Reinforcement learning-based control for safe 3D navigation of articulated tracked robot manipulators

Andrei Mitriakov

DOCTORAT BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPÉRIEURE
MINES-TÉLÉCOM ATLANTIQUE BRETAGNE
PAYS-DE-LA-LOIRE - IMT ATLANTIQUE

ÉCOLE DOCTORALE 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par
Andrei MITRIAKOV

Reinforcement Learning-based Control for Safe 3D Navigation of Articulated Tracked Robot Manipulators

Thèse présentée et soutenue à Brest, France, le 18/01/2022
Unité de recherche : IMT Atlantique, Lab-STICC, UMR 6285, team RAMBO, F-29238 Brest, France
Thèse N° : 2022IMTA0285

Rapporteurs avant soutenance :

Simon LACROIX	Directeur de Recherche LAAS/CNRS, Robotics and InterationS group
Tomas SVOBODA	Professeur Czech Technical University

Composition du Jury :

Président :	Benoit CLEMENT	Professeur ENSTA Bretagne, Lab-STICC
Examinateurs :	Simon LACROIX	Directeur de Recherche LAAS/CNRS, Robotics and InterationS group
	Anne SPALANZANI	Maître de conférences (HDR) Université Grenoble-Alpes INRIA Rhône-Alpes
	Alexandre CHAPOUTOT	Maître de conférences ENSTA Paris, U2IS
Dir. de thèse :	Serge GARLATTI	Professeur IMT Atlantique
Co-encadrant :	Panagiotis PAPADAKIS	Maître de conférences IMT Atlantique

Acknowledgement

This work would not be possible without my research supervisor Associate Prof. Panagiotis Papadakis. I address my deep and sincere gratitude to him for providing me the opportunity to do this research and for giving valuable guidance throughout the thesis. His motivation, vision, cross-domain level of expertise, dynamism, sincerity, and motivation have deeply inspired me. He has taught me the methodology to carry out the research and to present the research works as clearly as possible. I am extremely grateful for what he has offered me. It was a great privilege and honor to work and study under his guidance. I would also like to thank him for his friendship, empathy, constant attention and numerous useful discussions.

My appreciation is devoted to my chief supervisor Prof. Serge Garlatti for helping me with different aspects of my work and his kindness.

I extend my thanks to Associate Prof. Sao Mai Nguyen for her supervision in the beginning of the thesis. Her expertise in Artificial Intelligence motivated me to deepen the knowledge. I am thankful to her for plenty of valuable suggestions.

I address my special gratitude to R&D engineer Jérôme Kerdreux for his sense of humor, kindness and deep technical vision both in electronics and informatics. His help with the preparation of equipment, repair of different mechanical and electronic devices cannot be undervalued and has positively influenced the advancement of my work.

Prof. Ioannis Kannelos was always giving off positive energy and good vibes during the years of my thesis. I'm very happy to have met such a bright and kind person like him.

I express my gratitude to the members of *Comité de suivi individuel* Prof. Gasteratos Antonios and Associate Prof. Anne Spalanzani for following me throughout the years and giving valuable suggestions about work and research organization.

I wish to thank Dr. Simon Lacroix and Prof. Tomas Svoboda for accepting to review my dissertation. I particularly appreciate the work of Tomas Svoboda. Work results of his research group were the source of inspiration and initial insights in my thesis.

The direction and staff of IMT Atlantique provide perfect conditions and facilities to perform research. I address to them my great acknowledgment.

Résumé long

Les robots d'assistance ou de compagnie font l'objet d'une attention croissante de la part de la société, du commerce et de la recherche en tant que moyen d'améliorer la qualité de vie et le bien-être des personnes [1, ch. 53, 54], par exemple en soutenant les activités de la vie quotidienne [2]. Cela concerne plus particulièrement les pays développés où la baisse du taux de natalité, combinée à l'amélioration de la qualité de vie, entraîne une augmentation de l'espérance de vie et, par conséquent, un manque de professionnels humains pour accompagner les personnes âgées. Face à ce défi, des efforts constants sont déployés pour pallier les déficiences motrices ou cognitives en complétant la perte d'autonomie par la technologie robotique [3, 4]. Poussés par la nécessité de développer des robots capables de soutenir physiquement mais aussi socialement ou émotionnellement les personnes dans le besoin, des paradigmes technologiques tels que l'assistance à l'autonomie à domicile (AAD), les soins infirmiers robotisés et la robotique d'assistance ont vu le jour [5] et peuvent compléter les capacités motrices et sensorielles d'une personne de différentes manières [6].

Parmi les différents défis qui persistent pour faire des services robotiques susmentionnés une réalité, un élément crucial consiste en la capacité de naviguer en toute sécurité dans des environnements peuplés d'humains, composés de sols en 2D qui sont généralement reliés par des marches ou des escaliers. Doter les robots mobiles de capacités de navigation 3D avancées devient donc un sujet de recherche actif, tant au niveau de la conception du matériel et de la structure cinématique qu'au niveau du contrôle du comportement.

En ce qui concerne le matériel, la capacité de naviguer en 3D et de négocier des escaliers dépend étroitement du type de locomotion du robot sous-jacent, tel que *roues*, *jambes*, *hybride* ou *piste*. [7].

La locomotion des robots à chenilles représente une alternative [8, 9] car ces robots ont en général un centre de gravité plus bas et une plus grande surface de contact avec le sol, ce qui les rend comparativement plus stables. Bien que des comportements ad-hoc ou codés en dur utilisant la cinématique exacte du robot et les caractéristiques de l'escalier aient été recherchés [10, 11], une telle approche n'est généralement pas robuste car les

comportements développés ne sont pas facilement transférables à différents robots ou escaliers. Cela devient encore plus évident lorsqu'il s'agit du transport d'objets. En effet, ce problème a été, à peine, traité conjointement avec la mobilité 3D, malgré l'attention considérable qu'il a suscité ces dernières années.

La tâche de transport d'objets potentiellement sensibles lors de la traversée d'un escalier est une problématique importante dans le contexte d'un scénario d'assistance personnelle. En ce qui concerne les robots à chenilles, cette tâche a été principalement étudiée dans les applications de recherche et de sauvetage où les robots sont confrontés à des contraintes différentes de celles de la robotique de service : [9, 12, 13]. Dans ce contexte, il est essentiel de garantir la sécurité des objets ou de l'environnement. En effet, les biens matériels des personnes peuvent être facilement endommagés dans les applications robotiques d'assistance, ce qui n'est guère acceptable, contrairement aux applications de recherche et de sauvetage.

La plupart des travaux antérieurs dans ce domaine s'appuyaient fortement sur l'expertise humaine et les approches conventionnelles de planification et de contrôle. À titre indicatif, les auteurs de [9] ont proposé une infrastructure logicielle pour la planification autonome de la trajectoire et du mouvement en 3D, incluant le contrôle des flippers pour les robots à chenilles. L'idée principale est de maintenir les flippers tangentiellement à la surface de contact. Deux essais ont été suffisant pour en démontrer toute l'efficacité. Dans l'article [14], un robot à chenilles avec des sous-rampes passives affronte des obstacles qui dépassent les capacités de négociation d'obstacles de [9]. Il utilise un système d'avertissement fondé sur la marge de stabilité énergétique normalisée (NESM) et un algorithme de traversée qui utilise les flippers pour exercer une force contre les obstacles traversés. Les approches fondées sur la NESM pour l'évaluation de la stabilité se sont répandues en robotique. Elles sont faciles à comprendre car le critère de stabilité repose sur le calcul de la déviation verticale par rapport à la position stable la plus basse du châssis robotique principal [15].

En raison du manque de généralité et de robustesse de ces approches pour différents robots ou environnements, l'utilisation d'approches fondées sur l'apprentissage automatique et, plus largement, sur l'intelligence artificielle, offre de nouvelles possibilités d'exploration pour faire progresser l'état de l'art. En particulier, la branche de l'apprentissage automatique liée à l'apprentissage par renforcement (RL) s'est révélée particulièrement prometteuse en robotique pour éviter les solutions ad hoc qui sont conçues pour des plateformes particulières, ce qui la rend particulièrement pertinente pour l'application de la négociation d'escaliers par un robot à chenilles. Néanmoins, l'application de la RL peut s'avérer

particulièrement difficile lorsque les espaces d'état et d'action sont de grande dimension tout en satisfaisant des contraintes de sécurité qui ne devraient jamais ou au mieux rarement être violées [16]. Un travail de référence dans cette direction, [17] a présenté l'application de la RL pour le scénario de la traversée d'un escalier en introduisant des contraintes dans l'algorithme de recherche de politiques contextuelles d'entropie relative [18], alternant entre la simulation et la réalité.

Dans l'ensemble, les points précédents convergent vers la nécessité d'un traitement plus élaboré des divers défis communs auxquels est confrontée la navigation de robots 3D par des robots à chenilles. En retour, cela motive l'examen de questions spécifiques qui constituent l'objet de recherche principal de cette thèse.

Pour résoudre le problème de la commande de la navigation sûre en 3D de robots articulés à chenilles, nous utilisons un paradigme d'apprentissage par renforcement en utilisant une approche de la politique fondée sur la recherche comme moyen de réduire la quantité de supervision experte et d'augmenter la robustesse et la généralité sous diverses conditions tout en respectant la sécurité.

Les questions de recherche permettant de répondre à ce scénario sont les suivantes :

1. Comment pouvons-nous développer une approche fondée sur l'apprentissage de la commande de robots pour la tâche de montée et de descente d'escaliers ?

2. Comment les actions générées par le robot peuvent-elles garantir l'accomplissement de l'objectif principal de la traversée de l'escalier tout en respectant les objectifs secondaires relatifs à la sécurité ?

3. Comment la tâche de traversée d'un escalier est-elle influencée par la présence d'un bras actif portant un objet ?

4. Comment réduire l'écart entre la simulation et la réalité pour qu'un comportement appris en simulation puisse être transféré avec succès dans la réalité ?

Pour répondre à ces questions, les contributions de cette thèse se situent à l'intersection de l'apprentissage par renforcement, de l'apprentissage par transfert et de l'apprentissage incrémental, de manière à rendre le développement de comportements de navigation de robots 3D moins dépendants d'une plateforme particulière ou de l'expertise humaine et, par conséquent, plus efficaces et généralisables à la réalité.

État de l'art de la navigation intérieure 3D des robots à chenilles

Architecture du système de navigation

Le champ d'application de cette thèse étant la navigation intérieure, cela suppose la capacité de naviguer aussi bien en 2D qu'en 3D, et plus particulièrement sur le sol et dans les escaliers. Au vu de la littérature existante, nous estimons que la navigation 2D standard ou fondée sur l'apprentissage a été traitée de manière exhaustive et que de nombreuses solutions peuvent être exploitées. Ceci est vrai pour la navigation sans apprentissage, tandis que les contrôleurs réactifs de bout en bout pour la navigation à plat développés avec des algorithmes RL profonds tendent à devenir également répandus et communs.

Alors que les solutions existantes pour la navigation 2D sont facilement utilisables, un système de navigation complet nécessite le développement d'un contrôleur de navigation 3D afin de couvrir l'ensemble des tâches de négociation d'obstacles dans les environnements intérieurs. Avec deux solutions de navigation de ce type, il serait alors simple de les intégrer dans une architecture hybride. Les auteurs de l'étude [19] ont jugé cette solution plausible en soulignant qu'il est bon de mettre en place les deux méthodes conventionnelles et RL. Un planificateur diviserait un trajet à plusieurs étages en un ensemble de sous-objectifs pour naviguer d'un point à un autre sur la surface plane et du début à la fin d'un escalier pour les transitions entre les étages. Ainsi, la navigation 2D point à point pourrait être traitée indépendamment de la traversée des escaliers.

Apprendre à naviguer et apprentissage par renforcement

Les méthodes RL se divisent principalement en deux catégories : les méthodes fondées sur la valeur et les méthodes fondées sur la politique du gradient (PG). La première fait en sorte qu'un agent évalue la qualité d'une action prise dans un certain état par le calcul de la fonction de valeur en fonction de la récompense cumulative attendue. Les méthodes du second groupe ont tendance à être plus avantageuses dans les applications robotiques. Les raisons en sont la facilité de mise en œuvre grâce à la complexité réduite de l'approximation de la politique et le fait que des petits changements de politique n'entraînent pas de changements radicaux dans le comportement, contrairement aux méthodes fondées sur la valeur où un agent peut commencer à exploiter une politique nettement moins favorable après la mise à jour de la politique.

À titre indicatif, nous adoptons la méthode Soft Actor-Critic (SAC) [20], qui déplace l'espace d'action discret vers un espace continu en tant qu'amélioration retardée de la politique. Elle s'appuie sur le double apprentissage Q clippé et le lissage de la politique cible combiné à une nouvelle maximisation de la politique fondée sur l'entropie. De plus, nous utilisons aussi un algorithme antérieur, Proximal Policy Optimization (PPO) [21], qui produit des mises à jour plus conservatrices.

Les travaux [22–24] sont particulièrement inspirants en ce qui concerne la minimisation de la complexité du modèle du monde pour le paradigme réactif ou, même, la recherche d'une méthode de bout en bout, car elle permettrait d'éviter les modèles du monde complexes au niveau de la perception par une mise en correspondance directe des observations brutes avec les actions sans phase de planification coûteuse en calcul. Ils s'appuient principalement sur des algorithmes RL profonds. A notre tour, nous aimerions utiliser les mêmes algorithmes car la structure d'un problème robotique peut être naturellement considérée comme un problème de RL [25], et elle montre la capacité de développer une expertise de niveau humain [26], tout en rendant possible les méthodes conventionnelles.

Ainsi, cela nous motive à résoudre un problème d'apprentissage du contrôle en utilisant des algorithmes RL avec une représentation de politique par un réseaux de neurones. L'idée principale consiste à effectuer l'ascension du gradient sur les paramètres de la politique afin de maximiser le rendement attendu du gradient. A propos du type de contrôleur, notre ambition est de doter le contrôleur de compétences pour atteindre l'objectif avec des comportements supplémentaires.

Nous pouvons distinguer deux types de réseaux neuronaux, à savoir, les réseaux superficiels [27, p. 223] et les réseaux profonds [27, p. 436]. Les NNs peu profonds avec une couche cachée ont plus de paramètres pour approximer la même fonction que les réseaux profonds. Les réseaux profonds ont moins de paramètres et la capacité supplémentaire d'apprendre différentes représentations à des niveaux intermédiaires. Leur utilisation comme représentation de la politique dans le RL a donné lieu à des succès étonnants en robotique [28, 29], et on les appelle le RL profond ou DRL.

Les algorithmes DRL peuvent obtenir d'excellents résultats dans les simulations. Cependant, dans la réalité, leur déploiement est limité par l'écart entre le monde simulé et le monde réel, appelé problème de transfert "sim-to-real" [30].

La littérature sur le transfert sim-to-réel pour les tâches de navigation a commencé à attirer l'attention ces dernières années. De manière indicative, les auteurs de [24] effectuent une randomisation du domaine par la création de trois environnements plats avec une

complexité croissante entre eux dans le cadre du "déplacement vers le but". On considère que le robot passe à l'environnement suivant plus complexe sur la base de ses résultats d'apprentissage. L'approche présentée dans [31] entraîne un robot à six pattes à atteindre un objectif sur une surface plane. La caractéristique principale de ce travail est le mélange entre l'apprentissage du programme [32] et la randomisation du domaine, qui modifie la complexité de l'environnement en fonction d'un programme prédéfini.

Cela nous motive à incorporer des méthodes permettant de surmonter l'écart entre la simulation et la réalité. Les escaliers varient dans les habitations humaines, et par conséquent, si nous formons un robot à négocier un escalier, il ne fonctionnera pas sur un autre. Ainsi, nous cherchons à incorporer la technique DR dans notre travail pour acquérir des compétences de négociation sur autant d'escaliers que possible.

Techniques d'évaluation de la sécurité

Un des problèmes clés traités dans cette thèse est l'incorporation de propriétés de sécurité dans le comportement du robot développé. Nous distinguons deux types de traversées d'escaliers - la montée et la descente - et quantifions la sécurité différemment dans chaque cas. Nous choisissons d'associer la stabilité classique du robot [33] comme une mesure de sa sécurité dans les négociations d'escaliers ascendants.

Nous basons notre travail sur le NESM comme l'un des critères les plus développés, qui a reçu une popularité considérable à travers différentes études sur les robots chenillés [9, 14, 34]. De plus, l'analyse de stabilité statique attire notre attention en tant que moyen complémentaire pour l'estimation de la stabilité.

Nous constatons que le robot peut naturellement descendre l'escalier en appliquant une vitesse constante sans aucune reconfiguration active, ce qui peut ne pas entraîner de violation de la stabilité. Cependant, le robot peut être endommagé par l'impact d'une chute ou un choc dû à une vitesse linéaire ou angulaire élevée lorsque le robot passe les bords de l'escalier. Pour résoudre ce problème, nous nous inspirons de l'idée de détection de bumpiness de [23] où les auteurs détectent un événement de bumpiness lorsque les magnitudes de vitesse angulaire mesurées par l'IMU dépassent un certain seuil. Ainsi, les magnitudes de vitesse angulaire pourraient nous servir de moyen pour quantifier l'impact de la chute auquel le robot est soumis.

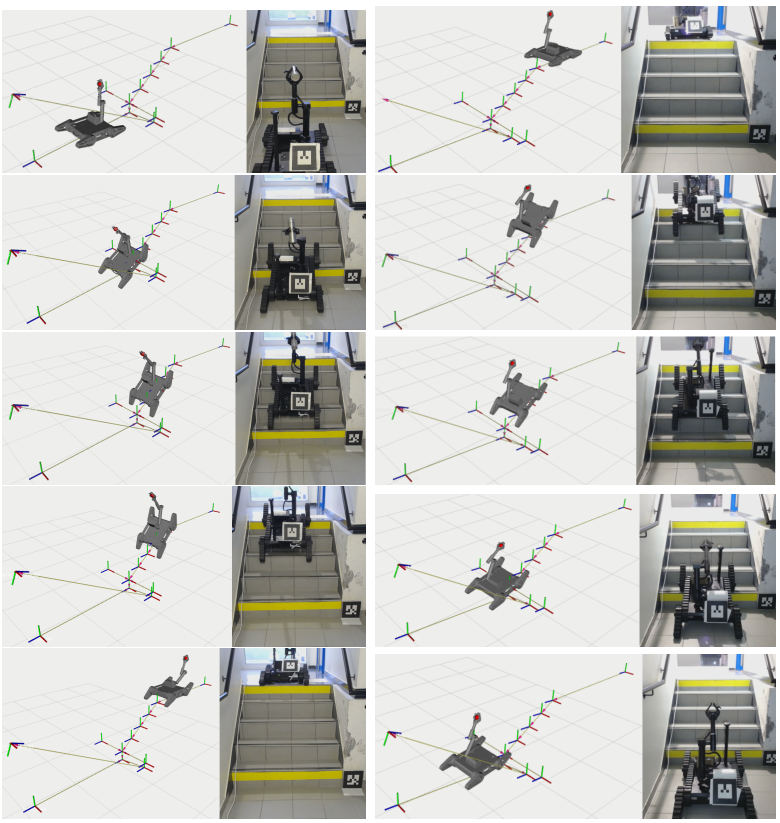


Figure 1: Instantanés de la négociation d'un escalier avec un contrôle congruent de 5 dimensions de liberté (DOF). Colonne de gauche : montée en minimisant la déviation du centre de gravité. Colonne de droite : descente en minimisant la vitesse angulaire de tangage.

Simulations de robots et d'environnements

La simulation de l'environnement et du modèle de robot est essentielle au développement des compétences des robots par le biais de la RL. Malgré une multitude d'environnements sophistiqués pour différents types de robots, les robots articulés à chenilles sont faiblement représentés, notamment dans le simulateur Gazebo, qui figure parmi les outils les plus utilisés pour le développement d'applications robotiques. Cependant, il est loin derrière Gazebo en termes d'intégration avec ROS, qui est l'état de l'art pour le développement de la robotique moderne. Nous nous appuyons sur Gazebo et ROS dans notre travail, ainsi que sur le modèle de mouvement sur la surface de contact [35]. Ce modèle trouve la force nécessaire pour déplacer le robot à une certaine vitesse. Il ne simule pas les crampons ("grouzers") ou les pistes déformables.

Prototypage d'un système de montée d'escalier fondé sur l'apprentissage par renforcement.

Au début du manuscrit, nous présentons une version initiale de l'infrastructure logicielle RL avec un environnement simplifié pour l'apprentissage du contrôle de la négociation d'escaliers. Au sein d'un petit nombre de trajectoires apprises qui correspondent, le robot a appris à traverser en toute sécurité des escaliers avec des contremarches et des angles d'escaliers variables, représentatifs du monde réel. Nous concevons et comparons trois fonctions de récompense alternatives. L'incorporation de la maximisation de la projection dans la fonction de récompense produit le comportement souhaité en termes d'adhérence du flipper à l'escalier. En même temps, la politique fondée sur le NESM se comporte mal par rapport à la politique fondée sur la maximization de projection. Les politiques apprises présentent une bonne capacité de généralisation lorsqu'elles sont appliquées à des paramètres d'escalier nouvellement rencontrés. Enfin, nous découvrons que les données sensorielles bruyantes peuvent diminuer le taux de convergence, mais que la politique de contrôle finale atteint la récompense maximale dans la plupart des cas.

Ce prototype sert de paradigme pour le développement de compétences plus élaborées du robot, afin de mieux prendre en compte la présence d'un bras actif et de traiter la seconde moitié de la tâche de négociation d'un escalier complet, à savoir la tâche plus risquée de la descente tout en portant potentiellement une charge. En outre, cette étude a permis d'identifier les points faibles de l'environnement de simulation et du robot simulé afin de faire évoluer notre infrastructure logicielle vers une simulation plus réaliste.

Montée complete d'escaliers fondée sur l'apprentissage par renforcement

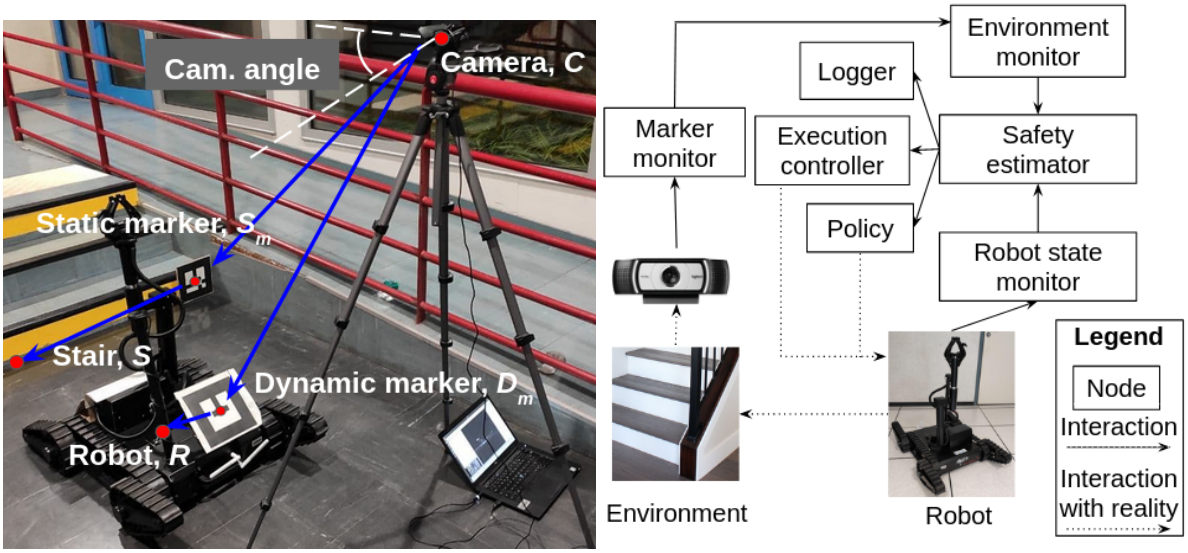
Nous utilisons un modèle CSM réaliste de chenilles qui est plus léger en termes de vitesse de calcul et plausible sur des terrains plats et accidentés. Nous instançions un modèle du robot Jaguar V4 et ajoutons une plateforme de bras manipulateur (voir Figure 1) en y incorporant ses paramètres géométriques et de distribution de masse. Les chenilles principales sont reliées au châssis par des joints fixes. Les flippers sont situées à chaque extrémité des deux chenilles et sont fixées par des articulations à rotule, qui peuvent tourner dans les limites de sécurité autour de la configuration "étendue". Les articulations à bras pivotant ont les mêmes limites de sécurité. Une charge utile de masse variable peut être fixée à l'effecteur du bras du robot. Le modèle peut être contrôlé par des topics de ROS. Les articulations du bras et du flipper sont contrôlées en position. Nous pouvons contrôler la vitesse du robot en définissant des valeurs de vitesse linéaire et angulaire. Le plugin Gazebo proposé dans [35] calcule et applique les forces correspondantes au robot. Le robot contient un capteur ROS IMU standard qui peut fournir des accélérations linéaires et angulaires ainsi que l'orientation. Enfin, nous plaçons un capteur RVB-D à l'avant du robot.

Dans cette thèse, nous instancions une formalisation et un traitement fondés sur l'apprentissage par renforcement de la tâche de négociation complete d'un escalier pour un robot activement articulé équipé d'un bras. Nous présentons une infrastructure logicielle fondée sur l'apprentissage par renforcement pour le problème de l'apprentissage de la politique de contrôle pendant la descente et la montée de l'escalier, en étudiant plus en détail l'influence d'un bras intégré, du bruit et de la présence d'une charge utile. Le robot est capable d'apprendre sa dynamique et ses contraintes de sécurité tout en négociant des escaliers variables. Nous prouvons que l'estimation automatisée du coefficient d'échelle est efficace pour contraindre les retours d'épisodes à des échelles appropriées et éviter de biaiser la convergence des politiques optimales obtenues.

Nous étudions l'optimisation des critères NESM, projection, COG et vitesse angulaire de tangage pour la montée et la descente. L'incorporation de la maximisation de la projection dans la fonction de récompense produit le comportement souhaité en termes d'adhérence du flipper à l'escalier. En même temps, la politique fondée sur le NESM se comporte mal par rapport à la politique de la maximization de projection. Les politiques apprises ont présenté une bonne capacité de généralisation lorsqu'elles ont été appliquées

à des paramètres d'escalier nouvellement rencontrés. Enfin, il a été découvert que les données sensorielles bruyantes peuvent diminuer le taux de convergence, cependant, la politique de contrôle finale atteint la récompense maximale dans la plupart des cas.

Les politiques fondées sur le COG ont montré de meilleures performances en termes de stabilité et ont qualitativement présenté le même comportement de contrôle optimal du bras en comparaison avec le contrôle ordinaire. Malgré l'ajout de plus de DOF, le contrôle du bras offre une meilleure stabilité globale au robot pendant la traversée. La minimisation de la vitesse angulaire a montré des améliorations mineures de la politique et a indirectement amélioré la déviation du COG. Le contrôle appris a montré une certaine résilience dans l'application à différentes charges de transport. Enfin, les évaluations des tests de la politique ont montré que l'optimisation du critère COG permet l'exécution de la montée et de la descente d'une manière plus sûre.



(a) (b)

Figure 2: (a) Configuration de l'expérimentation et hiérarchie du cadre de coordonnées ; (b) architecture du système robotique déployé.

Portabilité des politiques et transfert de la simulation à la réalité

Nous démontrons un transfert "zéro coup" des comportements appris en simulation vers un robot et des escaliers réels (voir figure 2), ainsi qu'une application à un second robot simulé ayant des capacités de mobilité similaires. Nous nous appuyons sur l'acquisition de contrôleurs de robot présentant les propriétés de comportement souhaitées, telles que la sécurité et la minimisation des bosses, par la conception de fonctions de récompense, conformément à nos résultats antérieurs. En particulier, nous consolidons la formation de contrôleurs efficaces en simulation, en développant et en comparant ces compétences sur deux robots articulés à chenilles distincts en simulation, puis nous transférons et déployons avec succès les politiques formées sur l'un des robots réels.

En plus d'atteindre notre objectif principal qui consiste en la négociation réussie d'un escalier par le robot réel, nous présentons également de nouvelles idées qui sont le produit d'une comparaison croisée quantitative et qualitative des comportements entre les variantes de tâches et entre les robots. Concrètement, le travail présenté fait progresser l'état de l'art sur les points suivants :

- Nous avons réussi à acquérir des contrôleurs de négociation d'escaliers basés sur RL pour deux plates-formes robotiques en simulation, toutes deux présentant les propriétés de comportement souhaitées.
- Nous comparons quantitativement les contrôleurs obtenus pour différents robots et variantes de tâches via la divergence de Kullback-Leibler.
- L'efficacité des contrôleurs obtenus est démontrée dans la réalité par le transfert à un robot commercial.

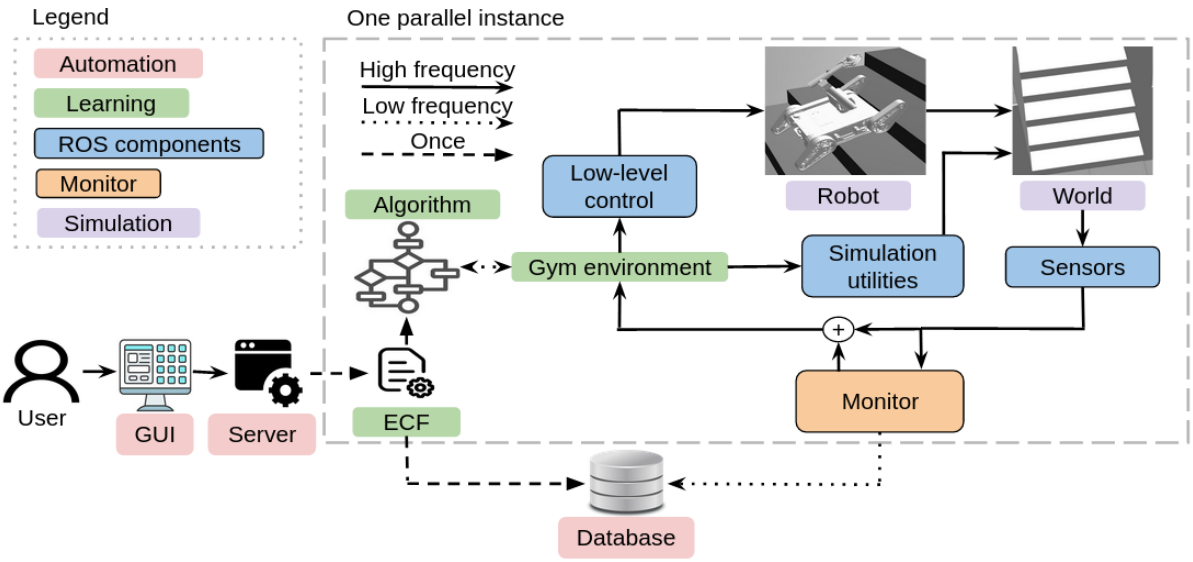


Figure 3: Vue d'ensemble de l'infrastructure logicielle proposée

Infrastructure logicielle d'apprentissage incrémental par randomisation de domaine

Une infrastructure logicielle accessible au public pour l'apprentissage et l'évaluation de la navigation dans des environnements intérieurs 3D (voir figure 3). Ce logiciel est unique en son genre en ce qui concerne le type de tâche pour lequel il est destiné à être utilisé, intégrant la randomisation du domaine et la possibilité d'un apprentissage incrémental, accompagné de deux modèles de robots articulés à chenilles.

Le logiciel appliqué à l'apprentissage du contrôle de la montée et de la descente d'escaliers avec des contraintes de sécurité a montré la capacité d'apprendre des compétences raisonnables avec le contrôle du bras articulé en se basant sur les caractéristiques de l'image de l'environnement. De plus, l'amélioration du DR avec l'échantillonnage des configurations de l'environnement à partir d'une distribution gaussienne, qui est contrôlée par l'estimation de la progression de l'apprentissage, a montré des résultats supérieurs en comparaison avec l'environnement uniforme.

Nous pensons que ce logiciel pourrait stimuler la recherche et l'expérimentation dans diverses directions. Par exemple, de futures améliorations possibles du logiciel pourraient prendre en compte la génération d'escaliers en spirale ou des variations plus complexes des obstacles au sol afin d'augmenter la complexité de l'apprentissage du contrôle 2D et de mieux prendre en compte la complexité structurelle des environnements du monde

réel. Une autre extension pourrait concerner la variation du nombre de DOF des robots dans un cadre d'apprentissage incrémental, par exemple, en commençant l'apprentissage à l'aide de 2 DOF correspondant au contrôle de la vitesse linéaire et angulaire, puis en ajoutant progressivement des DOF supplémentaires de contrôle du flipper et du bras à mesure que la complexité de l'environnement augmente.

Conclusion

Nous proposons une solution RL qui peut être appliquée sur une variété de plateformes et garantir la sécurité en quelques étapes d'apprentissage. En ce qui concerne l'applicabilité de nos résultats à la recherche et à des fins commerciales, nous promouvons les résultats de cette thèse et laissons un logiciel qui peut être utilisé pour la recherche et le développement futurs de la négociation d'escaliers et des compétences de navigation intérieure. Ce logiciel, qui incorpore nos résultats de recherche, permet d'éviter le développement d'algorithmes personnalisés pour le contrôle des robots articulés et, plus important encore, de doter le robot des comportements nécessaires, par exemple, l'orientation vers la sécurité qui a été étudiée dans cette thèse.

Malgré l'importance des résultats, notre travail souffre de plusieurs limitations. Tout d'abord, nous nous sommes appuyés sur la configuration extéroceptive dans notre travail. Pour y remédier, un des développements futurs possibles pourrait concerner la création d'un système de localisation basé, par exemple, sur la localisation et la cartographie simultanées (SLAM) [36] qui serait capable de suivre les nouvelles caractéristiques de l'escalier et de calculer la pose du robot.

Une perspective de travail possible pourrait consister à intégrer les contrôleurs fournis, un système de localisation d'escalier et la pile de navigation. Dans le cadre de ce travail, nous n'avons montré que la performance du contrôleur de bout en bout en simulation. Elle peut être étendue à la réalité.

Une autre limitation de notre travail est que nous produisons un contrôleur par tâche, cependant il est possible d'appliquer une approche hiérarchique comme dans [37] où un contrôleur étant continuellement formé sur différentes tâches traitera toutes les tâches. De même, quelqu'un peut s'inspirer de [23] et entraîner un réseau neuronal pour naviguer dans l'ensemble du bâtiment.

Table of Contents

List of Symbols

Learning

$\boldsymbol{a}$	action vector
$\boldsymbol{s}$	state or observation vector
t	time step
r	reward
γ	discount factor
e_ξ	training domain
$\boldsymbol{\xi}$	domain configuration
Ξ	space of domain configurations
R	episode return
A	action space
S	state space
τ	trajectory
c	margin of error
x	distance from the robot to the goal
Δx	travelled distance
$\boldsymbol{\theta}$	policy approximator parameters
α	learning rate
π	policy
J	expected return given the policy π
T	episode length
K	normalization coefficient
D_{KL}	Kullback-Leibler divergence
o	observation feature
σ	noise level
ι	noise level coefficient

Robot

ψ	flipper angle
ϕ	arm joint angle
v	linear velocity
w	angular velocity
p	robot position in the step frame
d	flipper length
l	chassis length
E	stability margin
I	instability margin
P	projection
D^{dev}	deviation
δ_x	projection distance along X axis
δ_y	projection distance along Y axis
θ	roll, pitch or yaw angle
F	volume
Z	inertia
ρ	density

Environment

D_i	distance where $i \in (1, 2, 3, 4, max)$
Δx	traversed distance
β	staircase inclination
D	environment size/scale parameter
W, L, C	obstacle geometrical parameters
ϵ	environment complexity parameter
Δ	configuration difference

Multi domain notations

H	height
N	number of countable units

Chapter 1

Introduction

Thanks to the recent evolution of robots, machines designed to work with and for human beings, the borders between science fiction and reality are constantly revised. Driven by decades of development of robotic technology, such machines are applied in diverse domains such as search and rescue, industry, agriculture, space, forestry and more lately in the tertiary sector [1].

Following this trend, assistive or companion robots have been receiving increasing societal, commercial and research attention as a means for improving the quality of life and well-being of people, for example by supporting activities of daily living [2]. This concerns developed countries more particularly, where birth rate decline combined with the improvement of quality of life lead to higher life expectancy and in turn lack of human professionals for accompanying seniors. In view of this challenge, consistent efforts are being made towards palliating motor or cognitive impairments by complementing autonomy loss with robotic technology [3, 4]. Driven by the need for developing robots that can physically but also socially or emotionally support people in need, technological paradigms such as ambient assisted living (AAL), robotic nursing, and assistive robotics have emerged [5] that can complement a person's motor and sensory skills in different ways [6] :

- Transportation; fetching a desired object or helping a person to move from one place to another
- Object manipulation; remote manipulation of a distance object, person feeding
- Route guidance

Among different challenges that persist in making the aforementioned robot services a reality, a crucial component consists in the capacity to navigate safely in human populated environments, composed of 2D floors that are typically interconnected by steps or staircases. Endowing advanced 3D navigation skills to mobile robots thus becomes an

active research topic both at the level of hardware design and kinematic structure as well as at the level of behavior control.

Insofar as hardware is concerned, being able to navigate in 3D and negotiate staircases is tightly dependent to the underlying robot locomotion type, such as *wheeled*, *legged*, *hybrid* or *tracked* [7]. Legged robot locomotion is instigated by biped (see Figure 1.1 (a, b)) or quadruped paradigms of mobility, which makes legged robot designs bear a significant potential for advanced 3D mobility such as staircase traversal. However, the combined effect of an elevated center of mass, small contact footprints and complex gait dynamics make the problem of safety very difficult to address. Although conventional wheel locomotion is destined for 2D navigation, hybrid wheel-based designs have been explored for obstacle negotiation tasks, for example using re-configurable pivoting segments (see Figure 1.1 (c, d)) or crawler-type platforms [38]. In view of such diversity in robot designs and locomotion types, developing control for 3D navigation for each robot instance becomes a particularly cumbersome task, be it in terms of research or engineering.

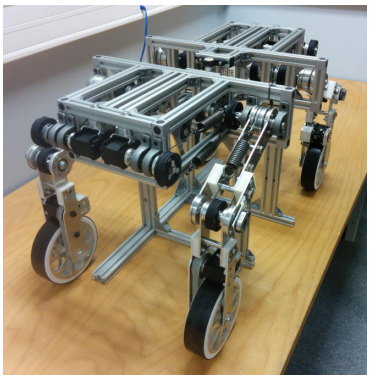
Tracked robot locomotion (see Figure 1.1 (e, f)) represents an alternative [8, 9] as such robots have in general lower center of mass and larger area of footprint contact, which make them comparatively more stable. Although ad-hoc or hard-coded behaviors that make use of the exact robot kinematics and staircase characteristics have been sought [10, 11], in general terms such an approach is not robust as the developed behaviors are not easily transferable to different robots or staircases. This becomes even more evident when object transportation is further concerned, where despite considerable attention in recent years [39–41] it has barely been treated jointly with 3D mobility.

The task of transporting potentially sensitive objects while performing staircase traversal is a topic of special interest in the context of a personal assistance scenario. As far as tracked robots is concerned, this task has been mainly studied in search and rescue applications where robots face different constraints compared to service robotics [9, 12, 13]. Crucially, ensuring object or environment safety is usually less of an issue in destructed sites whereas in assistive robot applications it is more probable and at the same time more inadmissible to damage human property.

Most earlier works in this field relied heavily on human expertise and conventional planning and control approaches. Indicatively, the authors of [9] proposed a framework for autonomous 3D path and motion planning, including flipper control for tracked robots where the main idea is to keep flippers tangentially in contact with the surface, demonstrating performance in just two trials. In [14], a tracked robot with passive sub-crawlers

(a) HRP4 [42] ©2019 IEEE

(b) HRP-4C [43] ©2010 IEEE



(c) Wheel-on-leg robot [44] ©2017 IEEE

(d) RT-Mover PType WA Mk-II [38]

(e) *Jaguar* V4 with Manipulator [45]

(f) Tracked vehicle [9] ©2008 IEEE

Figure 1.1: Robots with obstacle negotiation capabilities

faces obstacles that surpasses obstacle negotiation capabilities of [9], using a warning system based on the normalized energy stability margin (NESM) and a traversal algorithm that makes use of flippers to exert force against traversed obstacles. NESM-based approaches for stability assessment became widespread in robotics while at the same time being easy to understand because the stability criterion supposes the calculation of vertical deviation from the lowest stable position of the main robotic chassis [15].

Due to lack of generality and robustness of such approaches in different robots or environments, employing machine learning-based and more broadly speaking artificial intelligence-based approaches, provides new possibilities of exploration for advancing the state-of-the-art. In particular, the branch of machine learning related to Reinforcement learning (RL) has shown particular promise in robotics for avoiding ad-hoc solutions that are engineered to particular platforms [46], which makes it particularly relevant to the application of staircase negotiation by a tracked robot. Still, the application of RL can

prove particularly challenging when state and action spaces are high-dimensional while satisfying safety constraints that should never or at best rarely be violated [16]. A work of reference in this direction, [17] showcased the application of RL for the scenario of palette traversal by introducing constraints within the contextual relative entropy policy search (Contextual REPS) algorithm [18], altering between simulation and reality.

Overall, the previous points converge towards the need for a more elaborate treatment of the various common challenges that are faced in 3D robot navigation by tracked robot manipulators and the application of interest. In turn, this motivates the examination of specific questions that constitute the particular object of research of this thesis.

1.1 Research questions and contributions

To address the problem of control for 3D safe navigation of articulated tracked robot manipulators, we pursue a reinforcement-learning paradigm using a policy search-based approach as a means for reducing the amount of expert supervision and increasing robustness and generality to various conditions while respecting safety.

Different questions are raised to accommodate this scenario. How can we develop a learning-based approach for robot control for the task of staircase ascent and descent? How can the generated robot actions ensure the accomplishment of the primary goal of staircase traversal while respecting secondary goals pertaining to safety? How is the staircase traversal task influenced by the presence of an active arm carrying an object ? How can the gap between simulation and reality be reduced so that a behavior learnt in simulation can be successfully transferred to reality ?

To address these questions, the contributions of this thesis reside at the intersection of reinforcement-learning, transfer learning and incremental learning, so as to render the development of 3D robot navigation behaviors less dependent on a particular platform or human expertise and in turn, more efficient and generalizable to reality. In particular, the major contributions of this thesis dissertation are as follows :

1. **A reinforcement-learning based formalization and treatment of the complete staircase negotiation task for an actively articulated robot equipped with an arm.** In particular, we successfully tackle the problem of staircase ascent and descent by satisfying the primary goal of traversing the staircase as well as the secondary goal associated to safety constraints, using congruent control of the degrees of freedom associated to the tracks, flippers and arm joints.

2. **Demonstrated, zero-shot transfer of the behaviors learnt in simulation to a real robot and staircases, along with an application to a second simulated robot of similar mobility skills.** The transfer to reality and the application to two robots corroborate the generality and robustness of contribution #1.

3. **A publicly available software framework for navigation learning and evaluation in 3D indoor environments.** Its architecture allows a user to interface between different RL libraries and algorithm implementations. At the same time, learning can be customized to endow specific properties within a control skill. To

show its utility, we focus on the case of staircase ascent and descent using depth sensory data while respecting safety via reward function shaping.

1.2 Manuscript organization

The organization of the dissertation is as follows:

In **Chapter** 2, we review the state-of-the-art in the areas of interest of the dissertation highlighting the relevant tracked robot control architectures.

The subsequent chapters unfold the details of the contributions of this thesis. Due to the complexity of the overall problem, these contributions were sought during the thesis by starting from moderate hypotheses and constraints and progressively scaling up to harder challenges until the final application to the real robot. We chose to follow a similar mode and order of presentation in the organization of the manuscript, so as to better comprehend the different milestones that lead to the final contributions. An implicit, secondary contribution of the dissertation could thus consist in guiding the reader in the development of similar approaches in an incremental and principled manner.

Chapters 3 and 4 present the details of the claimed contribution #1. In **Chapter** 3, we formalize the problem of the staircase traversal and start by developing a wheel-based robot model within a simplified reconfigurable staircase environment. This leads to a prototype application where the integration of constraints within a policy is successfully employed through penalty introduction. In the sequel in **Chapter** 4, we proceed by introducing more accurate, simulated robot models using exact geometric characteristics and a robotic arm. The degrees of freedom of the arm are then introduced within the control learning problem, we extend the staircase negotiation problem to further account for staircase descent and identify adequate safety criteria for either ascent or descent. Using the proposed environment, we present extensive qualitative and quantitative results where simulated robots learn to negotiate staircases of variable size while being subjected to different levels of sensing noise.

Chapter 5 consolidates contribution #2 of the dissertation. Here we present the applicability of our framework on a real robot and juxtapose obtained performances in two simulated robots. We qualitatively and quantitatively discuss the comparison of the policies obtained on different platforms. Through variations of the task and the conditions of its execution, we demonstrate the framework's robustness at different levels of risk, stochasticity and control dimensionality.

Chapter 6 presents the epitome of the previous results and insights by providing an incremental learning-based, software framework based on standard tool-kits and middleware and its application to end-to-end control learning for 3D staircase negotiation in

simulation (contribution #3). Quantitative and qualitative results show favorable results when the robot makes use of depth images.

Chapter 7 concludes the dissertation by discussing its impact, remaining challenges and questions to be addressed in future developments.

Chapter 2

State-of-the-art in 3D Indoor Navigation of Tracked Robots

This chapter discusses related works that motivated and guided the developments and contributions sought through this dissertation. In Section 2.1 we specify the scope of our research topic within a general navigation framework. Then, in Section 2.2 we briefly overview learning approaches in navigation and present RL notions.

We present the hardware of modern tracked robots and their navigation behavior in Section 2.3, showing the evolution of tracked robots, their complexity, the various problems encountered towards increasing their autonomy, which altogether motivated our research goals. Given that one of our contributions concerns the integration of safe behaviors in robot control, we devote Section 2.4 to this theme. In Section 2.5, we conclude this chapter by presenting available simulation environments, suited for articulated robot modeling.

2.1 Fundamental navigation system architectures

Understanding the landscape of robot architecture paradigms and general navigation systems was instrumental in our study, as it precedes our preference for the development of a learning-based approach for control skills acquisition.

According to [47], there exist three main components in a mobile robot navigation system. The first one is *Sense* and performs data collection. This concerns not only exteroception but also proprioception to know the robot state as well. The second is *Plan* and follows the sensory data acquisition step, producing a path that optimizes a given criterion. Finally, the *Act* component is responsible for translating the plan to real-world actions that can be executed by the robot.

Within the component *Sense*, the robot has to perceive the environment, create its representation and localize itself within the environment. Commonly, this problem is referred to as simultaneous localization and mapping (SLAM) [36] and is a problem of fundamental importance in mobile robotics. To solve it, we can rely on diverse SLAM techniques [48–51], which manage to produce a map of the environment and estimate the robot pose, both of which are necessary to perform path planning.

The *Plan* component concerns the task of path planning within a navigation system, that can be performed at different spatio-temporal horizons, which allows to deal with the question of how to move from one place to another at different scales [52]. In local path planning one calculates the path accounting for the presence of dynamic and newly encountered objects. On the other hand, in global path planning, we are concerned with the global, static structure of the environment. Usually, both path planners implement an algorithm from the following groups: graph search-based algorithms [53, 54], rapidly-exploring random trees [55] or probabilistic roadmaps [56].

Once a path is found, the robot employs the *Act* component to execute the path. This component can be associated in different ways with respect to the previous two components. The first attempt resulted in the so-called hierarchical architecture presented in Figure 2.1 (a) and was commonly applied in early robotics works [57]. At step *SENSE*, the robot senses the environment, creates its model if necessary and localizes itself. Thereupon, the *Plan*-based module takes the perceived model and the goal and plans the directives to reach the goal. At *ACT*, the robot generates actions and actuator commands according to the calculated plan. However, the main inconvenience of this approach is the cost of world modeling and of the planning phase.

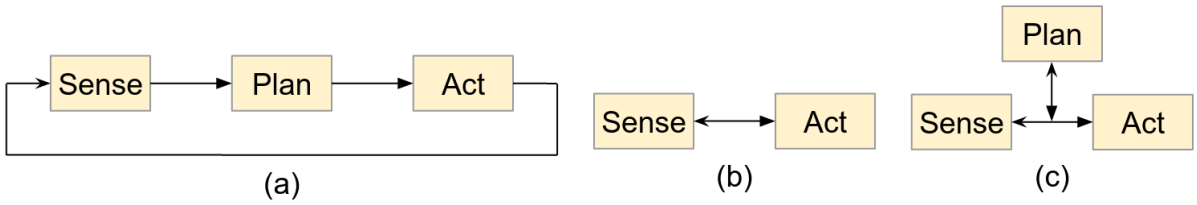


Figure 2.1: Robotics systems architectures: hierarchical architecture (a), reactive architecture (b), hybrid architecture (c)

These drawbacks motivated the development of the reactive architecture presented in Figure 2.1 (b), which omits the planning phase and results in no world model, making the sensing and acting phases tightly coupled. This architecture results in fast actions in the real world that make it efficient. For example, the subsumption architecture [58] enhances the purely reactive architecture through the layered structure of behaviors.

The authors of [59, p. 33] notice that the subsumption architecture could be directly mapped to behavior trees. They show tele-reactive programs, which allow the system to achieve a specific goal and to be responsive to environment changes defining the behavior of a robotic system. The tele-reactive program is represented by a set of a condition-action, where an action is usually durative, rules that directs the agent to a goal, while a sub-set of programs monitor the environment. It possesses an intuitive structure that is particularly responsive to environment changes, however, it is difficult to maintain and handle failures. The inconvenience of the reactive approach is, naturally, the absence of plan capability, making it challenging to achieve long-term goals.

To alleviate the problems of reactive and hierarchical architectures, the hybrid architecture [47] was proposed (see Figure 2.1 (c)). The robot creates a model of the environment and decomposes the accomplishment of the long-term goal into a set of sub-goals where each has an attributed behavior. The robot operates in a reactive manner moving between sub-goals executing a behavior, yet the *Plan* component interacts with it in three ways:

- Providing information on which the reactive layer acts;
- Changing the world state or updating the robot behavioral parameters;
- Interacting with the reactive layer in real-time.

The scope of this thesis being on control for 3D navigation, we concentrate on the control-related *Act* component of a robot architecture. Tzafestas et al. [60, 61] recently

provided a global overview of mobile robot control and navigation methodologies, identifying different families of methods: standard controllers, adaptive, robust, fuzzy, vision-based and neural network-based (NN) controllers. We refer in the sequel to a couple of those works to show the evolution in concepts and paradigms used.

For example, the authors of [62] implemented a feed-forward, pure-pursuit controller [63] to follow the path, wherein the *Sense* component relies on machine learning for traversability estimation. The work [64] presents the system that won in the Defense Advanced Research Projects Agency challenge. The developed software is broken down into several layers, where the control layer plans the trajectory in velocity and steering space, passed through two trajectory tracking controllers responsible for steering and throttle/brake control.

Alternatively to conventional learning-free controllers, NN-based controllers are receiving increasing attention in recent years [46]. Indicatively, the approach presented in [23] proposes a fully learning-based navigation system that operates outdoors while being trained on-line. Using a self-labeling technique, the authors develop a predictive model, which enables the system to search less-bumpy and collision-free paths towards the goal. On the other hand, the system necessitates considerable interaction with the environment, making it less suitable for indoor applications where collisions are unacceptable. Still, it can be thought as one of the most noticeable works of autonomous control.

Tai et al. [22] present a system capable of map-less navigation through end-to-end learning. They transform raw laser data into sparse rays, group up the latter with the goal position in the robot frame and pass it through a "map-less motion planner" to obtain linear and angular velocity commands. The trained planner is represented by a combination of dense NN layers [65, ch. 6] and trained by Deep Deterministic Policy Gradient algorithm [66]. The reward function guides learning towards the acquisition of skills to move the robot to the goal while avoiding obstacles.

Chaffre et al. [24] also present a robot for map-less 2D navigation. The difference in comparison to [22] is that they rely on the depth image observations and perform the policy update through Soft Actor-Critic (SAC) [20].

2.1.1 Recapitulation

The scope of this dissertation being on 3D indoor navigation, this supposes the capacity to navigate in 2D as well in 3D, and more specifically, onto floor and staircases. In view of the existing literature, we deem that standard or learning-based 2D only navigation has

been exhaustively addressed and numerous solutions can be exploited.

While existing solutions for 2D navigation are readily available to use, a complete navigation system requires a 3D navigation controller to be developed in order to cover the entire spectrum of obstacle negotiation tasks in indoor environments. Given two such navigation solutions, it becomes straightforward to integrate them into a hybrid architecture. That was assessed plausible by the authors of [19] who summarize that it is a good practice to match conventional and RL methods. A planner would divide a multi-floor path into a set of sub-goals to navigate point-to-point on the flat surface and from the beginning to the end of a staircase in inter-floor transitions. Thus, point-to-point 2D navigation could be treated independently from staircase traversal.

2.2 Learning to navigate

Guastella et al. [67] present a general overview of learning-based methods adopted for unmanned ground vehicles. Accordingly, learning-based methods can decomposed into two groups: end-to-end methods and terrain traversability analysis (TTA) [68] (see Figure 2.2), matching the distinction made within [46] as well. As we can see in that figure, TTA methods involve all three navigation components presented in Figure 2.1. End-to-end methods directly map the component *Sense* to the component *Act* relating to the reactive architecture.

TTA refers to the robot capability to move through terrain accounting for world and vehicle models, kinematic constraints, and eventual optimization criteria. Accordingly, it involves the *Plan* component. However, the problem of staircase negotiation is not complex in terms of planning, since the goal is located either upstairs or downstairs. Thus, in our work,we can overlook the *Plan* phase supposed by the TTA methods and focus more on the lower-level control which could generate actions in a reactive manner.

End-to-end methods directly map robot observations to actions putting perception and control into a single block leading to the reactive control paradigm and commonly employing supervised, self-supervised, imitation learning, or reinforcement learning.

Guastella et al. distinguish the main differences in training between common learning techniques. Supervised learning demands labeling of training data by an expert. Self-supervised learning assumes labeling of training data by an autonomous tool. Imitation learning capitalizes on expert demonstrations. RL relies on a rewarding environment and, commonly, artificial neural networks for policy approximations in a deep learning setting.

The authors of [46] have recently advocated the application of deep learning (DL) techniques in mobile robotics. They show that learning techniques are primarily used in robotics perception for object detection and environment/place visual recognition and control tasks. Most interestingly, DL is used in robotic control, where the authors organize

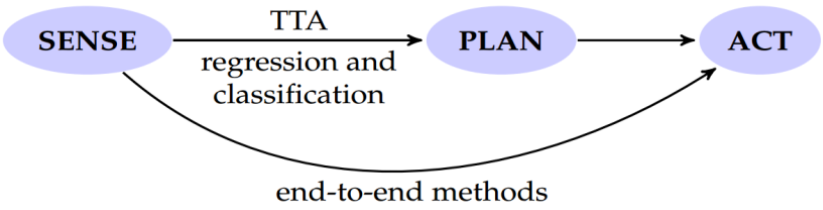


Figure 2.2: TTA and end-to-end methods comparison [67]

control problems treated by DL into three groups:

- Deep data-driven sensory-motor system;
- Deep model-predictive control;
- Deep reinforcement learning (DRL).

Deep data-driven sensory-motor models address the problem of visual feedback control, where the raw sensor inputs are directly related to the control commands, often through training of deep neural networks and convolutional neural networks (CNN) in a supervised way. Deep model-predictive control tackles complex non-linear dynamics of the robotic tasks learning a dynamics model and producing optimal control. Authors particularly mention that DRL suits well robotic tasks, because they both share the same temporal nature.

Works [22–24] are particularly inspiring in minimizing the complexity of the world model for the reactive paradigm through direct mapping of raw observations to actions without any computationally expensive planning phase. In our turn, we will partly rely on the paradigms put forward by such works because the structure of the robotic tasks that we address can be naturally viewed as a RL problem [25] and RL has shown the ability to develop human-level expertise [26].

2.2.1 Reinforcement learning

The key idea behind RL is the interaction between an agent and the environment in which it operates. Being in a particular state, the robot interacts with the environment through an action defined by a policy, transits to the next state receiving a reward issued from the reward function, which guides learning of the robot and can additionally incorporate penalty terms allowing to obtain a policy with desired properties. The objective of RL is to maximize the expected reward by learning an optimal policy.

RL methods could be mainly divided into **value-based** and **policy gradient** (PG) methods. The first makes an agent evaluate the quality of an action taken in some state by calculation of the value function in accordance with the expected cumulative reward. Deep Q-Network (DQN) was one of the first deep RL algorithms and had shown fascinating results at that time [26]. Over the years, RL algorithms have witnessed a significant evolution. The current state-of-the-art is composed of actor-critic [27, p. 331] algorithms

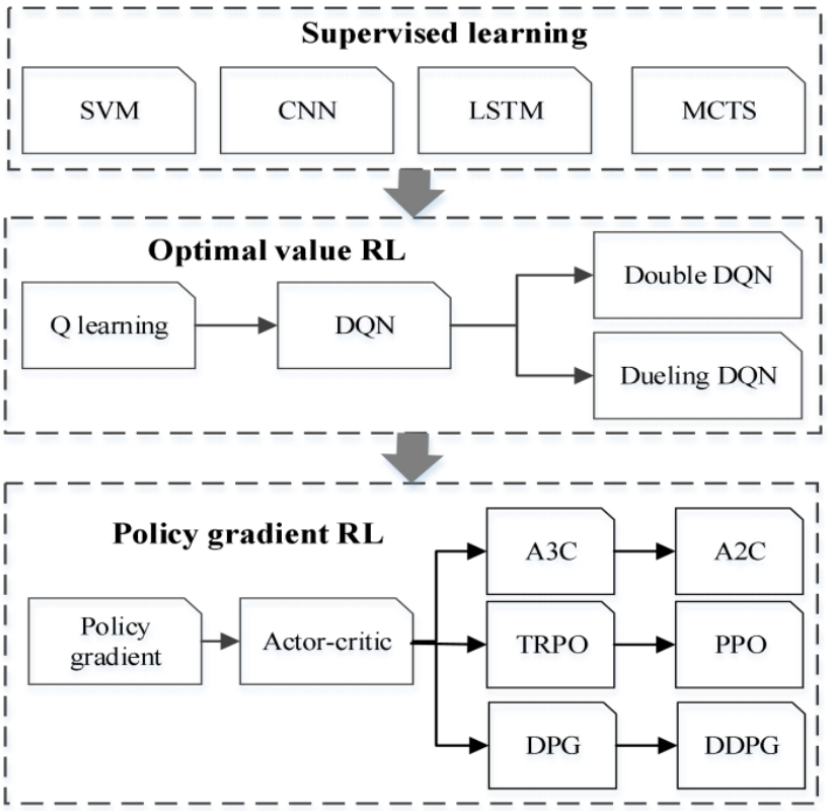


Figure 2.3: Evolution of interest in learning-based algorithms for navigation learning (cf. Zhou et al. [19, Figure 5])

that benefit from state-action value estimation as in pure value-based DQN and directly optimize policy parameters through gradient ascent.

Methods of the second group tend to be more advantageous in robotic applications [25, 69]. The authors of [19] track the evolution of attention towards RL (see Figure 2.3) in motion planning learning. The reasons for this are ease of implementation thanks to reduced complexity of policy approximation and the fact that small policy changes do not lead to drastic changes in behavior in contrast to value-based methods where an agent can start to exploit a significantly less favorable policy after policy update.

Indicatively, the off-policy [27, p. 257] Deep Deterministic Policy Gradient (DDPG) [66] algorithm moved discrete action space to a continuous one as an approximate DQN. Still, the significant improvement of DDPG, which has introduced Twin-Delayed DDPG (TD3) [70], corresponds to enhanced DDPG with delayed policy improvement, clipped double Q-Learning, and target policy smoothing. In parallel to the latter algorithm,

SAC [20] was proposed, which contains the same features combined with a novel policy entropy-based maximization. An earlier on-policy algorithm is Proximal Policy Optimization (PPO) [21] which yields more conservative updates. We will consider SAC and PPO algorithms as state-of-the-art in the work of this dissertation and use them to learn the various desired policies.

Policy approximators

Applying RL to robotics, it is unavoidable to use function approximators [25] due to high-dimensional action and observation spaces or the necessity to operate within continuous states and actions.

In designing a controller learned via RL, several alternatives can be considered [27]. Approximators such as tile or coarse coding tend to suffer from the curse of dimensionality and do not generalize well due to sensitivity to grain size. The main advantage of artificial neural networks over other function approximators is their generalization capability and straightforward increase of depth which was well studied.

PG algorithms directly optimizes policy parameters, which can be achieved in any way if the policy is differentiable to its parameters. A common choice is radial basis function RBF networks which were widely applied before the emergence of artificial neural networks. RBF networks converge quickly to global optima with fewer trials and errors. Still, they have proved of limited use due to the absence of tractable and stable integration into more complex neural network architectures.

We can distinguish two types of neural networks, namely, shallow [27, p. 223] and deep [27, p. 436]. Shallow NNs with one hidden layer have more parameters to approximate the same function than deep networks. Deep NNs have fewer parameters and the additional ability to learn different representations at intermediate levels. Their usage as a policy representation in RL led to amazing success in robotics [28, 29].

The above observations motivate us to address the control learning problem using RL algorithms, eventually using a NN policy representation. The main idea consists in performing gradient ascent over policy parameters to maximize an expected episode return. In the scope of this thesis, we considered the employment of DL to be disproportionate to the relatively small observation space, as will be shown in Chapter 3. Using DL becomes more pertinent as one shifts towards end-to-end learning, a direction that we only lately started to investigate in Chapter 6.

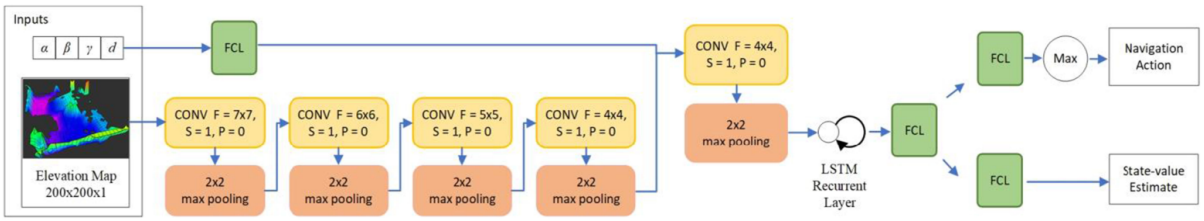


Figure 2.4: NN structure used in [71] ©2021 IEEE, where CONV represents a convolutional layer, FCL is a fully connected layer, filter size F, stride S, and padding P (cf. Goodfellow et al. [65, chapters 6 and 9]).

Simulation-to-reality transfer

RL algorithms can exhibit excellent results in simulations. However, their direct deployment in reality is limited by the gap between the simulated and real worlds, referred to as the "sim-to-real" transfer problem [30]. System identification is a natural method to overcome the discussed gap by creating a precise mathematical model of the robot within its environment. This is challenging and not always possible because of the necessity to account for varying conditions like temperature, humidity, and others. Learning with disturbances introduces environmental and reward perturbations in the simulation. Domain adaptation exploits the idea of source domain data usage for training in the target domain. This method is more broadly used in vision-based tasks. Domain randomization (DR) represents a method which aims to randomize the simulation environment parameters, so that they cover real-world parameter distribution. The authors distinguish visual randomization, where camera position, texture, and lighting vary, and dynamics randomization. The latter alters objects and robot link masses, object dimensions, robot joint damping coefficients, actuator force gains, and surface friction coefficients.

The sim-to-real transfer literature for the navigation tasks starts attracting more attention during last years. Indicatively, the authors of [24] perform domain randomization through the creation of three flat environments with increasing complexity between them in the "move to the goal" setting. The robot is considered to pass to the next more complex environment on the basis of its learning results. The approach presented in [31] trains a six-legged robot to reach a goal on the flat surface. The key feature of this work is the mix between curriculum learning [32] and domain randomization, which change the environment complexity according to a pre-defined curriculum.

Hu et al. [71] develop a navigation system based on deep NN and Asynchronous Ad-

vantage Actor-Critic [72]. This system achieves point-to-point navigation in 3D cluttered environments by a wheeled robot. One of the key features of this work is using the elevation map in the observation space and the robot pose. The rewarding process guides learning to the goal and avoids undesirable states. Another feature of this work addresses the simulation-to-reality gap in the scope of point-to-point navigation learning through domain and dynamics randomization. The authors first vary the terrain steepness, which allows avoiding unfeasible paths during deployment in reality. Then, they perform the motion disturbance where they apply disturbance represented as uniform distributions to the travel distance and the yaw rotation angle of the robot. Finally, they add Gaussian error to the pose estimation to treat eventual inaccurate localization in reality.

This section highlights our interest in methods enabling us to overcome the gap between simulation and reality. Staircases vary in human livings, and therefore, if we train a robot to negotiate one staircase, it would not work on another one. Thus, we will seek the integration of a DR stage in our work in order to endow the staircase negotiation skills with increased generalization and robustness when deployed in the real-world.

2.3 Tracked robot control

Robot navigation and control are dictated by the environment wherein the robot operates. As obstacle negotiation in outdoor and indoor environments may require advanced traversability skills, tracked robots equipped with additional sub-tracks (flippers) have been developed. More recently, the obstacle negotiation potential of such platforms is being studied for improving the autonomy of frail people, either for transporting humans or objects in 3D environments.

This has incited a particular interest within the scientific community for modeling the kinematics and dynamics of such robots in order to better exploit their potential. Although the articulated front and rear flippers drastically increase the negotiation capability of complex obstacles, they inevitably increase the complexity of control, kinematics calculation, and estimation of robot dynamics. Furthermore, the presence of a robotic arm increases even more the difficulty of control due to additional degrees of freedom (DOF) and safety constraints.

Among various conventional control methods reviewed by [60], neural-network-based control without reliance on expert, emerges as a means for learning effective controllers for high-dimensional observation and action spaces or decision making from raw input data such as pixels [67]. In the sequel, we review representative works from **learning-free (LF)** and **learning-based (LB)** categories that treated the problem of indoor navigation of tracked robots while highlighting limitations and open challenges. For reference, Table 2.1 provides a coarse overview of those works that we find the most representative.

2.3.1 Conventional control

Before the development of autonomous staircase negotiation controllers, the focus was directed onto remote robot control. Authors of [73] notice difficulties that a teleoperated articulated robot is unstable in an open-loop control due to limited sensor indications available to an operator and track slippage. Therefore, a stabilizing feedback controller for semi-autonomous staircase negotiation is proposed for *Andros Mark VI* (see Figure 2.5 (d)) to prevent misalignments which are due to track disturbance, gravity, and inherent differential track velocities. Accelerometers issue an acceleration signal proportional to the sine of heading, which is then passed through a low-pass filter, and heading deviation is calculated. This is then used to correct the heading for ascent stabilization of a teleoperated robot, making this work one of the very first to treat the staircase negotiation

Learning-free			
Work	**Method**	**DOF**	**Input data (sensors)**
Martens et al. [73]	Stabilizing feedback controller	1	Accelerometer
Helmick et al. [10]	Pole placement	3	LADAR, gyroscopes
Mourikis et al. [8]	State-feedback linearization	3	RGB camera,gyroscope
Nagatani et al. [9]	Fuzzy, vision-based	4	Laser sensors
Ben-Tzvi et al. [74]	Fuzzy	3	Motor encoders,compass
Colas et al. [75]	Fuzzy	4	Lidar, IMU
Zhang et al. [76]	Fuzzy, resolved motion rate controller	3	Gyroscope
Gianni et al. [77]	Feedback control	6	IMU, lidar
Endo et al. [78]	Proportional-integrative	4	IMU
Xie et al. [79]	Fuzzy	4	IMU
Oehler et al. [80]	Optimization	10	Odometry, IMU, lidar
Learning-based			
Zimmermann et al. [81]	Imitation learning, RL	4	Lidar, 3 cameras, IMU, GPS
Ejaz et al. [82]	Neural, RL end-to-end	2	Depth camera
Pecka et al. [17]	Neural, RL	2	IMU, laser sensors

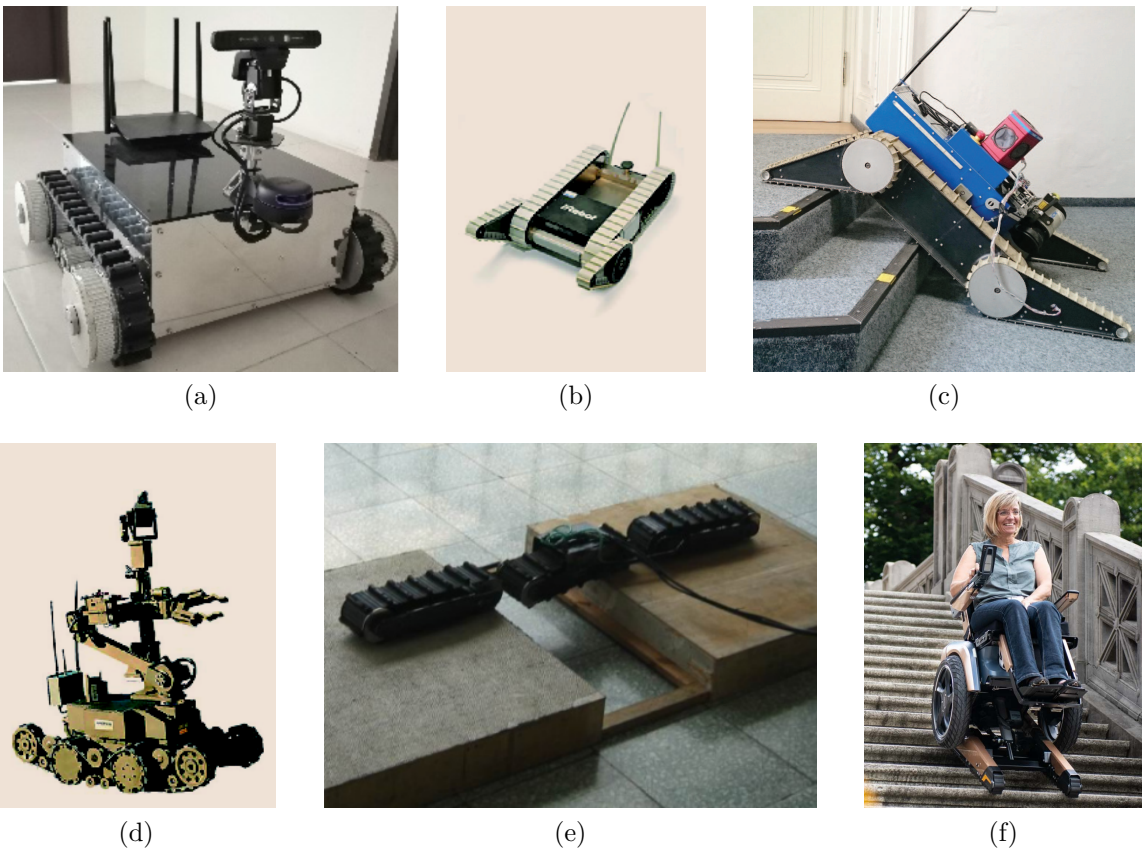
Table 2.1: Overview of related works

problem.

The analysis of a tracked mobile robot stair-climbing ability is proposed in [83] (see Figure 2.5 (e)). The authors evaluate the kinematics and dynamics of a specific modular robot architecture in three phases of the climbing process: riser climbing, riser crossing, and nose line climbing. One of the main drawbacks is that the modular mechanical design of the robot is unique in its kind and was not encountered in other works.

Authors of [10] develop a Kalman filter for laser and image data fusion and the attitude estimate (see Figure 2.5 (b)). To control a tracked robot in the staircase ascent, they create a physics-based controller which minimizes the heading error and optimizes the velocity. The proposed method is versatile, accounts for imagery data and localization techniques but still, it controls only 2 DOFs without accounting for flipper control.

Another early work was presented in [8] acclaiming robustness combined with performance under real-world conditions. The authors developed an extended Kalman filter for orientation and the offset from the staircase center estimation. This information is used by a centering controller, which provides the reference rotation signal. Using this information, the authors design a state-feedback heading controller which calculates the



(a) (b) (c)

(d) (e) (f)

Figure 2.5: Tracked robot mechanical complexity variation: (a) simple [82] ©2020 IEEE, (b) front flippers [84][1], (c) front and rear flippers [17] ©2016 IEEE, (d) flippers and manipulator [84][1], (e) custom design [83] ©2005 IEEE, (f) *Scewo Bro* wheelchair `https://scewo.ch/en/bro/`

required rotational velocity to align the robot with the heading direction computed by the centering controller. This work shows noticeable results. However, staircase descent was not considered, and the heading controller is customized for a specific platform. More critically, this work only addresses adjusting the angular velocity to keep it aligned to the staircase and keep the robot close to the centerline.

A highly customized *Linkage Mechanism Actuator LMA* (see Figure 2.6) tracked mobile robot was presented in [74]. The tracked robot consists of two main tracks, which contain actively articulated pendulums or, as it is referred to in work, flippers. It carries only a 3D compass for inclination estimation. Additionally, a rule-based algorithm per-

[1]Reprinted/adapted by permission from Springer Nature Customer Service Centre GmbH: Springer Nature, Robotics in Hazardous Applications by James P. Trevelyan, Sung-Chul Kang, William R. Hamel ©2008

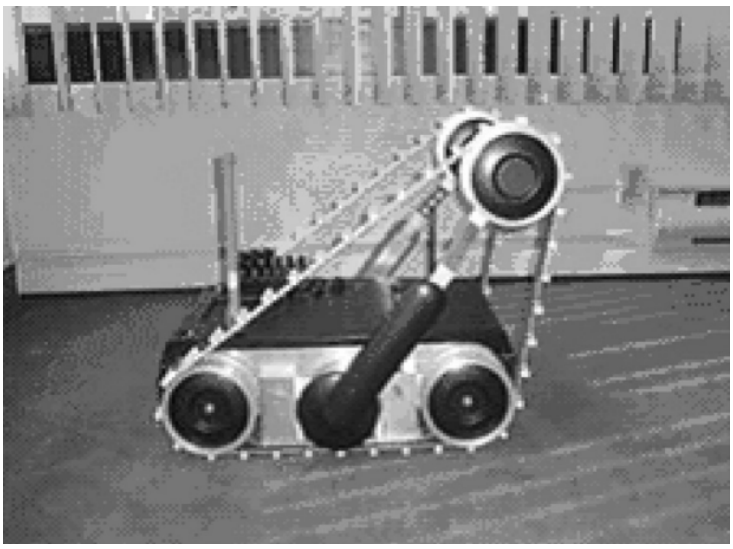


Figure 2.6: Side view of the *LMA* [74]

forms the step height estimation. The robot is destined to ascend and descend staircases. To do so, authors implement rule-based control algorithms which correspond to different stages of the traversal, such as the "riding on nose line," "going on nose line," and "landing" stages. Due to the particularities of this robot design, however, it is unknown whether its method of control can serve as a paradigm for other platforms.

Semi-autonomous control architectures were also considered to palliate the increased cognitive load experienced by users who teleoperate such platforms. The authors of [9] propose a framework for motion planning of active flippers, the direction of robot movement is assumed to be set by an operator. A robot named *KENAF* is used that has two laser range sensors placed on the left and right sides. Autonomous flipper motion control relies on range data and estimates every rotation angle of the flipper. Furthermore, authors perform stability analysis and integrate a rule based on the Normalized Energy Stability Margin (NESM) [15] within the system to stop motion and halt the task. Upon defining the robot negotiation limits, the authors consider that the system can negotiate any uneven terrain, yet this supposes the availability and use of exhaustive sensory data. At the same time, linear velocity and angular velocity control are not addressed. The system only alarms the user about tipping-over of the robot.

The authors of [75] develop a navigation system within the tracked robot *Absolem*. The authors accommodate 3D point cloud data and work on path planning and execution. They use sensor data to localized the robot using the iterative closest point algorithm. The

path is planned with *D* Lite* where the cost of every cell is adapted to ease traversability by the robot. For example, neighbors with lower slopes are preferred. The planned path is executed with two controllers. The first controller selects flipper configuration accordingly to the environment geometry around the robot. The second controller guides the robot to the farthest point in the path.

Suzuki et al. [14] propose a control framework. The robot receives the driving speed and movement direction commands from the user, and separate front and rear flippers are assumed to push against a traversed obstacle to improve traction, combined with an accident warning system based on the NESM. Still, the robot is not fully autonomous, and the overall approach is weakly generalizable.

The authors of [77] introduce a performant approach for effective path tracking. The proposed controller calculates commands to robot tracks and simultaneously adapts flipper configuration according to a provided feasible path. The authors rely on the precise kinematic analysis of the robot, that may not always be known or available to the expert.

Another motion control algorithm is presented in [79] for a tracked robot with two front flippers. This work also assumes accurate knowledge of the exact robot kinematics. Moreover, motion control is only applied to the traversal of a palette.

Another approach relying on the precise estimation of the staircase configuration was proposed in [78] where the robot could autonomously negotiate a staircase using proprioceptive state estimation based on sensory data of an IMU. Staircase ascent is thoroughly analyzed and divided into three stages: the pitch-up ascend, normal climbing up, and pitch-down ascend phases. Staircase descent is divided into the descend-ready, pitch-up descend, normal climbing-down, and pitch-down descend phases.

Most importantly, the authors describe failure modes. The first one is slippage which can be caused by insufficient traction between the tracks and the steps. Falling backward is one of the most common failures and occurs when the robot body rotates around the lowermost contact points. A erroneous robot heading can make the robot deviate from the goal, but it can hardly be considered as a failure. Still, the robot might tip over on zither side. Lastly, the robot might undergo excessive shocks, which exceed the acceptable level caused by falling and touching the surrounding environment. Flippers controllers are based on IMU data and designed with respect to the transition phase taking into account eventual failures. The approach showed prominent performance. However, the system could only negotiate a staircase known beforehand.

Another detailed analysis of kinematics was presented in [76] using a common tracked

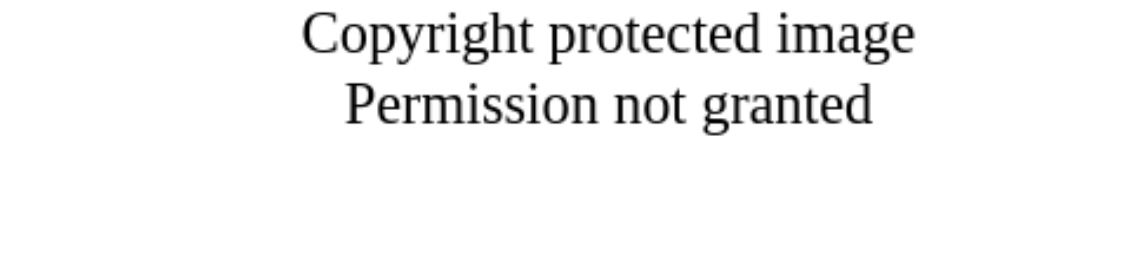


Figure 2.7: Stair climbing experiment by the *SSR robot* [76, Figure 16]

platform with two additional DOF due to a controllable arm (see Figure 2.7). The authors propose a new tip-over avoidance method based on adjustment of the system centroid position (SCP), which relies on the exact knowledge of robot geometry and mass distribution and 3-axial gyroscope data. The key idea of the developed strategy is to reconfigure the robot arm and place it in front of the robot when climbing a staircase or to move it back during the descent. The derived analytical solution indicates the high complexity of modeling the interaction between a multi-DOF system and the traversed surface. That system lacked congruent control of tracks, flippers and the arm, and needed full knowledge of geometry, kinematics and physical characteristics, which can be difficult to acquire.

Oehler et al. [80] advance ideas of [85] by proposing optimal path planning and path execution maximizing the contact points number and stability. While the reactive task corrects execution disturbances like slippage or unstable terrains, the robot obtains trajectories based on the environment map for all its joints through the pre-planning task, representing the focus of their work. The authors solve an optimization problem with constraints for the whole body motion separately for flippers. They achieve maximum traction through the maximization of the number of contact points and the arm. The latter is used to decrease the force angle stability margin [86] while avoiding self-collisions or with the environment. Their developments were tested both in simulation and in reality on two platforms *Hector Tracker* and *DRZ Telemax* (see Figure 2.8 (a, b)) and showed positive results. This work goes clearly beyond the previous standard control works regarding the number of DOFs, constraints, and portability to different platforms. However, it relies on a detailed map of the environment, while combining alternative constraints requires a new formalization of optimization objective functions and constraints, which may be tedious.

Overall, we highlight that the works of Zhang et al. [76] and Oehler et al. [80] are

particularly inspiring in the scope of this dissertation, as they correspond to the most rigorous approaches in the domain of learning-free robot control with safety constraints. Throughout the thesis, we wish to address their limitations through the RL setting application.

(a) (b)

Figure 2.8: Platforms [80, Figure 1]: (a) Hector Tracker, (b) DRZ Telemax

2.3.2 Learning-based approaches

In view of the different challenges encountered when modelling control of highly specialized vehicles, with difficult to model kinematics and dynamics, employing learning-based control can provide more flexibility. Works [67] present the diversity of employed learning techniques along with challenges. One of the main requirements is scalability to high-dimensional state and action spaces. While priors can bootstrap learning, more attention is required in the design of the search policy algorithm in order to ignore irrelevant policy properties. Generalization and robustness are particularly relevant for robots that are required to learn various tasks under varying conditions.

One of the early approaches that employed reinforcement learning for Adaptive Traversability learning is presented in [81]. The authors have developed an RL-based solution for rough terrain traversal. Their approach assumed a mapping between a state to a robot compliance mode represented by five different robot morphological configurations achieved through flipper control. The state contains Haar-like local neighborhood terrain features computed with the Digital Elevation Map, robot speed, pose, flipper angles, compliance thresholds, and actual flipper mode. The fitted Q-iteration algorithm [87] is applied to learn the Q-function. The expert's manual control is required to produce the first training samples after which the robot operates independently for data acquisition. Despite the prominent results shown, expert manual assistance is required in the learning process

while actions are limited by pre-defined morphological configurations while contemporary RL methods [20, 21] allow the robot to discover such configurations alone.

Another recent work [82] proposes a RL-based control for a tracked robot (see Figure 2.5 (a)). The authors improve dueling double deep Q-network (D3QN) by adding a normalization layer and noise injection in the network parameters, which correspondingly accelerate training and improves the exploitation-exploration trade-off towards more exploration nature. D3QN is a model-free, value-based method that tackles double-deep Q-network (DDQN) and deep Q-network (DQN) issues. The later models overestimate each action value in every state, but D3QN uses two networks instead: one calculates the state value, and another is used for the advantage action calculation. Learning was performed in simulation and the obtained controller was tested in reality showing promising results. Still, this approach is only employed for navigation on 2D surfaces.

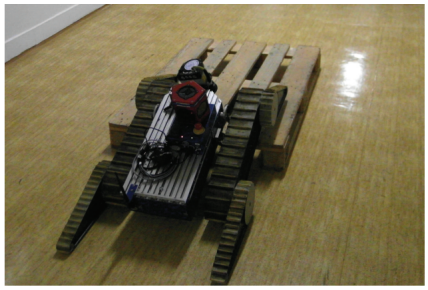


Figure 2.9: Absolem robot traversing a EUR palette [17] ©2016 IEEE

Another approach is proposed by Pecka et al. [17] for the platform in Figure 2.5 (c). The authors' contribution amounts to the development of RL algorithms for Search and Rescue scenarios. They implement constraints in the contextual relative entropy policy search (Contextual REPS) algorithm [18]. That extension showed that a small number of iterations is sufficient to learn flipper control for the traversal of an unknown obstacle (see Figure 2.9) where a physics-based simulator computes the safety of a rollout. The simulator represents software simulating the main interactions and influence of the real-world system and determining the safety of the rollout as 1 if it is safe or 0 otherwise. However, the authors did not investigate the impact of context represented by variables like the height of an obstacle that does not change during the task execution but might change from task to task. Furthermore, the necessity to perform fine-tuning in reality increases the training burden on the user, eventually making this approach difficult to transfer to reality. We consider this work as one of the most representative works in the field of

learning-based control learning of an actively articulated tracked robot. It motivates us to tackle its limitations, such as the necessity to perform real-world verifications and the absence of context addressing.

2.3.3 Recapitulation

Most previous works on the development of active 3D obstacle negotiation [10, 10, 79, 88, 89] by tracked robotic vehicles are customized to specific platforms, accurately known robot kinematics, robot characteristics, and staircase parameters [74]. The domain of tracked robot control has started to attract interest in search and rescue scenarios and more recently in navigation in spaces populated humans. The fact that a robot may carry payloads of different weight, from humans (see Figure2.5 (f)) to everyday objects further has a direct effect on the variation of the hardware design.

As a result, changes of the platform (recall Figure 2.5) or the task may render the adaptation of such solutions cumbersome, if not impossible. They complicate portability to platforms with different or unknown kinematics or poor environment observability, weakly considering the dynamics of physical interaction. Additionally, the development of a robot controller necessitates domain expertise or experience, which may not be easy or even feasible to acquire, e.g., for search and rescue environments [77].

On the other hand, learning-based, and in particular RL-based, control of flippers [81, 82] allows to make less restrictive hypotheses concerning the structure of the environment, robot kinematics, and dynamics. It also allows to more easily endow a behavior with additional properties. Using machine learning then raises the question of the type of learning to be sought, such as learning from demonstration (LfD) [90], behavior cloning [91], RL or their combination [92]. For the staircase negotiation task that we consider, it is impractical, if not impossible, to obtain sufficiently rich demonstrations because controlling multiple degrees of freedom in parallel is not intuitive and incurs a high cognitive load.

To increase autonomy in the process of learning robot mobility, we therefore investigate the acquisition of such skills via RL. Its algorithms in particular [28, 67], that have drastically developed during the last decade may even outperform human experts [26]. RL can favor the emergence of the desired control properties through the appropriate design of reward/penalization functions and state/action domains [93].

2.4 Safety assessment techniques

One of the key problems treated in this thesis is the incorporation of safety properties in the developed robot behavior. Independently of the underlying locomotion type, a robot is expected to operate in a safe and reliable manner. This is even more pertinent to actively articulated robots that are more subjected to tip-overs during staircase traversals due to active articulation of flippers and, most importantly, of an arm that shifts the robot center of gravity (COG) higher. Despite a plurality of stability measures that were initially proposed for walking robots [94], there is no single optimal criterion for all possible applications. This section presents the most common techniques for the estimation of robot stability.

Stability criteria can be static or dynamic. The latter group of criteria is popular in legged robotics due to dynamic effects that emerge during walking. A popular stability assessment method widely applied on legged robots is the Zero Moment Point (ZMP) [95]. It represents a point on the supporting polygon, formed by connecting footprints, where the moment due to terrain-reaction forces and moments becomes zero. The Force-Angle Stability Margin (FASM) [86] calculates the angle between the normal vector to the rotation axis from the COG and the resulting forces acting from the COG. FASM considers if angles are positive, then the system is dynamically stable. However, ground robots do not develop the same complex dynamics effects. In our work, we assume that the robots operates at low velocity or with low-velocity changes. Thus, we opt the static stability analysis as it was performed in [9, 14] and in the PhD thesis of Brunner [96].

One of the first static stability criteria was proposed in [97] and was named the Center of Gravity Projection Method. The robot operating at constant speed and direction is considered stable if its COG horizontal projection resides within the supporting polygon. The Static Stability Margin (SSM) [98] further elaborates the COG Projection Method and consists of the evaluation of the distance between the projection of the COG onto the supporting polygon (SP) and its lowest border. The system loses stability if this distance approaches zero. The more convenient Energy Stability Margin (ESM) [99] evaluates the potential energy necessary to tip over the robot. A popular extension of ESM named the Normalized Energy Stability Margin (NESM) [15] normalizes this energy to the robot's weight, and, as authors show, it is an adequate static stability margin.

We distinguish two types of staircase traversals - ascent and descent - and quantify safety differently in each case. We choose to associate the classical robot stability [33] as

a measure of its safety in ascent staircase negotiations and base our work on the NESM as one of the most developed criteria, which has received considerable popularity across different studies on tracked robots [9, 14, 34]. Additionally, the SSM attracts our attention as a complementary mean for stability estimation. Its consideration along with the NESM for designing a reward function in RL will be described later in chapter 4.1

We find that the robot can naturally descend the staircase through an application of constant speed without any active reconfiguration, which may not lead to any stability violation. Still, the robot could be damaged due to the drop impact or shock due to high linear or pitch angular velocity when the robot passes the staircase edges. To address this problem, we draw inspiration from the idea of bumpiness detection from [23] where the authors detect a bumpiness event whenever the angular velocity magnitudes measured by the IMU exceed a certain threshold. Thus, the angular velocity magnitudes could serve us as a mean for quantifying the drop impact that the robot is subjected to.

2.5 Robot and environment simulations

The simulation of the environment and the robot model are essential in the development of robot skills through RL. Despite a multitude of sophisticated environments for different types of robots, articulated tracked robots are weakly represented, especially within the simulator Gazebo [100], which is among the most widely used tools for robot application development. V-REP [101] is another common simulation tool commonly used in robotics that provides a user-friendly world and robot model configuration tools, exhibiting comparable computational cost and simulation accuracy [102]. Still, it is far behind Gazebo in terms of integration with ROS which is state-of-the-art for modern robotics development.

In relation to indoor navigation and control learning, the *AI2-THOR* framework [103] provides near-realistic 3D indoor scenes for AI research where agents can navigate and interact with the environment. Authors consider that it can be helpful not only for reinforcement learning but also for learning by interaction, imitation learning, etc. Still, this framework supports only discrete actions and does not address robot dynamics where agents operate in continuous action spaces. The work [104] presents *Interactive Gibson Benchmark* which includes a renderer and over one hundred 3D reconstructed environments. To the best of our knowledge, it is one of the best current environments able to generate highly realistic images and simulate physical interaction. In its most recent version, domain randomization is integrated for objects and materials [104]. However, tracked articulated robots are still not included into this framework.

Various research groups study navigation learning in indoor environments by using popular tools such as Gazebo and ROS. For example, authors of [105] train a robot to navigate in only two instances of a 2D environment. Tai et al. [22] also limit their environment variation by two pre-constructed settings. The robot can cross one real-world maze to reach a goal, but its scalability in a general setting is unknown. Concerning learning environments, authors of [106] offer a total set of 6 settings and three robots to use with RL algorithms. Yet, multi-floor domains are not considered neither is domain randomization for ground obstacles, which could serve as a data augmentation technique.

Indoor environments are often populated with complex but traversable terrains such as stairs, where robot dynamics become harder to model and necessitate additional attention to safety. Furthermore, the environment should vary sufficiently to cover the multitude of situations that can be encountered in reality [45]. Unfortunately, we can hardly refer to any available simulation environment that accounts for this demand for articulated

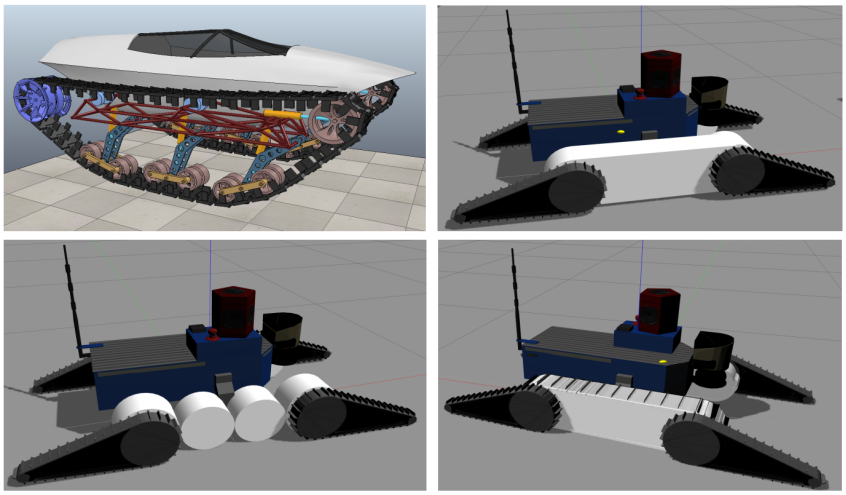


Figure 2.10: Models of tracks [35] ©2017 IEEE. Top left: Chain-like deformable tracks. Top right: CSM-based tracks. Bottom left: Track approximation with 4 wheels. Bottom right: Track made of 2 cm plates with grousers

tracked robots.

As it was stated, Gazebo is widely used in robotics applications thanks to its ability to accurately and precisely simulate robots in complex indoor and outdoor environments and bind with ROS. This simulator supports cylinder, sphere and box primitives out-of-the-box[1], or alternatively, one can use custom 3D meshes. Those components can be coupled together through the "joint" element which can be of different types. Commonly, fixed, revolute and continuous joints can be used. The different types of joints that are supported can be used to model the actuation of an arm or of the legs of an articulated robot.

Tracks modelling To model a robot track, one might use cylinder and box components with fixed, revolute or continuous hinge joints to create a precise chain-like track model (Figure 2.10, top left). However, this solution is computationally expensive and unstable due to multiple constrained dynamic elements [107]. Furthermore, authors of [108] reveal problems associated with the development of such models within Gazebo due to model instability.

Pecka et al. [35] perform the most detailed analysis of robot tracks within Gazebo and develops the Contact Surface Motion (CSM) model. In that work, it was shown that tracks made of wheels (see Figure 2.10, bottom left) are not plausible on rough terrains because

[1] `https://wiki.ros.org/urdf/XML`

friction forces have unrealistic directions. Still, it is computationally light and simple to produce. The model of tracks based on plates (see Figure 2.10, bottom right) can realistically simulate deformable tracks, grouser interaction with rough and flat surfaces. However, it slows down the simulation to the point that it can take 1 or 2 orders of magnitude more time than the CSM or wheeled-based model.

The no-friction model is also presented. Its principal idea is to set the track friction to zero and apply the constant force to the gravity center to move the robot. It is the most computationally efficient approach. However, it does not adequately address the rough terrain traversal due to the friction absence necessary to keep the robot static.

Finally, the authors also introduce the CSM model where they rework the dynamic simulation formulation, and this model finds the force to move the robot at a certain velocity. It does not simulate grousers or deformable tracks. Still, it is the best trade-off between track simulation adequacy and the computational overhead of simulation and plausible on flat and rough terrains.

Chapter 3

Prototyping Reinforcement Learning based Staircase Ascent

In this chapter we present a learning-based control approach for the problem of staircase ascent, applied to an articulated, wheeled-based robot model within a simplified simulated environment, employing a RL-based pipeline. Our focus here is on the constraints of the control policy that we wish to satisfy, such as safety or traction. To demonstrate the generalization ability of the framework, the policy is learnt and tested for a variety of staircases of different step heights and angles, while evaluating its capacity to cope with different levels of noise in its sensory data. This initial work on simulation can serve as paradigm for more elaborate development in other platforms but it can also provide insights when modelling the negotiation of staircases shapes of diverse riser and tread size.

The remaining of the chapter is organized as follows. In section 3.1 the proposed baseline reinforcement learning framework is presented, we formulate the notion of safe stair ascent and study a first reward function candidate. Finally, in section 3.2 the experimental setup is presented together with the obtained results.

3.1 Reinforcement Learning Framework

To address the problem in consideration, we develop a RL framework following [27]. RL resides in generating data through agent interaction with an environment where every time step t, an agent being in a state $\boldsymbol{s}_t \in S$ selects an action $\boldsymbol{a}_t \in A$ with respect to its policy $\pi_{\boldsymbol{\theta}}$ where $\boldsymbol{\theta}$ is the vector of policy parameters, transits to a new state $\boldsymbol{s}_{t+1}$ receiving a scalar reward r_t. The multitude of transitions from the initial state $\boldsymbol{s}_0$ to the termination state $\boldsymbol{s}_T$ is called a trajectory $\tau = (\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1, ..., \boldsymbol{s}_{T-1}, \boldsymbol{a}_{T-1}, \boldsymbol{s}_T)$ where T is the number of time steps of an episode. A policy $\pi_{\boldsymbol{\theta}}$ is a function parameterized with $\boldsymbol{\theta}$, that sets a rule for preferring a particular action among multiple alternatives. It can be either deterministic $\boldsymbol{a_t} = f(\boldsymbol{s_t})$ where an action is taken at a specific state or stochastic via $\pi_{\boldsymbol{\theta}}(\boldsymbol{a_t}|\boldsymbol{s_t})$ which specifies a conditional probability distribution and provides the action probabilities in a defined state from which we sample the most probable action. During learning, policy parameters $\boldsymbol{\theta}$ are trained to maximize the expected return:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} E_{\tau \sim \pi_{\boldsymbol{\theta}}}[R(\tau)] \tag{3.1}$$

where $R(\tau) = \sum_{t=0}^{T} \gamma^t r_t$ is the return over a trajectory τ, T is the episode length, γ is a discount factor, which can used to decrease the influence of future rewards in the learning process or set to 1 if we wish that the agent learns to maximize the return taking equally into account all rewards. A specific form given to r_t then becomes an object of research as will be discussed in the section 3.1.3.

Going beyond a basic treatment of the problem of RL as described above, it is crucial to account for differences occurring between training and testing time. This is particularly relevant in indoor environments that can vary significantly and unless this is accounted for during training, the gap between simulation and reality will reduce the performance of the policy during testing, whether it is deployed in simulation or in the real-world.

As a means for covering various possible environment configurations, we can endow RL with DR. To do so, we parameterize the training domain by $e_{\boldsymbol{\xi}}$ where each configuration $\boldsymbol{\xi} \in \Xi \subset \mathbb{R}^N$. If a policy is learnt on a multitude of parameterized environments chosen randomly, this favors generalization and we thus seek to maximize the expected return over a distribution of configurations. The optimal policy parameters are therefore obtained

by:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} E_{\boldsymbol{\xi}\sim\Xi}[E_{\tau\sim\pi_{\boldsymbol{\theta}},e_{\boldsymbol{\xi}}}[R(\tau)]] \tag{3.2}$$

In previous years, PG algorithms attracted significant attention because of their direct policy optimization in contrast to value-based methods where the policy is obtained via its relation with the learnt approximator. The main idea behind these algorithms is the gradient ascent over the policy approximator parameters $\boldsymbol{\theta}$ using the policy gradient $\nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}})$,

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha \nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}_k}) \tag{3.3}$$

where α is the learning rate and $J(\pi_{\boldsymbol{\theta}})$ is the expected return that could be written as follows considering domain randomization:

$$J(\pi_{\boldsymbol{\theta}}) = E_{\boldsymbol{\xi}\sim\Xi}[E_{\tau\sim\pi_{\boldsymbol{\theta}},e_{\boldsymbol{\xi}}}[R(\tau)]] \tag{3.4}$$

3.1.1 Flipper Control Problem Formalisation

Our next task concerns the expression of staircase ascent negotiation for an articulated tracked robot, on the basis of the RL framework presented above. In doing so, we observe that robot control could be divided into reactive main track and flipper control [14, 17] where reactivity implies that the controller directly maps sensor input to desired actions. In this chapter, we will assume an independently developed main track controller (i.e. a global path planner) that moves the robot forward at constant speed and focus only on the development of flipper control whose objective is to perform a mapping between the pose of a robot on a staircase and the appropriate flipper commands.

We further assume that the robot is initially orientated so that if faces the staircase. Naturally, such a set-up leads us to accept that a pair of front (or rear) flippers will have a single degree of freedom. In other words, that the flippers at each side should perform the same action, since we deal with straight staircases. We consider the robot to select two rotational displacements $\psi_a^{front}, \psi_a^{rear}$ where angle limits $[\psi_a^{min}, \psi_a^{max}]$ are defined by the user forming an action vector:

$$\boldsymbol{a} = (\psi_a^{front}, \psi_a^{rear}) \in [\psi_a^{min}, \psi_a^{max}]^2 \tag{3.5}$$

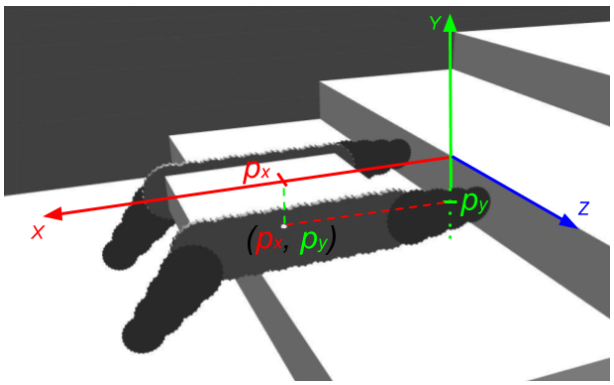


Figure 3.1: Relative distance of the robot centroid to the next step edge/nosing

for the front and rear flippers while constant effort is applied to each track moving it forward. The state vector:

$$\boldsymbol{s} = (p_x, p_y, \psi_s^{front}, \psi_s^{rear}) \in S \tag{3.6}$$

consists of the distances p_x, p_y of the robot centroid to the next step nosing along the axes X and Y (see Figure 3.1) where the origin corresponds to the step center. It further includes the flipper angles ψ_s^{front} and ψ_s^{rear} in each flipper frame which belong to $[\psi_s^{min}, \psi_s^{max}]$ defined by the robot design where $\psi_s^{min} < \psi_a^{min} < \psi_a^{max} < \psi_s^{max}$.

In accordance with the reactive control architecture 2.1, the task to be learned treats the first half of the complete staircase negotiation problem as stated by contribution #1, section 1.1, namely, flipper reconfiguration accomplishing the primary goal of the staircase ascent and the secondary goal of policy constraints satisfaction such as traction and stability.

3.1.2 Policy and its training

We wish to map the robot state represented by a vector $\boldsymbol{s}$ to a vector of actions $\boldsymbol{a}$ under the policy π, which requires a good policy function approximation. Following up on the discussion at section 2.2, artificial neural networks have demonstrated remarkable results in machine learning as well as in reinforcement learning in particular, and could be well suited for nonlinear function approximation [25, 27]. Having small input and output vectors of estimated robot state and corresponding actions, we approximate the control policy with a multi-layer perceptron, in contrast to [109] which employs a complex CNN

because of necessity to treat complex image input data. The first layer is composed of 4 neurons on which the state vector $\boldsymbol{s}$ is received every time step. In the sequel, the input is forwarded through 2 dense layers, where each of them has 64 neurons with *tanh* activation function, to the output layer which forms the action vector.

Among various possible PG algorithms, we have selected PPO [21, 110] as a state-of-the-art algorithm that exhibits good trade-off between ease of tuning, sample complexity, ease of implementation and good performance [111]. This algorithm updates the policy with stochastic gradient ascent while keeping a new policy close to the previous one that could be completed either by penalisation of Kullback–Leibler divergence or specialized clipping in the objective function. Besides the control policy approximation, PPO resides on a critic network for value function approximation, which has the same inner structure as the policy network.

3.1.3 Reward function design

Default reward function for primary goal

The first and most simple examined reward function remunerates the robot according to the travelled distance on the staircase. It receives the positive reward r_t^p:

$$r_t^p = \Delta x_t / \|D_{max}\| \tag{3.7}$$

where $\|D_{max}\| = x_0$, which is the distance from the robot centroid center to the goal at the episode first time step $t = 0$ (see Figure 3.2 (a)), and Δx_t is the travelled distance to the goal at time step t as represented by the next formula:

$$\Delta x_t = \begin{cases} |x_t - min(x_0, \ldots, x_{t-1})|, & \text{if } x_t < min(x_0, \ldots, x_{t-1}) \\ 0, & \text{otherwise} \end{cases} \tag{3.8}$$

where x_t and x_{t-1} are distances from the robot to the goal at time steps t and $t-1$. This implies that we do not assign the positive reward twice if the robot slips back and re-travels an already seen position. The robot receives the baseline reward when it starts to touch the staircase, which happens when it comes closer than $D_2 = d + 0.5l$ on the ground to the first stair step (see Figure 3.2), while the episode ends when the robot reaches a distance of D_4 away from the upper nosing on the stair landing.

That episode return hence ranges from 0 to 1 depending on the effectiveness of the

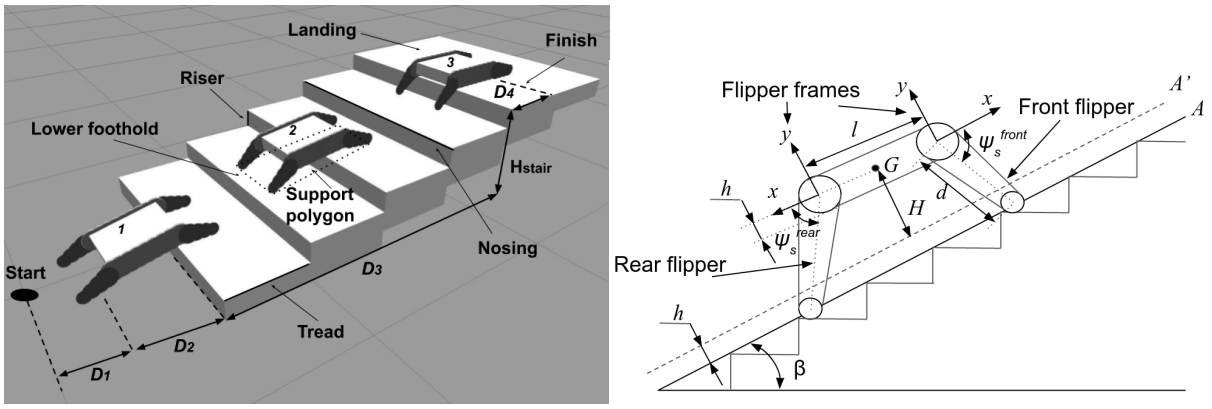
policy. If the robot gets closer than D_2 without appropriate flipper actions, it could be stuck (episode return 0). The episode is considered terminated if the distance D_4 is crossed on the upper landing (episode return 1). D_4 is chosen to be equal to D_2 in order to guarantee that the robot loses the contact with the stair at the end of the episode and the ascent is successfully accomplished.

NESM constraint reward function for secondary goal

Following the idea of applying negative rewards to unsafe states, we incorporate a negative penalty term by introducing a stability criterion into the reward function. In detail, we employ the NESM criterion that is based on the fact that the robot rotates around a support line when tumbling/tipping over. We deduce the margin E as the difference between the maximum centroid height and its current height [14]:

$$E = H_{max} - H \tag{3.9}$$

Based on this measure, which tends to be low the less stable the robot is, we associate robot instability I as equal to the current height H over the hyperplane A' (see Figure 3.2 (b)). Thus, when $I = 0$ the robot is considered as maximally safe, whereas when $I = H_{max}/1m$ the robot is maximally unsafe. Division by 1 m ensures independence from the chosen units of measurement. The value H_{max} is the distance of the robot centroid to the hyperplane A' when the robot is pivoting over the lower support polygon foothold. We consider the negative reward based on NESM to be $r_t^n = -I$. Thus, the reward per



(a) Stair ascent task illustration

(b) Robot and environment notations

Figure 3.2: Modelling of a staircase and the robot used for the RL problem description

time step is:

$$r_t = r_t^p - I \tag{3.10}$$

Every time step the robot can thus obtain a positive reward along with a negative one only if its gravity center G moves away from the hyperplane A', whenever the robot centroid projection is located within the first and last nosings on the hyperplane A'. We assume that flipper rotations do not shift the center of gravity position G, because the total mass of flippers is negligible to the base mass.

Projection constraint reward function for secondary goal

We refer to projection as a measure of robot tracks contact with the traversed surface and as a measure of stability. The idea of maximizing the contact of the tracked robot parts to the traversed surface is commonly employed as a measure of traversability. For example, authors of [9] suggested to set the orientation of flippers tangential with respect to the traversed surface during obstacle negotiation in order to maximize traction while in [14] flippers are even pushed against the traversed surface in order to enforce contact.

It is reasonable to assume that a minimal contact surface area should be maintained, between the flippers and the obstacle being negotiated, otherwise the exerted control actions might not have the desired effect. On the other hand, depending on the geometry of the robot and the staircase, maximizing surface contact might drive the robot to poses from which it would be impossible to successfully traverse the staircase.

Knowing the actual size proportions between a conventional staircase and commercial tracked robotic vehicles, maximizing both traction and stability while traversing a staircase indeed amounts to moving at an angle equal to the inclination of the staircase. Therefore, we assume that the maximum stabiltiy and traction with the surface are achieved when the robot maximizes its projection on the staircase hypersurface. The maximum surface area P_{max} is then the projected surface area of the robot on the hyperplane A when the robot is located on the staircase with a flat configuration, namely, with flat flippers.

We let P be the surface area of the robot body when orthogonally projected on the hyperplane A. In order to favor a control policy where the robot has decreased projection, a negative reward is considered at each time step $r_t^n = -(P_{max} - P)/(P_{max})$. Thus, the overall reward per time step is:

$$r_t = r_t^p - (P_{max} - P)/P_{max}. \tag{3.11}$$

3.2 Experiments

In this section, we present the experiments that we conducted to evaluate the RL control pipeline as was described earlier in Section 3.1, quantitatively as well as qualitatively, by varying the form of staircases and in the presence of different levels of simulated noise.

3.2.1 Experimentation description

Policy training protocol

We simulate the control-learning pipeline of the robot in the Gazebo, physics-based simulation environment (`gazebosim.org`). The tracked robot is endowed with front and rear flippers and its task is to learn how to mount a five-step staircase whose tread and riser are randomly chosen. Among various alternatives for simulating mobile robot tracks, we employ the common approximation of representing tracks by an ensemble of overlapping, non self-colliding, tracked wheels (cf. [35]).

Experiments were performed repeatedly for a given randomly generated configuration of a staircase whose geometry was representative of real-world scales. At each time step the robot observes the state vector $\boldsymbol{s}$ which is comprised of relative distances to the next step nosing p_x and p_y. The former varies from the minimum distance $p_x^{min} = 0$ to the horizon of step observation $p_x^{max} = 1\ m$, which is manually set based on the idea that the robot is moving on the flat surface if there are no close step observations. Relative distance along Y axis has as its maximum value the biggest height p_y^{max} which depends on the obstacle negotiation capability $H_{critical}$ and $\beta_{critical}$ of a particular robot, namely, the maximum riser height and staircase inclination that the robot is able to ascend.

In every episode, irrespective of the results of the previous episode, a new domain $e_{\boldsymbol{\xi}}$ of 5 steps staircase is generated in the following way. First, a random riser size $H_{rise} \in [H_{rise}^{min}, H_{rise}^{max}]$ is sampled where $H_{rise}^{min} \geq 0$ and $H_{rise}^{max} \leq H_{critical}$. Second, the stair angle $\beta \in [\beta_{min}, \beta_{max}]$, where $\beta_{min} \geq 0$ and $\beta_{max} \leq \beta_{critical}$, is randomly sampled. Last, the tread size is calculated as $D_{tread} = H_{rise}/\tan(\beta)$. Thus, the domain configuration $\boldsymbol{\xi}$ is represented by a vector $(H_{rise}, \beta, D_{tread})$ (see Figure 3.3).

The robot starts at a front-parallel position with respect to the staircase at a distance of $D_1 + D_2$ from the lowermost step. Once the robot goes through D_1, upon application of an action the robot obtains a reward r that accounts for the primary reward that is proportional to the traveled distance and potentially for the secondary goal of safety, quantified by the chosen penalty function (see section 3.1.3). An ascent traversal is deemed

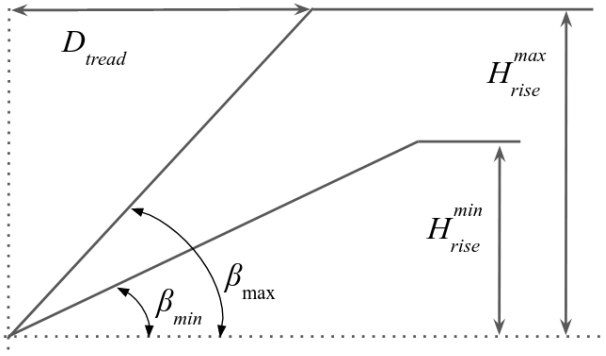


Figure 3.3: Domain configuration for each step of a staircase

as successful if the robot succeeds in traversing over D_4 on the upper landing from the last nosing. The episode return of one trajectory represents the proportion of travelled distance and could vary from 0 to 1 if we do not model safety. This sequence is repeatedly executed until the end of an episode that is set to occur after a maximum number of 100 time steps. If the standard deviation of 25 most recent robot position estimations along the X-axis drops under 0.01 m, the episode stops assuming that the robot has been "stuck". To set time boundaries of experiments, we allocate 10000 time steps in total for all experiments that translates to 250 episodes on average, as a consequence of shortened episodes after fast traversals or accident induced terminations. Remaining parameters were chosen in accordance with [21].

Policy testing

While a policy is being learnt it is in parallel evaluated on stair configurations that have not necessarily been encountered. In particular, policy testing starts to be performed after the first policy update and subsequently at a fixed frequency of 8 policy updates on three stair configurations of different difficulty, whose parameters are presented in Table 3.1. We have empirically chosen that testing frequency as an acceptable trade-off between policy learning speed and testing regularity. During this assessment, we plot the evolution of the episode return (see Figure 3.4 (b)), stability (see Figure 3.5 (b)) and projection (see Figure 3.6 (b)) averaged over 3 cases which correspond to *small*, *medium* and *big* staircases also called configurations.

Table 3.1: Staircase configurations used in evaluation

Type	***Step height***	***Stair angle***
Small	H_{rise}^{min}	β_{min}
Medium	$0.5(H_{rise}^{min} + H_{rise}^{max})$	$0.5(\beta_{min} + \beta_{max})$
Big	H_{rise}^{max}	β_{max}

3.2.2 Reward function evaluation

Evaluation metrics

We assess the effectiveness of the learning process on the basis of the average episode return evolution during learning. This evaluation enables us to estimate whether learning has converged, after which further learning does not improve an agent's performance. Typically, at the beginning of learning, the average episode return shows high variance and quick increase of its value, which indicates that learning has been initiated. Towards the end, the mean value reaches a plateau while variance decreases, that indicates learning convergence.

For every type of reward function, we performed a total 3 training trials used for computing the average performance. We adopt this experimentation setting throughout the different experiments presented in the dissertation. The number of trials naturally raises the question about confidence of learning and our reasoning about successful learning convergence. The Student's t-distribution [112] allows to calculate the margin of error, on which we rely, for small samples. The margin of error (ME) of a sample can be calculated as follows [113, ch. 9]:

$$c = t\frac{\sigma}{\sqrt{n}} \tag{3.12}$$

where $\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$ is the sample standard deviation, n is the sample size, t is the confidence level for the t-distribution and $\bar{x}$ is the sample mean of all samples x_i. For example, if we wish to estimate the reward confidence interval within an episode, the sample size is 3 that leads to 2 degrees of freedom and, if we want to obtain 95% confidence, the confidence level is $t = 2.92$.

To evaluate correctness of learning, we can rely on the evolution of the ME of the episode return. The policy parameters $\boldsymbol{\theta}$ are random in the beginning, which can lead to

different behaviours in the simulation leading to varying episode returns and high ME. During training, they are optimized to an optimum. Therefore, if all policies reach the same optimum parameters $\boldsymbol{\theta}^*$ the robot has to exhibit similar behaviors and episode returns should not be very different which translates to a smaller ME than in the beginning. Thus convergence of learning can be deemed to be achieved when the ME becomes small but also stable across the most recent history of episodes, as further policy updates do not change the policy.

Reward evaluation

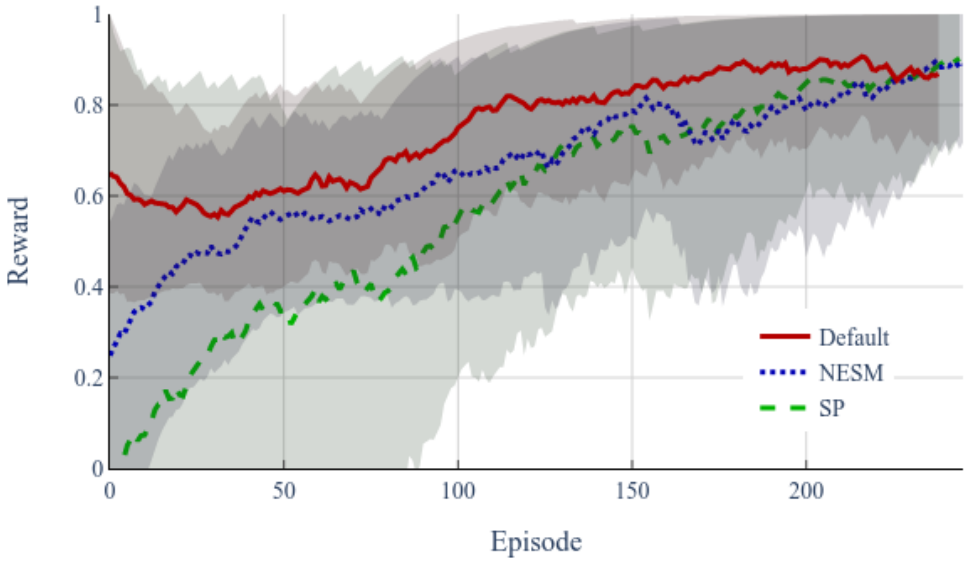
Along with the quantitative results presented later in this section, we provide a video of qualitative results of the learnt policies for staircases of varying difficulty, accessible in (`https://partage.imt.fr/index.php/s/pBSzKaoeDnqFSyA/download`).

During experiments, episode return (see Figure 3.4), NESM-based instability measure I and estimated robot projection ratio P/P_{max} were measured (see Figures 3.5 and 3.6 respectively).

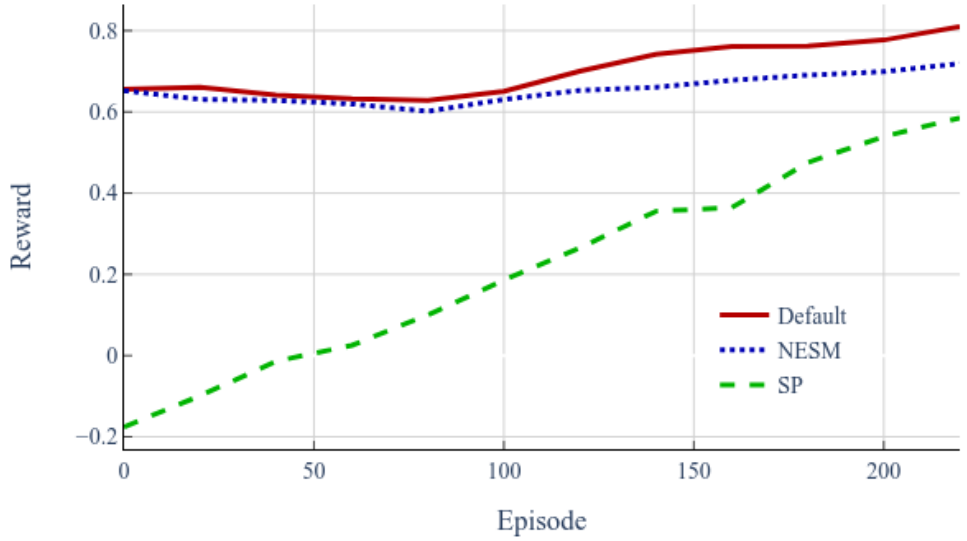
Figure 3.4 (a) presents the smoothed return per learning episode and similarly for testing in 3.4 (b), using min-max bands. Hereinafter, the mentioned smoothing is accomplished using an exponential moving average (EMA) [114] with a coefficient 0.95 and 0.8 for learning and testing curves respectively.

With respect to learning performance (Figure 3.4 (a)), the NESM-based and projection-based policies converge as early as the default policy (at around 175 episodes) and their mean episode returns vary from 0.5 at the beginning of learning, converging to 0.9 at the end. We can observe the presence of non-zero episode return in the beginning that could be provoked by random actions that the robot makes in the beginning of training which allows it to reach half of its chassis length on the staircase. On the other hand, the optimal reward values per episode are slightly lower for policies that incorporate safety, because of unavoidable necessity to perform less safe actions in order to mount the staircase. We can further observe that max bands of learning curves reach the maximum return after 100 episodes for all types of reward function. This means that although policies are able to perform appropriate flipper control min bands are constantly increasing due to behaviour improvement on unseen staircases. By 200 episodes, the mean curves converge suggesting that learning is complete.

With respect to testing performance (Figure 3.4 (b)) for stair configurations of varying difficulty (cf. Table 3.1), we observe that the mean reward tends to become higher during



(a) Learning



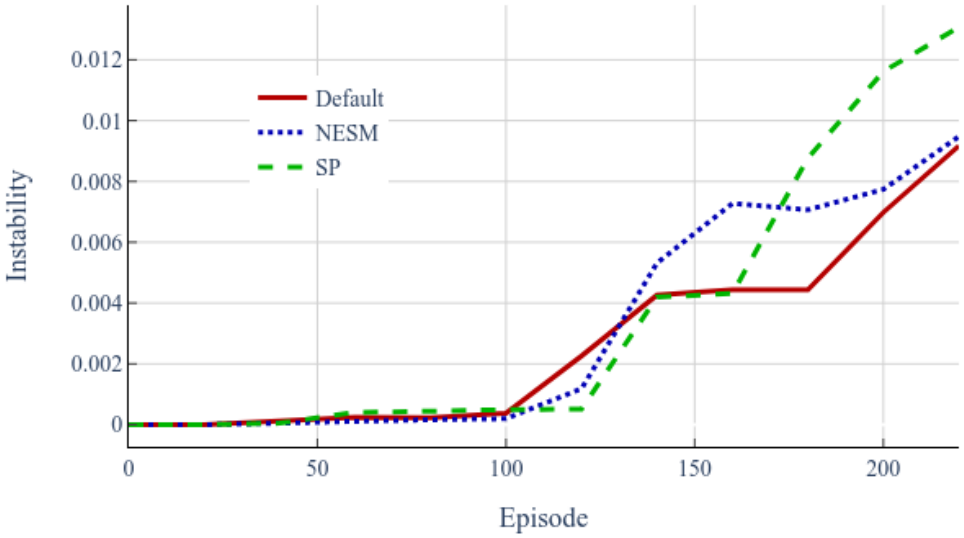
(b) Testing

Figure 3.4: Smoothed episode return $R(\tau)$ during (a) learning and (b) testing

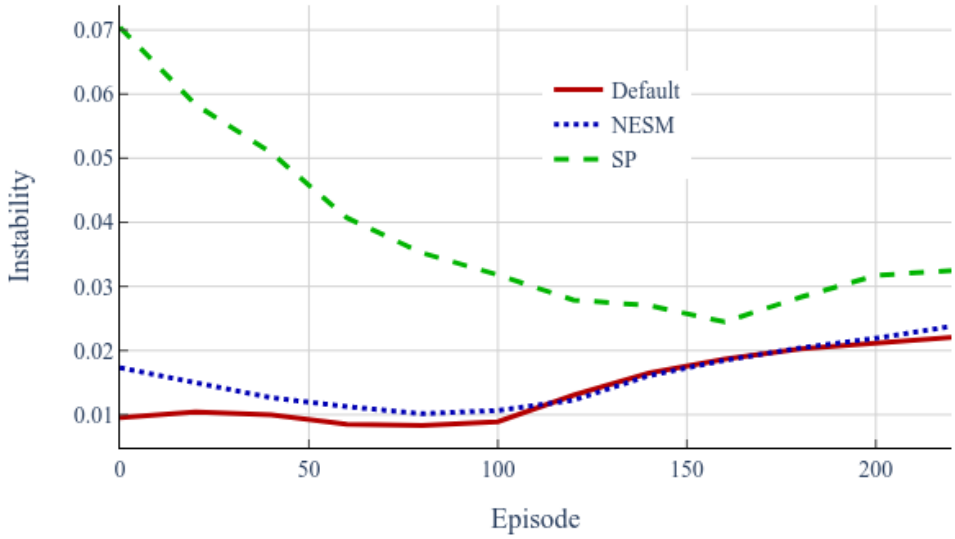
learning for the various staircases being encountered for all three reward function designs.

Specific behavior development

We now shift our attention from the evolution of the total reward, on how different policies affect a given measure for achieving the secondary goal of safety. In the first case where safety is quantified by instability I (recall eq. (3.10), based on the results shown in Figure



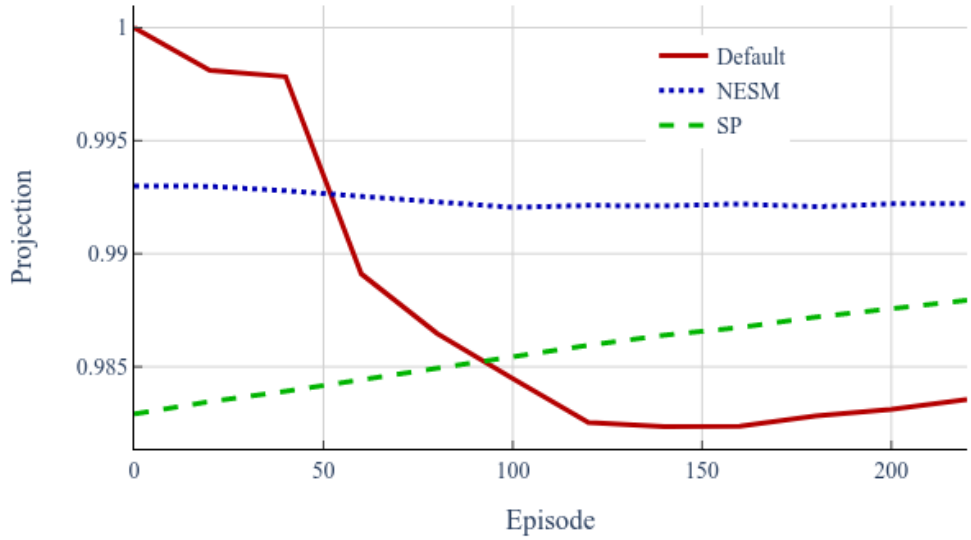
(a) Learning



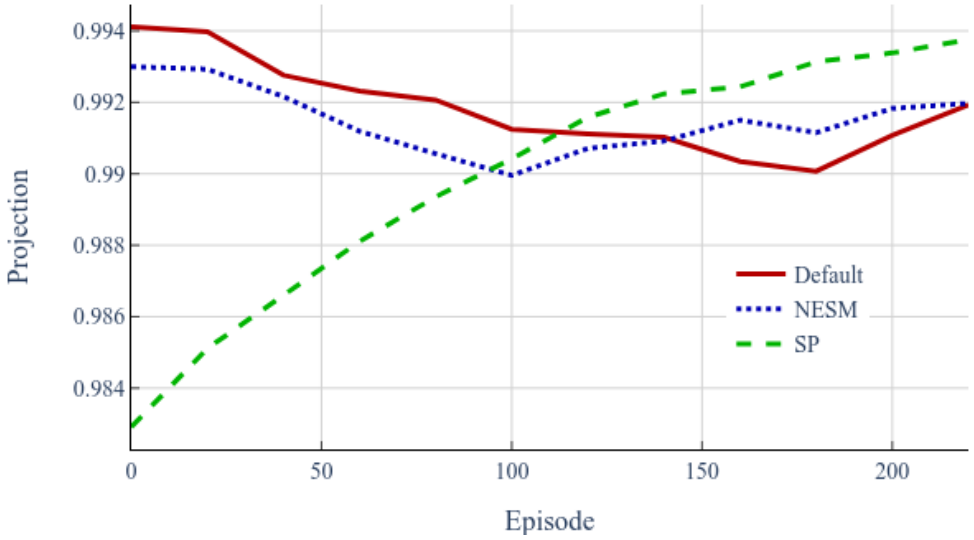
(b) Testing

Figure 3.5: Smoothed instability I during learning and testing for all three alternative reward designs

3.5 (a), we can deduce that the robot is maximally safe in the beginning of learning, i.e. before 100 episodes, but this barely allows the robot to traverse the staircase and very often leads to episode failures. Instability for all 3 policy types starts to increase after 100)episodes which correlates to learning curves whose max bands reach the maximum episode return 1. **We can therefore deduce that a safety drop is unavoidable in order to accomplish the main task of stair ascent**. By the end of the training,



(a) Learning



(b) Testing

Figure 3.6: Smoothed projection ratio P/P_{max} during learning and tests

the lowest instability corresponds to NESM-based and default (unsafe) policies, while the instability of the projection-based policy tends to be higher. Test instability curves (see Figure 3.5 (b)) demonstrate the same behaviour, the projection-based policy is less safe by the end of training, while the NESM-based and unsafe policies have the same instability values. One notable difference on test learning curves (Figure 3.5 (b)) in the beginning of learning, namely, is that the projection-based policy is considerably more unstable compared to the two other policy types.

In the second case where traction and stability are quantified by the ratio of the surface area projection of the robot, based on the results shown in Figure 3.6 (a), the projection-based policy tends to better ensure a flat robot configuration compared to the other policies. The projection-based policy tends to increase its projection, while the NESM-based policy does not influence it. In accordance with the tests in Figure 3.6 (b), the projection-based policy tends to increase robot alignment with the staircase. The NESM-based policy does not have an impact on the projection. The default policy even decreases projection.

Interestingly, the safety-based NESM and default policy seem to exhibit the same behaviour (see Figures 3.5 (b), 3.6 (b)) in terms of instability and traction. In our understanding, the reason for such behavior is that the robot tends to be flat to traverse a staircase as expected in the NESM-based policy, but this configuration turns out to be necessary more generally as a default behavior within the default policy. We believe this to happen due to inadequate interaction of tracks and flippers with the staircase, that are modelled in the simulation by a series of wheels. This suboptimal simluation of tracks leads to constant slippage effects and necessity to maximize the number of contact points to reduce it. That maximization is possible when the main tracks touch the staircase surface, which also lowers the robot gravity center and decreases instability I.

3.2.3 Resilience of policy learning to noise

Our targeted application in the context of AAL where the robot is integrated within a smart space/house [3], suggests a relatively well-controlled environment. The part of the state of the robot concerning its relative position with respect to the traversed steps could be obtained by fusion of on-board sensors with ambient sensors. Nevertheless, due to potential slipping of the platform while traversing the staircase it is important to assess the robustness of policy learning in the presence of varying levels of noise, since erroneous state estimation could lead to wrong actions and influence the robot performance even making it impossible to attain the desired goal.

Thus, noise should be accounted for during learning at the moment when we provide the robot with estimation of its relative position to the next step. We simulate corrupted estimation of p_x, p_y via a Gaussian error model, namely, $\widehat{p_i} \sim \mathcal{N}(p_i, \sigma_i)$ where $\widehat{p_i}$ is the noisy position, p_i is the true position given by the simulation environment, $\sigma_i = \iota \cdot V_{max}$ is the chosen standard deviation, i denotes either X or Y axes, V_{max} stands for the maximum possible observation value and ι is a parameter modelling different noise levels.

We experiment with three levels of noise, abbreviated as *low*, *medium* and *high*, that correspond to $\iota = \{0.1, 0.3, 1.0\}$.

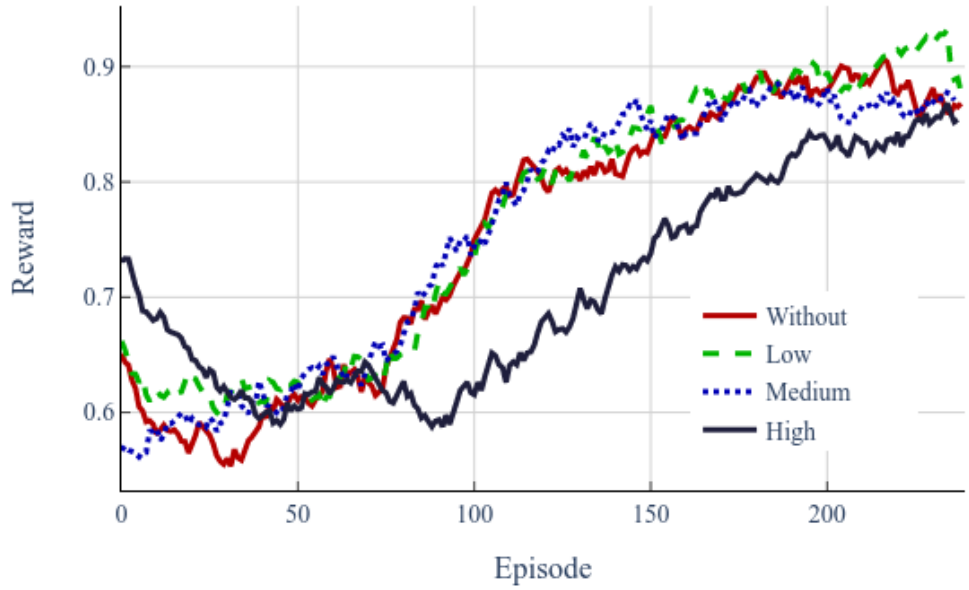
Firstly, three sets of experiments were conducted for each noise level using the default reward function. We can observe (see Figure 3.7 a)) that policies learned with low or medium noise converge as the default policy without the presence of noise. On the other hand, high noise in state estimation decreases the learning convergence rate, but eventually, the same episode return is attained after 250 episodes. During testing (see Figure 3.7 (b)), high noise levels seemed to have a more notable impact on the final episode return, yet, we do not expect such extreme noise levels to be representative of real conditions of an AAL scenario. For reference, the average values obtained over all 250 episodes provided in Tables 3.2, 3.3 demonstrate that noisy state estimation does not have an noticeable impact on safety and projection constraints.

Table 3.2: Resilience to noise for safety and projection constraints during learning

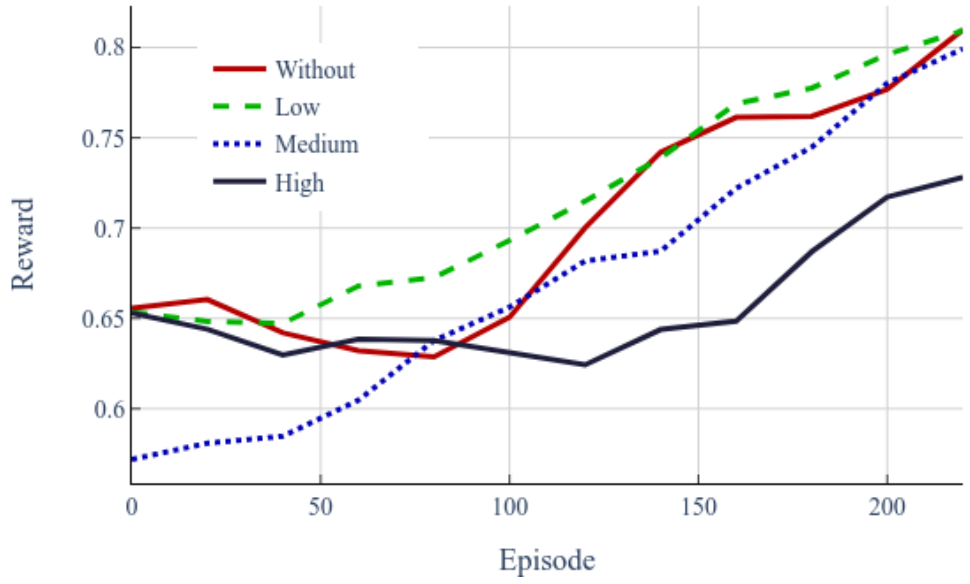
Measure type	**Level of noise**			
	Without	**Low**	**Medium**	**High**
I	0.017 ± 0.021	0.015 ± 0.018	0.013 ± 0.025	0.005 ± 0.009
P/P_{max}	0.962 ± 0.05	0.986 ± 0.007	0.966 ± 0.028	0.967 ± 0.024

Table 3.3: Resilience to noise for safety and projection constraints during testing

Measure type	**Level of noise**			
	Without	**Low**	**Medium**	**High**
I	0.018 ± 0.009	0.022 ± 0.008	0.017 ± 0.009	0.015 ± 0.009
P/P_{max}	0.991 ± 0.003	0.99 ± 0.003	0.992 ± 0.002	0.99 ± 0.003



(a) Learning



(b) Testing

Figure 3.7: Smoothed episode return $R(\tau)$ during (a) learning and (b) testing for the default reward function when perturbed by different noise levels.

3.3 Summary

In this chapter we have presented an initial version of the RL framework with a simplified environment for the staircase negotiation control learning. Within 200 trajectories which correspond to 60 policy updates, the robot learned how to safely traverse staircases consisting of 5 steps with varying riser and stair angle that were representative of real-world ranges.

We have designed and compared three alternative reward functions. Incorporating the projection maximization into the reward function produces the desired behavior in terms of the flipper sticking to the staircase. At the same time, the NESM-based policy behaved less favorably with respect to the projection-based policy. Overall, learnt policies exhibited a good generalization capability when applied to newly encountered stair parameters. Finally, it was observed that noisy sensory data may decrease convergence rate, however, the final control policy attains the maximum reward in most cases.

This prototype serves as a paradigm for the development of more elaborated robot skills, in order to further account for the presence of an active arm and treat the second half of the complete staircase negotiation task, namely, the riskier task of descent while potentially carrying a load. Furthermore, this study allowed to identify weak points of the simulation environment and the simulated robot so as to upgrade our framework to a more realistic simulation.

Chapter 4

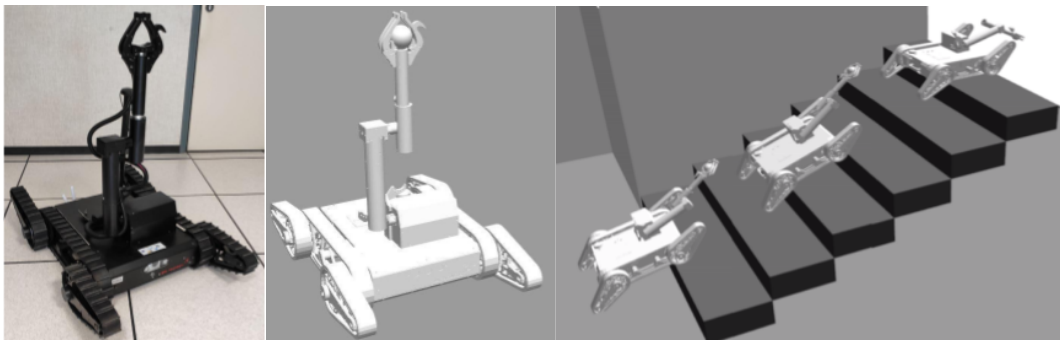
Complete Staircase Negotiation with Arm Control

The prototype presented in the previous chapter serves as a paradigm based on which we can now focus on reducing the gap between simulation and reality and address more thoroughly the problem of staircase negotiation for a real tracked robot manipulator. This means going beyond a wheel-based model of tracks that is sub-optimal in modelling the interaction of the robot with a staircase, further account for an active arm transporting a load and finally, tackle the problem of staircase descent. In detail :

- We upgrade the simulated robot model using geometric, kinematic and physical characteristics of the commercial robot *Jaguar V4 with Arm*[1] and adopt the Contact Surface Motion (CSM) [35] approach for the simulation of robot tracks (see Figure 4.1).
- We revise the initial RL-based problem formalization to allow learning of staircase ascent as well as descent and further investigate the influence of an active arm during staircase negotiation.

The methodological details of these aspects are provided in section 4.1 and finally, in section 4.2, the experimental setup is presented together with the obtained qualitative and quantitative results.

[1] `jaguar.drrobot.com/specification_V4Arm.asp`



(a) (b) (c)

Figure 4.1: Jaguar V4 Manipulator with raised arm (a), the corresponding CSM simulated model carrying a load (b) and simulated ascent staircase negotiation using a learnt policy (c)

4.1 Reinforcement Learning Framework

In this section, we rely on the same RL formalism detailed in Section 3.1.

4.1.1 Extended Flipper Control Problem Formalisation

In the previous chapter where we opted for independent tracks and flipper control, the robot was able to learn a behavior required for accomplishing staircase ascent while respecting NESM-based and projection-based criteria. However, with the inclusion of the DOF of an articulated arm, separate control of the main tracks with the arms actuators can be sub-optimal.

This motivated us to extend the previous action and state (recall eq. (3.5) and (3.6)) spaces and explore two types of action spaces; (i) when the robot controls 3 DOF consisting of the robot base linear velocity, front flipper and rear flipper angles and (ii) $3 + 2 = 5$ DOF where the 2 additional DOF correspond to the joints of an arm. With reference to (i), the robot selects 3 action parameters, namely, front and rear flippers rotation angles and velocity of the base forming an action vector:

$$\boldsymbol{a} = (\psi_a^{front}, \psi_a^{rear}, v_a) \in [\psi_a^{min}, \psi_a^{max}]^2 \times [v_{min}, v_{max}] \tag{4.1}$$

where ψ_a^{front} and ψ_a^{rear} are front and rear rotational displacements and v_a is the applied linear velocity to main tracks. For case (ii), the previous action vector is extended with

the joint angles of the arm:

$$(\phi_a^1, \phi_a^2) \in [\phi_1^{min}, \phi_1^{max}] \times [\phi_2^{min}, \phi_2^{max}] \tag{4.2}$$

that correspond respectively to the rotational displacements of the first and second joints (see Figure 4.2).

The state space structure is set as follows. For case (i), the observation vector is represented as:

$$\boldsymbol{s} = (p_x^{front}, p_y^{front}, p_x^{rear}, p_y^{rear}, v_s, \psi_s^{front}, \psi_s^{rear}) \tag{4.3}$$

where p_x^{front}, p_x^{rear} are distances of the chassis geometrical center relative to the next and previous nearest step edge along the X-axis of the robot as shown in Figure 4.2, with p_y^{front}, p_y^{rear} corresponding to the relative distances along the Y-axis and $v_s \in [v_{min}, v_{max}]$ being the current linear velocity. The state further includes ψ_s^{front} and ψ_s^{rear} as the current flipper angles. For case (ii), the state vector is extended with the arm joint angles ϕ_s^1, ϕ_s^2.

Despite these changes, we can still rely on the same RL-based problem statement described in section 3.1 after changing accordingly the input and output neurons of the neural network to match the new state and action vectors. On the other hand, a more mindful treatment is required for the design of reward functions due to the increased risk of staircase traversal failure.

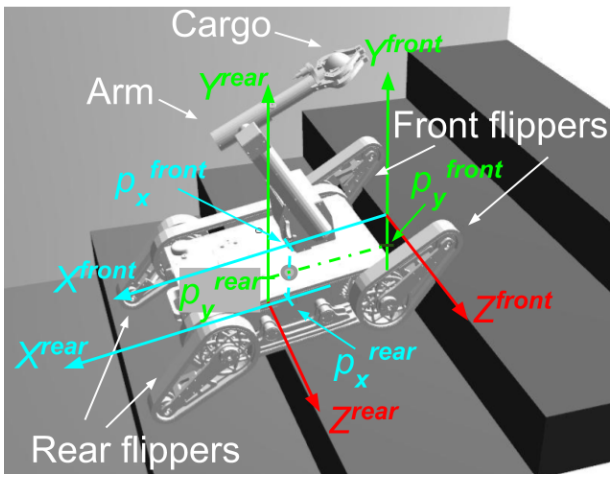


Figure 4.2: Front and rear obstacle (step edge) coordinate systems

4.1.2 Policy and its training

We use a two-layer perceptron for policy representation and PPO for its parameters optimization as in Section 3.1.2.

4.1.3 Reward function design

In this section we present reward functions used for learning staircase ascent and descent. In detail, we employ the same positive reward as in chapter 3 representing the total travelled relative distance on the stairs which drives learning according to eq. (3.7). As stated earlier, learning with constraints can be performed through accounting for a negative reward (penalty) within an episode. However, it is necessary to rethink the negative penalty which drives acquisition of specific skills due to presence of the arm and of different dynamics during descent.

We first propose a reward function that addresses the problem of robot stability for ascent negotiation and respects both SM and NESM criteria. Then, a reward function meant to reduce the pitch angular velocity experienced by the platform during descent is proposed.

Center of Gravity stabilization

The center of gravity of the robotic platform is known to be instrumental in preventing tip-over accidents and improving stability [76, 115]. In chapter 3 we enabled a robot to learn a stable behavior by favoring robot poses with low center of gravity with respect to the underlying surface A' that represents NESM. The presence of an arm, nonetheless, may place the COG low but also close to the "safety zone" border [76], without violating the NESM.

To overcome this problem, we use the SSM [98] that estimates the distance between the lower footprint and the COG projection on the ground. Since it is generally not straightforward to estimate where exactly the lower footprint touches the ground and it is necessary to acquire no or low penalties when the robot is in or near its stable state, we instead minimize the deviation of the COG projection on the stair surface C_x from the projection point O of the centroid of the robot base. We termed earlier the point C_x as "projection" as well as for the rest of the manuscript for convenience and simplicity. More formally, C_x is obtained when a line parallel to gravity that passes through COG intersects the hyperplane A'. Minimizing this deviation, we can guarantee that the robot

respects the SM criterion. We require that the arm contributes in placing the COG as close as possible to the point O - the most stable point (see Figure 4.3) along both the X and Y axes, optimizing both for SM and the NESM.

Thus, we need minimize $D^{dev} = \sqrt{\delta_x^2 + \delta_y^2}$, that we refer as "deviation", where δ_x and δ_y are the distances between O and the projections of the COG on X and Y axes. This minimization serves the purpose of improving stability through the SM and NESM criteria but also redistribute symmetrically the weight forces exerted by the moving robot tracks to the staircase.

We further wish to strictly penalize the robot when it loses stability and tips over that happens when the pitch angle of the robot $|\theta_{pitch}|$ reaches $\pi/2$.

Thus, letting the COG deviation at every time step be D_t^{dev}, the corresponding normalized negative reward is defined as:

$$r_t^{n,\ COG} = \begin{cases} -1, & \text{if tip over } |\theta_{pitch}| > \pi/2 \\ -K_D * D_t^{dev}, & \text{otherwise} \end{cases} \tag{4.4}$$

where the scaling coefficient K_D is used for normalization as will be explained later in 4.1.3. Thus, the total time step reward for the COG deviation is:

$$r_t = r_t^p + r_t^{n,\ COG} \tag{4.5}$$

Besides safety, we further observed that **the contact of the robot tracks with**

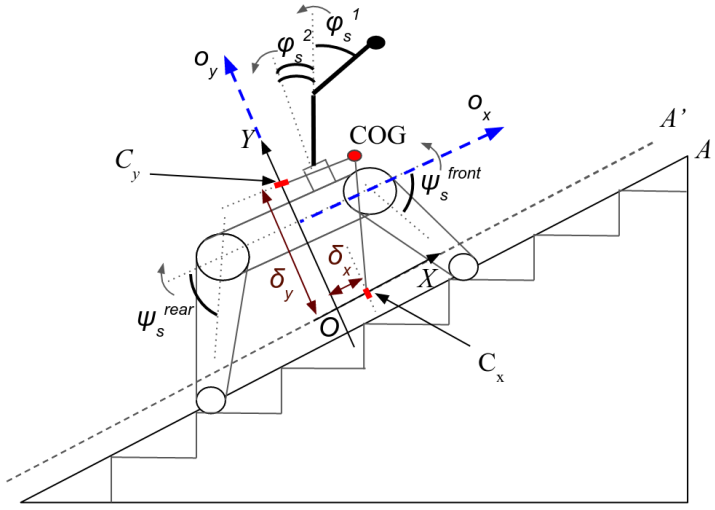


Figure 4.3: Side schematic view of the robot manipulator on a staircase

the surface cannot alone guarantee proper traction. In practice, it is essential that sufficient force is exerted from the robot tracks to the ground. This is better ensured when the projection of the COG C_x is located near the projection O of the geometrical robot center onto the staircase plane, as this leads to a more uniformly distributed traction along the entire tracks of the platform.

Drop impact reduction

Robot dynamics are different between ascending and descending a staircase, as gravity hinders the accomplishment of the former and it may conduct the robot to fall in the latter. Even if no accident happens, repetitive collisions caused by every step negotiation during descent would have impact on the robot. Such events appear when the COG crosses step edges and the chassis starts rotating downwards which yields an increase of pitch angular velocity that we refer as drop impact (bumpiness). As a way to prevent such behavior, the robot can learn to better control such dynamics through a pitch angular velocity based penalty. The robot has also increased chances of experiencing a frontal tip-over when the pitch angle of the robot reaches $-\pi/2$, that we wish to penalize as well.

We follow the idea of drop impact measurement discussed in Chapter 2.4. W_t denote the threshold average pitch angular velocity as the last time step mean value over observations of W_i, which is the *i-th* threshold pitch angular velocity of the last time step defined as follow:

$$W_i = \begin{cases} |W_i|, & \text{if } |W_i| > W_{threshold} \\ 0, & \text{otherwise} \end{cases} \tag{4.6}$$

$W_{threshold}$ is the threshold angular velocity, which is manually adjusted to cut off the observation noise of the IMU. Then, the corresponding normalized negative reward is:

$$r_t^{n,\ ang} = \begin{cases} -1, & \text{if tip over } |\theta_{pitch}| > \pi/2 \\ -K_W * W_t, & \text{otherwise} \end{cases} \tag{4.7}$$

This penalty is assigned only in descent tasks and when the robot traverses the staircase, namely, when the COG projection on the staircase is located between the staircase nosing limits. In this case, the total time step reward is:

$$r_t = r_t^p + r_t^{n,\ ang} \tag{4.8}$$

Adjustment of scaling coefficient

A learned policy is determined by the reward function that guides learning of a specific task. At the same time, it can endow the policy with specific characteristics such as improvement of traction and respect of safety. While there is no ambiguity when a task is learned with a single reward, direct summing of multiple rewards or penalties, may mislead - bias the robot to concentrate on respecting the auxiliary constraints rather than the main task.

It becomes therefore necessary to control the arbitrary bias of (positive or negative) rewards, through scale normalization, that we also refer as reward or penalty scaling, that can balance the learning of the main task along with the desired property. Note that for the positive, default reward related to the travelled distance (recall section 3.1.3) this is straightforwardly obtained.

To assess the amplitudes and in turn the biases for the negative penalties described previously, we initiate a small set of trial experiments for the first n steps where we do not assign negative rewards, in order to acquire sufficient statistics related to the target value V, for example, COG distance D^{dev} or angular velocity W_i (see eq. 4.6). When used as a superscript, V denotes either COG or ang. This results in making the robot optimize its behavior only with respect to the main task as specified by the positive reward, setting aside the consideration of the penalty term. Upon the completion of the n^{th} step, we calculate the "sub-optimal" mean target value $\hat{V} = \sum_{t=0}^{n} V_t/n$, then the scaling coefficient as follows:

$$K_V = (\hat{V} \cdot T_{max})^{-1} \tag{4.9}$$

where T_{max} is the maximum episode length. The episode return could be expressed as follows:

$$R(\tau) = R^p(\tau) + R^n(\tau) = \sum_{t=0}^{T} r_t^p + \sum_{t=0}^{T} r_t^{n,V} \tag{4.10}$$

where $r_t^{n,V}$ is the target value normalized negative reward. The total positive trajectory return R^p varies from 0 to 1 by definition. Normalization guarantees that the total trajectory penalty without tipping over penalization $R^n(\tau)$ varies from -1 to 0. This can be easily seen by defining the :

$$R^n(\tau) = K_V \sum_{t=0}^{T} V_t = \frac{\sum_{t=0}^{T} V_t}{\hat{V} \cdot T_{max}} \cdot \frac{T}{T} = \frac{\hat{V}_\tau T}{\hat{V} \cdot T_{max}} \tag{4.11}$$

where $\hat{V}_\tau$ is the mean target value during the episode after trial experiments.

We suppose that in the beginning of learning the policy is sub-optimal, thus the "sub-optimal" mean target value $\hat{V}$ calculated in trial experiments relates to further target value mean observations as $\hat{V}_\tau \leq \hat{V}$. Since the rollout length does not exceed its maximal length $T \leq T_{max}$, we can conclude:

$$\hat{V}_\tau T \leq \hat{V} \cdot T_{max} \tag{4.12}$$

It was finally observed that the tip-over penalty prevents learning of sub-optimal policies, therefore we added auxiliary negative reward -1 and end the episode in such cases, thus, $R^n(\tau) \in [-2, 0]$. Finally, we obtain the limits for the episode return as $R(\tau) \in [-2, 1]$.

4.2 Experiments

In this section we present the experiments that we conducted, allowing us to obtain staircase ascent and descent policies with desired properties. We also provide a qualitative analysis of learned behaviours (available at `partage.imt.fr/index.php/s/JBdmEXaWcjLmgmB/download` as a part of the publication [45]).

4.2.1 Robot model improvement

We shift from a wheeled-based simulation of robot tracks to the more realistic CSM model [35] that is more lightweight in terms of computational speed and plausible on even and rough terrains. We instantiate a model of the robot Jaguar V4 and add a manipulator arm platform (see Figure 4.4) incorporating its geometrical and mass distribution parameters. The hardware description of the robot, when the manipulator is folded, are presented in Table 4.1 from where the manipulator can reach $0.707m$, while mass is set to half of the real robot to deal with instabilities occurring in the simulation environment Gazebo due to increased forces. Therefore, the COG is not altered and corresponds to the same position as for the original total mass.

The change of mass does not impact angular velocity either. When the robot rotates around a step edge it transforms its potential energy into rotational kinetic energy that we could approximate through the next formula $mgh_{pot} \propto ZW^2$, which gives $W \propto \sqrt{\frac{mgh_{pot}}{Z}}$, where m is the robot mass, g is the gravitational constant, h_{pot} is the vertical displacement of the COG, Z is the moment of inertia and W is the pitch angular velocity. We can state that $m = \rho F$, where ρ is the mean density, F is the robot volume. We assume that the body density is uniform, because the robot is composed of heavy metal parts, therefore we can neglect the presence of plastic parts. Therefore $Z = \rho \int_F x^2 dF$, where x is the distance from the axis of rotation to the element dF. We can deduce that $W \propto \sqrt{\frac{\rho F g h_{pot}}{\rho \int_F x^2 dF}} = \sqrt{\frac{Fgh}{\int_F x^2 dF}}$. Thus, pitch angle velocity does not depend on the density, and mass change of the platform has an insignificant impact under assumption that the body density distribution is uniform.

The main tracks are connected to the chassis through fixed joints. Flippers are located at each end of both tracks and attached through the revolute joints, which can rotate in the following safety limits $[-\pi/4, \pi/4]$ around the "extended" configuration. Revolute arm joints 1 and 2 have the same safety limits. A payload with variable mass can be attached at the end-effector of the arm joint 2. The model can be controlled through ROS topics.

The arm and flipper joints are position controlled. We can control the velocity of the robot by setting linear and angular velocities. The Gazebo plugin proposed in [35] calculates and applies corresponding forces to the robot. The robot contains a standard ROS IMU sensor which can provide linear and angular accelerations and orientation. Finally, we place an RGB-D sensor at the front of the robot.

4.2.2 Experimentation set-up

Policy training protocol

We improve the simulation environment of Chapter 3 as illustrated in Figure 4.4. Staircase size is still varied in ranges corresponding to real-world staircases to address the domain randomization as in Chapter 3. However, we increased the maximum total number of steps from 5 to 10, and the number of steps is randomly generated every episode.

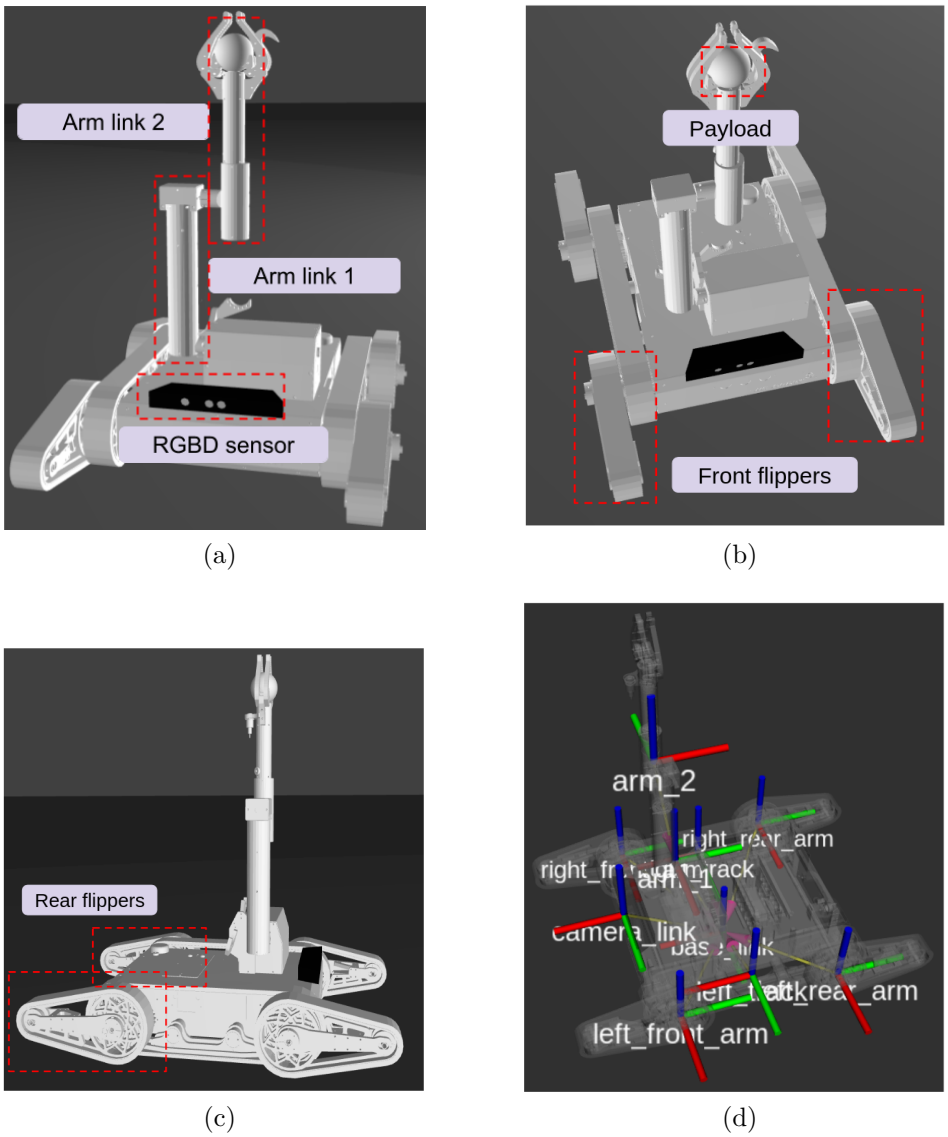
To account for the influence of transporting objects we perform learning with a cargo added permanently to the robot end-effector that constitutes 10% of total robot mass and corresponds to the heaviest permissible load of the real robot. In the beginning of the ascent tasks, the robot starts at the start point (see Figure 4.5) being orientated towards the stairs. It has to traverse the distance from the start to the finish line. For the descent tasks, start and goal positions are swapped. We assign both positive and negative rewards from the beginning of the episode to its end. The episode ends when either the robot reaches the finish line, goes out of the training zone (rectangle defined by red, blue and green lines), tips over or number of time steps exceeds the maximal episode length.

We studied learning a total of 5 variants of the staircase traversal task, distinguished by the staircase negotiation direction, DOF involvement and penalty optimization criterion as summarized in Table 4.2, where COG deviation and angular velocity criteria correspondingly refer to Equations 4.4 and 4.7.

For all experiments the robot arm starts with the same, vertically stretched, initial configuration perpendicular to the robot base (see Figure 4.4 (c)). We select this initial arm pose as it cuts through symmetrically the arm workspace and thus does not add a

Parameter	**Mass, kg**	**Length, m**	**Width, m**	**Height, m**
Value	20.5	0.98	0.7	0.4

Table 4.1: *Jaguar* characteristics in simulation



(a) (b)

(c) (d)

Figure 4.4: CSM Jaguar model in the "extended" configuration. Robot front view (a), robot top view (b), robot side view (c), robot joints (d)

particular bias to the COG when moving on flat terrain. On the other hand, such pose may severely hinder staircase traversal and should therefore incite the robot to learn to move the arm at a better pose. We recall that in the case of a 3DOF action space, the arm remains fixed in this pose.

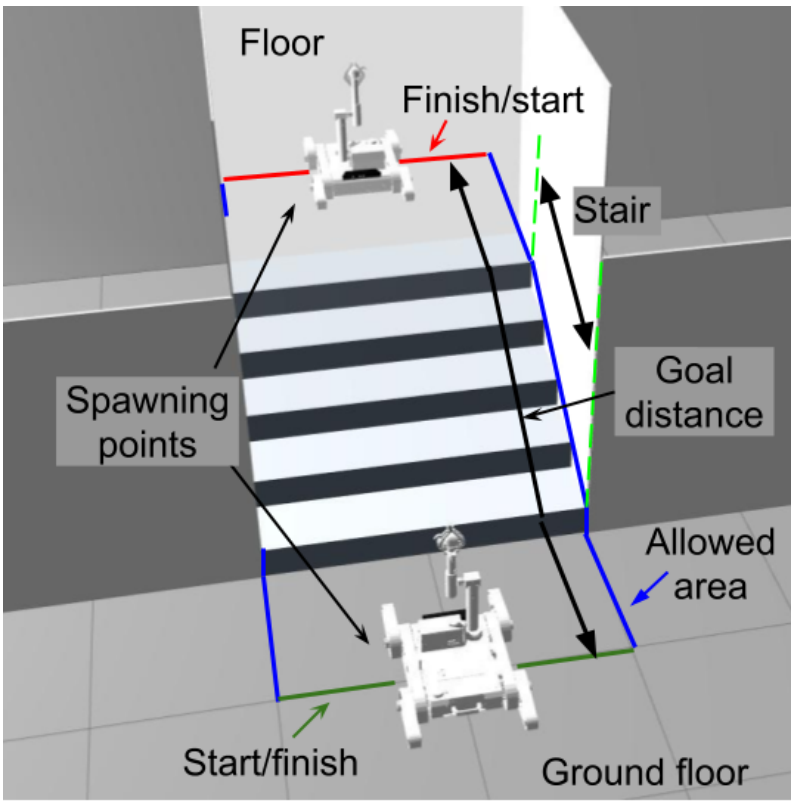


Figure 4.5: Simulation environment with robot positions

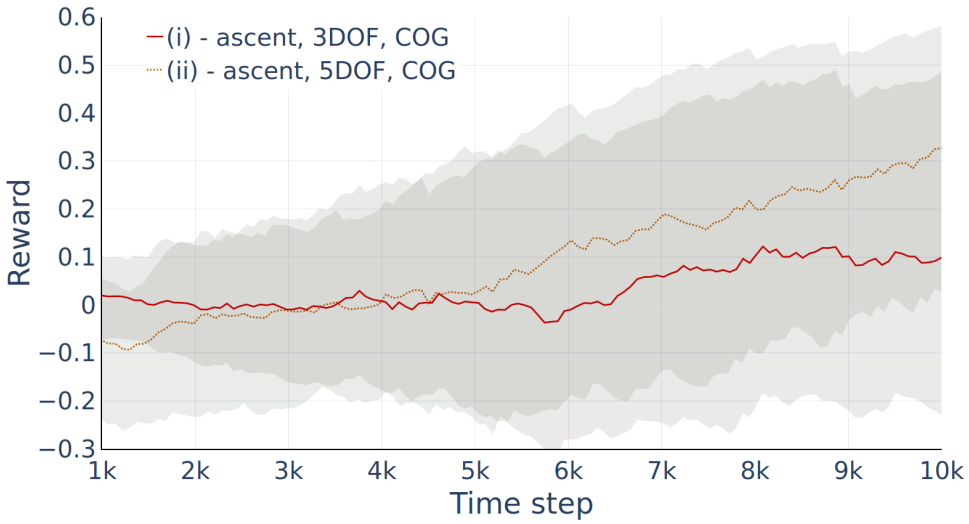
4.2.3 Reward function evaluation

Staircase ascent performance for simulated Jaguar V4 with Arm

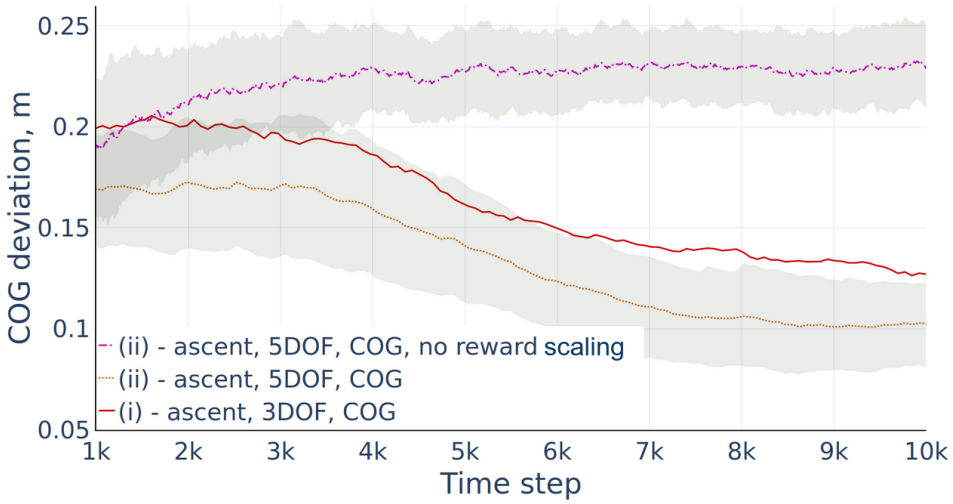
Three independent learning trials were performed for each task (i) and (ii) (see Table 4.2), after which we average the obtained performances. Figure 4.6 (a) presents the average over all trials of the exponentially smoothed episode return with min-max bands, as was done in Chapter 3.

Task id	Direction	DOF	Criterion
i	Ascent	3	COG
ii	Ascent	5	COG
iii	Descent	3	Ang. vel.
iv	Descent	5	Ang. vel.
v	Descent	5	COG

Table 4.2: Description of tasks



(a)



(b)

Figure 4.6: Ascent task learning analysis

At the end of learning, a total relative reward difference of 0.21 is observed between the tasks, clearly suggesting that **the robot learns to control the arm in a way that optimizes the respective criterion**. Even if the min-max episode reward spread increases by the end of learning, we conclude that the robot acquires the necessary skills, because the COG deviations curves (Figure 4.6 (b)) converge to a plateau while decreasing the min-max COG deviation spread.

Episode return curves do not reach their maximum positive return value of 1 because of the minimal COG bias that results from its inability to put the COG closer to the base

centroid than a certain distance. It is worth noticing of the considerable distance between the two curves by the end of learning, explained by the fact that the robot with static raised arm and only using its flippers, is unable to attain the same level of positive reward because of the higher COG deviation.

Figure 4.6 (b) shows how the COG deviation evolves during the learning process. The purple curve associated with the policy (ii) "no reward scaling" shows the importance of the scaling coefficient presented in Section 4.1.3. This curve is produced by directly accounting for the penalty but without the K_D multiplier in Formula 4.5. We see that the COG deviation goes up and reaches a plateau around $0.225m$ by 3000 time steps, which indicates absence of any meaningful optimization even degeneration in comparison to the beginning of learning.

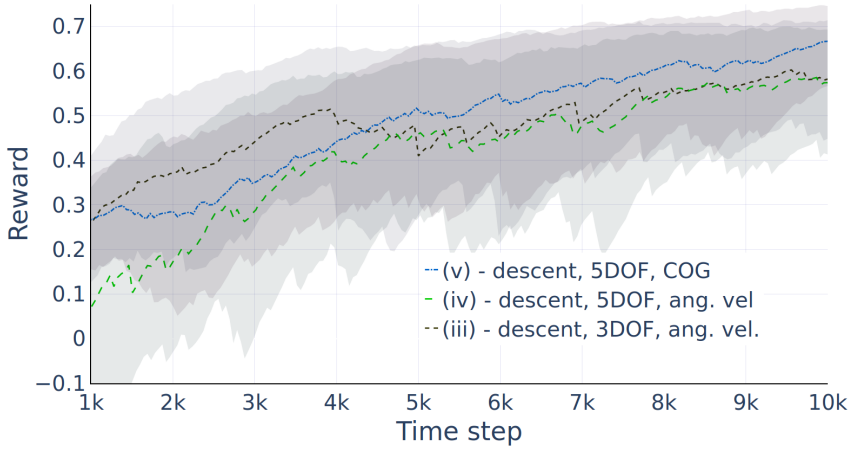
By contrast, the policies (i) and (ii) exhibit successful learning through constant decrease of the COG deviation and reaching a plateau by 8000 time steps correspondingly at $0.13m$ and $0.11m$. The robot learns to control better its dynamics in both tasks. It also shows that the robot converges to a better behaviour in a similar pace. Eventually, the arm control in task (ii), allows to decrease the COG deviation by 0.03 m lower compared to the policy (i) by the end of learning, which favors usage of the arm control.

Staircase descent performance for simulated Jaguar V4 with Arm

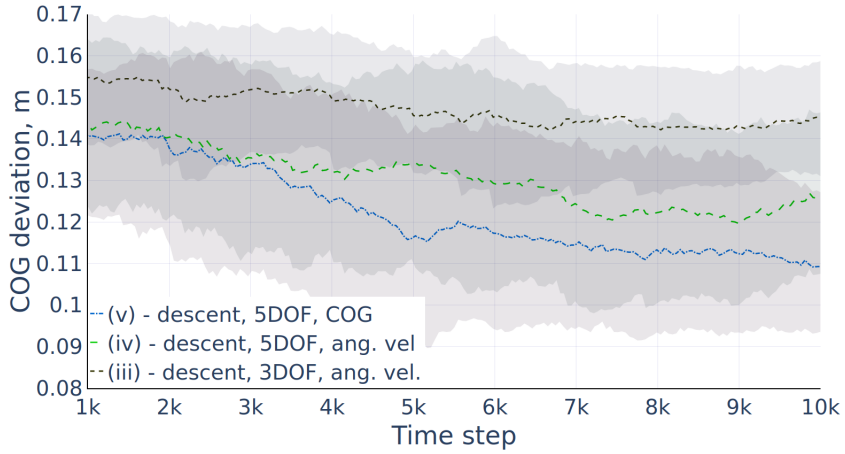
Episode return curves are presented in Figure 4.7a. We can see that the reward for all tasks reaches approximately the value 0.6 and does not attain the maximal reward 1 because of constant presence of a minimal negative reward.

Figure 4.7b shows the evaluation of the COG deviation during learning. As expected, we observe that the most unstable robot behavior is obtained for task (iii), where the reduction of COG distance due to the flipper control from the centroid projection point is low. Results of pitch angular velocity optimization with moving arm (iv) seems to improve stability, allowing to reduce the COG deviation by a total of $0.02m$. Still, direct minimization of the COG distance (v) provides the best overall results, wherein the COG decreases by more than 0.03 m and attains the lowest absolute value among tasks.

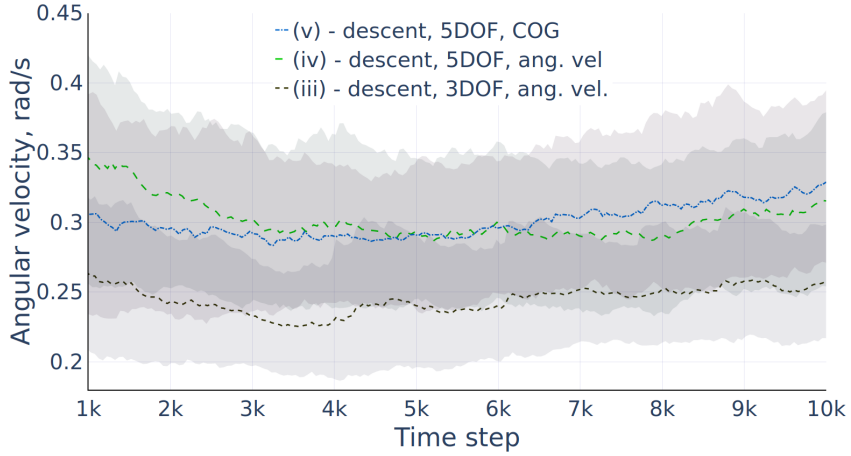
The pitch angular velocity evaluation is presented in Figure 4.7c. Comparing the pitch angular velocity and COG optimizations (iv) and (v) we can observe that, as expected, the former yields a slightly lower angular velocity. Comparing with the performance of task (iii), we could more confidently claim that the mean pitch angular velocity is smaller if the arm does not move, which merits further attention. As we have seen, the initial,



(a)



(b)



(c)

Figure 4.7: Descent task learning analysis

vertically stretched arm performs the worst in terms of stability easily leading to frontal tip-over, end of episode and penalization by -1. Thus, starting in a vertically stretched arm position, the robot would experience more tip-overs that drives the arm control to more stable configuration even if this increases pitch angular velocity during movements. Therefore, this turns out to be a plausible behavior in reality, as we would prefer to undergo small bump impacts due to increased pitch angular velocity instead of a drastic accident.

Policy test analysis

To assess the effectiveness of learning, we no longer focus on the set-up presented in Section 3.2.1 as we consider that test curves of episode return do exhibit the same evolution as corresponding values during learning.

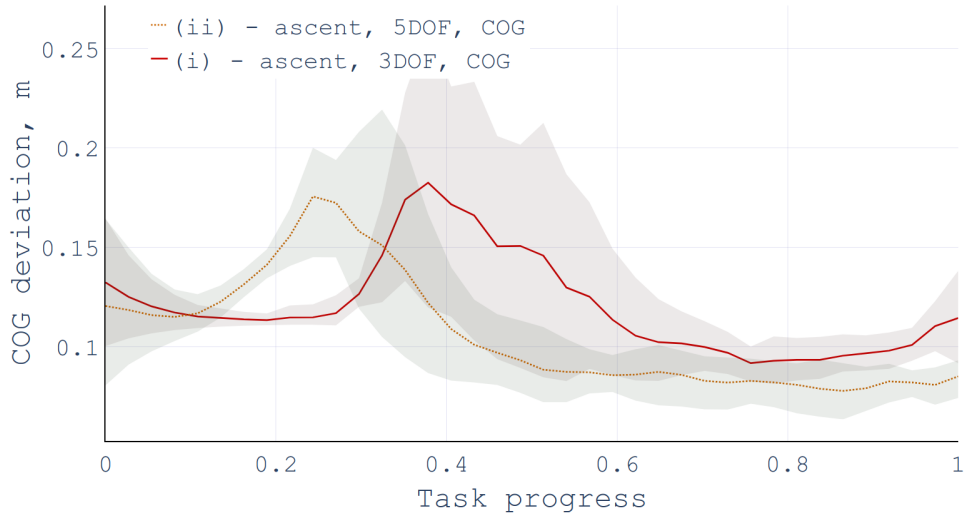
Instead, we now perform testing of policies after the complete learning convergence by measuring the COG deviation and pitch angular velocity in trials on the *medium* staircase (see Section 3.2.1). This is shown in Figure 4.8, where we show evolution of optimized parameters during a trial and discuss it after.

We refer the interested reader to the qualitative results provided in the accompanying video[2], which presents the policies learned for the ascent and descent tasks in simulation. Policy learning time is constrained by action execution duration in the simulation. Using a contemporary machine, a 10000 time step simulation required approximately 20 minutes.

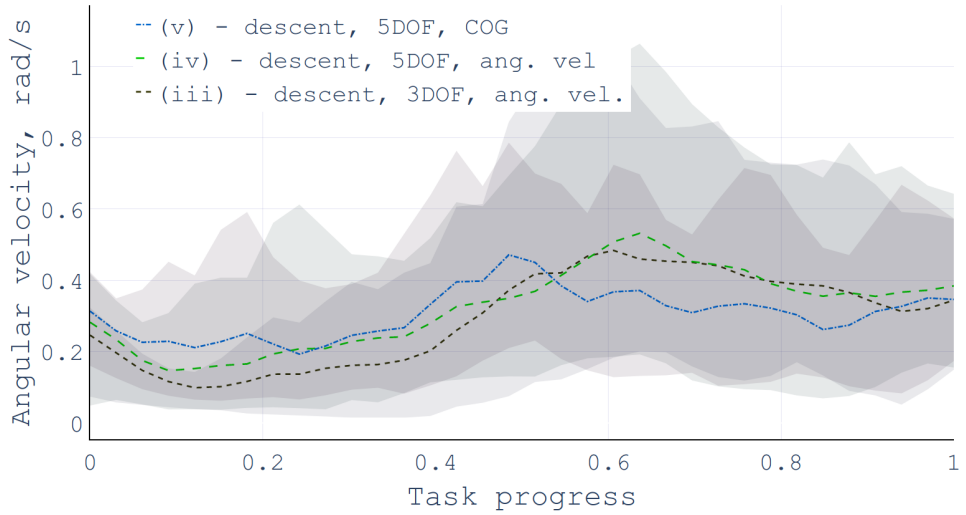
Figure 4.8a presents mean values with min-max of the COG distance during 10 test episodes in simulation for ascent tasks (i) and (ii). We can see that after the initial phase of the task progress, i.e. from 30% of the ascent of the staircase, the COG deviation for the policy (ii) is lower than for the policy (i), further illustrated by Figure 4.10 which presents mean values of COG deviations and angular velocities with their standard deviations during simulation test deployment for the corresponding experiments of Table 4.2. The mean COG deviation of the policy (i) is higher than the policy (ii) by 0.03 m.

Figure 4.9 provides indicative snapshots of robot configurations during ascent (a) and descent (b) transitions while respecting the COG criterion. **We can see that the robot pushes its arm in front during ascent, thanks to which tip-over of the platform is largely avoided. The front flippers are raised during ascent while rear flippers are pushed down and vice versa in descent. Such configuration**

[2] `partage.imt.fr/index.php/s/JBdmEXaWcjLmgmB/download`



(a)

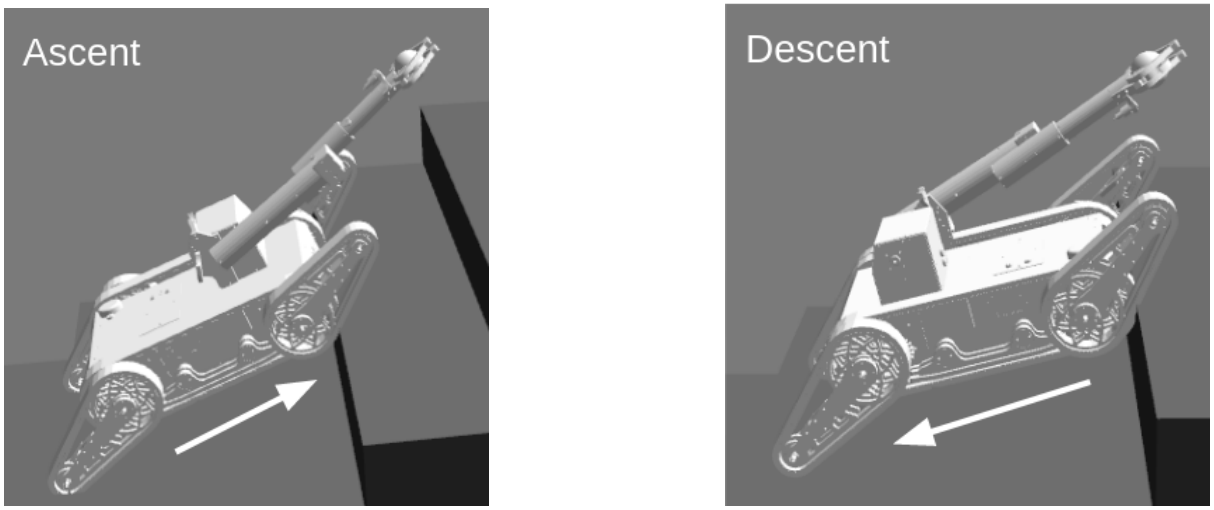


(b)

Figure 4.8: Evolution of penalty values during task execution

improves robot traction with front and rear step edges and also reduces the chance of a tip-over.

Mean curves with min-max bands given in Figure 4.8b show the evolution of the angular velocity during descent negotiation in simulation for tasks (iii), (iv) and (v). To evaluate policy effectiveness in the descent, tasks we compare quantitatively corresponding policies with the help of Figure 4.10. Angular velocity optimization shows better results for the policies (iii) and (iv) in contrast to the policy (v) which is correspondingly higher



(a) front arm configuration (b) rear arm configuration

Figure 4.9: Robot during transitions on the staircase

by around 0.04 rad/s and 0.05 rad/s. Yet, the latter shows the smaller COG deviation by 0.018 m and 0.028 m.

The robot configuration respecting the COG criterion during descent negotiation is shown in Figure 4.9 (b). Flipper rotation angles are set in a manner where front and rear flippers touch correspondingly lower and upper obstacles making transitions smoother. Thus, front flippers prevent the robot from dropping forward while rear flippers allow the robot to descend smoother from the rear step edge.

While we favor pitch angular velocity reduction in the descent tasks (iii) and (iv) achieving the better optimization for the policy (iv), preserving stability in task (v) remains important. We deem that the angular velocity minimization with arm control as the most favorable criterion to optimize in descent, as it directly leads to minimization of the angular velocity while it also seems to implicitly improve stability of the robot through arm manipulation.

Payload presence analysis

As we have mentioned, learning is performed with a payload which represents 10% of robot mass. This is used to show the adaptation of the policy to load variation via test roll-outs with masses that constitutes 5% to 15% of total robot mass.

We have performed additional tests on the *medium* staircase configuration (see Section 3.2.1) in the ascent and descent tasks for the same policies when the end effector holds

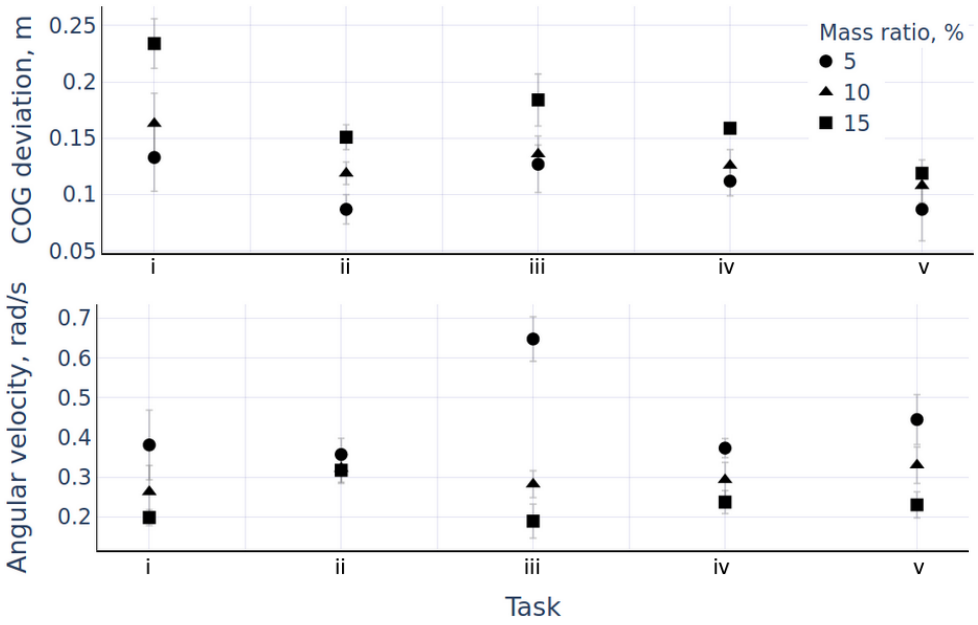


Figure 4.10: Performance of learnt policies with varying weight of transported object (see Table 4.2 for task definitions)

loads constituting 5 to 15% of the robot mass. We can observe from Figure 4.10 that the robot is capable of controlling the COG deviation if load mass changes in the ascent tasks (i) and (ii). For example, the robot controls the arm in a manner that it improves the target value for tests with loads of $5, 10$ and 15% of the robot mass by $0.046, 0.044$ and 0.083 m in comparison with the stretched arm tests. In the descent tasks, the COG deviation is the highest in the task (iii) for all load types. Then, it is decreased if the robot controls its arm to minimize the angular velocity, but the minimal target value corresponds to the policy of the task (v). The mean angular velocity decreases along with load mass augmentation for all tasks. We can conclude that the angular velocity is smaller for tasks (iii) and (iv) than for task (v).

4.3 Summary

We have presented an improved RL-based framework for the problem of control policy learning during staircase descent and ascent, further investigating the influence of an integrated arm. Within 150 episodes, the robot is able to learn its dynamics and safety constraints while negotiating varying staircases. The automated scaling coefficient estimation proved effective by constraining episode returns to appropriate scales and avoiding biasing the convergence of the obtained optimal policies.

We have studied optimization of the COG and pitch angle velocity criteria for both ascent and descent. The COG based policies have exhibited better performance in terms of stability and qualitatively presented the same optimal arm control behavior as it would be expected by classical control. Despite the addition of more DOF, the control of the arm yields better overall stability to the robot during traversal. The angular velocity minimization seemed to indirectly improve the COG deviation. The learned control has shown resilience and appropriate adaptation to different carrying loads.

Chapter 5

Policy Portability and Transfer from Simulation to Reality

In this chapter, we built upon the acquisition of robot controllers with desired behaviour properties such as safety and bumpiness minimization through reward function design, in line with our earlier works [45, 116] and previous chapters. In particular, we consolidate the training of effective controllers in simulation, by developing and comparing these skills on two distinct articulated, tracked robot vehicles in simulation and further successfully transfer and deploy the trained policies onto one of the real robots.

On top of attaining our main goal that consists in successful staircase negotiation by the real robot, we also present new insights that are the product of quantitative as well as qualitative cross-comparison of behaviors among task variants and between robots. Concretely, the work presented in this chapter and detailed in [117] advances the state-of-the-art in the following points :

- We successfully acquire RL-based staircase negotiation controllers for two robotic platforms in simulation, both exhibiting the desired behavior properties.
- We quantitatively compare controllers obtained for different robots and task variants via Kullback-Leibler (KL) policy divergence.
- The efficacy of the obtained controllers is demonstrated in reality by transfer to a commercial robot (see Figure 5.1).

In the sequel, we present the experiments that we performed with two different platforms, namely, **Absolem**[1] [17] and **Jaguar V4 Manipulator**[2]. Extensive simulated and real-world experiments were performed with varying degree of difficulty for the task of staircase negotiation, allowing us to successfully deal with the associated challenges (see also associated video `partage.imt.fr/index.php/s/LDyp6QRp4nGGKd2/download`).

[1] `bluebotics.com`, `github.com/mariogianni/trav_nav_indigo_ws`
[2] `jaguar.drrobot.com/specification_V4Arm.asp`

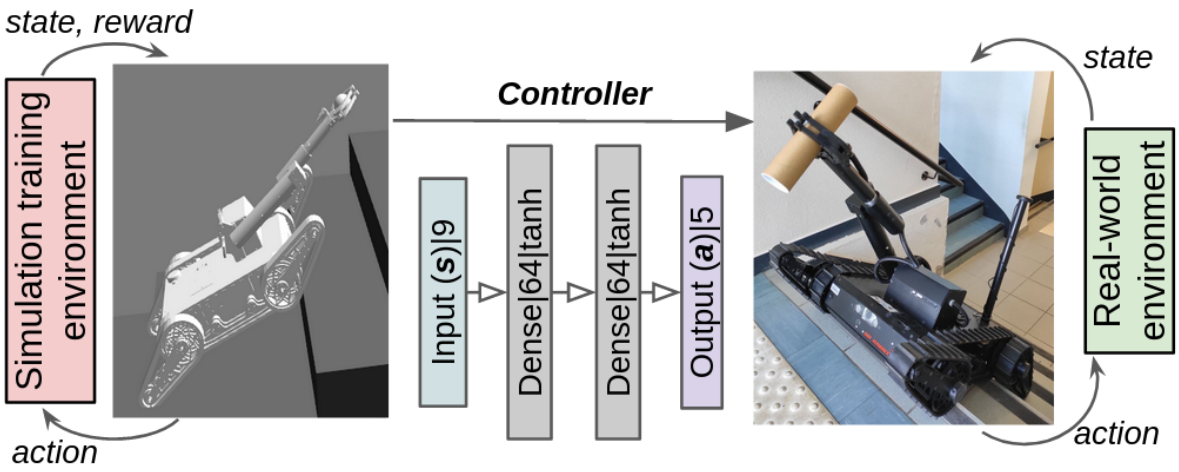


Figure 5.1: Pipeline overview: the desired behavior is developed via RL in simulation and transferred in reality

The rest of the chapter is organized as follows. Section 5.1 presents the robot platforms along with the characteristics of the tasks and the policies that were trained followed by the presentation of policy comparison. We discuss ascent and descent learning results and the procedure for transferring and evaluating the obtained behaviors onto a real robot in section 5.2.

5.1 Policy comparison

5.1.1 Description of platforms and associated tasks

We rely on the simulation environment, learning workflow, action and observation spaces from chapter 4 and additionally acquire a model of *Absolem* (see Figure 5.2) [3] that we endow it with the CSM model for track simulation, whose hardware description can be found in Table 5.1 and we refer to the Ph.D. thesis of M. Pecka [118, p. 10] for details. Adding a new robot has no impact on the overall training time since each robot controller can be trained independently and all of them in batch, server-side simulations.

In Table 5.2 we juxtapose the two articulated tracked robots for which we trained the various controllers. Their simulated models matched the geometric characteristics of the real ones, making use of vendor provided The mass characteristics of *Absolem* matches the real ones while those of *Jaguar* follow the description of Section 4.2.1. The two robots possess similar obstacle negotiation properties thanks to the presence of flippers, with the main difference being that Absolem can independently move each flipper in contrast to Jaguar where front and rear flippers are coupled. To make consistent comparisons, we therefore chose to couple Absolem flippers as well.

[3] `bluebotics.com`, `github.com/mariogianni/trav_nav_indigo_ws`

Parameter	Mass, kg	Length, m	Width, m	Height, m
Value	29	1.196	0.61	0.46

Table 5.1: *Absolem* hardware description

Figure 5.2: *Absolem*

Task	**Policy id**	**Policy id**
Ascent, 3DOF, Default	Jag-Asc-3-Def	Abs-Asc-3-Def
Ascent, 3DOF, COG	Jag-Asc-3-COG	Abs-Asc-3-COG
Descent, 3DOF, Ang.	Jag-Des-3-Ang	Abs-Des-3-Ang
Ascent, 5DOF, COG	Jag-Asc-5-COG	None
Descent, 5DOF, Ang.	Jag-Asc-5-Ang	None

Table 5.2: Task and policy variants to the tested robots

It follows that each robot has at most 4 DOFs corresponding to linear velocity, angular velocity, front and rear flipper angles, with Jaguar having 2 additional DOFs due to its arm. For the staircase traversal task though we have deemed that yaw angular velocity control was unnecessary during traversal after the following observations; (i) a robot can easily align itself with the staircase before traversal and (ii) the robot remains aligned during traversal thanks to the coupled front and rear flippers grip on the steps. Thus, the maximum number of DOF is 3 for Absolem and 5 for the Jaguar. We fix the heaviest admissible payload to the Jaguar end-effector in tasks during learning to account for the presence of a payload, where all 5 DOFs are involved. When *Jaguar* is trained with 3 DOFs for comparison with *Absolem*, we remove that payload.

Each task of Table 5.2 is thus characterized by the direction of traversal (Ascent or Descent), the number of DOF of the action space and the type of reward used to guide learning (*Default* refers to the use of Equation 3.7, *COG* refers to the use of Equation 4.5 and *Ang* refers to the use of Equation 4.8).

We evaluate 8 task variants as listed in Table 5.2 differing by traversal direction, DOF variation, applied criterion and robotic platform. The Jaguar robot model starts always with a vertically stretched arm that is fixed in 3 DOF tasks, flippers of both platforms are parallel to the floor in the beginning and $\phi_a^1, \phi_a^2 \in [-0.5\pi, 0.5\pi]^2$. Finally, penalties are evaluated and assigned only when robots reside on the staircase.

5.1.2 Policy comparison

A recurrent question arises from the moment where we can develop multiple different policies, depending on the reward function design, task characteristics or robot platform; **how can we quantitatively compare policies and the associated behaviors?** In answering this question, we make use of a commonly used probability distribution divergence measure, namely, the KL divergence [119]. Our motivation comes from the observation that, RL algorithms often exploit the divergence as metric to be optimized or as a constraint of policy updates as for example in PPO where updates of the policy parameters are constrained through a penalty on KL divergence.

We denote the reference policy for a given task as π_q and we wish to find how the policy π_p diverges from π_q (assuming same observation and action spaces). The process of learning a policy amounts to learning the mean and variance statistics of the underlying probability distribution, rather than the exact probability distribution for each possible state-action pair. Making the hypothesis of a normally distributed policy with mean and variance that depend on the state, a policy can then be sampled randomly over the total space of observations with actions chosen according to that policy. Thus, once a total set of observations has been obtained, the corresponding actions are prescribed by one of the two policies being compared, namely, the reference policy π_q. After having determined the state-actions pairs over which the policies will be compared, we then calculate the divergence as:

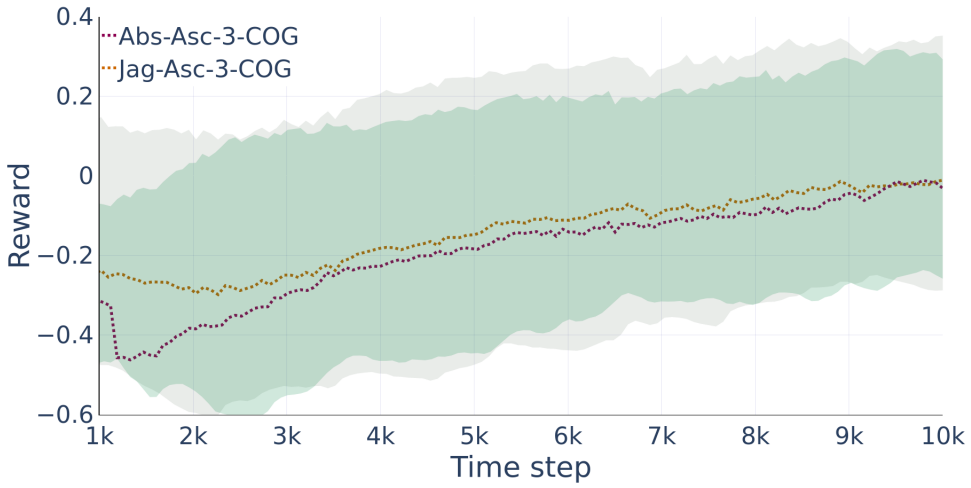
$$D_{KL}(\pi_p, \pi_q) = \sum_{\boldsymbol{s},\boldsymbol{a}} \pi_p(\boldsymbol{s}, \boldsymbol{a}) log \frac{\pi_p(\boldsymbol{s}, \boldsymbol{a})}{\pi_q(\boldsymbol{s}, \boldsymbol{a})} \tag{5.1}$$

Training a policy is always a stochastic procedure and as such the vector of policy parameters is a random variable itself, together with the number of state-action pairs that we choose to calculate divergence. Recalling our discussion in Section 3.2.2 on confidence levels, the number of times a policy is trained to assess its convergence rate, will also determine our confidence for the KL-divergence between the compared policies.

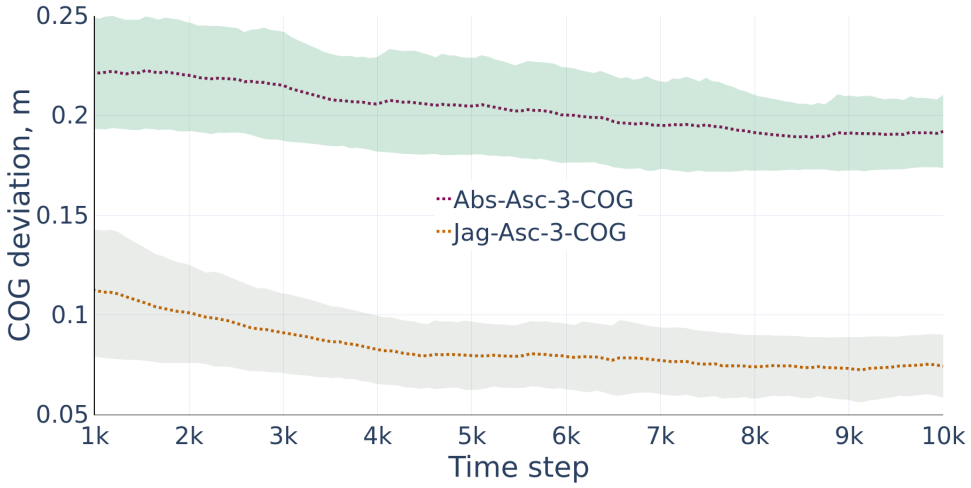
5.1.3 Learning performance in ascent tasks

Figure 5.3 (a) shows the mean smoothed episode return over independent learning trials with min-max bands, for (Jag-Asc-3-COG) and (Abs-Asc-3-COG).

Mean episode return starts from -0.23 and -0.44 for the two tasks, because of the



(a)



(b)

Figure 5.3: Ascent task learning analysis

negative penalty and the failure of the robots to advance to the goal, and converges to 0 by the end of learning for both policies. The max bands can attain the values of 0.26 and 0.28 episode return respectively. This means that the robots learn to control flipper angles and the linear velocity. The episode return for the policy (Abs-Asc-3-COG) is lower in the beginning because the Absolem COG is higher and car tip-over more easily. None of the two learning curves reaches the maximum return 1 as it is impossible to put the COG closer to the point O (see Figure 4.3) than a certain distance. This is normal since the robots cannot be entirely parallel to the ground while they traverse the staircase.

Figure 5.3 (b) shows the evolution of the COG during learning. Both policies reduce

the COG deviation at the same pace and reach minimal values after 8000 time steps, or 120 episodes. The Absolem policy exhibits higher COG deviation equal to 0.19 m due to the higher COG placement in the initial configuration where flippers are extended, still, it decreases the initial COG by 17%. The Jaguar policy decreases the COG deviation by 36% down to 0.075 m that approximately represents the COG deviation along the Y axis.

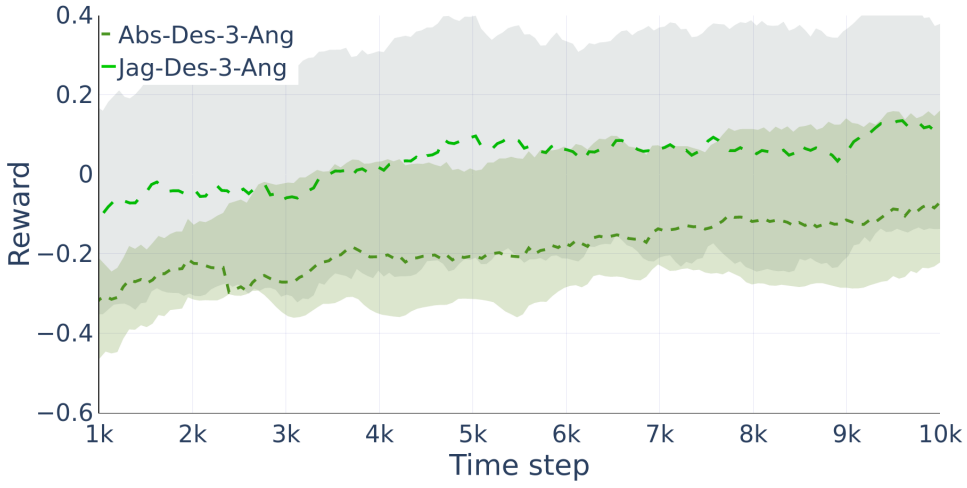
The learning process for two robots is fairly similar. Furthermore, the same value of episode return is achieved by the end of training. The COG curves exhibit similar behaviour and converge at different values, due to differences in mass distributions. Those differences also change robot control features, still both robots achieve the goal by minimizing the COG deviation. Note, however, that the absolute COG deviation difference between Absolem and Jaguar noticed at the end of learning in Figure 5.3 (b), is not transposed to an equivalent absolute difference in the total reward in Figure 5.3 (a), thanks to the normalization of each negative penalty. These results strongly indicate that the framework is applicable to different robot models for the ascent task.

5.1.4 Learning performance in descent tasks

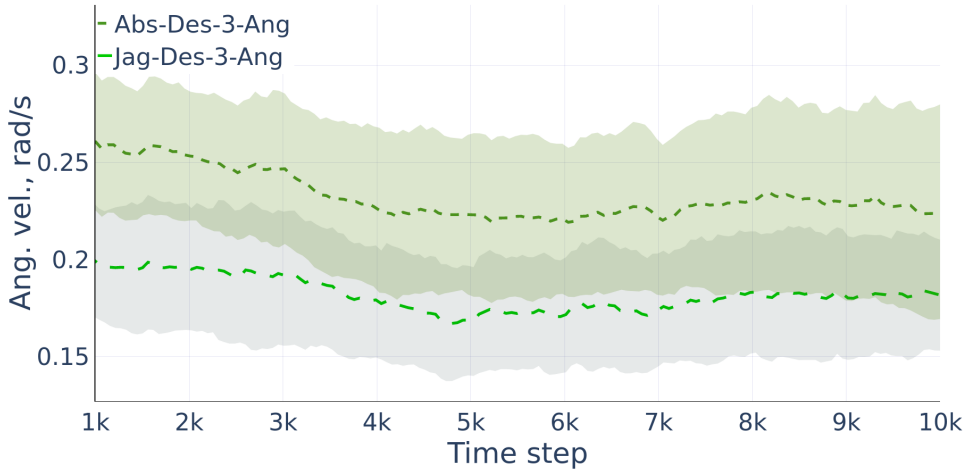
Figure 5.4 (a) presents learning curves for tasks (Jag-Des-3-Ang) and (Abs-Des-3-Ang). The firsts starts at -0.1 episode return and converges to the value 0.1. The second policy convergences similarly but starts at -0.3 episode return and ends at -0.1. Both platform policies exhibit similar learning behaviour separated by around 0.2 that can be explained by robot dynamics, because the Absolem is exposed to higher pitch angular velocity. This is seen in Figure 5.4 (b) where the mean angular velocity of a platform during an episode is higher for the Absolem, due to a higher centroid position over the surface in the resting state. The Absolem policy decreases its angular velocity by the end of learning by 13%, while the Jaguar policy drops it by 10%. Minimal angular velocity values are attained by 5000 steps, but from this moment onward the learning curves slight increase. This may happen due to optimization of traversal time as the robot spends less time in staircase traversal. As before, these results suggest that the framework is effective for both robots.

5.1.5 KL divergence between policies

Figure 5.5 presents KL divergences for learned controllers in the 3 DOF tasks. We omitted 5 DOF tasks since the simulated model of Absolem robot does not include an arm. An $(i, j)^{th}$ cell of the heat map represents the divergence between a controller with row index



(a)



(b)

Figure 5.4: Descent task learning analysis

i and a controller with column index j, namely, to $D_{KL}(\pi_i, \pi_j)$. If the KL divergence is low for two policies, then this implies that very similar actions are chosen for the same observations.

The heat map allows the extraction of some very useful insights. For example, for a given staircase traversal direction and total reward design, the two robots develop different policies. This can be particularly seen in divergences (1,4) and (2,5) and demonstrates the importance of separate controller training for the different robots. **Despite the fact that, in simulation, the robots share the same motors, the same observations and the same action space, the differences in geometry and mass distribution**

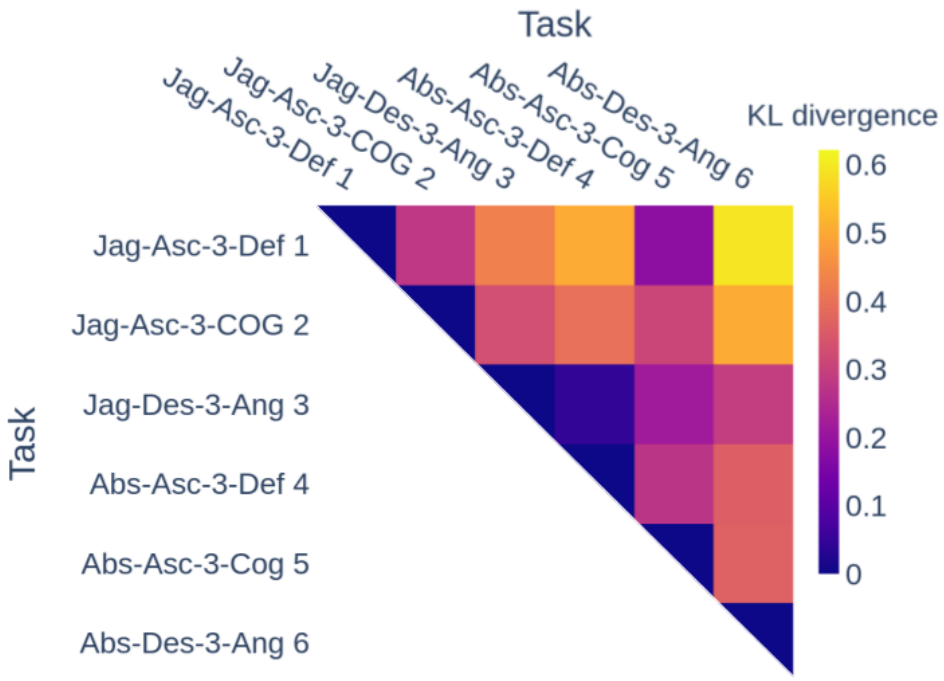


Figure 5.5: Kullback-Leibler divergence between policies

result in different necessary flipper actions throughout the traversal. It further suggests an operator that is sufficiently skilled to operate a given platform, will not be necessarily equally competent to operate a slightly different platform. We can further observe that for a given robot and traversal direction, alternating the total reward function also induces a quantitative change in the employed actions, for example in divergences (1,2) and (4,5). This justifies our interest for optimizing staircase negotiation with more elaborate criteria than merely arriving to the goal.

Finally, the highest divergences are noticed between policies where the staircase traversal is different, namely for (1,6) and (2,6). This clearly suggests that staircase ascent and descent require independent treatment, as dynamics and risks significantly differ. Two pairs of policies were found unexpectedly similar however, i.e. (1,5) and (3,4), an event that can be attributed to the stochasticity of training.

We finally performed a quantitative analysis of policy adequacy on different platforms provided in Figure 5.6. This experiment is destined to assess how well a policy trained on one robot would perform when deployed to another robot. We performed 10 such trials (rollouts) for a given task. Policies (Jag-Asc-3-Def) and (Abs-Asc-3-Def) are evaluated on the basis of time steps spent in a rollout. As we can see, if we deploy a policy on a

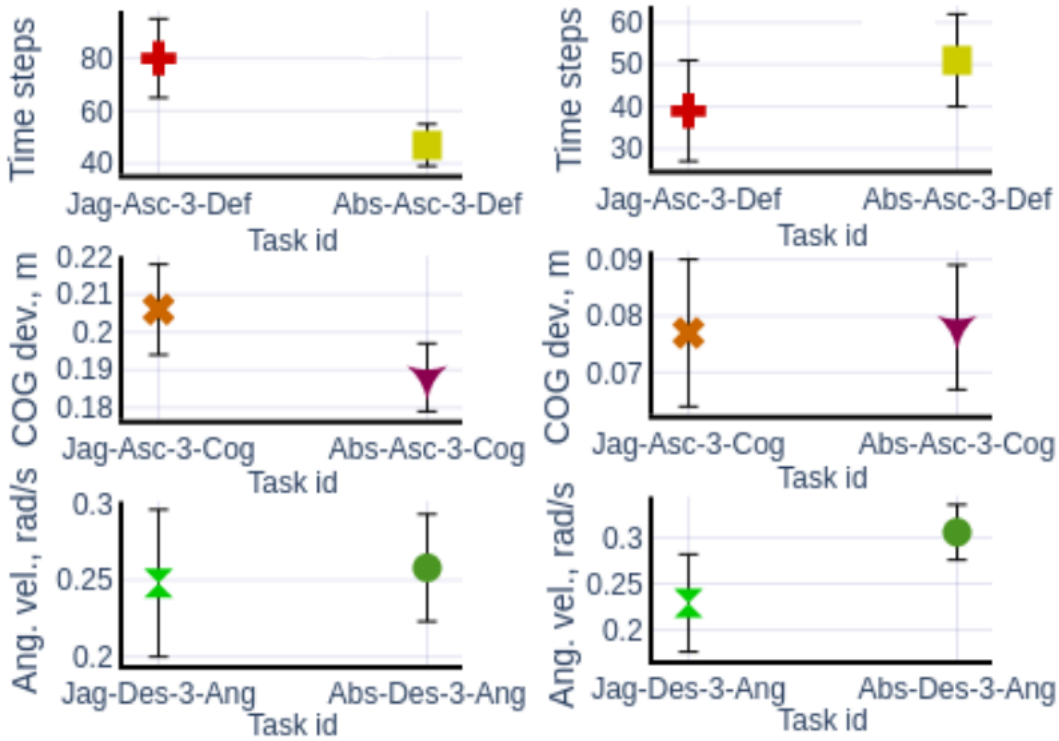


Figure 5.6: Performance evaluation of policies when deployed on Absolem (left) and Jaguar (right) robots.

platform for which it was trained, then the robot spends less episodes until task completion. Characteristically, rollous tested on Absolem last around 47 time steps if we use the (Abs-Asc-3-Def) policy against 80 time steps if (Jag-Asc-3-Def) policy is used where performance is significantly degraded. Similarly, deployment of the (Jag-Asc-3-Def) on Jaguar requires on average 39 time steps as opposed to 51 when the (Abs-Asc-3-Def) policy is applied.

Deployment of (Jag-Asc-3-Cog) on Absolem further reveals a slight performance degradation in terms of COG deviation measured at 0.206 against 0.188 when (Abs-Asc-3-Cog) is deployed. In contrast, the deployment of these two policies on Jaguar seems equally performant.

With respect to descent, the deployment of the (Jag-Des-3-Ang) and (Abs-Des-3-Ang) policies on the Jaguar robot clearly favors the robot specific policy, but the same policies perform equally well on Absolem. We can conclude that a policy trained and deployed on the same robot provides the best performance, otherwise performance can only be degraded.

5.2 Policy transfer

5.2.1 Platform description

In this section we describe the commercial platform **Jaguar V4 with Manipulator Arm**[4], that we refer as *Jaguar*, used for the deployment of the trained policies. An overview look is given in Figure 5.7, Table 5.3 shows its physical characteristics. *Jaguar* is a ground actively articulated tracked robot with dimensions $0.98x0.7x0.4m$ (manipulator at rest), manipulator reach is $0.707m$, and its mass is $41kg$ with the maximum payload $4kg$. The robot is equipped with GPS and 9 DOF IMU, which includes a gyroscope, accelerometer and compass for autonomous navigation. The real robot arm has 4 DOFs, however, only 2 DOFs, related to joints 1 and 2, changes the COG position.

The velocity-controlled main tracks are attached to the chassis and fixed. Each track has a flipper on both ends. Front and rear flippers are coupled and positional-controlled. They can continuously rotate, however, we programmatically provide limits ($[-\pi/4, \pi/4]$ from the "extended configuration") to avoid potentially dangerous states leading to high currency and burining out of motors. The robot manipulator Figure 5.7 (b) has the same limits $[-\pi/4, \pi/4]$ to avoid self-collision and, also, position-controlled. According to the specification, the robot can climb obstacles with the height up to $0.3m$.

5.2.2 System description

We begin by presenting the experimentation set-up used to obtain the ground-truth robot and environment state. This allows to establish an upper bound on the effectiveness of a controller in reality with moderate state errors while allowing the development and evaluation of alternative robot localization approaches later on. On the other hand, we opted for injecting noise in the state during policy learning in simulation to make it more robust to errors in general as well as to reduce the risk of accidents.

[4] `jaguar.drrobot.com/specification_V4Arm.asp`

Parameter	Mass, kg	Length, m	Width, m	Height, m
Value	41	0.98	0.7	0.4

Table 5.3: *Jaguar* hardware description

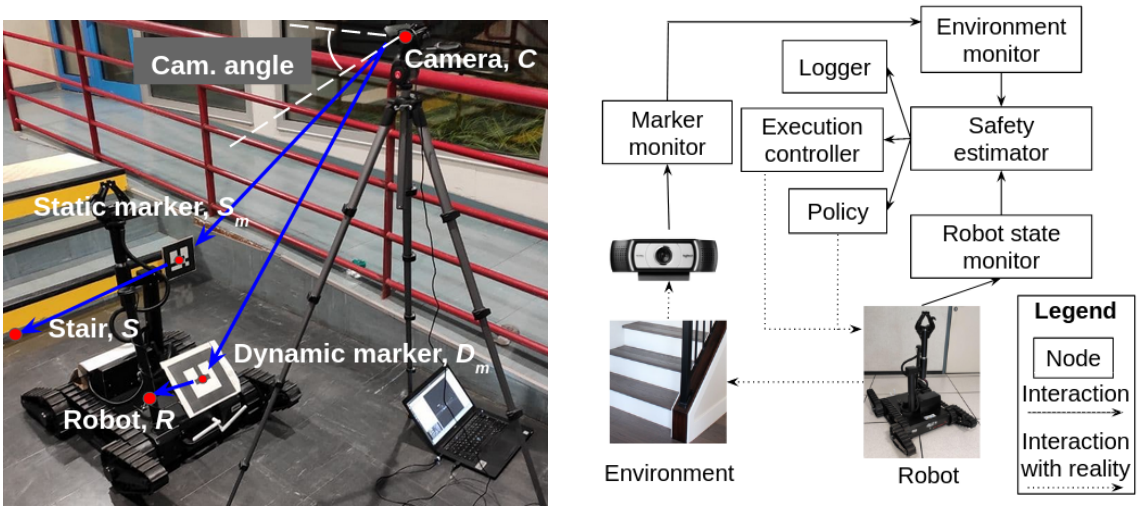
(a) (b)

(c) (d)

Figure 5.7: *Jaguar* in the "extended" configuration. Front view (a), top view (b), side view (c), "in-action" view (d)

To obtain the robot pose we employed visual markers whose pose can be reliably obtained via calibrated cameras and standard toolkits[5][6]. We distinguish markers into *static* and *dynamic* with associated coordinate frames S_m and D_m (see Figure 5.8). We use the notation ${}^{j}\boldsymbol{T}_{i}$ for the transformation between coordinate frames, denoting translation and rotation of the coordinate frame i within the coordinate frame j. We can then obtain transformations ${}^{C}\boldsymbol{T}_{S_m}$ and ${}^{S}\boldsymbol{T}_{S_m}$, the latter being measured manually. The dynamic marker is placed onto the robot and used to locate its base coordinate frame with respect to camera C, via a constant transformation ${}^{R}\boldsymbol{T}_{D_m}$ set manually.

[5] `http://wiki.ros.org/aruco_detect`
[6] `http://wiki.ros.org/camera_calibration`



(a) (b)

Figure 5.8: (a) Experimentation setup along with coordinate frame hierarchy; (b) architecture of deployed robot system

The camera was placed in front of the operational space of the robot, ensuring proper coverage of the robot as well as the staircase. Its purpose is to associate the robot and the staircase into a common hierarchy of coordinate frames, allowing at any moment to retrieve the coordinates of the front and rear step edges with respect to the robot. The staircase is then represented as a series of static transformations ${}^{S_i}\boldsymbol{T}_S$ where S_i is the coordinate frame associated to a step edge.

We note that the staircase perception task could also be performed by a robot (either by dynamic [120] or by static environment cures [121]) using its on-board sensors but we refrain to do so to keep our set-up generic and independent of the robot. Perception of a staircase before descent is more challenging yet this problem resides out of the scope of this work.

5.2.3 Policy deployment from simulation to reality

In the beginning of an experiment the robot faces the staircase and the controller is activated upon the approaching of front and rear step edges. For the task of staircase negotiation, it turns out that only a subspace of the entire action space spanned by the flipper and arm joints is useful. This is something that we take into account by constraining the action space from which actions can be sampled. As a direct consequence of the fact

that joints are allowed to move in a significantly smaller space, this provides sufficient time to perform a necessary action and adaptation of low-level controllers from the simulated platform to the real one and makes this transfer zero-shot. Crucially, constraining the action space further serves in ensuring safety of the platform, since learning is stochastic and non-previously encountered conditions may lead to accidents. For example, overly raising rear flippers while ascending or front flippers while descending, combined with acceleration can provoke a tip-over. To prevent such behaviours, the limits of rear flippers while ascending and front flippers while descending were set as $\psi_s^{rear}, \psi_s^{front} \in [-\pi/4, 0]^2$ respectively.

The deployed system architecture is shown in Figure 5.8 (b). The *Marker monitor* detects markers in the camera image. The *Environment monitor* builds and maintains the geometric relationships between the staircase and the robot, within a single transformation hierarchy and provides the front-most and rearmost step coordinates. The *Robot state monitor* provides information about the robot flippers and arm configuration, its velocity and IMU data. The *Safety estimator* receives the output of *Environment monitor* and *Robot state monitor*, evaluates safety metrics such as the COG deviation and angular velocity and its output is concatenated with data provided by *Environment monitor*. That output is fed to *Execution controller* which decides whether to block motors in the case of upcoming accidents or halt the system when the experiment terminates. Otherwise, state output data pass to *Policy* which samples the action vector $\boldsymbol{a}$ which is sent to the robot actuators. The latter executes those actions and *Marker monitor* observes changes in the environment. Finally, the output of *Safety estimator* is logged via *Logger*.

5.2.4 Real-world performance

Policies (Jag-Asc-5-COG) and (Jag-Des-5-Ang) were tested in two staircases presented in Table 5.4 (we recall that the arm is actively involved in the optimized controllers). The respective experiments are included in the supplementary video[7]. Overall, we have performed 10 trials for each policy id (5 trials per staircase) discussed in this section, all of which were successful.

Policy (Jag-Asc-5-COG) was tested on the big staircase, the robot achieving the task by keeping the rear flippers pushed down, the front flippers up, the first arm joint was inclined clockwise and the second arm joint counter-clockwise. The flipper configuration

[7] `partage.imt.fr/index.php/s/LDyp6QRp4nGGKd2/download`

Name	**Number of steps**	**Height**	**Length**
Big	5	0.195	0.275
Small	3	0.17	0.305

Table 5.4: Staircases configurations

Policy id	**Stair**	Cy, m	D, m	Ang. vel., rad/s
Jag-Asc-5-COG	Big	0.11 ± 0.01	0.12 ± 0.01	0.33 ± 0.06
Jag-Asc-5-COG	Small	0.1 ± 0.01	0.11 ± 0.01	0.36 ± 0.09
Jag-Des-5-Ang	Big	0.16 ± 0.02	0.09 ± 0.02	0.23 ± 0.06
Jag-Des-5-Ang	Small	0.12 ± 0.02	0.05 ± 0.02	0.23 ± 0.09

Table 5.5: Observed mean target values on real staircases

ensures that the robot has the contact points with rear and front stair steps. The entire arm is moved forward to decrease COG deviation, which minimizes tip-over risks or getting stuck. A point of ambiguity emerges for the arm as certain configurations decrease C_y and increase C_x at the same time, or vice versa, hence minimizing COG deviation D is not straightforward. Remarkably, the obtained results of the learnt controller show that the robot learnt to incline forward the first arm link and backward the second link, which is indeed the best configuration. For reference, Table 5.5 provides average observed values for some key parameters.

The accomplishment of the descent task (Jag-Des-5-Ang) on the big staircase merits more attention. One of the most important aspects is the linear velocity control (see provided video), where the robot moves smoothly and adapts its velocity very attentively that leads to increased time in the staircase traversal. Naturally, this greatly reduces the mean perceived angular velocity by 0.1 rad/s, as opposed to the ascent task.

The ascent and descent tasks were further successfully accomplished on the small staircase composed of 3 steps, each of height 0.17 m and length 0.305 m. The values reported in Table 5.5 allow us to draw the same conclusions between ascent and descent, as for the big staircase.

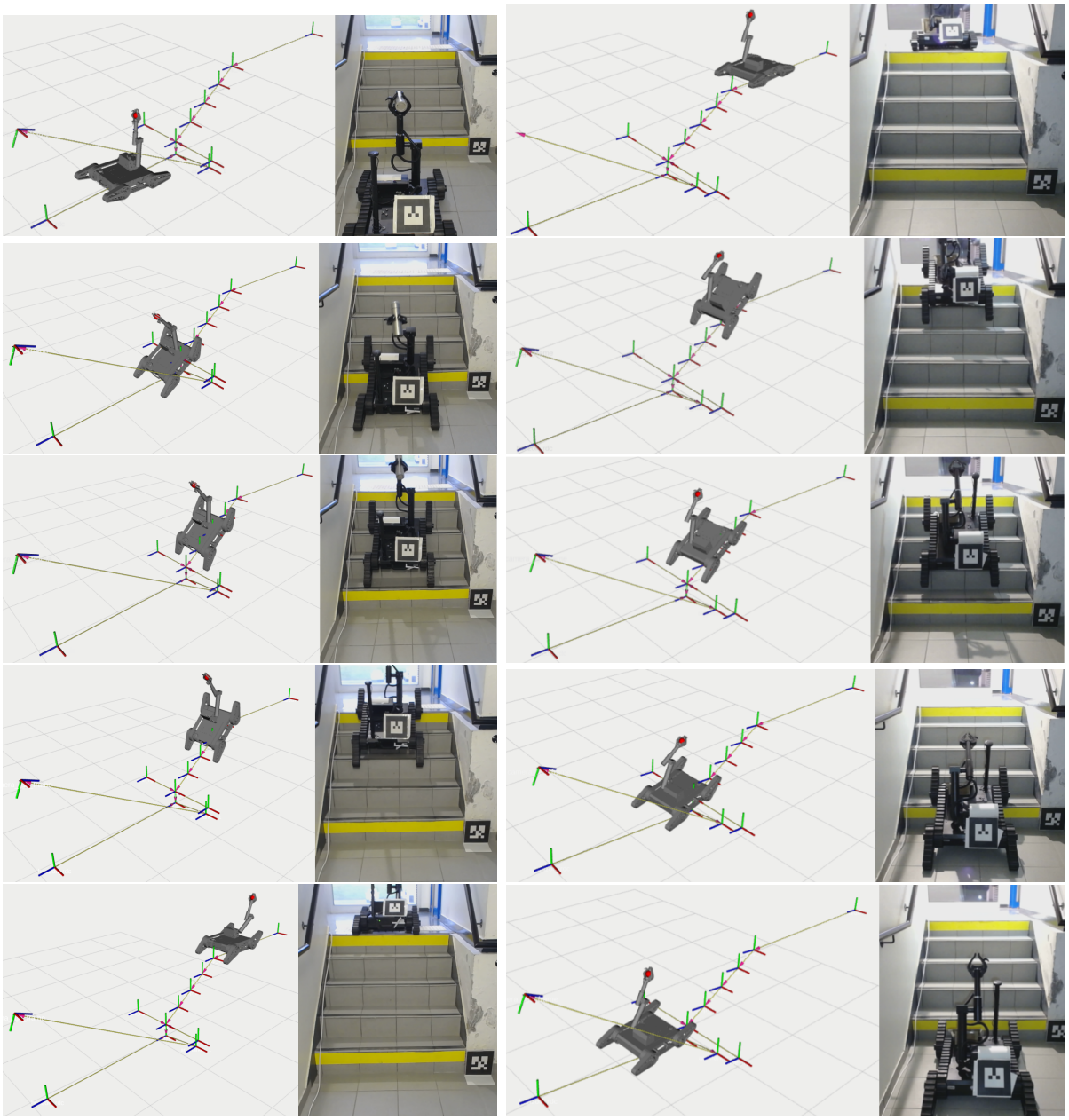


Figure 5.9: Snapshots of staircase negotiation with congruent control of 5 DOF. Left column; ascent by minimizing COG deviation. Right column; descent by minimizing pitch angular velocity.

5.3 Summary

We have presented an effective RL framework for control learning of staircase negotiation in multiple task variants. In particular, we trained controllers for two robotic platforms in simulation with reward function designs suited for ascent and descent and different staircases. Results show learning convergence and optimization of targeted behaviour features for both platforms within 150 episodes. We employed KL divergence to quantitatively compare the obtained policies, allowing us to consolidate earlier empirical findings.

Most importantly, we succeeded in deploying the controllers learned in simulation to a commercial robotic platform in firstly encountered real-world staircases. The robot was able to successfully ascend and descend while respecting the underlying criteria.

Chapter 6

Incremental Domain Randomization Learning Framework

At the intersection of the contributions presented in the previous chapters, we unfold in this chapter the architectural details of the developed software that binds together learning, simulation and evaluation into a single entity. Recalling Chapter 2, there exists a small number of frameworks for robot control learning in 3D environments, none of which seems to allow domain randomization for generating diverse staircases. That motivated us to create and share a generic an incremental domain randomization learning framework in order to facilitate future research on this topic.

In addition, there is no publicly available software framework that addresses robot safety by simulating control learning in the case of ascent and descent of staircases for tracked robots, optionally equipped with an arm. This is combined with a lack of common utilities for development and comparison of tracked robot controllers, which constrains researchers to develop custom robot models from scratch.

In view of these shortcomings, our goal in to provide a software framework (see Figure 6.1) [122] (under revision) that goes beyond the current state-of-the-art in the following directions :

- Development of a simulation framework and pipeline for 2D navigation combined with staircase negotiation using RL for articulated tracked robots.
- Provision of two trainable robot models.
- Examples of successful control learning for staircase ascent and descent, showing the clear benefits of incremental domain randomization over conventional uniform domain randomization.

The remainder of the Chapter is organized as follows. Section 6.1 discusses the intention of the developed framework. Section 6.2 provides a formal description of the RL problem

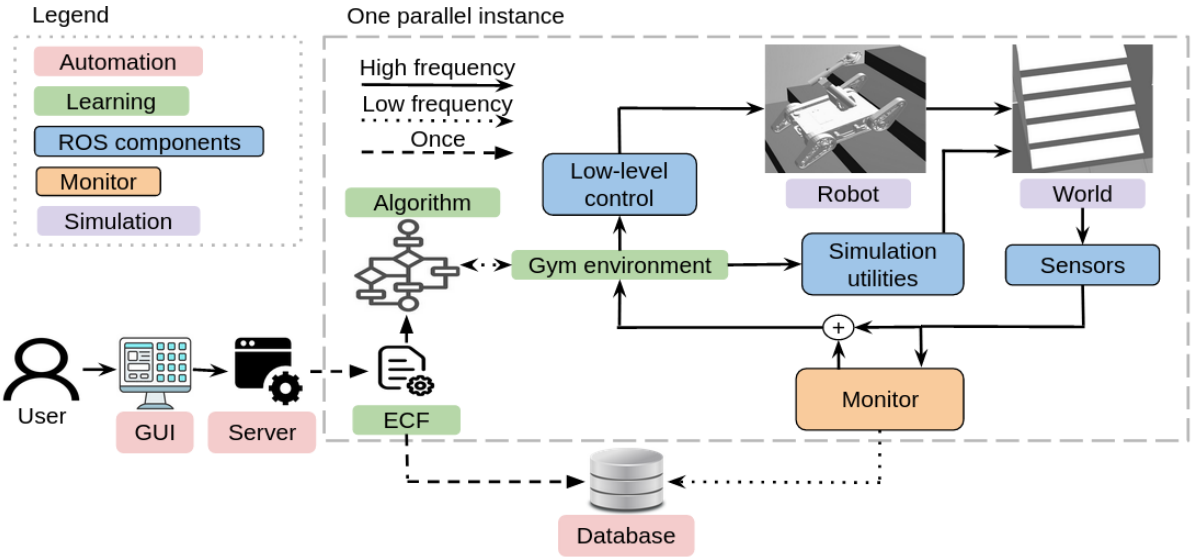


Figure 6.1: Global overview of the proposed software framework

that is treated by our framework. Section 6.3 presents all framework components allowing potential users to experiment with alternative ways of learning control for tracked robots in indoor 3D environments. Finally, in Section 6.4 we show its application on control learning for staircase ascent and descent traversals.

6.1 Objective of the framework

We propose a framework which consists of a ready-to-use simulation environment that can be used for control learning of articulated tracked robots via RL. The framework is customizable, allowing the user to control the environment and adapt the evaluation accordingly. It is built upon and extends our earlier works [116], [45] and [117] that allowed learning in simulation and transferring of policies onto a real commercial robot. Our framework supports out-of-the-box NNs, although other types of controllers can be used since they do not interfere with framework components. To solve a control learning problem, it applies RL along with contemporary algorithms implemented in [110].

In an indoor navigation setting , the control learning problem can be distinguished in three principal tasks: **2D navigation, ascent and descent traversals of a staircase**. In each of them, the learning objective is to move the robot to a goal, namely, a point in space, avoiding ground obstacles on the floor or negotiating a staircase. The user can then impose additional constraints as secondary goals, which can be optimized jointly with goal reaching.

6.2 General problem statement

To address the problem of 2D, ascent and descent navigation, the framework contributes a RL episode life-cycle similar to that used in Chapters 3 and 4. We have already introduced the notion of the action and state spaces, yet, we present in this section a more general problem formalization for the case when the robot relies directly on its camera sensor. When using the framework, the user can then choose what parts of state and action vectors he(she) wants to adopt.

We assume that to navigate within an indoor environment a robot has access to the following state:

$$\boldsymbol{s} = (v_s, w_s, \psi_s^{front\_left}, \psi_s^{front\_right}, \psi_s^{rear\_left}, \psi_s^{rear\_right}, \phi_s^1, \phi_s^2, \theta_p, \theta_r, o_1, ..., o_{N_f}) \quad (6.1)$$

where v_s and w_s represent state linear and angular velocities, $\psi_s^{front\_left}$, $\psi_s^{front\_right}$, $\psi_s^{rear\_left}$, $\psi_s^{rear\_right}$, ϕ_s^1 and ϕ_s^2 flipper and arm joint angles (see Figure 4.2), where rotation of robot links is considered around red x-axes, the arm link 2 rotates around the green y-axis (see Figure 4.4 (d)), θ_p and θ_r are pitch and roll angles of the platform chassis, $o_1, ..., o_{N_f}$ are N_f environment depth features. Note that this general form of the state vector can be altered by the user through an experiment configuration file (ECF), for example, in a case where the robot does not require angular velocity or arm control.

The control vector $\boldsymbol{a}$ is composed of commands that can be sent to the robot as follows:

$$\boldsymbol{a} = (v_a, w_a, \psi_a^{front\_left}, \psi_a^{front\_right}, \psi_a^{rear\_left}, \psi_a^{rear\_right}, \phi_a^1, \phi_a^2) \quad (6.2)$$

where v_a and w_a are linear and angular velocity commands, $\psi_a^{front\_left}$, $\psi_a^{front\_right}$, $\psi_a^{rear\_left}$ and $\psi_a^{rear\_right}$ are flipper rotation angles, ϕ_a^1 and ϕ_a^2 are arm control angles. As in the case of the state vector, the usage of the control vector components can be controlled through the ECF and connected to used robot parts, as will be presented in detail in the next section.

6.3 Framework architecture

This section presents the framework and its components, unfolds its workflow from its most low-level parts such as the simulator to the high-level ones such as RL algorithm employment and automation tools. It serves as a guide for the manual modification of inner components to integrate new algorithms, sensors and experiment workflows. Hereafter, we present the simulation environment, the robot command dispatching and perception utilities and the learning environment which drives the learning process. Finally, we discuss algorithm and library integration utilities. We recommend to the reader to use Figure 6.1 as a basis and a guide for keeping track of the various notions that will be presented along with their interactions.

World Our framework is based on the open source physics-based Gazebo simulator which resides at the core of our system. This simulator is widely used and popular within the robotics community and already contains multiple functionalities for control of simulated robot models. *World* accepts requests from *simulation utilities* which will be described next Section 6.3.1 and spawns a corresponding 3D simulated environment for training. *Robot* operates in this environment and provides the output to *Sensors* 6.3.1.

Robot The framework operates with a simulated tracked robot model in the unified robotic description format (URDF[1]) which consists of a body, front and rear flippers and that can be equipped with an arm, whose mass and geometry can be set to match those of a real robot. Interaction between tracks and staircase surface is performed by adopting the CSM model [35]. Front and rear flippers can be jointly or separately controlled. This component receives commands from *Low-level control* 6.3.1.

Simulated *Jaguar* (see Figure 4.4 and Table 4.1) and *Absolem* (see Figure 5.2 and Table 5.1) can be trained or operated in simulation. Each robot possesses at least 4 DOFs which consist of linear velocity, angular velocity, front and rear flipper angles. This means that the pair of front flippers is controlled by only DOF, and similarly for the pair of rear flippers. In the case of separate flipper control, robots have 6 DOFs while if an arm is also present and controlled, the total number of DOF raises to 8. These robots possess similar negotiating capabilities. The user can add the payload to the end-effector of *Jaguar* to enhance resilience to the payload influence.

[1] `https://wiki.ros.org/urdf/`

6.3.1 ROS components

ROS (Robot Operating System) [123] is open source software under a BSD license. It helps to develop software for robot applications through provision of commonly used libraries and tools. They consist of device drivers, hardware abstraction, visualizers, generic libraries, package management, message-passing, and more. The following components of our framework are developed using ROS standards.

Simulation utilities

By careful parameterization of the source domain in simulation, i.e. the definition of Ξ and its samples $\boldsymbol{\xi}$ (see Chapter 3, Section 3.1), we seek to generate a sufficiently rich and representative set of situations for the three main tasks of indoor navigation corresponding to 2D navigation, staircase ascent and descent whose environments are presented below. *Simulation utilities* accept commands from *Gym environment* 6.3.3, produce models that are usable within *World* and requests the latter to load them.

Domain Randomization for 2D navigation In indoor environments which are organized in piece-wise orthogonal configurations (see *Manhattan world* assumption [124]), the robot has to ordinarily traverse hallways or perform more complex zigzag navigation on a 2D ground, as shown in Figure 6.2 (a) and (b). D, W, L are constant environment parameters which define the size/scale of the environment, the width and the length of the obstacle area. By varying obstacle parameters within the constant environment, the framework forms an environment configuration vector $\boldsymbol{\xi} = (W_1, L_1, W_2, L_2, C)$, where $W_i \in [\frac{W}{2} - \frac{C}{2}, W - C]$ and $L_i \in [L - W, L]$ are the width and length of obstacles 1 and 2 respectively and $C \in [C_{min}, C_{max}]$ is the minimum distance between obstacles.

The framework generates with equal probability a hallway or a zigzag environment. In the first case, L_1 and L_2 are equal L, C and W_1 are uniformly sampled within their limits, W_2 is sampled from $[0, L - W_1 - C]$. In the second case, we sample $\boldsymbol{\xi}$ from defined uniform distributions.

The user defines in the experiment configuration file (ECF) whether goal and robot spawning is random or fixed. The appearance at random can be seen as a part of domain randomization enhancement where we vary unseen situations through experiment episode initialization. To spawn a robot and the goal, the framework provides 2 separated ROS[2] services. In the case of random goal appearing, the goal can appear either in the rectangle

[2] `http://wiki.ros.org/Documentation`

of *random goal spawning area* (see Figure 6.2 (a)) or at its geometrical center in the fixed goal appearing situation. The framework puts the robot on *spawning line* either at a random point on the same line and random orientation to the goal or at the center of the spawning line with perpendicular orientation.

Domain Randomization for staircase negotiation Two other tasks involved in indoor navigation concern staircase ascent and descent making a staircase generation an important feature of our framework. A straight staircase can be represented by the step length L_{step}, the step height H_{step} and the number of steps N_{step}, i.e. by an environment configuration $\boldsymbol{\xi} = (L_{step}, H_{step}, N_{step})$.

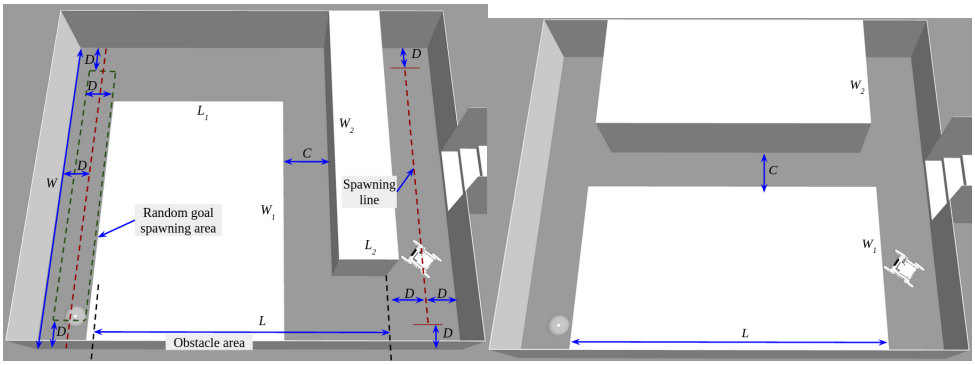
We enable two types of staircase generation. The first one assumes $\boldsymbol{\xi}$ is distributed uniformly to generate samples and we term as *uniform* environment. The second type assumes that $\boldsymbol{\xi}$ follows a normal distribution with a diagonal covariance matrix. The corresponding mean and covariance matrix depend on a parameter $\epsilon \in [0, 1]$ which regulates the complexity (and in turn the difficulty), of the generated staircase. We term environments produced with this technique as *incremental* Gaussian environments.

Equations 6.3a and 6.3b present sampling of the environment configuration $\boldsymbol{\xi}$, where $\boldsymbol{\Delta} = \boldsymbol{\xi}_{max} - \boldsymbol{\xi}_{min}$:

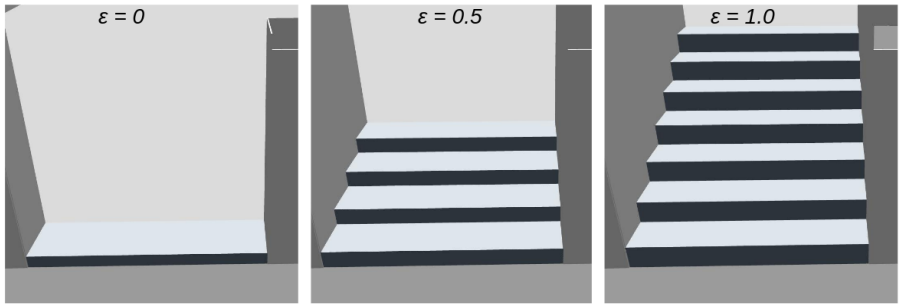
$$\boldsymbol{\xi} \leftarrow \mathcal{N}(\boldsymbol{\xi}_{min} + \epsilon \cdot \boldsymbol{\Delta}, \epsilon \cdot \mathrm{diag}(\boldsymbol{\Delta}_1, \boldsymbol{\Delta}_2, \boldsymbol{\Delta}_3)) \tag{6.3a}$$

$$\boldsymbol{\xi} = \begin{cases} \boldsymbol{\xi}_{max}, & \text{if } \boldsymbol{\xi} > \boldsymbol{\xi}_{max} \\ \boldsymbol{\xi}, & \text{if } \boldsymbol{\xi}_{min} \leq \boldsymbol{\xi} \leq \boldsymbol{\xi}_{max} \\ \boldsymbol{\xi}_{min}, & \text{otherwise} \end{cases} \tag{6.3b}$$

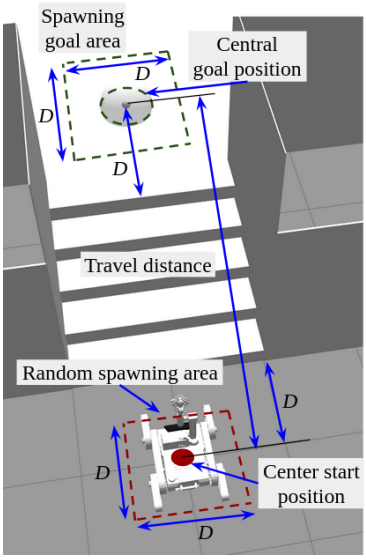
where $\boldsymbol{\xi}$ is a candidate configuration, $\leftarrow \mathcal{N}(\cdot, \cdot)$ indicates random sampling from a normal distribution, $\boldsymbol{\xi}_{min}$ and $\boldsymbol{\xi}_{max}$ represent minimum and maximum admissible environment configurations. Equation 6.3a indicates that we sample a candidate experiment configuration from the normal distribution with the mean $\boldsymbol{\xi}_{min} + \epsilon \cdot \boldsymbol{\Delta}$ and the covariance matrix $\epsilon \cdot \mathrm{diag}(\boldsymbol{\Delta}_1, \boldsymbol{\Delta}_2, \boldsymbol{\Delta}_3)$, where ϵ enables to increase the mean and covariance and is automatically reset according to learning progress that we define as mean positive episode reward over last N_ϵ episodes which is configured at the episode beginning. The value of N_ϵ was empirically determined allowing to obtain superior performance compared to non-incremental domain randomization. This leaves space for further performance gains if this point is addressed more thoroughly by the users of the framework. Equation 6.3b



(a) (b)



(c)



(d)

Figure 6.2: Illustration of zigzag (a) and hallway (b) environments, complexity increment of the staircase environment (c), ascent task environment (d)

clips $\boldsymbol{\xi}$, so that it fits limits.

Figure 6.2 (c) presents staircases sampled for 3 different ϵ values. As in the case of 2D environment, the user selects whether goal and/or robot are spawned randomly or at fixed preset positions. In ascent, when random spawning is chosen, the robot can appear at random position and orientation at *Random spawning area* (see Figure 6.2 (d)) or at *center start position* being aligned with the staircase in fixed spawning. The goal can be spawned in the same way either at *spawning goal area* or at *central goal position*. The goal and robot spawning areas are reversed in descent.

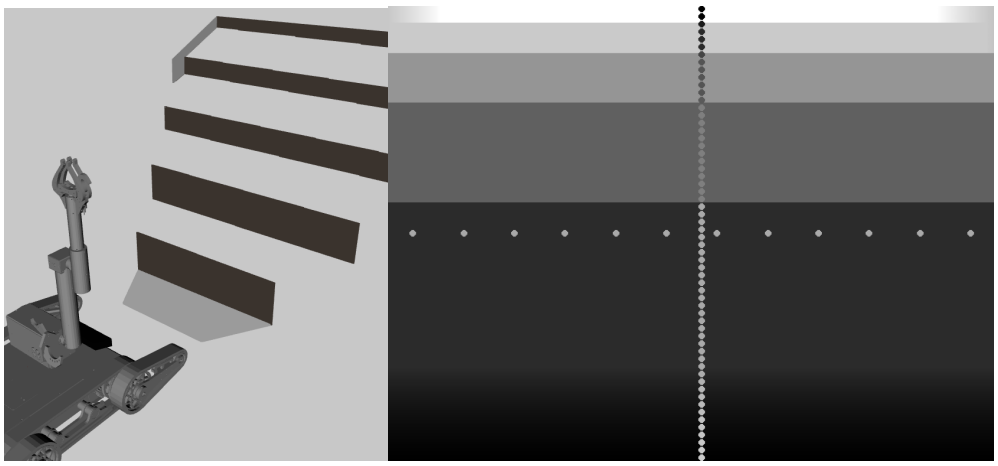
Low-level control

The framework provides control software which accepts a sole ROS-based message issued from *Gym environment* for controlling the entire action space of the robot, handles and dispatches it to corresponding ROS and Gazebo low-level controllers and further provides feedback through a message providing information about current linear and angular robot velocity and joint configuration state. Then, flipper and arm joint rotation angles are limited to reflect real robot operation. A spawned robot can be operated with a keyboard and a ROS message.

Sensors

Another central infrastructure component of our framework is related to perception. Using an RGB-D sensor to perceive the environment within *World*, the framework provides an elementary depth-based feature extractor and dispatches its calculations down to *Gym environment* and *Monitor*. We have preferred to rely on the depth image due to the poor realism of simulated RGB images in the Gazebo simulator. The robot facing a staircase and its depth perception is presented in Figure 6.3 (a) while Figure 6.3 (b) shows the associated features whose coloring is reversed to avoid melding with the background.

Borrowing the idea of feature extraction from [22], we convert every depth image into vertical and horizontal beam groups. The user can choose the cardinality of horizontal and vertical beams after which the framework calculates beams positions and averages non *NaN*, i.e. valid, depth image pixel values around them within an area predefined by a user. As it was shown in [22], 10 horizontal beams are sufficient to learn map-less 2D navigation, but we opt for using more vertical beams to better retrieve the structure of the staircase in ascent and descent tasks.



(a) (b)

Figure 6.3: (a) Robot in front of a staircase, (b) horizontal and vertical extracted features

The robot further contains a simulated inertial measurement unit, the output of which is published within the ROS ecosystem and used in forming of the observation vector and reward calculation. Alongside, the framework further provides the ground truth pose of the robot.

6.3.2 Monitor

Rewards and episode termination signals are issued from *Monitor* (see Figure 6.1) whose importance was particularly highlighted in our earlier work [117]. These calculations are produced with help of *Sensors* output and, being concatenated with image features, goes down to the *Gym environment*. An appropriate reward function drives learning towards the acquisition of complex behaviours and a well implemented termination signal can boost learning. As part of the framework and baseline methodology, we provide the exact same set of three reward functions (see Section 4.1.3) which can be used to learn ascent and descent staircase traversals, that could serve as a starting point for future research and bench-marking.

Finally, the component *Monitor* receives data from the most of ROS utilities to provide an adequate guidance of the learning process. It monitors the robot-goal distance, provides safety estimation, which monitors COG deviation and pitch angular velocity of the platform, forms termination signals and sends data to a *Database*. We consider three cases where *Monitor* triggers a termination signal, namely: (i) tipping over, (ii) exceeding

```
{
  "alg":"SAC",
  "angular":false,
  "arm":true,
  "complexity":"full",
  "env_type":"vect",
  "experiment":"asc_inc_cog",
  "experiment_series":"exp_paper",
  "load_path":"data/models/server/nothing",
  "log_path":"data/boards/server",
  "model_parameters":{
    "ppo_cliprange":"0.2",
    "ppo_ent_coef":"0.005",
    "sac_ent_coef":"auto_0.5",
    "sac_tau":0.05,
    "sac_train_freq":1
  },
  "penalty_angular":false,
  "penalty_deviation":true,
  "policy":"default",
  "rand":false,
  "save_path":"data/models/server",
  "sigma":0,
  "task":"ascent",
  "time_step_limit":50,
  "total_timesteps":20000
}
```


Figure 6.4: ECF example

episode time steps and (iii) reaching the goal. Safety estimation is performed for staircase ascent when the component continuously monitors mean time step center of gravity deviation and mean time step pitch angular velocity for descent.

6.3.3 Learning

This subsection unfolds the details of the *learning* components of the framework, as shown in the beginning in Figure 6.1. Our framework allows to integrate different implementations of reinforcement learning algorithms as long as they support the OpenAI Gym [125] which is a standard toolkit and operates as an interface between a RL algorithm and an agent environment. The key component *Learning* of our framework encompasses RL algorithms instantiation, an OpenAI Gym implementation and their ECF.

ECF Before every experiment, the user can instantiate an *ECF* (see Figure 6.4) which is received from *Server* and contains all necessary details of the experiment loaded by *Algorithm* and saved by *database*. File templates can vary for integration of custom experiments and RL libraries. Finally, the ECF is used for policy testing saving robot performance such as travelled distance and safety measures. **Algorithm** As soon as an experience starts

ECF is consumed by *Algorithm*. The latter selects a library and a corresponding RL algorithm with defined parameters. This also instantiates *Gym environment* after which policy training is triggered.

Gym environment Our framework integrates an OpenAI Gym environment which receives signals and calculations of *Monitor*, provides experiment data back to *Algorithm*, requests *Simulation utilities* for a new environment and, finally, robot and goal respawning controlling *Robot* through *Low-level control*. *Gym environment* dispatches rewards on every time step to *Algorithm* as well as episode termination signals received from *Monitor*. It forms the observation vector (cf. eq. (6.1)), which is passed to the policy π_θ, obtains an action vector and sends it to *Low-level control*. Finally, *Gym environment* receives the output from *Monitor* related to time step reward and termination signals, based on which the RL algorithm continues to perform policy optimization if required and the cycle repeats.

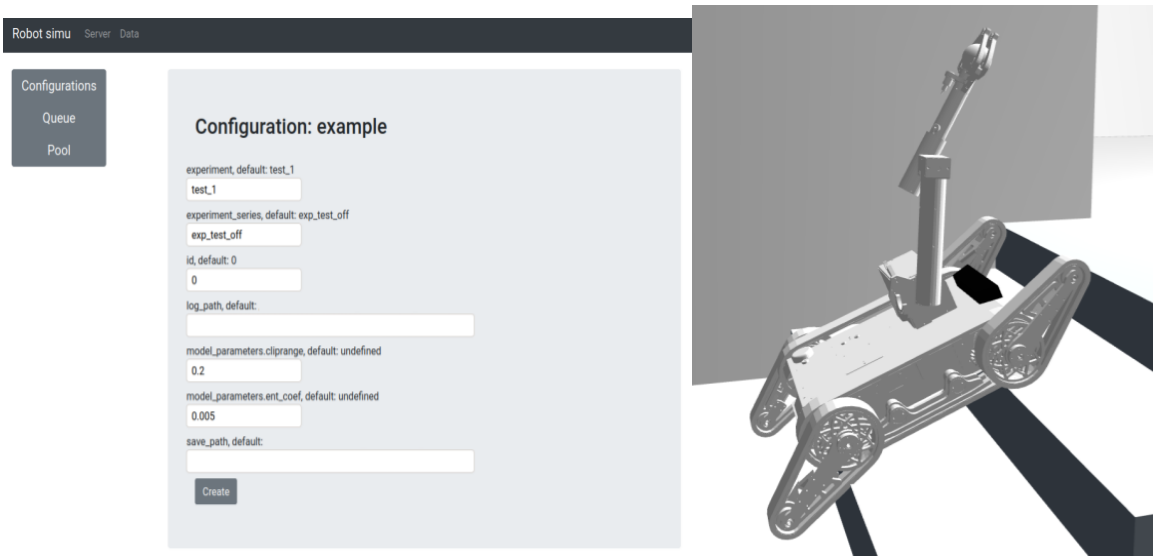
6.3.4 Automation

Launching of all aforementioned components, creating the *ECF* and data saving can be manually performed by the user. However, to simplify interaction with the environment we provide certain *Automation* tools, listed below.

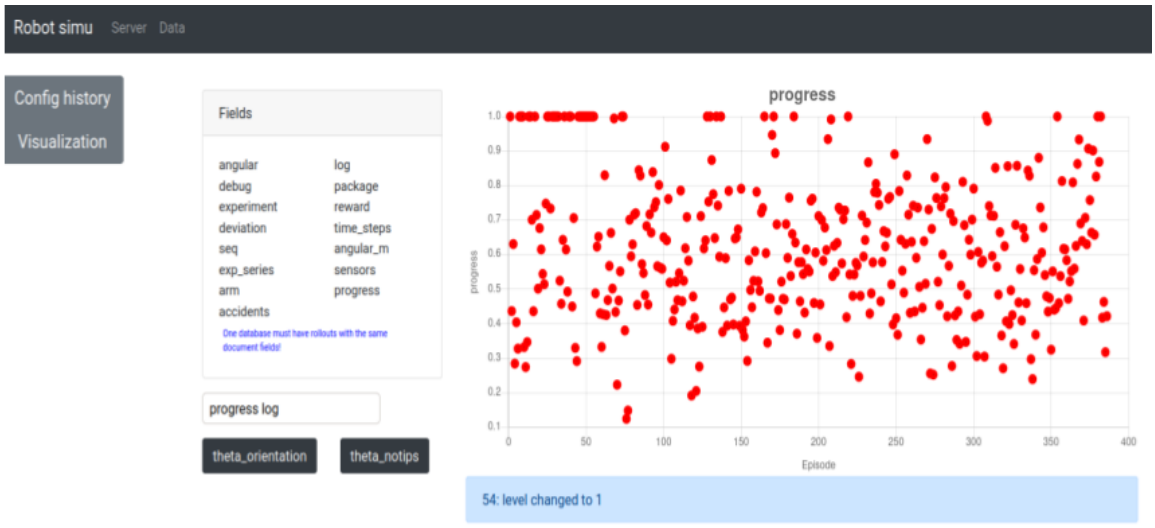
Graphical user interface (GUI) This component enables managing of experiments and visualization of results through sending commands to *Server*. Figure 6.5 (a) and (c) shows screenshots of its windows, the first one presenting the creation of a *Configuration* file and the second a visualization of ongoing learning results.

Server This is a light-weight process which functions in parallel to the rest of the framework, helps to visualize learning results and run experiments, without it every experiment has to be manually launched. The user can start an experiment through *GUI* which requests the *Server*, the later creates the *ECF* and launches *Algorithm*. To visualize the data of the experiment, *GUI* interacts with *Server*, which requests *Database*, and returns data.

Database To promote a disciplined approach to data management we employ a database within the component *Database*. The later is a program which works in parallel to other components and stores data from *ECF* and every learning episode statistics performing ordinary "create", "read", "update" and "delete" operations.



(a) (b)



(c)

Figure 6.5: Framework GUI and simulation views: (a) experiment configuration, (b) robot learning ascent task, (c) episode reward scatter

6.4 Experiments

This section presents conducted experiments which allowed to obtain ascent and descent policies with optimization of desired properties. We were inspired by the perception principles from [22] where authors study the case of 2D navigation learning and investigate its extension for control learning in ascent and descent tasks. The experiments that we present in the sequel are provided as proof-of-concept for the utility of the framework, with the hope of stimulating further research and allowing to benchmark different RL approaches for learning control of articulated tracked robots in indoor environments. The chapter is accompanied by a repository where the framework is stored `github.com/gwaxG/robot_ws`, additionally, GUI is located on `github.com/gwaxG/robot-simu`.

In the scope of this chapter, we employ SAC [20]. This is a recent RL algorithm which has shown better performance compared to its counterparts. It optimizes a stochastic policy in an off-policy way where its key feature is maximization of both the policy entropy and the expected return. We employ SAC mostly with its original hyperparameters, policy and Q-function parameters from the *Stable baselines3* RL library [110] which contains many other state-of-the-art algorithms.

Environment configuration We deploy our framework and obtain policies *asc-inc-cog*, *asc-uni-cog*, *des-inc-ang* and *des-uni-ang* for tasks presented in Table 6.1 where the staircase parameters are correspondingly sampled from incremental (**-inc-**) and uniform (**-uni-**) distributions and *Jaguar* was employed using 5 DOFs which are linear velocity, angles of paired front and rear flippers and 2 arm joint angles. We fix a payload to the end-effector representing 10% of robot mass to enhance resilience to the influence of a transported object. Commonly, in every experiment the robot starts off facing the staircase. Angular velocity control is redundant in such setting because the robot can move only forward, therefore it is omitted in the action vector as well as horizontal features and roll base angle in the observation vector, and it makes sense to couple left and right flipper angles at the front and at the rear respectively. The observation vector is populated with $(v_s, \psi_s^{front}, \psi_s^{rear}, \phi_s^1, \phi_s^2, \theta_p, o_1, ..., o_{60})$, i.e. linear velocity, front and rear flipper angles where front and rear flippers are coupled, arm joint angles, pitch angle of the platform and 60 vertical features extracted from the depth image. The action vector contains $(v_a, \psi_a^{front}, \psi_a^{rear}, \phi_a^1, \phi_a^2)$ and represents commands for linear velocity, front and rear flippers, and arm joints. We employ a 2-layer perceptron in which each inner layer possesses 64 neurons whose weights are updated by SAC.

Each task is performed three times in total and lasts up to 20000 time steps but can be terminated earlier if the average return over the latest 30 episodes reaches an empirical threshold value of 0.6 for descent and 0.5 for ascent when learning converges.

Task id	**Direction**	**Environment**	**Criterion**
asc-inc-cog	ascent	incremental	COG
asc-uni-cog	ascent	uniform	COG
des-inc-ang	descent	incremental	ang. vel.
des-uni-ang	descent	uniform	ang. vel.

Table 6.1: Learning tasks

Ascent task performance analysis Figure 6.6 presents smoothed episode return curves during *asc-inc-cog* and *asc-uni-cog* tasks learning with min-max bands. Since the duration of each experiment can vary, the x-axis presents a universal *learning time* which reflects scaling of all experiments time steps to the $[0, 1]$ space. For the *asc-inc-cog* task, we can see that episode return reaches up to 0.4 by the end of learning time. The task *asc-uni-cog* exhibits a similar evolution, however its reward curve is mainly located below the curve of *asc-inc-cog* task and converges to 0.0 which shows boost of learning with Gaussian sampling of environment in comparison to uniform one. The convergence of the COG deviation illustrated in Figure 6.7 shows the respective criteria are clearly optimized. If the evolution of reward curves seems slightly different in comparison to Chapter 4, this can be due to the differences in input modalities and, most importantly, to the usage of

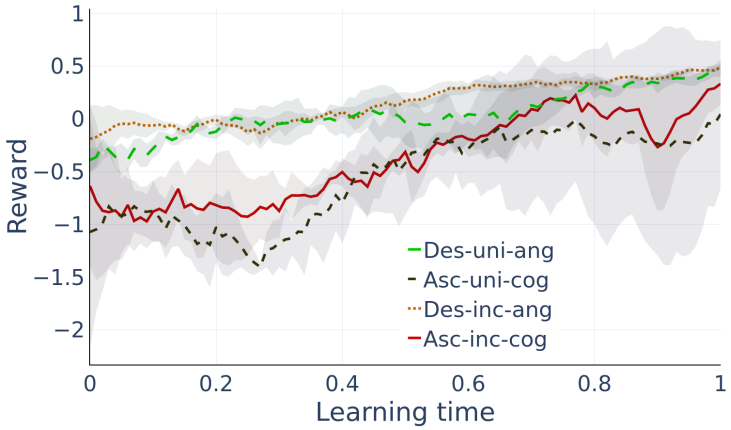
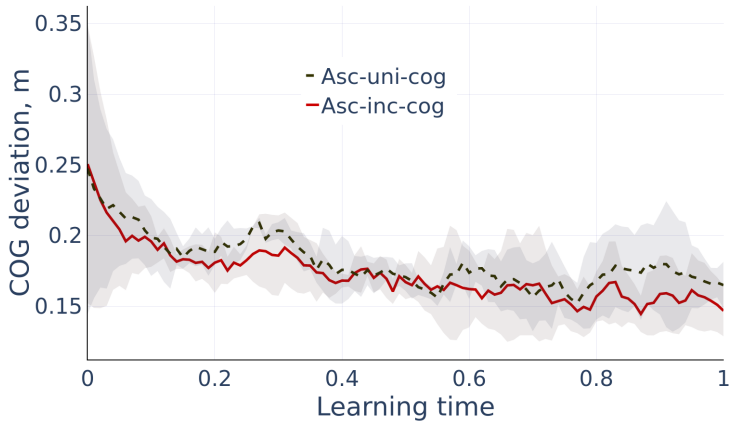
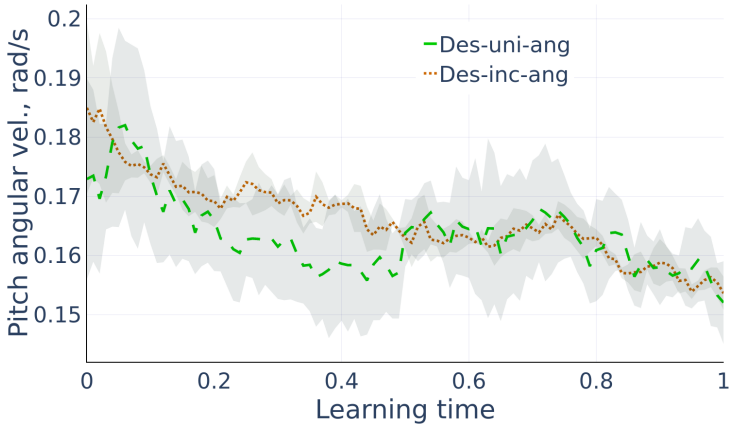


Figure 6.6: Reward convergence



(a)



(b)

Figure 6.7: Evolution of optimized values during task execution

SAC instead of PPO for optimizing the policy.

Figure 6.7 (a) shows evolution of COG deviation during learning, we can see that its values significantly drops from 0.25 m down to 0.14 m. The agent shows its capability to learn staircase traversal relying on visual perception and to increase its safety through COG deviation minimization. Speaking about performance of the *asc-uni-cog*, we can see that COG deviation is not optimized equally well and converges to 0.17 m, suggesting that the incremental domain randomization leads to a more optimized behaviour.

Descent task performance analysis Figure 6.6 (b) presents episode return in *des-inc-ang* and *des-uni-ang* tasks. Reward curves and mean episode pitch angular velocity show the same pace of convergence and attain similar values. Reward curves begin at -0.2 and -0.42, then they drastically increase up to 0.5 by the end of learning, and

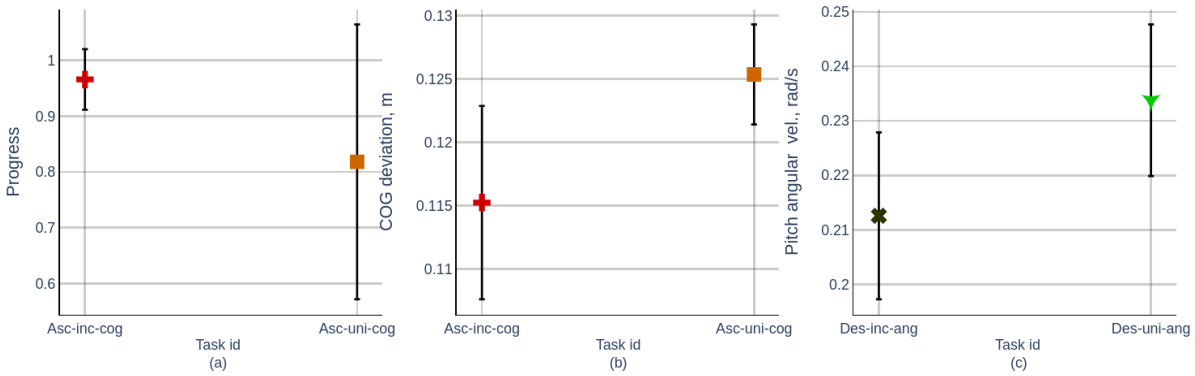


Figure 6.8: Comparison of environments

they slowly continue to improve after 0.8 of learning time. They both trigger termination by the early stopping criterion when the value 0.5 is reached. Overall, however, the task learnt incrementally *des-inc-ang* has fewer performance oscillations and hence seems more stable than the non-incremental.

Mean time step angular velocity (see Figure 6.7 (b)) drops from 0.185 and 0.173 for *des-inc-ang* and *des-uni-ang* respectively, to 0.155 by the end of learning for both tasks. As previously, we can notice here too that overall the task learnt incrementally oscillates less. In descent, the robot is always able to advance downstairs through even a small velocity application, the principal goal is to mitigate drop impacts which could occur even more often by inappropriate behaviours. Nonetheless, the curves of pitch angular velocity evolution show that the robot achieves the prescribed goal.

Test performance Figure 6.8 presents how much the performance of a policy trained in the uniform environment differs from a policy trained in the incremental environment for ascent and descent tasks. To obtain these statistics, a policy trained for a given task was applied 10 trials on a fixed staircase with an environment configuration of the highest complexity $\boldsymbol{\xi} = (H_{step}^{max}, L_{step}^{max}, N_{step}^{max})$. Then, relative travelled distance (progress), COG deviation and angular velocity were recorded during trials.

Figure 6.8 (a) presents mean progress in ascent tasks over all trials, with error bars corresponding to one standard deviation (we refrain from using progress as performance measure in descent tasks, since the goal is reached more easily). As we can see, a policy *asc-inc-cog* exhibits nearly perfect performance 0.965 ± 0.05 in contrast to *asc-uni-cog* policy 0.82 ± 0.24 which tends to reach the goal less often and has higher variance. At the same

time, the COG deviation of the policy (see Figure 6.8 (b)) *asc-inc-cog* is 0.115 ± 0.008 m which is lower than the COG deviation of the policy *asc-uni-cog* 0.125 ± 0.005 m by 0.01 m.

The incremental environment also improves performance in descent tasks (see Figure 6.8 (c)) where the policy *des-inc-ang* exhibits better performance 0.215 ± 0.15 m against the one of the policy *des-uni-ang* 0.235 ± 0.14 m.

6.5 Summary

This chapter presented a RL-based software framework for control learning of articulated tracked robots. The framework is unique in its kind in terms of the type of task for which it is destined to be used, integrating domain randomization and the possibility of incremental learning, accompanied with two articulated tracked robot models.

The framework applied on control learning for ascent and descent staircase traversals with safety constraints has shown ability to learn reasonable skills with joint arm control relying on depth. Furthermore, enhancement of DR with sampling of environment configurations from a Gaussian distribution, that is controlled by the estimation of learning progress, has shown superior results in comparison to the uniform environment.

We believe that the framework could stimulate research and experimentation in various directions. For example, possible future improvements of the framework could further account for generation of spiral staircase generation or more complex variations of floor obstacles to increase complexity of the 2D control learning and better address the structural complexity of real-world environments. Another extension could concern dynamics randomization. To start, we can vary the number of DOFs of the robot in an incremental learning setting, for example, by starting learning using 2 DOFs corresponding to linear and angular velocity control and then progressively adding additional DOFs of flipper and arm control while environment complexity increases. One could go even further by varying dimensions and mass of chassis, tracks, arm and flippers. Thus, the learnt policy could eventually attain generalization over different platforms.

Chapter 7

Conclusion

This dissertation investigated the problem of safe 3D navigation in indoor environments for articulated tracked robots, focusing on the most challenging part related to staircase negotiation. Based on a thorough and transverse study of the state-of-the-art, we inferred that previously proposed solutions are largely over-customized, limiting their applicability in different contexts. This prompted us to explore a reinforcement-learning based, control learning paradigm in order to reduce the amount of expert supervision and increase flexibility to varying conditions.

In view of the complexity in modelling and addressing the complete task of staircase negotiation, our approach in this dissertation was to start from moderate hypotheses and constraints and progressively scale up to harder challenges until the final application to the real robot. We began working within a simplified simulation environment where a robot with tracks simulated by wheels was able to learn its primary goal of ascending a staircase in a few learning episodes. Subsequently, we started to explore secondary goals by endowing the main negotiation behavior with different properties created with the help of designed reward functions based on conventional safety criteria such as SM and NESM. This allowed us to have a first idea of the appropriateness of criteria that were introduced in classic control, when employed in a RL setting. With a view towards a better generalization in the real-world, we specifically accounted for the presence of noise in state estimation while testing in unseen staircases.

The understanding of the main constituents of the problem then allowed us to further investigate the integration of a robotic arm, descent traversal and improvement of the fidelity of the simulation. This resulted in the development of a CSM-based simulated model of the commercial robot *Jaguar V4 with Arm*, increasing the effectiveness and efficiency of the simulation and significantly reducing the gap between simulation and real-world, in terms of robot-surface interaction. To improve learning convergence of desired skills and control bias of secondary properties of a behavior, we developed a sound approach to estimate the scale of secondary rewards and used it as scaling coefficient. We

further designed two dedicated reward functions for ascent and descent traversals based on optimization of the COG and of the pitch angle velocity. Experimental results obtained in simulation showed that behaviours respect safety constraints while negotiating varying staircases with the active help of the arm. Despite the additional degree of complexity due to the arm, the robot yields better overall stability during both traversal types even in the presence of a carrying load.

As an ultimate indicator of the value of the previous work, we succeeded in performing a zero-shot transfer to the real, commercial platform. The cross policy application tests and KL divergence analysis have further allowed us to assess the applicability of the approach to different platforms, demonstrating that the learnt behaviours are adapted to a given platform and the imposed safety constraints.

To consolidate our framework and allow subsequent improvements by the research community, we make it available publicly in an open-source repository[1], destined for 3D navigation and control learning of articulated tracked robots. This work is unique in terms of type of tasks, the integration of domain randomization for staircases and the possibility of incremental learning accompanied with two CSM-based articulated tracked robot models. Using the framework, we could seek an end-to-end control learning in the same environment setting as previously described.

[1] `https://github.com/gwaxG/robot_ws`

7.1 Challenges and perspectives

With a view towards a ready-to-deploy system that would integrate the solutions developed in this dissertation, a number of secondary open points would merit further attention. First of all, we have relied on a exteroceptive perception setup which allowed to estimate poses of the robot base and of the staircase. This is a reasonable hypothesis for an indoor service robot that can communicate with a smart living environment. If a higher degree of autonomy is required, then the robot would inevitably need to rely on SLAM as well as a staircase perception skill.

With a view towards a complete 3D indoor navigation system, the developed staircase negotiation controllers should be coupled with conventional 2D navigation based on planning or end-to-end based for map-less navigation. The software framework that we provide supports the second paradigm. Preferring the first of the second paradigm are interesting topics as well as the transfer to reality in the second case.

Transfer and deployment to reality of the complete navigation system would face various challenges, from domain differences between simulated and real RGB-D images, to varying levels of noise or perturbations of the platform due to shaking or the question of sensor placement on the platform.

We already rely on domain randomization in our work to partially deal with such challenges that could be even more alleviated by dynamics randomization. To do so, we could vary mass and geometric characteristics, dynamics parameters of chassis, tracks, flippers and arm links. Such technique could improve policy transferability to the destined platform making policy deployment more reliable and address more difficult dynamics-related problems. Furthermore, one could achieve transferability of one policy onto different platforms making generalization over environments and robots at the same time.

Finally, while we manage to produce a dedicated controller per task, it could be envisioned to apply a hierarchical approach where one controller could be continuously trained on multiple different tasks, allowing one to train a single neural network to navigate an entire building. In all cases, transferring to reality would be the most challenging problem and at the same time the best criterion for the efficacy of the developed research.

7.2 Publications of the dissertation

This content of the dissertation has been based on the following published articles:

1. A. Mitriakov, P. Papadakis, S. M. Nguyen and S. Garlatti, *Learning-based modelling of physical interaction for assistive robots,* in **Journées Francophones sur la Planification, la Décision et l'Apprentissage pour la conduite de systèmes**, 2019, `https://hal.archives-ouvertes.fr/hal-02341202/`

2. A. Mitriakov, P. Papadakis, S. M. Nguyen and S. Garlatti, *Staircase traversal via reinforcement learning for active reconfiguration of assistive robots,* in **IEEE International Conference on Fuzzy Systems**, 2020, `https://doi.org/10.1109/FUZZ48607.2020.9177581`

3. A. Mitriakov, P. Papadakis, S. M. Nguyen and S. Garlatti, *Staircase negotiation learning for articulated tracked robots with varying degrees of freedom,* in **IEEE International Symposium on Safety, Security and Rescue Robotics,** 2020, `https://doi.org/10.1109/SSRR50563.2020.9292594`

4. A. Mitriakov, P. Papadakis, J. Kerdreux and S. Garlatti. *Reinforcement learning based, staircase negotiation learning in simulation and transfer to reality for articulated tracked robots,* in **IEEE Robotics and Automation Magazine**, vol. 28, no. 4, pp. 10–20, 2021, `https://doi.org/10.1109/MRA.2021.3114105`

The work 2 was appreciated and supported with a conference registration grant for PhD students and work 4 has been invited for presentation in IEEE International Conference on Robotics and Automation 2022. The next article is under review:

1. *A. Mitriakov, P. Papadakis and S. Garlatti, "A software framework for reinforcement learning-based control of tracked robots in simulated indoor environments",* in **Advanced Robotics**, 2021.

As an open source contribution presented in that paper, a repository is made available at `https://github.com/gwaxG/robot_ws`. We hope that it will serve as a basis for future indoor navigation and control skills learning through reinforcement learning.

7.3 Funding

The present work is performed in the context of the leadership program M@D (Chaire Maintien@Domicile), project REACT financed by Brest Métropole and the region of Brittany (France) and supported by project VITAAL (Vaincre l'Isolement par les TIC pour l'Ambient Assisted Living), cofinanced by European Regional Fund (FEDER).

BIBLIOGRAPHY

[1] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Berlin, Heidelberg: Springer-Verlag, 2007.

[2] K. Doelling, J. Shin, and D. O. Popa, "Service robotics for the home: A state of the art review," in *Int. Conf. on Pervasive Technologies Related to Assistive Environments*, 2014.

[3] P. Papadakis, C. Lohr, M. Lujak, A. Karami, I. Kanellos, G. Lozenguez, and A. Fleury, "System design for coordinated multi-robot assistance deployment in smart spaces," in *IEEE Int. Conf. on Robotic Computing*, 2018.

[4] M. Devanne, P. Papadakis, and S. M. Nguyen, "Recognition of activities of daily living via hierarchical long-short term memory networks," in *IEEE Int. Conf. on Systems, Man and Cybernetics*, 2019, pp. 3318–3324.

[5] E. G. Christoforou, A. S. Panayides, S. Avgousti, P. Masouras, and C. S. Pattichis, "An overview of assistive robotics and technologies for elderly care," *IFMBE Proceedings*, 2019.

[6] A. S. Prabuwono, K. Allehaibi, and K. Kurnianingsih, "Assistive robotic technology: A review," *Computer Engineering and Applications Journal*, vol. 6, pp. 71–78, 07 2017.

[7] F. Rubio, F. Valero, and C. Llopis-Albert, "A review of mobile robots: Concepts, methods, theoretical framework, and applications," *Int. J. of Advanced Robotic Systems*, vol. 16, no. 2, 2019.

[8] A. I. Mouriakis, N. Trawny, S. I. Roumeliotis, D. M. Helmick, and L. Matthies, "Autonomous stair climbing for tracked vehicles," *The Int. J. of Robotics Research*, vol. 26, no. 7, pp. 737–758, 2007.

[9] K. Nagatani, A. Yamasaki, K. Yoshida, T. Yoshida, and . Koyanagi, "Semi-autonomous traversal on uneven terrain for a tracked vehicle using autonomous control of active flippers," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2008.

[10] D. M. Helmick, S. I. Roumeliotis, M. C. McHenry, and L. Matthies, "Multi-sensor, high speed autonomous stair climbing," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2002.

[11] J. A. Hesch, G. L. Mariottini, and S. I. Roumeliotis, "Descending-stair detection, approach, and traversal with an autonomous tracked vehicle," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2010.

[12] M. Brunner, T. Fiolka, D. Schulz, and C. M. Schlick, "Design and comparative evaluation of an iterative contact point estimation method for static stability estimation of mobile actively reconfigurable robots," *Robotics and Autonomous Systems*, vol. 63, pp. 89 – 107, 2015.

[13] M. Pecka and K. Zimmermann, "Safe exploration for reinforcement learning in real unstructured environments," in *20th Computer Vision Winter Workshop*, 2015.

[14] S. Soichiro, H. Satoshi, and O. Masayuki, "Remote control system of disaster response robot with passive sub-crawlers considering falling down avoidance," *ROBOMECH Journal*, vol. 1, no. 1, p. 20, Nov 2014.

[15] S. Hirose, H. Tsukagoshi, and K. Yoneda, "Normalized energy stability margin and its contour of walking vehicles on rough terrain," in *IEEE Int. Conf. on Robotics and Automation*, 2001.

[16] G. Dulac-Arnold, D. J. Mankowitz, and T. Hester, "Challenges of real-world reinforcement learning," *CoRR*, vol. abs/1904.12901, 2019.

[17] M. Pecka, S. Valansky, K. Zimmermann, and T. Svoboda, "Autonomous flipper control with safety constraints," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2016.

[18] A. G. Kupcsik, M. P. Deisenroth, J. Peters, and G. Neumann, "Data-efficient generalization of robot skills with contextual policy search," in *Proceedings of the Twenty-Seventh AAAI Conf. on Artificial Intelligence*, 2013.

[19] C. Zhou, B. Huang, and P. Fränti, "A review of motion planning algorithms for intelligent robotics," *arXiv*, vol. abs/2102.02376, 2021.

[20] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine, "Learning to walk via deep reinforcement learning," *Int. Conf. on Machine Learning*, 2018.

[21] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.

[22] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," *CoRR*, vol. abs/1703.00420, 2017.

[23] G. Kahn, P. Abbeel, and S. Levine, "Badgr: An autonomous self-supervised learning-based navigation system," *ArXiv*, vol. abs/2002.05700, 2020.

[24] 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 *17th Int. Conf. on Informatics, Automation and Robotics, ICINCO*, 2020.

[25] J. Kober, J. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The Int. J. of Robotics Research*, vol. 32, pp. 1238–1274, 09 2013.

[26] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," in *NIPS Deep Learning Workshop*, 2013.

[27] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 2021.

[28] S. Aradi, "Survey of deep reinforcement learning for motion planning of autonomous vehicles," *arXiv*, vol. 2001.11231, 2020.

[29] A. Azar, A. Koubaa, N. Ali Mohamed, H. Ibrahim, Z. Ibrahim, M. Kazim, A. Ammar, B. Benjdira, A. Khamis, I. Hameed, and G. Casalino, "Drone deep reinforcement learning: A review," *Electronics*, vol. 10, 2021.

[30] 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*, 2020.

[31] B. Qin, Y. Gao, and Y. Bai, "Sim-to-real: Six-legged robot control with deep reinforcement learning and curriculum learning," in *4th Int. Conf. on Robotics and Automation Engineering*, 2019.

[32] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th Annual Int. Conf. on Machine Learning*, 2009.

[33] E. Garcia, J. Estremera, and P. Gonzalez-de-Santos, "A classification of stability margins for walking robots," *Robotica*, vol. 20, no. 6, pp. 595–606, 2002.

[34] N. Sato, M. Kitani, and Y. Morita, "Control method for rollover recovery of rescue robot considering normalized energy stability margin and manipulating force," in *2019 12th Int. Workshop on Robot Motion and Control (RoMoCo)*, 2019, pp. 160–165.

[35] M. Pecka, K. Zimmermann, and T. Svoboda, "Fast simulation of vehicles with non-deformable tracks," *CoRR*, vol. abs/1703.04316, 2017.

[36] S. Parsons, "Probabilistic robotics by sebastian thrun, wolfram burgard and dieter fox, mit press, 647 pp., 55.00, isbn 0-262-20162-3," *The Knowledge Engineering Review*, vol. 21, no. 3, p. 287–289, 2006.

[37] C. Colas, P. Fournier, O. Sigaud, and P.-Y. Oudeyer, "Curious: Intrinsically motivated multi-task, multi-goal reinforcement learning," *CoRR*, vol. abs/1810.06284, 2018.

[38] S. Nakajima, "Stair-climbing gait for a four-wheeled vehicle," *Robomech J.*, 2020.

[39] M. Menna, M. Gianni, F. Ferri, , and F. Pirri, "Real-time autonomous 3d navigation for tracked vehicles in rescue environments," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2014.

[40] K. Thurow, L. Zhang, H. Liu, S. Junginger, N. Stoll, and J. Huang, "Multi-floor laboratory transportation technologies based on intelligent mobile robots," *Transportation Safety and Environment*, vol. 1, no. 1, pp. 37–53, 2019.

[41] H. Chung, C. Hou, Y. Chen, and C. Chao, "An intelligent service robot for transporting object," in *IEEE Int. Symp. on Industrial Electronics*, 2013.

[42] S. Caron, A. Kheddar, and O. Tempier, "Stair climbing stabilization of the hrp-4 humanoid robot using whole-body admittance control," *IEEE Int. Conf. on Robotics and Automation*, 2019.

[43] S. Kajita, M. Morisawa, K. Miura, S. Nakaoka, K. Harada, K. Kaneko, F. Kanehiro, and K. Yokoi, "Biped walking stabilization based on linear inverted pendulum tracking," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2010.

[44] A. Bouton, C. Grand, and F. Benamar, "Obstacle negotiation learning for a compliant wheel-on-leg robot," in *IEEE Int. Conf. on Robotics and Automation*, 2017.

[45] A. Mitriakov, P. Papadakis, S. Mai Nguyen, and S. Garlatti, "Staircase negotiation learning for articulated tracked robots with varying degrees of freedom," *IEEE Int. Symposium on Safety, Security, and Rescue Robotics*, 2020.

[46] L. Tai and M. Liu, "Deep-learning in mobile robotics - from perception to control systems: A survey on why and why not," *ArXiv*, vol. abs/1612.07139, 2016.

[47] D. Kortenkamp, R. G. Simmons, and D. Brugali, "Robotic systems architectures and programming," in *Springer Handbook of Robotics, 2nd Ed.*, 2016.

[48] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, "Simultaneous localization and mapping: A survey of current trends in autonomous driving," *IEEE Transactions on Intelligent Vehicles*, vol. 2, no. 3, pp. 194–220, 2017.

[49] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar slam," in *IEEE Int. Conf. on Robotics and Automation*, 2016.

[50] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *Proceedings IEEE Int. Symposium on Safety, Security and Rescue Robotics*, 2011.

[51] J. Fuentes-Pacheco, J. Ascencio, and J. Rendon-Mancha, "Visual simultaneous localization and mapping: A survey," *Artificial Intelligence Review*, vol. 43, 11 2015.

[52] F. Peralta, M. Arzamendia, D. Gregor, D. G. Reina, and S. Toral, "A comparison of local path planning techniques of autonomous surface vehicles for monitoring applications: The ypacarai lake case-study," *Sensors*, vol. 20, no. 5, 2020.

[53] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. SSC-4(2), pp. 100–107, 1968.

[54] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 354–363, 2005.

[55] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., 1998.

[56] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[57] L. Coles, C. Rosen, B. Raphael, T. Garvey, and R. Duda, "Application of intelligent automata to reconnaissance," *Technical report*, p. 159, 11 1969.

[58] R. Brooks, "A robust layered control system for a mobile robot," *IEEE J. on Robotics and Automation*, vol. 2, no. 1, pp. 14–23, 1986.

[59] M. Colledanchise and P. Ögren, "Behavior trees in robotics and ai," Jul 2018.

[60] S. K. Tzafestas, V. Vassiliades, F. Stulp, S. Calinon, and J. Mouret, "Mobile robot control and navigation: A global overview," *J. of Intelligent & Robotic Systems*, vol. 91, pp. 35–58, 2018.

[61] S. G. Tzafestas, *Introduction to mobile robot control*. Elsevier, 2014.

[62] M. Bajracharya, A. Howard, L. Matthies, B. Tang, and M. Turmon, "Autonomous off-road navigation with end-to-end learning for the lagr program," *J. Field Robotics*, vol. 26, pp. 3–25, 01 2009.

[63] A. Kelly, *RANGER: Feedforward Control Approach to Autonomous Navigation*. Boston, MA: Springer US, 1997, pp. 105–144.

[64] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, and P. Mahoney, "Stanley: The robot that won the darpa grand challenge." *J. Field Robotics*, vol. 23, pp. 661–692, 01 2006.

[65] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[66] 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 *Int. Conf. on Learning Representations*, 2016.

[67] D. C. Guastella and G. Muscato, "Learning-based methods of perception and navigation for ground vehicles in unstructured environments: A review," *Sensors*, vol. 21, no. 73, 2021.

[68] P. Papadakis, "Terrain traversability analysis methods for unmanned ground vehicles: A survey," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 4, pp. 1373 – 1385, 2013.

[69] M. P. Deisenroth, G. Neumann, and J. Peters, "A survey on policy search for robotics," *Found. Trends Robot*, vol. 2, pp. 1–142, 2013.

[70] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," *CoRR*, vol. abs/1802.09477, 2018.

[71] H. Hu, K. Zhang, A. H. Tan, M. Ruan, C. Agia, and G. Nejat, "A sim-to-real pipeline for deep reinforcement learning for autonomous robot navigation in cluttered rough terrain," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6569–6576, 2021.

[72] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proceedings of The 33rd Int. Conf. on Machine Learning*, 2016.

[73] J. Martens and W. Newman, "Stabilization of a mobile robot climbing stairs," in *IEEE Int. Conf. on Robotics and Automation*, 1994.

[74] P. Ben-Tzvi, S. Ito, and A. A. Goldenberg, "A mobile robot with autonomous climbing and descending of stairs," *Robotica*, vol. 27, no. 2, p. 171–188, 2009.

[75] F. Colas, S. Mahesh, F. Pomerleau, M. Liu, and R. Siegwart, "3d path planning and execution for search and rescue ground robots," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2013.

[76] H. Zhang and A. Song, "System centroid position based tipover stability enhancement method for a tracked search and rescue robot," *Advanced Robotics*, vol. 28, no. 23, pp. 1571–1585, 2014.

[77] M. Gianni, F. Ferri, M. Menna, and F. Pirri, "Adaptive robust three-dimensional trajectory tracking for actively articulated tracked vehicles*," *J. of Field Robotics*, vol. 33, no. 7, pp. 901–930, 2016.

[78] D. Endo, A. Watanabe, and K. Nagatani, "Stair climbing control for 4-dof tracked vehicle based on internal sensors," *J. of Robotics*, pp. 1–18, 10 2017.

[79] Q. Xie, B. Su, J. Guo, H. Zhao, and W. Lan, "Motion control technology study on tracked robot with swing arms," in *IEEE Int. Conf. on Unmanned Systems*, 2017.

[80] M. Oehler, S. Kohlbrecher, and O. von Stryk, "Optimization-based planning for autonomous traversal of obstacles with mobile ground robots," in *Int. J. of Mechanics and Control*, 2020.

[81] K. Zimmermann, P. Zuzanek, M. Reinstein, and V. Hlavac, "Adaptive traversability of unknown complex terrain with obstacles for mobile robots," in *IEEE Int. Conf. on Robotics and Automation*, 2014.

[82] M. M. Ejaz, T. B. Tang, and C.-K. Lu, "Vision-based autonomous navigation approach for a tracked robot using deep reinforcement learning," *IEEE Sensors J.*, vol. 21, no. 2, 2021.

[83] J. Liu, Y. Wang, S. Ma, and B. Li, "Analysis of stairs-climbing ability for a tracked reconfigurable modular robot," in *IEEE Int. Safety, Security and Rescue Rototics, Workshop*, 2005.

[84] J. P. Trevelyan, S.-C. Kang, and W. R. Hamel, *Robotics in Hazardous Applications.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 1101–1126.

[85] M. Norouzi, J. Valls Miro, and G. Dissanayake, "Planning stable and efficient paths for reconfigurable robots on uneven terrain," *J. of Int. and Robotic Systems*, vol. 87, 08 2017.

[86] E. Papadopoulos and D. Rey, "The force-angle measure of tipover stability margin for mobile manipulators," *Vehicle System Dynamics - VEH SYST DYN*, vol. 33, pp. 29–48, 2000.

[87] D. Ernst, P. Geurts, and L. Wehenkel, "Tree-based batch mode reinforcement learning." *J. of Machine Learning Research*, vol. 6, pp. 503–556, 04 2005.

[88] K. Otsu, G. Matheron, S. Ghosh, O. Toupet, and M. Ono, "Fast approximate clearance evaluation for rovers with articulated suspension systems," *J. of Field Robotics*, 2019.

[89] B. D. Ilhan, A. M. Johnson, and D. E. Koditschek, "Autonomous stairwell ascent," *Robotica*, vol. 38, no. 1, p. 159–170, 2020.

[90] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, *Handbook of Robotics*. MIT Press, 2007, no. 59, ch. Robot Programming by Demonstration.

[91] F. Torabi, G. Warnell, and P. Stone, "Behavioral cloning from observation," in *Proceedings of the 27th Int. Joint Conf. on Artif. Intell.*, 2018.

[92] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems*, vol. 29. Curran Associates, Inc., 2016.

[93] A. Bouton, C. Grand, and F. Benamar, "Obstacle negotiation learning for a compliant wheel-on-leg robot," in *IEEE Int. Conf. on Robotics and Automation*, 2017.

[94] E. Garcia, J. Estremera, and P. Gonzalez-de Santos, "A comparative study of stability margins for walking machines," *Robotica*, vol. 20, 2002.

[95] M. Vukobratovic, A. A. Frank, and D. Juricic, "On the stability of biped locomotion," *IEEE Transactions on Biomedical Engineering*, vol. BME-17, no. 1, pp. 25–36, 1970.

[96] M. Brunner, C. M. Schlick, and F. Flemisch, "Rough terrain motion planning for actively reconfigurable mobile robots," 2015.

[97] R. McGhee and A. Frank, "On the stability properties of quadruped creeping gaits," *Mathematical Biosciences*, vol. 3, pp. 331 – 351, 1968.

[98] R. B. Mcghee and G. I. Iswandhi, "Adaptive locomotion of a multilegged robot over rough terrain," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 4, pp. 176–182, 1979.

[99] D. A. Messuri, "Optimization of the locomotion of a legged vehicle with respect to maneuverability," *The Ohio State University. ProQuest Dissertations Publishing.*, 1985.

[100] N. Koenig and A. Howard, *Gazebo-3d multiple robot simulator with dynamics*. Technical report, 2006.

[101] E. Rohmer, S. P. N. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2013.

[102] L. Nogueira, "Comparative analysis between gazebo and v-rep robotic simulators," *Seminario Interno de Cognicao Artificial-SICA*, 2014.

[103] Y. Zhu, R. Mottaghi, E. Kolve, J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," *IEEE Int. Conf. on Robotics and Automation*, 2017.

[104] B. Shen, F. Xia, C. Li, R. Martín-Martín, L. Fan, G. Wang, S. Buch, C. D'Arpino, S. Srivastava, L. P. Tchapmi, M. E. Tchapmi, K. Vainio, L. Fei-Fei, and S. Savarese, "igibson, a simulation environment for interactive tasks in large realistic scenes," *CoRR*, vol. abs/2012.02924, 2020.

[105] P. Wenzel, T. Schön, L. Leal-Taixé, and D. Cremers, "Vision-based mobile robotics obstacle avoidance with deep reinforcement learning," *IEEE Int. Conf. on Robotics and Automation*, 2021.

[106] I. Zamora, N. G. Lopez, V. M. Vilches, and A. H. Cordero, "Extending the openai gym for robotics: a toolkit for reinforcement learning using ros and gazebo," *ArXiv*, vol. abs/1608.05742, 2016.

[107] B. Kenwright and G. Morgan, *Practical introduction to rigid body linear complementary problem (LCP) constraint solvers.* IGI Global, 2012.

[108] M. Sokolov, I. Afanasyev, R. Lavrenov, A. Sagitov, L. Sabirova, and E. Magid, "Modelling a crawler-type ugv for urban search and rescue in gazebo environment," in *Int. Conf. on Artificial Life and Robotics*, 01 2017.

[109] G. Paolo, L. Tai, and M. Liu, "Towards continuous control of flippers for a multi-terrain robot using deep reinforcement learning," *CoRR*, vol. abs/1709.08430, 2017.

[110] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann, "Stable baselines3," https://github.com/DLR-RM/stable-baselines3, 2019.

[111] Ł. Kidziński, S. P. Mohanty, C. F. Ong, Z. Huang, S. Zhou, A. Pechenko, A. Stelmaszczyk, P. Jarosik, M. Pavlov, S. Kolesnikov, S. Plis, Z. Chen, Z. Zhang, J. Chen, J. Shi, Z. Zheng, C. Yuan, Z. Lin, H. Michalewski, P. Milos, B. Osinski, A. Melnik, M. Schilling, H. Ritter, S. F. Carroll, J. Hicks, S. Levine, M. Salathé, and S. Delp, "Learning to run challenge solutions: Adapting reinforcement learning methods for neuromusculoskeletal environments," in *The NIPS '17 Competition: Building Intelligent Systems.* Cham: Springer Int. Publishing, 2018, pp. 121–153.

[112] Student, "The probable error of a mean," *Biometrika*, pp. 1–25, 1908.

[113] A. Papoulis, *Probability, Random Variables and Stochastic Processes.* McGraw-Hill, Inc., 1991.

[114] F. Klinker, "Exponential moving average versus moving exponential average," *Mathematische Semesterberichte*, vol. 58, no. 1, p. 97–107, 2010.

[115] K. Sasaki, Y. Eguchi, and K. Suzuki, "Stair-climbing wheelchair with lever propulsion control of rotary legs," *Advanced Robotics*, vol. 34, no. 12, pp. 802–813, 2020.

[116] A. Mitriakov, P. Papadakis, S. Mai Nguyen, and S. Garlatti, "Staircase traversal via reinforcement learning for active reconfiguration of assistive robots," *IEEE Int. Conf. on Fuzzy Systems*, 2020.

[117] A. Mitriakov, P. Papadakis, J. Kerdreux, and S. Garlatti, "Reinforcement learning based, staircase negotiation learning: Simulation and transfer to reality for articulated tracked robots," *IEEE Robotics and Automation Magazine*, vol. 28, no. 4, pp. 10–20, 2021.

[118] M. Pecka, "Safe autonomous reinforcement learning," *PhD Thesis*, 2020.

[119] J. M. Joyce, *Kullback-Leibler Divergence.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 720–722.

[120] P. Papadakis and P. Rives, "Binding human spatial interactions with mapping for enhanced mobility in dynamic environments," *Auton. Robots*, vol. 41, no. 5, pp. 1047–1059, 2017.

[121] A. Sinha, P. Papadakis, and M. R. Elara, "A staircase detection method for 3d point clouds," in *IEEE Int. Conf. on Control Automation Robotics Vision*, 2014, pp. 652–656.

[122] A. Mitriakov, P. Papadakis, and S. Garlatti, "A software framework for reinforcement learning-based control of tracked robots in simulated indoor environments," *Advanced Robotics*, 2022.

[123] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system," *ICRA Workshop on Open Source Software*, 2009.

[124] J. Coughlan and A. L. Yuille, "The manhattan world assumption: Regularities in scene statistics which enable bayesian inference," *Advances in Neural Information Processing Systems*, vol. 13, 2001.

[125] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *ArXiv vol. abs/1606.01540*, 2016.

DOCTORAT BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Contrôle Fondé sur un Apprentissage par Renforcement pour la Navigation 3D Sécurisée de Robots Manipulateurs Articulés à Chenilles

Mot clés : Apprentissage par renforcement, robotique mobile, robots à chenilles, contrôle sûr

Résumé :

Le transport d'objets par un robot d'assistance à chenilles équipé d'un bras, constitue un moyen pour pallier à la perte d'autonomie des personnes atteintes de déficiences cognitives ou motrices. Les travaux de recherche précédents reposant sur un contrôle conventionnel pour la navigation 3D, ils manquent de généralisation et de portabilité à différents robots ou environnements. Le contrôle basé sur l'apprentissage par renforcement (RL) constitue une alternative moins supervisée, reposant sur des hypothèses moins restrictives quant à la connaissance de la dynamique du robot ou de la structure de l'environnement.

Cette thèse propose une solution de navigation 3D fondée sur le RL qui utilise tous les degrés de liberté d'un robot. En particulier, nous contribuons à la formalisation ainsi qu'au traitement du problème de négociation d'escalier pour la montée et pour la descente avec contrôle du bras. Nous analysons la portabilité de la solution à différents robots et réalisons une démonstration du transfert des contrôleurs sur un robot réel. Enfin, nous avons développé et mis à disposition un environnement logiciel pour l'apprentissage de navigation intérieure 3D en simulation, capable d'effectuer un apprentissage du contrôle de bout-en-bout de façon incrémentale.

Title: Reinforcement Learning-based Control for Safe 3D Navigation of Articulated Tracked Robot Manipulators

Keywords: Reinforcement learning, mobile robotics, tracked robots, safe control

Abstract:

Object transportation by an assistive tracked robot equipped with an arm constitutes a way to palliate the autonomy loss of people suffering from cognitive or motor impairments. As previous research works rely on conventional control to perform 3D navigation, they lack generalization and portability to different robots or environments. Reinforcement learning (RL) based control is a less supervised alternative, based on less restrictive assumptions on the knowledge of the robot dynamics or the structure of the environment.

This thesis proposes a 3D navigation solution based on RL that uses all degrees of freedom of a robot. In particular, we contribute to the formalization and the treatment of the staircase negotiation problem for ascent and descent with arm control. We analyze the portability of the solution to different robots and perform a demonstration of the transfer of controllers on a real robot. Finally, we have developed and made available a software environment for learning indoor 3D navigation, capable of performing end-to-end control learning in an incremental manner.