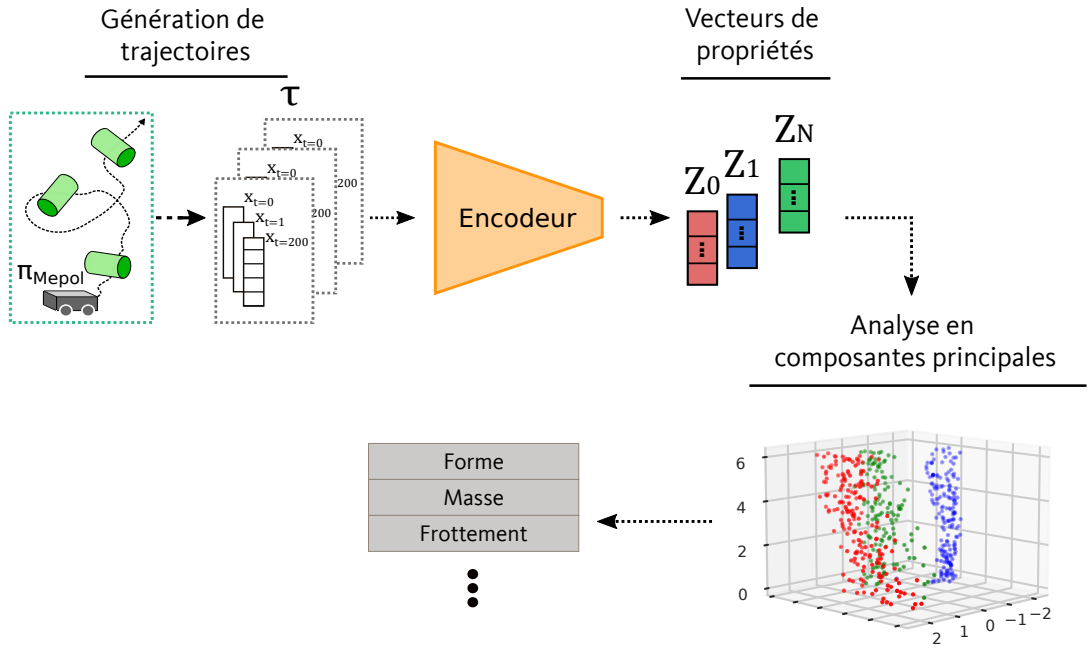


FIGURE V.4 – Illustration du processus d'obtention des propriétés d'objets après un encodage d'un ensemble de N trajectoires dans des espaces **z** suivi d'une analyse en composantes principales.

De façon à limiter une redondance inter-composantes dans **z**, nous appliquons comme dans [Lu et al., 2019, Zheng et al., 2018] une analyse en composantes principales pour compresser l'information de **z** dans un nouvel espace $\mathbf{z}_{PCA}$ (voir Figure V.4). Nous obtenons ainsi un vecteur où chaque variable est susceptible d'être associée à une propriété de l'objet ce qui facilite son extraction en connaissant la relation entre la composante et la propriété. Pour appliquer la PCA, nous générerons un ensemble de vecteurs **z** à partir d'un ensemble de trajectoires face à des objets de propriétés P choisies aléatoirement. Nous formons ainsi un ensemble de données D_{test} contenant les vecteurs **z**. Pour le besoin de l'évaluation uniquement (section suivante), sont également conservées les propriétés réelles des objets rencontrés. La PCA est utilisée sur D_{test} donnant lieu à un ensemble contenant les vecteurs $\mathbf{z}_{PCA}$ où chaque vecteur doit être une caractérisation des propriétés d'un objet.

V–2.3 Évaluation de la représentation par la mesure d'un score de corrélation

La difficulté est maintenant d'analyser le contenu de $\mathbf{z}$ ou $\mathbf{z}_{PCA}$ sachant que les propriétés peuvent être continues (masse, frottement, ...) ou discrètes (forme, mobile, ...). Le but est de pouvoir ensuite évaluer la qualité de la représentation selon la politique utilisée pour collecter les observations, mais aussi de comparer la performance selon l'architecture du modèle mis en place.

V–2.3.a Estimation de propriétés continues

Pour les propriétés continues, nous investiguons de potentielles relations monotones entre chaque composante de $\mathbf{z}_{PCA}$ et chaque propriété (notons par exemple z^1_{PCA} la première composante de $\mathbf{z}_{PCA}$). Les valeurs seraient alors interprétables par l'homme : "Plus $z^1_{PCA} \in \mathbf{z}_{PCA}$ est grand, plus l'objet est lourd" ou encore "Plus $z^5_{PCA} \in \mathbf{z}_{PCA}$ est faible, plus l'objet est glissant". La solution la plus simple est de faire l'hypothèse que la relation est linéaire en mesurant le coefficient de corrélation linéaire de Pearson. Seulement, rien ne nous indique que c'est le cas. Pour des variations de masse entre quelques grammes et quelques kilos, les différences de déplacements entre 2 pas de temps sont suffisamment grandes pour estimer la masse au travers d'une variable. Pour des masses bien plus élevées, par exemple de 20 ou 40 kilos, il est possible que les objets ne puissent quasiment pas être déplacés. Dans ce cas, la composante concernée de $\mathbf{z}_{PCA}$ notée z^{xx}_{PCA}, peut encoder des valeurs identiques alors que les deux masses sont bien différentes. La relation est alors non linéaire. Dans [Zheng et al., 2018], une corrélation linéaire est mesurée entre chaque composante de $\mathbf{z}_{PCA}$ et la propriété. Cependant, pour la masse, les auteurs mesurent la corrélation entre la composante de l'espace et le logarithme de la masse (sans donner une explication).

Plutôt que de chercher une loi affine entre la composante et une fonction de la propriété, nous avons choisi d'utiliser une mesure de corrélation indiquant la monotonie de la distribution. En statistique, deux mesures sont souvent utilisées : la corrélation de Spearman et le tau de Kendall. La corrélation de Spearman mesure la corrélation ordinale (l'ampleur n'est pas importante, seul le rang l'est) entre les variables X et Y. Le tau de Kendall utilise le même principe mais la différence réside dans le traitement des valeurs des rangs obtenues à partir des variables X et Y. Le tau de Kendall est en général plus strict et moins facile à interpréter que la corrélation de Spearman qui reste proche de la corrélation de Pearson. Nous choisissons dans la suite d'utiliser le coefficient de Spearman pour étudier des propriétés continues. Dans ce cas, plus la valeur est proche de 1 ou -1 plus la corrélation est forte.

V–2.3.b Identification de propriétés discrètes

Pour l'identification des propriétés discrètes, nous utilisons l'algorithme de partitionnement non-supervisé *k-means*. Nous cherchons à savoir s'il existe une composante principale z_{PCA}^{xx} pour laquelle appliquer un k-means produit des groupements (ou *clusters*) évidents, associés à chacune des propriétés discrètes. Bien que nous appliquions l'algorithme à une seule composante z_{PCA}^{xx} à la fois, nous pouvons étendre son utilisation à plusieurs composantes $\mathbf{z}_{PCA}$ pour traiter des propriétés plus complexes. La limite de cette méthode d'identification est la nécessité de définir l'hyper-paramètre k indiquant le nombre de groupes. Dans notre situation, cela signifierait par exemple avoir des connaissances à priori sur le nombre de formes différentes que l'agent peut rencontrer. Cette situation favorable n'est pas supposée ici. La solution est donc de déterminer le nombre optimal de clusters. Il existe deux approches classiques : la méthode *Elbow* ou le *Silhouette Score*.

La méthode Elbow nécessite de calculer pour différentes valeurs de k la valeur WCSS pour *Within Cluster Sum of Squares*. Ce score mesure la distance moyenne au carré entre les points d'un groupe et leur centroïde respectif, voir Eq. (V.2). Il faut ensuite tracer le score WCSS (ordonnée) en fonction du nombre de clusters k (abscisse). Plus k va augmenter, plus le score va diminuer. Le nombre de cluster optimal se situe alors au point d'inflexion de la courbe. Il est néanmoins parfois difficile de déterminer ce point d'inflexion, voir Figure V.5 à droite. La méthode n'est donc pas optimale pour déterminer k, d'autant plus qu'elle demande une analyse visuelle des courbes pour chaque composante z_{PCA}.

$$\text{WCSS} = \sum_{j}^{k} \frac{1}{n_{c_j}} \sum_{x \in c_j} (x - \mu_j)^2 \tag{V.2}$$

où k est le nombre de clusters, c_j le set de points associés au cluster j, n_{c_j} le nombre de points dans le set c_j et μ_j le centroïde du cluster j.

L'autre méthode pour le choix de k, en général privilégiée, est celle du *Silhouette Score*. Le coefficient de silhouette, voir Eq. (V.3), mesure la proximité entre un point i d'un groupe et les points des groupes voisins. Cette mesure varie entre -1 et 1, 1 étant le meilleur et -1 le pire score. Le k optimal correspond donc à celui pour lequel la moyenne des scores S_i est la plus élevée (voir Figure V.6).

$$S_i = \frac{b_i - a_i}{\max(a_i, b_i)} \tag{V.3}$$

où a_i représente la distance moyenne entre le point i et tous les autres points du même cluster et b_i représente la distance moyenne entre le point i et tous les points du cluster le plus proche de son propre cluster.

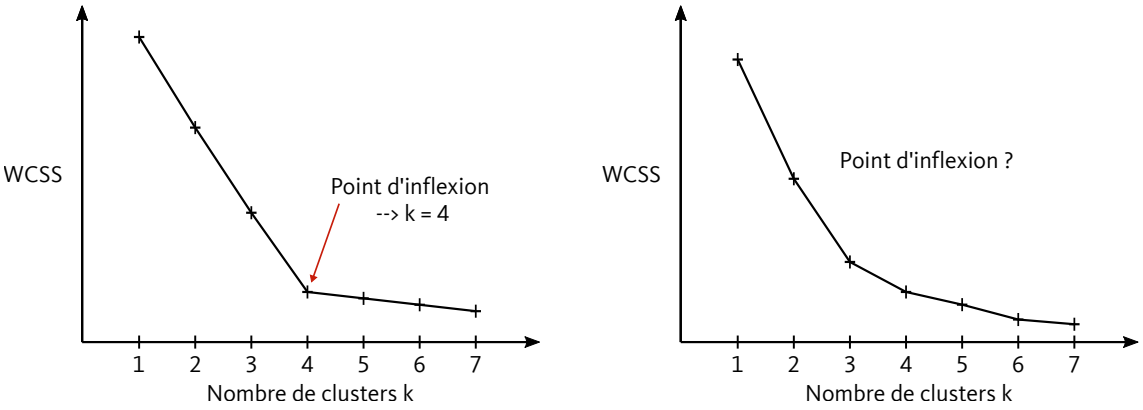


FIGURE V.5 – Identification du paramètre k optimal à l'aide de la méthode Elbow. A gauche, le point d'inflexion est bien visible et donne $k = 4$. A droite, le point d'inflexion est difficilement identifiable.

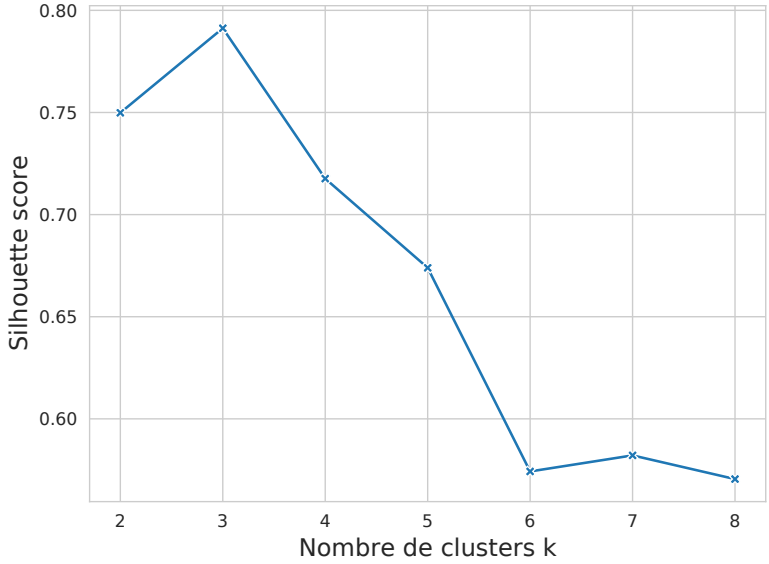


FIGURE V.6 – Identification du paramètre k optimal à l'aide du Silhouette Score. Dans ce cas, $k_{optimal} = 3$.

V–2.3.c Métrique d'évaluation pour quantifier la performance d'un modèle

Nous disposons maintenant d'une méthode pour analyser le contenu de l'espace latent et découvrir des propriétés discrètes ou continues. Dans l'optique d'une comparaison des performances des architectures des modèles mis en place, une métrique de comparaison est nécessaire.

Pour un modèle donné, nous calculons un *score de classification* pour chaque propriété discrète et un *score de corrélation de Spearman* pour chaque propriété continue. Le score global du modèle est obtenu en calculant la moyenne des scores de classification et de corrélation préalablement normalisés entre 0 et 1. Cependant, une propriété peut ne pas être définie pour un objet, comme par exemple la propriété de résistance

au roulement pour un cube. Si nous considérons que la variable z^i_{PCA} encode l'information de cette résistance au roulement pour tous les objets en possédant, alors le score de corrélation peut être faussé par les points du cube. Nous calculons donc le score de corrélation de Spearman de manière indépendante selon le type de l'objet. De cette manière, nous nous assurons que les propriétés de chaque objet peuvent être encodées dans l'espace latent. En disposant d'un set de test comportant les vérités terrain des objets, l'estimation des propriétés d'un objet rencontré est possible quelle que soit l'exploration du moment qu'en phase de test une relation monotone, ou des groupements évidents par rapport à une propriété physique, ait pu être trouvé. Dans le cas d'un environnement totalement inconnu, les vérités terrains peuvent ne pas être à notre disposition. Dans cette situation, il n'y a pas de moyen de s'assurer que l'espace **z** contient des propriétés, autrement qu'en utilisant cet espace pour résoudre des tâches et en évaluant si cet apport rend la résolution de la tâche plus simple et/ou efficace.

V–3 Mise en place d'un ensemble d'architectures pour l'évaluation de la méthode sur un robot mobile

Dans les sections qui vont suivre, nous allons instancier la méthode dans notre environnement de simulation et proposer diverses versions d'architectures pouvant répondre à la tâche de prédiction d'observations.

V–3.1 Conditions expérimentales et entraînement

L'environnement dans lequel cette phase de découverte non supervisée des propriétés est effectuée est identique à celui mis en place lors de l'apprentissage de la politique (illustré à nouveau sur la Figure V.7). L'agent et l'objet sont placés avec des positions et orientations aléatoires. Les propriétés de l'objet sont une nouvelle fois tirées de manière uniforme dans leurs espaces de validité, donnés dans le Tableau IV.1. L'agent suit le comportement appris durant T pas de temps de la première phase, et conserve à chaque pas de temps les observations x_t qui contiennent les mêmes mesures que celles utilisées lors de la validation de la politique avec un modèle supervisé :

$$\mathbf{x}_{t-1} = [e^{(-r^{t-1})}, \cos\theta^{t-1}, \sin\theta^{t-1}, d_x^{t-1}, d_y^{t-1}, v_x^{t-1}, v_y^{t-1}, c^{t-1}, v_a^{t-1}, v_l^{t-1}, a_{av}^{t-1}, a_{ro}^{t-1}]^\intercal$$

L'observation à prédire est alors :

$$\mathbf{x}_t = [e^{(-r^t)}, \cos\theta^t, \sin\theta^t, d_x^t, d_y^t, v_x^t, v_y^t, c^t, v_a^t, v_l^t]^\intercal$$

Dans la méthode, nous indiquons que la prédiction s'effectue sur une nouvelle tra-

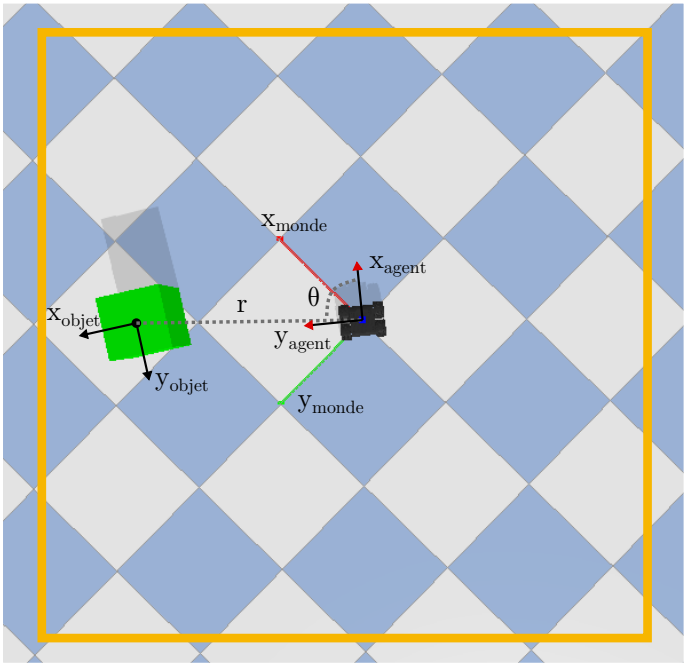


FIGURE V.7 – Environnement aux conditions initiales. Le carré orange représente la zone d'apparition possible de l'objet.

jectoire différente de celle ayant servi à obtenir $\mathbf{z}$. En pratique, nous générons avec la politique une trajectoire suffisamment grande que nous découpons ensuite en deux parties de longueurs égales $T' = 200$ en laissant plusieurs pas de temps entre les deux sections de la trajectoire. Après environ 30 pas de simulation, nous considérons que l'agent s'est suffisamment éloigné du dernier état de la première section pour considérer une nouvelle trajectoire. Aussi, afin d'éliminer la période où l'agent se déplace en direction de l'objet et donc n'effectue aucune interaction, nous ignorons les 20 premiers pas de temps de la trajectoire. Ainsi, nous avons sur une même trajectoire : $\mathbf{x}_t^{encodage}, t \in [20, T' + 20 - 1]$ et $\mathbf{x}_t^{prediction}, t \in [T' + 20 + 30, 2T' + 20 + 30 - 1]$ (voir Figure V.8). Une trajectoire complète correspond donc à $T = 2T' + 20 + 30 + 1 = 451$ pas de temps pour être en mesure de comparer la dernière prédiction $\hat{\mathbf{x}}_{450}$. Avec ces observations, nous constituons les sets de données D_{train} et D_{val} contenant respectivement 2000 et 500 trajectoires. De même, 500 trajectoires d'observations $\mathbf{x}_t^{encodage}$ sont utilisées pour obtenir $\mathbf{z}$ une fois le modèle entraîné et constituer ainsi D_{test}. Nous choisissons arbitrairement $\mathbf{z}$ de dimension 15 que nous réduisons ensuite à une dimension $dim(\mathbf{z}_{PCA}) = 8$ par l'application de l'analyse en composantes principale. Nous étudierons l'influence de la taille de l'espace $\mathbf{z}$ en fin de chapitre. La seconde portion d'observations $\mathbf{x}_t^{prediction}$ est utilisée pour la partie prédiction des modèles où chaque pas sert à prédire uniquement le suivant. Cela signifie que 200 erreurs sont moyennées pour chaque trajectoire et propagées sur l'ensemble du modèle. Nous ajoutons à cela un batch de taille 8 pour éviter des instabilités lors de l'apprentissage si une des

trajectoires est de mauvaise qualité.

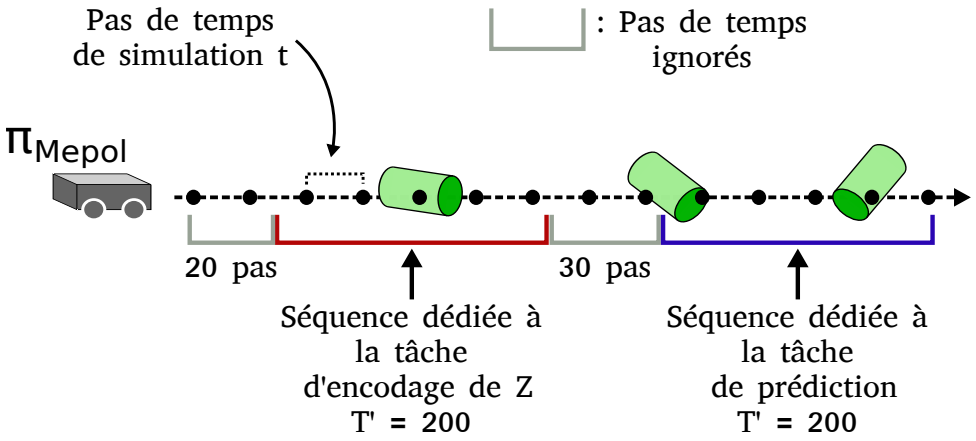


FIGURE V.8 – Découpage d'une trajectoire de l'agent pour obtenir la séquence d'observations $\mathbf{x}_t^{encodage}$ (représentée en rouge) dédiée à l'encodage des caractéristiques de l'objet, et la séquence d'observations $\mathbf{x}_t^{prediction}$ (représentée en bleu) dédiée à la tâche de prédiction.

Afin de vérifier si les éléments de $\mathbf{z}$ sont reliés à des propriétés de l'objet, nous sauvegardons les propriétés réelles de masse, forme, résistance au roulement et frottement latéral des trajectoires de D_{test}. L'ensemble des observations est dans un premier temps supposé acquis sans bruits sur les mesures. Nous réalisons l'entraînement du modèle sur un maximum de 3000 epochs et conservons les meilleurs paramètres par early-stopping en mesurant l'erreur la plus faible sur le set D_{val}. Nous avons choisi d'évaluer la qualité de la représentation à partir des scores obtenus avec 5 modèles entraînés sur 5 sets D_{train} différents. Cela nous permet d'évaluer la variance du score selon les trajectoires utilisées lors de l'apprentissage du modèle. Ces 5 modèles sont évalués sur le même D_{val} et D_{test}. Nous sommes, de cette manière, certains que le modèle généralise bien à l'ensemble des trajectoires issues d'une même politique.

V–3.2 Modèles de représentation des propriétés de l'objet

Nous proposons maintenant diverses architectures, compatibles avec notre méthode, au travers d'adaptations de modèles de l'état de l'art : une variante de l'architecture de perception-prédiction issue de [Zheng et al., 2018], et sa version auto-encodeur variationnel issue de [Lu et al., 2019], ainsi qu'une architecture basée sur l'utilisation d'une cellule d'attention inspirée de modèles de la famille des Neural Processes [Kim et al., 2019]. Enfin, nous essayerons d'extraire directement les caractéristiques de l'objet depuis la mémoire de la politique π_{MEPOL} entraînée Chapitre IV.

V–3.2.a Architecture de perception-prédiction

Le premier modèle que nous mettons en place est une adaptation de l'architecture perception-prédiction [Zheng et al., 2018], détaillée précédemment dans la section état de l'art. Pour rappel, la version originale utilise un réseau particulier prenant en compte les interactions entre objets, et construit en parallèle une mémoire de la trajectoire en introduisant dans le réseau traitant les observations un vecteur initialisé aléatoirement au premier pas de temps. Notre version de l'architecture ainsi que les paramètres des réseaux sont donnés sur la Figure V.9.

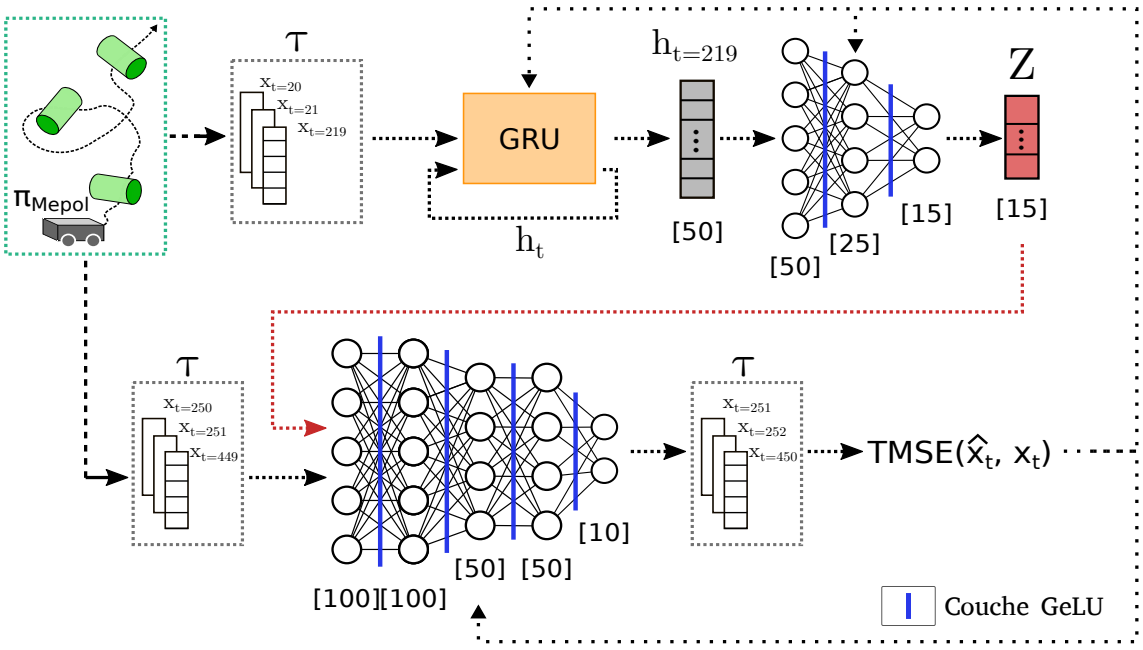


FIGURE V.9 – Détails de l'architecture perception-prédiction.

La première partie de la trajectoire $\{\mathbf{x}_{20}, \mathbf{x}_{21}, ..., \mathbf{x}_{219}\}$ est encodée à l'aide d'une cellule récurrente GRU. Le dernier état caché $\mathbf{h}_{219}$ est ensuite introduit dans un réseau de neurones entièrement connectés pour réduire sa dimension et obtenir le vecteur latent $\mathbf{z}$. L'obtention de $\mathbf{z}$ correspond à la partie du modèle nommée "perception".

La partie "prédiction" est modélisée par un simple réseau dense qui prend comme entrée une seule observation combinant $\mathbf{x}_t, t \in [250, 449]$ et $\mathbf{z}$ pour prédire seulement l'état suivant $\mathbf{x}_{t+1}$. A chaque epoch, toutes les observations de la seconde section de la trajectoire sont introduites dans le réseau. Les paramètres de la cellule GRU et des deux réseaux denses sont mis à jour par une descente de gradient avec optimiseur Adam, en calculant la fonction de coût donnée Eq. (V.1). Nous utilisons un learning rate de 10^{-4}. Le modèle appris avec cette architecture est nommé π^{PP}_{MEPOL} ou π^{PP}_{PPO} selon la politique utilisée pour générer les observations.

V–3.2.b Architecture AttnPP

Nous proposons sur la Figure V.10 une version alternative du modèle Perception-Prédiction en remplaçant la cellule GRU par une cellule d'auto-attention pour encoder les observations de la trajectoire. Cette idée est inspirée des méthodes de Neural Processes et plus particulièrement de la version Attentive Latent Neural Processes. Tout comme un processus gaussien, où la loi *a priori* de Y sachant X est définie à partir d'échantillons $(\mathbf{x}_i, \mathbf{y}_i)$, nous allons introduire les observations de la trajectoire par les couples de points de contexte $(\mathbf{x}_t, \mathbf{x}_{t+1}), t \in [20, 219]$ pour estimer le point cible $\mathbf{x}_{t+1}, t \in [250, 449]$ sachant les points de contexte et l'observation $\mathbf{x}_t, t \in [250, 449]$. Les méthodes de Latent Neural Processes n'utilisent pas directement les échantillons de contexte pour prédire la cible. Le principe est de passer par un espace latent gaussien $\tilde{\mathbf{z}}$, extrait des entrées à l'aide d'un réseau de neurones, et duquel sont échantillonnées une moyenne et une variance. L'avantage du réseau de neurones est de pouvoir traiter des données de grandes dimensions ainsi que les cas de corrélation temporelle entre les échantillons (ce que ne permettent pas les processus gaussien). Dans notre cas, nous allons simplement construire un vecteur non stochastique à partir des points de contexte et considérer ce $\tilde{\mathbf{z}}$ comme notre espace de caractéristiques des propriétés de l'objet $\mathbf{z}$.

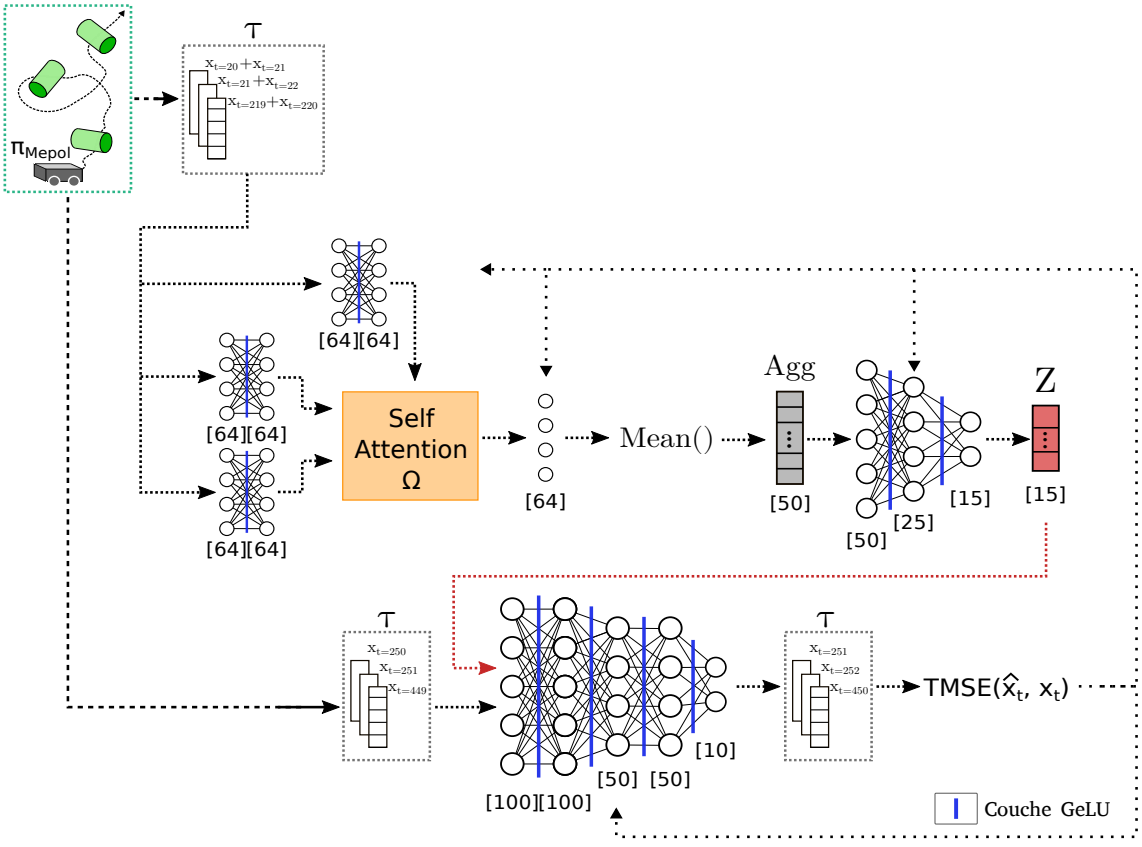


FIGURE V.10 – Détails de l'architecture Attentive Perception-Prediction.

Pour obtenir cet espace, nous utilisons une cellule d'auto-attention où les entrées *Query*, *Key* et *Value* sont les couples $(\mathbf{x}_t, \mathbf{x}_{t+1}), t \in [20, 219]$ de la première portion de la trajectoire (Figure V.10). Une fois la sortie de cette cellule passée dans une couche dense puis une fonction invariante aux permutations (ici la moyenne) pour obtenir le vecteur latent $\mathbf{z}$ de dimension 15. La suite du processus reste inchangée par rapport à celui du Perception-Prédiction. La fonction de coût est uniquement la TMSE donnée à l'Eq. (V.1). Nous noterons $\pi^{AttPP}_{\text{MEPOL}}$ ou π^{AttPP}_{PPO} les modèles issus de cette architecture.

V–3.2.c Architecture autoencodeur variationnel

Le modèle VAE (voir Figure V.11) inspiré de [Lu et al., 2019], est quasiment identique à celui de Perception-Prédiction décrit à la section V–3.2.a. La différence se situe dans la construction de l'espace latent Z qui ne correspond plus à la sortie d'un réseau entièrement connecté. À la place, la série d'observations $\{\mathbf{x}_{20}, \mathbf{x}_{21}, ..., \mathbf{x}_{219}\}$ est utilisée pour prédire un vecteur composé de moyennes μ_z associées à des variances σ_z^2. Chaque élément de $\mathbf{z}$ est ensuite échantillonné suivant une loi normale $\mathcal{N}(\mu_z, \sigma_z^2)$. Cela permet d'imposer des contraintes de régularisation sur l'espace latent. Cette technique est bénéfique pour plusieurs raisons selon [Lu et al., 2019]. Elle augmente l'indépendance entre chaque variable de l'espace, et minimise l'utilisation des variables latentes, forçant le modèle à conserver les données les plus importantes. Un défaut de l'échan-

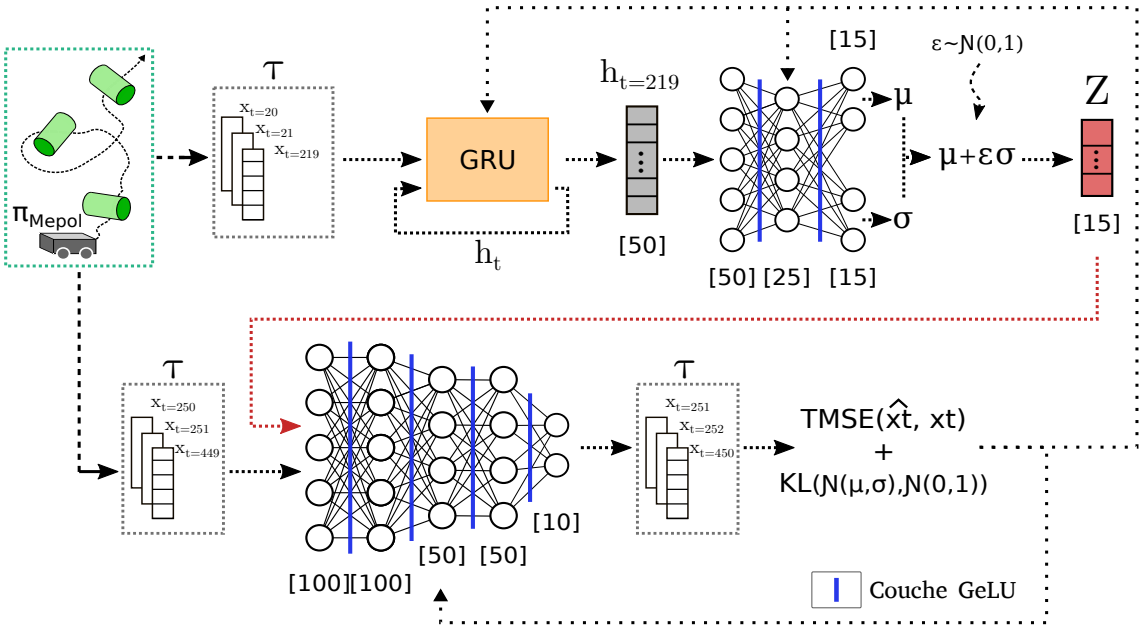


FIGURE V.11 – Détails de l'architecture VAE.

tillonnage des moyennes et variances est que ce processus n'est pas différentiable, ce qui rend l'étape de propagation des gradients impossible à effectuer. Le problème peut

être résolu par l'astuce du reparamétrage VAE en prenant :

$$\mathbf{z} = \mu_z + \epsilon \cdot \sigma_z^2 \tag{V.4}$$

où $\epsilon \sim \mathcal{N}(0,1)$ est échantillonné de manière indépendante pour chaque élément de $\mathbf{z}$.

La suite de l'architecture reste inchangée. L'ensemble des paramètres est mis à jour avec une fonction de coût définie à l'Eq. (V.5) à laquelle un terme de régularisation VAE est ajouté.

$$f_{Loss} = \frac{1}{N}\sum_{n}^{N}\sum_{t=251}^{450}\sum_{j}^{J}(\hat{\mathbf{x}}_{t_n,j_n} - \mathbf{x}_{t_n,j_n})^2 + \sum_{z_n=0}^{14} D_{KL}(\mathcal{N}(\mu_{z_n}, \sigma_{z_n}^2)||\mathcal{N}(0,1)) \tag{V.5}$$

où $\mathbf{x}_j$ est la $j^{\text{ème}}$ composante de l'observation $\mathbf{x}$, $\mathbf{z} = [z_0, z_1, ..., z_{14}]^\intercal$ est l'espace latent des caractéristiques de l'objet, et N correspond à la taille du batch.

Le terme de régularisation issue d'une divergence de Kullback-Leibler D_{KL}, contraint la distribution $\mathcal{N}(\mu_z, \sigma_z^2)$ à approcher une loi normale centrée réduite $\mathcal{N}(0,1)$. Les hyper-paramètres d'apprentissage restent également inchangés. En phase d'évaluation, nous prenons $\mathbf{z} = \mu_z$ et effectuons la découverte et l'extraction des propriétés physiques de l'objet. Notons π^{VAE}_{MEPOL} ou π^{VAE}_{PPO} le modèle ainsi entraîné.

V–3.2.d Architecture AttnLNP

Nous avons également souhaité comparer les performances de notre modèle $AttPP$ avec la version originale de l'Attentive Latent Neural Process [Kim et al., 2019]. Cependant, nous pensons que cette architecture n'est pas adaptée à notre problème. En effet, dans la reproduction de l'AttnLNP que nous mettons en place (voir Figure V.12), les points de contextes sont utilisés pour construire deux espaces en parallèle. Un chemin stochastique créant un espace latent $\mathbf{z}$ dont la méthode d'obtention est similaire à celle de notre modèle $AttPP$ avec l'utilisation d'une cellule d'auto-attention Ω. Un chemin déterministe, qui après avoir passé les points de contextes dans une cellule d'auto-attention Φ, réalise une cross-attention entre la sortie de l'auto-attention et le point $x = \mathbf{x}_t, t \in [250, 449]$. La sortie de cette cross-attention est notée r^*. La prédiction de la cible $y = \mathbf{x}_{t+1}, t \in [250, 449]$ est ensuite réalisée à partir des points cibles $x = \mathbf{x}_t, t \in [250, 449]$, le vecteur latent $\mathbf{z}$ et l'espace r^*. Le fait d'avoir un chemin déterministe ajoute la possibilité de retenir plus d'informations concernant la première partie de la trajectoire et rien n'empêche les caractéristiques de l'objet d'être réparties dans les espaces r^* et $\mathbf{z}$. Il serait alors difficile d'obtenir un seul espace de faible dimension contenant l'ensemble des propriétés de l'objet. Il est par contre possible que la reproduction de l'état cible y soit de très bonne qualité. Cette version de l'état de

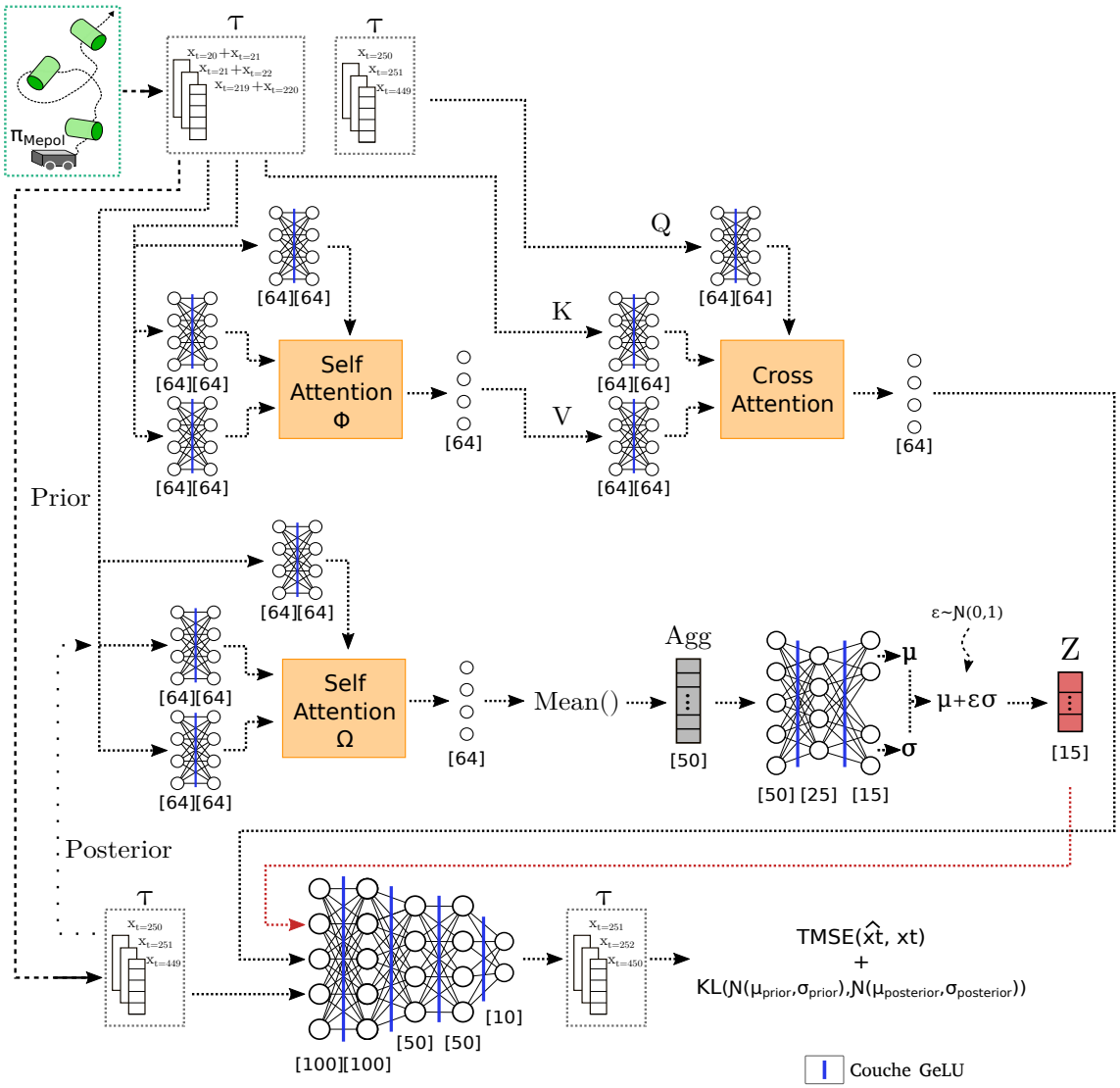


FIGURE V.12 – Détails de l'architecture Attentive Neural Process.

l'art des Neural Processes apporte en théorie une augmentation des performances de prédiction, de la vitesse d'entraînement ainsi que de l'éventail de fonctions qui peuvent être modélisées.

La fonction de coût est une somme de la TMSE avec la divergence de Kullback-Leibler entre la distribution prior $\mathcal{N}(\mu_{prior}, \sigma^2_{prior})$ des points de contexte et la distribution posterior $\mathcal{N}(\mu_{posterior}, \sigma^2_{posterior})$ issue des points cibles :

$$\begin{aligned} f_{Loss} = & \frac{1}{N} \sum_{n}^{N} \sum_{t=251}^{450} \sum_{j}^{J} (\hat{\mathbf{x}}_{t_n,j_n} - \mathbf{x}_{t_n,j_n})^2 \\ & + \sum_{z_n=0}^{14} D_{KL}(\mathcal{N}(\mu_{z_n,prior}, \sigma^2_{z_n,prior}) || \mathcal{N}(\mu_{z_n,posterior}, \sigma^2_{z_n,posterior})) \end{aligned} \tag{V.6}$$

En phase d'évaluation, nous prenons $\mathbf{z} = \mu_z$. Notons $\pi^{ALNP}_{\text{MEPOL}}$ ou π^{ALNP}_{PPO} le modèle ainsi entraîné.

V–3.2.e Utilisation de la mémoire politique

Nous concluons cette section par une méthode qui s'éloigne de la stratégie d'obtention de $\mathbf{z}$ que nous avons décrite dans la section précédente. Lors de l'entraînement de la politique, les propriétés de l'objet peuvent avoir été implicitement apprises dans l'espace latent de la mémoire de la politique. Une proposition est de récupérer l'espace $\mathbf{h}_T$ issue de la cellule GRU de la politique une fois la trajectoire complétée. Cet espace constitue la représentation de l'objet sur laquelle nous effectuons l'analyse en composante principale. Bien que nous soyons peu confiants sur la présence d'information liée à la dynamique de l'objet, nous souhaitons nous assurer que ce n'est pas un moyen plus simple pour obtenir une représentation de l'objet.

Ainsi, nous établissons un ensemble D_{test} contenant 500 espaces $\mathbf{h}_T$ sur lesquels nous appliquons le même traitement que sur les vecteurs $\mathbf{z}$ issus des architectures précédentes. Notons π^{Pol}_{MEPOL} ou π^{Pol}_{PPO} ce processus d'obtention de $\mathbf{z}$.

V–4 Analyse de la qualité de la représentation de l'objet selon la politique et le modèle de perception utilisé

Dans cette section, nous évaluons la qualité des représentations des propriétés d'objets obtenues par notre méthode en comparant les performances selon la politique suivie, mais aussi selon le modèle de perception utilisé. La première sous-section compare les modèles de perception lorsque les observations utilisées pour les entraîner proviennent de notre politique π_{MEPOL} apprise en Chapitre IV. Ensuite dans une seconde sous-section, nous choisissons le modèle donnant les meilleurs résultats et évaluons le score de corrélation moyen lorsque celui-ci est entraîné avec des observations de politiques menant à des entropies de plus faibles valeurs. Une troisième sous-section compare l'ensemble des modèles de perceptions lorsqu'une politique naïve est utilisée, telle que mise en place au Chapitre IV pour générer des observations. Enfin, nous concluons cette section par une étude de l'influence du bruit, de la longueur de la trajectoire, ainsi que de la taille de l'espace de représentation $\mathbf{z}$.

TABLE V.1 – Analyse de l'espace latent $\mathbf{z}_{PCA}$ issu de différentes architectures pour une même politique π_{MEPOL}. La première partie du tableau donne les résultats moyens après entraînement de 5 modèles sur la même architecture. La seconde partie du tableau donne les corrélations du modèle, noté $\pi_{\text{MEPOL}}*$, ayant obtenu le meilleur score.

	Masse			Frottement latéral			Résistance au roulement			Forme	**Score**
Politique	Cylindre	Cube	Sphère	Cylindre	Cube	Sphère	Cylindre	Cube	Sphère		
π^{Pol}_{MEPOL}	0.43±0.05	0.38±0.04	0.33±0.06	0.33±0.05	0.55±0.07	-	0.20±0.05	-	0.51±0.08	62±6	0.42±0.02
$\pi^{ALNP}_{\text{MEPOL}}$	0.37±0.14	0.35±0.14	0.40±0.09	0.24±0.05	0.39±0.16	-	0.13±0.03	-	0.44±0.13	46±6	0.35±0.08
π^{VAE}_{MEPOL}	0.45±0.10	0.66±0.03	0.62±0.04	0.37±0.06	0.71±0.04	-	0.29±0.08	-	0.86±0.01	55±4	0.56±0.02
$\pi^{AttPP}_{\text{MEPOL}}$	**0.76**±0.04	0.70±0.13	0.78±0.06	**0.60**±0.06	**0.92**±0.03	-	0.51±0.12	-	0.87±0.06	90±7	0.75±0.04
π^{PP}_{MEPOL}	0.69±0.13	**0.71**±0.08	**0.85**±0.10	0.56±0.05	0.88±0.02	-	**0.62**±0.09	-	**0.87**±0.04	**92**±7	**0.76**±0.05
$\pi^{Pol}_{\text{MEPOL}}*$	0.46	0.41	0.33	0.30	0.58	-	0.20	-	0.59	67	0.44
$\pi^{ALNP}_{\text{MEPOL}}*$	0.52	0.48	0.47	0.32	0.62	-	0.12	-	0.58	43	0.44
$\pi^{VAE}_{\text{MEPOL}}*$	0.58	0.65	0.65	0.32	0.70	-	0.41	-	0.86	59	0.60
$\pi^{AttPP}_{\text{MEPOL}}*$	0.73	0.79	0.79	**0.66**	**0.95**	-	0.56	-	0.89	**97**	0.79
$\pi^{PP}_{\text{MEPOL}}*$	**0.87**	**0.79**	**0.93**	0.57	0.87	-	**0.66**	-	**0.90**	96	**0.82**

TABLE V.2 – Erreur de reproduction TMSE calculée pour chacune des architectures. Moyenne et variance calculées sur D_{test} pour 5 entraînements.

	Architecture utilisée			
	PP	*AttPP*	*ALNP*	*VAE*
TMSE	$0.240_{\pm 0.001}$	$0.238_{\pm 0.001}$	$0.230_{\pm 0.002}$	$0.299_{\pm 0.002}$

V–4.1 Identification des propriétés physiques avec la politique issue de MEPOl

Les résultats donnés dans le Tableau V.1 donnent une comparaison de la capacité de chaque architecture à encoder dans $\mathbf{z}_{PCA}$ les caractéristiques de l'objet s'ils utilisent des observations de π_{MEPOL}. Pour chaque forme d'objet possible, le score de corrélation le plus élevé de l'ensemble des éléments z^i_{PCA} est fourni. Comme 5 entraînements sont effectués en parallèle pour une même architecture, nous donnons dans ce tableau les moyennes et variances de la corrélation et la classification la plus élevée. Entre deux entraînements, la plus forte corrélation peut être sur une composante z^i_{PCA} différente. C'est la raison pour laquelle nous prenons la corrélation la plus forte et non pas la corrélation moyenne sur une composante. Pour chaque architecture, nous indiquons par une étoile $*$, les résultats du modèle ayant obtenu le meilleur score global. Afin de faciliter la lecture de ces résultats, nous allons, au fil de l'analyse, extraire dans des sous-tableaux (V.3, V.4) les éléments majeurs. Nous fournissons en parallèle le Tableau V.2 qui indique l'erreur de reconstruction TMSE pour chaque modèle sur le set D_{test}. Ici encore nous donnons la moyenne et la variance de la mesure sur les 5 entraînements effectués. Enfin, la Figure V.14 est une projection $2D$ de certaines des composantes z^i_{PCA} choisies comme étant celles les plus corrélées avec la propriété de forme et l'une des propriétés continue. Sur un axe vertical est ajoutée la vérité terrain de cette propriété continue. La couleur indique la véritable forme de l'objet. Ce mode de représentation permet de visualiser de potentiels groupes d'objet et les corrélations avec les propriétés.

Une sélection des corrélations et des taux de classification moyens (Tableau V.3) montre que les architectures *PP* et *AttPP* sont capables de révéler les propriétés cachées des objets avec des valeurs de corrélation généralement au-delà de 0.7. Seuls, ces deux modèles permettent de distinguer, avec peu d'erreur, la forme de l'objet. L'architecture *VAE* obtient un score total moyen 20% en dessous des deux premières architectures. Ce modèle montre cependant des bonnes capacités à retenir certaines caractéristiques comme la résistance au roulement de la sphère (0.86), le coefficient de frottement du cube (0.71) ou encore des corrélations moyennes pour la masse (0.58). Les scores restent tout de même trop faibles pour confirmer une relation monotone

avec les propriétés. De plus, la forme n'est pas identifiable. Notre approche *Pol* montre

TABLE V.3 – Score de corrélation moyen obtenu pour chaque modèle.

	Architecture utilisée				
	PP	*AttPP*	*VAE*	*ALNP*	*Pol*
Score total	0.76±0.05	0.75±0.04	0.56±0.02	0.35±0.08	0.42±0.02
Masse cylindre	0.69±0.13	0.76±0.04	0.45±0.10	0.37±0.14	0.43±0.05
Masse cube	0.71±0.08	0.70±0.13	0.66±0.03	0.35±0.14	0.38±0.04
Masse sphère	0.85±0.10	0.78±0.06	0.62±0.04	0.40±0.09	0.33±0.06
Frottement cube	0.88±0.02	0.92±0.03	0.71±0.04	0.39±0.16	0.55±0.07
Roulement sphère	0.87±0.04	0.87±0.06	0.86±0.01	0.44±0.13	0.51±0.08
Forme	92±7	90±7	55±4	46±6	62±6

que la mémoire latente de la politique ne contient pas les propriétés dynamiques de l'objet avec lequel l'agent interagit. Cependant, nous observons une légère capacité à différencier la forme de l'objet avec un taux de classification moyen de 62% et dans le meilleur cas un taux de 67%. C'est un résultat logique puisque l'agent va interagir différemment selon l'objet étant donné que leurs comportements sont très différents. Cela peut également être un indicateur de sur-apprentissage des formes d'objets rencontrés durant l'entraînement. Cet indicateur pourrait être pris en compte lors d'une future étude des capacités de généralisation de l'agent. Enfin, les faibles performances du modèle *AttLNP* ne sont pas surprenantes. En effet, comme indiqué plus tôt, la prédiction de l'état suivant s'effectue à partir d'un espace $\mathbf{z}$ et r^* dont l'ensemble facilite trop la tâche. Le fait d'avoir deux espaces implique que le modèle ne conserve pas uniquement le minimum d'informations utiles, ou alors la sépare dans les deux espaces.

Nous voyons pourtant dans le Tableau V.2 que le modèle *AttLNP* donne les meilleures prédictions sur la tâche de reconstruction là ou le modèle *VAE* a plus de mal que les autres. [1] Cela reflète que le terme de régularisation VAE à un impact négatif sur l'apprentissage et la génération de l'espace $\mathbf{z}$. Les modèles *AttPP* et *PP* affichent des résultats similaires avec une erreur de reconstruction proche de celle de *AttLNP*. Seulement, ici, la reconstruction se base uniquement sur $\mathbf{z}$ et l'état précédant, ce qui force le modèle à retenir les caractéristiques de l'objet et non pas des informations parasites de la trajectoire. Les meilleurs modèles $\pi^{PP}_{\text{MEPOL}}*$, $\pi^{AttPP}_{\text{MEPOL}}*$ atteignent des corrélations fortes avoisinant un coefficient entre 0.8 et 0.9. De plus, les taux de classification, 96% et 97%, issues de la séparation K-means, montrent que les formes d'objets

1. L'architecture *Pol* ne dispose pas de score TMSE étant donné que ce n'est pas un modèle entraîné sur la tâche de reproduction de l'état.

sont aisément identifiables. Le Silhouette score calculé pour $\pi^{PP}_{\text{MEPOL}}*$, Figure V.13, af-

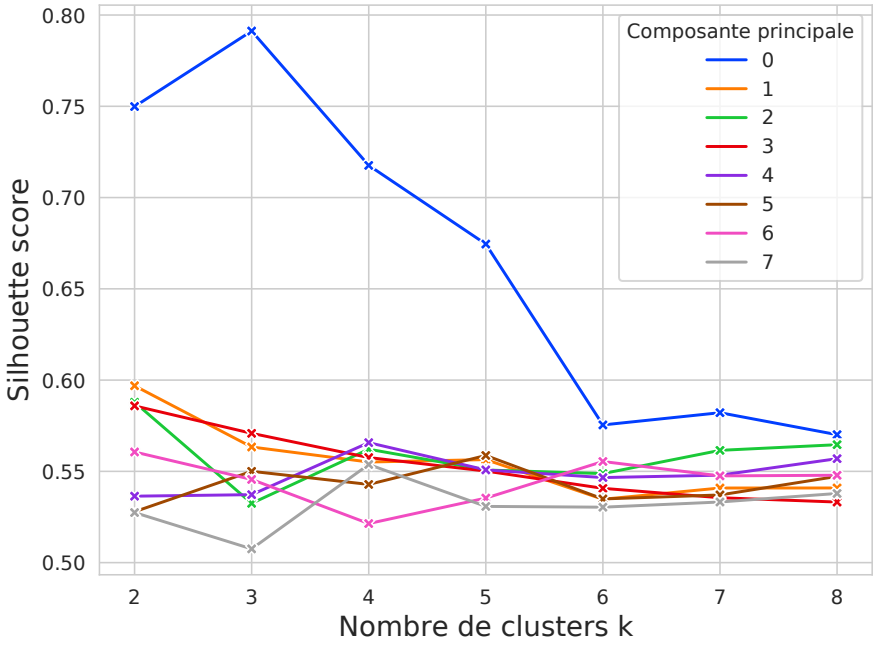


FIGURE V.13 – Calcul du Silhouette score sur chaque composante principale $\mathbf{z}_{PCA}$ obtenue avec $\pi^{PP}_{\text{MEPOL}}*$.

fiche en effet un pic pour un nombre de clusters $k = 3$,correspondant au nombre de formes. En utilisant cette approche k-mean et silhouette score, catégoriser les objets semble possible, sans aucune supervision, selon une propriété qui fait grandement varier leur comportement dynamique, ici la forme. Pour finir, notons de plus faibles

TABLE V.4 – Score de corrélation des propriétés continues pour le meilleur modèle de l'architecture PP.

	Masse	Frottement latéral	Résistance au roulement
Cylindre	0.87	0.57	0.66
Cube	0.79	0.87	-
Sphère	0.93	-	0.90

corrélations pour l'ensemble des modèles, sur les propriétés de frottement latéral et résistance au roulement concernant le cylindre (Tableau V.4). Les corrélations sont généralement entre 0.55 et 0.65 pour les architectures $AttPP$ et PP. Ce sont des propriétés très délicates à prédire puisqu'elles sont reliées à la masse et peuvent en plus se confondre entre elles à cause des observations utilisées. Pour rappel, nous utilisons uniquement des mesures de positions et de vitesses relatives et n'observons jamais visuellement l'objet. Il est donc, dans certains cas, impossible de savoir si l'objet se déplace parce qu'il glisse ou parce qu'il roule. Si le frottement latéral est trop faible alors

l'objet glissera tout le temps au lieu de rouler peu importe la direction de poussée. Ce phénomène peut se produire pour de nombreuses combinaisons (masse, frottement latéral, résistance au roulement) et est dû à la simulation et au choix des plages de valeurs. Notons que la corrélation avec la masse du cylindre est aussi impactée par ce phénomène (à plus petite échelle) avec des scores moyens souvent plus faibles.

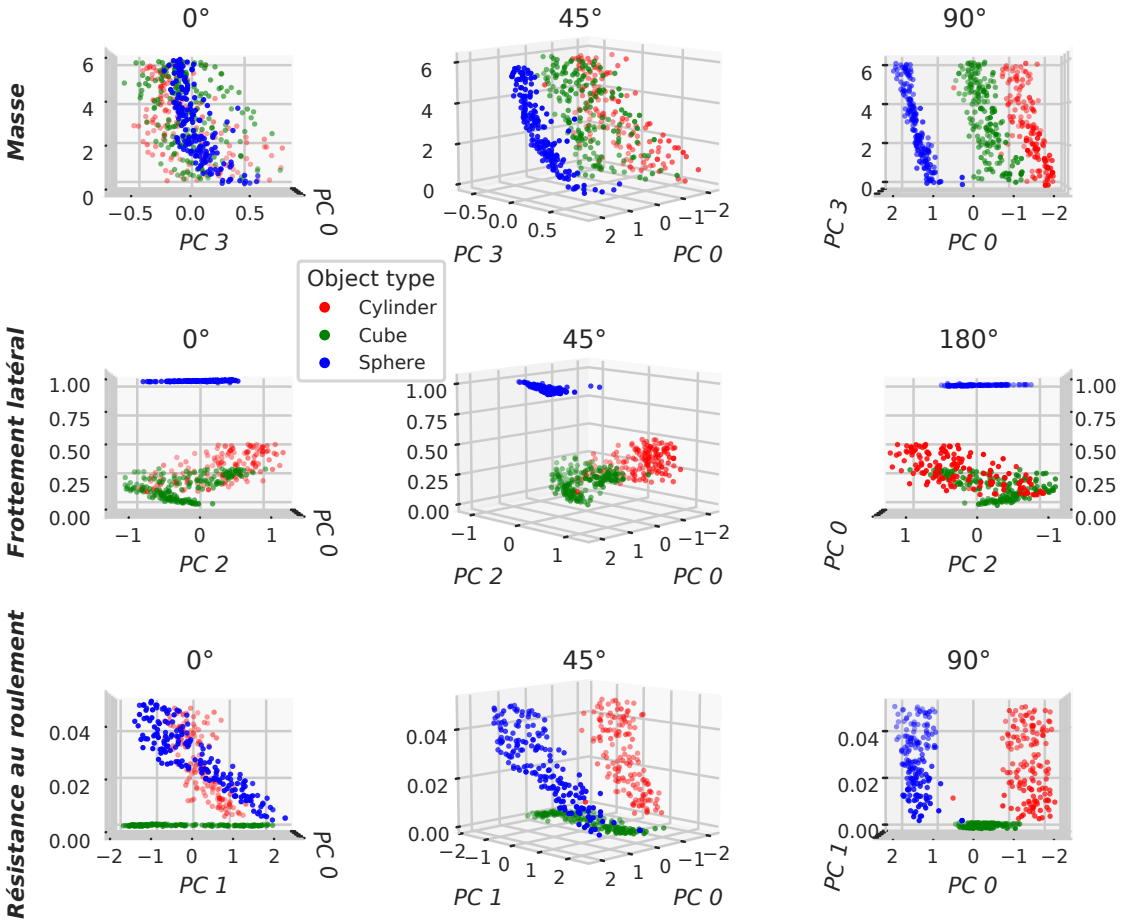


FIGURE V.14 – Projection 2D des composantes principales les plus corrélées avec la forme et l'une des propriétés continue. Chaque représentation d'une ligne est une vue différente de la même propriété. La propriété représentée est de haut en bas : la masse, le frottement latéral, la résistance au roulement. La politique utilisée ici est π_{MEPOL}.

Il est donné, au vu des résultats du Tableau V.1, un léger avantage de performance à la version perception-prédiction PP. Le nuage de points Figure V.14 est construit à partir des espaces latents obtenus avec π^{PP}_{MEPOL} sur l'ensemble de D_{test}. Nous avons ainsi une meilleure visualisation des résultats donnés dans le Tableau V.1. Un point flagrant ici, est que la composante z^0_{PCA} indique avec des séparations évidentes la propriété de forme. Ensuite, les composantes z^1_{PCA}, z^2_{PCA} et z^3_{PCA} semblent respectivement corrélées aux propriétés de résistance au roulement, frottement latéral et masse. Si

cela semble visuellement valide, la propriété de la masse du cylindre est plus fortement corrélée avec la composante z^2_{PCA} qu'avec z^3_{PCA}. De même, le frottement latéral du cube est plus corrélé avec z^1_{PCA} qu'avec z^2_{PCA}. La corrélation maximale d'une propriété avec chaque objet n'est donc pas obligatoirement mesurée sur la même composante. Cette corrélation reste tout de même forte (même si elle n'est pas maximale) sur la composante quel que soit l'objet. Nous pouvons par conséquent certifier que chacune des propriétés d'un objet est bien présente dans une des variables de l'espace latent. Notons que les propriétés cherchées sont toujours encodées dans les 4 premières composantes z^0_{PCA}, z^1_{PCA}, z^2_{PCA} et z^3_{PCA}. Ceci n'est pas un hasard puisque la PCA place dans les premières composantes les données les plus représentatives. Nous avons donc ici une preuve que la forme, la masse, le frottement latéral et la résistance au roulement sont les propriétés qui jouent le plus sur le comportement de l'objet.

V–4.2 Comparaison selon différents niveaux d'entropie de la politique MEPOL

Nous souhaitons maintenant montrer l'impact de la politique sur la capacité du modèle à générer un espace latent qui contient les propriétés de l'objet. Pour cela, nous réalisons des mesures sur l'influence du niveau d'entropie de la politique π_{MEPOL} et la capacité de la politique π_{PPO} à fournir des observations suffisamment riches pour obtenir un espace contenant les propriétés.

La Figure V.15 montre l'influence du niveau d'entropie de la politique obtenue avec MEPOL sur les performances globales obtenues avec l'architecture π^{PP}_{MEPOL}. Les mesures sont une nouvelle fois évaluées sur un seul D_{test} après 5 apprentissages sur des sets D_{train} différents. Nous observons un gain net de performance avec l'augmentation du niveau d'entropie. Le score total passe de 0.1 pour une politique complètement aléatoire à une valeur qui dépasse le score de 0.7 pour des entropies plus élevées. Ici encore, nous avons la preuve que la stratégie d'interaction de l'agent fournie des observations qui permettent de construire une représentation de l'objet de bien meilleure qualité lorsque l'entropie augmente sur l'espace sélectionné. Toutes les politiques ne sont donc pas équivalentes pour une compréhension efficace de l'environnement.

V–4.3 Comparaison avec la politique PPO

La question est maintenant de savoir si une politique qui interagit sans diversité avec l'objet suffit à obtenir une représentation de ce dernier. Pour rappel, dans le Chapitre IV, nous avons obtenu de très bonnes prédictions avec π_{PPO} lors d'un entraînement supervisé. Le Tableau V.5 est construit de la même manière que celui des résultats des modèles entraînés avec les observations de la politique π_{MEPOL}, mais cette fois-ci

TABLE V.5 – Analyse de l'espace latent $\mathbf{z}_{PCA}$ issue de différentes architectures pour une même politique π_{PPO}. La première partie du tableau donne les résultats moyens après l'entraînement de 5 modèles sur la même architecture. La seconde partie du tableau donne les corrélations du modèle, noté $\pi_{\text{PPO}}*$, ayant obtenu le meilleur score. Une comparaison est donnée avec les résultats du meilleur modèle π_{MEPOL}.

	Masse			Frottement latéral			Résistance au roulement			Forme	**Score**
Politique	Cylindre	Cube	Sphère	Cylindre	Cube	Sphère	Cylindre	Cube	Sphère		
π_{PPO}^{ALNP}	0.42±0.19	0.38±0.12	0.45±0.08	0.21±0.02	0.33±0.13	-	0.10±0.02	-	0.55±0.06	49±10	0.37±0.08
π_{PPO}^{VAE}	0.30±0.06	0.27±0.11	0.35±0.03	0.18±0.07	0.49±0.07	-	0.32±0.06	-	0.456±0.03	47±2	0.35±0.03
π_{PPO}^{AttPP}	0.41±0.05	0.44±0.05	0.45±0.02	0.25±0.06	0.81±0.02	-	0.32±0.06	-	0.76±0.03	52±3	0.49±0.02
π_{PPO}^{PP}	0.33±0.06	0.36±0.06	0.46±0.03	0.18±0.04	0.74±0.05	-	0.34±0.06	-	0.71±0.01	52±2	0.46±0.01
$\pi_{\text{PPO}}^{ALNP}*$	0.62	0.57	0.52	0.23	0.53	-	0.09	-	0.60	67	0.48
$\pi_{\text{PPO}}^{VAE}*$	0.33	0.43	0.33	0.26	0.60	-	0.27	-	0.48	45	0.39
$\pi_{\text{PPO}}^{AttPP}*$	0.42	0.51	0.45	0.25	0.79	-	0.36	-	0.81	51	0.51
$\pi_{\text{PPO}}^{PP}*$	0.41	0.40	0.50	0.13	0.67	-	0.42	-	0.71	51	0.47
$\pi_{\text{MEPOL}}^{PP}*$	**0.87**	**0.79**	**0.93**	**0.57**	**0.87**	-	**0.66**	-	**0.90**	**96**	**0.82**

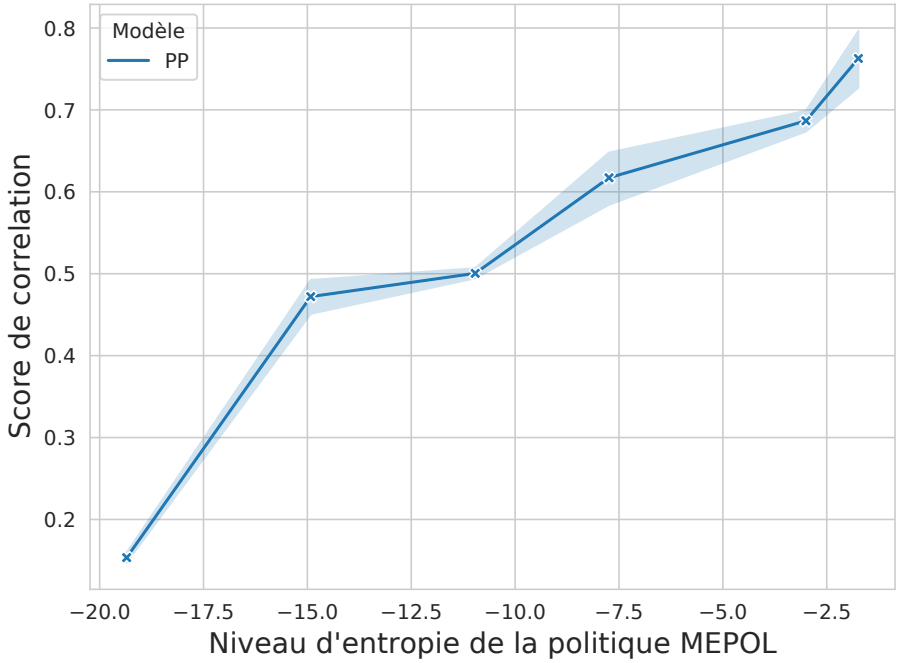


FIGURE V.15 – Mesure du score du modèle π^{PP}_{MEPOL} pour des politiques entraînées avec MEPOL menant à différents niveaux d'entropie.

TABLE V.6 – Erreur de reproduction TMSE calculée pour chacune des architectures avec des observations de π_{PPO}. Moyenne et variance calculées sur D_{test} pour 5 entraînements.

	Architecture utilisée			
	PP	$AttPP$	$ALNP$	VAE
TMSE	$0.134_{\pm 0.002}$	$0.143_{\pm 0.003}$	$0.147_{\pm 0.003}$	$0.152_{\pm 0.002}$

les observations sont acquises avec π_{PPO}. Il donne pour chaque architecture un score global qui est une moyenne des scores de corrélation avec les propriétés continues et du score de classification de la forme. Nous donnons ces résultats pour chaque architecture pour nous assurer qu'il n'existe pas une meilleure manière de traiter ce type d'information et que les résultats sont bien liés aux observations recueillies. Nous ajoutons, en dernière ligne de ce tableau, les performances du meilleur modèle $\pi^{PP}_{\text{MEPOL}}*$ issue de la politique π_{MEPOL}. Nous mettons en avant dans les tableaux V.7 et V.8, certains de ces résultats afin de gagner en lisibilité. Nous donnons également dans le Tableau V.6, l'erreur de reconstruction TMSE pour chaque modèle sur son set de test. Enfin, la Figure V.16 est une projection 2D des composantes z^i_{PCA} pour lesquelles les corrélations sont les plus fortes sur le meilleur modèle obtenu. Ces nuages de points sont à mettre en comparaison avec ceux Figure V.14 pour visualiser de potentielles différences dans la qualité de représentation que permettent ces deux types de politique.

Dans le cas de cette politique, le modèle (voir Tableau V.7) qui semble le plus appro-

TABLE V.7 – Score de corrélation moyen obtenu pour chaque modèle.

	Architecture utilisée			
	PP	*AttPP*	*VAE*	*ALNP*
Score total	0.46±0.01	0.49±0.02	0.35±0.03	0.37±0.08

prié pour identifier des propriétés est π^{AttPP}_{PPO} bien que les performances restent proches de π^{PP}_{PPO}. Cependant, les résultats montrent des écarts de scores moyens deux fois inférieurs à ceux obtenus avec π^{PP}_{MEPOL}. Les corrélations avec les propriétés continues sont en général toutes faibles sauf pour une ou deux propriétés données. Par exemple,

TABLE V.8 – Comparaison des scores entre le meilleur modèle issue de la politique π_{PPO} et le meilleur modèle obtenu avec les observations de π_{MEPOL}.

	$\pi^{PP}_{\text{PPO}}*$	$\pi^{PP}_{\text{MEPOL}}*$
Masse cylindre	0.42	0.87
Masse cube	0.51	0.79
Masse sphère	0.45	0.93
Frottement cylindre	0.25	0.57
Frottement cube	0.79	0.87
Roulement cylindre	0.36	0.66
Roulement sphère	0.81	0.90
Forme	51	96

pour le meilleur modèle $\pi^{PP}_{\text{PPO}}*$ de l'architecture *PP*, les corrélations sont inférieures à 0.5 sauf pour la résistance au roulement de la sphère et le frottement latéral du cube (Tableau V.8). Notons aussi que pour le cylindre, les scores sont bien plus faibles que les autres ce qui est logique puisqu'il est nécessaire de frapper l'objet selon deux directions orthogonales pour observer des roulements et glissements. Contrairement, au cas de prédiction supervisé mis en place au Chapitre IV, ici le modèle n'est pas capable d'identifier les 3 formes. Le taux de classification approche 50% ce qui est un peu plus élevé qu'un choix complètement aléatoire de 33%. La différence de précision entre le cas supervisé et non supervisé pour la politique π_{PPO} s'explique par plusieurs raisons. D'une part, dans le cas supervisé, le réseau pouvait s'appuyer sur la différence des distributions de certaines observations pour classifier l'objet (comme nous l'avons observé avec la distance au centre de l'objet). Dans le cas non-supervisé, comme l'agent effectue une trajectoire rectiligne, le modèle a moins besoin d'encoder dans $\mathbf{z}_{PCA}$ les propriétés de l'objet pour effectuer une bonne estimation de l'état suivant. Le modèle a juste besoin d'encoder des informations concernant la récursivité de la série temporelle qu'est la trajectoire, plutôt que de s'intéresser à l'objet en lui-même. L'erreur de

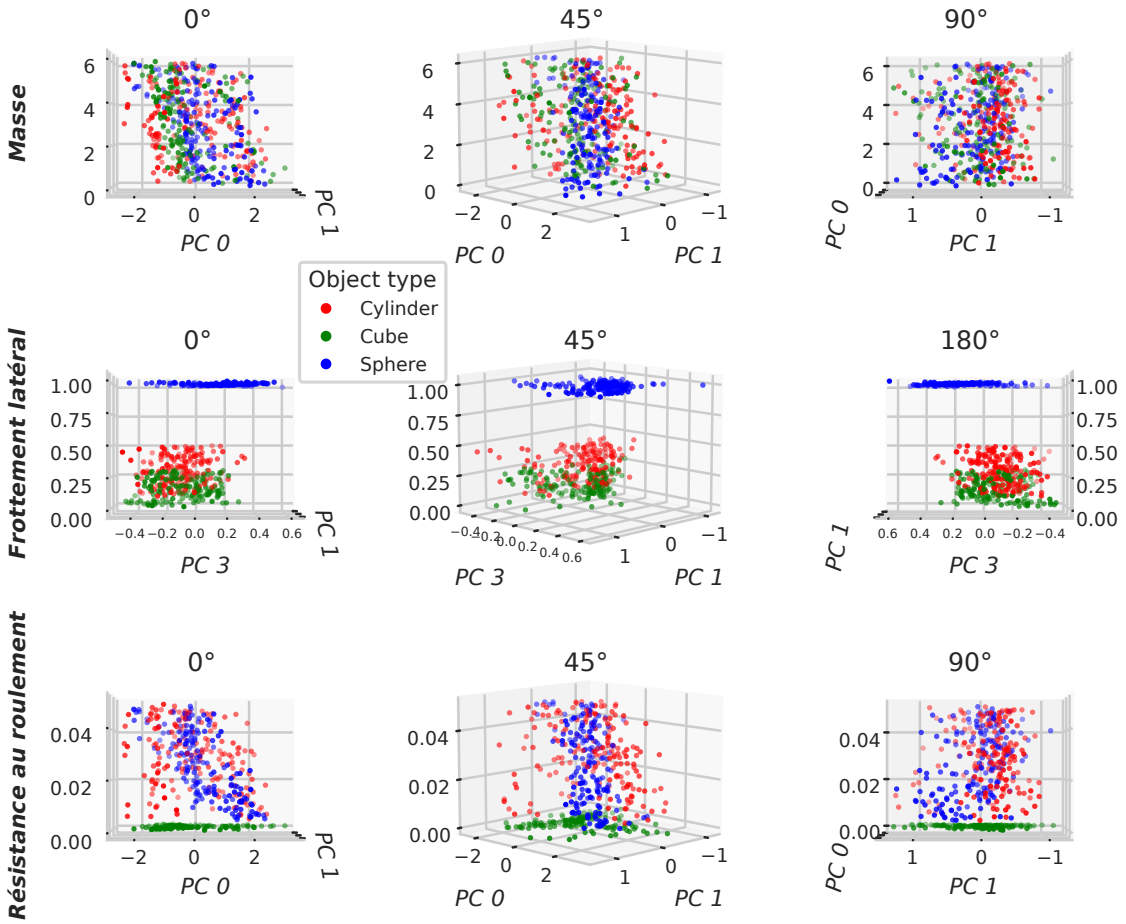


FIGURE V.16 – Projection 2D des composantes principales les plus corrélées avec la forme et l'une des propriétés continue. Chaque représentation d'une ligne est une vue différente de la même propriété. La propriété représentée est de haut en bas : la masse, le frottement latéral, la résistance au roulement. La politique utilisée ici est π_{PPO}.

reconstruction (Tableau V.6) est en moyenne très bonne en étant deux fois plus faible que les erreurs moyennes obtenues avec π_{MEPOL}.

Pour conclure, les nuages de points Figure V.16, issus des espaces générés par π_{PPO}^{AttPP}, illustrent l'incapacité du modèle à construire un espace représentatif de l'objet. Nous distinguons peu de tendances monotones pour les propriétés continues et il semble impossible de distinguer des groupements liés à la forme de l'objet. Là où pour π_{MEPOL} la forme apparaissait clairement sur la composante z^0_{PCA}, ici c'est la composante z^1_{PCA} qui permet le mieux de déterminer la forme. Cette composante de la forme n'est donc pas l'information primordiale retenue par l'analyse en composante principale. Le mélange des points de ces classes montre que ce n'est pas une information utile au modèle pour la prédiction du prochain état de l'objet. Notons tout de même que l'ensemble des propriétés restent les plus corrélées sur les 4 premières composantes z^0_{PCA},

z^1_{PCA}, z^2_{PCA} et z^3_{PCA}. Cependant, les corrélations d'une propriété continue ne sont jamais regroupées sur une même composante pour toutes les formes d'objets contrairement à ce qui peut être le cas lorsque la politique π_{PPO} est utilisée.

V–4.4 Étude de l'influence d'hyperparamètres sur la performance des meilleurs modèles.

Dans cette sous-section, nous étudions l'influence de certains paramètres comme la longueur de la trajectoire, le bruit d'observation et la dimension de l'espace latent **z**. Nous avons choisi d'évaluer la variation de la performance sur les modèles π^{PP}_{MEPOL} et $\pi^{AttPP}_{\text{MEPOL}}$, qui ont montré les meilleurs scores. Nous souhaitons ainsi repérer si l'une de ces architectures est plus sensible à certains de ces paramètres. Le score de corrélation est toujours calculé comme la moyenne de l'ensemble des corrélations et classifications normalisées entre $[0, 1]$. Nous réalisons cette mesure pour chacun des 5 entraînements effectués par architecture.

Variation de la dimension de z : Nous affichons, Figure V.17, les performances

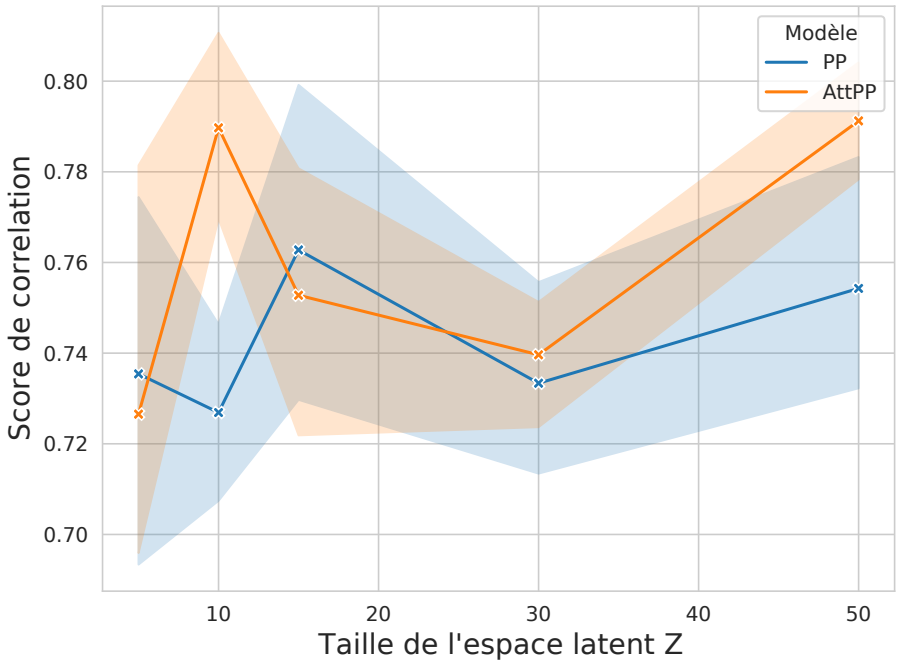


FIGURE V.17 – Score de corrélation du modèle PP (en bleu) et $AttPP$ (en orange) selon la taille de **z**. Moyenne pour 5 entraînements.

pour des tailles de **z** pouvant prendre les dimensions suivantes : $[5, 10, 15, 30, 50]$. Nous appliquons ensuite l'analyse en composantes principales, qui reste de taille fixe égale à 8 (5 dans le cas où $dim(\mathbf{z}) = 5$). Le fait d'appliquer la PCA sur le vecteur **z** fait que les scores sont tout de même évalués sur la même dimension final. Les

informations sur l'objet sont peut-être réparties sur un plus grand nombre de variables de **z** lorsque celui-ci est grand, mais une fois la PCA appliquée, ces informations sont regroupées et nous retrouvons la même qualité de représentation des caractéristiques de l'objet. Pour cette raison, nous avons aussi tracé, sur la Figure V.18, la variation du score en calculant les corrélations et scores de classification directement sur **z** sans passer par la PCA.

Dans le cas où nous utilisons l'analyse en composantes principales Figure V.17, nous distinguons peu de variations dans le score global pour l'architecture PP. Nous distinguons plus d'instabilité pour l'$AttPP$. Pour les deux architectures, si l'espace **z** est équivalent au nombre de propriétés, alors la performance diminue. Les plus fortes corrélations s'observent pour de plus faibles tailles de **z** : 10 pour l'architecture $AttPP$ et 15 pour l'architecture PP. Lorsque la taille de l'espace augmente, les informations des propriétés ne semblent pas perdues contrairement à ce que nous pensions. Nous pouvons imaginer qu'en augmentant la taille de **z**, nous donnons davantage de capacités au modèle pour "tricher" et encoder des relations par rapport à la série temporelle, or ce n'est pas le cas. Si nous regardons maintenant les scores lorsque les corrélations

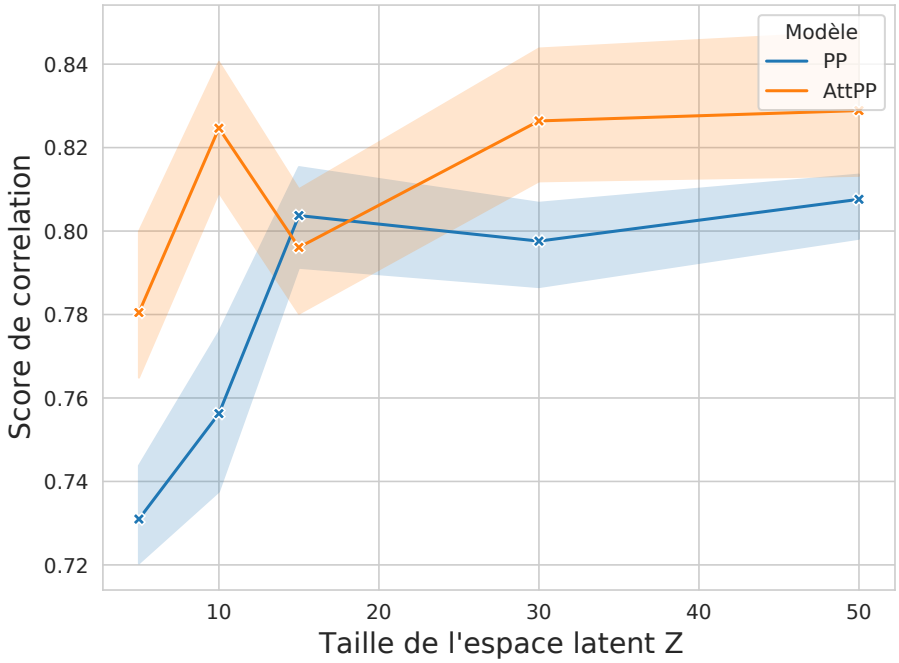


FIGURE V.18 – Score de corrélation du modèle (PP ou AttPP) selon la taille de **z**. Moyenne sur 5 entraînements de modèles.

sont directement mesurées sur **z** (Figure V.18), nous pouvons observer pour l'architecture PP, une tendance à augmenter jusqu'à une dimension de 15 puis à stagner. Pour le modèle $AttPP$, il y a moins de différence selon les dimensions de **z** mis à part lorsque sa dimension est trop faible. Notons que dans les deux situations (avec ou sans

PCA), l'architecture $AttPP$ semble en moyenne plus performante sauf pour le paramètre par défaut ($dim(\mathbf{z}) = 15$) que nous avons choisi. De plus, le score total semble toujours plus élevé d'une valeur d'environ +0.05 si les mesures des corrélations et clustering sont effectuées sur l'espace $\mathbf{z}$ au lieu de $\mathbf{z}_{PCA}$. L'inconvénient est que l'espace $\mathbf{z}$ peut présenter plusieurs composantes corrélées à une seule propriété. De plus, rien ne contraint chaque composante à être associée à une des propriétés pour n'importe quel objet. Nous avons observé sur $\mathbf{z}_{PCA}$ qu'une composante avait tendance à encoder une caractéristique physique par exemple le frottement latéral mais que, selon l'objet, une corrélation plus forte pouvait être trouvée sur une autre composante pour cet objet. Dans l'optique d'avoir une composante par caractéristique dans de futures expériences nous avons décidé de conserver la méthode de traitement basée sur la PCA.

Influence de la longueur de la trajectoire T servant à établir le vecteur z : Comme pour le cas supervisé, nous souhaitons maintenant étudier l'influence du nombre d'interactions sur la qualité de la représentation obtenue. Pour cela, nous faisons varier uniquement la longueur de la portion de la trajectoire qui sert à obtenir $\mathbf{z}_{PCA}$. Nous faisons varier la longueur de la trajectoire en prenant 10, 25, 50, 100, 150

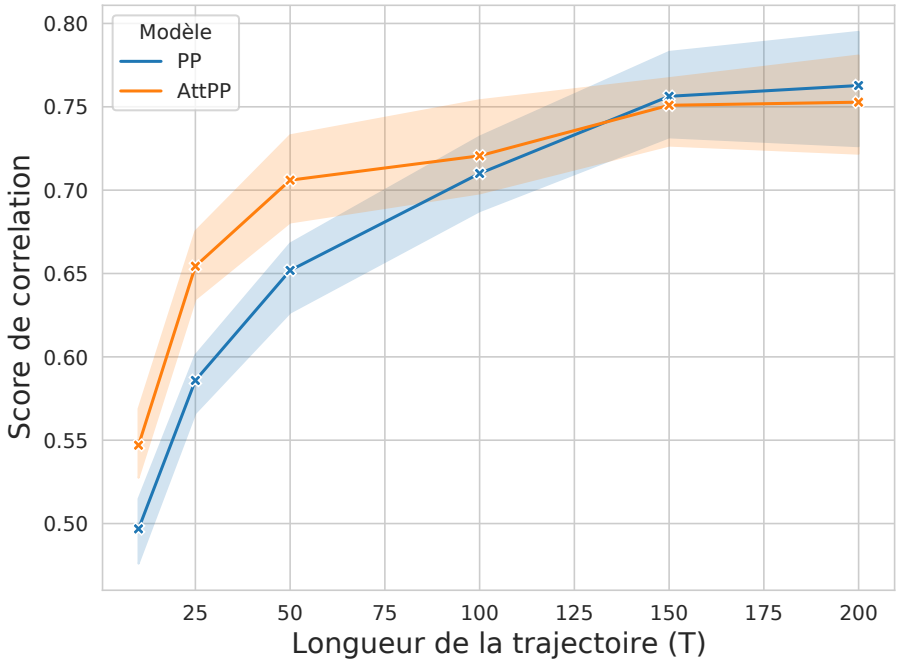


FIGURE V.19 – Score de corrélation du modèle PP (en bleu) et $AttPP$ (en orange) selon la longueur de la trajectoire ayant servi à obtenir $\mathbf{z}$. Moyenne sur 5 entraînements de modèles.

puis 200 observations (la dernière étant notre paramètre par défaut). Nous traçons sur la Figure V.19, le score total de corrélation en fonction de la longueur de la trajectoire de perception, pour les architecture PP et $AttPP$ entraînés avec les observations de π_{MEPOL}. Ici encore, plus le score est élevé, plus les composantes sont corrélées avec

l'ensemble des propriétés étudiées.

Nous constatons pour les deux architectures, que la qualité de la représentation augmente avec une plus grande quantité d'interactions, mais que le score croît peu une fois 100 pas de temps effectués. En dessous, le score diminue fortement jusqu'à atteindre 0.5 pour 10 pas de temps, équivalent à 2 ou 3 interactions physiques. Malgré peu d'interactions, les scores restent plus élevés que la meilleure politique PPO (Tableau V.6), où il est fourni 200 observations. Cela montre l'efficacité de notre politique π_{MEPOL} à acquérir des observations représentatives de la dynamique de l'objet. Notons qu'en divisant par 2 la longueur par défaut de la trajectoire (200) nous restons au delà de 0.7 et que, si nous utilisons l'architecture $AttPP$, nous pouvons encore diviser par 2 le temps d'interaction tout en restant au-dessus de ce seuil. En effet, le modèle $\pi^{AttPP}_{\text{MEPOL}}$ semble moins sensible à une diminution de la taille de la séquence de perception. Ce modèle est donc à privilégier dans le cas où nous souhaitons un agent très efficace en un court laps de temps. L'architecture PP est très légèrement plus performante lorsque la trajectoire de perception est longue. Notons que l'architecture $AttPP$ a une variance entre plusieurs entraînements un peu plus élevée principalement lorsque le nombre d'observations diminue.

Effet du bruits d'observations sur la représentation de l'objet : Pour conclure cette étude, nous simulons une dégradation des moyens de perception de l'agent par l'ajout d'un bruit Gaussien sur la mesure de position du centre de l'objet. Nous avons observé dans le cas d'un entraînement supervisé que les prédictions étaient robustes au bruit. Vérifions maintenant si l'espace de représentation de l'objet obtenu en auto-supervision reste insensible au bruit. Pour rappel, le bruit se propage à une majorité des observations que nous utilisons pour construire l'espace $\mathbf{z}$. Notamment les observations $[e^{(-r^t)}, \cos(\theta^t), \sin(\theta^t), d^t_x, d^t_y, v^t_x, v^t_y]$. La Figure V.20 trace pour les deux modèles, π^{PP}_{MEPOL} et $\pi^{AttPP}_{\text{MEPOL}}$, le score de corrélation total pour des bruits de $[0, 1, 2, 5, 10, 20]$cm. Contrairement au cas supervisé où l'erreur de prédiction restait stable lorsque la politique π_{MEPOL} était utilisée, la Figure V.20 montre que la qualité de la représentation diminue avec l'augmentation du bruit. Ce n'est pas un résultat surprenant étant donné qu'en ajoutant un bruit aléatoire à chaque observation, nous brisons le lien dynamique entre $\mathbf{x}_t$ et $\mathbf{x}_{t+1}$ et mettons à jour les paramètres du modèle à partir de couples $(\mathbf{x}_t, \mathbf{x}_{t+1})$ où l'écart entre les deux observations peut représenter des variations de la masse ou du frottement/résistance au fil des pas de temps alors que l'objet est le même. Il est donc plus difficile d'encoder dans $\mathbf{z}$ des propriétés physiques qui permettent de prédire sans erreurs une autre trajectoire autour du même objet.

Les deux modèles restent cependant relativement performants jusqu'à un bruit de 5cm puis le score est dégradé jusqu'à atteindre 0.6 pour un bruit de 20cm. Notons que ce score est d'environ 0.1 au-dessus de celui de modèles entraînés sans bruits avec π_{PPO}. Ainsi, nous démontrons une nouvelle fois la qualité de l'exploration réalisée par

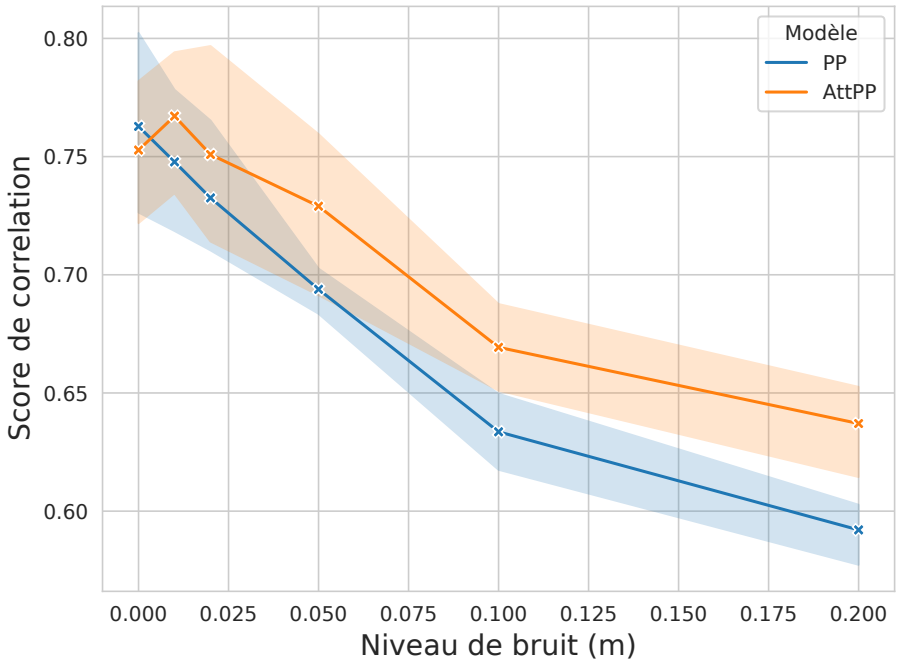


FIGURE V.20 – Score de corrélation du modèle PP (en bleu) et $AttPP$ (en orange) selon le bruit ajouté sur l'estimation de la position du centre de l'objet. Moyenne sur 5 entraînements de modèles.

notre politique π_{MEPOL} qui permet d'acquérir des observations bien plus représentatives sur l'objet. Aussi, nous constatons également que l'architecture $AttPP$ est plus robuste au bruit que l'architecture PP. Sa variance est plus élevée, mais reste constamment au-dessus des performances de π^{PP}_{MEPOL} lorsque le bruit dépasse 2cm. Pour un bruit de 5cm le score moyen obtenu avec $AttPP$ est équivalent à celui de l'architecture PP où un bruit de 2cm est appliqué. Il semble donc que l'architecture $AttPP$ soit à privilégier pour une application en conditions réelles. Sans bruit, l'architecture PP est celle pour laquelle le plus haut score a été atteint.

V–5 Conclusion

Dans ce chapitre, nous avons mis en place une tâche de prédiction avec un entraînement auto-supervisé grâce aux observations acquises par l'agent lors de ses interactions. Cette tâche consiste à prédire la prochaine observation de l'agent à partir de son état actuel, de l'action qu'il va entreprendre et d'un vecteur représentant les caractéristiques dynamiques de l'objet. Si une seule observation est utilisée pour prédire la suivante, alors sans ce vecteur de propriétés, il est à priori impossible de prédire la future observation. Une fois le modèle de perception et de prédiction entraîné sur la tâche, nous utilisons uniquement celui de perception pour créer à partir d'une trajec-

toire, un vecteur latent **z** dont les composantes principales font ressortir les propriétés physiques de l'objet. Pour contraindre le modèle de perception à encoder depuis une série d'observations des informations liées aux propriétés de l'objet, les observations dédiées à la tâche de prédiction doivent être acquises sur le même objet lors d'une nouvelle trajectoire. Dans nos expériences, nous avons appliqué cette règle en séparant une longue trajectoire en 2 séquences séparées par plusieurs pas de temps de simulation. Nous évitons ainsi que le modèle de perception conserve dans **z** une mémoire d'autres éléments liés à la récurrence de la trajectoire.

Pour étudier la contenance de **z**, nous avons mis en place une méthode d'évaluation basée sur le calcul de la corrélation de Spearman (pour les propriétés continues) et d'une classification k-means (pour les propriétés discrètes). Nous avons utilisé une base de données avec connaissance des propriétés de l'objet pour calculer un score de corrélation entre chaque composante principale du vecteur **z** et un certain nombre de propriétés. Nous avons évalué une architecture nommée Perception-Prédiction en la comparant à 4 autres variantes sous différentes conditions : ajout de bruits d'observations, variation de la taille de l'espace **z**, et variation de la longueur de la trajectoire de perception. Nous avons conclu que cette architecture et sa variante utilisant un mécanisme d'attention sont celles qui permettent d'établir les meilleures représentations des caractéristiques d'un objet.

La conclusion principale de ce chapitre réside dans la démonstration que seule la combinaison de la politique π_{MEPOL} développée au Chapitre IV avec l'architecture PP (ou sa variante $AttPP$) permet d'extraire, sans aucune supervision, une représentation dont les composantes correspondent aux propriétés physiques d'un objet. Les résultats ont montré une différence indéniable entre la politique π_{MEPOL}, ses versions de plus faibles entropies, et la politique naïve π_{PPO} dans leurs capacités à fournir des observations qui permettent d'établir une représentation des propriétés physiques d'un objet. En effet, nous avons vu que plusieurs composantes du vecteur latent **z** produit par π^{PP}_{MEPOL} étaient fortement corrélées avec les propriétés de masse, de frottement latéral et de résistance au roulement. De plus, il apparaît clairement sur l'une des composantes 3 groupes caractéristiques de la forme de l'objet.

Pour convertir l'espace de représentation en valeurs compréhensibles par l'homme, notre méthode nécessite une base de test dans laquelle nous avons connaissance des propriétés de l'objet. Ainsi, nous sommes capables d'identifier les composantes de l'espace latent reliées à des propriétés et d'avoir une estimation de la propriété grâce à une relation monotone. Dans le cas où cette base de test n'est pas disponible, l'acquisition d'information dans l'espace latent reste à un état de supposition et seule l'application à une tâche annexe nécessitant de connaître certaines des propriétés permettra de vérifier que l'espace contient en effet des caractéristiques de l'objet. C'est une méthode de validation que nous essayons de mettre en place dans le prochain chapitre.

CHAPITRE VI

Utilisation des connaissances acquises pour la résolution de sous-tâches

Sommaire

VI–1 Introduction

Nous avons développé au Chapitre V une méthode permettant de construire un espace de représentation de l'objet contenant plusieurs informations sur ses propriétés physiques. Ce chapitre met en avant l'intérêt de disposer des connaissances sur un objet pour faciliter la résolution de tâches. L'objectif final étant d'intégrer aux observations de l'agent l'espace latent des caractéristiques de l'objet, tel que créé dans le chapitre précédent en s'appuyant sur la politique comportementale apprise au Chapitre IV, et ainsi améliorer la qualité d'interaction de l'agent avec l'objet pour résoudre plus efficacement une tâche (voir Figure VI.1).

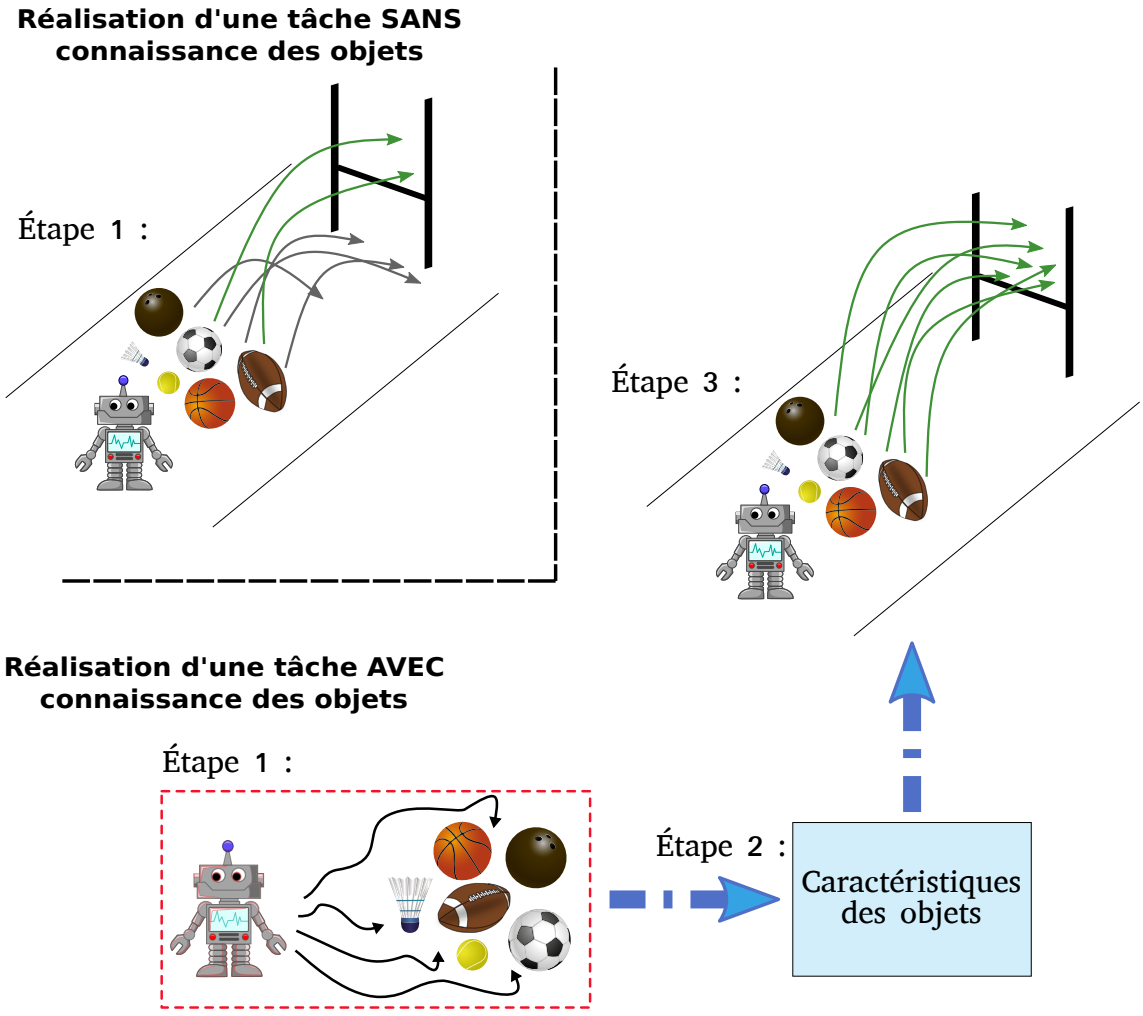


FIGURE VI.1 – Résultat d'un entraînement sur une tâche de lancer d'objets de différentes natures. Sans connaissance de ceux-ci, l'agent ne peut trouver qu'une politique sous-optimale pour résoudre sa tâche. Connaissant les propriétés, il peut adapter son comportement pour la résoudre de manière plus efficace.

Pour atteindre cet objectif, nous devons répondre à des interrogations concernant les capacités de l'agent à tirer profit du savoir acquis. Bien qu'il soit intuitif pour nous,

humains, d'adapter nos actions en fonction de connaissances qui nous auraient été transmises, l'agent lui n'a pas cette capacité de raisonnement par défaut et doit l'apprendre pour gagner en efficacité dans sa tâche. Dans ce chapitre, nous allons valider par des expériences simplifiées que l'agent est en effet capable d'apprendre un raisonnement liant ses actions aux conséquences possibles sur un objet de propriétés parfaitement connues par l'agent. Il ne nous semble pas pertinent d'apprendre à utiliser une représentation imparfaite des caractéristiques de l'objet si l'apprentissage ne fonctionne pas dans un cas de connaissances parfaites. Pour conduire ces expériences préliminaires, nous mettons en place une tâche pour laquelle l'agent est entraîné à résoudre avec une méthode de RL. La tâche doit être conçue pour que sans observation des propriétés, l'apprentissage mène à une politique sous-optimale.

Nous décrivons dans la prochaine section le scénario imaginé, dans un environnement proche de celui utilisé dans les deux chapitres précédents, ainsi que sa configuration pour l'apprentissage avec un algorithme RL. Un robot mobile doit pousser le plus rapidement possible, jusqu'à une zone cible, un objet dont les propriétés varient entre deux épisodes. Sans connaissance de la masse ou des frottements et résistances, l'agent pourrait réussir à apprendre un comportement universel mais pas optimal. Avec la connaissance de ces propriétés, il pourrait s'adapter à l'objet et prévoir des poussées plus ou moins fortes en fonction des paramètres de masse et de frottement pour gagner en efficacité et en précision. Notre intuition est qu'il peut être difficile de créer des liens entre plusieurs propriétés de type continu et discret et donc de ralentir ou même d'empêcher l'apprentissage. C'est pourquoi, dans une troisième section, nous évaluons l'agent au travers d'une tâche de complexité croissante : tout d'abord en variant une seule propriété discrète de l'objet, puis en introduisant plusieurs variations de propriétés discrètes, pour finalement varier un ensemble de propriétés continues et discrètes. Nous avons plusieurs critères d'évaluation à notre disposition. L'amélioration des performances de l'agent peut se traduire par une réduction du temps d'apprentissage, ou par des actions plus judicieuses entraînant des récompenses moyennes plus élevées et un temps de réussite de la tâche plus court. C'est cette seconde option que nous avons choisie pour évaluer notre agent. L'ensemble des expériences de ce chapitre sera réalisé avec la vérité terrain des propriétés des objets. Nous conclurons ce chapitre par une énumération d'expériences qu'il reste à effectuer avant de réaliser un couplage complet avec l'espace de représentation de l'objet issue de notre politique interactive.

VI–2 Tâche sélectionnée

Cette section détaille la tâche imaginée pour évaluer l'influence des connaissances sur l'efficacité de sa résolution. Elle décrit également la méthode de résolution basée

sur un algorithme d'apprentissage par renforcement de l'état de l'art.

VI–2.1 Environnement

La tâche que nous mettons en place est celle d'un ralliement de point avec un objet. Nous souhaitons que l'agent pousse l'objet pour l'amener vers une zone cible tel qu'illustré sur la Figure VI.2. La tâche est accomplie si le centre de masse de l'objet se situe au sein de la zone et s'il est resté 4 secondes consécutives au sein de cette zone. Dans notre simulation, cela équivaut à 20 pas de temps consécutifs [1].

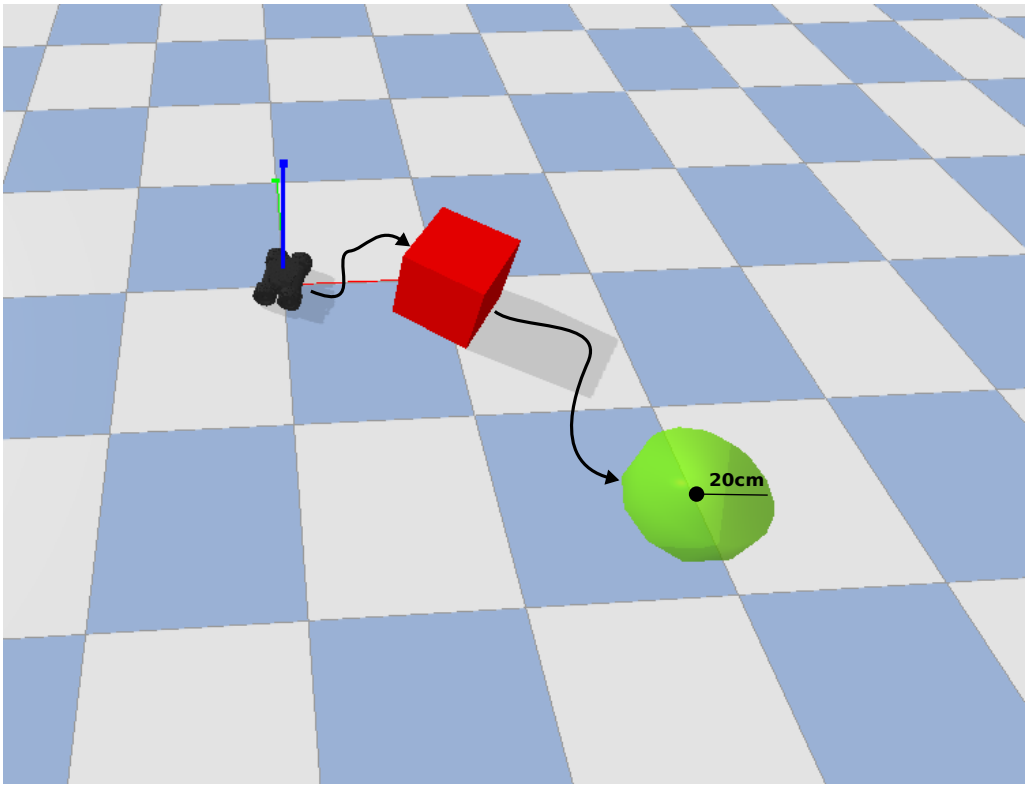


FIGURE VI.2 – Environnement de l'agent mis en place pour la tâche de ralliement de point.

L'agent est initialement positionné au centre du repère monde défini par $R^{monde} = \{x_{monde}, y_{monde}, z_{monde}\}$, avec une rotation autour de z_{monde} aléatoire (Figure VI.3). Un objet de propriétés P ={Forme, Masse, Frottement latéral, Résistance au roulement} échantillonnées uniformément dans leur espace de définition (voir Tableau IV.1), est positionné aléatoirement dans un arc de cercle formant un angle de 120° et dont le rayon est de 2m. Une zone cible représentée par un cylindre de rayon 20cm est positionnée également dans un arc de cercle de 90° dont le rayon est de 4m. Le milieu

1. Pour la tâche, l'action est répétée 42 fois au lieu de 72 (MEPOL). Cela permet à l'agent d'être plus précis dans ses mouvements et manipulations.

des deux arcs de cercle est toujours orienté dans la même direction que celle de la direction d'avance du robot à l'instant initial. De plus, nous faisons en sorte que la cible soit toujours positionnée derrière l'objet. Les cônes d'apparition ont pour but de simplifier la tâche en réduisant les distances maximales à parcourir pour rejoindre l'objet puis la cible. Le temps d'un épisode est par conséquent réduit (car il y a moins d'exploration à effectuer) ce qui permet d'accélérer l'apprentissage.

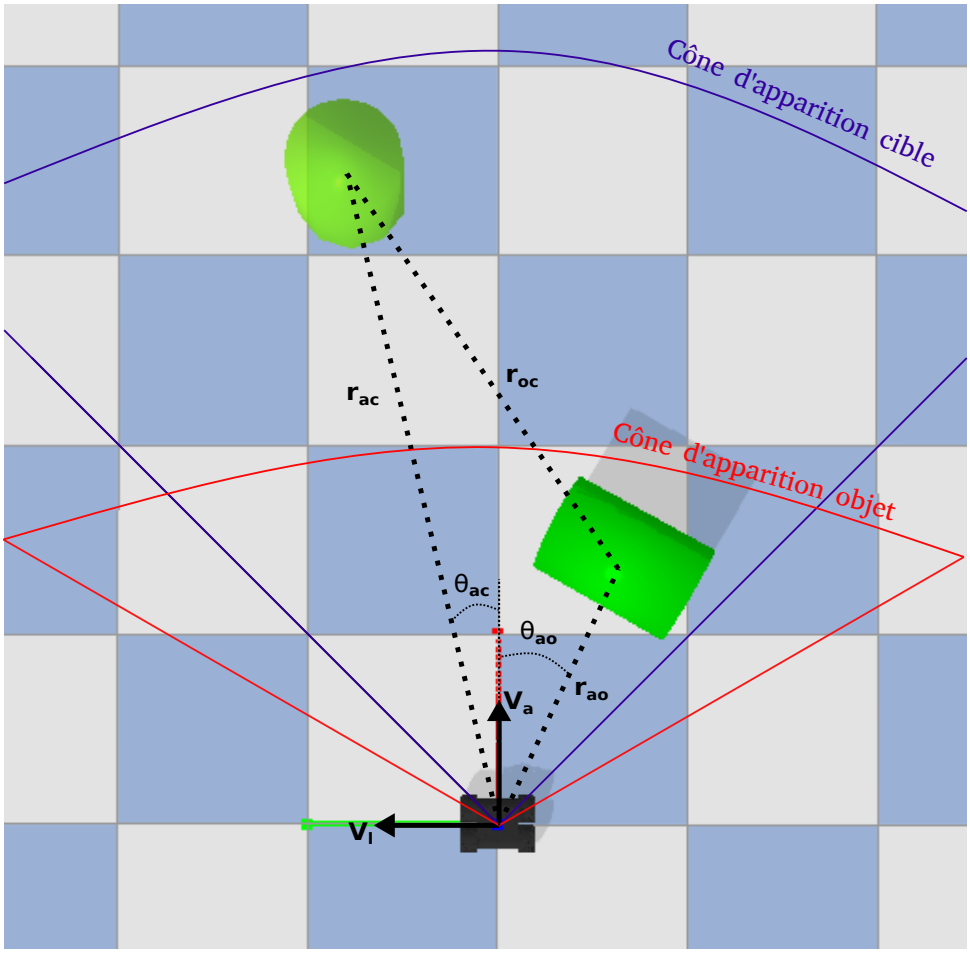


FIGURE VI.3 – État de l'environnement à l'initialisation d'un épisode.

Nous avons opté pour cette configuration initiale car, bien que la tâche soit d'apparence simple, l'espace d'état est lui de grande dimension puisque l'agent à la possibilité de se déplacer sur tout le plan. Cela signifie que l'agent va devoir explorer suffisamment l'environnement pour trouver dans un premier temps l'objet puis pour comprendre ensuite qu'il doit l'amener sur la cible. Si l'agent est récompensé uniquement par la récompense :

$$r = \begin{cases} 1 & \text{si objet à l'arrêt sur cible,} \\ 0 & \text{sinon.} \end{cases} \tag{VI.1}$$

alors la difficulté est décuplée et le cas d'étude s'inscrit dans la problématique classique d'apprentissage avec une *sparse reward*. Dans le Chapitre II plusieurs pistes de résolution pour faire face à ce problème ont été proposées. Dans un premier temps, pour faciliter l'exploration de l'agent, nous avons introduit une initialisation adaptative de l'environnement. Au cours des premiers épisodes, l'objet et la cible sont positionnés à proximité de l'agent, puis la distance maximale d'apparition augmente progressivement. Cette augmentation est réalisée de manière linéaire en fonction du nombre de pas de temps effectués pendant l'apprentissage. Les chances de pousser l'objet précisément sur la cible avec des actions aléatoires sont donc très faibles. Nous avons utilisé la technique du *reward shaping* pour adapter la fonction de récompense. Nous la détaillons dans la prochaine sous-section. L'initialisation adaptative de l'environnement n'a elle pas été conservée.

L'espace d'état de l'agent est de dimension 12 avec des variables normalisées entre -1 et 1. Il est issu des observations suivantes :

- $\{r_{ao}, \theta_{ao}\}$: distance relative et angle relatif entre le centre de masse de l'agent et celui de l'objet.
- $\{r_{ac}, \theta_{ac}\}$: distance relative et angle relatif entre le centre de masse de l'agent et celui de la cible.
- r_{oc} : distance entre l'objet et la cible.
- o_{ao} : indicateur de l'orientation de l'agent par rapport à l'objet. Pour le cube, il vaut 1 si l'agent est orienté perpendiculairement à l'une de ses faces et -1 s'il est orienté dans la direction de ses coins. Pour le cylindre, il vaut 1 si l'agent est dans la direction du roulement et -1 s'il est perpendiculaire à cette direction. Pour la sphère, il vaut toujours 1.
- $\{v_a, v_l\}$: vitesse d'avance et vitesse latérale de l'agent exprimées dans son repère à l'instant courant.
- $\{v_{xm}, v_{ym}\}$: vitesse de l'objet mesurée dans le repère monde et exprimée dans celui de l'agent à l'instant courant.

L'état donné à la politique est alors à chaque pas de temps :

$$\begin{aligned}\mathbf{s}^{\text{permanent}} = [&e^{(-r_{ao})}, \sin(\theta_{ao}), \cos(\theta_{ao}), e^{(-r_{ac})}, \sin(\theta_{ac}), \cos(\theta_{ac}),\\ &e^{(-r_{oc})}, o_{ao}, v_a, v_l, v_{xm}, v_{ym}]^\intercal\end{aligned} \tag{VI.2}$$

Nous modifions la distance r via une fonction exponentielle négative afin de manipuler une variable comprise entre $[0, 1]$ que nous normalisée ensuite entre $[-1, 1]$[2]. Plus

2. Cette technique a également été utilisée dans les chapitres précédents pour normaliser les observations de distance.

la distance est faible, plus cette valeur est proche de 1, plus elle est grande, plus elle tendra vers 0 donc -1 avec la normalisation. De ce fait, nous n'avons pas besoin de connaître les bornes maximales sur ces mesures de distance. Aussi, nous n'utilisons pas directement les angles θ mais plutôt leur cosinus et sinus pour éviter les discontinuités aux extrêmes de l'espace $[-180°, 180°]$.

Nous avons fait le choix d'utiliser cet espace car nous ne souhaitons pas fournir à l'agent une représentation visuelle de son environnement. L'objectif est d'éviter de fournir à la politique la propriété de forme sans passer par l'espace de représentation établi au Chapitre V. Cependant, l'effet de la poussée de l'agent sur l'objet est variable selon l'angle de contact de l'agent et selon le point d'impact. C'est pour cette raison que nous avons ajouté l'information o_{ao}. Elle n'est pas optimale, car le point d'impact sur l'objet n'est pas donné, mais cela fournit l'information à l'agent de son orientation par rapport au sens de poussée le plus approprié. Son espace d'action, lui, reste inchangé par rapport à celui des autres phases de la méthode.

VI–2.2 Entraînement

L'entraînement de l'agent sur la tâche décrite est effectué avec un algorithme RL de l'état de l'art : le *Twin Delayed Deep Deterministic Policy Gradient* aussi appelé TD3 [Fujimoto et al., 2018]. Le TD3 est un choix classique pour la résolution de nombreuses tâches d'apprentissage par renforcement, notamment en robotique [Meng et al., 2021, Tian et al., 2021, Yuan et al., 2022], en raison de ses performances globales, de son adaptabilité à différents problèmes et de sa stabilité. [3] C'est une méthode off-policy où l'architecture est de type acteur-critique.

La tâche étant trop difficile si la récompense est uniquement un signal positif lorsque l'objet atteint la cible, nous avons opté pour une fonction de récompense dite de reward shaping basée sur les travaux de Badnava et al. [2023]. La fonction de récompense est exprimée sous la forme d'une fonction potentielle F :

$$F(\mathbf{s}, \mathbf{s}') = \gamma\phi(\mathbf{s}') - \phi(\mathbf{s}) \tag{VI.3}$$

où γ est le facteur de pondération du MDP et, ϕ une fonction choisie comme suit :

$$\begin{aligned} \phi &= \mathcal{S} \to \mathrm{R} \\ \mathbf{s} &\mapsto \phi(\mathbf{s}) = e^{(-r_{oc}(\mathbf{s}))} \end{aligned} \tag{VI.4}$$

3. Nous avons également entraîné un agent avec les algorithmes SAC et PPO, mais qui ont ensuite été écartés pour leurs moins bonnes performances sur la tâche.

La fonction de récompense utilisée s'écrit donc :

$$\mathcal{R} = (\gamma e^{(-r_{oc}(\mathbf{s}'))} - e^{(-r_{oc}(\mathbf{s}))})\beta \tag{VI.5}$$

où $\beta = 10$ est utilisé pour contrebalancer les valeurs de faibles amplitudes que cette fonction de récompense renvoie. En utilisant une telle fonction, l'agent est constamment récompensé, ce qui permet d'adapter régulièrement les paramètres de la politique. Si l'agent pousse suffisamment l'objet en direction de la cible, alors il reçoit une récompense positive. Inversement, s'il éloigne l'objet de la cible, il est pénalisé. Enfin, s'il ne touche pas l'objet, alors $e^{(-r_{oc}(\mathbf{s}'))} = e^{(-r_{oc}(\mathbf{s}))}$ et la fonction renvoie un signal négatif puisque $\gamma \leq 1$. Le facteur γ agit en quelque sorte comme une pénalité de temps.

Les paramètres utilisés dans TD3 sont donnés dans le Tableau VI.1. Pour favoriser l'exploration de l'agent, nous avons ajouté un bruit gaussien sur les actions renvoyées par la politique. L'amplitude du bruit dépend du nombre de pas de temps d'apprentissage t déjà effectué en suivant une loi linéaire.

$$\mathbf{a}_{bruit} \sim \mathcal{N}(\mathbf{a}, \sigma^2(t)) \tag{VI.6}$$

où

$$\sigma^2(t) = \sigma^2_{initial} + \frac{t}{t_{max}}(\sigma^2_{final} - \sigma^2_{initial}) \tag{VI.7}$$

avec **a** l'action choisie par l'agent, $\sigma^2_{initial}$ la variance appliquée à l'état initial, σ^2_{final} la variance appliquée à l'état final et t_{max} le nombre de pas de temps t d'entraînement où $t \in [0, t_{max}]$.

TABLE VI.1 – Paramètres d'apprentissage utilisés dans TD3.

Paramètre	Valeur
Learning rate	0.01
Taille du replay buffeur	500000
Taille des batchs	256
Facteur de réduction (γ)	0.99
Nombre de steps d'entraînement (t_{max})	$2 * 10^6$
Bruit sur actions à l'état initial ($\sigma^2_{initial}$)	0.8
Bruit sur actions à l'état final (σ^2_{final})	0.01

VI–3 Résolution avec connaissance parfaite de l'objet

L'objectif est ici de comparer les performances d'un agent lorsque les propriétés de l'objet sont disponibles dans son espace d'observation avec un agent qui n'a pas cette capacité de perception. Nous entraînons pour cela 5 politiques lorsque l'espace d'observation est donné sans propriétés, cf. Eq. (VI.2), puis 5 politiques contenant en plus les vérités terrain des propriétés variables de l'objet, cf Eq (VI.8). Nous considérons dans ces expériences que les propriétés sont transmises directement dans l'espace d'observation de l'agent :

$$\begin{aligned}\mathbf{s} = [&e^{(-r_{ao})}, \sin(\theta_{ao}), \cos(\theta_{ao}), e^{(-r_{ac})}, \sin(\theta_{ac}), \cos(\theta_{ac}),\\ &e^{(-r_{oc})}, o_{ao}, v_a, v_l, v_{xm}, v_{ym}, p_1, p_2, p_3, \ldots]^\intercal\end{aligned} \tag{VI.8}$$

Selon les propriétés de l'objet, la complexité de la tâche peut varier. Un objet très glissant est difficile à positionner sur la cible et la trajectoire à chaque contact n'est pas la même selon la forme. Pour ces raisons, dans un premier temps nous avons divisé le problème en sous-problèmes où toutes les propriétés ne sont pas variables. Dans une première section, nous traitons le problème avec un ensemble discret d'objets possibles. Puis dans une seconde section, nous évaluons le cas où ces propriétés sont uniformément échantillonnées dans un espace continu. Ces résultats sont finalement comparés à une politique apprise sur un objet de propriétés fixes.

VI–3.1 Évaluation sur un ensemble de cas discret

VI–3.1.a Forme fixe et 2 comportements distincts

Dans cette première expérience, l'objet est de forme fixe et sa dynamique varie selon 2 modes. En jouant sur les paramètres de frottement latéral, résistance au roulement et de masse, nous faisons en sorte qu'un objet soit très glissant et un autre bien plus lourd et maniable (objet type 1 et objet type 2 du Tableau VI.2). Pour cela, les propriétés sont les suivantes :

Pour ce test, la propriété donnée à l'agent est un booléen de valeur -1 si l'objet est de type 1, et de valeur 1 si l'objet est de type 2. Nous réalisons l'apprentissage pour chaque forme possible, c'est-à-dire le cylindre, le cube et la sphère. La Figure VI.4 compare les scores moyens des 5 politiques pour chaque forme, avec ou sans la présence de l'information du type d'objet. Ces scores représentent la récompense moyenne par épisode et la longueur moyenne de l'épisode.

Les résultats donnés par la Figure VI.4 montrent que l'agent est pour l'instant capable d'exploiter la distinction fournie entre les deux objets pour gagner en efficacité sur la tâche, et ce sur toutes les formes possibles de l'objet. Un gain d'environ 0.5 sur

TABLE VI.2 – Espaces des propriétés pouvant être prises par les objets.

		Objet type 1	Objet type 2
Cylindre	Masse	0.1	6
	Frottement latéral	0.1	0.33
	Résistance au roulement (10^{-2})	0.4	5
Cube	Masse	0.3	5.2
	Frottement latéral	0.03	0.3
	Résistance au roulement (10^{-2})	0.1	0.1
Sphère	Masse	0.8	5.3
	Frottement latéral	1	1
	Résistance au roulement (10^{-2})	0.8	4

la récompense moyenne est enregistré. L'amélioration de l'efficacité est prouvée en regardant le graphique de longueur moyenne d'épisode, où la tâche est résolue entre 5 et 10 pas de temps plus rapidement. Nous pouvons aussi noter une variance plus faible sur la longueur d'épisode, ce qui semble démontrer une meilleure sélection d'actions en fonction du type d'objet. Concernant la vitesse d'apprentissage de ces politiques, nous n'avons pas observé de différences notables et n'affichons donc pas ces résultats.

A priori, sans connaissance du type de l'objet, l'agent ne sait pas quelle force utiliser. S'il tape l'objet de type 1 avec trop de puissance, il va l'envoyer au loin et potentiellement l'éloigner de la cible. Les résultats devraient donc être moins bons sans observation de la propriété et ne pas forcément permettre à l'agent de réussir la tâche à coup sûr. Cependant, nous observons que ces résultats restent bons et peu éloignés de ceux obtenus lorsque l'objet est connu. Cela signifie qu'il existe une politique universelle pour laquelle la tâche est réalisable en appliquant la même force pour les deux objets. Une solution est par exemple de frapper fort quand l'objet est loin puis avec une plus faible puissance lorsque l'objet se rapproche de la cible. De cette manière, l'agent mettra plus de temps à amener l'objet lourd sur l'objectif impliquant une longueur d'épisode plus longue. Après une étude de la distribution des actions au moment du contact avec l'objet, aucune conclusion évidente ne peut être tirée. L'agent pourrait avoir tendance, sur les premiers pas de temps, à pousser avec la plus grande force possible l'objet, puis à réduire sa vitesse légèrement lorsqu'il se rapproche de la cible. Ce n'est pas le cas à chaque fois, nous constatons de manière visuelle que l'agent pousse l'objet le plus fort possible en direction de la cible puis cherche à l'arrêter sur la cible en se déplaçant derrière pour arrêter l'objet. Il est également probable que l'agent arrive

de lui-même à différencier les deux objets uniquement à partir de l'espace d'état. Nous fournissons à l'agent les informations $\{v_{xm}, v_{ym}\}$ qui représentent la vitesse de l'objet, mesurée dans le repère monde. Comme nous avons un objet très glissant et léger, la vitesse de celui-ci est forcément d'une amplitude différente de celle d'un objet lourd et peu glissant permettant à l'agent d'identifier le type d'objet.

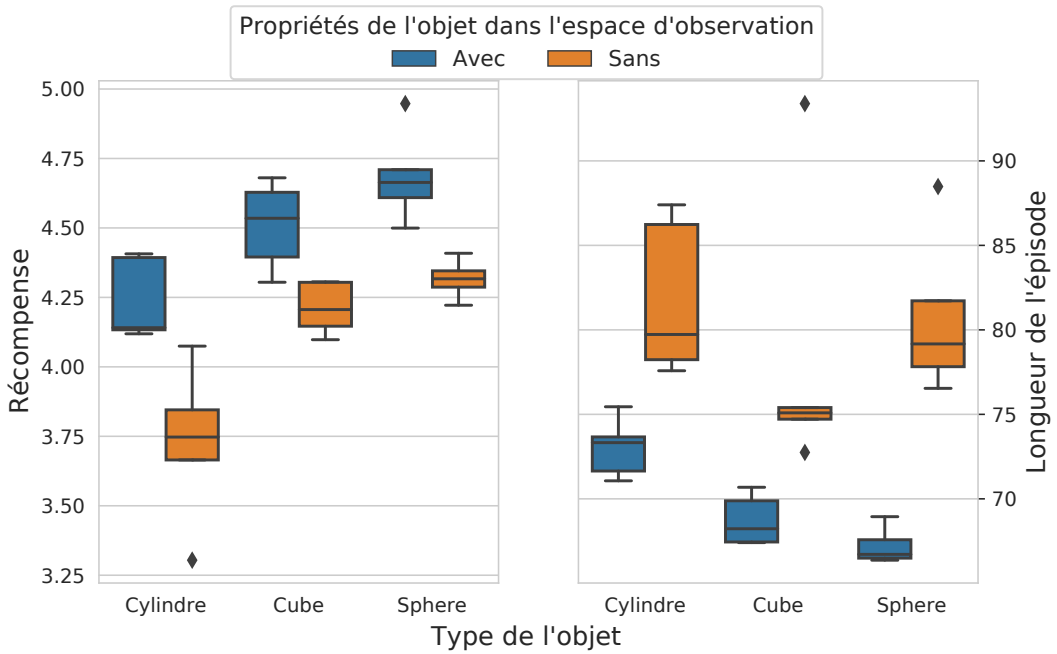


FIGURE VI.4 – Performance de l'agent sur un objet de forme fixe pouvant prendre deux comportements distincts. La performance est mesurée selon la récompense moyenne par épisode (à gauche) et la longueur de l'épisode (à droite). En bleu, l'information sur le type de l'objet est donnée dans l'espace d'observation. En orange, cette information est absente.

La mise en évidence d'un gain de performance en présence d'informations supplémentaires sur l'objet n'est pas si évidente. La tâche est probablement trop simple pour obtenir de grands écarts. Cette complexité peut être augmentée par une variation de nombreux paramètres comme les propriétés, la taille de la cible, le temps d'immobilité dans la cible, l'ajout de la variation de la forme, et plus encore. La variation de l'un de ces paramètres peut rapidement rendre impossible ou trop simple la tâche. Un objet trop lourd, trop glissant ou encore une cible trop petite peut empêcher l'agent de terminer l'épisode en atteignant par chance la cible et ainsi recevoir la récompense associée à cette action idéale. Inversement une tâche avec un objet trop maniable et peu glissant va être résolue trop facilement par l'agent sans même qu'il ait besoin de différencier le type d'objet. La conception de cette tâche est donc une étape particulièrement difficile et met en avant des problèmes classiques régulièrement rencontrés en RL (sparse reward, manque d'exploration, triche, ...). Nous restons satisfaits des résultats de la Figure VI.4 qui montrent tout de même une différence de performance et prouvent que connaître l'objet avant de le manipuler est un réel bénéfice.

VI–3.1.b Ajout de la variation de forme

Nous faisons maintenant varier, en plus du type, la forme de l'objet (les valeurs des propriétés continues restent identiques à celles du Tableau VI.2). L'agent a donc la possibilité de faire face à 6 objets de comportement bien distinct. Dans le cas où les propriétés sont fournies à l'agent, le vecteur d'état classique, voir Eq. (VI.2), est complété par les deux propriétés variables :

$$\mathbf{s} = [\mathbf{s}^{\text{permanent}}, p_{\text{forme}}, p_{\text{type}}]^\intercal$$

avec

$$\begin{cases} p_{\text{forme}} \in \{-1, 0, 1\} & \text{pour } \{\text{cylindre, cube, sphere}\} \\ p_{\text{type}} \in \{-1, 1\} & \text{pour } \{\text{type 1, type 2}\}. \end{cases}$$

Les résultats de la Figure VI.5 comparent la récompense moyenne par épisode et la longueur moyenne de l'épisode, pour 5 politiques entraînées avec ou sans ajout des propriétés de forme et type. En sélectionnant dans les deux cas la meilleure politique parmi les 5, celle qui utilise les informations complémentaires est légèrement plus performante. Les écarts observés dans la section précédente se retrouvent ici. Nous pouvons néanmoins constater que les performances des politiques utilisant les propriétés ont une variance élevée. Cela est dû au fait que l'entraînement de la politique avec propriétés est devenu moins stable et peut dans certains cas ne pas ou mal converger.

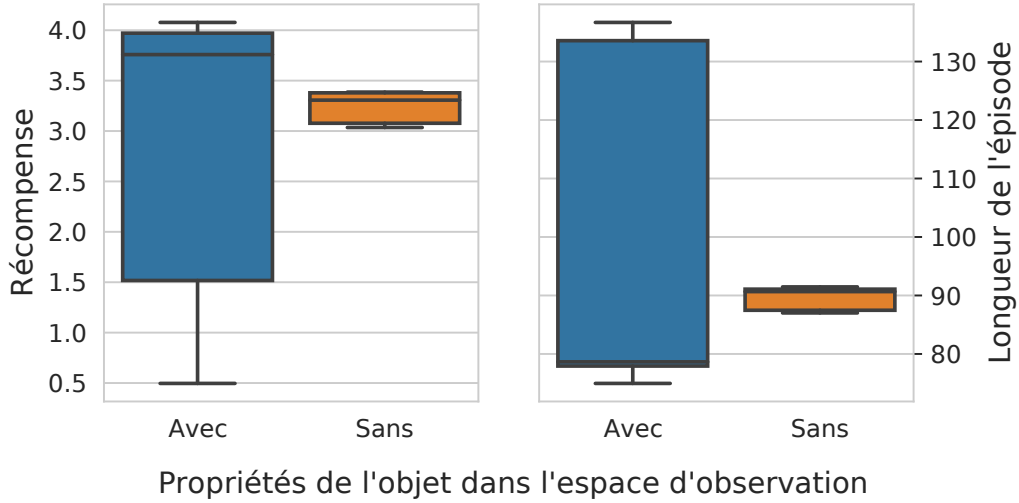


FIGURE VI.5 – Performance de l'agent sur un objet de forme variable pouvant prendre deux comportements distincts.

Cette variance élevée entre plusieurs apprentissages peut potentiellement être expliquée par le fait que l'espace d'état soit de bien plus grande dimension à cause de

cette composante supplémentaire. Par l'ajout de la propriété de forme, le nombre de combinaisons possibles dans l'espace d'état est multiplié par 3. L'exploration est alors bien plus complexe et les chances de tomber dans un minimum local élevées car certains états ne seront probablement jamais observés. Bien qu'il soit intuitif pour un humain que la connaissance d'une information supplémentaire puisse faciliter la manipulation, d'un point de vue algorithmique, cela augmente le champ des possibilités et peut conduire à l'effet inverse.

VI–3.2 Évaluation avec échantillonnage uniforme des propriétés d'objet

Après validation de l'intérêt de fournir à l'agent la capacité de distinguer des objets sur des cas simples, nous complexifions un peu plus l'apprentissage de la tâche en réalisant ces mêmes expériences dans le cas où les propriétés choisies ne sont plus binaires mais échantillonnées aléatoirement et de manière uniforme dans un espace prédéfini. Les bornes minimales des propriétés continues (masse, frottement résistance) pour chaque objet sont celles de l'objet type 1 du Tableau VI.2. Les bornes maximales sont celles de l'objet type 2. Les conditions d'entraînement restent les mêmes et les performances sont mesurées sur 5 apprentissages parallèles de la politique.

VI–3.2.a Forme fixe et variation aléatoire des propriétés continues

Avant de se placer dans les conditions de difficulté maximale avec la variation de l'ensemble des propriétés, nous réalisons un premier test sans faire varier la forme de l'objet. Plus précisément, nous cherchons à comparer les résultats dans le cas d'un cube ayant des propriétés continues aléatoires (masse, frottement). Dans ce cas, l'espace d'observation donné à l'agent est le suivant :

$$\mathbf{s} = [\mathbf{s}^{\text{permanent}}, p_{\text{masse}}, p_{\text{frottement latéral}}]^\intercal$$

où p_{masse} et $p_{\text{frottement latéral}}$ sont tirés aléatoirement puis normalisés entre $[-1, 1]$.

Les résultats de la Figure VI.6 comparent les métriques d'évaluation pour 5 politiques entraînées avec ou sans ajout des propriétés continues. La récompense moyenne semble légèrement plus élevée si les propriétés sont observées. La récompense atteinte est plus grande que celle des précédentes expériences. La distribution continue des propriétés échantillonnées fait que, dans une majorité des cas, la manipulation des cubes est très simple car il y a peu de glissement. L'apprentissage de la politique pour les cas complexes est donc aidée par l'exploration effectuée lorsque le cube est très maniable. Même sans les observations complémentaires, le processus d'apprentissage est facilité par ces exemples simples.

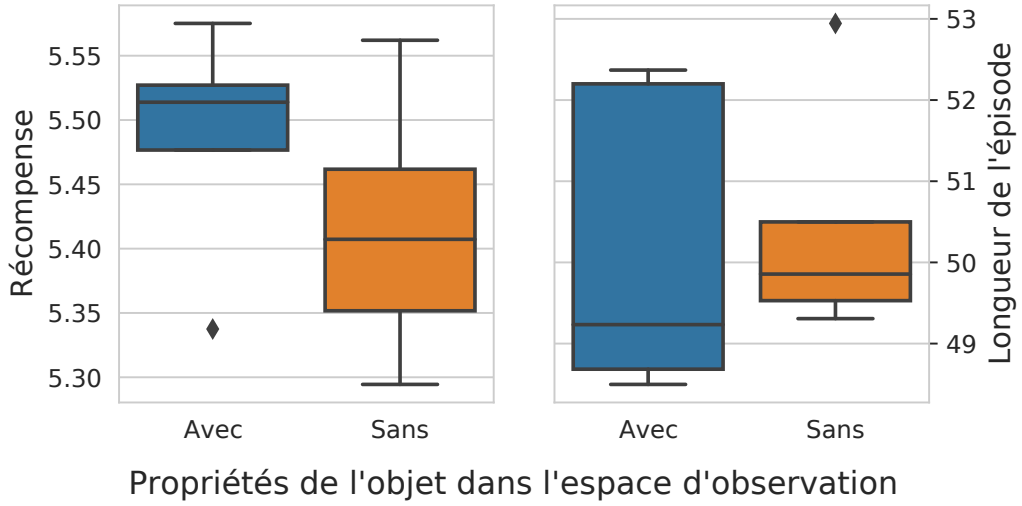


FIGURE VI.6 – Comparaison de la performance de politiques entraînées sur un cube de propriétés continues choisies aléatoirement, cette information étant donnée ou non à l'agent.

Nous pouvons supposer qu'avec l'information du glissement et du poids de l'objet, l'agent a la possibilité d'augmenter sa force lorsque l'élément manipulé est moins glissant et par conséquent gagner du temps et des récompenses. Sans ces informations, l'agent a plutôt intérêt à pousser l'objet plus faiblement pour être sûr de ne pas dépasser la cible dans le cas où la masse et le frottement sont faibles. Ce sont des hypothèses qui restent difficiles à prouver de par le peu de différences observées.

VI–3.2.b Variation aléatoire de l'ensemble des propriétés

Nous réalisons maintenant la même expérimentation en faisant varier la forme, en plus des autres propriétés. Les informations de forme et de résistance au roulement pour les cylindres et les sphères sont donc ajoutées à l'espace d'observation :

$$\mathbf{s} = [\mathbf{s}^{\text{permanent}}, p_{\text{masse}}, p_{\text{frottement latéral}}, p_{\text{résistance au roulement}}, p_{\text{forme}}]^\intercal$$

où $\{p_{\text{masse}}, p_{\text{frottement latéral}}, p_{\text{résistance au roulement}}\}$ sont tirés aléatoirement puis normalisés entre $[-1, 1]$. p_{forme} prend les valeurs $-1, 0, 1$ selon la forme de l'objet.

Les résultats de la Figure VI.7 comparent les métriques d'évaluations pour 5 politiques entraînées avec ou sans ajout des propriétés. Cette fois-ci, l'ajout des propriétés a un effet négatif sur les performances puisque la récompense moyenne sans ajout de cet espace est bien plus élevée. Une seule politique atteint des performances correctes (un score de récompense moyenne par épisode d'environ 4) se rapprochant de celles que nous obtenons pour des politiques sans connaissance des propriétés.

Cet écart est probablement lié à la problématique de l'exploration de l'espace d'état

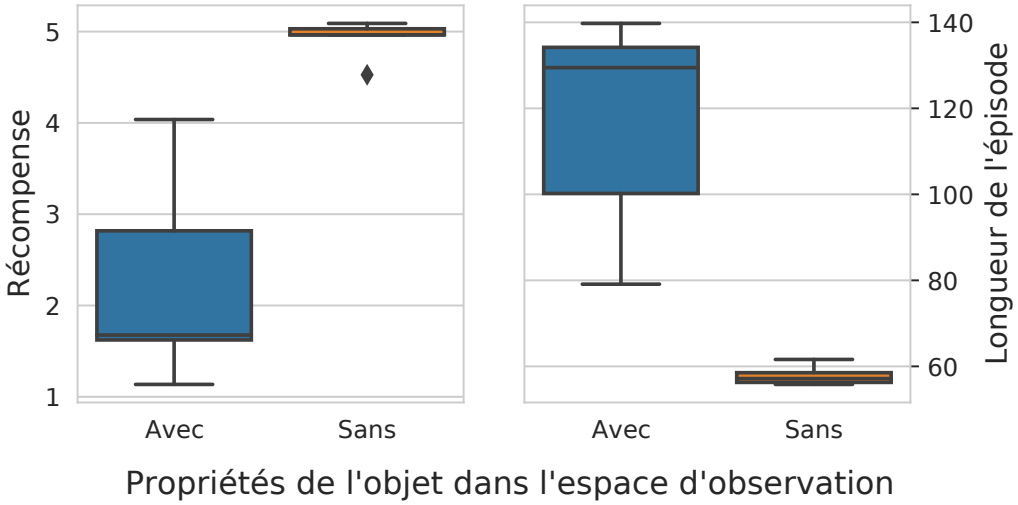


FIGURE VI.7 – Comparaison de la performance de politiques entraînées sur des objets dont les propriétés sont tirées aléatoirement et dont cette information est donnée ou non à l'agent.

où les possibilités ont été décuplées par l'ajout des 4 propriétés variables. Une politique valide pour un objet d'une forme donnée n'est pas forcément valide pour un autre objet de propriétés continues identiques, mais de forme différente. L'agent doit donc avoir observé tous les cas possibles pour être efficace. Dans la situation actuelle, il reste difficile d'expliquer de telles performances dans le cas où les propriétés ne sont pas fournies. Notre intuition se tourne vers la méthode de transmission de l'information de forme qui est dans ce cas une valeur discrète alors que l'ensemble des autres observations de l'agent sont des variables continues.

Dès lors que nous avons ajouté des propriétés à l'espace d'observation de l'agent, nous avons observé des instabilités ou des problèmes de convergence sur la tâche imaginée. Par exemple dans le cas d'objets discrets, nous avons observé les premières instabilités lorsque nous avons ajouté la variation de la forme. Dans le cas continu, la performance de l'agent a brutalement baissé lorsque nous avons fait varier 4 propriétés au lieu de 2. Le problème semble pointer vers une question de conditionnement des informations pour l'utilisation dans un apprentissage d'une politique via RL. L'agent semble en difficulté pour explorer son espace d'observation et comprendre rapidement le lien entre les comportements de l'objet et les propriétés.

VI–3.3 Entraînement de politiques propres à chaque objet

Nous souhaitons maintenant comparer les performances des politiques qui précèdent avec celles de politiques entraînées sur un objet unique de propriétés fixes. Dans les expériences précédentes, nous étions dans le cadre d'un POMDP si les informations des propriétés n'étaient pas données en plus de $\mathbf{s}^{\text{permanent}}$, ici la politique

est apprise sur un MDP et doit par conséquent faciliter la convergence vers une politique optimale. La question est de savoir si une politique entraînée sur un objet fixe est équivalente à celle où l'agent dispose des informations complémentaires (qui est également un MDP).

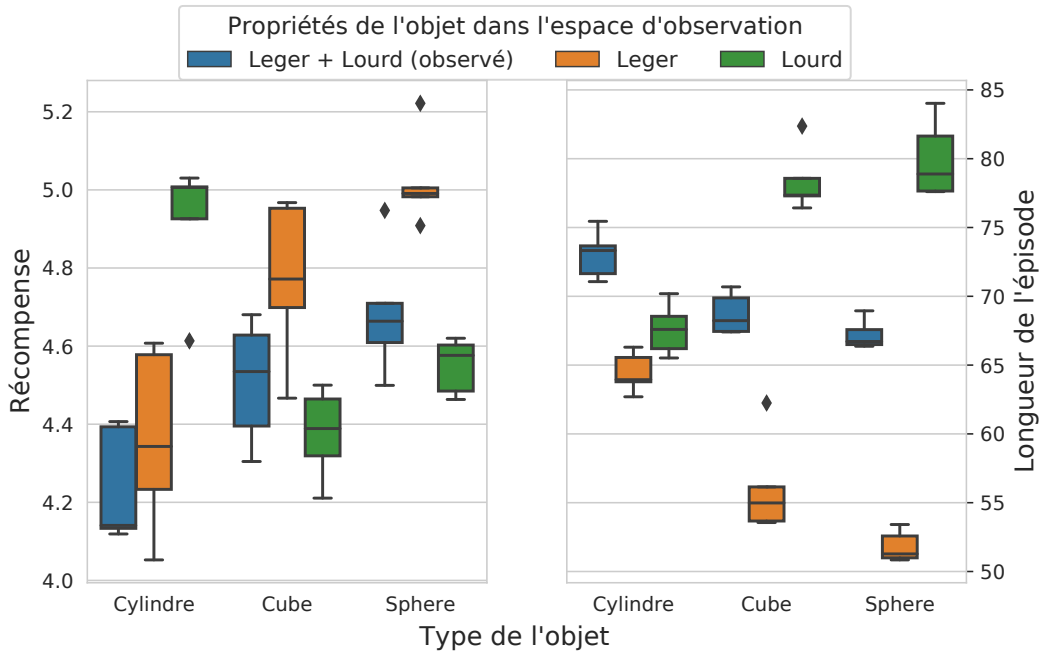


FIGURE VI.8 – Résultats de l'apprentissage où chaque modèle est entraîné sur un objet unique de forme et propriétés continues fixes. "Léger + Lourd" donnent les performances d'un modèle unique entraîné sur un objet de forme fixe mais de propriétés variables.

Pour cela, nous considérons le cas de propriétés discrètes avec 2 types de comportements possibles par forme (les propriétés utilisées restent identiques à celles du Tableau VI.2). Nous entraînons 5 modèles par cas possibles (6 cas) puis comparons ces résultats à ceux d'un modèle unique obtenu sur chaque forme. La Figure VI.8 affiche en orange (objet léger de type 1) et vert (objet lourd de type 2) les performances des modèles entraînés sur un objet fixe. En bleu, sont affichés les scores d'un modèle unique selon une variation du type déjà exposés en Figure VI.4. Dans le cas du cylindre, il semble plus intéressant d'avoir un modèle unique par objet. Pour le cube et la sphère, les performances sont équivalentes ce qui doit être vrai en théorie si l'espace d'état est suffisamment exploré.

Si maintenant nous comparons les modèles uniques aux résultats de la Figure VI.5, où un seul modèle est utilisé pour les 6 cas, il semble bien plus efficace de conserver les modèles uniques. Nous pourrions donc imaginer un algorithme global de contrôle où l'espace latent des propriétés est utilisé comme moyen de sélection de la politique appropriée. Cette solution reste peu pratique et peu réaliste étant donné qu'il faudrait entraîner une infinité de politiques pour tous les objets possibles, puis être capable de

relier l'espace latent à la politique correspondante. Le nombre de politiques pourrait cependant être réduit en considérant dans un même modèle des objets de propriétés et comportement similaires.

VI–4 Conclusion

Dans ce chapitre, nous avons évalué les capacités d'un agent à tenir compte des informations de propriétés acquises sur l'objet. Ainsi, au travers de cas simples où l'objet peut prendre 2 comportements distincts puis en augmentant le nombre de variations possibles, nous avons pu nous rendre compte que le transfert de l'information n'est pas trivial et que la résolution par des méthodes d'apprentissage par renforcement pose quelques problèmes. Il semble intuitif que des informations supplémentaires sur l'objet permettent à l'agent de gagner en efficacité. Cependant, cette idée est vérifiée uniquement dans des cas où il y a peu de variations de propriétés. À partir d'une certaine complexité, l'ajout des propriétés entraîne une baisse des performances de l'agent.

Les expériences réalisées mettent en avant de nombreux sous-problèmes liés à l'utilisation des connaissances d'un objet.

La tâche étant réalisée en simulation, le moteur physique du simulateur peut rendre la dynamique de l'objet irréaliste ou trop similaire après des variations de propriétés. Il semble facile, dans certaines conditions, pour l'agent de trouver une solution au problème. Seulement, si la difficulté est trop augmentée alors il est possible que l'agent ne puisse jamais converger, car son exploration n'est pas suffisante pour découvrir par chance des états de réussite.

L'emploi du reward shaping pour faciliter la convergence de la politique peut également être un problème et causer des erreurs d'évaluation. L'utilisation du reward shapping biaise probablement les trajectoires que l'agent génère empêcherait d'accroître l'efficacité par l'ajout de connaissances en rendant trop simple la découverte d'une solution. L'argument qui nous permet d'affirmer cela est que l'agent semble capable de trouver sans ces connaissances une politique fonctionnelle pour tous les cas bien qu'il nous semble intuitif qu'il existe une politique optimale où l'agent doit, par exemple, adapter son dernier contact pour être précis sur la cible à atteindre. Il rapprocherait le plus rapidement possible l'objet vers la cible avant d'adapter son dernier coup. Cette politique optimale semble trop difficile à trouver puisqu'elle nécessite une exploration complète de l'espace d'état pour être découverte. La difficulté de cette exploration augmente notamment avec l'ajout des propriétés ce qui rend l'agent contre-productif.

Enfin, il est possible que l'intégration des propriétés dans l'espace d'observation de l'objet ne soit pas la méthode la plus appropriée. Nous ajoutons au vecteur d'observation de l'agent des valeurs qui restent constantes le temps de la trajectoire et qui

peuvent être discrètes comme continues. Il existe peut-être d'autres moyens de fournir ces connaissances à l'agent ce qui pourrait également réduire le temps d'apprentissage supplémentaire induit par un ajout direct dans l'espace d'observation de l'agent.

Nous avons, lors d'autres expériences non présentées dans ce chapitre, essayé un conditionnement de type Film [Perez et al., 2018] pour introduire les propriétés à la politique de l'agent. Nous avons également testé de coupler la méthode d'interaction et de découverte de propriétés (chapitre IV et V) à la tâche pour fournir à l'agent l'espace latent des propriétés $\mathbf{z}$ au lieu des vérités terrain. Les expériences n'ont pas produit des résultats convaincants et demandent des investigations plus approfondies avant que nous puissions formuler des affirmations. La question de l'intégration des propriétés reste donc ouverte, la fusion de ces informations dans l'espace d'état ne semblant pas être la solution appropriée.

CHAPITRE VII

Conclusion et perspectives

Sommaire

VII–1 Résumé des contributions

Un environnement réaliste comprend de nombreux objets complexes qu'il est difficile de comprendre complètement sans interaction physique. Cette thèse a pour objectif principal la proposition et l'analyse de stratégies d'apprentissage des interactions entre un robot autonome et des objets inconnus, sans connaissance préalable, en utilisant ses capteurs pour extraire des propriétés physiques des objets. La collecte de ces propriétés vise à aider l'agent dans la réalisation de diverses tâches où une interaction physique avec l'objet est nécessaire.

Pour atteindre cet objectif, nous faisons l'hypothèse qu'il existe des stratégies d'interaction permettant d'extraire ces propriétés plus efficacement. Ainsi, nous pensons qu'en apprenant à l'agent à diversifier les mouvements de l'objet par ses actions, il peut établir une représentation plus précise des caractéristiques de l'objet. Cette représentation doit mettre en évidence les propriétés de l'objet qui, lorsqu'elles sont intégrées aux observations de l'agent sur un apprentissage d'une tâche, doivent l'aider à choisir les actions optimales pour la réaliser.

Dans ce manuscrit, nous avons dans un premier temps décrit la méthode d'un point de vue général. Elle se décline en 2 phases. La première est l'utilisation de l'apprentissage par renforcement non-supervisé comme moyen de concevoir une politique qui favorise l'acquisition d'observations contenant des informations corrélées aux propriétés de l'objet. La seconde phase correspond à la caractérisation de ces propriétés au sein d'un espace établi grâce une tâche de prédiction auto-supervisée. Nous validons cette méthode dans une simulation où un robot mobile interagit par la poussée avec des objets de propriétés aléatoires, incluant la forme, la masse, le frottement latéral et la résistance au roulement. Pour conclure cette étude, nous avons cherché à mettre en application les connaissances acquises dans une tache de positionnement d'objet vers une zone cible, tout en évaluant l'incidence sur les performances de l'agent lorsque les propriétés sont fournies ou non. Le processus se déroule donc en trois étapes (interaction, découverte, utilisation) et a pour but d'initier un cycle d'action-perception orienté objet pour un robot autonome au sein de divers environnements.

VII–1.1 Apprentissage non-supervisé d'une politique d'interaction

Dans le chapitre IV, nous avons exposé une solution non-supervisée d'exploration active d'un objet par l'interaction. Elle s'appuie sur l'apprentissage par renforcement en combinaison avec un signal de motivation intrinsèque permettant à l'agent de se récompenser seul et sans ajout d'un biais humain. Ce signal est basé sur un calcul d'entropie d'une distribution traduisant la diversité des mouvements d'un objet. Dans l'expérience réalisée, le signal est une mesure de la vitesse de l'objet exprimée dans

un repère fixe avec comme origine son centre de masse. La méthode d'apprentissage ainsi proposée ne nécessite pas de connaissance a priori sur l'objet, ni de connaissances parfaites de l'environnement dans lequel l'agent est amené à évoluer. De plus, nous avons fait en sorte que l'agent puisse naviguer et prendre ses décisions uniquement à partir de mesures issues de ses propres capteurs.

Visuellement nous constatons que l'agent réalise des actions cohérentes par rapport au signal de motivation puisqu'il vient frapper l'objet selon différentes directions et différentes vitesses tout en variant la zone du point d'impact. À ce stade, il n'y a aucune preuve que la stratégie développée est plus intéressante qu'une autre pour extraire des propriétés d'objets. C'est pour cette raison que nous avons évalué plusieurs stratégies d'interaction à l'aide de modèles de prédictions de propriétés entraînés de façon supervisée. La politique basée sur la motivation intrinsèque est comparée à une politique apprise par RL classique où l'agent est récompensé pour toucher un maximum de fois l'objet. Cette dernière stratégie est choisie, car elle ne nécessite de connaissances a priori ni sur l'objet ni sur l'environnement.

Nous avons validé l'hypothèse que l'utilisation de l'entropie comme source de récompense permet à l'agent d'acquérir un ensemble d'observations à partir desquelles des propriétés physiques d'un objet peuvent être inférées par un apprentissage supervisé. En effet, nous avons observé que la précision des prédictions sur la propriété de forme, de masse, de frottement latéral et de résistance au roulement, augmente avec le niveau d'entropie amenée par la politique de l'agent. Ces prédictions sont également de meilleure qualité lorsque le modèle est entraîné avec les observations de la politique intrinsèquement motivée plutôt qu'avec les observations de la politique naïve.

Nous montrons par des variations de paramètres comme le bruit, la longueur de la trajectoire d'interaction ou encore le nombre d'exemples d'entraînement que les observations issues de notre politique permettent une inférence plus robuste contrairement à la politique naïve. D'une part, l'agent a besoin d'un nombre plus faible d'interactions pour avoir une prédiction plus précise des propriétés. D'autre part, le modèle de prédiction a besoin de moins d'exemples pour obtenir des résultats similaires à la politique de comparaison. Enfin, seule notre politique conserve une erreur de prédiction stable malgré l'ajout de bruit de plus en plus important dans les mesures effectuées par l'agent. Ceci prouve que les observations acquises sont plus riches en informations corrélées aux propriétés de l'objet. Les résultats obtenus sont donc encourageants pour établir, dans la phase 2, une représentation de qualité des caractéristiques dynamiques.

VII–1.2 Découverte des propriétés d'objets par auto-supervision

Nous avons proposé au Chapitre V une méthode pour encoder les propriétés physiques d'un objet dans un espace de représentation provenant d'un apprentissage auto-supervisé. Pour cela, nous avons mis en place, en adaptant une méthode de l'état de l'art, une tâche de prédiction des états futurs de l'objet à partir des observations acquises par l'agent sur une trajectoire face à un objet. La trajectoire complète est décomposée en deux nouvelles séquences où l'une est utilisée pour établir l'espace de représentation des caractéristiques et où l'autre est utilisée pour la tâche de prédiction avec un modèle qui prend comme entrée l'état de l'objet et l'espace issu de la première séquence. L'hypothèse que nous testons dans ce chapitre est que l'application de cet apprentissage à des trajectoires produites par la politique maximisant l'entropie conduit à une représentation latente permettant de faire émerger les propriétés physiques de l'objet.

La méthode est appliquée dans le même environnement de simulation que celui qui a été utilisé pour entraîner la politique. Nous évaluons le contenu de l'espace latent par des mesures de corrélations entre les composantes de l'espace et les propriétés véritables des objets. Dans le cas de propriétés continues, nous mesurons la corrélation de Spearman. Dans le cas de propriétés discrètes, nous détectons des clusters à l'aide de l'algorithme k-means et établissons un score de classification. que l'application de cet apprentissage à des trajectoires produites par la politique maximisant l'entropie conduit à une Nous proposons également plusieurs variations de l'architecture de Perception-Prédiction et comparons le score de chacun des modèles entre eux pour différentes politiques. La politique naïve exploitée dans le Chapitre IV est de nouveau utilisée ici.

Les résultats montrent que les corrélations entre les propriétés physiques d'objets et les composantes des espaces sont fortes uniquement lorsque notre stratégie d'interaction est utilisée comme moyen d'acquisition des observations. Le niveau de corrélation augmente avec le niveau d'entropie de la politique. Pour une entropie maximale, nous distinguons des corrélations évidentes entre les propriétés continues et discrètes et les composantes principales de l'espace de représentation latente. Nous discernons aisément, sur la première composante principale, 3 clusters correspondants aux 3 formes possibles des objets simulés. Les 3 autres composantes suivantes sont elles corrélées avec les propriétés de masse, de frottement latéral et de résistance au roulement. Nous validons ainsi l'hypothèse qu'une stratégie d'interaction maximisant la diversité des mouvements d'un objet combinée à l'architecture développée dans ce chapitre fait émerger une représentation des objets encodant leurs propriétés.

Le second résultat majeur de ce chapitre est que la politique naïve ne fait apparaître aucune corrélation évidente concernant la forme ou les propriétés continues. Pourtant,

dans le Chapitre IV, l'erreur de prédiction était relativement faible lorsque les propriétés étaient inférées de façon supervisée. Ceci était probablement dû à des régularités dans les observations qui montrent que notre méthode d'évaluation supervisée n'est pas parfaite. Par contre, l'absence de corrélation dans la représentation latente montre qu'il est suffisant d'interagir dans le but de maximiser l'entropie d'un vecteur d'état de l'agent pour opérer ensuite l'extraction, par une méthode non-supervisée, des propriétés d'un objet. C'est une hypothèse de plus que nous venons de confirmer, qui justifie la mise en place d'une politique apprise pour recueillir des observations sur l'objet plutôt que d'effectuer comme un grand nombre de méthodes de l'état de l'art des observations passives d'objets en mouvement.

Pour conclure ce chapitre, nous avons mis en évidence que deux architectures, l'une utilisant une cellule GRU pour encoder la trajectoire et l'autre utilisant un mécanisme d'attention, sont pertinentes pour encoder les propriétés dans l'espace de représentation. De plus, ces modèles sont robustes au bruit dans les observations et conservent une représentation de qualité malgré moins d'interactions. Cela prouve une nouvelle fois que les actions de l'agent sont particulièrement bien adaptées pour la découverte des propriétés physiques d'un objet.

VII–1.3 Utilisation des connaissances acquises

Nous avons consacré le chapitre VI à l'étude de l'influence de connaissances a priori d'un objet dans la résolution d'une tâche de manipulation. L'objectif est ici de montrer dans un cas d'application simple que la connaissance des propriétés de l'objet permet à l'agent d'être plus performant sur sa tâche. Dans l'expérience réalisée, nous évaluons un agent entraîné par une méthode d'apprentissage par renforcement sur sa capacité à utiliser des connaissances pour positionner plus efficacement un objet sur une zone cible. Étant donné que nous ne disposons pas de modèle physique de l'environnement ni de connaissance sur celui-ci, la question est de savoir si l'agent est capable d'apprendre de lui-même les relations entre les propriétés qui lui sont données et l'impact d'une de ses actions sur l'objet. Nous supposons dans nos expériences qu'en transmettant les propriétés à l'agent par son espace d'état, alors il sera capable d'apprendre de telles relations.

Dans une tâche de ralliement de point de difficulté croissante, nous avons mesuré les performances de l'agent lorsque les propriétés lui sont fournies ou non. Cette performance peut s'évaluer par le calcul de la récompense moyenne par épisode ou encore du nombre moyen de pas de simulations nécessaires. Nos premiers résultats avec un nombre d'objets de dynamiques différentes limitées, ont confirmé que l'agent est capable de tirer profit de la connaissance des propriétés. Cependant, en augmentant le nombre de propriétés et l'espace de variation de ces propriétés, nous avons observé

que cet écart de performance ce réduit. Nous avons réalisé ici qu'un conditionnement naïf d'une politique à des propriétés introduit des difficultés que nous n'avions pas anticipées. La cause reste cependant inconnue. Une des possibilités est que l'agent n'arrive pas à comprendre les relations entre les propriétés données et les objets de part peut-être un manque d'exploration.

Nous avons conclu de cette étude que l'intégration directe des propriétés identifiées dans l'espace d'état de l'agent ne suffit pas systématiquement à rendre exploitables ces propriétés dans une tâche qui en est dépendante. Il existe dans l'état de l'art des méthodes de conditionnement et de fusion pour traiter ce type de situation. Bien que nous ayons essayé l'une d'entre elles sans résultat satisfaisant pour tirer des conclusions, la question reste ouverte pour trouver la meilleure approche dans le cas d'un apprentissage de la tâche par RL. Le couplage avec les modèles des Chapitres IV et V a également été mis en place pour fournir à l'agent l'espace de représentation latent plutôt que les vraies valeurs des propriétés. Malheureusement, comme la stratégie d'application à la tâche n'est pas fonctionnelle avec les propriétés réelles des objets, nous avons laissé de côté le couplage en attendant de trouver une solution pour intégrer les propriétés. L'évaluation avec l'espace latent créé au Chapitre V est la suite logique de ces travaux pour valider le processus complet d'interaction, d'identification puis d'utilisation.

VII–2 Limites et perspectives

La méthode proposée est une ébauche d'un processus cyclique d'interaction / perception où l'agent explore et interagit en totale autonomie avec son environnement pour le comprendre, et ce afin de pouvoir réaliser des tâches plus efficacement. Dans cette thèse, nous avons dû faire de nombreux choix afin de simplifier la tâche et de valider un concept que nous souhaitions suffisamment général pour être appliqué à divers agents et environnements.

De nombreux axes restent à explorer : l'application dans un environnement encombré avec présence d'obstacles et plusieurs objets, le conditionnement de la représentation des caractéristiques physiques pour la résolution d'une tâche, l'ajout de nouvelles propriétés, l'interaction avec des objets non rencontrés en apprentissage, le transfert vers un système réel.

VII–2.1 Complexité de l'environnement

L'environnement simulé dans lequel nos expériences ont été réalisées est un terrain plat sans obstacle et sans autre objet présent. Nous sommes donc encore assez éloigné d'un environnement réaliste qui comporte en général un grand nombre d'éléments pouvant agir sur le comportement d'un objet, et même le coincer.

L'ajout d'obstacles et de multiples autres objets a plusieurs impacts sur les différentes étapes de notre méthode. D'une part, la politique MEPOL doit être entraînée pour qu'elle puisse s'adapter à ce type de situation. Il sera probablement nécessaire d'ajouter de nouveaux moyens d'interaction puisqu'un objet peut rapidement se retrouver bloqué dans un coin d'une pièce et la poussée ne permettra pas de le décoincer. De nouvelles observations vont donc s'ajouter à l'espace d'état de l'agent afin qu'il puisse manipuler l'objet en tenant compte des éléments extérieurs. Cette problématique a été anticipée par l'utilisation d'une carte d'occupation comme espace d'observation. Ainsi, il est possible de localiser l'objet par rapport à l'ensemble des éléments de l'environnement ou d'indiquer l'objet à explorer.

D'autre part, la présence d'obstacles peut être une force pour distinguer certains paramètres comme les frottements en poussant un objet contre un autre. Si la dynamique de l'objet est modifiée par des éléments extérieurs, alors il faut fournir au modèle d'identification des propriétés les observations suffisantes pour distinguer une interaction qui a été perturbée d'une interaction qui ne l'a pas été. En présence de perturbation, il n'est pas assuré que le modèle actuel arrive à encoder aussi bien les propriétés dans un espace latent. De nombreuses études dans des contextes moins réalistes ont proposé des architectures de réseaux capable de prendre en compte les interactions entre objets [Battaglia et al., 2016, Chang et al., 2017]. Cette solution pourrait être utilisée dans notre cas pour adapter notre architecture de perception prédiction.

Un environnement peut également présenter différentes hauteurs de terrain et par conséquent l'agent peut avoir la possibilité d'exploiter ce dénivelé tout comme se retrouver dans des situations bloquantes. L'espace sur lequel est mesuré l'entropie induite par la politique considère actuellement un problème plan. Mais la question est de savoir si cet espace s'adapte aisément à un problème à trois dimensions et si les capteurs de l'agent seront suffisants pour percevoir correctement l'environnement.

VII–2.2 Généralisation de la politique et du modèle de perception à de nouveaux objets

Nous avons toujours entraîné puis évalué les performances de l'agent sur des objets de mêmes formes dont les distributions des propriétés continues étaientt identiques. Nous avons considéré que l'espace des propriétés est suffisamment vaste pour ne pas tomber dans un cas de sur-apprentissage. Cependant, la capacité de la politique et celle du modèle d'identification des propriétés sur des objets jamais rencontrés en entraînement n'ont jamais été testées.

Nous sommes confiants sur le fait que la politique apprise avec MEPOL se généralise bien à de nouveaux objets de dynamique similaire à celles des objets utilisés en

entraînement. Mais qu'en est-il d'un objet très différent comme par exemple un objet pouvant basculer de manière irréversible d'un état de roulement à un état de glissement ou bloquant ? Peut-il exister un ensemble d'objets pour lequel l'entraînement permettrait à la politique d'effectuer une exploration efficace pour tous les objets possibles ? La question se pose également pour l'apprentissage du modèle d'identification des propriétés où les corrélations risquent de chuter pour des objets jamais observés.

VII–2.3 Nouvelles propriétés

Dans l'étude menée, le nombre et le type de propriétés ont été conditionnés à l'environnement pour un problème plan, sans encombrement et où l'interaction est la poussée. Mais les propriétés d'un objet ne s'arrêtent pas à sa masse et à son frottement / sa résistance. Certains sont déformables, d'autres fragiles et peuvent se briser, certains peuvent basculer ou même rebondir. L'objectif est d'être capable également d'identifier ces types de propriétés.

La question du choix de $\mathbf{s}^{\text{ent}}$ se pose alors, $\mathbf{s}^{\text{ent}}$ pouvant actuellement sembler biaisé par rapport aux propriétés que nous recherchions. Est-ce que les variations de la vitesse de l'objet seront suffisantes pour découvrir un coefficient de restitution ? Ne faut-il pas compléter cet espace par l'ajout d'autres mesures favorisant la généralisation à d'autres propriétés ? Peut-être que l'ajout d'un moyen de préhension de l'objet avec une mesure de la vitesse de l'objet selon l'axe vertical du repère monde dans $\mathbf{s}^{\text{ent}}$ est suffisant pour découvrir ces propriétés.

VII–2.4 Transfert de la simulation vers le réel

Pour chaque étape de la méthode, nous avons pris soin d'utiliser des observations les plus réalistes possible afin que l'agent puisse agir seul dans un environnement sur lequel nous ne disposons pas d'a priori. Les observations qui constituent $\mathbf{s}^{\text{ent}}$ sont les plus délicates à obtenir étant donné qu'elles demandent à définir un repère fixe et d'observer les rotations de l'objet pour exprimer correctement la vitesse dans ce repère à partir de mesures de distances réalisées par l'agent.

Tous ces efforts ont pour but de faciliter un transfert vers une plateforme réelle. Si l'espace d'entropie est trop difficile à obtenir en conditions réelles peut-être que nous pourrions entraîner cette phase en simulation puis effectuer uniquement un transfert sans fine-tuning.

Nous avons montré que malgré le bruit dans les mesures d'estimation de position, la politique et le modèle d'identification des propriétés sont capables de fournir de bonnes performances. Le bruit n'a cependant pas été appliqué lors de l'entraînement de la politique car cela pose un problème semblable à celui du Noisy-TV Problem

connu en apprentissage par renforcement. Si les mesures de $\mathbf{s}^{\text{ent}}$ sont bruitées, alors même sans bouger, l'agent est récompensé car le bruit augmente l'entropie. L'agent ne sera donc pas en mesure d'apprendre correctement à toucher l'objet. Il peut être intéressant d'évaluer d'autres signaux de motivation intrinsèque (nouveauté, surprise , ...) pour déterminer les avantages et les inconvénients de chacun et choisir la méthode d'entraînement la plus adaptée à un transfert potentiel vers le réel.

Bibliographie

P. Agrawal, A. Nair, P. Abbeel, J. Malik, and S. Levine. Learning to poke by poking : Experiential learning of intuitive physics. In *Advances in Neural Information Processing Systems 29 : Annual Conference on Neural Information Processing Systems*, 2016. xviii, 15, 53, 54, 58, 59, 60, 61, 68

J. Ajgl and M. Šimandl. Differential entropy estimation by particles. *IFAC Proceedings Volumes*, pages 11991–11996, 2011. 79

M. Al Rabbani Alif, S. Ahmed, and M. A. Hasan. Isolated bangla handwritten character recognition with convolutional neural network. In *20th International Conference of Computer and Information Technology (ICCIT)*, pages 1–6, 12 2017. 29

E. Arkin, N. Yadikar, X. Xu, A. Aysa, and K. Ubul. A survey : object detection methods from CNN to transformer. *Multim. Tools Appl.*, pages 21353–21383, 2023. 4

B. Badnava, M. Esmaeili, N. Mozayani, and P. Zarkesh-Ha. A new potential-based reward shaping for reinforcement learning agent. In *13th IEEE Annual Computing and Communication Workshop and Conference, CCWC*, pages 1–6. IEEE, 2023. 145

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv :1409.0473*, 2014. 32

R. Baillargeon. Infants' physical world. *Current Directions in Psychological Science*, pages 89–94, 2004. 14

R. Bajcsy. Active perception. *Proc. IEEE*, pages 966–1005, 1988. 9

F. Baradel, N. Neverova, J. Mille, G. Mori, and C. Wolf. Cophy : Counterfactual learning of physical dynamics. In *8th International Conference on Learning Representations, ICLR*. OpenReview.net, 2020. 52, 54, 56, 57

P. W. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, and K. Kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems 29 : Annual Conference on Neural Information Processing Systems*, pages 4502–4510, 2016. 57, 66, 163

P. Becker-Ehmck, M. Karl, J. Peters, and P. van der Smagt. Learning to fly via deep model-based reinforcement learning. *CoRR*, 2020. 6

S. Bell, P. Upchurch, N. Snavely, and K. Bala. Material recognition in the wild with the materials in context database. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 3479–3487. IEEE Computer Society, 2015. 53, 62

C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, H. P. de Oliveira Pinto, J. Raiman, T. Salimans, J. Schlatter, J. Schneider, S. Sidor, I. Sutskever, J. Tang, F. Wolski, and S. Zhang. Dota 2 with large scale deep reinforcement learning. *CoRR*, 2019. 5

A. Byravan and D. Fox. Se3-nets : Learning rigid body motion using deep neural networks. In *IEEE International Conference on Robotics and Automation, ICRA*, pages 173–180. IEEE, 2017. 53, 60

T. Chaffre, J. Moras, A. Chan-Hon-Tong, and J. Marzat. Sim-to-real transfer with incremental environment complexity for reinforcement learning of depth-based robot navigation. In *Proceedings of the 17th International Conference on Informatics in Control, Automation and Robotics, ICINCO*, pages 314–323. ScitePress, 2020. 6, 7

M. Chang, T. D. Ullman, A. Torralba, and J. B. Tenenbaum. A compositional object-based approach to learning physical dynamics. In *5th International Conference on Learning Representations, ICLR*. OpenReview.net, 2017. 57, 163

D. S. Chaplot, D. Gandhi, A. Gupta, and R. Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. In *Advances in Neural Information Processing Systems 33 : Annual Conference on Neural Information Processing Systems, NeurIPS*, 2020a. 6, 8

D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov. Learning to explore using active neural SLAM. In *8th International Conference on Learning Representations, ICLR*. OpenReview.net, 2020b. 5, 8

D. S. Chaplot, H. Jiang, S. Gupta, and A. Gupta. Semantic curiosity for active visual learning. In *Computer Vision - ECCV - 16th European Conference*, pages 309–326. Springer, 2020c. 10

M. Chareyre, P. Fournier, J. Moras, Y. Mezouar, and J.-M. Bourinet. Towards generic object property estimation using unsupervised reinforcement learning. In *IROS Workshop on Mobile Manipulation and Embodied Intelligence (MOMA) : Challenges and Opportunities*, 2022. 103

H. L. Chiang, J. Hsu, M. Fiser, L. Tapia, and A. Faust. RL-RRT : kinodynamic motion planning via learning reachability estimators from RL policies. *IEEE Robotics Autom. Lett.*, pages 4298–4305, 2019. 5

K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation : Encoder-decoder approaches. In *Proceedings of SSST@EMNLP, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111. Association for Computational Linguistics, 2014. 30

C. Colas, O. Sigaud, and P. Oudeyer. GEP-PG : decoupling exploration and exploitation in deep reinforcement learning algorithms. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, pages 1038–1047. PMLR, 2018. 17

E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2022. 11, 12, 81

F. de Avila Belbute-Peres, K. A. Smith, K. R. Allen, J. Tenenbaum, and J. Z. Kolter. End-to-end differentiable physics for learning and control. In *Advances in Neural Information Processing Systems 31 : Annual Conference on Neural Information Processing Systems, NeurIPS*, pages 7178–7189, 2018. 64

M. Deitke, W. Han, A. Herrasti, A. Kembhavi, E. Kolve, R. Mottaghi, J. Salvador, D. Schwenk, E. VanderBilt, M. Wallingford, L. Weihs, M. Yatskar, and A. Farhadi. Robothor : An open simulation-to-real embodied AI platform. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, pages 3161–3171. Computer Vision Foundation / IEEE, 2020. 7, 11

M. Denil, P. Agrawal, T. D. Kulkarni, T. Erez, P. W. Battaglia, and N. de Freitas. Learning to perform physics experiments via deep reinforcement learning. In *5th International Conference on Learning Representations, ICLR*. OpenReview.net, 2017. xviii, 53, 62, 63, 70, 71

S. Dong, P. Wang, and K. Abbas. A survey on deep learning and its applications. *Comput. Sci. Rev.*, page 100379, 2021. 4

S. Ehrhardt, Á. Monszpart, N. J. Mitra, and A. Vedaldi. Unsupervised intuitive physics from past experiences. *CoRR*, 2019. 52, 57, 58

A. El-Fakdi and M. Carreras. Two-step gradient-based reinforcement learning for underwater robotics behavior learning. *Robotics Auton. Syst.*, pages 271–282, 2013. 6

H. Fan, H. Su, and L. J. Guibas. A point set generation network for 3d object reconstruction from a single image. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 2463–2471. IEEE Computer Society, 2017. 53, 62

Z. Fang, A. Jain, G. Sarch, A. W. Harley, and K. Fragkiadaki. Move to see better : Self-improving embodied object detection. In *32nd British Machine Vision Conference, BMVC*, page 270. BMVA Press, 2021. 9, 10, 15

A. Faust, K. Oslund, O. Ramirez, A. G. Francis, L. Tapia, M. Fiser, and J. Davidson. PRM-RL : long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning. In *IEEE International Conference on Robotics and Automation, ICRA*, pages 5113–5120. IEEE, 2018. 5

C. Finn, I. J. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. In *Advances in Neural Information Processing Systems 29 : Annual Conference on Neural Information Processing Systems*, pages 64–72, 2016. 52, 59, 68

M. Fortunato, M. G. Azar, B. Piot, J. Menick, M. Hessel, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, C. Blundell, and S. Legg. Noisy networks for exploration. In *6th International Conference on Learning Representations, ICLR*. OpenReview.net, 2018. 45

S. Fujimoto, H. van Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, pages 1582–1591. PMLR, 2018. 43, 145

C. Gan, J. Schwartz, S. Alter, D. Mrowca, M. Schrimpf, J. Traer, J. D. Freitas, J. Kubilius, A. Bhandwaldar, N. Haber, M. Sano, K. Kim, E. Wang, M. Lingelbach, A. Curtis, K. T. Feigelis, D. Bear, D. Gutfreund, D. D. Cox, A. Torralba, J. J. DiCarlo, J. Tenenbaum, J. H. McDermott, and D. Yamins. Threedworld : A platform for interactive multi-modal physical simulation. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks*, 2021. 11, 12

X. Gao and T. Zhang. Unsupervised learning to detect loops using deep neural networks for visual SLAM system. *Auton. Robots*, pages 1–18, 2017. 5

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. 24

I. Goodfellow, Y. Bengio, and A. Courville. *L'apprentissage profond*. Quantmetry, 2018. 4

M. Gouko, Y. Kobayashi, and C. H. Kim. Online exploratory behavior acquisition model based on reinforcement learning. *Appl. Intell.*, pages 75–86, 2015. 15, 53, 62

M. Gouko, C. H. Kim, and Y. Kobayashi. Active perception model extracting object features from unlabeled data. In *18th International Conference on Advanced Robotics, ICAR*, pages 518–523. IEEE, 2017. 53, 67, 70

O. Groth, F. B. Fuchs, I. Posner, and A. Vedaldi. Shapestacks : Learning vision-based physical intuition for generalised object stacking. In *Computer Vision - ECCV - 15th European Conference*, pages 724–739. Springer, 2018. xvii, 52, 55, 56

T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic : Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, pages 1856–1865. PMLR, 2018. 43, 45

T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine. Learning to walk via deep reinforcement learning. In *Robotics : Science and Systems XV*, 2019. 6, 7

X. Han, H. Liu, F. Sun, and X. Zhang. Active object detection with multistep action prediction using deep q-network. *IEEE Trans. Ind. Informatics*, pages 3723–3731, 2019. 10

I. Haughton, E. Sucar, A. Mouton, E. Johns, and A. J. Davison. Real-time mapping of physical scene properties with an autonomous robot experimenter. In *Conference on Robot Learning, CoRL*, pages 118–127. PMLR, 2022. 53, 62, 69

E. Hazan, S. M. Kakade, K. Singh, and A. V. Soest. Provably efficient maximum entropy exploration. In *Proceedings of the 36th International Conference on Machine Learning, ICML*, pages 2681–2691. PMLR, 2019. 49

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, pages 1735–80, 1997. 30

S. Ivaldi, S. M. Nguyen, N. Lyubova, A. Droniou, V. Padois, D. Filliat, P. Oudeyer, and O. Sigaud. Object learning through active exploration. *IEEE Trans. Auton. Ment. Dev.*, pages 56–72, 2014. 14

M. Janner, S. Levine, W. T. Freeman, J. B. Tenenbaum, C. Finn, and J. Wu. Reasoning about physical interactions with object-oriented prediction and planning. In *7th*

International Conference on Learning Representations, ICLR. OpenReview.net, 2019. xvii, 17, 52, 56, 57

M. Jaques, M. Burke, and T. M. Hospedales. Physics-as-inverse-graphics : Unsupervised physical parameter estimation from video. In *8th International Conference on Learning Representations, ICLR*. OpenReview.net, 2020. 64

R. K. Kandukuri, J. Achterhold, M. Möller, and J. Stueckler. Physical representation learning and parameter identification from video using differentiable physics. *Int. J. Comput. Vis.*, pages 3–16, 2022. 15, 54, 64, 69

N. K. Kannabiran, I. Essa, and C. K. Liu. Estimating mass distribution of articulated objects through physical interaction. *CoRR*, 2019. xviii, 16, 53, 62, 63, 70, 71

A. I. Károly, P. Galambos, J. Kuti, and I. J. Rudas. Deep learning in robotics : Survey on model structures and training strategies. *IEEE Trans. Syst. Man Cybern. Syst.*, pages 266–279, 2021. 4

M. Khamassi, G. Velentzas, T. Tsitsimis, and C. S. Tzafestas. Robot fast adaptation to changes in human engagement during simulated dynamic social interaction with active exploration in parameterized reinforcement learning. *IEEE Trans. Cogn. Dev. Syst.*, pages 881–893, 2018. 6

H. Kim, A. Mnih, J. Schwarz, M. Garnelo, S. M. A. Eslami, D. Rosenbaum, O. Vinyals, and Y. W. Teh. Attentive neural processes. In *7th International Conference on Learning Representations, ICLR*. OpenReview.net, 2019. 116, 120

M. Laskin, D. Yarats, H. Liu, K. Lee, A. Zhan, K. Lu, C. Cang, L. Pinto, and P. Abbeel. URLB : unsupervised reinforcement learning benchmark. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks*, 2021. 17, 46, 47, 48

Y. LeCun and Y. Bengio. Convolutional networks for images, speech, and time series. In *Handbook of Brain Theory and Neural Networks*, page 3361. MIT Press, 1995. 27

Y. LeCun, Y. Bengio, and G. E. Hinton. Deep learning. *Nat.*, pages 436–444, 2015. 24

I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. *Int. J. Robotics Res.*, pages 705–724, 2015. 4, 5

A. Lerer, S. Gross, and R. Fergus. Learning physical intuition of block towers by example. In *Proceedings of the 33nd International Conference on Machine Learning, ICML*, pages 430–438. JMLR.org, 2016. 52, 56

C. Li, F. Xia, R. Martín-Martín, M. Lingelbach, S. Srivastava, B. Shen, K. E. Vainio, C. Gokmen, G. Dharan, T. Jain, A. Kurenkov, K. Liu, H. Gweon, J. Wu, L. Fei-Fei, and S. Savarese. igibson 2.0 : Object-centric simulation for robot learning of everyday household tasks. In *Proceedings of the 5th Conference on Robot Learning*, pages 455–465. PMLR, 2022. 11, 12

J. K. Li, W. S. Lee, and D. Hsu. Push-net : Deep planar pushing for objects with unknown physical properties. In *Robotics : Science and Systems XIV*, 2018. 15, 53, 54, 58, 59, 60, 61, 68

W. Li, S. Azimi, A. Leonardis, and M. Fritz. To fall or not to fall : A visual approach to physical stability prediction. *CoRR*, 2016. xvii, 52, 55, 56

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. In *4th International Conference on Learning Representations, ICLR*, 2016. 43

M. Link, M. Schwarz, and S. Behnke. Predicting physical object properties from video. In *International Joint Conference on Neural Networks, IJCNN*, pages 1–7. IEEE, 2022. 54, 64

H. Liu and P. Abbeel. Behavior from the void : Unsupervised active pre-training. In *Advances in Neural Information Processing Systems 34 : Annual Conference on Neural Information Processing Systems, NeurIPS*, pages 18459–18473, 2021. 49

M. Lohmann, J. Salvador, A. Kembhavi, and R. Mottaghi. Learning about objects by learning to interact with them. In *Advances in Neural Information Processing Systems 33 : Annual Conference on Neural Information Processing Systems, NeurIPS*, 2020. xviii, 54, 65, 69

P. Y. Lu, S. Kim, and M. Soljacic. Extracting interpretable physical parameters from spatiotemporal systems using unsupervised learning. *CoRR*, 2019. 54, 65, 66, 67, 110, 116, 119

W. Luo, P. Sun, F. Zhong, W. Liu, T. Zhang, and Y. Wang. End-to-end active object tracking and its real-world deployment via reinforcement learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 1317–1332, 2020. 6

J. Mahler, M. Matl, X. Liu, A. Li, D. V. Gealy, and K. Goldberg. Dex-net 3.0 : Computing robust vacuum suction grasp targets in point clouds using a new analytic model and deep learning. In *IEEE International Conference on Robotics and Automation, ICRA*, pages 1–8. IEEE, 2018. 5

V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State. Isaac gym : High performance GPU based physics simulation for robot learning. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks*, 2021. 7

Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra. Habitat : A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 11, 12

M. McCloskey. Intuitive physics. *Scientific American*, pages 122–131, 1983. 51

L. Meng, R. Gorbet, and D. Kulic. Memory-based deep reinforcement learning for pomdps. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pages 5619–5626. IEEE, 2021. 145

T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Sci. Robotics*, 2022. 5

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, 2013. 42, 45

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nat.*, pages 529–533, 2015. 5, 40

V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML*, pages 1928–1937. JMLR.org, 2016. 43

R. Mottaghi, H. Bagherinezhad, M. Rastegari, and A. Farhadi. Newtonian image understanding : Unfolding the dynamics of objects in static images. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 3521–3529. IEEE Computer Society, 2016a. 52, 55

R. Mottaghi, M. Rastegari, A. Gupta, and A. Farhadi. "what happens if..." learning to predict the effect of forces in images. In *Computer Vision - ECCV - 14th European Conference*, pages 269–285. Springer, 2016b. 52, 55

M. Mutti, L. Pratissoli, and M. Restelli. A policy gradient method for task-agnostic exploration. *CoRR*, 2020. 49, 77

I. Nematollahi, O. Mees, L. Hermann, and W. Burgard. Hindsight for foresight : Unsupervised structured dynamics models from physical interaction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pages 5319–5326. IEEE, 2020. 53, 58, 59, 60

D. Nilsson, A. Pirinen, E. Gärtner, and C. Sminchisescu. Embodied visual active learning for semantic segmentation. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI , Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI*, pages 2373–2383. AAAI Press, 2021. 10

L. M. Oakes and H. A. Baumgartner. Manual object exploration and learning about object features in human infants. In *IEEE International Conference on Development and Learning and Epigenetic Robotics, ICDL-EPIROB*, pages 1–6. IEEE, 2012. 14

OpenAI. Kinds of rl algorithms, 2023. URL https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html#citations-below. xvii, 41

P. Oudeyer and F. Kaplan. What is intrinsic motivation ? A typology of computational approaches. *Frontiers Neurorobotics*, page 6, 2007. 14, 17, 46

E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. C. Courville. Film : Visual reasoning with a general conditioning layer. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18)*, pages 3942–3951. AAAI Press, 2018. 156

J. Pérolat, B. D. Vylder, D. Hennes, E. Tarassov, F. Strub, V. de Boer, P. Muller, J. T. Connor, N. Burch, T. W. Anthony, S. McAleer, R. Elie, S. H. Cen, Z. Wang, A. Gruslys, A. Malysheva, M. Khan, S. Ozair, F. Timbers, T. Pohlen, T. Eccles, M. Rowland, M. Lanctot, J. Lespiau, B. Piot, S. Omidshafiei, E. Lockhart, L. Sifre, N. Beauguerlange, R. Munos, D. Silver, S. Singh, D. Hassabis, and K. Tuyls. Mastering the game of stratego with model-free multiagent reinforcement learning. *CoRR*, 2022. 5

A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. Stable-baselines3 : Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, pages 1–8, 2021. 42

S. K. Ramakrishnan, Z. Al-Halah, and K. Grauman. Occupancy anticipation for efficient exploration and navigation. In *Computer Vision - ECCV - 16th European Conference*, pages 400–418. Springer, 2020. 8

R. Rayyes. Intrinsic motivation learning for real robot applications. *Frontiers in Robotics and AI*, 2023. 14

H. Roggeman. *Amélioration de performance de la navigation basée vision pour la robotique autonome : une approche par couplage vision/commande. (Performance improvment of vision-based navigation for autonomous robotics : a vision and control coupling approach)*. PhD thesis, University of Paris-Saclay, France, 2017. 9

F. Rosenblatt. The perceptron : A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65 :386–408, 1958. 24

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning Internal Representations by Error Propagation*, page 318–362. MIT Press, 1986. 33

J. F. Schmid, M. Lauri, and S. Frintrop. Explore, approach, and terminate : Evaluating subtasks in active visual object search based on deep reinforcement learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pages 5008–5013. IEEE, 2019. 6, 10

J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning, ICML*, pages 1889–1897. JMLR.org, 2015. 43

J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. In *4th International Conference on Learning Representations, ICLR*, 2016. 43

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, 2017. 43, 45, 96

J. Schulman, B. Zoph, C. Kim, J. Hilton, J. Menick, J. Weng, J. F. C. Uribe, L. Fedus, L. Metz, M. Pokorny, R. G. Lopes, S. Zhao, A. Vijayvergiya, E. Sigler, A. Perelman, C. Voss, M. Heaton, J. Parish, D. Cummings, R. Nayak, V. Balcom, D. Schnurr, T. Kaftan, C. Hallacy, N. Turley, N. Deutsch, V. Goel, J. Ward, A. Konstantinidis, W. Zaremba, L. Ouyang, L. Bogdonoff, J. Gross, D. Medina, S. Yoo, T. Lee, R. Lowe, D. Mossing, J. Huizinga, R. Jiang, C. Wainwright, D. Almeida, S. Lin, M. Zhang, K. Xiao, K. Slama, S. Bills, A. Gray, J. Leike, J. Pachocki, P. Tillet, S. Jain, G. Brockman, N. Ryder, A. Paino, Q. Yuan, C. Winter, B. Wang, M. Bavarian, I. Babuschkin, S. Sidor, I. Kanitscheider, M. Pavlov, M. Plappert, N. Tezak, H. Jun, W. Zhuk, V. Pong, L. Kaiser, J. Tworek, A. Carr, L. Weng, S. Agarwal, K. Cobbe, V. Kosaraju, A. Power, S. Polu, J. Han, R. Puri, S. Jain, B. Chess, C. Gibson, O. Boiko, E. Parparita, A. Tootoonchian, K. Kosic, and C. Hesse. Introducing chatgpt, 2022. 32

Y. Seo, L. Chen, J. Shin, H. Lee, P. Abbeel, and K. Lee. State entropy maximization with random encoders for efficient exploration. In *Proceedings of the 38th International Conference on Machine Learning, ICML*, pages 9443–9454. PMLR, 2021. 49

D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. P. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. Mastering the game of go without human knowledge. *Nat.*, pages 354–359, 2017. 5

H. Singh, N. Misra, V. Hnizdo, A. Fedorowicz, and E. Demchuk. Nearest neighbor estimates of entropy. *American journal of mathematical and management sciences*, pages 301–321, 2003. 78

T. Standley, O. Sener, D. Chen, and S. Savarese. image2mass : Estimating the mass of an object from its image. In *1st Annual Conference on Robot Learning, CoRL*, pages 324–333. PMLR, 2017. xviii, 53, 62

R. S. Sutton and A. G. Barto. *Reinforcement Learning : An Introduction*. The MIT Press, 2018. 5, 34

R. S. Sutton, D. A. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12, [NIPS Conference]*, pages 1057–1063. The MIT Press, 1999. 42

L. Tai, G. Paolo, and M. Liu. Virtual-to-real deep reinforcement learning : Continuous control of mobile robots for mapless navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pages 31–36. IEEE, 2017. 6

R. Thoppilan, D. D. Freitas, J. Hall, N. Shazeer, A. Kulshreshtha, H. Cheng, A. Jin, T. Bos, L. Baker, Y. Du, Y. Li, H. Lee, H. S. Zheng, A. Ghafouri, M. Menegali, Y. Huang, M. Krikun, D. Lepikhin, J. Qin, D. Chen, Y. Xu, Z. Chen, A. Roberts, M. Bosma, Y. Zhou, C. Chang, I. Krivokon, W. Rusch, M. Pickett, K. S. Meier-Hellstern, M. R. Morris, T. Doshi, R. D. Santos, T. Duke, J. Soraker, B. Zevenbergen, V. Prabhakaran, M. Diaz, B. Hutchinson, K. Olson, A. Molina, E. Hoffman-John, J. Lee, L. Aroyo, R. Rajakumar, A. Butryna, M. Lamm, V. Kuzmina, J. Fenton, A. Cohen, R. Bernstein, R. Kurzweil, B. A. y Arcas, C. Cui, M. Croak, E. H. Chi, and Q. Le. Lamda : Language models for dialog applications. *CoRR*, 2022. 32

C. Tian, S. Shaik, and Y. Wang. Deep reinforcement learning for shared control of mobile robots. *IET Cyber-Systems and Robotics*, pages 315–330, 2021. 145

O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, Ç. Gülçehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. P. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver. Grandmaster level in starcraft II using multi-agent reinforcement learning. *Nat.*, pages 350–354, 2019. 5

C. Wang, J. Wang, Y. Shen, and X. Zhang. Autonomous navigation of uavs in large-scale complex environments : A deep reinforcement learning approach. *IEEE Trans. Veh. Technol.*, pages 2124–2136, 2019. 6

N. Watters, D. Zoran, T. Weber, P. W. Battaglia, R. Pascanu, and A. Tacchetti. Visual interaction networks : Learning a physics simulator from video. In *Advances in Neural Information Processing Systems 30 : Annual Conference on Neural Information Processing Systems*, pages 4539–4547, 2017. 52, 57, 58

L. Weihs, M. Deitke, A. Kembhavi, and R. Mottaghi. Visual room rearrangement. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021. 3

S. D. Whitehead and D. H. Ballard. Active perception and reinforcement learning. *Neural Comput.*, pages 409–419, 1990. 9

J. Wu, I. Yildirim, J. J. Lim, B. Freeman, and J. B. Tenenbaum. Galileo : Perceiving physical object properties by integrating a physics engine with deep learning. In *Advances in Neural Information Processing Systems 28 : Annual Conference on Neural Information Processing Systems*, pages 127–135, 2015. 54, 63, 64

J. Wu, J. J. Lim, H. Zhang, J. B. Tenenbaum, and W. T. Freeman. Physics 101 : Learning physical object properties from unlabeled videos. In *Proceedings of the British Machine Vision Conference, BMVC*. BMVA Press, 2016. xviii, 54, 63, 64

J. Wu, E. Lu, P. Kohli, B. Freeman, and J. Tenenbaum. Learning to see physics via visual de-animation. In *Advances in Neural Information Processing Systems 30 : Annual Conference on Neural Information Processing Systems*, pages 153–164, 2017. 53, 56, 64

Z. Xu, J. Wu, A. Zeng, J. B. Tenenbaum, and S. Song. Densephysnet : Learning dense physical object representations via multi-step dynamic interactions. In *RSS*, 2019. xviii, 15, 17, 53, 54, 60, 61, 62, 63, 68, 69

J. Yang, Z. Ren, M. Xu, X. Chen, D. J. Crandall, D. Parikh, and D. Batra. Embodied amodal recognition : Learning to move to perceive objects. In *IEEE/CVF International Conference on Computer Vision, ICCV*, pages 2040–2050. IEEE, 2019. 10

D. Yarats, R. Fergus, A. Lazaric, and L. Pinto. Reinforcement learning with prototypical representations. In *Proceedings of the 38th International Conference on Machine Learning, ICML*, pages 11920–11931. PMLR, 2021. 49

T. Ye, X. Wang, J. Davidson, and A. Gupta. Interpretable intuitive physics model. In *Computer Vision - ECCV - 15th European Conference*, pages 89–105. Springer, 2018. 54, 65

Q. Yu, W. Shang, Z. Zhao, S. Cong, and Z. Li. Robotic grasping of unknown objects using novel multilevel convolutional neural networks : From parallel gripper to dexterous hand. *IEEE Trans Autom. Sci. Eng.*, pages 1730–1741, 2021. 5

X. Yuan, Y. Wang, R. Zhang, Q. Gao, Z. Zhou, R. Zhou, and F. Yin. Reinforcement learning control of hydraulic servo system based on td3 algorithm. *Machines*, 2022. 145

Y. Zaky, G. Paruthi, B. Tripp, and J. Bergstra. Active perception and representation for robotic manipulation. *CoRR*, 2020. 9, 15

W. Zhao, J. P. Queralta, and T. Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics : a survey. In *IEEE Symposium Series on Computational Intelligence, SSCI*, pages 737–744. IEEE, 2020. 7

D. Zheng, V. Luo, J. Wu, and J. B. Tenenbaum. Unsupervised learning of latent physical properties using perception-prediction networks. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI*, pages 497–507. AUAI Press, 2018. xvii, xviii, 17, 54, 58, 65, 66, 67, 108, 109, 110, 111, 116, 117