Génération de mouvements robotiques complexes et sûrs

Sebastien Lengagne

UNIVERSITÉ CLERMONT AUVERGNE

RAPPORT

pour obtenir le diplôme de

Habilitation à Diriger des Recherches

de : **École doctorale Sciences Pour l'Ingénieur de Clermont-Ferrand**

par

Sébastien LENGAGNE

le 24 janvier 2025

préparée à l'Institut Pascal :

Génération de mouvements robotiques complexes et sûrs

Jury :

Rapporteurs :	David Filliat, Professeur, ENSTA-Paris
	Luc Jaulin, Professeur, ENSTA-Bretagne
	Nicolas Mansard, Directeur de recherche, LAAS-CNRS, Toulouse
Examinateurs :	Pierre-Brice Wieber, Chargé de recherche, INRIA, Grenoble
	Thierry Chateau, Professeur, Institut Pascal, Clermont-Ferrand
	Youcef Mezouar, Professeur, Institut Pascal, Clermont-Ferrand (responsable tutélaire)

À Audrey, Chloé et Robin

Remerciements

Mes premiers remerciements vont envers les membres du jury qui ont accepté d'évaluer mon travail : David Filliat, Luc Jaulin et Nicolas Mansard en tant que rapporteurs, Pierre-Brice Wieber, Roland Lenain en tant qu'examinateurs et Youcef Mezouar en tant que responsable tutélaire.

Je souhaite également remercier Rawan Kalawoun, Mélodie Daniel, Mehdi Mounsif et Samuel Beaussant : les différents doctorants que j'ai encadrés. La qualité de leur travail a permis plusieurs contributions qui sont présentées dans ce manuscrit. Ce fut un réel plaisir de travailler avec chaun d'entre eux. A cette liste, il faut inclure les directeurs et co-encadrants de thèse : Youcef Mezouar, Lounis Adouane, Olivier Stasse, Benoît Thuilot et Juan Ramon Corralès.

Ce manuscrit ne serait pas ce qu'il est sans toutes les rencontres qui ont parsemé mon parcours : Christine Chevallereau, Yannick Aoustin et Sylvain Miossec pour ma première expérience de recherche et mon stage de Master au JRL, Philippe Fraisse et Nacim Ramdani mes encadrants de thèse au LIRMM, Kazuhito Yokoi, Abderrahmane Kheddar et Eiichi Yoshida pour m'avoir accueilli pendant mon post-doctorat au JRL, sans oublier toute l'équipe du JRL de cette époque qui m'ont fait passer deux années incroyables (Olivier Stasse, Claire Dune, François Keith, Pierre Gergondet et Sébastien Druon), Tamim Asfour pour m'avoir accueilli au KIT et Sébastien Druon, alias le petit lapin blanc, pour m'avoir proposé un an de post-doctorat au LIRMM et m'avoir permis et incité à réaliser les enseignements nécessaires à la qualification au poste de maître de conférences et tout ceux que je n'ai pas pu citer par manque de place.

Je tiens également à remercier Pierre Breul et Christophe Pasquier pour l'appui de ma demande de CRCT qui m'a permis de rédiger ce manuscrit dans de bonnes conditions, sans oublier Romuald Aufrere pour avoir assurer les enseignements d'automatique pendant ce laps de temps.

Globalement, je tiens à remercier les collègues enseignants, les élèves et les membres de Polytech Clermont et de l'Institut Pascal qui essayent de me supporter ce qui n'est peut être pas toujours évident, avec une mention spéciale pour Jacques Laffont et Benoît Thuilot qui m'ont vu envahir leurs bureaux tel un éléphant dans un jeu de quille.

Enfin, comment ne pas remercier, ma compagne Audrey qui elle aussi me supporte, depuis longtemps maintenant, mais surtout pour tout ce qu'elle m'apporte dans la vie et bien évidemment mes deux grumeaux Chloé et Robin qui nous occupent beaucoup mais qui la majorité du temps égayent nos journées ! Je vous aime tous profondément.

“L’intelligence ce n’est pas ce que l’on sait,
mais ce que l’on fait quand on ne sait pas.”

Jean Piaget, pshychologue suisse (1896-1980).

Table des matières

Liste des abréviations vii
Liste des figures viii
Liste des tableaux x

1 Activités scientifiques, pédagogiques et administratives 1
1.1 Curriculum Vitae 1
1.1.1 Formation 1
1.1.2 Parcours professionnel 1
1.2 Activités d'enseignement 2
1.2.1 Responsabilités pédagogiques 3
1.3 Activités scientifiques 5
1.3.1 Contextes et équipes de recherche 5
1.3.2 Encadrements de thèse 7
1.3.3 Encadrements de stages 9
1.3.4 Publications 10
1.3.5 Autres activités 13

2 Introduction 15

3 Génération de mouvements complexes 18
3.1 Contexte 18
3.2 Positionnement scientifique 19
3.2.1 Planification à partir de modèles simples 19
3.2.2 Planification *kino-dynamique* 20
3.2.3 Planification à partir du modèle dynamique complet 20
3.3 Modélisation et contraintes du mouvement 20
3.3.1 Particularité des robots à pattes 20
3.3.2 Modélisation géométrique et dynamique 21
3.3.3 Contraintes articulaires 21
3.3.4 Contraintes géométriques 22
3.3.5 Équilibre 23
3.3.6 Collisions et auto-collisions 24
3.4 Formulation du problème de génération de mouvements 25
3.4.1 Problème infini de la génération de mouvements 25
3.4.2 Paramétrisation du mouvement 26
3.5 Gestion des contraintes continues 27
3.5.1 Contraintes sur les trajectoires articulaires 27
3.5.2 Approximation polynomiale 28
3.5.3 Contraintes égalités 29

3.5.4 Contraintes inégalités . . . 29
3.6 Résultats de générations de mouvements . . . 30
3.7 Complexité et garantie vs. temps de calcul . . . 33

4 L'Analyse par Intervalles(AI) 34
4.1 Contexte . . . 34
4.2 Introduction à l'Analyse par Intervalles(Analyse par Intervalles (AI)) . . . 35
4.2.1 Définitions . . . 35
4.2.2 Surapproximation . . . 36
4.2.3 Optimisation globale . . . 36
4.2.4 Utilisation de l'AI en robotique . . . 36
4.2.5 Calcul d'ensemble faisable . . . 37
4.3 Approximation par fonctions de base . . . 38
4.3.1 Propriétés des fonctions de base . . . 38
4.3.2 Estimation multi-variables . . . 39
4.3.3 Utilisation lors du processus d'optimisation . . . 40
4.3.4 Fonctions de bases . . . 41
4.3.5 Valeurs et optimalité des fonctions de base . . . 44
4.3.6 Évaluation du pessimisme pour un robot 3D . . . 45
4.3.7 Evaluation des performances durant un processus d'optimisation . . . 46
4.4 Garantie vs. temps de calcul et complexité . . . 48

5 l'IA : Intelligence Artificielle 50
5.1 Contexte . . . 50
5.2 Le *Reinforcement Learning*(RL) . . . 52
5.3 Manipulation d'objet déformable . . . 53
5.4 Transfert de compétences . . . 55
5.4.1 Formalisation du problème . . . 55
5.4.2 Etat de l'art . . . 58
5.4.3 Différentes approches de transfert . . . 61
5.4.4 *CoachGAN* . . . 62
5.4.5 *Task-Specific Loss*(TSL) . . . 64
5.4.6 *Universal Notice Network*(UNN) . . . 65
5.4.7 Le *Delay Aware Universal Notice Network* (DA-UNN) . . . 72
5.4.8 Le *Latent Space Universal Notice Network* (LS-UNN) . . . 74
5.5 Transfert : résultats et perspectives . . . 81
5.6 Complexité et temps de calcul vs. garantie . . . 82

6 Conclusion et projet de recherche 83
6.1 Conclusion . . . 83
6.2 Projet de recherche . . . 84
6.2.1 Développement des méthodes de transfert pour la génération de mouvements . . . 84
6.2.2 Garanties théoriques pour la génération de mouvements . . . 86
6.2.3 L'humain dans le transfert . . . 88

A Description des activités d'enseignement **103**
A.1 Enseignements récurrents . . . 103
A.1.1 Automatique : . . . 103
A.1.2 Automatique numérique : . . . 103
A.1.3 Initiation à ROS : . . . 103
A.1.4 C++, CMAKE et versionning : . . . 104
A.1.5 Robots manipulateurs et introduction à l'Intelligence Artificielle : . . . 104
A.1.6 Introduction à la robotique humanoïde : . . . 104
A.1.7 Projet de Synthèse : . . . 104
A.2 Enseignements ponctuels . . . 104
A.2.1 Robotique mobile : . . . 105
A.2.2 Réseaux et html . . . 105
A.3 Projets . . . 105
A.3.1 Projet industriel en **GE** : . . . 105
A.3.2 Sous-traitance : . . . 105
A.3.3 Autres projets . . . 106
A.4 Enseignements à l'étranger . . . 106
A.4.1 Djibouti : . . . 106
A.4.2 Norvège : . . . 107

B Sélection d'articles **108**
B.1 Génération de mouvements complexes . . . 109
B.2 Utilisation des fonctions de base dans un processus d'optimisation. . . . 141
B.3 Transfert de compétences . . . 164

Table des abréviations

AE	*Auto Encoder*
AI	Analyse par Intervalles
AIST	*National Institute of Advanced Industrial Science and Technology*
BAM	*Base Abstracted Modelling*
CNRS	Centre National de la Recherche Scientifique
DA-UNN	*Delay Aware Universal Notice Network*
DDPG	*Deep Deterministic Policy Gradient*
DEMAR	DEambulation et Marche ARtificielle
GAN	*Generative Adversarial Networks*
IA	Intelligence Artificielle
INRIA	Institut National de Recherche en Informatique et en Automatique
ISPR	Image, Systèmes de Perception, Robotique
ISV	*Intermediate State Variables*
JRL	*Joint Robotics Laboratory CNRS-AIST JRL UMI3218/CRT*
KIT	*Karlsruher Institut für Technologie*
LIRMM	Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier
LS-UNN	*Latent Space Universal Notice Network*
MACCS	*Modeling, Autonomy and Control in Complex Systems*
MDP	*Markov Decision Process*
PPO	*Proximal Policy Optimization*
RL	*Reinforcement Learning*
SEF	Stimulation Électrique Fonctionnelle
TAC	*Task-Centered*
TEC	*TEacher-Centered*
TSL	*Task-Specific Loss*
UCA	Université Clermont Auvergne
UNN	*Universal Notice Network*
VAE	*Variational Auto Encoder*
ZMP	*Zero Moment Point*

Table des figures

1.1 Charge d'enseignements depuis ma prise de poste à Polytech Clermont (les CM sont x1.5). 3

2.1 Représentation de quelques outils utilisés pour la génération de mouvements en fonction de leur complexité, temps de calcul et garantie. 16

3.1 Illustration des articulations virtuelles permettant de définir la pose d'un robot mobile. Source [91]. 21
3.2 Description du contact de la main gauche du robot HRP-2 avec la table. Le repère $\mathbf{X}_i$ est fixé à la main gauche du robot et doit coïncider avec le repère $\mathbf{X}_i^e$ situé sur la table. La position du repère $\mathbf{X}_i^e$ peut être imposée ou issue de processus d'optimisation. 22
3.3 Description des forces de contact pour le pied gauche. Nous considérons 4 forces linéaires, une pour chaque coin de la surface de contact. 24
3.4 Illustration des contraintes de continuité. Source [120]. 26
3.5 llustration de la décomposition du mouvement en plusieurs phases et intervalles de temps. 27
3.6 Validation expérimentale d'un mouvement multi-contact. 31
3.7 Mouvement de marche sur sol plan. 31
3.8 Marche du robot avec pas sur une marche de 15cm 31
3.9 Environnement du robot HOAP-3 à Béziers télé-opéré par la pensée depuis un IRM en Israël. 32
3.10 Représentation des efforts de contact du robot HRP-2 lors d'un mouvement de marche normale et en émulant un pied fragile. source [108] 32
3.11 Mouvement de marche simulant une blessure au genou 33
3.12 Mouvement de marche simulant une blessure au pied 33
3.13 Simulation d'un mouvement de transfert pour un patient paraplégique. . . 33

4.1 Représentation simpliste de la différence entre les méthodes statistiques et les méthodes ensemblistes. 34
4.2 Défintion des paramètres du coup de pied et de l'espace atteignable autour du mouvement optimal. 37
4.3 Représentation temporelle de la position x du pied et de la contrainte d'équilibre (ZMP dans le plan sagittal), le mouvement adapté ainsi que l'ensemble des mouvements faisables. 38
4.4 Representation of the diameter of the evaluation using basis function regarding intervalle analysis. 47
4.5 Comparaison du nombre d'itérations. 47
4.6 Comparaison du temps de calcul (sans prise en compte du temps de préparation. 48

4.7 Comparaison du temps de calcul total. 48

5.1 Représentation ironique de mes premières impressions concernant la justification de l'utilisaton des réseaux de neurones en robotique via une analogie avec l'automobile. 51
5.2 Schéma de principe du *Reinforcement Learning*(*Reinforcement Learning* (RL)) représentant l'interaction entre l'agent et l'environnement. 52
5.3 Manipulation d'objet déformable source : [37]. 54
5.4 Illustration des points appartenant à l'objet déformable en bleu et leurs positions désirées en rouge. 54
5.5 Représentation de l'approche mise en place pour le contrôle de la déformation d'un objet souple via l'algorithme *Deep Deterministic Policy Gradient* (DDPG). 55
5.6 Expérimentation de contrôle de la déformation d'un objet. 56
5.7 Représentation artistique du transfert de compétences en robotique, source : [1]. 57
5.8 Construction de l'espace de représentation invariant, source [61]. 59
5.9 Représentation des modules de tâches et de robots et définition des mondes *source* et *target*, d'après [44]. 60
5.10 Comparaison des différents types d'approches pour le transfert de compétences (source [130]). 61
5.11 Principe du GAN : le générateur essaie de produire des images pour tromper le discriminateur (source). 62
5.12 Principe du coachGAN : le discriminateur est appris sur des trajectories expertes, puis il guide l'apprentissage du générateur/étudiant (source : [130]). 63
5.13 Représentation d'un *Variational Auto Encoder* (VAE), source : [130]. . . . 65
5.14 Représentation du VAE pour la reconstruction de l'action d'un robot expert, source : [130]. 65
5.15 Principe du transfert de la méthode *Task-Specific Loss* (TSL), seul le décodeur du robot cible est appris, source : [130]. 66
5.16 Principe de contrôle humain via l'espace latent en utilisant la méthode TSL source : [130]. 66
5.17 Architecture de l'*Universal Notice Network* (UNN) 67
5.18 Architecture de l'UNN en cas de réalisation de tâches multi-robots 68
5.19 Principe du *Base Abstracted Modelling* (BAM) où on considère un robot virtuel ramené à son effecteur, source [130]. 69
5.20 Ensemble des robots et tâches utilisés pour valider la méthode UNN, source [133]. 70
5.21 Illustration de la tâche de tennis, source [130]. 70
5.22 Courbes d'apprentissage et de transfert pour la tâche de tennis, du robot KUKA au robot Blue, source [130]. 71
5.23 Biais de comportement induit par l'UNN, source [130]. 72
5.24 Illustration du processus d'apprentissage dans le cas d'un système à retard. 72
5.25 Architecture du DA-UNN. 73
5.26 Représentation de la tâche d'asservissement en position d'une bille sur un rail. Le robot de gauche sera commandé afin de monter ou descendre le rail. Ici le robot de droite ne sert que de support (il ne participe pas à la réalisation de la tâche). 74

5.27 Protocole d'apprentissage et d'évaluation du DA-UNNpour la tâche d'asservissement en position d'une bille sur un rail (source [130]). 74
5.28 Résultat de simulation et d'expérimentation pour la tâche d'asservissement en position d'une bille sur un rail (source [130]). 75
5.29 Architecture du *Latent Space Universal Notice Network* (LS-UNN) 75
5.30 Les 4 étapes de la mise en place du LS-UNN, source [8]. 76
5.31 Représentation de la tâche *primitive* de *reaching*. Le robot doit positionner son effecteur sur la cible violette, source [8]. 77
5.32 Représentation des différents critères à prendre en compte lors de l'apprentissage des bases pour le LS-UNN, considérant un robot r_b et un robot r_p, source [8]. 77
5.33 Protocole d'évaluation du LS-UNN, sources [8]. 79
5.34 Validation de la tâche de *peg insertion* sur les robots réels. La position du trou est obtenue en début d'expérience par un système de motion capture, sources [8]. 80

6.1 Représentation des contributions vers un algorithme de génération de mouvements idéal. 84
6.2 Proposition de gestion des retards dans le cadre du transfert. 85
6.3 Proposition de gestion des capteurs différents pour un même robot. 86
6.4 Propositions d'architecture d'une tâche de co-manipulation pour intégrer un ou plusieurs humains à la place d'un ou de plusieurs robots. 89
6.5 Modification de l'action proposée par l'UNN en fonction d'une émotion désirée. 89

A.1 Autres exemples de projet étudiants. 106

Liste des tableaux

1.1 Chronologie des activités administratives liées à l'enseignement à Polytech Clermont. 3
1.2 Chronologie des activités de recherche. 5

4.1 Représentation des différentes fonctions de base pour des ordres 2 à 6. . . . 44
4.2 Valeurs des matrices $\mathbf{B}^{-1}$ pour les différentes fonctions de base et pour des ordres 1 à 36. 45
4.3 Valeurs des différents critères de chaque fonction de base pour des ordres 1 à 3. 46

5.1 Comparaison des performances sur la tâche de tennis. 71
5.2 Performances de transfert pour la tâche de *peg insertion* en simulation. Les résultats sont donnés en pourcentage de succès pour 1000 essais. 80
5.3 Performances de transfert pour la tâche de *peg insertion* en réel. Les résultats sont donnés en pourcentage de succès pour 28 essais. 81
5.4 Tableau récapitulatif des tâches dont le transfert a été validé en simulation ou en expérimentation réelle en utilisant les différentes méthodes. 81

Chapitre 1

Activités scientifiques, pédagogiques et administratives

1.1 Curriculum Vitae

Sébastien Lengagne
Né le 2 mai 1983, à Grande-Synthe (59)
Pacsé, deux enfants (Chloé et Robin)
Site web : `http://cloud.ip.uca.fr/~lengagne/`
contact : `sebastien.lengagne@uca.fr`, `lengagne@gmail.com`

1.1.1 Formation

— 2009 : Doctorat en robotique au Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM) sur la *Planification et re-planification de mouvements sûrs pour les robots humanoïdes*

— 2006 : Master 2 Recherche Automatique et Systèmes de Production, parcours robotique à l'École Centrale de Nantes.

— 2005 : Master 1 Automatique et Systèmes Électriques à l'Université Lille 1.

— 2004 : Licence Ingénierie Electrique à l'Université Lille 1.

— 2003 : DUT GEII (Diplôme Universitaire de Technologie Génie Électrique et Informatique Industrielle) option électronique à l'Université Lille 1.

— 2001 : Baccalauréat Sciences et Techniques Industriels, génie électronique.

1.1.2 Parcours professionnel

— sept 2013 - maintenant : Maître de Conférences dans l'axe Image, Systèmes de Perception, Robotique (ISPR) , équipe *Modeling, Autonomy and Control in Complex Systems* (MACCS) de l'Institut Pascal (UMR 6602 - UCA/ CNRS) et rattaché à l'école d'ingénieur universitaire Polytech Clermont, dans les départements Génie électrique et Génie physique.

— sept.2012 - août 2013 : Post-doctorat dans l'équipe EXPLORE du LIRMM, dans le cadre du projet R.HEX (Robot HEXapode), sur *la conception et le contrôle d'un robot hexapode.*

— nov.2011 - juil 2012 : Post-doctorat dans le Humanoids and Intelligence Systems Lab de l'Institute for Anthropmatics du *Karlsruher Institut für Technologie* (KIT) sur *la transposition des mouvements humains au robot humanoïde ARMAR-IV* à Karlsruhe en Allemagne.

— nov.2009 - oct.2011 : Post-doctorat au *Joint Robotics Laboratory CNRS-AIST JRL UMI3218/CRT* (JRL) sur la *génération de mouvements dynamiques multi-contacts pour les robots humanoïdes et avatars* à Tsukuba au Japon.

— oct.2006 - oct.2009 : Doctorat en robotique au LIRMM dans l'équipe DEMAR (DEambulation et Marche ARtificielle) de l'INRIA sur la *Planification et re-planification de mouvements sûrs pour les robots humanoïdes*, à Montpellier en France.

— mai 2006- sept. 2006 : Stage recherche de Master 2 au JRL sur l'*optimisation de mouvement multi-contacts pour le robot HRP-2* à Tsukuba au Japon.

— avril 2003 - juin 2003 : Stage de DUT à l'IEMN (Institut d'Electronique, de Microélectronique et de Nanotechnologie) UMR 8520, sur la *production d'une horloge à 7.6 GHz* à Lille en France.

1.2 Activités d'enseignement

En tant que maître de conférences, je suis rattaché au département *Génie Électrique : Conversion d'Énergie, Systèmes Embarqués et Robotique* (**GE**) et au département *Génie Physique* (**GP**) de Polytech Clermont, école d'ingénieur universitaire membre de Clermont Auvergne INP et du réseau Polytech.

Le département **GE**, dans lequel j'effectue la majorité de mes activités d'enseignement, forme des ingénieurs capables de piloter des projets en spécifiant, concevant et réalisant des systèmes électroniques complexes [1] alliant à la fois des compétences logicielles et matérielles. Les étudiants de **GE** sont répartis dans deux options pour les semestres 8 et 9 de leur formation :

— option A : Systèmes de conversion d'énergie et robotique,

— option B : Systèmes informatiques embarqués.

Mes activités d'enseignement consistent en des cours/TD/TP en **GE**, dont une part est mutualisée avec le département **GP**, et en des projets principalement dans le département **GE**, et ponctuellement pour le département de Polytech Clermont : *Ingénierie Mathématiques et Data Sciences* (**IMDS**) et dans le *Parcours des Écoles d'Ingénieurs Polytech* (**PEIP**). J'interviens également dans le Master 2 *Perception Artificielle et Robotique* (**PAR**) et dans le Master 1 *Automatique-Robotique* (**AR**) de l'École Universitaire de Physique et d'Ingénierie de l'Université Clermont Auvergne (UCA) (EUPI).

Depuis la crise sanitaire du Covid, au cours de l'année 2020-2021, les conditions de télé-enseignement m'ont poussé à mettre mes cours en ligne. Ils sont disponibles sur cette chaîne en ligne que j'essaie de maintenir et d'agrémenter de nouvelles vidéos. L'évolution de la répartition et de ma charge d'enseignements au cours des différentes années est présentée sur la Figure 1.1 et détaillée en Annexe A. Lors de l'année 2023-2024, j'ai bénéficié d'un Congé pour Recherches ou Conversions Thématiques (CRCT) pour le premier semestre.

1. source : `https://polytech-clermont.fr/diplome-dingenieur-en-genie-electrique`

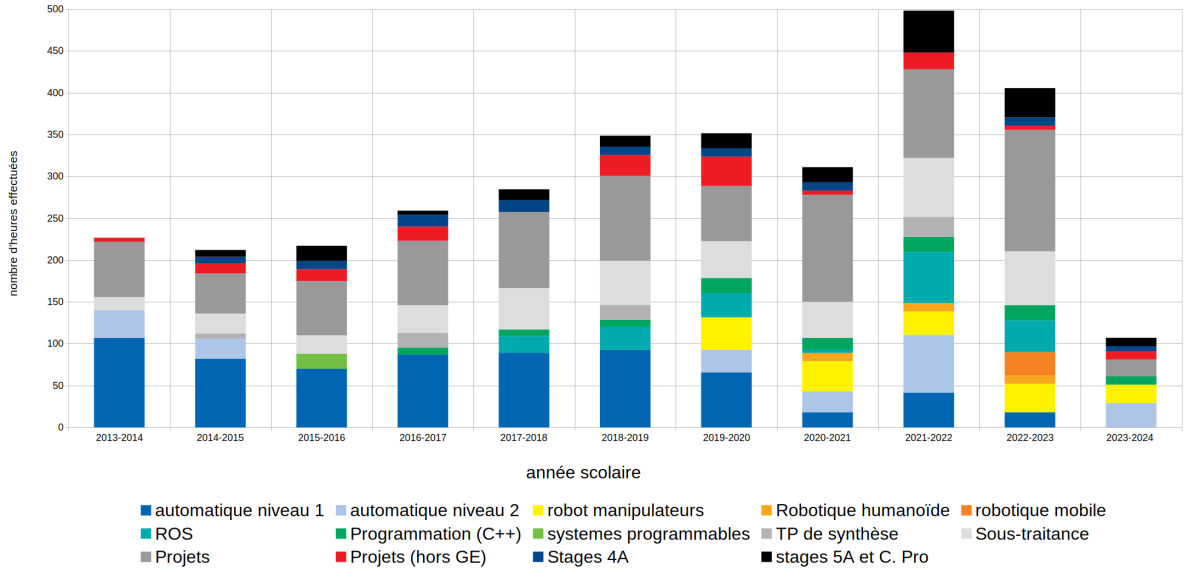


FIGURE 1.1 – Charge d'enseignements depuis ma prise de poste à Polytech Clermont (les CM sont x1.5).

1.2.1 Responsabilités pédagogiques

Depuis ma prise de poste, j'ai pu assumer différentes responsabilités en lien avec mes activités pédagogiques. Elles sont détaillées ci-dessous et répertoriées dans la Table 1.1.

Année / Responsabilités	2015-2016	2016-2017	2017-2018	2018-2019	2018-2019	2019-2020	2020-2021	2021-2022	2022-2023	2023-2024	2024-2025
Responsable Année (GE4)											
Stages **GE4** à l'étranger											
Relations Internationales **GE**											
Page youtube du **GE**											
Relations Internationales Polytech											
Stages **GE5**											
Adjoint du **GE**											
Option A											
Montage de la FISA											

TABLE 1.1 – Chronologie des activités administratives liées à l'enseignement à Polytech Clermont.

Responsable adjoint du département Génie Electrique de Polytech-Clermont :

Entre septembre 2021 et juin 2024, ma principale mission, en plus d'épauler le responsable de département, est d'animer les discussions autour d'une évolution de la maquette pédagogique avec les enseignants, la direction de l'école et les acteurs socio-économiques dans le but d'accroître la visibilité de notre formation et permettre à nos futurs diplômés de répondre aux défis du futur auxquels ils devront faire face.

Responsable de l'option Conversion d'Energie et Robotique du département Génie Electrique de Polytech-Clermont :

Depuis septembre 2021, ma mission est de mettre en place les réflexions et d'établir les moyens nécessaires afin d'apporter aux étudiants une expertise dans ce domaine et d'accroître la visibilité de cette thématique au sein du site clermontois et au niveau national, notamment à l'intention des futurs étudiants.

Montage d'une formation par apprentissage :

Depuis 2022, les deux responsabilités précédentes m'ont logiquement conduit à organiser le montage d'une formation par apprentissage au sein du **GE** autour des systèmes de conversion d'énergie électrique. Cette tâche regroupe à la fois le contact avec les entreprises, le dépôt de dossier auprès de la CTI (Commission des Titres d'Ingénieur) et à partir de 2024 la définition et la mise en place du contenu pédagogique en se basant sur l'Approche Par Compétences (APC).

Correspondant des relations internationales (Europe du Nord) :

Depuis 2021, Polytech Clermont a organisé ces relations internationales de manière à avoir un responsable par zone géographique. On m'a proposé d'être en charge de l'Europe du Nord. Ma mission est d'accompagner les étudiants (entre 3 et 5 chaque année) qui souhaitent faire une mobilité (semestre ou double diplôme) dans un pays d'Europe du Nord. Dans ce contexte, j'ai initié une collaboration avec l'Université Arctique de Norvège pour mettre en place des mobilités (entrantes et sortantes d'étudiants et de personnels) entre les deux institutions.

Responsable des stages à l'étranger pour les étudiants de 4ème année pour le département Génie Electrique de Polytech-Clermont :

Depuis 2019, je m'assure du bon déroulement des stages autant du point de vue du contenu que de l'accueil sur place de nos étudiants (entre 29 et 43 chaque année). Cette partie s'accompagne de la responsabilité des relations internationales pour le département Génie Electrique.

Responsable de la page youtube du département GE

Afin de communiquer sur nos activités d'enseignement, j'ai proposé la création d'une page youtube pour le département **GE** depuis 2019. Cette page disponible ici contient principalement les vidéos de projets des étudiants de dernière année mais également quelques vidéos de présentation du département.

Responsable d'année :

Entre 2015 et 2019, cette mission consistait à gérer l'emploi du temps des étudiants de **GE4** [2], accompagner les étudiants lors de leur choix d'option et assurer un suivi auprès des étudiants qui seraient le plus en difficulté.

2. **GEX** (ou **GPX**) est utilisé pour désigner les étudiants du département génie électrique (ou génie physique) de X-ème année

Responsable des stages de fin d'études :

Lors de l'année 2021-2022, j'ai assuré le remplacement du responsable des stages de fin d'études pour les **GE5** en vérifiant le bon déroulement des stages autant du point de vue du contenu que de l'accueil de nos étudiants sur place.

1.3 Activités scientifiques

Dans le cadre de mes activités de recherche, j'ai abordé plusieurs thématiques, décrites dans les chapitres suivants, dans plusieurs laboratoires en encadrant plusieurs doctorants. La Table 1.2 propose une représentation chronologique de mes activités de recherche détaillée ci après.

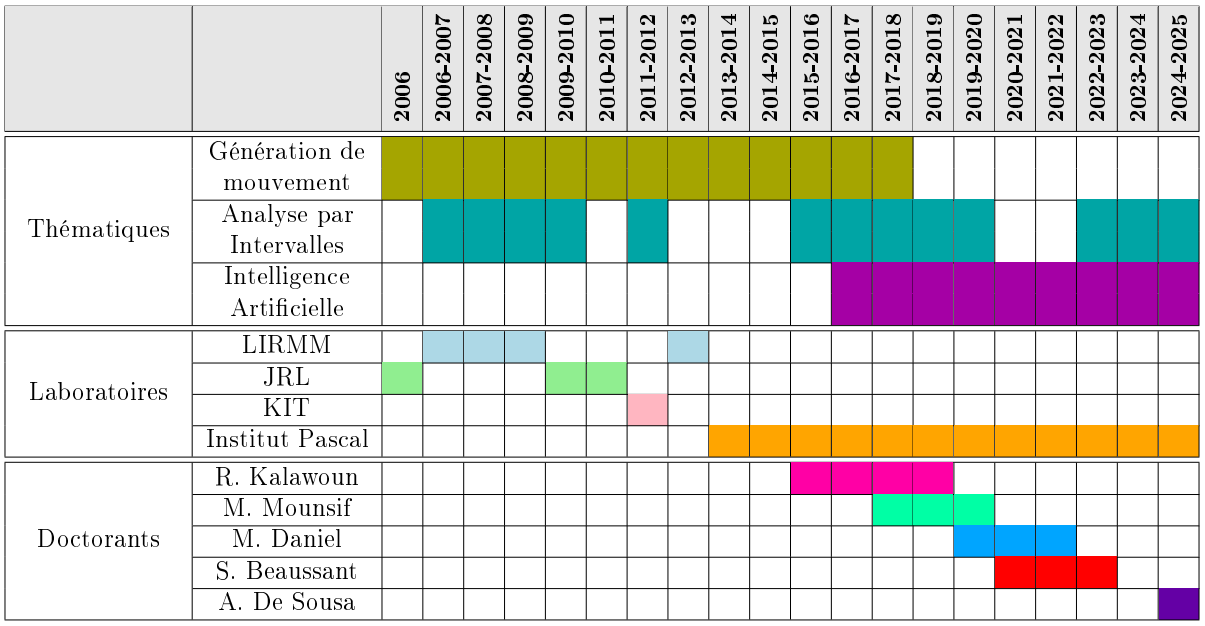


TABLE 1.2 – Chronologie des activités de recherche.

1.3.1 Contextes et équipes de recherche

2006-2009 : LIRMM - équipe DEambulation et Marche ARtificielle (DEMAR), Montpellier

Mes travaux de thèse, co-financés par l'Institut National de Recherche en Informatique et en Automatique (INRIA) et la Région Languedoc-Roussillon, ont été menés au sein de l'équipe-projet DEMAR située au LIRMM. L'équipe DEMAR [3] se concentrait sur la Stimulation Électrique Fonctionnelle (SEF) pour restaurer le mouvement de membres paralysés pour des patients paraplégiques. Leurs recherches incluaient des aspects tels que la modélisation, la commande, la synthèse et l'analyse du mouvement humain, ainsi que l'interface patient, en utilisant des approches de robotique et d'automatique. Ma mission portait sur le développement d'une méthode de planification de mouvements applicable à des patients sous SEF. Cette méthode a pu être validée sur le robot humanoïde HOAP-3 (Humanoïd for Open Architecture Platform 3) de l'entreprise Fujitsu.

3. L'équipe-projet DEMAR s'est arrêtée, le projet continue dans l'équipe CAMIN.

2006 et 2009-11 : JRL, Tsukuba, Japon

J'ai d'abord effectué une première période au JRL lors de mon stage de Master 2 qui a été financé par le projet ImmerSence EU CEC, Contrat n 27141 (FET Presence) dans le cadre du 6ème programme de recherche. Ensuite, j'ai effectué deux ans de Post-doctorat avec un financement de la *Japan Society for the Promotion of Science* (JSPS ; Grant-in-Aid for JSPS Fellows P09809 and for Scientific Research (B), 22300071, 2010). Le JRL à Tsukuba est une collaboration internationale entre le *National Institute of Advanced Industrial Science and Technology* (AIST) au Japon et le Centre National de la Recherche Scientifique (CNRS) en France. Les principaux sujets de recherche du JRL comprennent la planification et le contrôle des tâches et des mouvements, l'interaction multimodale avec les humains et l'environnement à travers la perception, et la robotique cognitive. Lors de mes deux périodes au JRL, j'ai orienté mes travaux sur la génération de mouvements multi-contacts pour les robots humanoïdes en considérant des modèles corps complets. J'ai pu valider mes travaux grâce à la plateforme HRP-2 [89], un robot humanoïde de 1.54m de hauteur conçu par la société Kawada en interaction avec l'AIST.

2011-2012 : KIT, Karlsruhe, Allemagne

En 2011-2012, j'ai effectué un an de post-doctorat dans l'équipe *Humanoids and Intelligence Systems Lab* de l'*Institute for Anthropmatics* du KIT, financé par la *Deutsche ForschungsGemeinschaft* (DFG). L'équipe de recherche développe des technologies pour permettre aux robots humanoïdes d'interagir avec les humains et d'accomplir diverses tâches dans le monde réel. Dans ce cadre, j'ai participé à l'élaboration du modèle cinématique et dynamique du robot ARMAR-IV [4]. J'ai également proposé une méthode de reconstruction des points de contact des mouvements humains issus de systèmes de *motion capture* via l'utilisation d'un modèle standardisé du modèle humain : le *Master Motor Map* (MMM) [166].

2012-2013 : LIRMM - équipe Explore, Montpellier

En 2012-2013, j'ai intégré l'équipe EXPLORE du LIRMM qui se focalise sur la conception et l'utilisation de robots sous-marins dans le cadre de la réalisation autonome de missions complexes en dépit des conditions imposées par le milieu extérieur. Ma mission était de participer à la conception et à la commande d'un robot hexapode terrestre pour des missions d'intervention dans des environnements fragiles avec un minimum d'impact sur l'environnement. Ce post-doctorat était financé par le Labex NUMEV (NUMérique pour l'Environnement et le Vivant) et localisé dans les locaux de l'IUT de Beziers.

2013-maintenant : Institut Pascal - axe ISPR, équipe MACCS, Clermont-Ferrand

Depuis septembre 2013, je suis maître de conférences dans l'équipe MACCS de l'Institut Pascal [4], UMR 6602, une unité mixte de recherche et de formation interdisciplinaire de 400 personnes placée sous la tutelle de l'Université Clermont Auvergne (UCA) et du CNRS. Le laboratoire développe des connaissances et des technologies contribuant à trois domaines d'application : l'usine (incluant les écosystèmes), les transports et l'hôpital du

4. source intégrale disponible : `http://www.institutpascal.uca.fr/index.php/fr/`

futur. L'Institut Pascal est membre de FACTOLAB, laboratoire commun avec MICHELIN. L'unité est structurée en cinq Axes de recherche :

— Génie des Procédés, Energétique et Biosystèmes (GePEB)

— Mécanique, Génie Mécanique, Génie Civil, Génie Industriel (M3G)

— Photonique, Ondes, Nanomatériaux (PHOTON)

— Thérapies Guidées par l'Image (TGI)

— Image, Systèmes de Perception, Robotique (ISPR)

Je suis un membre de l'axe ISPR qui œuvre dans le domaine de la perception et de la vision artificielle pour la commande des systèmes robotiques. Son objectif est donc le développement de concepts théoriques, méthodologiques et architecturaux pour la perception et le contrôle des systèmes [5]. L'axe ISPR est composé de quatre équipes :

— Vision artificielle : *Computer that Sees* (ComSee)

— Systèmes de perception multi-sensorielle (PerSyst)

— Architecture matérielle et logicielle pour la perception (DREAM)

— *Modeling, Autonomy and Control in Complex Systems* (MACCS)

Mes travaux s'inscrivent dans les thématiques de l'équipe MACCS qui portent principalement sur les véhicules autonomes et sur la modélisation, l'apprentissage et la perception pour la commande des robots manipulateurs.

1.3.2 Encadrements de thèse

Lors de mes activités scientifiques en tant que maître de conférences à l'Institut Pascal, j'ai eu l'occasion d'encadrer quatre étudiants en thèse (+1 qui débute en automne 2024) :

Rawan Kalawoun (2015-2019) :

Titre : *"Motion planning of multiple robotic system for air-plane stripping"*,

Directeur de thèse : Youcef Mezouar.

Financement : programme FUI-20, projet AEROSTRIP.

Descriptif : Le sujet de thèse visait à développer un système automatique de nettoyage écologique des avions en utilisant un abrasif projeté à grande vitesse (maïs). La thèse se concentrait sur l'optimisation des trajectoires d'un système robotique pour un décapage optimal. Elle contribue à l'étape de prétraitement en proposant une nouvelle fonction d'inclusion pour calculer l'espace de travail du robot, réduisant ainsi le pessimisme de l'Analyse par Intervalles. De plus, une contribution dans l'algorithme d'optimisation hybride permet de déterminer le nombre optimal de robots et leurs positions pour un décapage efficace. Les méthodes développées ont été testées sur des surfaces variées (voiture, avion, ...).

Mélodie Daniel (2019-2022) :

Titre : *"Optimizing Decision-Making for Human-Robot Collaboration"*,

Directeur de thèse : Youcef Mezouar.

5. source : `http://www.institutpascal.uca.fr/index.php/fr/presentation-ispr`

Co-encadrant : Juan Antonio Corrales Ramón.

Financement : Région Auvergne-Rhône-Alpes dans le cadre du projet ATTRIHUM et du programme de recherche et d'innovation Horizon 2020 de l'Union européenne sous l'accord de subvention n° 869855 (Projet 'SoftManBot').

Mobilité vers le laboratoire CiTIUS à Santiago de Compostela en Espagne : ERASMUS + Stages, projet I-SITE Clermont de CAP 20-25.

Descriptif : Cette thèse traitait de l'optimisation de la collaboration homme-robot, en mettant particulièrement l'accent sur l'amélioration du processus décisionnel du robot. En utilisant un framework global, l'objectif était de maximiser les performances de la collaboration sans augmenter les capacités physiques du robot. Appliqué à une tâche d'assemblage complexe, le framework montre son efficacité en prenant en compte différentes métriques de performance, indépendamment du comportement humain. La recherche s'étend également à une application plus complexe impliquant la déformation d'un objet, nécessitant une amélioration du contrôle bas-niveau du robot pour optimiser la collaboration homme-robot dans des scénarios de manipulation délicate.

Mehdi Mounsif (2017-2020) :

Titre : *"Exploration of Teacher-Centered and Task-Centered paradigms for efficient transfer of skills between morphologically distinct robots"*,

Directeur de thèse : Lounis Adouane,

Co-encadrant : Benoit Thuilot.

Financement : Allocation de recherche ministérielle attribuée par l'Ecole Doctorale des Sciences Pour l'Ingénieur.

Descriptif : Cette thèse s'intéressait aux transferts de compétences en robotique dans le cadre de méthodes d'apprentissage, soulignant que l'apprentissage à partir de zéro requiert souvent des millions d'interactions. Pour atténuer cette exigence en données, cette thèse visait à transférer des comportements d'un robot à un autre malgré leurs différences morphologiques. Deux approches, *Task-Centered* et *Teacher-Centered*, explorent le transfert de connaissances entre robots de structures différentes. *Task-Centered* dissocie le savoir-faire lié à la réalisation de la tâche et les stratégies de contrôle propres à chaque robot, tandis que *Teacher-Centered* utilise un agent expert pour distiller le savoir dans l'agent cible, surmontant les différences structurelles. Ces approches visent à rendre les robots plus adaptables et efficaces dans des environnements variés et ont été validées avec des robots industriels simulés lors de tâches dynamiques, de précision et en collaboration multi-robots.

Samuel Beaussant (2020-2023) :

Titre : *"Transfer Learning between robots with state abstraction"*

Directeur de thèse : Olivier Stasse,

Co-encadrant : Benoit Thuilot.

Financement : Contrat doctoral relatif au projet ANR-IA de l'Université Clermont Auvergne / I-SITE CAP 2025.

Descriptif : Cette thèse s'inscrit dans la continuité des travaux sur les transferts de compétences en robotique dans le cadre de méthodes d'apprentissage. Même si les résultats précédents semblent prometteurs, l'hypothèse de base est que les robots utilisés pour le transfert (source ou cible) ont des capacités similaires leur permettant de réaliser les tâches demandées de manière similaire. Dans l'optique d'exploiter complètement les capacités des systèmes robotiques, la définition des limites (butées, vitesse, collision, auto-collision, ...) des robots doit être prise en compte et doit être gérée lors des transferts de compétences d'un robot vers un autre robot disposant de capacités différentes, voire réduites. L'objectif de cette thèse est de proposer des méthodes de transfert qui permettent à plusieurs robots possédant des limites hétérogènes de réaliser la tâche avec des stratégies adaptées.

Antonio Claudio De Sousa Dos Santos Filho (début octobre 2024) :

Titre : *"Apprentissage et transfert de compétences pour des applications robotiques"*

Directeur de thèse : Sébastien Lengagne [6],

Co-encadrant : Benoit Thuilot.

Financement : Allocation de recherche ministérielle attribuée par l'Ecole Doctorale des Sciences Pour l'Ingénieur.

Descriptif : Cette thèse reprend la thématique du transfert compétences en explorant d'avantage les méthodes *Task-Centered* afin de transférer des comportements d'un robot à un autre malgré leurs divergences morphologiques. Toutefois, les méthodes précédentes supposent que les robots ont des capacités similaires. L'objectif de ces travaux de thèse est donc de prendre en compte les différences de capacités des robots (vitesse, limites physiques, etc.) et d'intégrer la notion de boucle fermée lors du transfert, pour adapter les stratégies de résolution de tâches selon les caractéristiques propres à chaque robot. Les méthodes proposées seront évaluées en simulation et en pratique.

1.3.3 Encadrements de stages

J'ai également eu l'opportunité d'encadrer des étudiants de Licence 3 à Master 2 dans le cadre de leur stage.

- 2014 : **Axelle Piot**, *Optimisation de mouvements complexes pour les robots à pattes*, étudiante de L3 Informatique Ecole Normale Supérieure de Paris,
- 2016 : **Amina Mohamedou**, *Reconstruction de mouvements à partir de séquences vidéos*, étudiante du Master 2 Recherche Robotique de l'Université Clermont Auvergne,
- 2016 : **Jessy Louis**, *Implémentation sur FPGA de modèles robotiques*, étudiant du département Génie Électrique de Polytech Clermont et du Master 2 Recherche Robotique de l'Université Clermont Auvergne,
- 2017 : **Akila Ben Charrada**, *Génération d'une séquence de postures pour la manipulation des objets*, étudiante du Master 2 Recherche Robotique de l'Université Clermont Auvergne,

6. Une demande de dérogation a été déposée à l'ED SPI, pour pouvoir commencer la thèse avant l'obtention de l'HDR.

— 2019 : **Mélodie Daniel**, *Contrôle multimodal de l'interaction humain-humanoïde*, étudiante du Master 2 **PAR**, UCA,

— 2019 : **Florent Audonnet**, *Interface de commande d'un robot (Interface USB Servomoteur à base de PIC)*, étudiant en BSc (Hons) Computing Science de l'université de Glasgow,

— 2020 : **Florent Audonnet**, *Modélisation et simulation d'un robot Hexapode sous ROS2*, étudiant en BSc (Hons) Computing Science de l'université de Glasgow,

— 2020 : **Samuel Beaussant**, *Machine learning and transfer learning in robotics : application to real world robots*, étudiant de Polytech Clermont Ferrand Génie Electrique en double cursus avec le Master 2 **PAR**, UCA,

— 2023 : **Antonio Claudio De Sousa Dos Santos Filho**, *Computer Vision, Reinforcement Learning and Skill Transfer in Robotic : Application to a Dynamic Task*, étudiant de l'Universidade Federal do Maranhão (UFMA), São Luís, Brésil et étudiant en 4ème année de Polytech Clermont Ferrand Génie Electrique (Programme Brafitec),

1.3.4 Publications

Mes publications depuis le début de ma carrière sont listées dans cette partie. Les noms soulignés correspondent aux étudiants encadrés.

Revues internationales

(1). 2024 : S. Beaussant, **S. Lengagne**, B. Thuilot, O. Stasse. *Towards Zero-Shot Cross-Agent Transfer Learning via Latent-Space Universal Notice Network.* Robotics and Autonomous Systems, In press, 184, pp.104862.

(2). 2024 : **S. Lengagne**. *Comparison of several basis functions for interval-based optimization*, soumis à International Journal of Approximate Reasoning.

(3). 2023 : M. Mounsif, **S. Lengagne**, B. Thuilot, L. Adouane. *Universal Notice Networks: Transferring Learned Skills Through a Broad Panel of Applications.* Journal of Intelligent & Robotic System, vol. 107, n° 18

(4). 2021 : M. Daniel, **S. Lengagne**, J.A. Corrales Ramon, Y. Mezouar. *General Framework for the Optimization of the Human-Robot Collaboration Decision-Making Process Through the Ability to Change Performance Metrics.* Frontiers in Robotics and AI, section Human-Robot Interaction vol. 8 article 736644

(5). 2020 : **S. Lengagne**, R. Kalawoun, F. Bouchon, Y. Mezouar. *Reducing pessimism in Interval Analysis using Bsplines Properties: Application to Robotics.* Reliable Computing Volume 27 pp. 63-87, July 2020

(6). 2014 : O. Cohen, S. Druon, **S. Lengagne**, A. Mendelsohn, R. Malach, A. Kheddar, D. Friedman. *fMRI-Based Robotic Embodiment: Controlling a Humanoid Robot by Thought Using Real-Time fMRI.* Presence : Teleoperators and Virtual Environments, 23 (3), 229-241

(7). 2013 : **S. Lengagne**, Joris Vaillant, and Eiichi Yoshida. *Generation of Whole-body Optimal Dynamic Multi-Contact Motions*. The International Journal of Robotics Research, page 17, April 2013.

(8). 2013 : J. Jovic, **S. Lengagne**, P. Fraisse, C. Azevedo-Coste. *Impact of Functional Electrical Stimulation of Knee Joints during Sitting Pivot Transfer Motion for Paraplegic People.* International Journal of Advanced Robotic Systems, ARS International, vol. 10, n° 25, 2013

(9). 2011 : **S. Lengagne**, N. Ramdani, P. Fraisse. *Planning and Fast Re-Planning Safe Motions for Humanoid Robots.* IEEE Transactions on robotics vol. 27 pages 1095-1106

Chapitre de livre

(10). 2022 : M. Mounsif, **S. Lengagne**, B. Thuilot L. Adouane. *Task-Specific Loss: A Teacher-Centered Approach to Transfer Learning Between Distincly Structured Robotic Agents.* Part of the Lecture Notes in Electrical Engineering book series (LNEE,volume 793)

Conférences internationales

(11). 2022 : M. Daniel, M. Aranda, L. Lequievre, **S. Lengagne**, J.A. Corrales Ramon, Y. Mezouar. *Robotic Control of the Deformation of Soft Linear Objects Using Deep Reinforcement Learning.* 2022 IEEE 18th International Conference on Automation Science and Engineering (CASE), Mexico City, Mexico, pp. 1516-1522

(12). 2021 : S. Beaussant, **S. Lengagne**, Benoit Thuilot, Olivier Stasse. *Delay Aware Universal Notice Network: Real world multi-robot transfer learning.* 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, pp. 1251-1258

(13). 2020 : M. Mounsif, **S. Lengagne**, B. Thuilot, L. Adouane.CoachGAN : *Fast Adversarial Transfer Learning between Differently Shaped Entities.* 17th International Conference on Informatics in Control, Automation and Robotics (ICINCO) (2020)

(14). 2020 : M. Mounsif, **S. Lengagne**, B. Thuilot, L. Adouane. *BAM ! Base Abstracted Modeling with Universal Notice Network:Fast Skill Transfer Between Mobile Manipulators.* 7th 2020 International Conference on Control, Decision and Information Technologies (IEEE-CoDIT 2020)

(15). 2019 : M. Mounsif, **S. Lengagne**, B. Thuilot, L. Adouane. *Universal Notice Network: Transferable Knowledge Among Agents.* IEEE - 6th 2019 International Conference on Control, Decision and Information Technologies (IEEE-CoDIT 2019), Apr 23-26, 2019, Paris - France

(16). 2018 : R. Kalawoun, **S. Lengagne**, F. Bouchon, Y. Mezouar. *BSplines properties with Interval Analysis for Constraint Satisfaction Problem: Application in robotics.* 15th International Conference on Intelligent Autonomous Systems IAS-15, Baden-Baden, Germany, June 2018

(17). 2018 : R. Kalawoun, **S. Lengagne**, Y. Mezouar. *Optimal robot base placements for coverage tasks.* 14th IEEE International Conference on Automation Science and Engineering (CASE 2018), Munich, Germany, August 20-24, 2018

(18). 2015 : J-C. Quinton, **S. Lengagne**. *Abstract planning over control primitives for robotic manipulation.* In International Conference on Spatial Cognition (ICSC)

(19). 2012 : **S. Lengagne**, O. Terlemez, S. Laturnus, T. Asfour, R. Dillmann. *Retrieving Contact Points Without Environment Knowledge.* 2012 IEEE-RAS International Conference on Humanoid Robots, Nov. 29th - Dec. 1st, Osaka, Japan

(20). 2012 : Y. Lee, **S. Lengagne**, A. Kheddar, Y. Kim. *Accurate Evaluation of a Distance Function for Optimization-based Motion Planning.* 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 7-12 October 2012, Algarve, Portugal

(21). 2012 : **S. Lengagne** , J. Jovic, C. Pierella, P. Fraisse, C. Azevedo Coste. *Generation of Multicontact Motions with Passive Joints: Improvement of Sitting Pivot Transfer Strategy for Paraplegics.* BioRob : Biomedical Robotics and Biomechatronics, Jun 2012, Roma, Italy. pp.1440-1445,

(22). 2012 : O. Cohen, S. Druon, **S. Lengagne**, A.Mendelsohn, R. Malach, A. Kheddar, D. Friedman. *fMRI based robotic embodiment: a pilot study.* IEEE/RAS-EMBS Int. Conf. on Biomedical Robotics and Biomechatronics (BioRob), 24-27 June 2012, Roma, Italia.

(23). 2011 : **S. Lengagne**, A. Kheddar, S.Druon, E. Yoshida. *Emulating Human Leg Impairments and Disabilities in Walking with Humanoid Robots.* 2011 IEEE International Conference on RObotics and BIOmimetics (IEEE-ROBIO). 7-11 December, 2011, Puket Island, Thailand.

(24). 2011 : **S. Lengagne**,A. Kheddar,E. Yoshida. *Considering Floatting Contact and Un-Modeled Effects for Multi-Contact Motion Generation.* Workshop on Humanoid Service Robot Navigation in Crowded and Dynamic Environments at the IEEE Humanoids Conference, 26-28 October 2011, Bled, Slovenia

(25). 2010 : **S. Lengagne**, Paul Mathieu, Abderrahmane Kheddar, Eiichi Yoshida. *Generation of Dynamic Motions Under Continuous Constraints: Efficient Computation Using B-Splines and Taylor polynomials.* IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS 2010) Oct 2010, Taipei, Taiwan.

(26). 2010 : **S. Lengagne**, P. Mathieu, A. Kheddar, E. Yoshida. *Generation of Dynamic Multi-Contact Motions: 2D case studies.* 2010 IEEE-RAS International Conference on Humanoid Robots, December 6-8, 2010, Nashville, TN, USA

(27). 2009 : **S. Lengagne**, N. Ramdani, P. Fraisse. *Safe motion planning computation for databasing balanced movement of Humanoid Robots.* IEEE International Conference on Robotics and Automation, ICRA 2009 Kobe, Japan. pp. 1669-1674

(28). 2009 : **S. Lengagne**, N. Ramdani, P. Fraisse. *Planning and Fast Re-Planning of Safe Motions for Humanoid Robots : Application to a Kicking Motion.* 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, St Louis, MO, USA. (Finalist of RoboCup Best Paper Award) pp. 441–446

(29). 2009 : S. Miossec, **S. Lengagne**, A. Kheddar, K. Yokoi. *Motion Optimization of Robots, Application to HRP-2.* BFG'09 - 14th Belgian-French-German Conference on Optimization, Sep 2009, Leuven, Belgium

(30). 2008 : **S. Lengagne**, N. Ramdani, P. Fraisse. *A new method for generating safe motions for humanoid robots.* IEEE International Conference on Humanoid Robots, Humanoids 2008. pp. 105-110

(31). 2007 : **S. Lengagne**, N. Ramdani, P. Fraisse. *Guaranteed computation of constraints for safe path planning.* IEEE International Conference on Humanoid Robots, Humanoids 2007, Pittsburg, PA, USA. pp. 312–317

Conférences nationales

(32). 2011 : **S. Lengagne**, A. Kheddar, S.Druon, E. Yoshida. *Programming Humanoid Motion to Reproduce Human Leg Injuries and Diseases.* 29th annual conference of the Robotics Society of Japan (RSJ), September 7-9, 2011, Tokyo, Japan .

(33). 2010 : **S. Lengagne**, P. Mathieu, A. Kheddar, E. Yoshida. *Multi-Contact Motion Generation: Continuous Constraints and Contact Forces.* 2010 : 28th annual conference of the Robotics Society of Japan (RSJ), September 22-24, 2010, Nagoya, Japan.

(34). 2008 : **S. Lengagne**, N. Ramdani, P. Fraisse. *Méthode pour la planification de trajectoires garanties.* Journées Francophones de Planification, Décision et Apprentissage pour la conduite de systèmes, Juin 2008, Metz, France

(35). 2008 : S. Miossec, **S. Lengagne**, A. Kheddar, K. Yokoi. *Motion optimization of robotic systems and validation of HRP-2 robot.* RSJ National Conference

Publications pédagogiques

(36). 2023 : **S. Lengagne** *Mise en pratique de compétences en robotique à partir d'un robot réel et de sa simulation.* CETSIS 2023 : Colloque de l'Enseignement des Technologies et des Sciences de l'Information et des Systèmes , republié dans J3eA, Journal sur l'enseignement des sciences et technologies de l'information et des systèmes.

(37). J. Laffont, **S. Lengagne**, *Mise en place de projets industriels pour des élèves ingénieurs* CETSIS 2023 : Colloque de l'Enseignement des Technologies et des Sciences de l'Information et des Systèmes , republié dans J3eA, Journal sur l'enseignement des sciences et technologies de l'information et des systèmes.

Thèse

(38). 2009 : **S. Lengagne**, *Planification et re-planification de mouvements sûrs pour les robots humanoïdes.* Université Montpellier II - Sciences et Techniques du Languedoc

Communications grand public

(39). 2012 : *Man and robot linked by brain scanner*, BBC

(40). 2012 : *Robot avatar body controlled by thought alone*, New Scientist

(41). 2011 : *Humanoid Robots find hurdles can help*, Publication Reuters (identification requise).

(42). 2010 : *Robots learn to walk like a senior citizen*, New scientist

1.3.5 Autres activités

Dans le cadre de mes activités de recherche, j'ai répondu à différentes demandes en tant que relecteur, éditeur associé ou membre de jury de thèse :

Éditeur associé : (les années sont entre parenthèses)

- IEEE-RAS International Conference on Humanoid Robots (2017,2018,2019,2020,2022),
- IEEE International Conference on Robotics and Automation (ICRA) (2019),
- European Conference on Mobile Robots (ECMR) (2019,2021,2023).

Relecteur : (le nombre de contributions est entre parenthèses)

Revues :

- Journal of Intelligent & Robotic Systems (16),
- IEEE Transactions on Robotics (12),
- Advanced Robotics (8),
- Robotics and Autonomous Systems (3),
- International Journal of Humanoid Robotics, Journal of Systems Architecture, Optimal Control Applications and Methods, IEEE Robotics and Automation Letters, Frontiers Humanoids and AI (1).

Conférences :

- IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (21),
- IEEE International Conference on Robotics and Automation (ICRA) (10),
- European Conference on Mobile Robots (ECMR) (7),
- IEEE-RAS International Conference on Humanoid Robots (4),
- International Symposium on Robot and Human Interactive Communication (RO-MAN), Robotics : Science and Systems" (RSS) (2),
- IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), Small Workshop on Interval Methods (SWIM), IEEE/SICE International Symposium on System Integration (SSI), ACM Symposium On Applied Computing (SAC), International Federation of Automatic Control World Congress (IFAC) (1).

Membre de jury de thèse :

- **Yohan Breux**, "Contribution à la modélisation d'environnement pour la robotique autonome en interaction avec l'humain " soutenue le 29 novembre 2018 au LIRMM, Montpellier, France.

Chapitre 2

Introduction

Mes activités de recherche se focalisent sur la problématique de génération rapide et garantie de mouvements complexes pour les robots manipulateurs, à pattes et humanoïdes. Ces travaux ont été réalisés via la supervision d'étudiants en stage et en doctorat tout en essayant de conserver une activité de recherche personnelle.

Les méthodes de génération de mouvements pour les robots déterminent leur capacité à interagir de manière autonome avec leur environnement. Cette thématique doit principalement prendre en compte trois aspects, comme montré sur la Figure 2.1 :

— **la complexité des mouvements** englobe la diversité des tâches que les robots peuvent accomplir, allant des mouvements simples aux actions plus sophistiquées. Elle est influencée par le nombre de degrés de liberté du robot, la nature de l'environnement dans lequel il opère, et les exigences spécifiques de la tâche pouvant nécessiter des capacités d'ordonnancement et d'anticipation ainsi qu'une dextérité accrue.

— **le temps de calcul** est un élément critique, en particulier dans des environnements en évolution rapide où la réactivité du robot est essentielle. Pour permettre une utilisation dans une boucle de commande ce temps de calcul doit être suffisamment faible (inférieur au dixième, voire au centième de seconde). Dans le cas d'un temps de calcul trop important, l'algorithme ne sera pas utilisé en temps réel, mais pourra s'exécuter hors ligne, de manière déportée ou en tant que superviseur de plus haut niveau. La constance du temps de calcul est également un aspect à prendre en compte, en effet certains algorithmes basés sur des méthodes itératives ou mettant en œuvre de l'aléatoire peuvent avoir des temps de calcul différents d'une itération à l'autre.

— **la garantie** est un impératif fondamental dans la gestion des mouvements des systèmes robotiques. Les mouvements générés doivent garantir la sécurité des robots, de leur environnement et des humains qui les entourent. Dans ce document, nous définissons la garantie du point de vue de la prévention des collisions, de la gestion des imprévus, de la prise en compte des incertitudes liées à l'environnement ou à la modélisation ainsi qu'à la recherche de l'optimum global en évitant les optima locaux. Les méthodes de génération de mouvements doivent être capables de garantir une performance sûre et fiable, même dans des conditions non optimales.

En se basant sur ces principales caractéristiques, représentées par les cercles sur la Figure 2.1, on peut définir l'algorithme idéal de génération de mouvements comme celui permettant de réaliser des tâches complexes en garantissant l'optimalité du mouvement,

la sécurité du robot et de son environnement dans un temps de calcul suffisamment faible pour permettre une implémentation en temps réel.

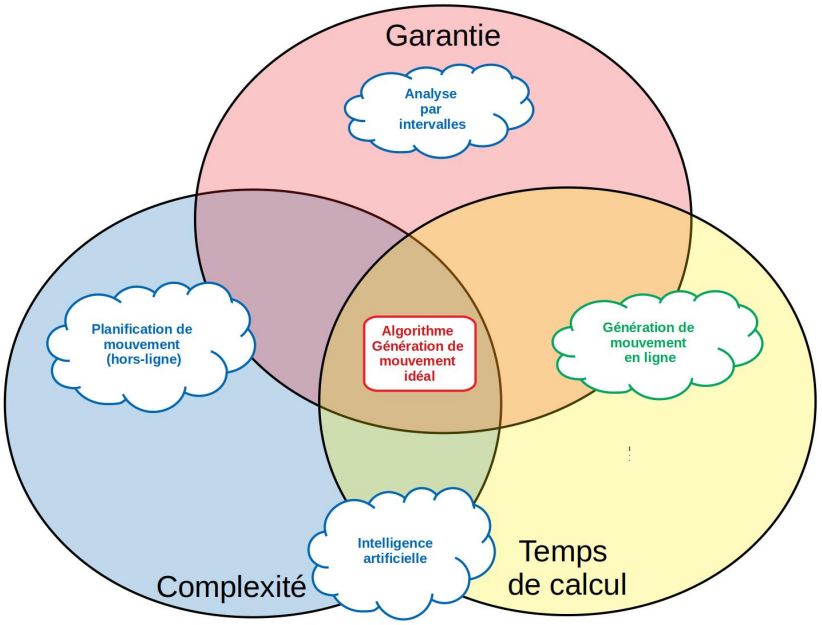


FIGURE 2.1 – Représentation de quelques outils utilisés pour la génération de mouvements en fonction de leur complexité, temps de calcul et garantie.

Les algorithmes de génération de mouvement en ligne, en vert sur la Figure 2.1, constituent une part importante des méthodes de l'état de l'art et reposent principalement sur des méthodes de commande. Ils permettent de générer en temps réel le mouvement pour atteindre un objectif/réaliser une tâche. Ces méthodes basées sur des modèles et des boucles d'asservissement, permettent un calcul en temps réel sur des tâches relativement simples. Ces méthodes permettent généralement d'avoir des preuves de stabilité. Il est cependant difficile de gérer des mouvements complexes uniquement avec une loi de commande. Dans le cadre de mes activités de recherche, j'ai volontairement orienté mes contributions sur les trois outils présentés en bleu dans la Figure 2.1 et présentés ci après :

— **La planification de mouvement** repose sur la prédétermination d'une séquence d'actions pour atteindre un objectif, en se basant sur un modèle du robot et de l'environnement. Elle regroupe la génération de trajectoire et l'ordonnancement d'actions plus haut niveau. Bien que conceptuellement simple, la complexité de cette méthode augmente avec le nombre de degrés de liberté du robot, la durée du mouvement et la complexité de l'environnement. Le temps de calcul interdit généralement une implémentation en temps réel, ce qui compromet la réactivité du robot face à des événements imprévus ou des incertitudes de modélisation.

— **L'Analyse par Intervalles(AI)**, et plus généralement les méthodes ensemblistes, constituent un outil qui offre une approche différente. En bornant les incertitudes l'AI permet d'assurer le respect des contraintes et la génération de l'optimum global. Néanmoins, l'utilisation de ces méthodes peut mener à une sur-évaluation des fonctions par rapport à leur vraie valeur. Ce phénomène, appelé pessimisme, peut limiter les performances du robot ou produire des temps de calcul prohibitifs. L'utilisation la plus courante des méthodes ensemblistes est donc hors ligne lors d'étapes de

planification, de validation a posteriori ou de validation du comportement souhaité vis-à-vis des incertitudes.

— **L'Intelligence Artificielle (IA)**, en particulier les approches basées sur le *Machine Learning* et les réseaux de neurones, a montré des résultats prometteurs dans la génération de mouvements. Cependant, la complexité de ces modèles, combinée aux exigences de calcul lors de la phase d'apprentissage, peut rendre leur déploiement dans des environnements réels difficile. Le principal inconvénient de ces réseaux est qu'ils sont généralement vus comme des boîtes noires sur lesquelles il est difficile de garantir un bon fonctionnement dans toutes les situations. Ces méthodes sont généralement dédiées à une tâche spécifique réalisée par un robot particulier et sont difficilement généralisables.

En conclusion, chaque méthode de génération de mouvements pour les robots présente des compromis entre la complexité, le temps de calcul et la garantie. Certaines solutions hybrides, par exemple associant l'analyse par intervalles aux résultats de planification, permettent de profiter des avantages de chaque méthode en minimisant les inconvénients. Dans ce contexte, et afin de présenter une sélection des travaux réalisés en lien avec la génération de mouvement en robotique, la suite du manuscrit est organisée de la façon suivante. Le Chapitre 3 présente principalement mes contributions antérieures à ma prise de poste actuelle, à savoir, la mise en place d'algorithmes de génération de mouvements permettant de considérer des mouvements plus complexes et la validation sur des mouvements dynamiques et multi-contacts. Dans le cadre de ma recherche propre, j'ai proposé une amélioration des algorithmes ensemblistes permettant de réduire le nombre d'itérations et donc le temps de calcul pour les processus d'optimisation globale, qui est proposée dans le Chapitre 4. Finalement, les travaux les plus récents, menés par mes derniers doctorants, se basant sur des structures de réseaux de neurones sont présentés dans le Chapitre 5, dans l'optique de diminuer les temps d'apprentissage en transférant les compétences d'un robot à un autre. Finalement, le Chapitre 6 conclut sur les travaux réalisés et discute des différentes perspectives d'évolution de ces travaux.

Chapitre 3

Génération de mouvements complexes

3.1 Contexte

Dès le début, mes activités de recherche se sont orientées vers la génération de mouvements complexes avec l'optimisation du mouvement de coup de pied pour le robot HRP-2 [127, 128]. Là où la plupart des recherches se focalisaient sur des approches utilisables en ligne, généralement basées sur des modèles simplifiés [86] pour des mouvements de marche, je m'intéressais à exploiter au maximum les capacités physiques (mécaniques) du robot afin de générer des mouvements hors de portée des méthodes de l'état de l'art pour des tâches de locomotion et manipulation dans des environnements complexes et encombrés. Ces travaux se sont déroulés entre 2006 et 2013 dans les laboratoires du LIRMM, du JRL et du KIT.

Afin de pouvoir appréhender la génération de mouvements, il est intéressant de considérer les ensembles de mouvements suivants :

— l'ensemble des mouvements du robot : $\mathcal{M}$,

— l'ensemble des mouvements faisables respectant les limitations propres du robot telles que les butées, les auto-collisions : $\mathcal{F}$,

— on peut définir un ensemble de mouvements $\mathcal{D}$ ne respectant pas les limites et pouvant être potentiellement dangereux pour l'intégrité du robot ou de son environnement :

$$\mathcal{D} \cup \mathcal{F} = \mathcal{M}, \text{ et } \mathcal{F} \cap \mathcal{D} = \emptyset \tag{3.1}$$

— l'ensemble des mouvements pouvant être générés par des méthodes basées sur des modèles complexes $\mathcal{C}$ respectant les limitations propres du robot tels que les butées et les couples articulaires ou les auto-collisions :

$$\mathcal{C} \subseteq \mathcal{F} \tag{3.2}$$

— l'ensemble des mouvements pouvant être générés par des méthodes basées sur des modèles simples (qui respectent des hypothèses de modélisation) : $\mathcal{S}$. Si on suppose que tous les mouvements générés via des modèles simples peuvent également être générés avec des modèles complets, on peut considérer que :

$$\mathcal{S} \subseteq \mathcal{C} \subseteq \mathcal{F} \tag{3.3}$$

L'ensemble $\mathcal{S}$ contient principalement les mouvements de manipulation et de marche pour les robots humanoïdes.

— Le choix d'une méthode de commande va également conditionner l'ensemble des mouvements contrôlables. Dans mes recherches, je suppose que les mouvements générés pourront être réalisés via l'algorithme de commande.

L'objectif de mes travaux de recherche est donc de proposer une méthode permettant de générer un ensemble de mouvements $\mathcal{L}$, principalement ceux qui ne sont pas réalisables via un modèle simplifié [1], comme les mouvements de locomotion mettant en œuvre les membres supérieurs :

$$\text{Trouver : } \mathcal{L} \subset \mathcal{C} \subseteq \mathcal{F}, \text{ avec } \mathcal{L} \not\subseteq \mathcal{S} \text{ et } |\mathcal{C}| \gg |\mathcal{S}| \tag{3.4}$$

Cette particularité de mes travaux à générer des mouvements non réalisables par les méthodes de l'état de l'art a provoqué, chez mes collègues du JRL, une certaine appréhension vis-à-vis de l'intégrité du robot. Les mouvements générés ont été appliqués au robot un grand nombre de fois, pour les besoins de la rédaction des différents articles mais aussi pour des démonstrations, sans provoquer de dysfonctionnement direct au robot. Par contre, nous avons pu constater l'accentuation d'une faiblesse mécanique au niveau de la hanche qui avait déjà été remarquée sur d'autres robots HRP-2.

3.2 Positionnement scientifique

En 2019, Boston Dynamics met en ligne une vidéo [2] de son robot ATLAS réalisant des figures de parkour [3], notamment un salto arrière. Malheureusement, aucune information n'a été publiée sur les méthodes mises en place. On peut supposer que ces mouvements se basent à la fois sur des algorithmes de planification et de contrôle du mouvement pour exploiter au mieux les capacités physiques hors normes du robot, comme présenté dans [27]. En se focalisant davantage sur l'aspect planification, on peut identifier trois méthodes en fonction de la complexité du modèle considéré.

3.2.1 Planification à partir de modèles simples

Ces premières méthodes de planification utilisent des modèles qui assimilent le robot à une masse ponctuelle [85] et utilisent des critères de stabilité basiques tels que le *Zero Moment Point* (ZMP) [173]. Bien que limités dans leur capacité à générer des mouvements complexes, ces modèles simplifiés sont extrêmement utiles pour des tâches basiques, comme la marche sur sol plat. Cependant, un modèle trop simplifié ignore toutes les contraintes cinématiques et dynamiques et considère que le robot est seulement soumis à des forces de contact unilatérales sur les pieds. Une des limitations concerne le modèle de masse ponctuelle qui suppose que le moment angulaire autour du centre de masse est nul, ce qui ne permet pas de modéliser l'effet du balancement des bras lors de la marche [82]. Le déplacement d'un robot sur un terrain irrégulier (non plat) ou composé d'obstacles nécessite des modifications au niveau du contrôle et de la planification [6, 145]. Ce type de méthode est difficilement applicable à des mouvements plus complexes comme le saut [26] ou l'escalade [171].

1. La notation $|F| \gg |S|$ indique qu'on souhaite avoir un ensemble de mouvement F bien plus grand que l'ensemble de mouvement S

2. `https://www.youtube.com/watch?v=_sBBaNYex3E`

3. Le parkour est une méthode d'entraînement pour franchir toutes sortes d'obstacles dans des environnements urbains ou naturels : `https://fr.wikipedia.org/wiki/Parkour`.

3.2.2 Planification *kino-dynamique*

Pour pallier ces problèmes, d'autres approches combinent la modélisation cinématique (étude des mouvements sans tenir compte des forces qui les provoquent) et la modélisation dynamique (étude des forces en mouvement). L'aspect cinématique consiste à générer les trajectoires articulaires permettant la réalisation du mouvement comme le bon placement des points de contact (pied par exemple) aux endroits désirés, alors que l'aspect dynamique considère, généralement, un modèle simplifié du robot comme le *Centroidal angular momentum* qui correspond à la rotation du robot autour de son centre de masse [27, 33]. Cette méthode permet de générer des mouvements plus complexes tels que des mouvements de sauts acrobatiques comme présenté dans [27], mais ne prend pas en compte les limites dynamiques du robot tels que les couples articulaires maximum.

3.2.3 Planification à partir du modèle dynamique complet

Afin de résoudre ces problèmes, nous avons proposé une méthode, publiée dans [111], qui utilise toutes les contraintes présentées dans la Section 3.3 et formule le problème comme indiqué en Section 3.4. Nous avons généré des mouvements complexes, dynamiques et multi-contacts pour les robots humanoïdes et les avatars virtuels comme illustrés dans la Section 3.6. Ces travaux ont depuis été portés sur GPU afin de paralléliser le calcul des contraintes tout au long du mouvement afin d'accélérer la résolution du problème [28, 29].

Les mouvements générés se représentent par des trajectoires en position, vitesse et couple articulaire à appliquer au robot. Il est important de noter qu'un simple asservissement articulaire permet d'exécuter les trajectoires en position et en vitesse avec une précision satisfaisante. Par contre, les trajectoires en couple, qui servent à assurer des efforts internes et l'équilibre nécessitent la mise en place d'un contrôle en couple qui n'était pas encore opérationnel pour les robots humanoïdes à ce moment, même si des travaux récents [39, 43, 151] proposent une solution pour pallier ce problème.

3.3 Modélisation et contraintes du mouvement

Dans cette section, nous allons rapidement évoquer l'aspect modélisation d'un robot à pattes ainsi que les différentes limites à respecter afin de garantir l'intégrité et la sécurité du robot et de son environnement lors de la réalisation d'un mouvement.

3.3.1 Particularité des robots à pattes

Pour définir la posture d'un robot humanoïde, ou plus généralement d'un robot à pattes (donc mobile), il est fréquent de considérer les N_a articulations actives q_a (commandables) du robot ainsi que 6 articulations virtuelles q_v permettant de définir la pose d'un corps du robot, (le plus souvent le bassin), dans un repère de référence, généralement désigné comme le repère monde, comme montré dans la Figure 3.1.

Dans le cadre de la génération de mouvements pour les robots mobiles, l'espace de recherche est de dimension $N_a + 6$, ce qui peut amener à des espaces de dimension 30 à 40, voire plus contrairement à des robots manipulateurs pour lesquels la dimension de l'espace de recherche dépasse rarement 10. L'autre principale différence réside dans la notion d'équilibre, décrite dans la Section 3.3.5, en effet les 6 articulations virtuelles n'étant pas directement contrôlables il faut prendre plusieurs précautions afin d'en garder le contrôle

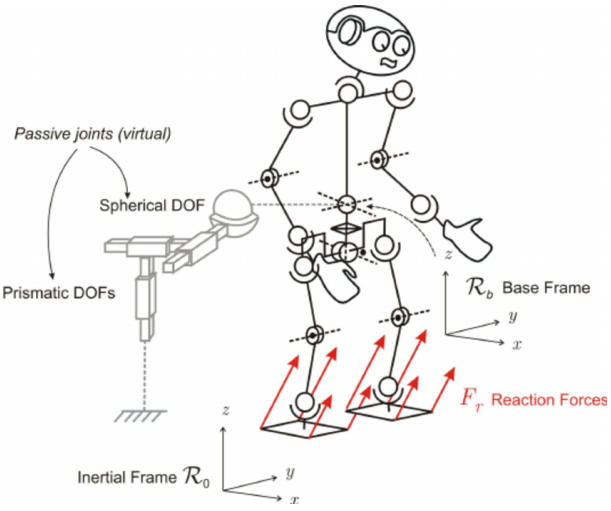


FIGURE 3.1 – Illustration des articulations virtuelles permettant de définir la pose d'un robot mobile. Source [91].

via les autres articulations afin de ne pas provoquer une perte d'équilibre, contrairement aux robots manipulateurs qui sont généralement ancrés à leur environnement de travail.

3.3.2 Modélisation géométrique et dynamique

En supposant des trajectoires articulaires et des efforts extérieurs définis, les modèles géométrique et dynamique permettent de calculer la pose, la vitesse et l'accélération de repères fixés à différents corps du robot ainsi que les couples articulaires. Ces modèles sont généralement obtenus via des méthodes récursives comme celles présentées dans [161]. Le modèle dynamique inverse du robot peut se mettre sous la forme suivante :

$$\begin{bmatrix} \Gamma \\ 0^6 \end{bmatrix} = \begin{bmatrix} \mathbb{M}_1(q) \\ \mathbb{M}_2(q) \end{bmatrix} \ddot{q} + \begin{bmatrix} \mathbf{H}_1(q,\dot{q}) \\ \mathbf{H}_2(q,\dot{q}) \end{bmatrix} + \begin{bmatrix} \mathbb{J}_1^T(q) \\ \mathbb{J}_2^T(q) \end{bmatrix} \mathbf{F} \tag{3.5}$$

où $q(t) \in \mathbb{R}^{(N_a+6)}$ est un vecteur contenant les (N_a+6) positions articulaires $(q_i(t))$, $\dot{q}(t) \in \mathbb{R}^{(N_a+6)}$ et $\ddot{q}(t) \in \mathbb{R}^{(N_a+6)}$ rassemblent les vitesses et accélérations articulaires, $\Gamma(t) \in \mathbb{R}^{N_a}$ est le vecteur des couples des articulations actives, $\mathbf{F}(t) = \{F_1, F_2, \ldots, F_{N_f}\} \in \mathbb{R}^{3N_f}$ est le vecteur des N_f forces de contact linéaires actives, $\mathbb{M}_1(q) \in \mathbb{R}^{N_a \times (N_a+6)}$, $\mathbb{M}_2 \in \mathbb{R}^{6\times(N_a+6)}$ sont les deux composantes de la matrice inertielle, $\mathbf{H}_1 \in \mathbb{R}^{N_a}$ et $\mathbf{H}_2 \in \mathbb{R}^6$ les deux composantes du vecteur dû à la gravité, aux effets centrifuges et de Coriolis, $\mathbb{J}_1 \in \mathbb{R}^{N_a \times 3N_f}$ et $\mathbb{J}_2 \in \mathbb{R}^{6\times 3N_f}$ sont les composantes de la matrice jacobienne. Il est important de noter que les six efforts liés aux articulations virtuelles doivent être nuls (d'où le 0^6 dans l'Équation (3.5)).

Pour qu'un mouvement soit réalisable et garantisse l'intégrité du robot et de son environnement il faut considérer plusieurs limites concernant les efforts de contact ainsi que les positions, vitesses, accélérations et couples articulaires.

3.3.3 Contraintes articulaires

Les contraintes articulaires regroupent les valeurs limites des positions, vitesses, accélérations et couples articulaires admissibles par le robot et doivent être validées sur toute

la durée du mouvement $[\Delta t]$, tels que :

$$\forall i \in \{1, \ldots, N_a\}, \forall t \in [\Delta t] \quad \underline{q_i} \leq q_i(t) \leq \overline{q_i} \tag{3.6}$$

$$\forall i \in \{1, \ldots, N_a\}, \forall t \in [\Delta t] \quad \underline{\dot{q}_i} \leq \dot{q}_i(t) \leq \overline{\dot{q}_i} \tag{3.7}$$

$$\forall i \in \{1, \ldots, N_a\}, \forall t \in [\Delta t] \quad \underline{\ddot{q}_i} \leq \ddot{q}_i(t) \leq \overline{\ddot{q}_i} \tag{3.8}$$

$$\forall i \in \{1, \ldots, N_a\}, \forall t \in [\Delta t] \quad \underline{\Gamma_i} \leq \Gamma_i(t) \leq \overline{\Gamma_i} \tag{3.9}$$

Généralement les valeurs maximales en position, vitesse et couple sont définies par le constructeur du robot, ce qui est plus rare pour les limites en accélération.

3.3.4 Contraintes géométriques

En se basant sur la modélisation présentée dans la Section 3.3.2, cette section présente comment les contraintes géométriques permettent de définir un mouvement. Dans ce contexte, nous supposons uniquement des contacts sans glissement.

Un contact i peut être défini par des contraintes de type égalité entre la pose d'un repère fixe sur l'environnement $\mathbf{X}_i^e$ et la pose d'un repère $\mathbf{X}_i$ attaché au corps du robot en contact, comme montré sur la Figure 3.2. Où $\mathbf{X}$ est défini comme un vecteur de six composantes contenant la position et l'orientation ($\mathbf{X} = [x, y, z, \theta_x, \theta_y, \theta_z]^T$) du repère i dans un repère de référence. On peut noter que la pose du repère $\mathbf{X}_i^e$ peut être connue ou composée de paramètres d'optimisation pour laisser "libres" certaines coordonnées du contact (comme la position ou l'orientation par rapport à la normale d'un plan)

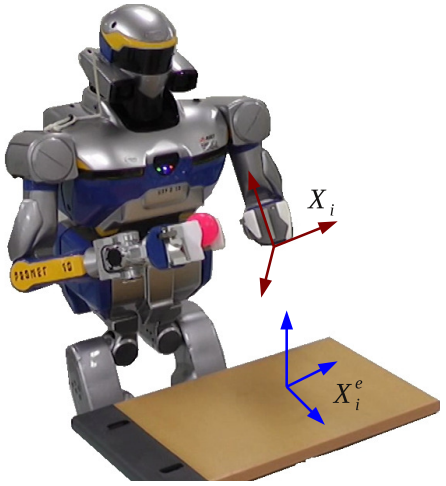


FIGURE 3.2 – Description du contact de la main gauche du robot HRP-2 avec la table. Le repère $\mathbf{X}_i$ est fixé à la main gauche du robot et doit coïncider avec le repère $\mathbf{X}_i^e$ situé sur la table. La position du repère $\mathbf{X}_i^e$ peut être imposée ou issue de processus d'optimisation.

Pour considérer un ensemble de N_c contacts entre le robot et l'environnement durant un intervalle de temps $[\Delta t]$, il faut considérer un ensemble de contraintes égalités continues telles que :

$$\forall t \in [\Delta t], \forall i \in \{1, \ldots, N_c\} \quad \mathcal{D}_i(\mathbf{X}_i(t), \mathbf{X}_i^e) = 0 \tag{3.10}$$

Avec $\mathcal{D}_i$ une fonction de distance générique qui permet de considérer des contacts planaires, ponctuels ou linéaires (rotation autour d'une arête). Le respect de ces contraintes permettra d'imposer la partie géométrique du mouvement. Afin de prendre en compte

la dynamique du mouvement et de garantir l'équilibre il faut considérer les efforts dynamiques comme présenté dans la partie suivante.

3.3.5 Équilibre

L'équilibre des robots humanoïdes est souvent décrit en utilisant le *Zero Moment Point* [173] avec un modèle simplifié assimilant le robot à un pendule inverse. Cette méthode, bien que très utilisée pour contrôler les mouvements de marche [84] n'est pas adaptée en cas de contacts autres qu'entre les pieds et le sol. Pour qualifier l'équilibre de manière plus générale, nous reprenons la notion du *Contact Wrench Sum*(CWS)[70] qui s'assure que les forces de contact évitent tout décollement ou tout glissement non désiré. Dans le cas de mouvement multi-contacts et connaissant les trajectoires articulaires $q(t)$, il existe une indétermination des couples Γ en fonction des efforts de contact $\mathbf{F}$, ce qui se traduit par l'existence d'un grand nombre d'efforts internes différents possibles. Certains efforts internes permettront de garantir l'équilibre du robot, d'autres provoqueront la chute ou au minimum la perte d'un contact. Nous avons proposé une méthode pour gérer cette indétermination lors de la génération du mouvement. Pour cela, focalisons-nous sur la deuxième partie de l'Équation (3.5) :

$$0^6 = \mathbb{M}_2(q)\ddot{q} + \mathbf{H}_2(q,\dot{q}) + \mathbb{J}_2^T(q)\mathbf{F} \tag{3.11}$$

et définissons le vecteur des effets dynamiques :

$$\mathbf{D}_2 = \mathbb{M}_2(q)\ddot{q} + \mathbf{H}_2(q,\dot{q}) \tag{3.12}$$

Nous considérons l'ensemble de N_f forces linéaires $\mathbf{F} = \{F_1, F_2, \ldots, F_{N_f}\}$ pour chaque corps du robot en contact avec l'environnement comme montré sur la Figure 3.3. Pour éviter tout glissement ou décollement inattendu d'un corps en contact, toutes les forces F_i doivent valider les contraintes de non-glissement et de non-décollement :

$$\forall t \in [\Delta t], \forall i \in \{1, \ldots, N_f\} \quad \begin{cases} F_i^n(t) > 0 \\ ||F_i^t(t)||^2 \leq \mu_i^2 F_i^n(t)^2 \end{cases} \tag{3.13}$$

où F_i^n est la composante normale de la force de contact, F_i^t est le vecteur 2D de la force tangente et μ_i est le coefficient de frottement au contact i. Sans oublier que les forces de contact doivent compenser les effets dynamiques en satisfaisant :

$$\mathbf{D}_2(q,\dot{q},\ddot{q}) + \mathbb{J}_2^T(q)\mathbf{F} = 0^6 \tag{3.14}$$

Afin de trouver des efforts de contact compensant les effets dynamiques et incitant le non-glissement et le non-décollement des corps en contact nous avons mis en place une pseudo-inverse $\Omega(\mathbb{J}_2(q), \gamma)$ pondérée telle que :

$$\mathbf{F} = \Omega(\mathbb{J}_2(q), \gamma).\mathbf{D}_2(q,\dot{q},\ddot{q}) \tag{3.15}$$

Où γ sont des coefficients de répartition des efforts faisant partie des paramètres d'optimisation afin d'optimiser les efforts internes. Le détail du calcul de $\Omega(\mathbb{J}_2(q), \gamma)$ est présenté dans [111], également disponible en Annexe B.1. Cette formulation incite les forces de contact à être le plus proche possible de la normale, mais une vérification des contraintes de non-glissement et non-décollement demeure nécessaire via l'ajout des contraintes de l'Équation (3.13). Cette formulation permet de s'affranchir des contraintes égalités de l'Équation (3.11) qui sont difficilement intégrables dans le processus d'optimisation.

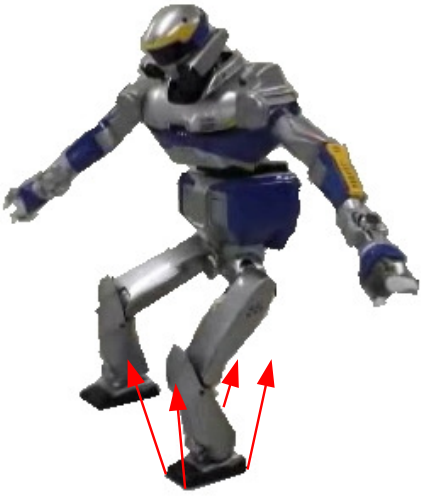


FIGURE 3.3 – Description des forces de contact pour le pied gauche. Nous considérons 4 forces linéaires, une pour chaque coin de la surface de contact.

3.3.6 Collisions et auto-collisions

Lors des phases de contact, la collision définie à l'avance entre un corps du robot et son environnement est inévitable et même souhaitée. Cependant, dans les autres cas, les collisions entre le robot et l'environnement et les auto-collisions (collision entre deux corps du robot) doivent être évitées. Pour cela, on suppose que tous les corps du robot peuvent être définis comme un ensemble de N_l corps convexes. Ensuite en considérant $\mathcal{P}_v$ le v-ième corps convexe du robot et $\mathcal{O}_w$ le w-ième obstacle à éviter, on peut décrire l'évitement des collisions comme suit :

$$\forall t \in [\Delta t],\ \forall \{v, w\}\ \ d(\mathcal{P}_v, \mathcal{O}_w)(t) \geq \epsilon \tag{3.16}$$

où $d(X, Y)(t)$ est la distance séparant le corps X du corps Y, la paire v, w est la liste des liens/obstacles définis à considérer et $\epsilon > 0$ une marge de sécurité. Cette contrainte signifie que la distance de séparation entre deux corps quelconques doit rester positive (à la marge ϵ-près) tout au long du mouvement. Les auto-collisions sont contrôlées de la même manière :

$$\forall t \in [\Delta t],\ \forall v, \forall w\ \ d(\mathcal{P}_v, \mathcal{P}_w)(t) \geq \epsilon\ \ v \neq w \text{ et } v \neq ante(w) \tag{3.17}$$

en ignorant de vérifier la distance entre un corps et lui-même ($v \neq w$) et entre un corps et son antécédent auquel il est mécaniquement relié (donc en collision) ($v \neq ante(w)$). Il existe plusieurs méthodes pour calculer la distance entre des paires d'objets géométriques. Plus la géométrie est simple, plus le calcul est rapide. En robotique, nous utilisons des volumes limites pour les liens (par exemple, des sphères, des capsules, des boîtes limites orientées, etc.). Dans le cadre de notre méthode, nous utilisons des algorithmes d'optimisation qui nécessitent une continuité du gradient des contraintes (et du critère). Dans ces travaux, nous avons choisi d'utiliser les *Sphere-Tori-Patches Bounding Volume*(STP-BV), qui permettent d'obtenir un calcul de la distance rapide et un gradient C^1 continu [10, 52].

Cependant, des travaux récents pourraient être considérés pour déterminer la contrainte de non-collision, par exemple ceux de [20] qui formalise la non collision par l'existence d'un

plan entre deux objets convexes. Cette méthode a été appliquée dans le cadre de la génération de mouvements pour un robot manipulateur [186].

3.4 Formulation du problème de génération de mouvements

Dans la section précédente, nous avons détaillé les contraintes entrant en jeu lors de la planification de mouvements pour des robots humanoïdes (ou plus généralement à pattes). Dans cette partie, nous détaillons comment formuler le problème d'optimisation qui pourra être résolu par des algorithmes existants (C-FSQP [102] et IPOPT [174] dans notre cas).

3.4.1 Problème infini de la génération de mouvements

D'une manière générale, on peut définir le problème de génération de mouvements, comme étant la recherche d'une trajectoire articulaire $q(t)$ qui garantit la sécurité du robot, induit un comportement désiré et minimise un critère comme défini par :

$$\begin{array}{cc} \text{trouver : } q(t) & \text{qui minimise: } \mathcal{C}\left(q(t)\right) \\ \text{avec:} & \\ \forall i \in 1,\ldots,N_i, \forall t \in [\Delta_i] & g_i(q(t)) \leq 0 \\ \forall j \in 1,\ldots,N_j, \forall t \in [\Delta_j] & h_j(q(t)) = 0 \\ \forall k \in 1,\ldots,N_k & z_k(q(t_k)) \leq 0 \end{array} \tag{3.18}$$

Avec $\mathcal{C}$ le critère à minimiser, qui peut considérer différentes grandeurs telles que la durée du mouvement [13, 146], les couples articulaires [18], la consommation d'énergie globale [128], la variation des accélérations [147] ou la variation des couples articulaires [170], ou toute combinaison pondérée. N_i et N_j sont les nombres de contraintes inégalités g_i et égalités h_j qui doivent être respectées sur les intervalles de temps $[\Delta_i]$ et $[\Delta_j]$. Lors d'un déplacement, les phases d'appui sont généralement décrites via les N_j contraintes égalités (pour les contraintes géométriques). De plus certains aspects du comportement (position et vitesse du bras lors d'un geste de lancer par exemple) sont généralement décrits via les N_k contraintes ponctuelles qui doivent être respectées à un instant donné.

Le problème défini par l'Équation (3.18) est appelé un *Infinite Programming Problem* [2] car il consiste à trouver une trajectoire, assimilable à une infinité de points discrets, validant des contraintes continues, elles-aussi assimilables à une infinité de contraintes discrètes.

Pour résoudre ce problème, on peut réduire ces infinités de valeurs et contraintes en un grand nombre de valeurs discrètes via des *multiples shooting methods*[27, 45, 149] qui cherchent l'état du système et la commande à appliquer à chaque instant en ajoutant des contraintes de continuité liées au modèle dynamique comme indiqué sur la Figure 3.4.

La principale difficulté de cette méthode est la détermination du nombre de points à considérer quand la durée du mouvement n'est pas définie, ainsi que le grand nombre de contraintes égalités à satisfaire qui peut être difficilement gérable par l'algorithme d'optimisation. Dans notre cas, nous préférons utiliser une paramétrisation du mouvement.

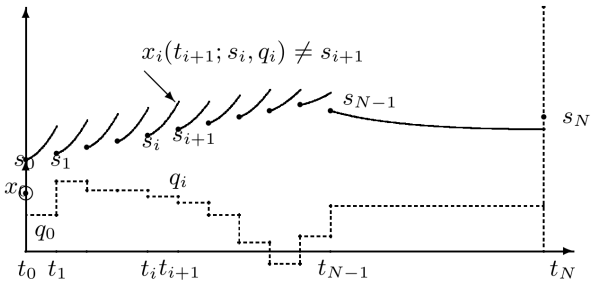


FIGURE 3.4 – Illustration des contraintes de continuité. Source [120].

3.4.2 Paramétrisation du mouvement

Pour résoudre les problèmes liés à l'infinité d'une trajectoire continue, une méthode est de paramétrer les trajectoires afin d'obtenir :

$$q(t) = p(\alpha, T_f, t) \tag{3.19}$$

Où α et T_f représentent les paramètres et la durée des trajectoires (ou de morceaux de trajectoires) et p une fonction de paramétrisation. Il est important de noter, que le choix de la paramétrisation va restreindre l'ensemble $\mathcal{C}$ des mouvements du robot de l'Équation (3.2) à un ensemble de mouvements paramétrables $\mathcal{P}$ et qu'un mauvais choix de paramétrisation peut exclure les mouvements désirés de $\mathcal{P}$ et de ce fait rendre la génération de mouvements impossible. En utilisant l'Équation (3.19) dans la définition du problème de la génération de mouvements (Équation (3.18)) on obtient :

$$\begin{array}{cc} \text{trouver : } \alpha, \beta, T_f & \text{qui minimise: } \mathcal{C}(\alpha, \beta, T_f) \\ \text{avec:} & q(t) = p(\alpha, T_f, t) \\ \forall i, \forall t \in [\Delta_i] & g_i(q(t), \beta) \leq 0 \\ \forall j, \forall t \in [\Delta_j] & h_j(q(t), \beta) = 0 \\ \forall t_k & z_k(q(t_k), \dot{q}(t_k), \ddot{q}(t_k), \beta) \leq 0 \end{array} \tag{3.20}$$

Avec β un ensemble de variables d'optimisation additionnel qui peuvent être utilisées pour optimiser la position du point d'appui comme indiqué en Section 3.3.4 ou la répartition des efforts de contacts comme indiqué dans la Section 3.3.5. Le problème obtenu à l'Équation (3.20) est un *Semi Infinite Programming Problem*[154] pour lequel l'espace de recherche est de dimension finie ($dim(\alpha) + dim(\beta) + dim(T_f)$) et qui doit valider les contraintes sur des intervalles de temps continu, assimilables à une infinité d'instants discrets. La gestion des contraintes continues sera présentée dans la Section 3.5.4.

En nous basant sur des travaux précédents [103], nous avons choisi de façonner les trajectoires articulaires avec des courbes B-Splines cubiques [4] uniformes. Chaque trajectoire articulaire $q_i(t)$ est la somme pondérée de fonctions de base d'ordre $k-$ définies par un nombre m de points de contrôle :

$$\forall t \in [\Delta t] \quad q_i(t) = \sum_{j=1}^{m} b_j^k(t) \alpha_{i,j} \tag{3.21}$$

Les fonctions de base $b_j^k(t)$ sont calculées à l'aide de la récursion de Cox-de-Boor [42] et les points de contrôle $\alpha_{i,j}$ font partie des paramètres α du mouvement. Au final, le

4. on considérera $k = 3$ dans l'Équation (3.21)

mouvement est décomposé en phases d'appui, pour lesquelles la durée est décomposée en plusieurs intervalles de temps. Les trajectoires articulaires sont construites à partir des fonctions de base comme montré sur la Figure 3.5. Ce choix fait qu'à chaque instant une trajectoire articulaire ne dépend que de 4 paramètres par articulation, permettant de réduire la dépendance des contraintes à un nombre limité de paramètres d'optimisation ce qui réduit les erreurs numériques internes à l'algorithme d'optimisation lui permettant de trouver l'optimum plus efficacement.

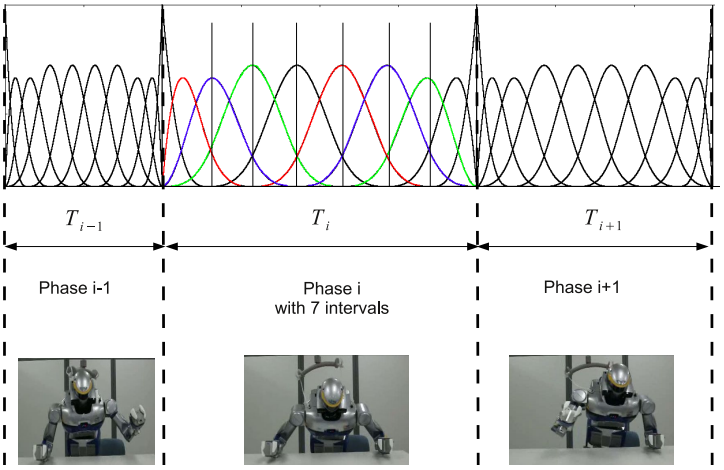


FIGURE 3.5 – llustration de la décomposition du mouvement en plusieurs phases et intervalles de temps.

Au final, on peut regrouper toutes les variables d'optimisation dans un seul vecteur χ et reformuler le problème de l'Équation (3.20) en :

$$\begin{array}{cc} \text{trouver : } \chi & \text{qui minimise: } \mathcal{C}(\chi) \\ \forall i, \forall t \in [\Delta_i] & g_i(\chi, t) \leq 0 \\ \forall j, \forall t \in [\Delta_j] & h_j(\chi, t) = 0 \\ \forall t_k & z_k(\chi, t_k) \leq 0 \end{array} \tag{3.22}$$

On notera que les choix de modélisation (notamment des efforts) et la paramétrisation du mouvement font que l'algorithme impose qu'au moins un corps soit en contact avec l'environnement à chaque instant. Il exclut donc les mouvements de saut ou de course.

3.5 Gestion des contraintes continues

Dans les Sections 3.3 et 3.4, nous avons défini les contraintes à prendre en compte pour permettre de garantir l'intégrité du robot et réaliser la tâche souhaitée (dans notre cas un déplacement) et introduit la paramétrisation du mouvement. Dans cette section, nous allons voir comment formuler ces contraintes continues, qui doivent être validées sur un intervalle de temps, en un nombre fini de contraintes assimilables par les algorithmes d'optimisation.

3.5.1 Contraintes sur les trajectoires articulaires

Comme indiqué précédemment, les trajectoires articulaires sont calculées via les fonctions de base $b_j^k(t)$ construites suivant la récursion de Cox-de-Boor [42]. Ces fonctions de

base ont les deux propriétés suivantes :

$$\forall k, \forall t \in [\Delta t] \quad \sum_{j=1}^{m} b_j^k(t) = 1.0 \tag{3.23}$$

et :

$$\forall j, \forall k, \forall t \in [\Delta t] \quad 0 \leq b_j^k(t) \leq 1.0 \tag{3.24}$$

Ces propriétés font que la fonction $q_i(t)$, décrite par les m points de contrôle $\alpha_i^m = \{\alpha_{i,1}, \ldots, \alpha_{i,m}\}$ comme défini dans l'Équation (3.21) est entièrement contenue dans l'intervalle formé par les extrema des points de contrôle donc on peut en déduire que :

$$\begin{array}{ll} \text{si} & \forall j \in [1, m] \;\; \underline{q_i} \leq \alpha_{i,j} \leq \overline{q_i} \\ \text{alors} & \forall t \in [\Delta t] \;\; \underline{q_i} \leq q_i(t) \leq \overline{q_i} \end{array} \tag{3.25}$$

Cette particularité des fonctions de base, permettant de déterminer les extrema d'une fonction à partir des points de contrôle $\alpha_{i,j}$, constitue une base importante des méthodes utilisées dans la Section 3.5.4 et dans le Chapitre 4.

Au final, en contraignant la valeur des points de contrôle, qui font partie des paramètres d'optimisation, on peut facilement assurer les limites des positions articulaires du robot [5]. De même, il est possible de rajouter des contraintes linéaires, qui sont facilement prises en compte par les algorithmes d'optimisation, sur les points de contrôle pour garantir les limites sur les vitesses et accélérations articulaires.

Pour les autres contraintes, nous nous basons sur une approximation polynomiale (en fonction du temps) des fonctions à contraindre, comme montré dans la section suivante.

3.5.2 Approximation polynomiale

L'estimation de contraintes inégalité continues a été résolue en se basant sur l'Analyse par Intervalles(AI) dans [110]. Cependant, le calcul ne prenait pas en compte la corrélation entre les différentes sous-fonctions [185] amenant à une surestimation importante. Il est possible de diminuer cette surestimation via un processus de bissection requérant un grand temps de calcul, ce qui rend son utilisation peu pertinente. Pour surmonter cet inconvénient, nous définissons chaque fonction $f(\chi, t)$ sur un intervalle de temps $\Delta t = [t_s, t_e]$ comme une fonction polynomiale d'ordre n :

$$\forall t \in [\Delta t] \; f(\chi, t) \approx \sum_{i=0}^{n} a_i(\chi) \times t^i + [\varepsilon] \tag{3.26}$$

où n est l'ordre de l'approximation, $\{a_0(\chi), a_1(\chi), \ldots, a_n(\chi)\} \in \mathbb{R}^{n+1}$ sont les coefficients du polynôme dépendant des paramètres d'optimisation χ. Contrairement à [12], nous pouvons omettre de calculer l'intervalle d'erreur restant $[\epsilon]$, en supposant que nous choisissons l'ordre n de l'approximation polynomiale de manière à pouvoir la négliger [109]. A partir de cette approximation polynomiale, il est possible de définir les contraintes de type égalité et inégalité.

5. A noter que le réciproque à l'Équation (3.25) n'est pas vrai. Le choix de gérer les contraintes articulaires via cette méthode est donc sous-optimal, mais facilite grandement la résolution du problème d'optimisation.

3.5.3 Contraintes égalités

Pour définir un contact, il faut imposer les contraintes géométriques égalités présentées en Section 3.3 tout au long d'un (ou de plusieurs) intervalles de temps. Pour cela, en se basant sur l'approximation de l'Équation (3.26), il est possible d'imposer des contraintes sur les coefficients des polynômes tel que :

$$\forall i \in \{0, \cdots, N_e\} \quad a_i(\chi) = 0 \tag{3.27}$$

La contrainte d'égalité continue étant d'ordre n, nous fixons $N_e \leq n$ afin d'éviter une formulation trop contrainte, menant à des difficultés de résolution. Le compromis entre la précision et le taux de convergence de l'optimisation est réglé de manière empirique avec le choix de $N_e = 2$. En fait, ceci met en évidence un problème fondamental plus profond lié au choix de la paramétrisation des trajectoires articulaires dans le cas de chaînes cinématiques fermées.

3.5.4 Contraintes inégalités

Dans cette Section nous détaillons la gestion des contraintes inégalités dans le problème d'optimisation présenté dans l'Équation (3.22) :

$$\forall i \in 1, \ldots, N_i, \forall t \in [\Delta_i] \quad g_i(\chi, t) \leq 0 \tag{3.28}$$

Les algorithmes d'optimisation tels que [102, 174] ne peuvent pas directement prendre en compte une contrainte telle qu'elle est définie dans l'Équation (3.28) car elle traduit le respect d'une contrainte sur une fonction continue, assimilable à une infinité de valeurs comme détaillé dans la Section 3.4.2.

La solution la plus fréquemment utilisée est de discrétiser cette fonction continue en plusieurs instants ce qui revient à prendre en compte les contraintes inégalités de la manière suivante :

$$\forall i \in 1, \ldots, N_i, \forall d \in \{1, \ldots, d_{max}\}, t_d \in [\Delta_i] \quad g_i(\chi, t_d) \leq 0 \tag{3.29}$$

Cette solution utilisée dans [127, 128] permet d'obtenir un nombre fini de contraintes, ce qui permet d'interfacer ce problème avec les algorithmes d'optimisation. Cependant, dans [110], nous avons montré que, même si les contraintes étaient satisfaites aux instants t_d considérés, la fonction continue $g_i(\chi, \beta)$ pouvait violer la contrainte inégalité à des instants non considérés. Pour résoudre cette violation de contrainte non détectée, nous avons proposé d'utiliser l'analyse par intervalles pour calculer les extrema des fonctions $g_i(\chi, \beta)$ et considérer les contraintes de la manière suivante :

$$\forall i \in 1, \ldots, N_i, \forall t \in [\Delta_i] \quad \max\left(g_i(\chi, t)\right) \leq 0 \tag{3.30}$$

Des mouvements de coup de pied pour le robot HOAP-3 ont été générés [110] en garantissant le respect des contraintes sur l'intégralité de l'intervalle de temps considéré en bissectant l'intervalle de temps mais au prix d'un temps de calcul prohibitif (35h pour un mouvement de quelques secondes). Une méthode hybride a donné des résultats beaucoup plus efficaces (environ 30 minutes) en discrétisant les contraintes suivant une grille temporelle puis en vérifiant grâce à l'analyse par intervalles la violation de contraintes afin de pénaliser les contraintes jusqu'à validation totale des contraintes.

Afin de réduire les temps de calcul, nous avons géré les contraintes inégalité en nous basant sur l'approximation polynomiale présentée dans l'Équation (3.26) afin d'en contraindre le maximum comme indiqué dans l'Équation (3.30). Certaines méthodes approchent l'extremum d'un polynôme [40] sans pouvoir restreindre la recherche à un intervalle donné. D'autres méthodes permettent de trouver l'extremum en se basant sur la recherche de racines polynomiales des dérivées du polynôme. Ces approches analytiques échouent pour les polynômes ayant un ordre $n > 5$. Afin d'estimer cet extremum pour tout ordre n, nous avons proposé une contribution [111] basée sur la propriété des B-Splines expliquée dans la Section 3.3.3.

Considérons une fonction $g(t)$ étant un polynôme (ou assimilable à un polynôme) :

$$g(t) = a_0(\chi) + a_1(\chi)t + \ldots + a_n(\chi)t^n = [1, t, \ldots, t^n].[a_0, a_1, \ldots, t_n]^T \tag{3.31}$$

il est possible de calculer les points de contrôle équivalents tels que :

$$g(t) = \sum_{i=0}^{n} b_i^n(t)\rho_i = [1, t, \ldots, t^n].\mathbf{B}^T.[\rho_0, \rho_1, \ldots, \rho_n]^T \tag{3.32}$$

où $\mathbf{B} \in \mathbb{R}^{(n+1)\times(n+1)}$ est une matrice carrée contenant les paramètres polynomiaux des fonctions [6] de base d'ordre n. On peut en déduire la relation liant les points de contrôle équivalents aux coefficients du polynôme :

$$[\rho_0, \rho_1, \ldots, \rho_n] = \mathbf{B}^{-1} \times [a_0, a_1, \ldots, a_n] \tag{3.33}$$

En reprenant la propriété de l'Équation (3.25), qui indique que la fonction $g(t)$ est contenue dans l'enveloppe convexe des points de contrôle équivalents ρ_i, nous pouvons définir les contraintes inégalité sur les points de contrôle tel que :

$$\forall i \in 1, \ldots, N_i, \forall p \in \{0, \ldots, n\} \quad \rho_{i,p}(\chi) \leq 0 \tag{3.34}$$

Cette méthode permet de traiter les contraintes de type inégalité, à l'exception des contraintes de collision et d'auto-collision qui ne peuvent être approximées par une fonction continue (la distance à chaque instant étant calculée via un processus d'optimisation). Dans notre cas, nous avons utilisé les travaux présentés dans [104], qui consiste en une méthode rapide, qui évalue la distance minimale le long d'un intervalle de temps donné.

3.6 Résultats de générations de mouvements

Nous avons validé notre méthode en générant plusieurs mouvements pour le robot humanoïde HRP-2 [89], comme le mouvement présenté sur la Figure 3.6 qui alterne des contacts pieds/sol et bras/bureau. Il présente la capacité de la méthode à générer des mouvements dans des environnements encombrés. La séquence des points de contact peut être définie via des algorithmes de génération de postures [16, 17, 50, 51], puis considéré constante ou comme faisant partie des variables d'optimisation. Les détails d'implémentation sont présentés dans [111] qui est également disponible en Annexe B. Les mouvements mettant en œuvre les bras sont joués en boucle ouverte, sans contrôle d'équilibre [7]

La méthode est suffisamment générale pour générer des mouvements de marche sur sol plat (Figure 3.7) ou sur sol irrégulier comme sur la Figure 3.8.

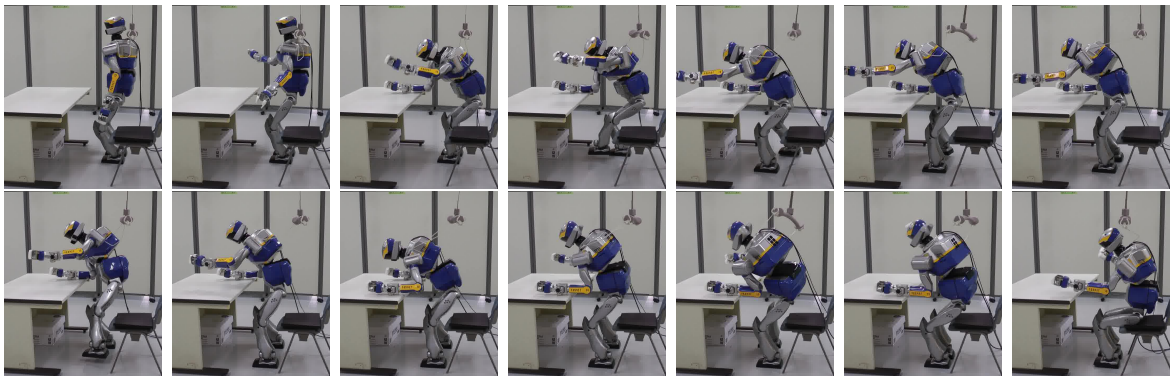


FIGURE 3.6 – Validation expérimentale d'un mouvement multi-contact.

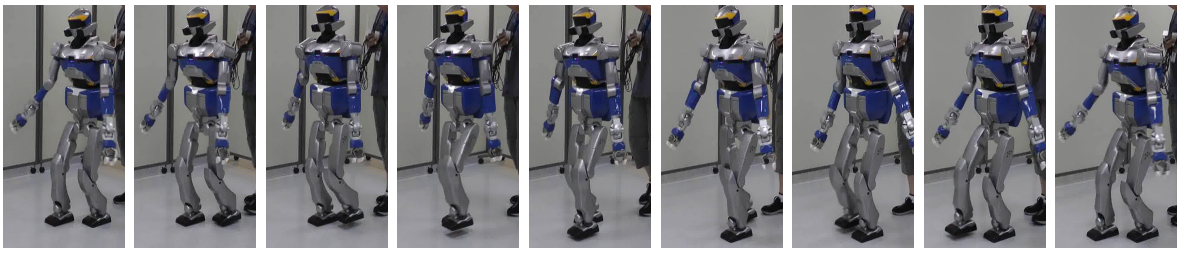


FIGURE 3.7 – Mouvement de marche sur sol plan.

Dans le cas où le robot est suffisamment rigide, nous avons généré des mouvements de marche pour le robot HOAP-3 qui ont été utilisés en boucle ouverte (sans stabilisateur) dans le cadre d'une expérience de téléopération d'un robot à Béziers par la pensée via un IRM en Israël comme montré sur la Figure 3.9.

Ce formalisme permet de prendre en compte certaines contraintes supplémentaires afin de reproduire les déficiences des jambes [108] en fixant une position constante du genou pour simuler un genou portant une attelle (Figure 3.11), ou en fixant des forces de contact maximales sur un pied pour émuler un pied cassé ou douloureux comme illustré par le mouvement de la Figure 3.12 et les courbes des efforts de contact mesurés où l'on voit clairement une phase de préparation en double support présenté sur la Figure 3.10.

Nous avons étendu cette méthode pour simuler des articulations passives sur les membres inférieurs afin d'étudier la fatigue des membres supérieurs d'une personne paraplégique lors d'un mouvement de transfert de fauteuil/chaise [106] en considérant un avatar virtuel et en modifiant l'Équation (3.14) pour y inclure des couples articulaires nuls au niveau des genoux comme montré sur la Figure 3.13.

6. Notez que les fonctions de base d'ordre n utilisées pour évaluer les extrema sont différentes de la fonction de base cubique utilisée pour définir les trajectoires articulaires dans l'Équation (3.21).

7. Seul l'asservissement local à chaque articulation est activé.

FIGURE 3.8 – Marche du robot avec pas sur une marche de 15cm

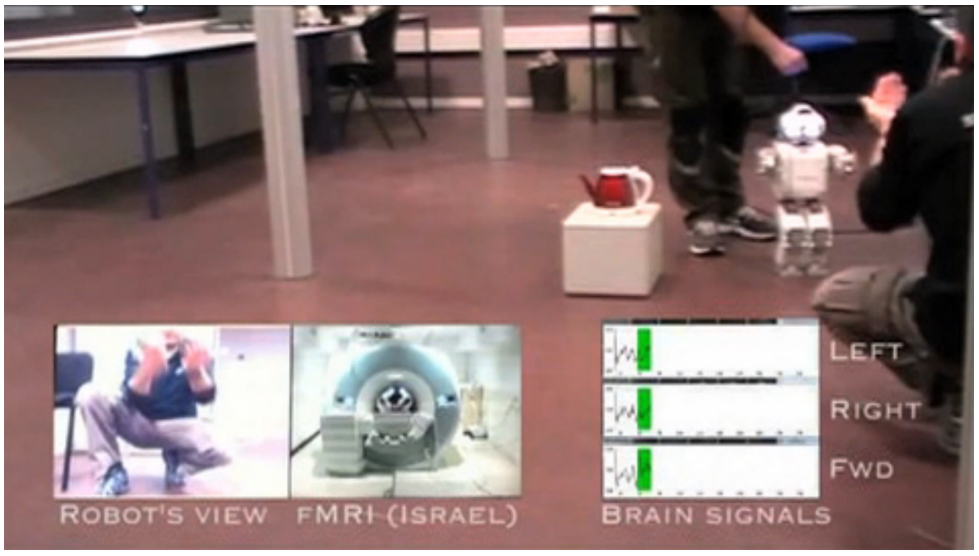
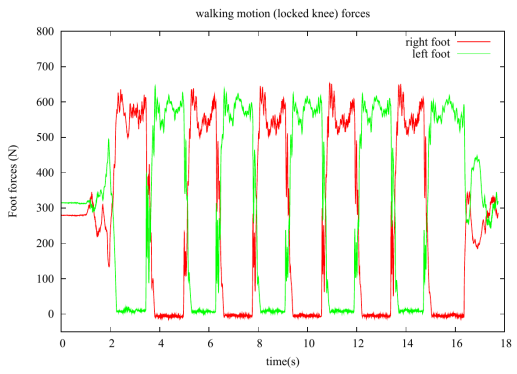
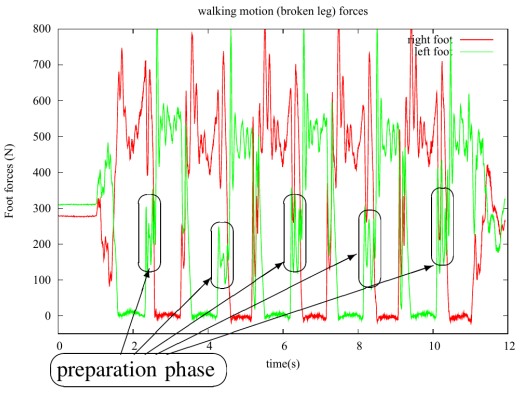


FIGURE 3.9 – Environnement du robot HOAP-3 à Béziers télé-opéré par la pensée depuis un IRM en Israël.



(a) marche normale.



(b) marche avec pied fragile.

FIGURE 3.10 – Représentation des efforts de contact du robot HRP-2 lors d'un mouvement de marche normale et en émulant un pied fragile. source [108]

Toutes ces expériences et simulations illustrent la polyvalence et l'efficacité de notre méthode à générer des mouvements complexes dynamiques et multi-contacts. Il est important de préciser que les mouvements sur le robot HRP-2 impliquant des contacts des membres supérieurs sont réalisés en utilisant uniquement les correcteurs PD (Proportionnel-Dérivé) sans autre retour des forces de contact ou contrôle de l'équilibre. Pour les mouvements de marche, nous utilisons le stabilisateur intégré à HRP-2 présenté dans [86] afin de traiter les flexibilités non modélisées.

Le principal inconvénient de cette méthode est lié au nombre d'itérations du processus d'optimisation et donc au temps de calcul. En effet, le grand nombre de variables et de contraintes prises en compte et la forte non-linéarité des fonctions (contraintes et critère) liés à la précision du calcul numérique font que le critère d'arrêt de l'algorithme est difficilement atteignable.

L'obligation de garder un corps en contact avec le sol constitue également un problème à la génération de mouvements de course ou de saut. La mise en place d'une paramétrisa-

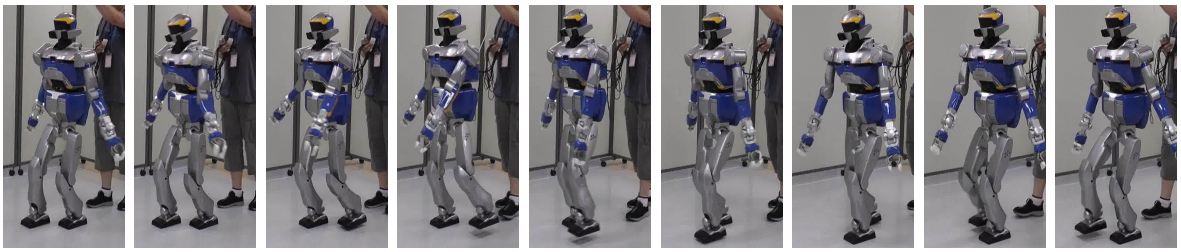


FIGURE 3.11 – Mouvement de marche simulant une blessure au genou

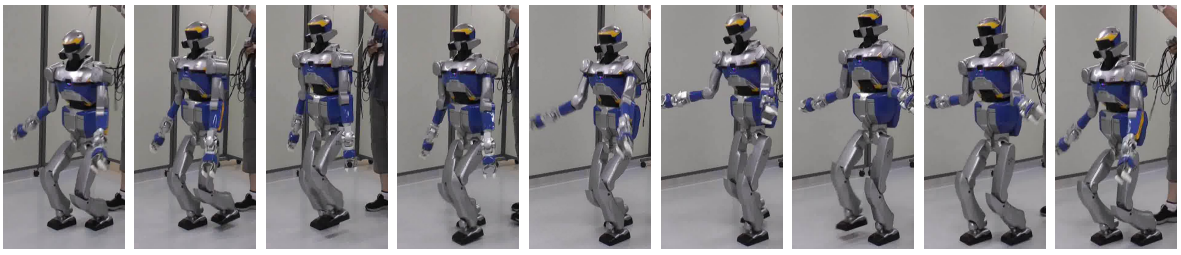


FIGURE 3.12 – Mouvement de marche simulant une blessure au pied

tion adaptée de la trajectoire du corps de référence (le bassin) pourrait amener à la bonne résolution du problème d'optimisation.

3.7 Complexité et garantie vs. temps de calcul

Cette méthode permet donc de générer des mouvements complexes, dynamiques, alternant différentes phases d'appui en garantissant l'intégrité du robot, sous la condition de disposer d'une bonne modélisation du système. Cependant les temps de calcul (allant de quelques minutes à plusieurs heures, voire plusieurs jours) ne permettent pas une application en temps réel ou une adaptabilité à un changement de l'environnement.

Afin de s'adapter à une légère modification de l'environnement, nous avons proposé une méthode de calcul de l'espace faisable (borne sur les paramètres d'optimisation) autour du mouvement optimal basée sur l'Analyse par Intervalles [110]. Une adaptation en ligne vérifie uniquement si les paramètres du mouvement respectent ces limites, ce qui permet de modifier un mouvement existant en moins de deux secondes.

Par ailleurs l'utilisation de méthodes d'IA afin de produire une première solution à proposer à l'algorithme d'optimisation pourrait également permettre de réduire le nombre d'itérations et donc le temps de calcul, mais ne résolverait pas le problème de convergence lié à la complexité et aux erreurs numériques.

FIGURE 3.13 – Simulation d'un mouvement de transfert pour un patient paraplégique.

Chapitre 4

L'Analyse par Intervalles(AI)

4.1 Contexte

Mes activités de recherche en lien avec l'Analyse par Intervalles (AI) ont démarré dès le début de ma thèse encadrée par Philippe Fraisse et Nacim Ramdani. Ce dernier effectuait des recherches en incluant la modélisation et l'analyse de systèmes en présence d'incertitude ainsi que l'estimation basée sur l'AI. Dans le cadre de l'équipe DEMAR, les premières applications de l'AI que j'ai découvertes portaient sur l'estimation de la posture des membres supérieurs dans le cas de la verticalisation d'un patient paraplégique sous stimulation électrique fonctionnelle [142] dans le cadre de la thèse de Gaël Pages [141].

J'ai rapidement été intéressé par le principe de l'AI qui consiste à garantir de manière formelle, grâce à une arithmétique dédiée, un résultat contrairement à une vérification statistique se rapprochant plutôt d'une probabilité que d'une vraie preuve, comme montré de manière simpliste sur la Figure 4.1.

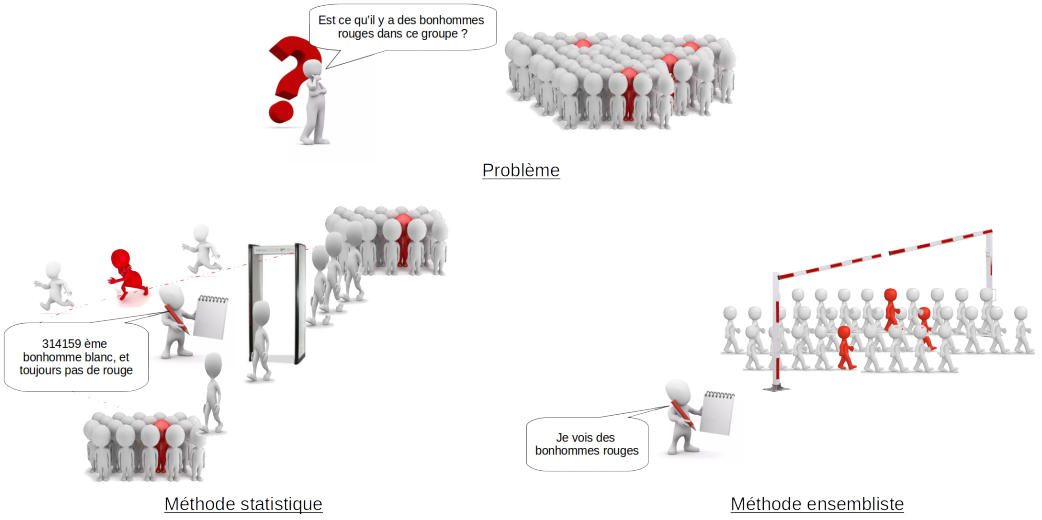


FIGURE 4.1 – Représentation simpliste de la différence entre les méthodes statistiques et les méthodes ensemblistes.

Sur la Figure 4.1, on peut voir qu'une méthode statistique vérifie certains éléments discrets de l'ensemble avec la possibilité d'en omettre quelques uns [1], alors qu'une méthode ensembliste, telle que l'AI effectue les opérations sur un ensemble de valeurs. Malgré la

1. on peut noter que des méthodes proposent des solutions pour pallier ce problème [31, 55]

garantie du résultat prodiguée par ces méthodes ensemblistes, on peut pressentir que la mise en place et la résolution de problème via ces méthodes impactera le temps de calcul.

En reprenant l'analogie, on peut considérer plus simple et plus rapide de tester les éléments un à un. Cependant à partir d'un certain nombre, ou si on considère un ensemble continu [2], si on souhaite obtenir une garantie (optimum global, respect des contraintes, ...), il parait plus pertinent d'utiliser une méthode ensembliste même si on ne dispose finalement pas d'une information précise. Dans le cas de la Figure 4.1, le nombre de bonhommes rouges est difficilement perceptible, ce qui permet de répondre à la question "Est ce qu'il y a des bonhommes rouges dans le groupe", mais n'apporte pas d'information complémentaire concernant leur nombre exact.

Dans le cadre ma thèse, j'ai commencé par utiliser l'AI pour garantir le bon respect des contraintes sur des intervalles de temps et pour construire un ensemble de trajectoires garantissant l'intégrité du robot lors de la tâche de coup de pied [110]. Les temps de calcul de l'AI sont rapidement limitants quand on augmente la dimension du problème. Mes travaux récents portent sur une méthode d'estimation ensembliste basée sur une approximation polynomiale et sur les propriétés des fonctions de base afin de réduire ce temps de calcul pour des espaces de dimension plus importante, comme indiqué dans [107, 112] [3].

4.2 Introduction à l'Analyse par Intervalles(AI)

4.2.1 Définitions

Initialement, l'Analyse par Intervalles(AI) a été développée pour quantifier les erreurs introduites par la représentation en nombres flottants de nombre réels et a été étendue à la validation numérique [129, 138, 164]. Certains auteurs, comme Hansen [65], assimilent le début de l'Analyse par Intervalles à la parution de l'ouvrage de Moore [129], en 1966. L'idée principale est très simple : remplacer les nombres réels par des intervalles auxquels ils appartiennent. Il est alors possible d'obtenir des encadrements numériques garantis par la transposition aux intervalles des fonctions classiques opérant sur des réels [78].

Pour cela, on définit un intervalle réel $[a] = [\underline{a},\ \bar{a}]$ comme un sous-ensemble continu de $\mathbb{R}$. L'ensemble de tous les intervalles réels de $\mathbb{R}$ est noté par $\mathbb{IR}$. Les opérations réelles (addition, soustraction, multiplication, division) sont étendues aux intervalles. Considérons un opérateur $\circ \in \{+, -, *, \div\}$ ainsi que $[a]$ et $[b]$ deux intervalles, alors on définit :

$$[a] \circ [b] = [inf_{u\in[a],v\in[b]}\, u \circ v, \quad sup_{u\in[a],v\in[b]}\, u \circ v] \tag{4.1}$$

De la même façon, les fonctions réelles peuvent être étendues aux intervalles. Considérons la fonction $\mathbf{g} : \mathbb{R}^n \longmapsto \mathbb{R}^m$; l'ensemble des valeurs de cette fonction pour un intervalle $[a]$ est donné par :

$$\mathbf{g}([\mathbf{a}]) = \{\mathbf{g}(\mathbf{u}) \mid \mathbf{u} \in [\mathbf{a}]\} \tag{4.2}$$

La fonction intervalle $[\mathbf{g}] : \mathbb{IR}^n \longmapsto \mathbb{IR}^m$ est une fonction d'inclusion de la fonction réelle $\mathbf{g}$ si :

$$\forall [\mathbf{a}] \in \mathbb{IR}^n,\ \mathbf{g}([\mathbf{a}]) \subseteq [\mathbf{g}]([\mathbf{a}]) \tag{4.3}$$

2. qu'on peut encore assimiler à une infinité de valeurs possibles
3. L'article [112] est disponible dans l'Annexe B.2

On définit qu'une fonction d'inclusion **g** peut être obtenue en remplaçant chaque occurrence de la variable réelle a par la variable intervalle correspondante $[a]$ et chaque fonction réelle par la fonction intervalle adéquate. Dans ce cas, la fonction obtenue est appelée fonction d'inclusion naturelle dont les performances dépendent de l'expression formelle de **g**.

4.2.2 Surapproximation

L'Analyse par Intervalles est donc un outil intéressant qui permet d'obtenir un résultat numérique garanti sur des opérations mettant en scène des ensembles de valeurs. Cependant son utilisation provoque intrinsèquement de la surapproximation [22] principalement liée à l'expression de la fonction d'inclusion. En effet, la présence de plusieurs occurrences de la même variable dans l'expression sans prise en compte des liens entre les occurrences crée une surestimation. Pour illustrer ce point, considérons simplement un intervalle $[t] = [2; 3]$, on voit simplement que la fonction $f(t) = t - t$ sera égale à zéro alors que l'utilisation de la fonction d'inclusion naturelle donnera $f([t]) = [2 : 3] - [2 : 3] = [-1 : 1]$. Même si cette évaluation contient bien 0, on voit qu'on a une surestimation de la solution (qui sera d'autant plus grande que le diamètre de l'intervalle $[t]$ est important).

4.2.3 Optimisation globale

De par sa nature, l'analyse par intervalles représente une approche intéressante pour résoudre de manière globale les problèmes d'optimisation (en évitant les minima locaux) ou pour prouver l'absence de solution à un problème sur-contraint. Contrairement aux méthodes traditionnelles qui utilisent des valeurs ponctuelles, cette technique utilise des intervalles pour représenter les variables, permettant ainsi de mieux gérer les incertitudes et les erreurs de modélisation. Comme l'a montré [66, 73], cette méthode explore systématiquement l'espace de recherche, garantissant des solutions globales en excluant les régions qui ne peuvent pas contenir de solutions optimales même en présence de fonctions non convexes et de contraintes difficiles. Il est intéressant de noter que cette méthode d'optimisation permet de gérer les incertitudes ou imprécisions sur les données, car elle fournit des bornes rigoureuses sur les solutions possibles.

Différents algorithmes d'optimisation existent afin d'orienter la recherche. L'optimisation par bissection est une méthode itérative utilisée pour trouver des solutions optimales en divisant systématiquement l'intervalle de recherche, c'est l'une des plus simples à mettre en place. D'autres méthodes plus élaborées existent telles que la contraction [23] qui réduit l'intervalle de recherche en fonction des différentes contraintes utilisées. Toutes ces méthodes souffrent du processus de surapproximation lié au pessimisme. Dans la suite de ce document nous proposerons une méthode afin de le réduire et l'évaluerons via des optimisations basées sur le processus de bissection.

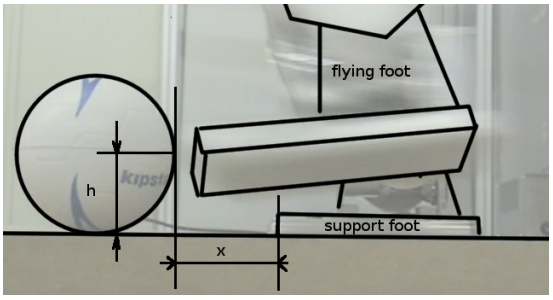
4.2.4 Utilisation de l'AI en robotique

L'Analyse par Intervalles(AI), en robotique, constitue une approche pertinente pour la gestion de l'incertitude qu'elle soit sur l'aspect identification, modélisation, mesure ou commande. La calibration géométrique (étalonnage) des manipulateurs robotiques peut être effectuée avec l'AI [35, 36] ou l'identification paramétrique [76, 77]. L'AI a également été utilisée pour la détermination des singularités de robots parallèles [54, 96] ainsi que pour l'estimation de la pose du robot au passage des singularités [95]. L'utilisation de l'AI

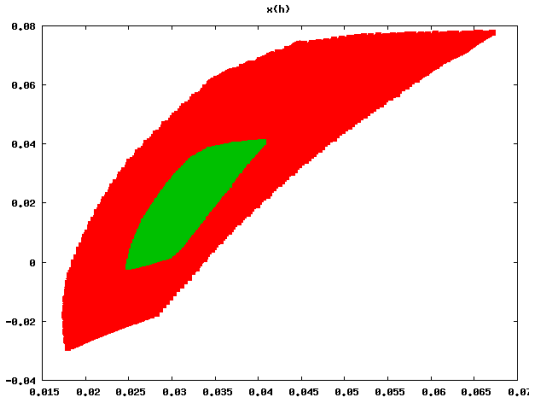
pour résoudre des problèmes d'estimation non-linéaires offre une alternative aux filtres de Kalman [92] en apportant la robustesse de l'approche face à l'incertitude et aux erreurs de modélisation dans des contextes complexes. Un exemple d'application concerne l'estimation des membres supérieurs lors de la verticalisation d'un patient paraplégique sous SEF [143]. Les problématiques de localisation [62] et de *Simultaneous Location And Mapping* (SLAM) [134] peuvent également être traitées de manière robuste. La création de lois de commande robustes [172] et l'identification des domaines d'état assurant la stabilité du système selon Lyapunov [79] bénéficient également des avantages de l'AI.

4.2.5 Calcul d'ensemble faisable

Comme présenté dans le Chapitre 3, nous avons utilisé l'AI dans le cadre d'une discrétisation temporelle pour la génération de mouvements pour un robot humanoïde [110]. Bien que permettant de garantir la viabilité du mouvement, l'utilisation de l'AI menait à des temps de calcul trop longs pour une utilisation en temps réel [4]. Nous avons donc proposé une méthode permettant de définir un ensemble de mouvement faisable autour du mouvement optimal, et l'avons appliqué à un mouvement de coup de pied dans une balle, comme montré sur la Figure 4.2.



(a) Représentation des paramètres (x et h) du coup de pied.



(b) Ensemble des caractéristiques (x, h) atteignables autour du mouvement optimal $x = 1cm$, $h = 3cm$ [5].

FIGURE 4.2 – Défintion des paramètres du coup de pied et de l'espace atteignable autour du mouvement optimal.

L'espace des mouvements faisables est définie par des bornes sur les paramètres des trajectoires articulaires qui peuvent être facilement gérées par les processus d'optimisation. Cet espace contient un ensemble de mouvements possibles dont la Figure 4.3 représente les différentes caractéristiques accessibles. De nouveaux paramètres appartenant à cet espace respectent l'ensemble des contraintes non linéaires qu'il n'est plus nécessaire de vérifier. En cas de position d'impact (x, h) différente de celle prévue lors de l'optimisation initiale, l'adaptation se résumé à résoudre un problème d'optimisation beaucoup plus simple, ce qui se fait en 1 à 2 secondes (au lieu d'une vingtaine de minutes).

4. voire prohibitifs pour une utilisation hors ligne (plusieurs heures voire jours de calcul).

5. les couleurs vert et rouge représentent les résultats des deux étapes de la méthode de calcul de l'espace faisable qui sont disponibles dans [105, 110] et que nous ne détaillerons pas ici.

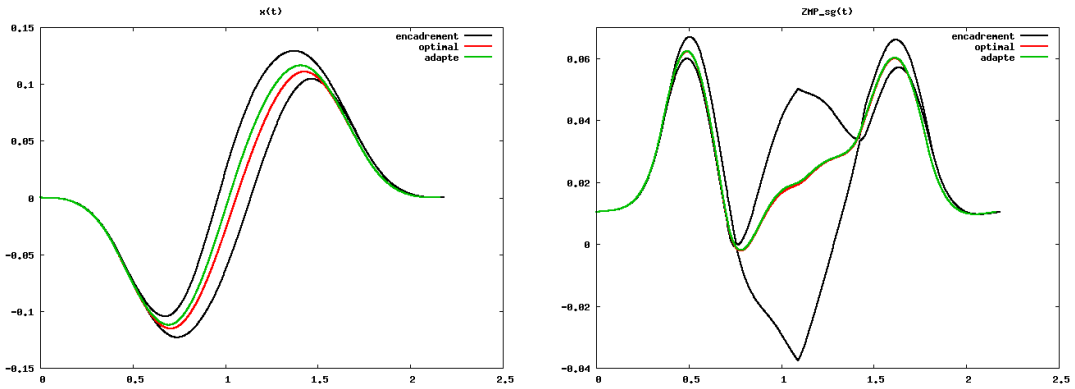


FIGURE 4.3 – Représentation temporelle de la position x du pied et de la contrainte d'équilibre (ZMP dans le plan sagittal), le mouvement adapté ainsi que l'ensemble des mouvements faisables.

Au final, l'un des principaux avantages de l'AI est son apport de garantie sur le résultat obtenu. Malheureusement, les temps de calculs interdisent une utilisation directe lors de processus temps réel, principalement à cause du phénomène de surapproximation : le pessimisme. Afin de réduire le pessimisme, la fonction d'inclusion naturelle peut être remplacée par les fonctions d'inclusion centrées sur une approximation de Taylor ou de Chebyshev [139, 150] ou via une reformulation de Horner [137]. Dans [87], nous avons comparé les différentes manières de réduire le pessimisme et avons montré que notre contribution décrite dans les sections suivantes permettait une meilleure évaluation via une approximation polynomiale ce qui permet de réduire les temps de calcul des processus d'optimisation.

4.3 Approximation par fonctions de base

Dans la Section 3.5, nous avons vu comment estimer les extrema d'une fonction temporelle (mono-variable) grâce à une approximation polynomiale et à l'identification de points de contrôle équivalents. Dans cette partie, nous adaptons la méthode à une fonction multi-variables et présentons différentes fonctions de base dont nous comparons les performances.

4.3.1 Propriétés des fonctions de base

Comme déjà introduit dans la Section 3.5.4, nous considérons un ensemble de fonctions de base $b^k(q)$ d'ordre k qui répond aux contraintes suivantes :

$$\forall q \in [\underline{q}, \overline{q}] \quad \sum_{i=1}^{m} b_i^k(q) = 1 \tag{4.4}$$

$$\forall i, \forall q \in [\underline{q}, \overline{q}] \quad 0 \leq b_i^k(q) \leq 1 \tag{4.5}$$

Ces deux contraintes imposent que si une fonction monovariable peut être définie comme :

$$F(q) = \sum_{i=1}^{m} b_i^k(q) p_i \tag{4.6}$$

alors on obtient :

$$\forall i \in [1,m] \;\; \underline{F} \leq p_i \leq \overline{F} \Rightarrow \forall q \in [\underline{q},\overline{q}] \;\; \underline{F} \leq F(q) \leq \overline{F} \tag{4.7}$$

Considérons que la fonction $F(q)$ peut se mettre sous la forme :

$$F(q) = a_0 + a_1 q + \ldots + a_m q^m = [1, q, \ldots, q^m].[a_0, a_1, \ldots, a_m]^T \tag{4.8}$$

il est possible de ré-écrire $F(q)$ en utilisant les points de contrôle équivalents p_i tel que :

$$F(q) = \sum_{i=0}^{m} b_i^m(q) p_i = [1, q, \ldots, q^m].\mathbf{B}^T.[p_0, p_1, \ldots, p_m]^T \tag{4.9}$$

où $\mathbf{B} \in \mathbb{R}^{(m+1)\times(m+1)}$ est une matrice carrée contenant les paramètres polynomiaux des fonctions de base d'ordre m. On peut en déduire la relation liant les points de contrôle équivalents aux coefficients du polynôme :

$$[p_0, p_1, \ldots, p_m] = \mathbf{B}^{-1} \times [a_0, a_1, \ldots, a_m] \tag{4.10}$$

Finalement, connaître les points de contrôle équivalents p_i d'une fonction $F(q)$ pour un intervalle donné $[q]$ permet d'avoir une évaluation conservative de la borne de $[f([q])]$.

4.3.2 Estimation multi-variables

Considérons, maintenant, une fonction polynomiale à n variables définie comme :

$$F(q) = [a_1, a_2, a_3, \ldots, a_i, \ldots] \times [1, q_1, q_2, q_1 q_2, \ldots, \mu_i, \ldots]^T \tag{4.11}$$

avec $q \in \mathbb{R}^n$, a_i les coefficients du polynôme, $\mu_i = \prod_{j=1}^{n} q_j^{o_{i,j}}$ le i-ème monôme du polynôme et $o_{i,j}$ l'ordre de q_j pour le i-ème monôme.

Principe de base :

Pour une fonction à n dimensions, l'Équation (4.6) peut se généraliser par :

$$F(q) = \sum_{i=1}^{m_1} \sum_{j=1}^{m_2} \ldots \sum_{z=1}^{m_n} \left(b_i^{m_1}(q_1) b_j^{m_2}(q_2) \ldots b_z^{m_n}(q_n) \right) \times p_{i,j,\ldots,z} \tag{4.12}$$

En définissant P, le vecteur des points de contrôle $p_{i,j,\ldots,z}$ et X le vecteur contenant les coefficients a_i des monômes, on peut généraliser l'Équation (4.10) par :

$$X = \mathbb{B} \times P \tag{4.13}$$

avec $\mathbb{B}$ une matrice carrée obtenue via le produit de Kronecker $\otimes$ dont les propriétés qui nous intéressent sont rappelées dans [107] :

$$\mathbb{B} = \mathbf{B}_1 \otimes \mathbf{B}_2 \otimes \ldots \otimes \mathbf{B}_i \otimes \ldots \otimes \mathbf{B}_n \tag{4.14}$$

Les matrices $\mathbf{B}_i$ sont des matrices carrées de taille m_i obtenues en fonction des coefficients des fonctions de base [6]. On peut en déduire l'expression des points de contrôle en fonction des coefficients des polynômes en utilisant la propriété d'inversibilité du produit de Kronecker :

$$P = \mathbb{B}^{-1} X = \left(\mathbf{B}_1^{-1} \otimes \mathbf{B}_2^{-1} \otimes \ldots \otimes \mathbf{B}_n^{-1} \right) X \tag{4.15}$$

6. dont le détail est disponible dans [113]

Normalisation des entrées :

Les premiers travaux [88] considéraient que le vecteur X de l'Équation (4.15) était constant tout le long du processus d'évaluation (et d'optimisation) et met à jour les matrices $\mathbf{B}_i$ en fonction des intervalles $[q_i]$ considérés. Le principal inconvénient de cette méthode réside dans les erreurs numériques liées à l'inversion de la matrice $\mathbf{B}_i{}^{-1}$ pour des intervalles $[q_i]$ étroits. Pour résoudre ce problème, nous avons choisi de normaliser les variables grâce à :

$$[q_i] = m_i + [q_i^{ref}]\frac{d_i}{2} \tag{4.16}$$

Avec m_i et d_i le milieu et la largeur de l'intervalle $[q_i]$ et $[q_i^{ref}]$ un intervalle de référence égal à $[q_i^{ref}] = [-1; 1]$. Cette normalisation permet que la matrice $\mathbb{B}_i$ ne dépende que des intervalles de référence $[q_i^{ref}]$ et, par conséquent, reste constante sur toute la durée du calcul, quelle que soit la valeur de $[q_i]$. Le vecteur des coefficients X dépendra des valeurs m_i et d_i pour tous les intervalles d'entrée $[q_i]$.

Fonctions non linéaires

Cette méthode se base sur une formulation polynomiale des fonctions (critère ou contraintes). En cas de fonctions non-linéaires on utilise une approximation de Taylor :

$$s([q_i]) = s(m_i) + \frac{ds(m_i)}{dq}\frac{d_i}{2}[q_i^{ref}] + \ldots + [\varepsilon_s(q_i)] \tag{4.17}$$

Avec $[\varepsilon_s(q_i)]$ le terme d'erreur qui est normalisé via $[\varepsilon_{s,i}{}^{ref}]$, $m_{s,i}$ et $d_{s,i}$ comme proposé dans l'Équation (4.16). Ce nouvel intervalle $[\varepsilon_{s_i}{}^{ref}]$ est considéré comme une nouvelle entrée de l'Équation (4.11). La valeur de l'intervalle $[\varepsilon_s(q_i)]$, donc les valeurs de $m_{s,i}$ et $d_{s,i}$ sont mises à jour dès que l'intervalle initial $[q_i]$ est modifié. A noter que cette erreur d'approximation devient rapidement négligeable quand les intervalles d'entrée sont étroits. Passé un certain seuil, le calcul de cette erreur n'est plus mis à jour et considéré comme constant afin de réduire le temps de calcul comme présenté dans [112].

4.3.3 Utilisation lors du processus d'optimisation

Lors du processus d'optimisation, on se pose principalement la question de savoir si l'équation suivante [7] est satisfaite :

$$[f([q])] \in [\underline{f}, \overline{f}] \tag{4.18}$$

En partant de la valeur de $[q]$ on effectue les étapes suivantes :

1. détermination des m_i et d_i,
2. calcul des coefficients de X,
3. calcul des points de contrôle p_i afin d'évaluer la fonction $[f([q])]$ à partir de la valeur minimale et maximale des p_i.

Lors de l'évaluation de la fonction $[f([q])]$, il peut se présenter plusieurs cas :

7. Cette équation regroupe à la fois les contraintes inégalité (une borne peut être $\pm\infty$) et également le critère d'optimisation où les bornes seront définies par $-\infty$ et par la valeur courante du critère en cas de problème de minimisation. Le détail complet du processus d'optimisation est disponible dans [107].

1. si $[f([q])] \cap [\underline{f}, \overline{f}] = \emptyset$, l'évaluation de la fonction est entièrement à l'extérieur de l'intervalle considéré, dans ce cas l'entrée $[q]$ considérée ne doit pas être conservée car elle ne respecte pas une contrainte ou ne constitue pas une meilleure solution vis-à-vis du critère.
2. $[f([q])] \subset [\underline{f}, \overline{f}]$, l'évaluation de la fonction est entièrement contenue dans l'intervalle considéré, dans ce cas l'entrée $[q]$ considérée peut être soumise au reste du processus (évaluation d'une autre contrainte, du critère ou bissection). Dans ce cas, si on considère une contrainte, il n'est plus utile d'évaluer de nouveau cette fonction pour tout intervalle issu d'une bissection de $[q]$.
3. Dans les autres cas, l'évaluation $[f([q])]$ de la fonction est à cheval avec l'intervalle considéré. L'entrée $[q]$ pourra être soumise à une autre contrainte, voire à la minimisation du critère, pour être finalement soumise à la bissection.

Dans les deux premiers cas, l'intégralité du vecteur P, défini par l'Équation 4.15, doit être calculé. Il est possible de déterminer si on est dans le dernier cas dès que deux éléments du vecteur P ne sont plus cohérents (un à l'extérieur, un à l'intérieur), il est inutile de calculer les autres coefficients p_i, ce qui permet d'accélérer les calculs.

4.3.4 Fonctions de bases

Dans cette section, nous présentons différentes fonctions de base afin d'évaluer leurs performances dans le cadre d'un processus d'optimisation sous contraintes. Certaines de ces fonctions sont également comparées dans [168] dans le cas de la recherche de simplexes enfermant des objets 3D avec un volume minimum.

Bernstein

Les fonctions de base de Bernstein sont utilisées dans la construction des courbes de Bézier. Elles sont définies [8] comme suit :

$$b_{i,k}(q) = \frac{k!}{i!(k-i)!}.q^i(1-q)^{k-i} \tag{4.19}$$

Ici, k est le degré de la courbe de Bézier, q est le paramètre et i est compris entre 0 et k. Les fonctions de Bernstein sont un outil efficace pour l'interpolation des points de contrôle et la définition de la géométrie des courbes de Bézier [53].

B-Splines

Les fonctions de base B-Splines peuvent être définies de manière récursive à l'aide de la formule de récurrence de Cox-de Boor [41]. Le cas de base pour $k = 0$ est donné par :

$$b_i^0(q) = \begin{cases} 1 & \text{if } q_i \leq q < q_{i+1} \\ 0 & \text{sinon} \end{cases} \tag{4.20}$$

Pour les degrés supérieurs $k > 0$, la définition récursive est la suivante :

$$b_i^k(q) = \left(\frac{q - q_i}{q_{i+k} - q_i}\right) b_i^{k-1}(q) + \left(\frac{q_{i+k+1} - q}{q_{i+k+1} - q_{i+1}}\right) b_{i+1}^{k-1}(q) \tag{4.21}$$

8. Cette équation est communément présentée. Elle définit les fonctions de Bernstein pour un intervalle $[0:1]$, dans notre cas il faut l'adapter pour un intervalle $[-1:1]$

Où $\mathbf{K} = \{q_0, q_1, ..., q_{n+k+1}\}$ est défini comme le vecteur nodal qui joue un rôle important dans la détermination de la continuité et du comportement de la B-Spline. Les B-Splines ont été utilisées pour calculer une trajectoire articulaire continue dans la génération de mouvements pour les robots humanoïdes [111]. Dans notre cas, puisque nous normalisons l'entrée comme expliqué dans la Section 4.3.2, nous avons défini la fonction de base B-Splines basée sur le vecteur de nœuds :

$$\mathbf{K} = \{-1-2k, -1-2(k-1), \ldots, -1, 1, \ldots, 1+2(k-1), 1+2k\} \tag{4.22}$$

Dans le cadre de la génération de mouvements, le choix du vecteur nodal permet d'avoir une continuité sur les fonctions de base entre deux intervalles.

Note : Si on considère le vecteur nodal $K = \{-1, -1, \ldots, 1, 1\}$ on retrouve les courbes de Bernstein que nous utilisons pour des intervalles de référence $[-1; 1]$.

MinVolume

Dans [168], l'auteur propose de construire les fonctions de base qui minimisent le volume de l'enveloppe convexe définie par les points de contrôle. Ils ont défini la fonction de base comme suit :

$$\begin{array}{lll}
\underline{\text{si } k \text{ est impair}} & b_i(q) = -\alpha_i(q-1)\prod_{j=1}^{\frac{k-1}{2}}(q-q_{i,j})^2 & i = 0, 2, \ldots k-1 \\
& b_i(q) = b_{n-1}(-q) & i = 1, 3, \ldots, k \\
\underline{\text{si } k \text{ est pair}} & b_i(q) = -\alpha_i(q+1)(q-1)\prod_{j=1}^{\frac{k-2}{2}}(q-q_{i,j})^2 & i \text{ un entier impair} \in [0, k/2-1] \\
& b_i(q) = \alpha_i \prod_{j=1}^{\frac{k}{2}}(q-q_{i,j})^2 & i \text{ un entier pair} \in [0, k/2-1] \\
& b_i(q) = b_{k-1}(-q) & i = k/2+1, \ldots, k
\end{array} \tag{4.23}$$

La méthode proposée optimise les coefficients α_i et les racines $q_{i,j}$ afin d'assurer la contrainte de l'Équation (4.4) et (4.5) et de minimiser l'équation suivante afin d'assurer un volume minimal de l'enveloppe convexe.

$$\text{minimize} - det(|\mathbf{B}|)^2 \tag{4.24}$$

On peut noter que l'optimisation proposée dans [168] ne garantit pas un minimum global pour les degrés supérieurs à 4.

Approximation de MinVolume

La résolution du problème d'optimisation défini par l'Équation (4.24) n'est pas triviale et est considérée comme impossible pour $k > 7$. Les auteurs de [168] ont donc proposé un algorithme d'approximation récursive des racines $(q_{i,j})$, tel que :

$$q_{i,j} = sin\left(\frac{c_0\left(i - \frac{s_{j,k}-1}{2}\right) + c_1\left(j - \frac{k-1}{2}\right)}{k + c_2}\right) \tag{4.25}$$

Les valeurs de $c_0 \approx 0.2735$, $c_1 \approx 3.0385$ et $c_2 \approx 0.4779$ ont été trouvées en interpolant les valeurs à partir des valeurs pour l'ordre $k < 7$. Les valeurs de α_i sont obtenues en résolvant la contrainte linéaire de l'Équation (4.4).

MinNorme

Dans [112], nous proposons d'utiliser le même processus d'optimisation que celui présenté pour les **MinVolume** mais en essayant de minimiser la norme de P pour tous les X dans l'Équation (4.13), donc en essayant de trouver les coefficients α_i et les racines $q_{i,j}$ de manière à :

$$\text{minimize}||\mathbf{B}^{-1}||^2 \tag{4.26}$$

MinVariance

Nous proposons également de minimiser la variance du vecteur P afin de minimiser la différence entre la valeur maximale et la valeur minimale de P, et nous définissons donc les fonctions de base par :

$$\text{minimize}||\mathbf{B}^{-1} - Average\left(\mathbf{B}^{-1}\right)||^2 \tag{4.27}$$

Avec $Average\,(A)$ est une matrice contenant la moyenne des colonnes de la matrice A :

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,k-1} & a_{1,k} \\ a_{2,1} & a_{2,2} & \dots & a_{2,k-1} & a_{2,k} \\ \dots & \dots & \dots & \dots & \dots \\ a_{k-1,1} & a_{k-1,2} & \dots & a_{k-1,k-1} & a_{k-1,k} \\ a_{k,1} & a_{k,2} & \dots & a_{k-1,k-1} & a_{k,k} \end{bmatrix} \quad Average\,(\mathbf{A}) = \begin{bmatrix} \langle a_1 \rangle & \langle a_1 \rangle & \dots & \langle a_1 \rangle & \langle a_1 \rangle \\ \langle a_2 \rangle & \langle a_2 \rangle & \dots & \langle a_2 \rangle & \langle a_2 \rangle \\ \dots & \dots & \dots & \dots & \dots \\ \langle a_{k-1} \rangle & \langle a_{k-1} \rangle & \dots & \langle a_{k-1} \rangle & \langle a_{k-1} \rangle \\ \langle a_k \rangle & \langle a_k \rangle & \dots & \langle a_k \rangle & \langle a_k \rangle \end{bmatrix} \tag{4.28}$$

Avec $\langle a_i \rangle = \frac{1}{k}\sum_{j=1}^{k} a_{i,k}$.

Recursive

Dans cette partie, nous présentons des fonctions de base qui pourraient accélérer le calcul, en produisant une partie éparse ou répétitive de la matrice $\mathbf{B}_i^{-1}$. Même si cette méthode ne cherche pas à avoir une estimation la plus conservative possible, nous espérons que l'économie de calcul d'une itération compensera le surplus d'itérations introduit par le pessimisme. En considérant les fonctions suivantes pour une entrée normalisée :

$$\beta_1(q) = 0.5 - 0.5q \tag{4.29}$$

$$\beta_2(q) = 0.5 + 0.5q \tag{4.30}$$

$$\delta_0(q) = 1.0 \tag{4.31}$$

nous avons construit la récurrence suivante et défini la fonction récursive comme suit :

$$\forall i \in \{1, \dots, k\} \begin{cases} b_i(q) = \delta_{i-1}(q).\beta_1(q) \\ \delta_i(q) = \delta_{i-1}(q).\beta_2(q) \end{cases} \tag{4.32}$$

Une variante de cette fonction de base, appelée Recursive2, peut être définie comme suit :

$$\forall i \in \{1, \dots, k\} \begin{cases} b_i(q) = \delta_{i-1}(q).\beta_2(q) \\ \delta_i(q) = \delta_{i-1}(q).\beta_1(q) \end{cases} \tag{4.33}$$

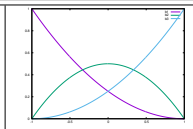
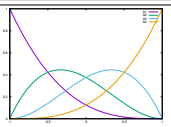
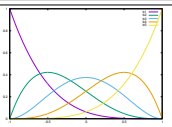
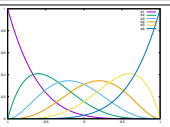
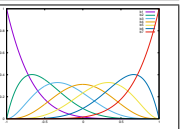
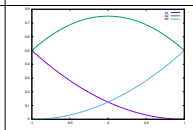
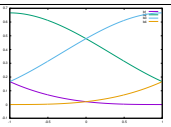
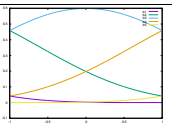
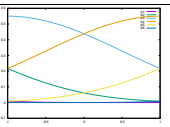
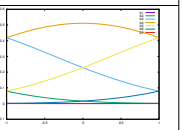
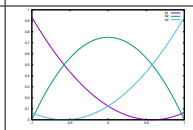
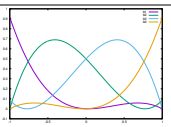
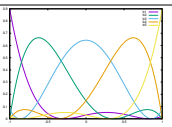
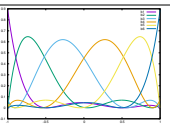
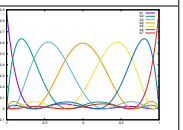
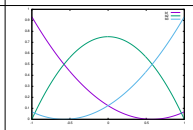
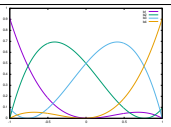
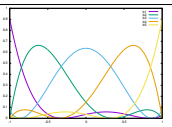
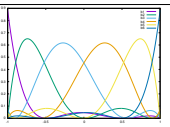
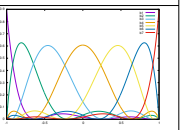
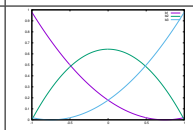
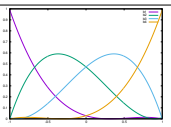
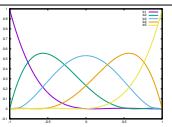
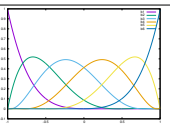
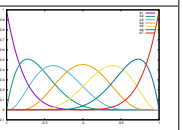
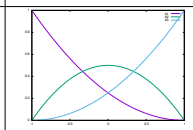
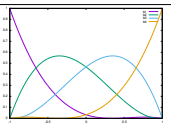
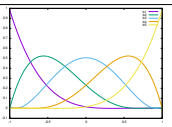
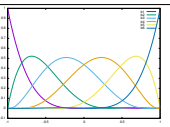
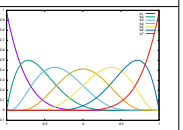
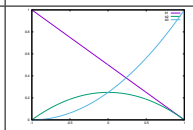
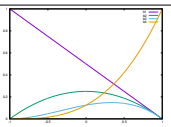
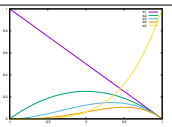
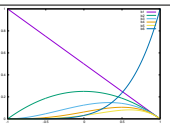
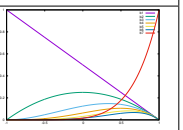
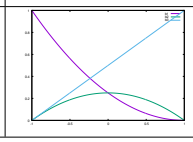
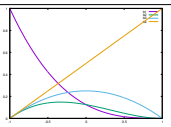
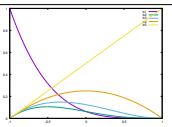
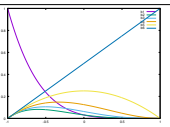
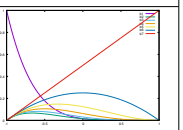
order :	2	3	4	5	6
Bernstein					
BSplines					
MinvVolume					
ApproxMinVo					
MinNorm					
MinVariance					
Recursive1					
Recursive2					

TABLE 4.1 – Représentation des différentes fonctions de base pour des ordres 2 à 6.

4.3.5 Valeurs et optimalité des fonctions de base

La représentation visuelle des fonctions de base est disponible sur la Figure 4.1 pour les ordres 2 à 6 et les matrices $\mathbf{B}^{-1}$ pour les ordres 1 à 3 sont disponibles dans le Tableau 4.2 [9]. On peut remarquer que les fonctions de base Bernstein, MinVolume, ApproxMinVolume et MinNorm produisent des valeurs relativement petites dans la matrice $\mathbf{B}^{-1}$ contrairement aux fonctions de base B-Splines ou Recursives qui peuvent produire des évaluations plus grandes (ce qui s'accentue avec l'ordre). Cependant, les fonctions de base Recursive et Recursive2 ont une partie triangulaire (inférieure gauche pour Recursive et supérieure gauche pour Recursive2) de cette matrice (avec seulement des 1 ou -1), ce qui permet de capitaliser les calculs et ainsi d'accélérer l'évaluation comme désiré.

Le Tableau 4.3 présente les résultats des critères de MinVolume, MinNorm et MinVariance pour toutes les fonctions de base pour les ordres 1 à 3 [10]. On peut voir que les fonctions de base Bernstein, MinVariance, MinNorm, MinVolume et ApproxMinVolume ont des performances assez similaires, tandis que les fonctions Recursive, Recursive2 et

9. Les valeurs numériques de la matrice $\mathbf{B}$ et $\mathbf{B}^{-1}$ pour les ordres 1 à 6 sont disponibles dans [113].
10. les résultats pour les ordres 1 à 6 sont disponibles dans [113].

type	$\mathbf{B}^{-1}$		
	1	2	3
Bernstein	$\begin{bmatrix} 1.000000 & -1.000000 \\ 1.000000 & 1.000000 \end{bmatrix}$	$\begin{bmatrix} 1.000000 & -1.000000 & 1.000000 \\ 1.000000 & 0.000000 & -1.000000 \\ 1.000000 & 1.000000 & 1.000000 \end{bmatrix}$	$\begin{bmatrix} 1.000000 & -1.000000 & 1.000000 & -1.000000 \\ 1.000000 & -0.333333 & -0.333333 & 1.000000 \\ 1.000000 & 0.333333 & -0.333333 & -1.000000 \\ 1.000000 & 1.000000 & 1.000000 & 1.000000 \end{bmatrix}$
BSplines	$\begin{bmatrix} 1.000000 & -1.000000 \\ 1.000000 & 1.000000 \end{bmatrix}$	$\begin{bmatrix} 1.000000 & -2.000000 & 3.000000 \\ 1.000000 & 0.000000 & -1.000000 \\ 1.000000 & 2.000000 & 3.000000 \end{bmatrix}$	$\begin{bmatrix} 1.000000 & -3.000000 & 7.666667 & -15.000000 \\ 1.000000 & -1.000000 & -0.333333 & 3.000000 \\ 1.000000 & 1.000000 & -0.333333 & -3.000000 \\ 1.000000 & 3.000000 & 7.666667 & 15.000000 \end{bmatrix}$
MinVolume	$\begin{bmatrix} 1.000000 & -1.000000 \\ 1.000000 & 1.000000 \end{bmatrix}$	$\begin{bmatrix} 1.000000 & -1.154701 & 1.000000 \\ 1.000000 & 0.000000 & -0.333333 \\ 1.000000 & 1.154701 & 1.000000 \end{bmatrix}$	$\begin{bmatrix} 1.000000 & -1.149096 & 1.093732 & -1.090170 \\ 1.000000 & -0.592096 & -0.000898 & 0.037136 \\ 1.000000 & 0.592096 & -0.000898 & -0.037136 \\ 1.000000 & 1.149096 & 1.093732 & 1.090170 \end{bmatrix}$
ApproxMinVo	$\begin{bmatrix} 1.000000 & -1.000000 \\ 1.000000 & 1.000000 \end{bmatrix}$	$\begin{bmatrix} 1.000000 & -1.156628 & 1.000000 \\ 1.000000 & -0.000000 & -0.331122 \\ 1.000000 & 1.156628 & 1.000000 \end{bmatrix}$	$\begin{bmatrix} 1.000000 & -1.160380 & 1.102051 & -1.097218 \\ 1.000000 & -0.573815 & -0.001433 & 0.046001 \\ 1.000000 & 0.573815 & -0.001433 & -0.046001 \\ 1.000000 & 1.160380 & 1.102051 & 1.097218 \end{bmatrix}$
MinNorm	$\begin{bmatrix} 1.000000 & -1.000000 \\ 1.000000 & 1.000000 \end{bmatrix}$	$\begin{bmatrix} 1.000000 & -1.043194 & 1.000000 \\ 1.000000 & 0.000000 & -0.556693 \\ 1.000000 & 1.043194 & 1.000000 \end{bmatrix}$	$\begin{bmatrix} 1.000000 & -1.016520 & 1.011659 & -1.006966 \\ 1.000000 & -0.502965 & -0.060748 & 0.366237 \\ 1.000000 & 0.502965 & -0.060748 & -0.366237 \\ 1.000000 & 1.016520 & 1.011659 & 1.006966 \end{bmatrix}$
MinVariance	$\begin{bmatrix} 1.000000 & -1.000000 \\ 1.000000 & 1.000000 \end{bmatrix}$	$\begin{bmatrix} 1.000000 & -1.000000 & 1.000000 \\ 1.000000 & -0.000000 & -0.999900 \\ 1.000000 & 1.000000 & 1.000000 \end{bmatrix}$	$\begin{bmatrix} 1.000000 & -1.003617 & 1.002520 & -1.001329 \\ 1.000000 & -0.536651 & -0.070682 & 0.435290 \\ 1.000000 & 0.536651 & -0.070682 & -0.435290 \\ 1.000000 & 1.003617 & 1.002520 & 1.001329 \end{bmatrix}$
Recursive1	$\begin{bmatrix} 1.000000 & -1.000000 \\ 1.000000 & 1.000000 \end{bmatrix}$	$\begin{bmatrix} 1.000000 & -1.000000 & 1.000000 \\ 1.000000 & 1.000000 & -3.000000 \\ 1.000000 & 1.000000 & 1.000000 \end{bmatrix}$	$\begin{bmatrix} 1.000000 & -1.000000 & 1.000000 & -1.000000 \\ 1.000000 & 1.000000 & -3.000000 & 5.000000 \\ 1.000000 & 1.000000 & 1.000000 & -7.000000 \\ 1.000000 & 1.000000 & 1.000000 & 1.000000 \end{bmatrix}$
Recursive2	$\begin{bmatrix} 1.000000 & -1.000000 \\ 1.000000 & 1.000000 \end{bmatrix}$	$\begin{bmatrix} 1.000000 & -1.000000 & 1.000000 \\ 1.000000 & -1.000000 & -3.000000 \\ 1.000000 & 1.000000 & 1.000000 \end{bmatrix}$	$\begin{bmatrix} 1.000000 & -1.000000 & 1.000000 & -1.000000 \\ 1.000000 & -1.000000 & 1.000000 & 7.000000 \\ 1.000000 & -1.000000 & -3.000000 & -5.000000 \\ 1.000000 & 1.000000 & 1.000000 & 1.000000 \end{bmatrix}$

TABLE 4.2 – Valeurs des matrices $\mathbf{B}^{-1}$ pour les différentes fonctions de base et pour des ordres 1 à 36.

surtout les B-Splines ont des performances très médiocres.

4.3.6 Évaluation du pessimisme pour un robot 3D

Tout d'abord, nous comparons les performances des différentes fonctions de base vis-à-vis du pessimisme avec le modèle complexe d'un robot 3D (KUKA LWR). Nous calculons la position de l'effecteur final et le couple articulaire pour une posture statique ainsi que la somme des couples au carré. Pour ce faire, nous utilisons le modèle cinématique direct et le modèle dynamique inverse comme présenté dans [161].

La Figure 4.4 montre les performances d'évaluation de ces fonctions en considérant plusieurs diamètres de boîte en entrée $[q]$. Cette figure se focalise sur la largeur de la somme des carrés des couples. Les résultats complets avec la valeur de l'intervalle sont disponibles dans [113].

On remarque que pour des petits diamètres d'intervalles d'entrée, l'évaluation en utilisant des fonctions de base donne toujours des meilleurs résultats (sauf pour les B-Splines). Pour les intervalles d'entrée les plus larges, ces méthodes donnent un résultat similaire à une fonction d'inclusion naturelle basée sur l'AI. A partir de ces premiers résultats on peut déduire que l'utilisation des B-Splines produira beaucoup de pessimisme ce qui induira davantage de bissections et de ce fait un temps de calcul important. Les autres fonctions de base semblent donner des résultats similaires et nécessitent une comparaison dans le cadre d'un processus d'optimisation.

order	type	$MinVolume$ $-det(\|\mathbf{B}\|)^2$	$MinNorm$ $\|\|\mathbf{B}^{-1}\|\|^2$	$MinVariance$ $\|\|\mathbf{B}^{-1} - Average\,(\mathbf{B}^{-1})\,\|\|^2$
1	all	-0.25	4	2
2	Bernstein	-0.0625	8	4.66667
	MinVariance	-0.101637	7.53553	**3.73773**
	BSplines	-0.00390625	30	18.6667
	MinNorm	-0.094799	**7.48641**	3.79204
	MinVolume	**-0.105469**	7.77778	3.85185
	ApproxMinVo	-0.105467	7.78522	3.85684
	Recursive	-0.015625	17	13.3333
	Recursive2	-0.015625	17	13.3333
3	Bernstein	-0.0197754	12.4444	8
	MinVariance	-0.0776098	10.9429	**5.99574**
	BSplines	-1.69542e-06	609.778	552
	MinNorm	-0.0703759	**10.9231**	6.01885
	MinVolume	**-0.110146**	12.1142	6.91991
	ApproxMinVo	-0.110036	12.1925	6.98117
	Recursive	-0.000244141	96	90
	Recursive2	-0.000244141	96	90

TABLE 4.3 – Valeurs des différents critères de chaque fonction de base pour des ordres 1 à 3.

4.3.7 Evaluation des performances durant un processus d'optimisation

Dans cette partie, nous évaluons l'intérêt de l'utilisation des fonctions de base en comparant leur impact sur le temps de calcul et le nombre d'itérations pour plusieurs dimensions d'entrée. Nous considérons deux types de problèmes pour générer une posture statique pour un système robotique :

— un robot planaire avec 2 à 5 degrés de liberté,

— et un robot 3D inspiré du robot KUKA-LWR avec 4 ou 6 degrés de liberté.

Les résultats présentés sont basés sur la moyenne de plusieurs cas (différentes positions de l'effecteur et minimisation de la somme des carrés des positions ou des couples des articulations), les résultats complets sont présentés dans [113].

Comme suggéré précédemment, la fonction de base BSplines produit les plus mauvais résultats par rapport aux autres fonctions de base, nous l'excluons de la description des résultats présentés ci-après. La Figure 4.5 présente le ratio du nombre d'itérations du processus d'optimisation par rapport à l'utilisation de la fonction d'inclusion naturelle. On constate que toutes les fonctions de base produisent un nombre d'itérations plus faible. Le gain est moins important pour les problèmes simples (avec un faible nombre de degrés de liberté), mais il est de l'ordre de 10 à 30% pour les problèmes complexes. Les différentes fonctions de base produisent presque les mêmes résultats, même si la fonction de base de Bernstein semble être légèrement meilleure.

La Figure 4.6 présente les résultats concernant le temps de calcul (sans tenir compte de la phase de préparation). Nous pouvons également remarquer que les performances sont améliorées avec la complexité du problème (les problèmes 3D-4 et 3D-6 sont les plus

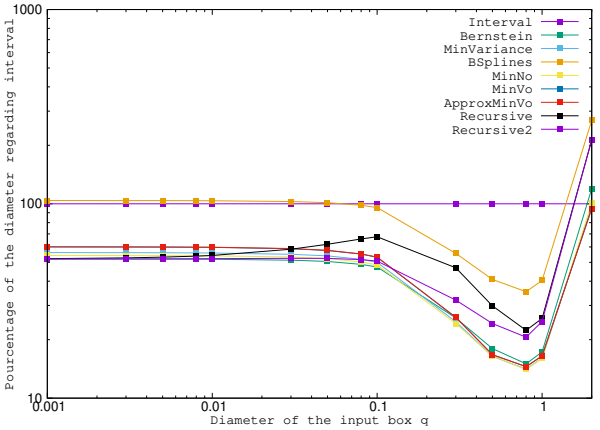
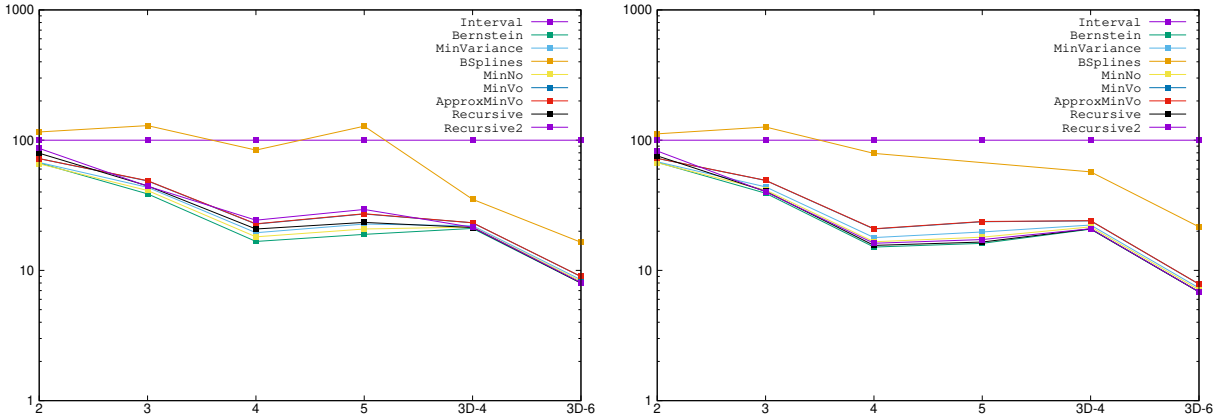


FIGURE 4.4 – Representation of the diameter of the evaluation using basis function regarding intervalle analysis.



(a) precision de 0.01 (b) precision de 0.001

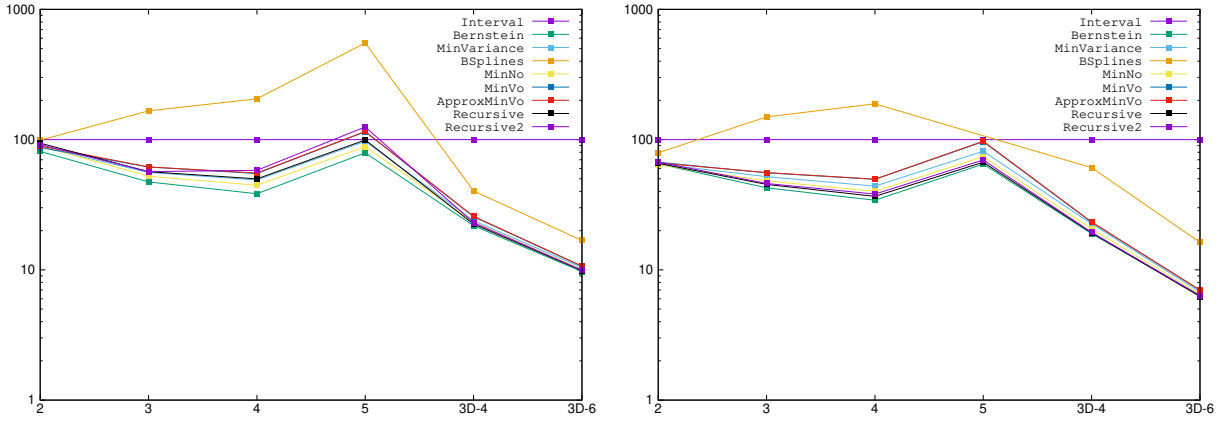
FIGURE 4.5 – Comparaison du nombre d'itérations.

complexes). Nous pouvons également constater que les fonctions de base produisent un plus faible temps de calcul avec une précision plus faible. Ceci peut être expliqué par la Figure 4.4, pour laquelle nous pouvons voir que les performances d'évaluations diminuent quand l'intervalle d'entrée augmente.

La Figure 4.7 présente les résultats concernant le temps de calcul en tenant compte de la phase de préparation (qui va de quelques secondes pour le cas 2D à plusieurs minutes pour les cas 3D). Comme précédemment, l'avantage de l'utilisation de la fonction de base est plus important lorsque l'on considère des systèmes complexes (3D-6). En effet, le temps de préparation devient négligeable devant le temps de résolution du problème d'optimisation (qui va de plusieurs heures à plusieurs semaines).

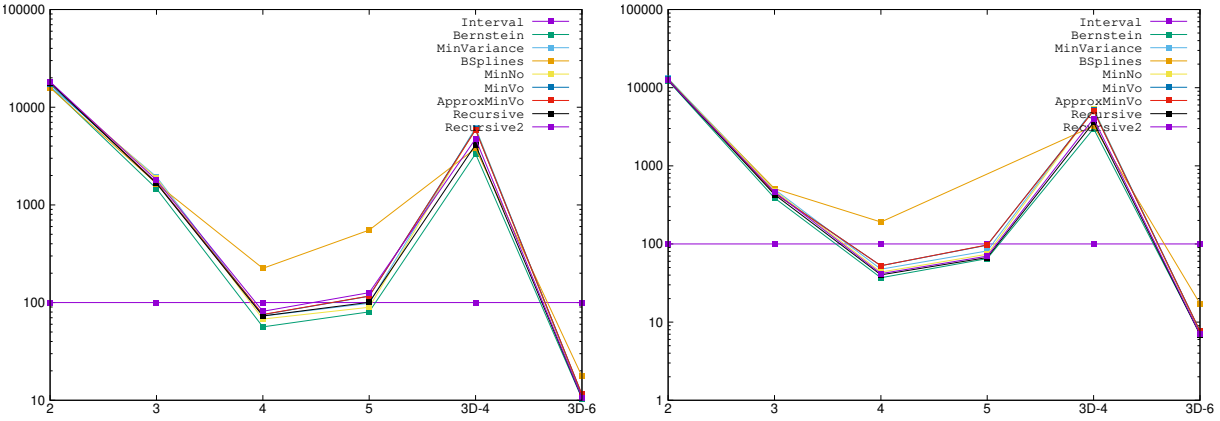
En ce qui concerne le temps de calcul des Figures 4.6 et 4.7, il apparaît que toutes les fonctions de base produisent des résultats similaires, même si nous pouvons remarquer que la fonction de base de Bernstein fournit des résultats légèrement meilleurs.

Parmi les différentes fonctions de base que nous avons évaluées, il semble que le choix initial d'utiliser la fonction de Bernstein produise les meilleurs résultats en termes de nombre d'itérations et de temps de calcul. Cette conclusion n'est pas intuitive au regard de la comparaison des critères (*MinVolume*, *MinNorm* ou *MinVariance*) du Tableau 4.3. De plus la différence des performances n'est pas suffisamment marquée, le type et le



(a) precision of 0.01 (b) precision of 0.001

FIGURE 4.6 – Comparaison du temps de calcul (sans prise en compte du temps de préparation.



(a) precision of 0.01 (b) precision of 0.001

FIGURE 4.7 – Comparaison du temps de calcul total.

nombre d'optimisations ne paraissent pas suffisants pour statuer définitivement sur la supériorité des fonctions de Bernstein. Dans tous les cas, l'utilisation d'une fonction de base dans le cadre d'une optimisation présente un bénéfice par rapport à l'utilisation de la fonction d'inclusion naturelle.

4.4 Garantie vs. temps de calcul et complexité

L'Analyse par Intervalles (AI) est un outil qui permet d'obtenir une garantie sur le résultat en fonction des hypothèses sur les données d'entrée. Dans le cas d'un processus d'optimisation il permet d'assurer l'optimalité du résultat et de garantir le respect des contraintes, notamment dans le cas de la génération de mouvements en robotique. Cependant, cette méthode souffre d'une sur-estimation des fonctions (le pessimisme) qui amène à des temps de calcul prohibitifs dans le cas de problèmes complexes. Plusieurs contributions à la réduction du temps de calcul pourraient être évaluées [11] en :

11. Une implémentation de ces contributions est actuellement en cours dans la librairie Popotam (Parallel Optimization using POlynomial Taylor Approximation Method) disponible sur `https://github.com/lengagne/popotam`.

— implémentant les algorithmes de manière paralléliser sur plusieurs CPU ou sur GPU,

— exploitant la valeur des points de contrôle pour guider le processus de bissection,

— partant d'une solution initiale provenant des méthodes d'optimisation classiques.

Même si toutes ces améliorations sont possibles, la réduction du temps de calcul pour la résolution de problèmes complexes ne sera pas suffisante pour une utilisation en temps réel dans une application robotique. Cependant, il est important de considérer les avantages de l'AI afin de proposer des preuves théoriques d'optimalité, de bon déroulement d'une tâche ou de respect de l'intégrité physique du robot et de son environnement. Parmi ces méthodes, on peut envisager les méthodes d'Intelligence Artificielle (IA) (principalement basées sur les réseaux de neurones) qui sont de plus en plus utilisées pour résoudre des tâches robotiques complexes en temps réel.

Chapitre 5

l'IA : Intelligence Artificielle

5.1 Contexte

A l'heure actuelle le terme d'Intelligence Artificielle(IA) est régulièrement mis en avant dans les médias pour décrire des avancées technologiques impressionnantes et alimente quelques débats sur les bienfaits et les dangers de son utilisation. Le grand public perçoit l'IA comme un domaine qui vise à créer des systèmes capables d'effectuer des tâches qui nécessitent généralement l'intelligence humaine. Ces tâches incluent la résolution de problèmes complexes comme l'apprentissage, la reconnaissance de motifs, la compréhension du langage naturel, la prise de décision et l'adaptation à des situations changeantes. Suivant cette définition, l'IA appliquée à la robotique englobe un large éventail de techniques, comme les algorithmes traditionnels de résolution de chemin ([15, 46, 98]), de génération de mouvements [27, 111] ou de commande [94, 124]. Dans la suite de ce document, nous assimilerons l'IA aux méthodes d'apprentissage basées sur les réseaux de neurones [157].

A partir des années 2010, le terme IA revenait dans de plus en plus de publications scientifiques. À cette époque, j'avais le sentiment que de nombreux travaux justifiaient le recours à l'IA par une similitude avec les réseaux de neurones humains et justifiaient l'utilisation de l'IA en robotique par de possibles progrès dans ces deux domaines : en robotique pour la réalisation de tâches complexes mais également en neurosciences afin de mieux comprendre le cerveau humain. Les premiers réseaux considérés étaient de simples perceptrons multi-couches datant des années 1950 [157]. À mes yeux, ces réseaux étaient très éloignés de la complexité et des capacités des réseaux de neurones humains, en effet le comportement des potentiels d'activation des neurones peuvent être vaguement reproduits par la fonction d'activation associée, mais qu'en est-il des potentiels d'inhibition qui bloquent un neurone et qui jouent un rôle important lors de l'apprentissage [56, 57, 136]. Les périodes réfractaires absolues (temps nécessaire à un neurone entre deux potentiels d'action) [14, 21] sont également complètement ignorées par le modèle du perceptron alors que cela constitue une piste intéressante pour la gestion séquentielle des événements, même si quelques outils comme les *Spiking Neural Networks* (SNN) [67] ou les *Long Short-Term Memory* (LSTM) [60] ont été introduits pour gérer les flux de données. Après ma thèse dans l'équipe DEMAR, entouré de chercheurs et doctorants travaillant sur la modélisation du fonctionnement des neurones [11, 48, 126] et des muscles, l'analogie entre le perceptron et le cerveau humain me semblait totalement injustifiée, comme si on pouvait considérer créer une voiture à pédale en s'inspirant d'une voiture de luxe, comme illustré ironiquement sur la Figure 5.1. Le retour d'expérience vers la connaissance du cerveau humain se faisait également attendre.

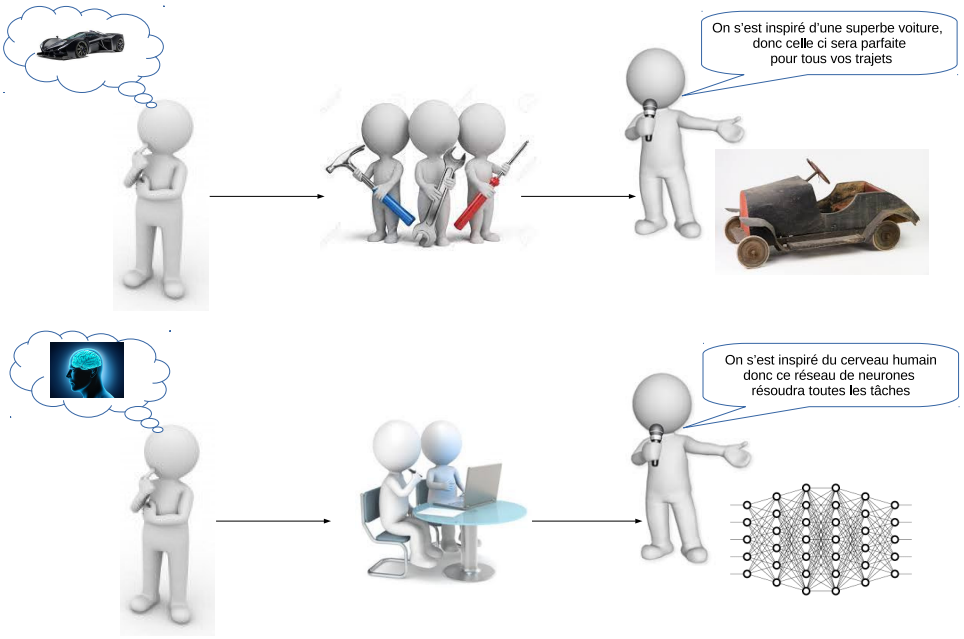


FIGURE 5.1 – Représentation ironique de mes premières impressions concernant la justification de l'utilisaton des réseaux de neurones en robotique via une analogie avec l'automobile.

En mettant de côté cette inspiration criticable, on peut également noter des grandes différences lors des phases d'apprentissage, en effet un enfant arrivera à apprendre une nouvelle tâche (marcher par exemple) en quelques semaines ou mois avec quelques essais par jour, alors que l'IA nécessitera des millions d'itérations correspondant à une longue période d'essai continue pour obtenir un résultat satisfaisant [116]. De plus, le résultat obtenu, qui est principalement les poids du réseau de neurones, sera difficilement interprétable et les preuves théoriques de bon fonctionnement seront difficilement accessibles. Généralement, on aura des preuves statistiques réalisées sur plusieurs essais. En reprenant l'analogie, le fabricant de la voiture à pédale de la Figure 5.1 ne serait pas en mesure d'expliquer ou de réparer une panne sur la voiture conçue. L'IA nous permet donc d'obtenir une boîte noire qui servira d'organe de commande ou de décision de notre système robotique.

Je restais donc assez sceptique sur la comparaison avec le cerveau humain, mais curieux des possibilités de ces réseaux de neurones et des impacts de leur utilisation en robotique tout en regrettant l'utilisation systématique de l'IA pour résoudre, comme par "magie", certains problèmes qui semblent accessibles via des méthodes analytiques.

Plus récemment, la justification des réseaux de neurones s'inspirant du cerveau humain n'est plus mise systématiquement en avant et ces réseaux de neurones ont prouvé leur aptitude à résoudre des problèmes complexes comme la reconnaissance d'objet dans les images [153], ou la manipulation d'objet [140]. Depuis 2017, avec la thèse de Mehdi Mounsif, nous avons commencé à étudier les possibilités de l'utilisation des réseaux de neurones pour des tâches robotiques en nous focalisant sur la capitalisation de l'étape d'apprentissage en explorant la thématique du transfert de compétences.

Actuellement, je considère toujours que les réseaux de neurones (convolutifs, récurrents, ...) ne reproduisent pas fidèlement les différents phénomènes mis en œuvre dans le cerveau humain. Je perçois les réseaux de neurones comme des fonctions d'approximation.

Ils permettent de reproduire des comportements qui peuvent s'apparenter aux comportements humains. De mon point de vue, la "magie" de ces réseaux de neurones intervient, grâce aux méthodes d'apprentissage, quand ils approximent un comportement qu'on ne sait pas (encore) décrire mathématiquement, comme la reconnaissance d'objet dans des images par exemple.

Plusieurs types de méthodes (supervisée, non supervisée, ...) peuvent être rattachées à l'IA. Dans la Section 5.2, je rappellerai le concept de base du *Reinforcement Learning* (RL). Dans la Section 5.3, je présenterai les travaux de la thèse de Mélodie Daniel sur l'utilisation de l'IA pour une tâche de manipulation d'objet déformable, puis je détaillerai les contributions établies dans le cadre des thèses de Mehdi Mounsif et Samuel Beaussant à la capitalisation de l'apprentissage considérant plusieurs robots définie comme le transfert de compétences dans la Section 5.4, avant de conclure sur les résultats obtenus et les perspectives concernant le transfert dans la Section 5.5.

5.2 Le *Reinforcement Learning*(RL)

Le RL ou Apprentissage par Renforcement en français, constitue un domaine de l'IA, qui explore comment les agents [1] peuvent apprendre à prendre des décisions optimales dans des environnements complexes et souvent imprévisibles. Cette approche se distingue par son accent sur l'apprentissage par l'expérience et la découverte, où les agents apprennent à atteindre leurs objectifs en interagissant avec leur environnement. À travers un système de récompenses et de pénalités, les agents apprennent à associer leurs actions à des résultats positifs ou négatifs, affinant ainsi progressivement leurs stratégies pour maximiser les gains cumulatifs. Le schéma classique de l'apprentissage en RL est présenté sur la Figure 5.2.

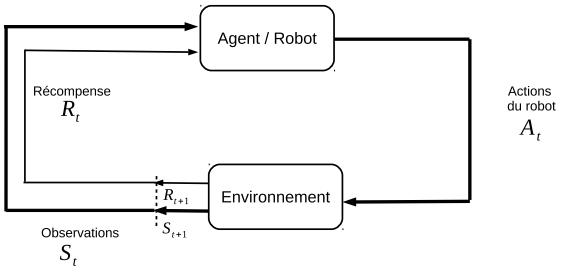


FIGURE 5.2 – Schéma de principe du *Reinforcement Learning*(RL) représentant l'interaction entre l'agent et l'environnement.

Avec :

— S_t l'observation de l'environnement à un instant t,

— A_t l'action à l'instant t issue de l'observation S_t et d'une politique π,

— R_t la récompense à l'instant t utilisée pour mettre à jour la politique π.

L'action A_t est issue de la politique (fonction) de l'agent et dépend des observations S_t, elle impactera l'environnement qui produira une nouvelle observation S_{t+1} et une nouvelle récompense R_{t+1} à l'instant $t+1$.

1. Dans ce manuscrit les termes **robot** et **agent** représentent la même chose et sont employés indifféremment.

La phase d'apprentissage se fait généralement dans un environnement de simulation qu'il faut choisir en fonction de la tâche désirée [184]. Dans le cadre de nos travaux nous avons principalement utilisé les environnements de simulation PyBullet [49] dans la Section 5.3 et Unity-3D [81] dans la Section 5.4.

Il existe plusieurs méthodes d'apprentissage qui permettent de mettre à jour les paramètres de la politique de l'agent (souvent matérialisée par un réseau de neurones) en fonction de la récompense, des observations et des actions [184]. Parmi les plus connues, nous avons principalement utilisé :

— DDPG est un algorithme de *Deep Reinforcment Learning* (DRL) basé sur les méthodes d'acteur critique [118]. Le DDPG est spécifiquement conçu pour traiter des tâches où l'action à prendre est continue, ce qui le rend adapté à la robotique. En utilisant un réseau de neurones profond pour approximer à la fois la politique (la manière de prendre des décisions) et la fonction de valeur (l'estimation de la récompense attendue), le DDPG apprend à optimiser la politique en maximisant la récompense cumulative au fil du temps. Il utilise des méthodes de gradient pour ajuster les paramètres du réseau de neurones afin d'améliorer progressivement la performance de la politique. Le DDPG utilise un algorithme *off-policy*, ce qui signifie qu'il maintient un historique des expériences passées pour mettre à jour sa politique.

— *Proximal Policy Optimization* (PPO) est un algorithme d'apprentissage par renforcement qui vise à résoudre des problèmes de contrôle en optimisant une politique de manière stable [159]. Il adopte une stratégie qui consiste à raffiner la politique actuelle en s'assurant qu'elle reste relativement proche de la politique précédemment établie. Cette méthode de mise à jour vise à favoriser des progrès constants et à prévenir les changements soudains et importants dans le processus d'apprentissage. Le PPO est réputé pour sa robustesse et sa capacité à traiter efficacement des tâches complexes en utilisant des réseaux de neurones profonds pour approximer la politique. Le PPO est un algorithme *on-policy*, ce qui signifie qu'il collecte activement de nouvelles données en utilisant la politique actuelle pour les mises à jour. Il ne réutilise pas les données historiques, ce qui peut le rendre plus efficace pour l'entraînement en ligne.

Le DDPG et le PPO sont principalement utilisés pour résoudre des problèmes de contrôle continu, même si le PPO est plus polyvalent. Le DDPG peut être plus sensible à des choix d'hyperparamètres mal réglés et peut être moins stable lors de l'entraînement sur certains problèmes alors que le PPO est réputé pour sa stabilité et sa facilité d'utilisation. Il est souvent préféré pour sa capacité à bien fonctionner avec des données bruitées ou des environnements mal modélisés.

5.3 Manipulation d'objet déformable

Contrairement aux objets rigides, les objets déformables ne peuvent pas être modélisés par des modèles standardisés, obligeant à développer des solutions sur mesure pour chaque type d'objet [63, 100, 175]. Dans ces conditions, il est compréhensible que le développement de méthodes de contrôle basées sur un modèle ne soit pas une tâche aisée et que l'utilisation de la simulation et des méthodes d'apprentissage constituent une piste à explorer.

Dans le cadre de la thèse de Mélodie Daniel [37], la manipulation d'un objet déformable a été mise en place comme présenté sur la Figure 5.3. Cette tâche de manipulation est

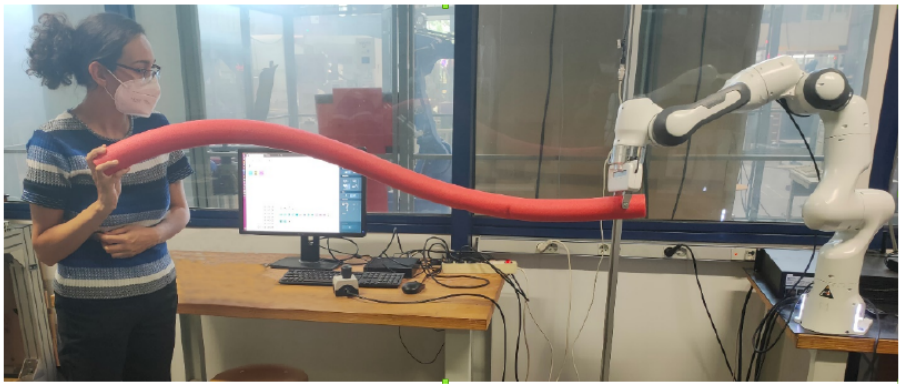


FIGURE 5.3 – Manipulation d'objet déformable source : [37].

définie via la minisation de la distance entre des points fixés à l'objet déformable et leurs positions désirées (issues de déformations préalables) comme montré sur la Figure 5.4.

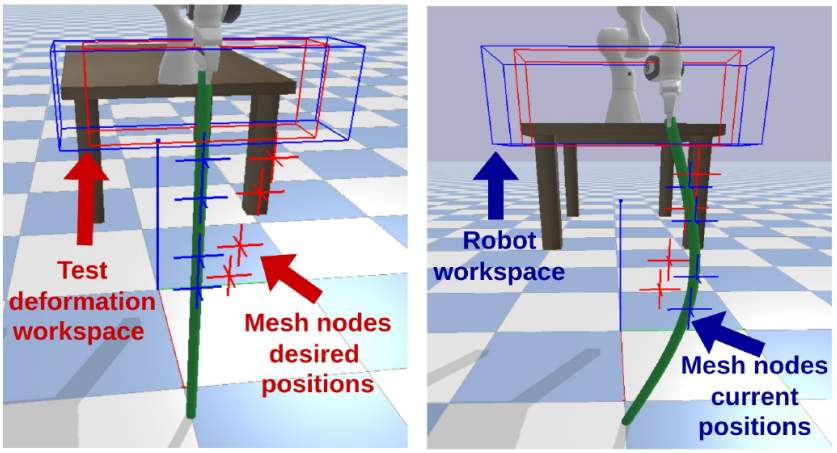


FIGURE 5.4 – Illustration des points appartenant à l'objet déformable en bleu et leurs positions désirées en rouge.

L'architecture d'apprentissage proposée pour résoudre cette tâche se base sur l'algorithme DDPG et est représentée sur la Figure 5.5. Pour réaliser la phase d'apprentissage, nous avons utilisé le simulateur PyBullet [49] pour lequel les paramètres de déformations de l'objet ont été déterminés par plusieurs expérimentations réelles afin d'obtenir une simulation réaliste. l'apprenttissage a été porté sur une architecture parallèle comprenant 32 coeurs CPU (1 par agent) et la librairie Python MPI [34] et a fourni un résultat en environ trois heures et demie. Ces travaux ont été validés en simulation et sur un robot réel comme présenté sur la Figure 5.6. Ils ont été publiés dans [64] dans le cadre de sa thèse et poursuivis avec une publication dans [38] dans le cadre d'un post-doctorat.

L'expérimentation sur un robot réel de cette tâche de manipulation d'un objet déformable a nécessité un gros investissement humain de la part de Mélodie Daniel afin de maîtriser les aspects simulation, apprentissage et expérimentation. Bien que ses travaux soient disponibles[2], la ré-utilisation de ses résultats par un autre utilisateur, en considérant un autre robot ou une tâche différente nécessiteront également un investissement important, notamment pour la phase d'apprentissage qui devra obligatoirement être relancée.

2. sur ce lien : `https://github.com/MelodieDANIEL/robotic_control_of_DLO_using_DRL`

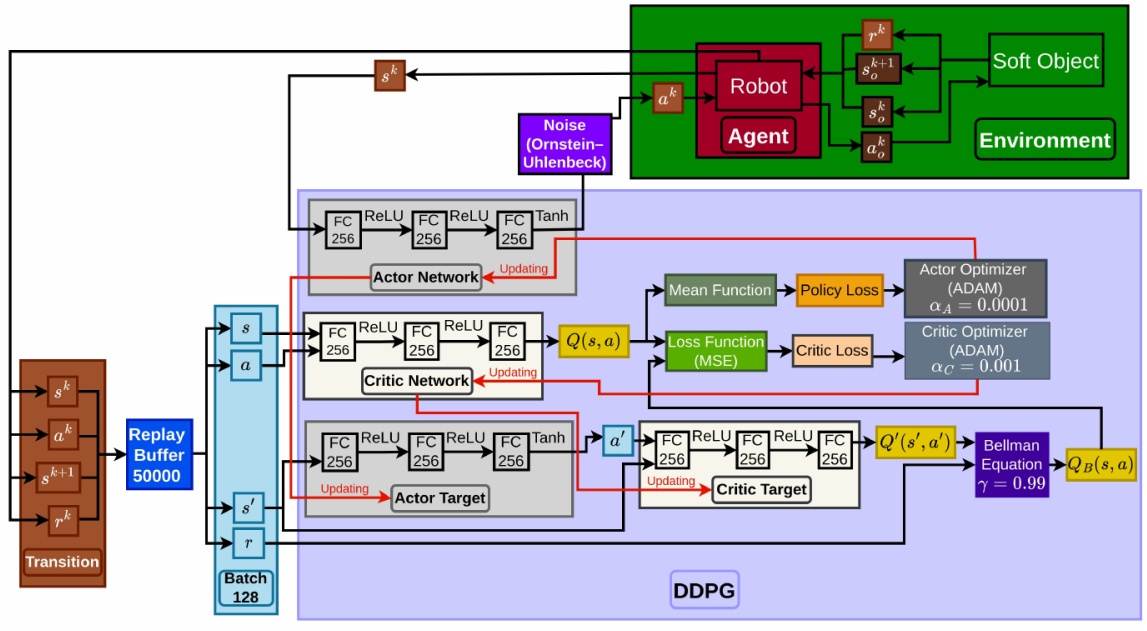


FIGURE 5.5 – Représentation de l'approche mise en place pour le contrôle de la déformation d'un objet souple via l'algorithme DDPG.

La section suivante aborde la possibilité de capitaliser les résultats obtenus lors de l'apprentissage d'une tâche par un robot afin de pouvoir les ré-utiliser dans le cas d'une tâche ou d'un robot différent.

5.4 Transfert de compétences

Comme présenté précédemment dans la Section 5.3, les méthodes de RL permettent de résoudre des problèmes complexes comme la manipulation d'objet déformable [74, 125, 155] ou la résolution d'un Rubik's Cube liant manipulation fine et ordonnancement [140]. Ces méthodes s'appuient sur une phase d'apprentissage qui nécessite un grand nombre d'itérations et qui doit être propre à chaque robot utilisé. Les travaux de [75] proposent un état de l'art sur la notion de transfert en la ramenant à trois questions : **Quand** le transfert est-il utile ? **Quoi** ou quelle partie doit être transférée ? et **Comment** mettre en place le transfert ? Dans cette section, nous allons proposer une réponse au "**Comment ?**" et au "**Quoi ?**"en proposant une architecture qui définit quelles parties du réseau sont transférables afin de capitaliser l'expérience acquise par un robot lors de sa phase d'apprentissage en vue de la réutiliser avec un autre robot comme le présente la vue d'artiste de la Figure 5.7.

5.4.1 Formalisation du problème

Nous allons commencer par définir l'ensemble des notions nécessaires à la formalisation du problème de transfert de compétences.

Robot (ou Agent) : Dans le domaine de l'IA on fait généralement appel au terme d'agent pour indiquer le processus agissant ou prenant des décisions. Dans ce document nous ne distinguerons pas le terme agent du terme robot. Un robot r sera donc défini par

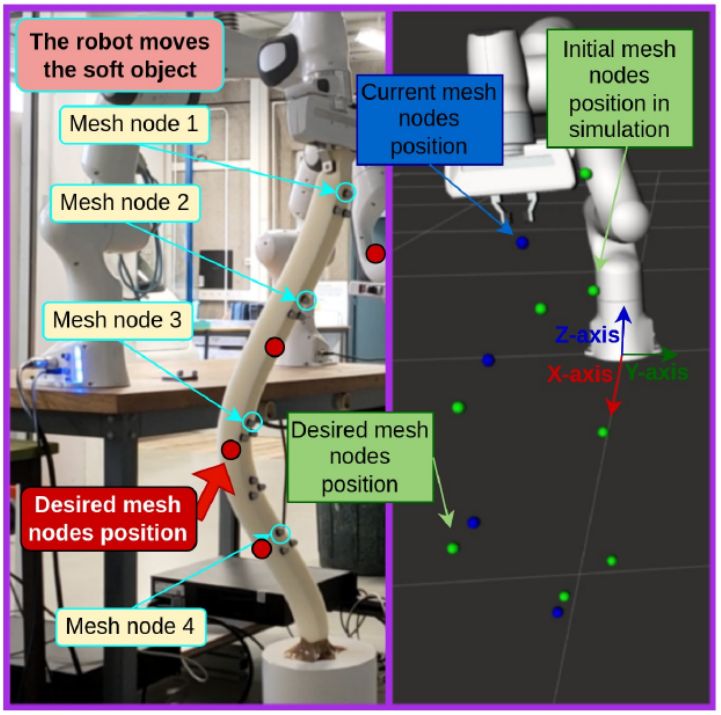


FIGURE 5.6 – Expérimentation de contrôle de la déformation d'un objet.

son espace d'état S_r et son espace d'action A_r. Les dimensions de ces deux espaces sont reliées au nombre de degrés de liberté du robot, elles peuvent donc différer d'un robot à un autre.

Tâche : Le dictionnaire Larousse définit une tâche comme un travail, un ouvrage à faire dans un temps déterminé et à certaines conditions [3]. Nous définissons une tâche $\mathcal{T}$ par l'état désiré sur une partie ou la totalité de l'environnement (position finale des objets, ...) et éventuellement par les différentes opérations nécessaires pour atteindre cet état. Par exemple, la tâche vissage serait définie par la position finale de l'écrou (sur la vis) en imposant l'utilisation d'un outil (visseuse, clé, ...).

Récompense : Lors de l'apprentissage, une tâche $\mathcal{T}$ sera définie par une fonction récompense $R_{r,\mathcal{T}}$ permettant de définir l'état désiré. Elle pourra également contenir des informations propres au robot (par exemple afin de minimiser son énergie, ...). Pour favoriser l'apprentissage, il est préférable d'avoir une récompense $R_{r,\mathcal{T}}$ "progressive" [4] qui permettra de guider l'apprentissage même si la tâche n'est pas encore réalisée.

Performance : Une fois l'apprentissage effectué, l'évaluation de la bonne réalisation de la tâche $\mathcal{T}$ est décrite par une métrique de performance $\mathcal{P}_{r,\mathcal{T}}$. La performance $\mathcal{P}_{r,\mathcal{T}}$ peut être binaire pour spécifier le succès ou l'échec de la réalisation de la tâche. La performance sera l'élément prépondérant lors de l'évaluation d'un transfert de compétences entre robots.

3. `https://www.larousse.fr/dictionnaires/francais/tache/76339`
4. Si on souhaite atteindre une cible, une récompense progressive telle que la distance permettra un meilleur apprentissage qu'une récompense binaire du type cible atteinte ou non atteinte.

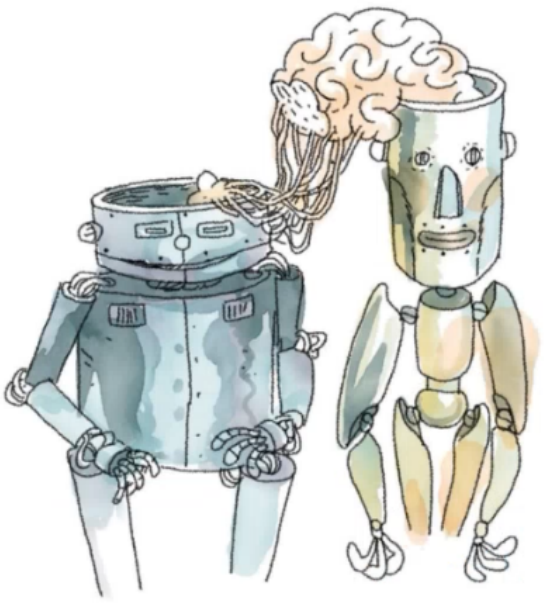


FIGURE 5.7 – Représentation artistique du transfert de compétences en robotique, source : [1].

Monde : Nous définissons un monde (*world*) $\mathcal{W}_{r,\mathcal{T}}$, comme la combinaison d'un robot r, d'une tâche $\mathcal{T}$ et de l'environnement (objets, ...) nécessaire à la réalisation de la tâche [5]. On peut formuler un monde sous la forme d'un *Markov Decision Process* (MDP) [71, 93] :

$$\mathcal{W}_{r,\mathcal{T}} = \langle S_r, A_r, \rho_{r,\mathcal{T}}, s_0, R_{r,\mathcal{T}} \rangle \tag{5.1}$$

avec :

- S_r et A_r : l'espace d'état et l'espace d'action du robot r,
- $\rho_{r,\mathcal{T}}$: la fonction de transition d'un état $s_r(t_i)$ à l'état $s_r(t_{i+1})$ sous l'action $a_r(t_i)$ (module lié à l'environnement dans la Figure 5.2),
- s_0 : l'état initial,
- $R_{r,\mathcal{T}}$: la fonction de récompense liée au robot r et à la tâche $\mathcal{T}$.

On désigne une politique comme une règle ou une stratégie qui définit les actions qu'un agent doit entreprendre en fonction des informations d'état de l'environnement donné. Dans les MDP, l'objectif est de trouver une politique $\pi_{r,\mathcal{T}}$ qui maximise une récompense cumulée G_φ sur un horizon de temps T telle que :

$$G_\varphi = \sum_{t=0}^{T} \gamma^t R_{r,\mathcal{T}} \tag{5.2}$$

Avec $\gamma \in [0:1]$ un hyper-paramètre permettant de pondérer les récompenses récentes par rapport aux plus anciennes et φ est une trajectoire définie comme une séquence d'état-action :

$$\varphi = ((s_0, a_0), (s_1, a_1), (s_2, a_2), \ldots) \tag{5.3}$$

Au final, on recherche une politique $\pi^*_{r,\mathcal{T}}$ qui maximise la récompense cumulée :

$$\pi^*_{r,\mathcal{T}} = \underset{\pi_{r,\mathcal{T}}}{\operatorname{argmax}}\, \mathbb{E}_{\varphi\sim\pi}\left[G_\varphi\right] \tag{5.4}$$

5. Pour faciliter les notations, nous ne considérons qu'un seul robot r pour réaliser la tâche, mais cette formulation peut être étendue à un ensemble de robots lors de tâches de co-manipulation, par exemple.

Univers : Dans le cadre du transfert de compétences on définit un univers $\mathcal{U}$ comme un ensemble de n mondes composé à partir de plusieurs robots et de plusieurs tâches à réaliser :

$$\mathcal{U} = \{\mathcal{W}_{r_1,\mathcal{T}_1}, \mathcal{W}_{r_2,\mathcal{T}_2}, \ldots, \mathcal{W}_{r_n,\mathcal{T}_n}\} \tag{5.5}$$

avec r_i et $\mathcal{T}_i$ le robot et la tâche considérée dans le monde i. On supposera que les mondes diffèrent l'un de l'autre par le robot ou par la tâche :

$$\forall i, \nexists j \neq i \text{ tel que } r_i = r_j \text{ et } \mathcal{T}_i = \mathcal{T}_j \tag{5.6}$$

Transfert de compétences : Nous définissons un univers source $\mathcal{U}_s$ contenant tous les mondes qui ont bénéficié d'une phase d'apprentissage afin d'obtenir une politique adaptée. Nous définissons également un univers objectif (*target*) $\mathcal{U}_t$ contenant tous les mondes n'ayant pas été utilisés lors de la phase d'apprentissage. On peut considérer que :

$$\mathcal{U}_s \cap \mathcal{U}_t = \emptyset \tag{5.7}$$

Dans le cadre de ce manuscrit nous considérons que l'objectif du transfert de compétences est de mettre en place une architecture qui permette de déduire les politiques de l'univers objectif $\mathcal{U}_t$ à partir de celles apprises dans l'univers source $\mathcal{U}_s$ en maximisant les performances [6] des robots à réaliser les tâches dans les deux univers :

$$\forall \mathcal{W} = \langle S_r, A_r, \rho_{r,\mathcal{T}}, \nu_0, R_{r,\mathcal{T}} \rangle \in \{\mathcal{U}_s \cup \mathcal{U}_t\} \quad \pi^*_{r,\mathcal{T}} = \underset{\pi_{r,\mathcal{T}}}{\operatorname{argmax}} \mathcal{P}_{r,\mathcal{T}} \tag{5.8}$$

Note : Suivant cette définition on peut conclure qu'une méthode de transfert pertinente utiliserait un univers source $\mathcal{U}_s$ ne comportant que peu de mondes comparés à ceux de l'univers objectif $\mathcal{U}_t$ ($dim(\mathcal{U}_s) \ll dim(\mathcal{U}_t)$).

Hypothèse : Pour pouvoir résoudre ce problème de transfert, nous faisons l'hypothèse que chaque robot ou tâche de l'univers objectif est présent au moins une fois dans un monde de l'univers source :

$$\forall r \in \mathcal{U}_t, \exists \mathcal{W} \in \mathcal{U}_s : r \in \mathcal{W} \quad \text{et} \quad \forall \mathcal{T} \in \mathcal{U}_t, \exists \mathcal{W} \in \mathcal{U}_s : \mathcal{T} \in \mathcal{W} \tag{5.9}$$

5.4.2 Etat de l'art

Vision par ordinateur

Le besoin de transfert de compétences apprises par un réseau de neurones s'est rapidement exprimé dans les tâches de vision par ordinateur. En effet, on s'est rapidement posé la question de savoir comment profiter des premiers travaux de classification basés sur la base de données ImageNet [68, 97, 162, 165, 182]. Les modèles entraînés sur ce jeu de données peuvent apprendre à extraire un large éventail de caractéristiques comme la décomposition des images en bords, formes et contours. La représentation d'une image dans les couches internes du réseau a de fortes chances de se généraliser à d'autres domaines [47, 152]. Comme ces caractéristiques sont assez générales, l'extracteur de caractéristiques peut être efficacement réutilisé pour d'autres tâches dans d'autres domaines comme la reconnaissance de pathologies sur des images médicales [176, 177, 179]. Après le transfert de l'extracteur de caractéristiques (partie interne du réseau), il ne reste plus qu'à

6. On se focalise bien ici sur les performances finales et pas sur la fonction de récompense.

affiner les dernières couches du réseau, réduisant ainsi considérablement l'apprentissage de la tâche.

Même si les images initiales n'ont pas la même dimension, il est assez facile de changer la taille d'une image pour qu'elle puisse être fournie au réseau. Si on considère une application robotique, la taille de l'entrée va dépendre du nombre de degrés de liberté du robot. Cette différence de degrés de liberté entre deux robots ne peut pas facilement être gérée pour passer de l'état d'un robot à un autre. Ces méthodes semblent donc intrinsèquement difficiles à mettre en place et apparaissent peu efficaces lorsque c'est possible [8].

Le transfert en robotique

Parmi les différents travaux qui traitent du transfert en robotique ([7, 122, 160, 163, 181]), nous présentons, dans cette partie, les méthodes qui ont inspiré le développement de nos contributions.

La mise en place d'un espace de représentation invariant (ou espace latent) est la base de nos travaux pour effectuer le transfert. L'objectif principal d'un espace de représentation est de produire un espace plus petit que l'espace initial dans lequel il est possible de trouver une politique plus efficace pour résoudre un problème [115]. Les travaux sur lesquels nous nous sommes penchés pour la construction de cet espace invariant ont été présentés dans [61] et sont représentés sur la Figure 5.8. Pour cela ils considèrent deux robots distincts et un ensemble d'états définis comme similaires. La construction de l'espace invariant se base sur deux structures encodeur-décodeur (une par robot) dont les paramètres sont optimisés (ou appris). L'objectif est de minimiser la distance entre la projection des états (l'encodage), via les fonctions f et g, des états des deux robots dans l'espace invariant et de permettre un décodage correct des états encodés, comme le montre la Figure 5.8.

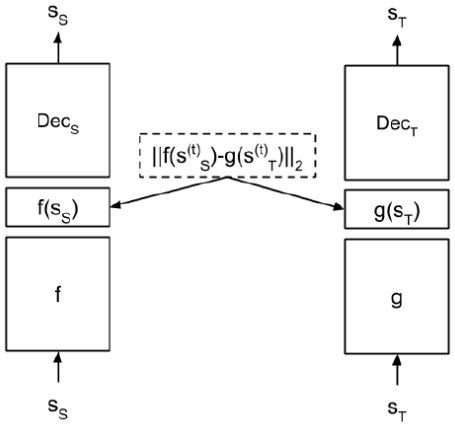


FIGURE 5.8 – Construction de l'espace de représentation invariant, source [61].

Une fois l'espace latent construit et qu'un robot source a appris une tâche, il est possible de guider l'apprentissage du robot *target* en rajoutant un terme afin que la politique du robot *target* se rapproche de ce qu'aurait donné la politique du robot source dans un état équivalent. Dans ce cadre on a donc un transfert indirect comme présenté dans la Section 5.4.3.

L'introduction d'une représentation modulaire est apparue très intéressante afin de pouvoir dissocier les comportements propres à chaque robot des comportements nécessaires à la réalisation de la tâche. Les travaux présentés dans [44] démontrent qu'il est possible d'associer chaque tâche et chaque robot à des modules différents. La réalisation de la tâche par un robot se fera alors en associant les modules correspondants comme montré sur la Figure 5.9.

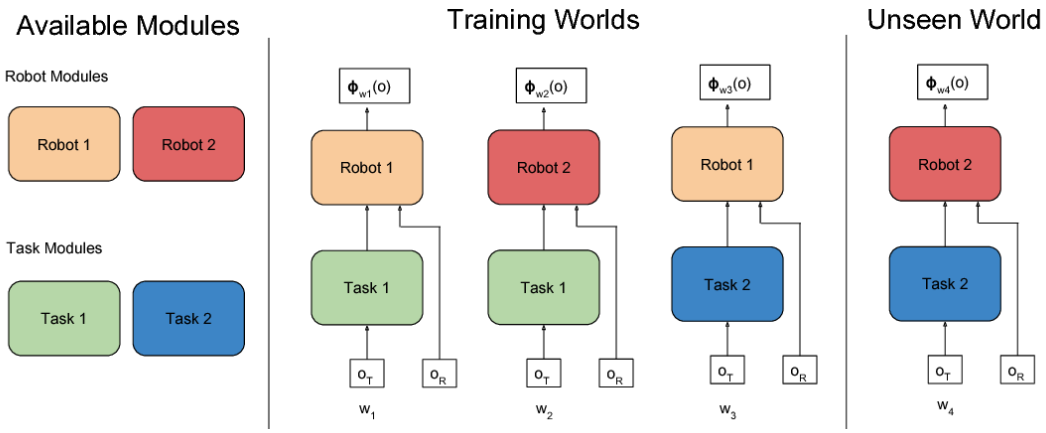


FIGURE 5.9 – Représentation des modules de tâches et de robots et définition des mondes *source* et *target*, d'après [44].

Dans ce cas, le transfert nécessite uniquement l'existence de modules indépendants sans requérir d'apprentissage supplémentaire, ce qu'on peut définir comme un transfert direct comme présenté dans la Section 5.4.3. La notion de monde introduite dans [44] nous a également servi de base de réflexion pour la définition du problème de transfert présentée dans la Section 5.4.1.

Du monde de la simulation au monde réel : le *Sim2real*

Dans la plupart des cas, l'objectif est de réaliser une tâche de manière autonome grâce à un robot réel. Cependant, l'apprentissage de la tâche dans le monde réel soulève des préoccupations en matière de sécurité, de temps et du volume important d'échantillons requis. Même si cela a déjà été réalisé avec des moyens conséquents [117], cette méthode reste hors de portée pour un grand nombre d'utilisateurs. Par conséquent, la phase d'apprentissage se déroule généralement dans un simulateur, profitant de la disponibilité de données quasi illimitées et évitant les dommages physiques potentiels. Malheureusement, les agents d'apprentissage par renforcement ont tendance à s'adapter excessivement à leur environnement. Même les simulateurs de pointe [123, 167] ne peuvent pas reproduire entièrement la richesse du monde réel, créant des écarts lors du transfert des modèles de la simulation au monde réel, menant au problème du *sim2real*. Ainsi, le transfert de la simulation (monde source) à la réalité (monde cible) peut également être considéré comme un problème de transfert de compétences suivant la définition présentée dans la Section 5.4.1.

Certains travaux ont mis en place des techniques spécifiques au problème du *sim2real*. Pour atténuer l'effet néfaste sur les performances des modèles, le *Domain Randomization* [140, 144] change de manière aléatoire certains paramètres physiques et dynamiques pour exposer l'agent à une grande variété d'environnements possibles, en espérant que cette variété englobe l'environnement réel.

D'autres travaux associent l'apprentissage issu de simulations et d'expérimentations réelles soit en faisant évoluer la distribution des paramètres physiques en fonction des

résultats de quelques expériences réelles [24] soit en rajoutant des termes de récompense afin de guider l'exploration de l'agent réel en fonction des performances de l'agent en simulation et vice-versa [178]. Un point important, que ce soit dans les méthodes de contrôle classique ou basées sur l'apprentissage, concerne la gestion des retards inclus dans les processus d'observations et de commande. Le retard n'a été que peu étudié [25, 156] dans le cadre du *sim2real*, peut-être car les travaux actuels supposent le retard connu et identifié afin de l'inclure dans l'apprentissage, ce qui est possible si on se restreint à ce type de problème. Mais si on considère le problème plus large défini dans la Section 5.4.1, où on inclut plusieurs robots potentiellement différents et pouvant avoir des retards différents, la gestion du retard est pour l'instant ignorée dans le problème du *sim2real*. Dans le cadre des travaux de recherche de Samuel Beaussant [8], nous avons proposé une contribution qui est détaillée dans la Section 5.4.7.

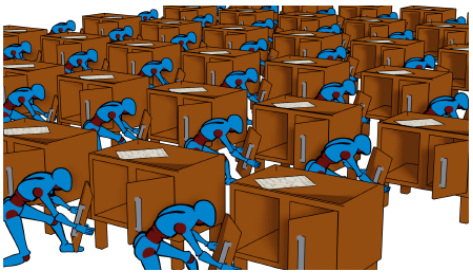
5.4.3 Différentes approches de transfert

On peut représenter les approches de transfert en deux grandes catégories :

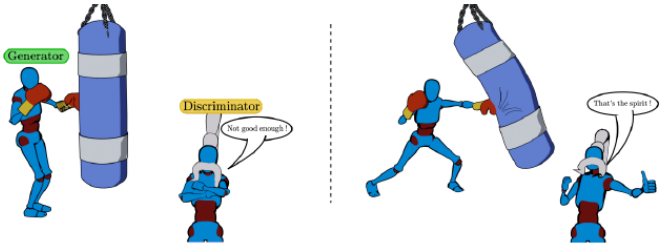
— Les approches de transfert directes construisent un nouvel agent en agglomérant des parties issues d'apprentissage précédents. Ces approches ne nécessitent pas d'apprentissages supplémentaire, en théorie.

— Les approches de transfert indirectes effectuent un apprentissage du nouvel agent en se basant sur d'autres agents déjà entraînés. Ces approches nécessitent un apprentissage supplémentaire qui devra être plus rapide ou plus simple à mettre en place qu'un apprentissage *from scratch* afin de conserver l'intérêt du transfert.

Dans le cadre de la thèse de Mehdi Mounsif [130], nous avons abordé deux types d'approche pour le transfert de compétences, les approches directes centrées sur la tâche *Task-Centered* (TAC) et les approches indirectes centrées sur l'enseignant *TEacher-Centered* (TEC) comme présentée sur la Figure 5.10.



(a) Les méthodes TAC se basent sur une description haut-niveau (une notice) de la tâche à réaliser.



(b) Les méthodes TEC se basent sur la compétence d'un expert pour guider l'apprentissage.

FIGURE 5.10 – Comparaison des différents types d'approches pour le transfert de compétences (source [130]).

Les approches directes TAC s'appuient sur l'idée de représentation haut-niveau, semblable aux notices utilisées par des humains non professionnels pour être rapidement capables de produire un objet (par exemple, monter un meuble) sans avoir été spécifiquement formés à ce type de tâche. Dans ce cas, le producteur de meubles rédige un ensemble d'instructions qui doivent permettre à une autre personne d'atteindre son objectif, à savoir le montage du meuble nouvellement acquis. Dans cette configuration, le producteur n'a

pas accès aux capacités physiques du client, mais fait l'hypothèse d'une motricité de base qui lui permettrait de se conformer à l'ensemble des instructions de la notice. Dans cette optique, les méthodes TAC visent d'abord à construire cette représentation haut-niveau (notice) indépendamment de la morphologie de l'agent agissant afin de le transmettre à d'autres agents qui exécuteront alors la tâche. Cette approche est schématisée sur la Figure 5.10a et plusieurs contributions sont présentées dans les Sections 5.4.6, 5.4.7, 5.4.8.

La deuxième approche est une approche indirecte TEC centrée sur l'enseignant. Dans cette optique, au lieu d'avoir une segmentation entre la morphologie de la connaissance et celle de l'agent comme dans les TAC, la connaissance de la tâche est liée à un agent, appelé enseignant ou expert. L'objectif des méthodes TEC est de fournir des moyens de distiller cette capacité dans un autre agent étudiant de morphologie potentiellement différente, comme présenté dans la Figure 5.10b. Cette configuration correspond à des situations très courantes dans la civilisation humaine. En effet, il existe de nombreux exemples où une personne non formée peut bénéficier des connaissances d'un expert, réduisant ainsi le temps nécessaire pour atteindre la maîtrise. Deux contributions sont présentées dans les Sections 5.4.4 et 5.4.5.

5.4.4 *CoachGAN*

Generative Adversarial Networks (GAN)

Comme son nom l'indique la méthode *CoachGAN* [133] reprend le principe des GAN : un type d'architecture d'IA utilisée pour la génération de données, d'images, de son, de texte, ...[58, 59, 90]. Le principe des GAN, présenté sur la Figure 5.11 repose sur deux réseaux neuronaux distincts qui travaillent en opposition :

— Le générateur a pour but de créer des données (par exemple des images). L'objectif est d'apprendre à produire des données de plus en plus réalistes au fil du temps.

— Le discriminateur a pour mission de distinguer les données réelles (fournies pendant l'entraînement) des données générées par le générateur. Il essaie de classifier les images en "fausse" ou "réelle".

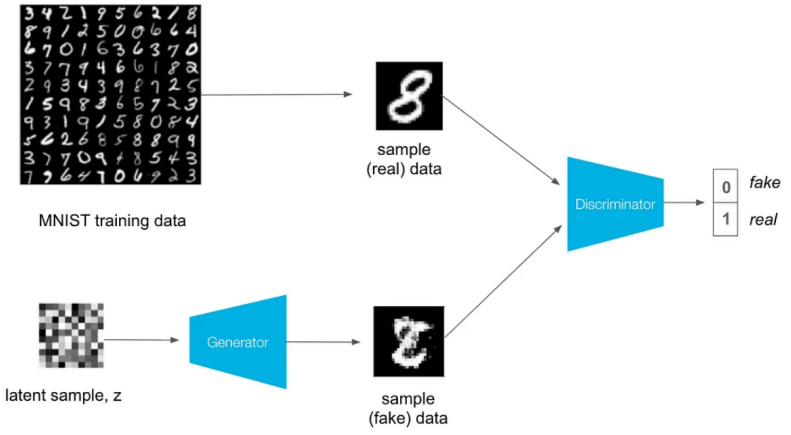


FIGURE 5.11 – Principe du GAN : le générateur essaie de produire des images pour tromper le discriminateur (source).

L'entraînement d'un GAN est donc une sorte de compétition où le générateur s'efforce de tromper le discriminateur en créant des données de plus en plus convaincantes, tandis

que le discriminateur s'améliore pour mieux distinguer le vrai du faux. Au fil du temps, les deux réseaux s'améliorent, conduisant à la production de données très réalistes par le générateur.

le GAN pour le transfert

Contrairement à la plupart des applications des GANs où le générateur est le seul modèle souhaité et où le discriminateur est généralement écarté, la méthode *CoachGAN* utilise le discriminateur comme l'enseignant qui va enseigner à l'étudiant. Le principe général du *CoachGAN*, présenté sur la Figure 5.12 issue de [130], est composé de deux étapes :

— l'apprentissage de l'enseignant (du discriminateur) a pour but de déterminer si des trajectoires (décrites dans un espace *Intermediate State Variables* (ISV)) sont issues d'un expert, réalisant la tâche, ou non.

— l'apprentissage du générateur de l'étudiant qui va optimiser l'évaluation du discriminateur pour produire des mouvements qui se rapprochent de ceux de l'expert.

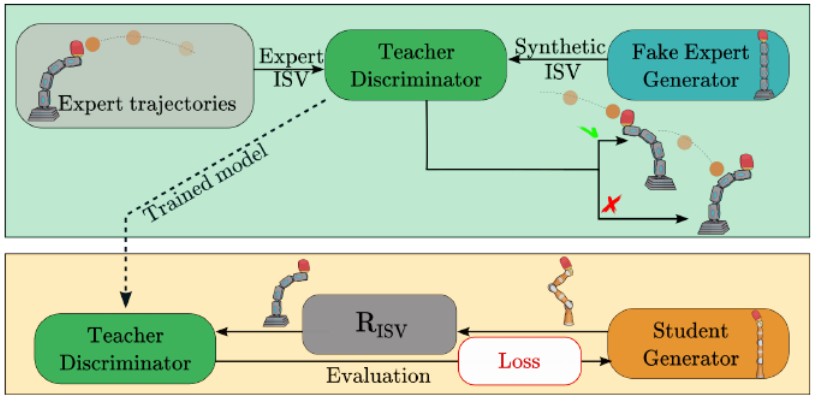


FIGURE 5.12 – Principe du coachGAN : le discriminateur est appris sur des trajectories expertes, puis il guide l'apprentissage du générateur/étudiant (source : [130]).

Le concept principal de cette approche est de définir un ISV qui peut être évalué par le discriminateur (enseignant), basé sur les représentations apprises sur les trajectoires des experts, et auquel l'étudiant peut se rattacher. Il sera alors possible de rétropropager l'erreur de l'étudiant dans son modèle et de l'optimiser en fonction de l'évaluation du discriminateur de l'enseignant. L'ISV est essentiel à l'approche CoachGAN. En effet, l'expert et l'élève ayant des morphologies différentes, il n'est pas évident de comparer directement leurs états/actions. Le rôle de l'ISV est de combler le fossé entre les deux agents en étant la représentation d'un état intermédiaire commun, défini par l'utilisateur.

Résultats

La méthode *CoachGAN* a été évaluée sur une tâche 2D d'un robot série devant intercepter une balle avec son effecteur. Le robot source était un robot à 4 degrés de liberté qui a appris la tâche en 4h avec un taux de réussite de 85%. Le robot *target* était un robot à 5 degrés de liberté avec un taux de réussite de 70% pour un temps de calcul inférieur à 20

minutes[7]. Ces premiers résultats de la méthode *CoachGAN* semblent prometteurs même si il sont perfectibles.

Au final, l'avantage de cette méthode est qu'il n'est pas nécessaire que les trajectoires "expertes" soient issues d'un apprentissage, elles pourraient provenir de trajectoires d'experts humains dans le cadre d'un apprentissage par imitation. Les principaux inconvénients portent sur la définition de ces ISV et sur la nécessité d'effectuer un nouvel apprentissage pour chaque monde $\mathcal{W}$ de l'univers objectif $\mathcal{U}_t$, c'est-à-dire pour chaque couple robot+tâche, ce qui n'est pas compatible avec la notion de transfert "direct".

5.4.5 *Task-Specific Loss*(TSL)

Principe

Comme nous l'avons vu dans la Section (5.4.4), la méthode CoachGAN permet de transférer les connaissances d'un expert à un étudiant morphologiquement distinct en se basant sur la comparaison de variables internes (ISV). L'idée de base de la *Task-Specific Loss*(TSL) est que si on considère deux états initiaux proches (pour l'expert et l'étudiant) alors la variation de ces états devrait être semblable. Cette méthode considère donc la comparaison de la variation de variables internes. Nous avons proposé la méthode *Task-Specific Loss*(TSL) [132] en se basant sur l'architecture VAE.

Variational Auto Encoder(VAE)

Les *Variational Auto Encoder*(VAE) ont initialement été utilisés pour faire du transfert de style d'images [119]. Sans entrer dans le détail, le principe d'un *Auto Encoder* (AE) est relativement simple : on considère un encodeur qui va "compresser" les données d'entrée en une information de dimension plus petite et un décodeur qui va "décompresser" ces informations afin de reconstruire l'information initiale. Dans le cas du transfert de style on peut visualiser cela par le fait que l'encodeur extrait les informations principales qui différencient les images et le décodeur permet de les reconstruire en incluant tout ce qui est commun aux images (comme le style). Les VAE, comme celui représenté sur la Figure 5.13, diffèrent des AE par l'ajout de contraintes probabilistes afin d'avoir une meilleure distribution de l'espace latent pour le rendre continu, ce qui permet une meilleure généralisation.

Procédure de transfert

Dans le cas du transfert de compétences en robotique, nous nous sommes intéressés à la reconstruction des actions du robot en utilisant le principe représenté sur la Figure 5.14. En connaissant, les trajectoires (états s et actions a) la TSL construit un espace latent z. De cet espace, en considérant l'état s du robot, le décodeur permettra d'approximer l'action $\tilde{a}$. Une fois ce VAE entraîné, on en conservera que la partie décodeur.

L'étape de transfert se fera alors comme montré sur la Figure 5.15, les états des robots sources et cibles seront en entrée de leur décodeur respectif ainsi que le signal z issu d'une probabilité de distribution (afin d'avoir une bonne généralisation). Les décodeurs permettent de générer une action qui entrainera un nouvel état pour chaque robot. Dans

7. Ces données sont considérées dans le cas où l'ISV est la position de l'effecteur. [130] présente également les résultats en prenant l'ISV comme une approximation de la position de l'effecteur et comme une prédiction de la vitesse du rebond.

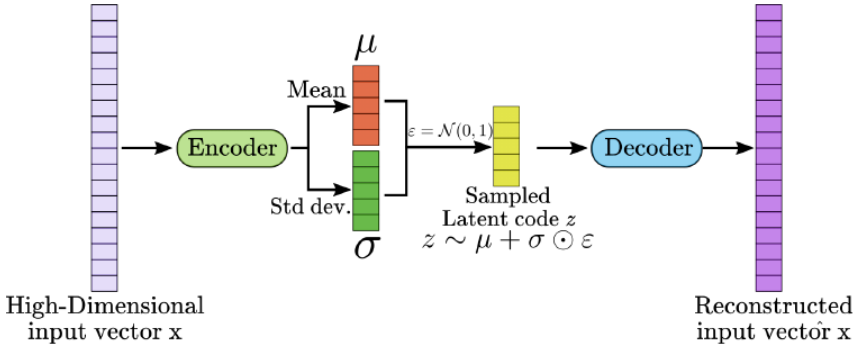


FIGURE 5.13 – Représentation d'un VAE, source : [130].

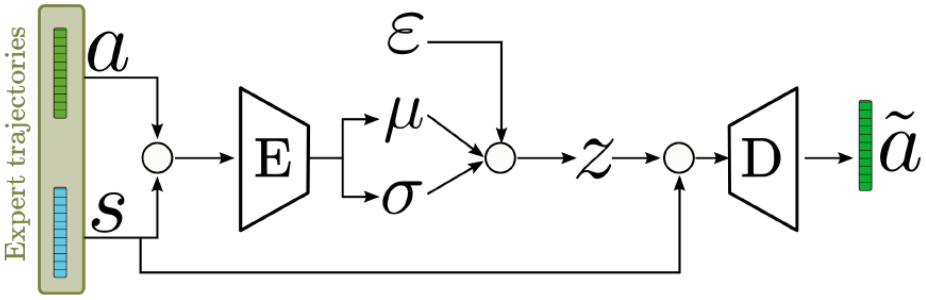


FIGURE 5.14 – Représentation du VAE pour la reconstruction de l'action d'un robot expert, source : [130].

cette étape de transfert, l'objectif est d'apprendre le décodeur du robot cible [8] afin qu'il fournisse une action menant à un état similaire si l'état initial est également similaire.

Résultats

Le TSL a été étudié et validé en simulation dans le cas d'un robot 2D passant d'un robot de 3 à 5 degrés de liberté. Les résultats sont présentés plus précisément dans [130]. Une extension de cette méthode a succintement été testée afin de permettre à un opérateur humain de piloter le robot via l'état z comme indiqué sur la Figure 5.16. Les premiers résultats semblent intéressants mais le panel de testeurs humains étant restreint, il nécessite d'autres tests pour valider une quelconque conclusion.

Les méthodes *CoachGAN* et TSL ont montré des capacités intéressantes en simulation. Cependant, dans ces cas, le transfert nécessite une phase d'apprentissage. Nous allons maintenant nous focaliser sur l'approche UNN dont l'objectif est de proposer un transfert direct.

5.4.6 *Universal Notice Network*(UNN)

L'*Universal Notice Network*(UNN) a été proposé initialement dans le cadre de la thèse de Mehdi Mounsif [130]. Les variantes DA-UNN et LS-UNN ont été proposées dans le cadre de la thèse de Samuel Beaussant [8].

8. L'encodeur du robot cible n'est jamais appris.

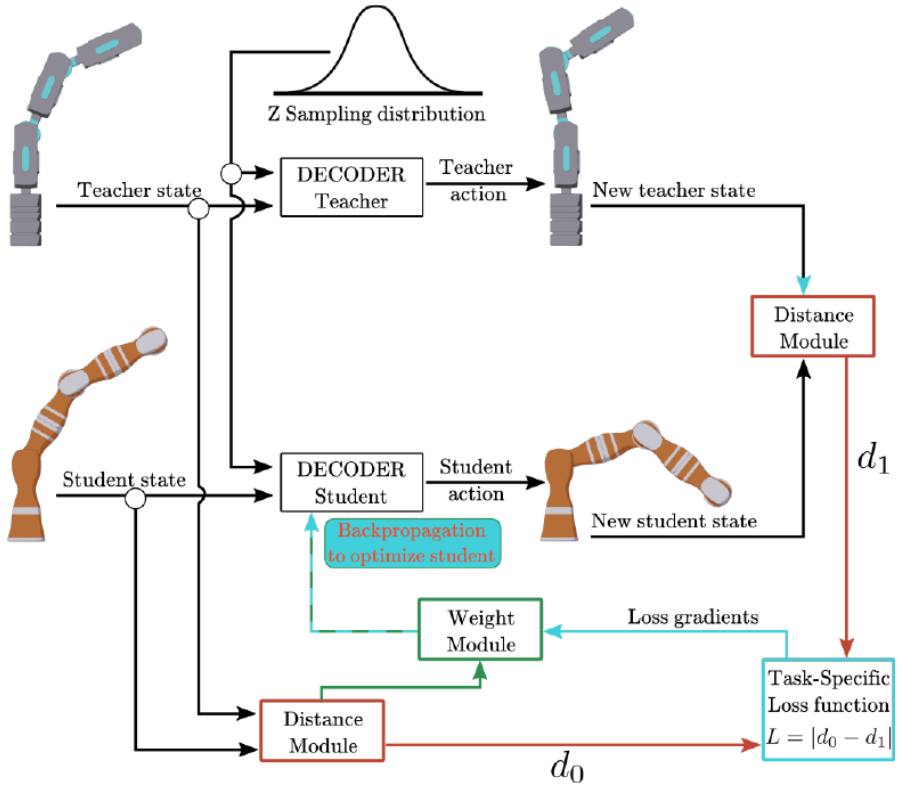


FIGURE 5.15 – Principe du transfert de la méthode TSL, seul le décodeur du robot cible est appris, source : [130].

Architecture

L'UNN se base sur les méthodes de RL présentées dans la Section 5.2. L'architecture générale de l'UNN est donc un cas particulier du schéma classique du RL présenté Figure 5.2. Nous décrivons la politique $\pi_{r,\mathcal{T}}$ d'un robot r pour réaliser une tâche $\mathcal{T}$ à partir de trois blocs comme montré sur la Figure 5.17. L'UNN suppose qu'il est possible de décomposer les observations en deux types d'informations : les informations propres au robot x_r et les informations propres à la tâche $o_{\mathcal{T}}$ [9] :

$$s = \{x_r, o_{\mathcal{T}}\} \tag{5.10}$$

9. Dans un premier temps, on peut supposer que les informations propres au robot x_r sont issues des capteurs proprioceptifs et les informations propres à la tâche $o_{\mathcal{T}}$ issues de capteurs extéroceptifs, mais une étude plus approfondie sur ce point est nécessaire pour pouvoir l'affirmer.

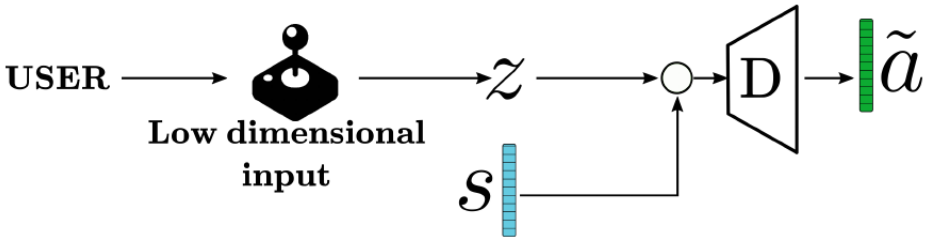


FIGURE 5.16 – Principe de contrôle humain via l'espace latent en utilisant la méthode TSL source : [130].

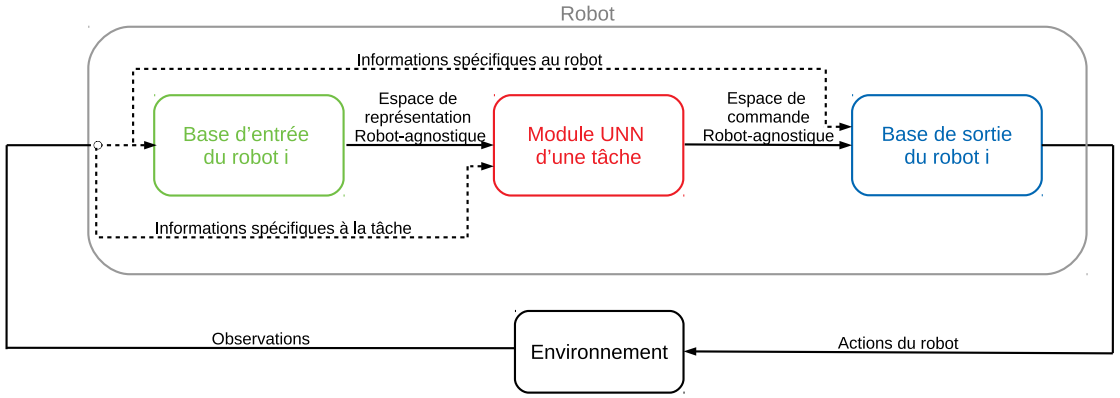


FIGURE 5.17 – Architecture de l'UNN

Les informations propres au robot x_r passent dans une base d'entrée B_i^r liée au robot r pour fournir une représentation z que l'on souhaite indépendante des paramètres du robot :

$$z = B_i^r(x_r) \tag{5.11}$$

Cette représentation z est alors concaténée avec les observations propres à la tâche $o_\mathcal{T}$ pour passer dans le module de la tâche $U_\mathcal{T}$ pour fournir une commande z_d indépendante des paramètres du robot :

$$z_d = U_\mathcal{T}(z, o_\mathcal{T}) \tag{5.12}$$

Au final, l'action z_d sera concaténée avec les informations propres au robot x_r [10] et convertie par une base de sortie propre au robot B_o^r pour fournir l'action a_r dans l'espace d'action du robot r :

$$a_r = B_o^r(x_r, z_d) \tag{5.13}$$

De manière globale, l'action sera issue de l'équation :

$$a_r = B_o^r(x_r, U_\mathcal{T}(B_i^r(x_r), o_\mathcal{T})) \tag{5.14}$$

Dans le cadre du transfert, chaque robot r dispose de ses propres bases d'entrée B_i^r et de sortie B_o^r, chaque tâche $\mathcal{T}$ est représentée par un module UNN $U_\mathcal{T}$.

Apprentissage et utilisation

Dans le cadre de l'architecture UNN, en vue de permettre le transfert de compétences relatives à différentes tâches et concernant un ensemble de robots, la phase d'apprentissage et d'utilisation, doit suivre trois étapes :

1. **Construction des bases $B_i^r(x_r)$ et $B_o^r(x_r, z)$** : en fonction du choix de l'espace de représentation robot-agnostique, discuté par la suite, la première étape est de construire ou d'apprendre les bases d'entrée $B_i^r(x_r)$ et de sortie $B_o^r(x_r, z)$, ces bases pouvant être analytiques ou sous forme d'un réseau de neurones.
2. **Apprentissage de l'UNN $U_\mathcal{T}$:** considérant un monde $\mathcal{W}_{r,\mathcal{T}} \in \mathcal{U}_s$ de l'univers source, composé d'un robot r dont nous connaissons les bases et d'une tâche $\mathcal{T}$ nous pouvons apprendre le module [11] de la tâche $U_\mathcal{T}$ en figeant les bases du robot considéré.

10. Dans le cas du LS-UNN présenté en Section 5.4.8, seule l'action z_d est utilisée.
11. que nous considérons toujours comme un réseau de neurones dans ces travaux

3. **Utilisation et transfert par agrégation des bases et de l'UNN** : à ce stade nous disposons donc d'un ensemble de robots r_i et d'un ensemble de tâches $\mathcal{T}_j$ représentées par leur module UNN $U_{\mathcal{T}_j}$. Au final, l'utilisation ($\mathcal{W}_{r_i,\mathcal{T}_j} \in \mathcal{U}_s$) ou le transfert ($\mathcal{W}_{r_i,\mathcal{T}_j} \in \mathcal{U}_t$) seront identiques puisqu'elles utiliseront les modules $B_i^r(x_{r_i})$, $B_o^r(x_{r_i}, z)$ et $U_{\mathcal{T}_j}$ comme présentés dans l'Équation (5.14).

Application à la réalisation de tâches multi-robots

L'architecture de l'UNN peut se généraliser dans le cas de tâches impliquant plusieurs robots comme indiqué sur la Figure 5.18. Théoriquement, il n'y a aucune restriction ou obligation quant à l'uniformité des robots que ce soit lors d'une phase d'apprentissage ou lors du transfert. La réalisation de tâches multi-robots peut également englober les robots marcheurs, si on considère que chaque membre est un robot indépendant [12].

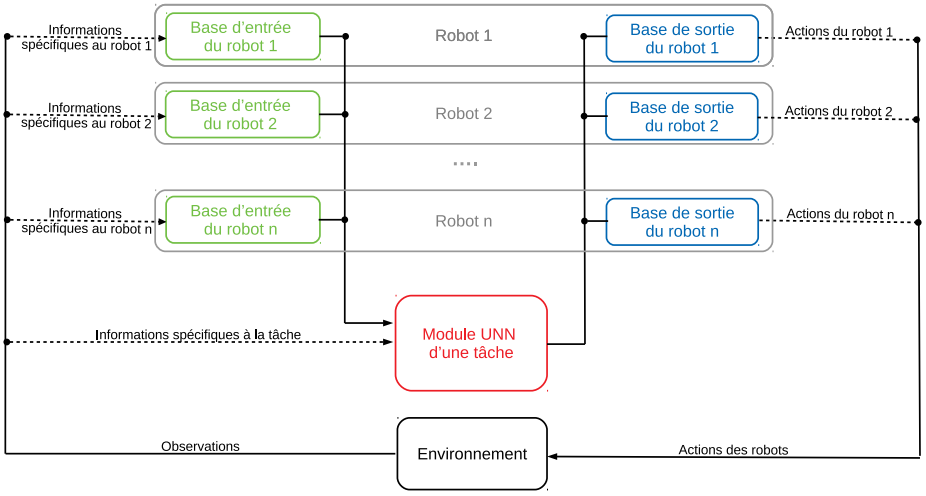


FIGURE 5.18 – Architecture de l'UNN en cas de réalisation de tâches multi-robots

Espace de représentation robot-agnostique

Le choix de l'espace de représentation robot-agnostique est important et nécessite une expertise en robotique. Afin de maximiser les performances lors du transfert, cet espace ne doit contenir aucune information propre aux caractéristiques et capacités physiques du robot. Dans le cadre de la thèse de Mehdi Mounsif [130], où nous cherchions à valider le concept du transfert via l'UNN, nous avons considéré cet espace robot-agnostique (entrée de l'UNN) comme la position et la vitesse (translation et rotation) de l'effecteur issues des modèles géométrique et cinématique directs ou d'un réseau de neurones entraîné à les reproduire. La sortie de l'UNN était considérée comme la vitesse (translation et rotation) de l'effecteur. La base de sortie $B_o^r(x_r, z)$ peut être une version analytique du modèle cinématique ou une version apprise par un réseau de neurones. Les travaux de thèse de Samuel Beaussant [8] plus orientés sur l'établissement d'autres espaces robot-agnostiques sont abordés dans le cadre du LS-UNN présenté en Section 5.4.8.

12. Dans ce cas, il y a de fortes chances que les robots soient considérés de même type, il est rare de voir des robots à pattes avec des membres différents (sauf à considérer une locomotion implicant les membres inférieurs et supérieurs pour un robot humanoïde).

BAM : le robot sans contrainte

En dépit de l'utilisation de l'UNN la tendance notable des méthodes de RL à tomber dans des minima locaux fait que l'UNN est susceptible de découvrir une stratégie efficace pour résoudre la tâche qui dépend de la configuration du robot (par exemple, bloquer un objet entre deux articulations). Bien que cela ne pose pas de problème pour l'agent actuel, cela est préjudiciable à l'efficacité d'un transfert futur vers un agent ayant une structure différente. Pour s'assurer que les contraintes de l'UNN ne sont pas enchevêtrées avec celles de l'agent, nous avons introduit le *Base Abstracted Modelling*(BAM) [130, 131, 132] présenté sur la Figure 5.19. Cette méthode consiste à considérer un robot virtuel uniquement composé de l'effecteur, ce qui revient à considérer les bases d'entrée $B_i^r(x_r)$ et de sortie $B_o^r(x_r, z)$ comme des fonctions identité.

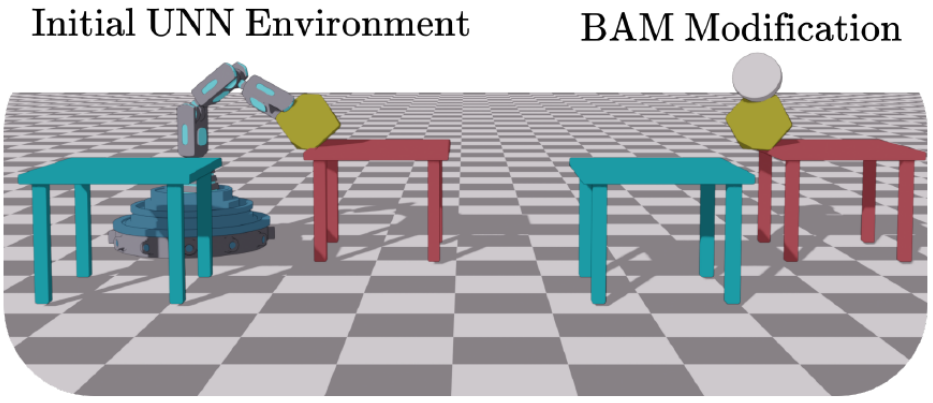


FIGURE 5.19 – Principe du BAM où on considère un robot virtuel ramené à son effecteur, source [130].

Fine-tuning

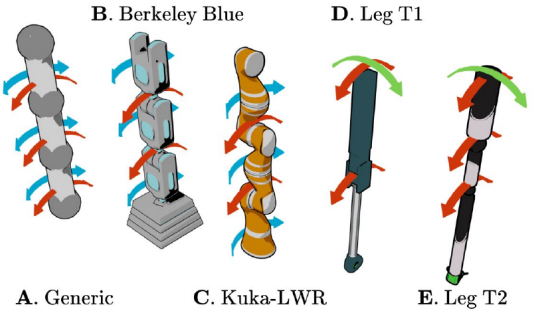
Dans le cas de transferts, comme présentés ci-après, certains résultats nécessitent une phase de *fine-tuning* qui consiste en une nouvelle phase d'apprentissage du module de la tâche $U_{\mathcal{T}_j}$ où les bases du robot $B_i^r(x_r)$ et $B_o^r(x_r, z)$ sont fixées. Cette nouvelle phase d'apprentissage du module $\hat{U}_{\mathcal{T}_j,r_i}$ ne représente qu'une portion (10 à 20 pourcents) du temps initial d'apprentissage du module $U_{\mathcal{T}_j}$ afin de conserver l'intérêt du transfert.

Résultats

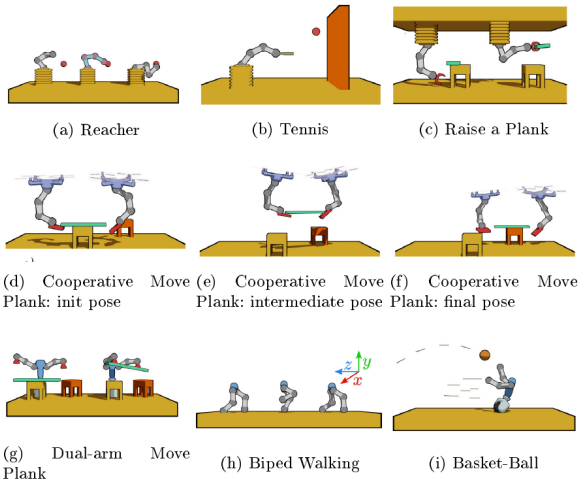
Les principaux résultats de l'approche UNN sont présentés dans la thèse de Mehdi Mounsif [130], il est également possible de voir quelques résultats dans cette vidéo[13]. La méthode a été évaluée sur les robots présentés en Figure 5.20a et en considérant les tâches présentées en Figure 5.20b. Dans cette partie, nous allons brièvement présenter quelques résultats de transfert et discuter de l'avantage de l'utilisation de l'UNN pour une application n'ayant pas obligatoirement vocation à être transférée.

Transfert Considérons la tâche de tennis illustrée sur la Figure 5.21 où un robot doit, en simulation, effectuer le plus de rebonds possible sur un mur. La récompense est

13. disponible sur `https://www.youtube.com/watch?v=bhxOSiZjANo`



(a) Les robots



(b) Les tâches

FIGURE 5.20 – Ensemble des robots et tâches utilisés pour valider la méthode UNN, source [133].

le nombre de rebonds sur le mur jusqu'à la durée maximale de la simulation ou jusqu'à ce que la balle tombe au sol.

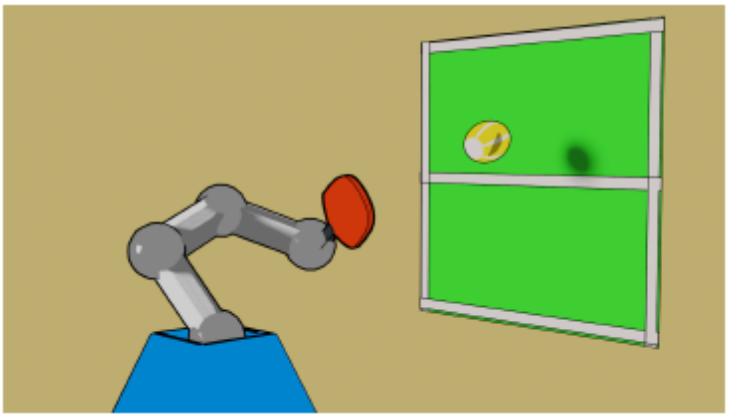


FIGURE 5.21 – Illustration de la tâche de tennis, source [130].

En regardant les courbes d'apprentissage de la Figure 5.22, on s'aperçoit que l'approche UNN (en bleu foncé) démarre avec une récompense supérieure au PPO, de plus l'UNN apprend de manière beaucoup plus efficace que le PPO ([159]) classique (en rouge) avec une récompense de 8 contre 1. De plus, après transfert sur le robot Blue(en bleu clair), on voit que la récompense est partiellement conservée (environ 2.5) et qu'après 10% du temps d'entrainement (*fine-tuning*) les performances semblent satisfaisantes (légèrement supérieur à 8 pour le robot source et légèrement inférieur à 8 pour le robot Blue après *fine-tuning*), alors que le transfert PPO (en orange) ne profite que très légèrement de l'expérience acquise par le PPO.

Si on se concentre sur les performances présentées dans le Tableau 5.1, on constate également que l'apprentissage initial via l'UNN surpasse le PPO classique et que le transfert est également de meilleure qualité que l'on considère ou non une étape de *fine tuning*[14].

14. Les pourcentages de *fine tuning* sont donnés par rapport aux nombres d'itérations du processus

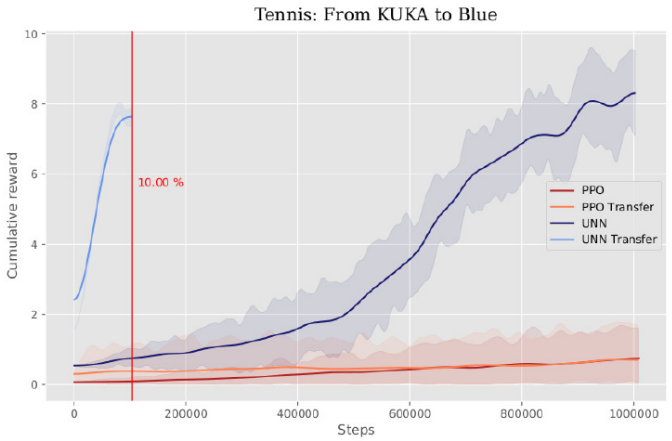


FIGURE 5.22 – Courbes d'apprentissage et de transfert pour la tâche de tennis, du robot KUKA au robot Blue, source [130].

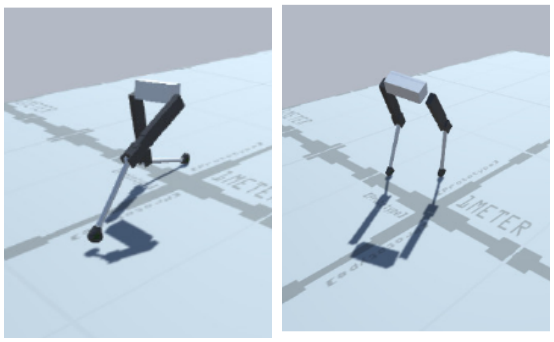
		Configuration	**Type d'entrainement**	**Performances** (in bounces)
	Original	Kuka	initial	18.74
	Transfer to	Berkeley Blue	transfert direct	11.88
UNN	Transfer to	Berkeley Blue	10% *fine tuning*	17.56
	Original	Kuka	initial	7.89
	Transfer to	Berkeley Blue	transfert direct	0.86
PPO	Transfer to	Berkeley Blue	100% *fine tuning*	6.81

TABLE 5.1 – Comparaison des performances sur la tâche de tennis.

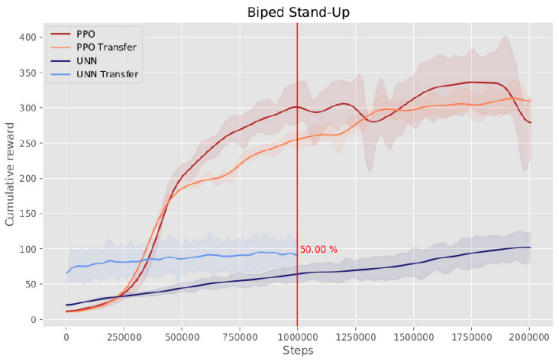
Biais de comportement induit par l'UNN Comme démontré précédemment, l'utilisation de l'UNN permet un apprentissage plus efficace dans la plupart des cas. Cet avantage vient du fait que les compétences propres au contrôle des parties du robot sont déjà incluses dans les bases d'entrée et de sortie, ce savoir ne devant pas être appris *from scratch*. Dans le cadre d'un mouvement de marche (que nous avons considéré comme une tâche multi-robots) nous avons pu observer que cette capacité permettait d'avoir des mouvements plus "naturels" qu'une méthode classique, comme montré sur la Figure 5.23a alors que la récompense finale était inférieure comme indiqué sur la Figure 5.23b.

Limitations Dans la thèse de Mehdi Mounsif, nous avons démontré en simulation que le transfert était possible en considérant un ensemble de robots et de tâches présentés dans la Figure 5.20. La première limitation est relative à la gestion des retards ignorés en simulation mais qui constituent un point bloquant pour le passage à des robots réels. La seconde limitation concerne le choix qui a été fait de considérer la position et orientation de l'effecteur comme l'espace robot-agnostique. Même si ce choix paraît naturel, il convient de s'interroger s'il est le plus pertinent. Ces deux limiations ont été abordées dans le cadre de la thèse de Samuel Beaussant et sont présentées dans les Sections 5.4.7 et 5.4.8.

d'apprentissage initial (dans ce cas 1 million).



(a) Représentation de la marche du robot bipède PPO à gauche, UNN à droite.



(b) Courbe d'apprentissage sur une tâche de marche pour un robot bipède.

FIGURE 5.23 – Biais de comportement induit par l'UNN, source [130].

5.4.7 Le DA-UNN

L'un des principaux objectifs de l'utilisation de l'IA en robotique est de permettre à des robots réels de réaliser des tâches que les méthodes de l'état de l'art ne permettent pas de réaliser. L'apprentissage et la validation en simulation n'est qu'une étape priliminaire à un déploiement sur un robot réel. Le passage de la simulation à la réalité est connu comme le problème du *sim2real* [140, 144] brièvement présenté dans la Section 5.4.2. Dans le cadre de la thèse de Samuel Beaussant, nous nous sommes intéressés au problème de la gestion du temps de retard généralement ignoré.

Définition du retard

Nous définissons le retard de manière uniforme, qu'il soit dû au traitement des informations de commande ou de perception, comme le temps nécessaire pour qu'une action ait un impact significatif sur les observations. Même si la méthodologie présentée par la suite se veut généralisable, dans le cadre de ces travaux nous supposons le retard comme connu (identifiable) et constant pendant la durée de la manipulation. Dans le cadre d'un apprentissage par RL on peut adapter la Figure 5.2 pour y inclure le retard comme le montre la Figure 5.24.

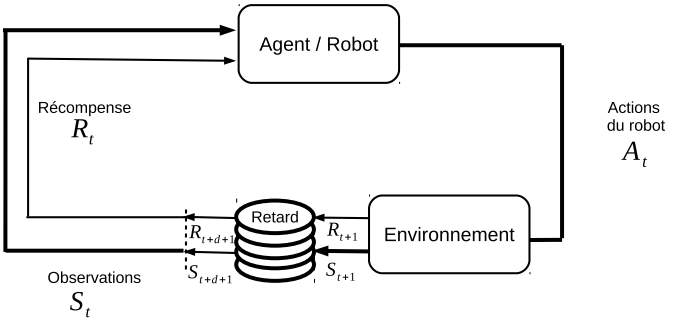


FIGURE 5.24 – Illustration du processus d'apprentissage dans le cas d'un système à retard.

En supposant un retard $t_d = d.T_e$ multiple de la période d'échantillonage T_e on peut représenter le retard comme une pile FIFO (First In First Out) entre l'environnement et

les observations, d étant la taille de la pile.

Architecture du DA-UNN

Afin de permettre un transfert de compétences entre des robots ayant des retards différents, nous avons proposé l'architecture présentée sur la Figure 5.25. Nous abordons le problème du retard en augmentant l'espace d'entrée du module UNN avec le retard estimé du système. Ce retard sera constant lors de la phase d'utilisation et relié au robot utilisé. Lors de la phase d'apprentissage, nous reprenons le principe du *Domain randomization* [24, 144] en échantillonnant le retard à partir d'une distribution uniforme discrète sous la forme de $U(d_{min}, d_{max})$ où d_{min} et d_{max} sont respectivement le retard minimum et le retard maximum considérés pour les environnements aléatoires lors des différentes interactions avec l'environnement.

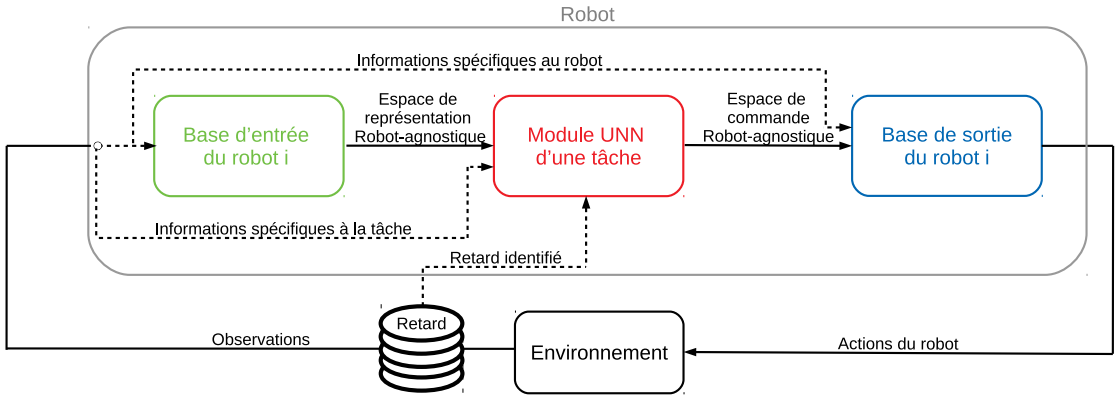


FIGURE 5.25 – Architecture du DA-UNN.

Application à un système physique réel

Pour évaluer l'architecture du DA-UNN nous considérons la tâche d'asservissement en position d'une bille sur un rail comme montré sur la Figure 5.26. La phase d'apprentissage a été réalisée en simulation en considérant le BAM, validée en simulation puis comparée sur le robot réel à une méthode *vanilla* comme montré sur la Figure 5.27.

Les courbes issues de la simulation avec le BAM (Figure 5.28a) comparent les résultats obtenus avec la version DA-UNN (*BAM delay aware*), la version du BAM entraînée avec le retard correspondant (*BAM optimal*) et la version du BAM entraînée sans prise en compte du retard (*BAM delay unaware*). Ces courbes montrent que malgré un retard important (de 0.3 à 0.7s), le module DA-UNN arrive à asservir la bille avec une bonne précision et un comportement dynamique satisfaisant, alors que la version "*unaware*" est clairement instable même avec un faible retard.

Les résultats du *sim2real* vers le robot Braccio sont présentés sur la Figure 5.28b entre la version *delay aware* et *delay unaware*. Comme en simulation, on constate que la version qui ne prend pas en compte le retard (*delay unaware*) ne permet pas d'avoir un comportement stable alors que la version DA-UNN permet une meilleure réalisation de la tâche, même si on peut noter quelques oscillations [15].

15. Les vidéos de ces expérimentations sont disponibles sur `https://www.youtube.com/watch?v=dQd4jfnWR8g`.

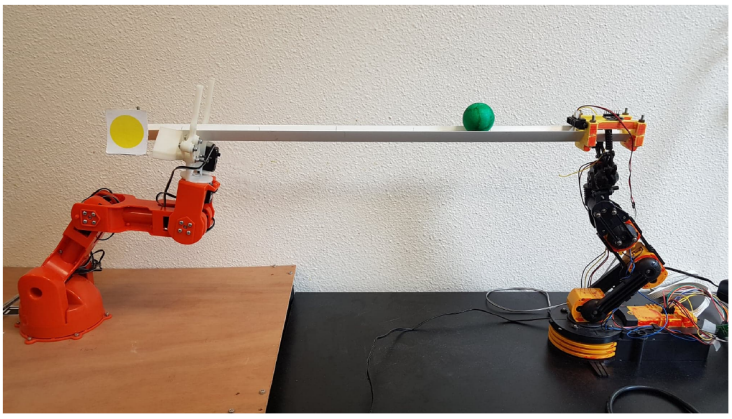


FIGURE 5.26 – Représentation de la tâche d'asservissement en position d'une bille sur un rail. Le robot de gauche sera commandé afin de monter ou descendre le rail. Ici le robot de droite ne sert que de support (il ne participe pas à la réalisation de la tâche).

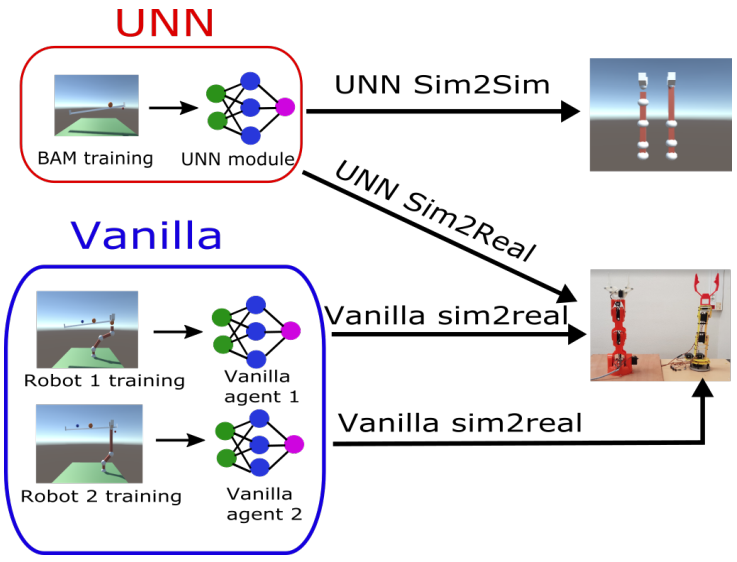


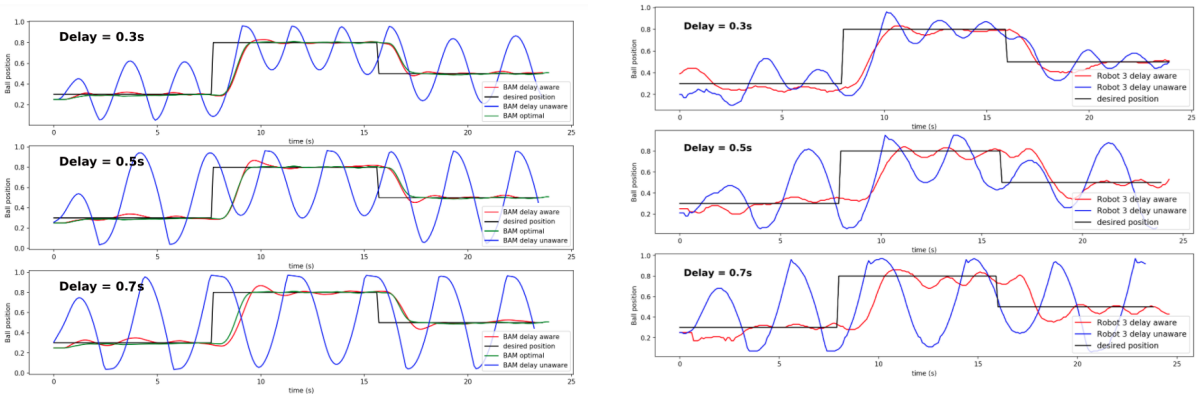
FIGURE 5.27 – Protocole d'apprentissage et d'évaluation du DA-UNNpour la tâche d'asservissement en position d'une bille sur un rail (source [130]).

La gestion du retard est un point primordial pour le transfert de compétences entre robots et dans le cadre du *sim2real*. Nous avons proposé une extension à la méthode UNN afin de prendre en compte les retards. La méthode DA-UNN a permis de faire un transfert de compétences entre robots sur une tâche relativement dynamique[16]. Ces premiers résultats semblent encourageants et démontrent la possibilité du transfert. Dans la Section 5.4.8, nous présentons une nouvelle extension à l'UNN en nous focalisant sur la définition d'un espace robot-agnostique où la tâche est réalisée en mettant de côté le retard.

5.4.8 Le *Latent Space Universal Notice Network* (LS-UNN)

Dans la Section 5.4.6, nous avons présenté la structure de l'UNN et nous avons démontré son efficacité dans le cadre du transfert de compétences. Nous avons souligné

16. L'état de la tâche n'est pas lié uniquement à la pose du robot mais également aux actions précédentes.



(a) Simulation avec le BAM

(b) Expérimentation réelle avec le robot Braccio.

FIGURE 5.28 – Résultat de simulation et d'expérimentation pour la tâche d'asservissement en position d'une bille sur un rail (source [130]).

l'importance du choix de l'espace robot-agnostique qui nécessite une compréhension de la tâche et qui va impacter sur sa réalisation. Dans cette partie, nous proposons le *Latent Space Universal Notice Network* (LS-UNN) qui a pour but d'apprendre un espace robot-agnostique dans lequel le module UNN pourrait résoudre la tâche. Plus de détails sont disponibles dans [9] et en Annexe B. Le LS-UNN a tiré son inspiration dans le TSL présenté en Section 5.4.5. Le schéma global du LS-UNN est présenté sur la Figure 5.29 [17].

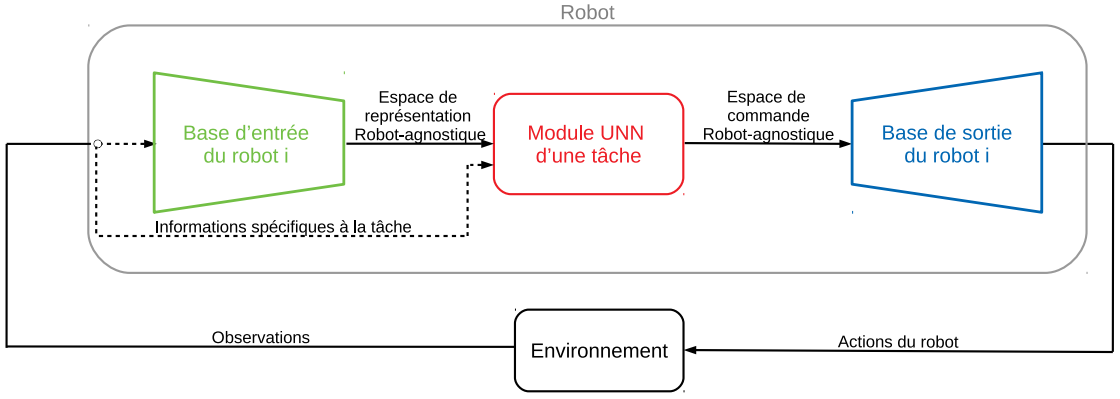


FIGURE 5.29 – Architecture du LS-UNN

Le principe du LS-UNN est d'amener des états similaires de robots différents vers un même état défini dans un espace latent. La résolution de la tâche se réalise alors dans cet espace. L'action (ou l'état désiré) sera alors ramené dans l'espace d'action du robot. Dans un espace débarrassé de toutes les particularités propres à chaque robot, la tâche peut alors être résolue indépendamment du robot. La mise en place du LS-UNN, comme définie dans la thèse de Samuel Beaussant et représentée sur la Figure 5.30 nécessite 4 étapes [8] :

— la collecte et l'appairement d'états similaires entre les robots,

17. Cette figure est semblable à la Figure 5.17, les modules d'entrée et de sortie sont représentés différemment pour indiquer la compression des données via un VAE. On notera également l'absence des informations intrinsèques au robot dans la base de sortie qui ne prend pas de sens dans le cas d'un VAE.

— la construction de l'espace latent via l'entrainement des bases d'entrée et de sortie,
— la création du module de la tâche en utilisant un robot source,
— Le transfert de la tâche vers un robot *target*.

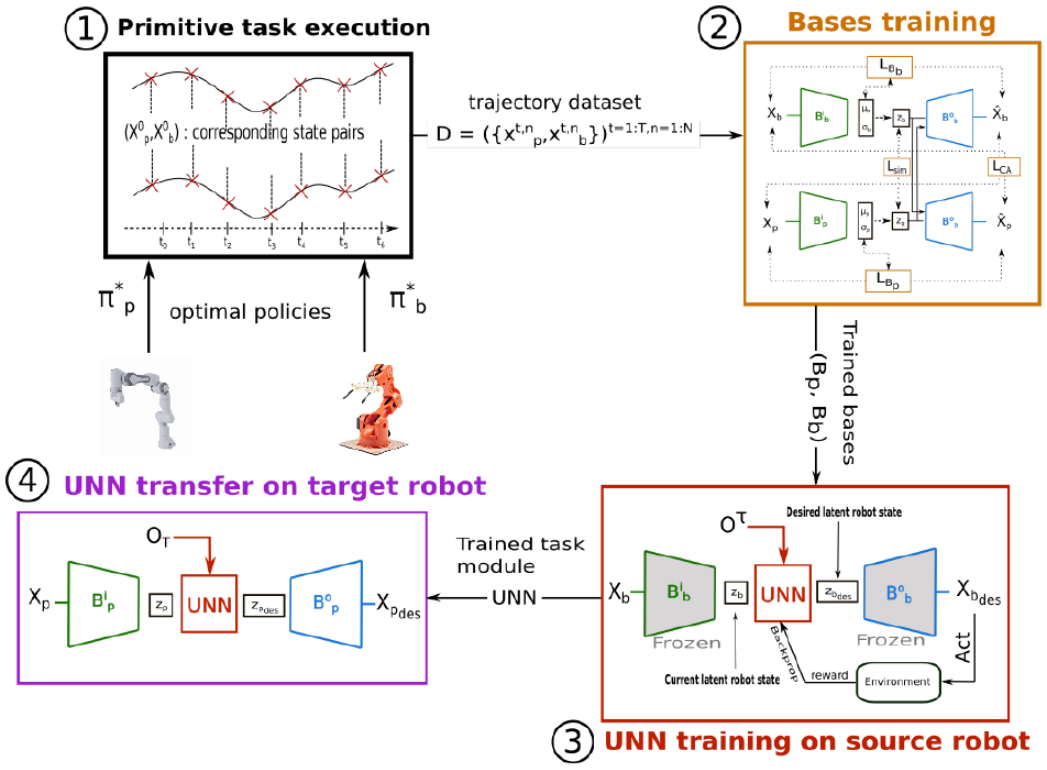


FIGURE 5.30 – Les 4 étapes de la mise en place du LS-UNN, source [8].

Appairement d'états similaires

Cette étape, même si elle paraît simple, conditionne grandement le résultat du transfert final. L'objectif est de disposer d'états similaires sur des robots ayant des morphologies et possiblement des capacités différentes. Nous avons considéré que si deux robots réalisent une même tâche, dite *primitive*, de manière optimale dans le même laps de temps alors les états de ces robots à chaque instant sont similaires. Si on définit $x^{r_1,\mathcal{T}}(t)$ et $x^{r_2,\mathcal{T}}(t)$ la trajectoire (l'ensemble des états) des robots r_1 et r_2 pour accomplir la tâche $\mathcal{T}$, alors on peut définir une relation de similarité "$\approx$" telle que :

$$x^{r_1,\mathcal{T}}(t) \approx x^{r_2,\mathcal{T}}(t) \Leftrightarrow t_1 = t_2 \tag{5.15}$$

Il existe d'autres définitions de similarité qui pourront être évaluées lors de futurs travaux, comme le *Dynamic Time Wrapping (DTW)* [135] ou le *Soft*-DTW [32] qui n'imposent pas une durée identique de résolution de la tâche. Dans notre cas, nous avons considéré une tâche *primitive* de *reaching* où les robots devaient toucher avec leur effecteur une cible bougeant de manière aléatoire comme montré sur la Figure 5.31. Au final nous obtenons une base de données D de 100 000 états appairés.

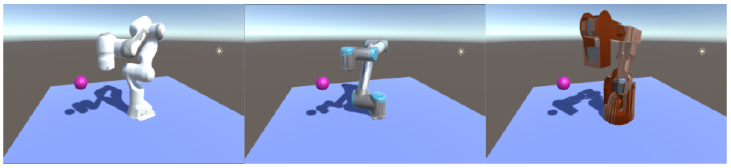


FIGURE 5.31 – Représentation de la tâche *primitive* de *reaching*. Le robot doit positionner son effecteur sur la cible violette, source [8].

Construction de l'espace latent

Dans cette partie nous présentons comment créer l'espace latent via l'apprentissage des bases d'entrée B_i^r et de sortie B_o^r en partant de la base de données D obtenue précédemment. Le principe de base s'inspire des *Variational Auto Encoder* (VAE) introduit dans la Section 5.4.5 en y ajoutant la notion de similarité. L'objectif est de trouver des bases d'entrée et de sortie pour les 2 robots considérés afin de prendre en compte trois aspects, comme montré sur la Figure 5.32.

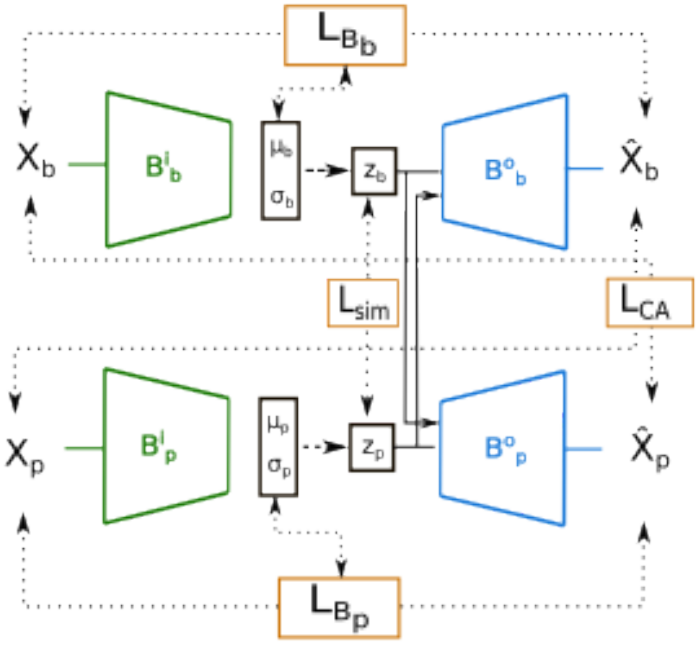


FIGURE 5.32 – Représentation des différents critères à prendre en compte lors de l'apprentissage des bases pour le LS-UNN, considérant un robot r_b et un robot r_p, source [8].

Au final l'apprentissage consistera à minimiser le critère suivant :

$$\min_{\theta_{r_1},\theta_{r_2},\phi_{r_2},\phi_{r_1}} \sum_{(x_{r_1},x_{r_2})\in D} L_{B_{r_1}} + L_{B_{r_2}} + \delta L_{sim} + \lambda L_{CA} \tag{5.16}$$

Avec θ_{r_i}, ϕ_{r_i} les poids des bases d'entrée $B_i^{r_i}$ et de sortie $B_o^{r_i}$, δ et λ des coefficients de pondération et :

— $L_{B_{r_i}}$: le critère de reconstruction[18] de l'état du robot i en encourageant, via le coefficient β la bonne régularisation de l'espace latent via la *KL divergence*[99], qui

18. différence entre X_i et $\hat{X}_i$ comme montré sur la Figure 5.32.

permet en général d'obtenir une bonne généralisation comme présenté dans [69] :

$$L_{B_{r_i}} = -\mathbb{E}_{z\sim B^i_{\phi_{r_i}}(\cdot|x_{r_i})}\left[\log B^o_{\theta_{r_i}}(x_{r_i}|z)\right] + \beta D_{KL}\left(B^i_{\phi_{r_i}}(z|x_{r_i})||\mathcal{N}(0,I)\right) \quad (5.17)$$

— L_{sim} : le critère de similarité qui définit que si deux états similaires $x_{r_1} \approx x_{r_2}$ sont projetés dans l'espace latent, alors leur projection $z_{r_i} \sim B^i_{\phi_{r_i}}(\cdot|x_{r_i})$ doit être proche :

$$L_{sim} = \mathbb{E}_{z_{r_1}\sim B^i_{\phi_{r_1}}, z_{r_2}\sim B^i_{\phi_{r_2}}}\left[||z_{r_1} - z_{r_2}||^2\right] \quad (5.18)$$

— et L_{CA} : le critère croisé inspiré de [158] afin d'améliorer le transfert qui compare l'état x_{r_1} à son état similaire x_{r_2} encodé par la base d'entrée du robot r_2 décodé par la base de sortie du robot r_1 (et vice versa) :

$$L_{CA} = \mathbb{E}_{z_{r_1}\sim B^i_{\phi_{r_1}}}\left[||B^o_{\theta_{r_2}}(z_{r_1}) - x_{r_2}||^2\right] + \mathbb{E}_{z_{r_2}\sim B^i_{\phi_{r_2}}}\left[||B^o_{\theta_{r_1}}(z_{r_2}) - x_{r_1}||^2\right] \quad (5.19)$$

La construction de l'espace latent se fait donc en utilisant deux robots. L'ajout d'un n-ième robot r_n se fera en considérant le même processus en utilisant un robot r_m ($m < n$) disposant déjà de ses bases $B^i_{r_m}$,$B^o_{r_m}$, mais en figeant les coefficients θ_{r_m} et ϕ_{r_m}. [19].

Apprentissage et transfert du module UNN

De manière complètement similaire à l'UNN présenté dans la Section 5.4.6, nous pouvons procéder à l'apprentissage d'un module $U_\mathcal{T}$ qui sera constitué d'un réseau de neurones paramétré par les poids $\psi_\mathcal{T}$, en gelant les coefficients θ_{r_i}, ϕ_{r_i} du robot r_i sur lequel est apprise la tâche.

Finalement le transfert peut se faire, idéalement, en mode *plug and play* en considérant les modules $B^i_{r_j}$ et $B^o_{r_j}$ du robot j et le module $U_\mathcal{T}$ lié à la tâche $\mathcal{T}$ apprise sur le robot i. Il se peut également que les performances après transfert nécessitent une phase de *fine tuning*.

Protocole d'évaluation

Lors de la thèse de Samuel Beaussant, nous avons validé cette méthode de transfert sur plusieurs tâches et en considérant plusieurs robots comme montré sur la Figure 5.33b [20] :

— tâches :

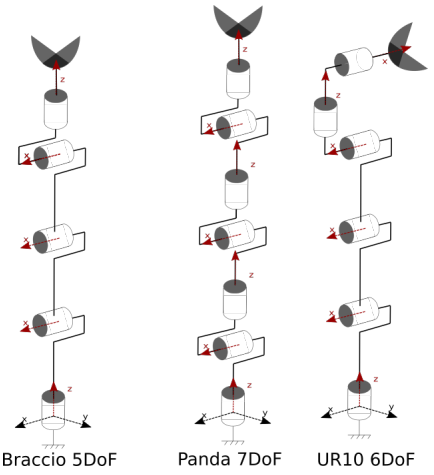
  — *pick and place* consiste à attraper un objet et à le déplacer vers une position définie. Cette tâche permet de valider le transfert de l'ordonnancement (saisie, déplacement, pose).

  — *peg insertion* consiste à remplacer l'effecteur du robot par un bâton (diamètre de 1.9cm) et à l'insérer dans un trou carré (dimension intérieure 2.25x2.25 cm). Cette tâche est particulièrement difficile dans le contexte du transfert entre robots car elle nécessite une politique de contrôle précise. Elle mettra en évidence la capacité de l'UNN à faire fonctionner avec précision les robots sur lesquels il est transféré, même lorsque leur structure cinématique varie de manière significative.

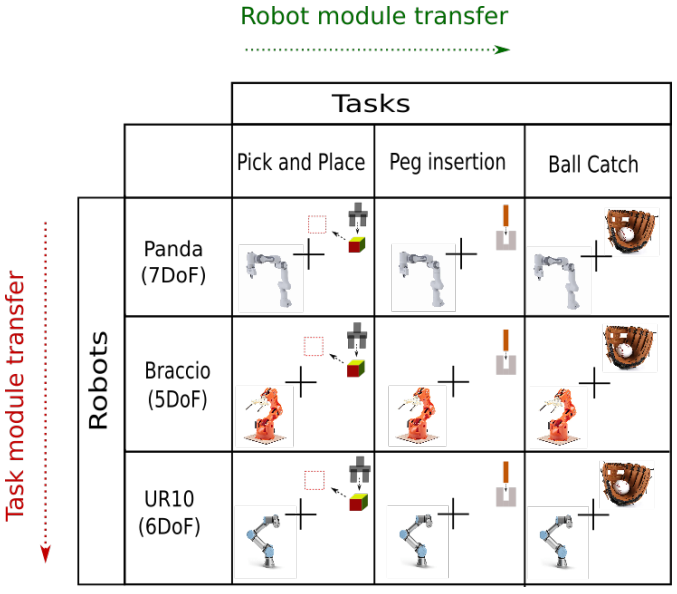
19. Dans ce cas, il paraît préférable de considérer les robots r_1 ou r_2 comme référence.
20. Le détail des tâches et des robots est disponible dans l'Annexe B

— *ball catcher* consiste à attraper une balle au vol. Dans ce cas, l'effecteur est remplacé par un panier afin de réceptionner la balle. La balle est envoyée de la même position mais avec des trajectoires différentes (qui varient lors de l'apprentissage). Cette tâche est intéressante pour démontrer la capacité du LS-UNN à gérer des tâches dynamiques.

— robots : Pour mettre en évidence les avantages de notre méthode de transfert entre robots, nous considérons des robots présentant des caractéristiques morphologiques distinctes, telles que le nombre d'articulations et la longueur des liens. Plus précisément, notre groupe de robots est composé d'un robot Panda avec 7 DoF, d'un UR10 avec 6 DoF et d'un robot Braccio avec 5 DoF (voir Figure 5.33a). Pour les simulations, nous normalisons la longueur totale de manière à ce que tous les robots considérés aient à peu près les mêmes espaces atteignables pour des raisons pratiques. Pour les expériences réalisées avec les robots physiques, nous incluons la différence d'échelle dans le cadre des tâches et des environnements. Tous les robots sont contrôlés en vitesse.



(a) Morphologies des robots considérés pour évaluer les performances du LS-UNN.



(b) Représentation des robots et tâches considérés pour évaluer les performances du LS-UNN.

FIGURE 5.33 – Protocole d'évaluation du LS-UNN, sources [8].

Les trois tâches ont été évaluées en simulation, la tâche de *peg insertion* a également été validée sur des robots réels comme le montre la Figure 5.34.

Résultats et perspectives

Dans cette partie, nous détaillons les résultats concernant la tâche de *peg insertion*, les résultats pour la tâche de *pick and place* et *ball catcher* sont disponibles dans [8] ainsi que [9] également disponible dans l'Annexe B.

Les Tableaux 5.2 et 5.3 présentent les résultats des transferts. Nous avons évalué les performances en considérant différents robots source et robots *target*. Les termes sur la diagonales indiquent le résultat de l'apprentissage. Les autres cases indiquent le résulat d'un transfert direct (zero-shot) puis en gras après un *fine-tuning* correspondant à environ

(a) UR10

(b) Panda

(c) Illustration du jeu disponible pour cette tâche.

FIGURE 5.34 – Validation de la tâche de *peg insertion* sur les robots réels. La position du trou est obtenue en début d'expérience par un système de motion capture, sources [8].

25% de l'apprentissage initial. La dernière colonne est le résultat d'un apprentissage en utilisant un PPO classique.

Le Tableau 5.2 démontre qu'un transfert direct est possible avec une faible perte de performances (82% dans le pire des cas). On peut également voir qu'une phase de *fine-tuning* permet de recouvrer toutes les performances initiales au prix d'une portion du temps d'apprentissage initial.

source*target*	**Braccio**	**Panda**	**UR10**	**ref**
Braccio	100	95 / **100**	92 / **100**	100
Panda	95 / **100**	100	82 / **100**	99
UR10	99 / **100**	90 / **100**	100	99

TABLE 5.2 – Performances de transfert pour la tâche de *peg insertion* en simulation. Les résultats sont donnés en pourcentage de succès pour 1000 essais.

Le Tableau 5.3 montre les résultats de transfert entre les robots simulés et les robots réels. Ces résultats couvrent donc la validation du *sim2real* d'un robot en simulation à son alterego réel. Ils couvrent également le transfert d'un robot simulé vers un robot réel différent [21]. Le calcul des performances a été réalisé sur 28 positions différentes du trou. Les étapes de *fine tuning* ont également été réalisées en simulation.

Les résultats témoignent de la qualité de la modélisation des robots réels et de la faible différence entre les robots simulés et réels. On peut voir que l'approche LS-UNN permet un apprentissage comparable aux méthodes de référence (93% contre 89% pour le Panda et 96% contre 100% pour l'UR10). Dans le cas du transfert, les résultats montrent une

21. Les validations sur le robot Braccio n'ont pas été réalisées en expérimentation réelle, les jeux mécaniques de ce robot *low cost* étant importants, un contrôle en boucle fermée avec mesure exacte de la position aurait été nécessaire. Ces phénomènes pourront être abordés lors de travaux futurs.

source (simulé) \ *Target* (réel)	**Panda (réel)**	**UR10 (reél)**	**ref**
Braccio	79 / **93**	75 / **93**	
Panda	93	79 / **96**	89
UR10	71 / **85**	96	100

TABLE 5.3 – Performances de transfert pour la tâche de *peg insertion* en réel. Les résultats sont donnés en pourcentage de succès pour 28 essais.

perte de performances plus importante que pour les résultats en simulation [22], même si les performances restent satisfaisantes (plus de 70% de réussite). La phase de *fine tuning* permet une amélioration des performances (supérieures à 85%).

Ces résultats montrent qu'un transfert vers des robots réels est possible pour des tâches telles que le *peg insertion* qui sont peu dynamiques mais qui nécessitent de la précision.

5.5 Transfert : résultats et perspectives

Les différentes méthodes de transfert présentées dans le cadre de mes activités de recherche (durant les thèses de Medhi Mounsif [130] et de Samuel Beaussant [8]) ont été validées en considérant des tâches dynamiques, d'ordonnancement, de précision en simulation et/ou lors d'expérimentation réelle. Un récapitulatif est disponible dans le Tableau 5.4.

validé en	méthode	ordonnancement	précision	dynamique
simulation	*Coach GAN*			tennis(2D)
	TSL		tracé de cercle (2D)	tennis(2D)
	UNN	*pick and place*, co-manipulation	*reaching*	tennis, marche bipède, *ball catcher*
	DA-UNN			bille sur rail
	LS-UNN	*pick and place*	*peg insertion*	*ball catcher*
réel	DA-UNN			bille sur rail
	LS-UNN		*peg insertion*	

TABLE 5.4 – Tableau récapitulatif des tâches dont le transfert a été validé en simulation ou en expérimentation réelle en utilisant les différentes méthodes.

Les résultats montrent que le transfert de compétences issu d'apprentissage est possible entre des robots morphologiquement différents. L'UNN et ses dérivées laissent même présager qu'un transfert direct de type "*plug and play*" est possible contrairement aux méthodes indirectes comme le TSL et *CoachGAN* qui nécessitent une étape d'apprentissage durant le processus de transfert. D'après les travaux présentés, quelques axes d'évaluation et d'amélioration émergent :

— La mise en place d'un protocole d'évaluation des méthodes de transfert afin de pouvoir les comparer entre elles, alors que pour l'instant nous avons comparé les performances entre des architectures d'apprentissage classique et nos méthodes de transfert.

22. Il faut prendre en compte le faible nombre d'essais sur le robot réel (28) par rapport aux transferts simulés (1000 essais). Un essai en réel représente 3.5% des résultats.

— La validation du transfert sur des tâches de manipulation d'objets déformables comme celles menées dans le cadre de la thèse de Mélodie Daniel et présentées en Section 5.3.

— La fusion des méthodes DA-UNN et LS-UNN afin de pouvoir bénéficier des capacités de l'espace latent et de la gestion des retards.

— L'étude de l'impact d'autres méthodes d'appairage des trajectoires sur le transfert ou la construction d'un espace latent en prenant en compte plus de deux robots.

— La gestion des contraintes pour la prise en compte de capacités physiques différentes. Actuellement, nous avons supposé des capacités physiques semblables, mais comment se comporterait l'UNN si deux robots pouvaient réaliser la même tâche mais de manière différente ?

— La mise en place d'une boucle fermée afin de prendre en compte des perturbations arrivant sur le robot qui ont été légèrement traitées avec la méthode du DA-UNN.

Mon projet de recherche présenté dans la Section 6.2 reprend plus en détail certains de ces points et présente également d'autres perspectives à plus long terme.

5.6 Complexité et temps de calcul vs. garantie

Les méthodes par apprentissage n'ont plus à prouver leur capacité à gérer des tâches complexes, parfois hors de portée des algorithmes de l'état de l'art. Cependant leur temps de calcul, la mise en place des apprentissages et le transfert (*sim2real*) vers une application réelle ne permettent pas une mise en place massive de ces méthodes dans des algorithmes de planification ou de contrôle en robotique. Dans le cadre du transfert, il est important de dissocier les temps de calculs liés à la phase d'apprentissage et ceux liés au transfert et à l'utilisation. Dans le cas des approches de type "*plug and play*", comme celles basées sur l'UNN, le gain paraît significatif dès le transfert d'une première tâche entre deux robots.

Même si certains travaux se focalisent sur cet aspect [72, 180, 183], le manque de garantie sur le bon déroulement de l'action risque de nuire à l'intégrité du robot et de son environnement (où peut être l'humain) ce qui est un frein à une utilisation directe et massive de ces approches [23]. Cependant, les méthodes d'IA et notamment les méthodes de transfert peuvent fournir une première solution de mouvement (sur laquelle on ne peut pas apporter de garantie) qui servira d'initialisation à des méthodes de planification ou de contrôle plus robustes et plus sûres.

23. Contrairement au déploiement de l'IA dans des domaines moins risqués, ou avec des risques moins directs, comme le traitement d'image ou du langage.

Chapitre 6

Conclusion et projet de recherche

6.1 Conclusion

Ce manuscrit illustre les activités de recherche que j'ai menées depuis 2006 autour de la génération sûre et rapide de mouvements robotiques complexes.

Dans un premier temps, je me suis intéressé à la formulation générale du problème de la génération de mouvements dynamiques et multi-contacts afin qu'il puisse être résolu par des algorithmes d'optimisation. Cette formulation permet de garantir, sous des hypothèses de bonne modélisation, l'intégrité physique du robot et le bon accomplissement de la tâche demandée. Ces travaux ont été validés par de nombreuses expérimentations réelles sur des robots humanoïdes (HRP-2, HOAP-3) et des avatars virtuels. Le temps de calcul étant conséquent, les mouvements ont été générés hors ligne, mais ils ont pu être exécutés en boucle ouverte avec succès, avec la possibilité d'utiliser un stabilisateur pour compenser les flexibilités non modélisées.

Dans ce cadre, l'utilisation de l'Analyse par Intervalles (AI) a permis de garantir la validité du mouvement sur toute sa durée. Le principal inconvénient de l'AI demeure le temps de calcul, qui est principalement lié à la notion de pessimisme qui impose dans la plupart des cas une utilisation hors ligne. J'ai proposé une méthode de réduction du pessimisme, et du temps de calcul via une approximation polynomiale et des fonctions de base. Bien qu'ayant amené des contributions à la fois sur la génération de mouvement et sur la réduction du pessimisme lié à l'AI, ces méthodes ne peuvent pas être utilisées en ligne pour la réalisation de tâches robotiques.

Les méthodes d'Intelligence Artificielle(IA) ont démontré de bonnes capacités à réaliser des tâches hors de portée des méthodes de l'état de l'art, généralement au prix de phases d'apprentissage considérablement longues. Depuis 2017, mes activités de recherche s'orientent vers la capitalisation des compétences acquises lors des phases d'apprentissage pour un transfert de ces compétences vers d'autres robots potentiellement morphologiquement différents. Les résultats préliminaires obtenus témoignent qu'un transfert de type "*plug and play*" est possible.

En reprenant la Figure 6.1, on peut résumer mes contributions sur les différents domaines à :

- formulation du problème et ajout de garantie (sous hypothèses de bonne modélisation) pour la génération de mouvements dynamiques complexes,
- méthode d'évaluation diminuant le pessimisme ce qui permet de diminuer les temps de calcul des méthodes d'AI ou de considérer des problèmes plus complexes,

— diminution et rationnalisation des temps d'apprentissage des méthodes d'IA (que j'ai dissocié dans cette Figure du temps d'exécution par rapport à la Figure 2.1, n'ayant pas apporté de contribution sur ce point).

Toutes mes contributions ont donc pour objectif de nous rapprocher de l'algorithme de génération de mouvements idéal alliant à la fois un temps de calcul permettant une implémentation en temps réel, la garantie d'optimalité et de bon fonctionnement et la complexité de la tâche à accomplir.

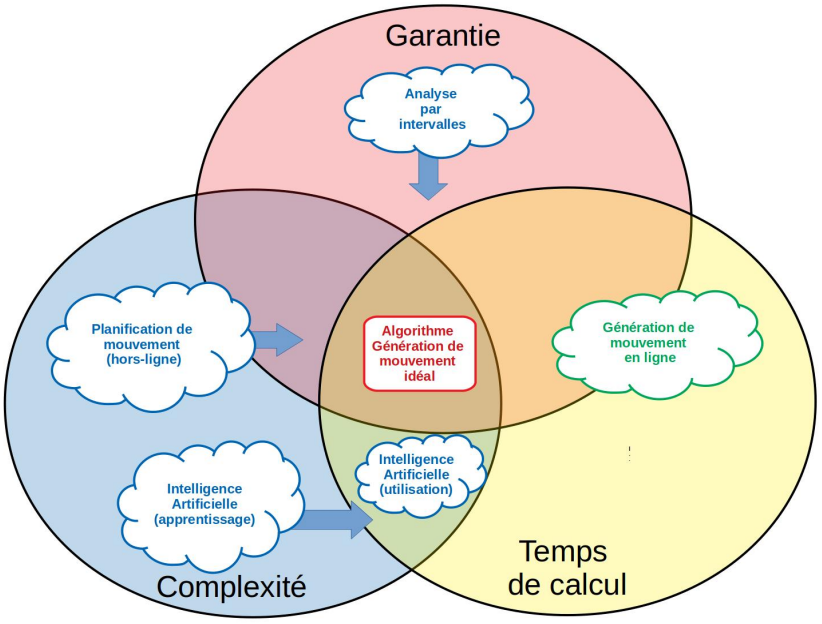


FIGURE 6.1 – Représentation des contributions vers un algorithme de génération de mouvements idéal.

6.2 Projet de recherche

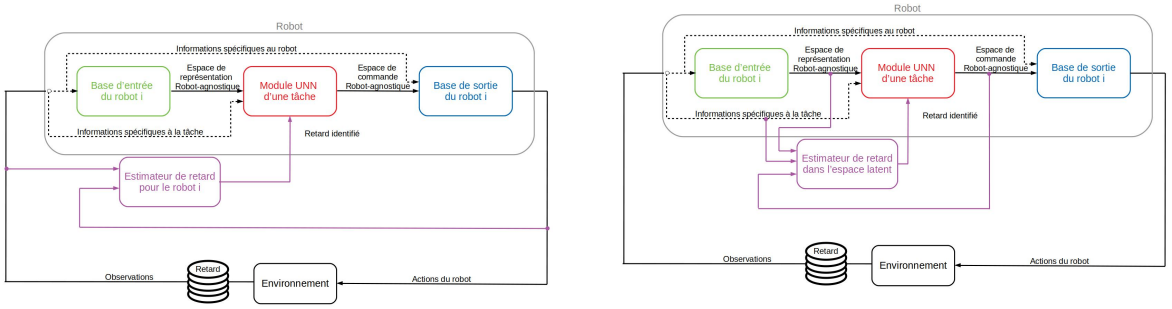
Mon projet de recherche s'inscrit naturellement dans la continuité de mes travaux précédents vers l'algorithme de génération de mouvement idéal. Pour cela trois principaux axes de recherche seront explorés.

6.2.1 Développement des méthodes de transfert pour la génération de mouvements

Les premiers résultats de transfert présentés dans le Chapitre 5 semblent prometteurs en simulation et en situation réelle pour des environnements structurés et connus à l'avance. L'objectif principal est de pouvoir adresser des transferts de compétences dans des environnements sujets à des perturbations (position des objets soumis à des incertitudes, modélisation imprécise du robot, ...). Pour cela, la mise en place d'une boucle fermée paraît tout à fait judicieuse mais soulève quelques verrous à lever :

Le retard : même si le DA-UNN présenté dans la Section 5.4.7, propose une première solution de gestion du retard, cette méthode a l'inconvénient de supposer un retard

connu, parfaitement identifié et constant. Pour surmonter ces inconvénients, quelques pistes peuvent être évaluées comme l'ajout d'un bloc permettant d'estimer le retard en fonction des entrées et des actions de l'agent comme montré sur la Figure 6.2a, ou en fonction des entrées et des actions dans l'espace latent comme montré sur la Figure 6.2b. Cette seconde option permettrait d'avoir uniquement le bloc d'estimation du retard à déterminer qui sera commun à tous les robots contairement à la première méthode qui nécessite un estimateur par robot. Une autre piste serait également de considérer des réseaux de neurones récurrents que ce soit pour décrire les bases du robot ou dans le bloc relatif à la tâche.



(a) Identification du retard au niveau de l'agent.

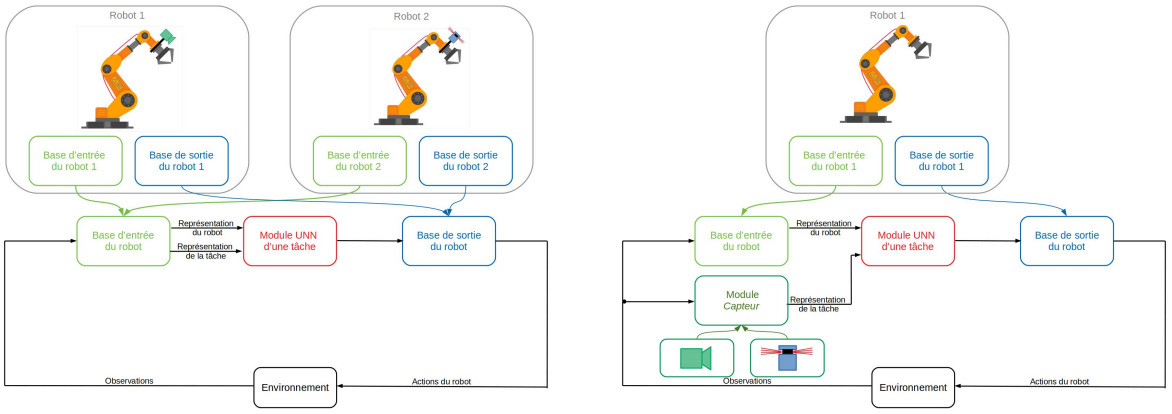
(b) Identification du retard au niveau de l'espace latent.

FIGURE 6.2 – Proposition de gestion des retards dans le cadre du transfert.

La gestion des capteurs : en effet, la mise en place d'une boucle fermée nécessite l'utilisation de capteurs proprioceptifs pour déterminer l'état du robot et extéroceptifs pour déterminer l'état de l'environnement et de la tâche à réaliser. Dans ce contexte, comment doit-on considérer deux robots identiques mais équipés de capteurs extéroceptifs différents (par exemple d'une caméra RGB-D pour l'un est d'un Lidar 3D pour l'autre) ? Faut-il les considérer comme deux robots différents, comme imagé sur la Figure 6.3a, donc déterminer pour chacun leur base d'entrée et de sortie ou conserver le découpage actuel de l'UNN en y ajoutant un module qui sera propre à chaque capteur comme présenté sur la Figure 6.3b ?

La gestion de capteurs augmente encore la difficulté du transfert si on considère des fréquences d'acquisition différentes, en effet une caméra peut fournir plusieurs dizaines d'image par seconde alors qu'un Lidar 3D va être beaucoup moins rapide. Cette incohérence des fréquences et des différentes informations devra donc être pris en compte.

Passage à l'échelle : Ce terme peut déterminer la gestion du transfert entre des robots de taille différente que je présenterai dans la Section 6.2.2. Ici il s'agit de s'intéresser à un autre aspect qui est la prise en compte d'un grand nombre de robots et d'un grand nombre de tâches. Actuellement, deux robots sont utilisés pour construire l'espace latent, via la résolution d'une tâche simple. On peut se demander comment choisir ces deux robots dans un large panel ? Vaut-il mieux choisir un robot avec des faibles capacités (sous-actionné) ou au contraire un robot avec des capacités plus large (redondant) ? On peut également s'intéresser au choix de la tâche utilisée pour construire l'espace latent. Est-elle suffisamment "riche" pour gérer tous les comportements qui seront nécessaires pour résoudre un grand ensemble de tâche ?



(a) Robot unique.

(b) Dissociation du robot et de ces capteurs.

FIGURE 6.3 – Proposition de gestion des capteurs différents pour un même robot.

Alors qu'actuellement, l'apprentissage se fait de façon séquentielle comme montré dans le Chapitre 5, quel serait l'impact d'un apprentissage massivement parallèle considérant toutes les tâches et tous les robots pour construire en un seul apprentissage les bases d'entrée et de sortie des robots et les modules liés aux tâches ? Dans ce cas, comment assurer une bonne convergence de l'apprentissage ? Alors que ce passage à l'échelle est nécessaire pour justifier l'importance du transfert, il semble être l'un des verrous les plus difficiles à lever.

Mise en place d'une architecture d'évaluation du transfert : Nous venons de lister quelques solutions pour lever les verrous d'un transfert en boucle fermée. Cependant pour évaluer l'intérêt de ces méthodes, il faut disposer d'une méthode générique d'évaluation de la qualité des méthodes de transfert. Dans nos travaux et dans la plupart des travaux qui traitent du transfert [1], l'évaluation du transfert se fait en comparant des résultats (performances et temps d'apprentissage) issus du transfert et ceux appris *from scratch* sur un panel réduit de robots et de tâches.

L'une des premières étapes de mon projet de recherche (qui sera traité lors d'un travail de thèse qui débutera à l'automne 2024), sera d'étudier les différents critères à prendre en compte pour qualifier un transfert. Parmi ces critères, on peut déjà envisager : la capacité à faire un transfert *zero shot* ou le temps de *fine tuning*, la valeur de la fonction de récompense, les performances. D'autres éléments comme le temps "humain" nécessaire, la consommation d'énergie pendant l'apprentissage ou pendant l'exécution pourraient également être considérés. L'objectif est de disposer d'une architecture d'évaluation du transfert comportant un panel important de robots et de tâches qui sera mis à disposition librement et que nous utiliserons pour quantifier la qualité de nos travaux futurs.

6.2.2 Garanties théoriques pour la génération de mouvements

Le point précédent de mon projet de recherche traitait principalement des axes de complexité de la tâche et de temps de calcul, que ce soit pour l'apprentissage ou lors de l'utlisation de l'IA. Pour garantir le bon accomplissement de la tâche, il est également important d'apporter des preuves théoriques en amont de l'expérimentation sur les robots.

1. Du moins, à ma connaissance

Apport de garanties théoriques : L'un des défauts des méthodes basées sur l'IA présenté dans la Section 5.1 est son côté boîte noire où on constate statistiquement que dans la majorité des cas cela fonctionne. Malheureusement, dans certains cas comme l'accomplissement d'une tâche à proximité ou en interaction avec un humain, cette "preuve" statistique n'est pas suffissante. Pour un déploiement du robot, il faut être en mesure d'apporter des preuves de son bon fonctionnement pour toutes les situations susceptibles de se produire. Ce problème peut se formaliser en imposant que pour chaque état du système et de l'environnement S^k à un instant k, l'action A^k définie par la politique conduise à un nouvel état S^{k+1} qui respecte toutes les contraintes g en lien avec la sécurité du robot et de son environnement ainsi que l'accomplissement de la tâche [2], tel que :

$$\forall S^k \text{ tel que } g(S^k) < 0 \text{ alors } g(S^{k+1}) < 0 \text{ avec } S^{k+1} = \xi(S^k, A^k) \tag{6.1}$$

La preuve théorique dans ce cas pourrait être obtenue via les méthodes AI en démontrant qu'il n'y a pas de solution à son problème dual (recherche d'un état S^k qui valide que $g(S^{k+1}) > 0$). Nous avons vu dans le Chapitre 4 que l'AI est assez efficace pour montrer l'absence de solution à un problème. Cependant, même si une première étape sera de tester ce principe, il est très probable que les temps de calcul soient prohibitifs à cause de la complexité du robot, de la tâche et des réseaux de neurones(dimension et non linéarité des réseaux de neurones). Plusieurs pistes pourront être explorées pour apporter des preuves théoriques comme l'étude de fonctions de bases non linéaires (qui pourraient être plus adaptées aux couches d'activation des réseaux de neurones), l'utilisation du *Modal Interval* [3, 121] ou de la notion d'inclusion différentielle [5, 19].

Capacité supposée à réaliser une tâche : A long terme, nous souhaitons donc disposer d'un grand panel de robots différents pouvant apprendre des tâches qui seront transmises à d'autres robots du panel. Dans ce cas, il paraît vraisemblable que ceux-ci n'auront pas les mêmes capacités que ce soit par leur différences morphologiques (manipulateur fixe, manipulateur sur base mobile, humanoïde, hexapode, ...) ou par la puissance qu'ils peuvent déployer. L'utilisation d'un facteur d'échelle pourrait permettre le transfert, par exemple : en considérant un robot deux fois plus petit que le robot source, on peut supposer qu'il pourra manipuler des objets également deux fois plus petits que le robot source.

Dans le cas où un robot avec des capacités très inférieures au robot source doit accomplir la même tâche (avec les mêmes objets, en appliquant le même effort) la notion de transfert peut perdre tout son sens. Si on inclut les robots mobiles, naviguer dans un environnement encombré pourrait être transféré d'un robot mobile (à roue) vers un robot humanoïde. Le transfert d'un humanoïde vers un robot mobile pose déjà plus de questions car si une stratégie est d'enjamber les obstacles, le robot mobile ne pourra pas le faire. Le transfert vers un robot manipulateur fixe lui n'a aucun sens.

En considérant une architecture modulaire, comme celle de l'UNN ou de ses dérivées, les robots et les tâches sont ramenés de manière abstraite à des réseaux de neurones. A long terme, il sera important de définir une notion de *capacité supposée* d'un robot à réaliser une tâche en se basant sur ces réseaux. L'étude de l'accessibilité des espaces latents en entrée et en sortie du bloc lié à la tâche pourrait fournir une métrique de compatibilité entre les modules propres au robot et ceux de la tâche.

2. cette formulation est volontairement simplifiée, il faudrait également prendre en compte les imprécisions sur l'état S^k et donc sur S^{k+1}

6.2.3 L'humain dans le transfert

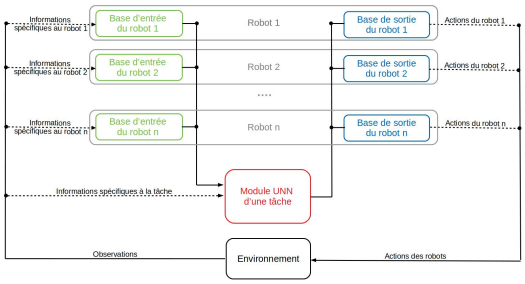
Transfert humain vers robot : Une application qui semble accessible avec les méthodes de transfert concerne l'imitation ou l'apprentissage par démonstration afin de transmettre un savoir-faire humain vers un ou plusieurs (dans notre cas) robots. En effet, si on suppose que l'état d'un humain et les actions qu'il mène sont observables, il peut être considéré comme une nouveau robot à appairer dans le cadre de l'architecture LS-UNN présenté dans la Section 5.4.8 via la tâche *primitive* et ainsi obtenir les modules de l'humain. Puis en considérant les modules de l'humain connus, on pourrait déterminer le module d'une nouvelle tâche qui serait transférable vers les autres robots, via des méthodes d'apprentissage supervisé par exemple. Bien que conceptuellement simple à imaginer, l'imitation ou le transfert de compétences humain vers robot se heurtera à la différence de capacité motrice et de dextérité entre l'humain et le robot dont nous avons déjà discuté entre robots dans la Section 6.2.2.

Co-manipulation homme-robot : Les premiers résultats de l'UNN présentés dans la Section 5.5 ont montré que les tâches de co-manipulation pouvaient également faire l'objet d'un transfert des compétences. En simulation, nous avons montré qu'une tâche apprise par un ensemble de robots peut être (transférée et) résolue en changeant un ou plusieurs de ces robots. Peut-on envisager de remplacer un ou plusieurs des robots par un ou plusieurs opérateurs humains ? Dans le cas d'une interaction, la communication est un point primordial. Si on considère une tâche représentée par un seul module UNN, comme sur la Figure 6.4a, on imagine que la communication se ferait au niveau des couches cachées. Dans ce cas, l'ajout de l'humain serait difficile à envisager. Au contraire, si on considère les systèmes (humain ou robot) comme indépendants, le module UNN pourrait être fragmenté comme sur la Figure 6.4b (en considérant des modules identiques ou pas). Dans cas, en évitant ainsi toute communication, la structure de l'UNN permet facilement le remplacement d'un robot par humain. Une solution intermédiaire pourrait être de limiter la communication à quelques dimensions comme montré en Figure 6.4c [3]. Dans cette solution la création d'un module *"communication"* nécessitera une phase d'apprentissage en manipulation réelle, la plus courte possible. De plus ce module *"communication"* devra être générique à tous les opérateurs et non spécifique afin de ne pas requérir de phase d'adaptation.

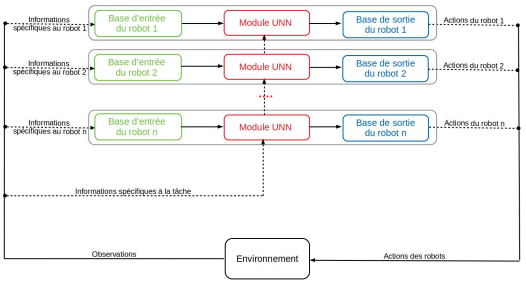
Les émotions dans l'interaction homme-robot : Cette communication entre robot et humain pourrait également se faire de manière non verbale. En tant qu'êtres humains, nous émettons de manière consciente ou inconsciente un grand nombre d'informations physiques, psychologiques et émotionnelles à travers nos mouvements. Nous sommes également capables de décrypter ces informations chez un interlocuteur ou une personne à proximité [83] et de nous y adapter [148]. Réciproquement, les mouvements que nous percevons affectent également notre état d'esprit [169]. L'expressivité des émotions par un robot pourrait donc avoir un impact sur la qualité d'une interaction homme-robot [80].

La génération de mouvement réalisant une tâche en laissant transparaître des émotions semble réalisable dans le cas de robots redondants en profitant de la dimension non nulle du noyau d'une tâche comme montré dans [30]. En effet, en supposant que l'espace latent dispose des mêmes propriétés on peut imaginer "déplacer" l'action émise par le module de

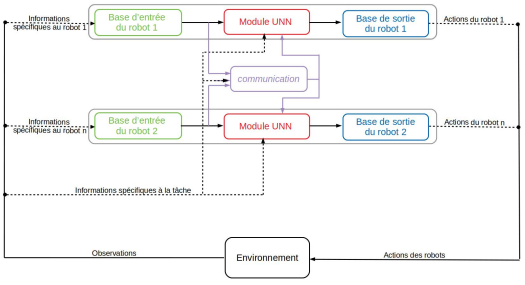
3. Cette figure ne considère que deux robots/humains pour simplifier la visualisation



(a) bloc UNN compact.



(b) bloc UNN fragmenté.



(c) bloc UNN avec communication réduite.

FIGURE 6.4 – Propositions d'architecture d'une tâche de co-manipulation pour intégrer un ou plusieurs humains à la place d'un ou de plusieurs robots.

la tâche afin d'imprégner une émotion au robot, comme montré dans la Figure 6.5, sans perturber la réalisation de la tâche.

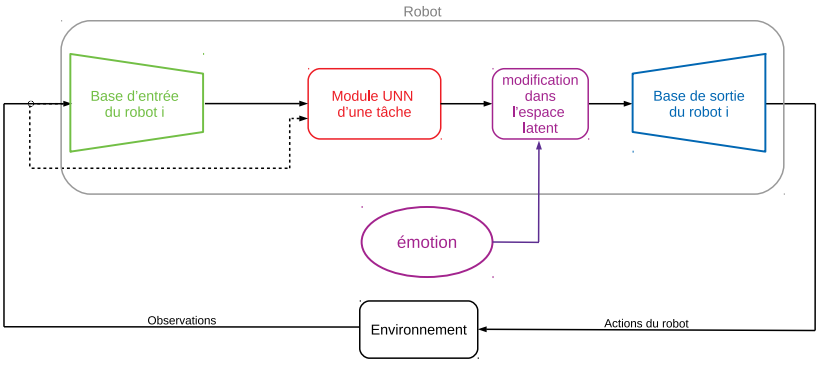


FIGURE 6.5 – Modification de l'action proposée par l'UNN en fonction d'une émotion désirée.

Dans ce manuscrit, j'ai présenté ma vision de l'algorithme de génération de mouvement idéal qui, à la fois sera apte à gérer la complexité du mouvement, à garantir la sécurité du robot et de son environnement et à respecter un temps de calcul cohérent avec une application réelle. J'ai présenté quelques unes de mes contributions ayant pour but de se rapprocher de cet algorithme idéal. Finalement, j'ai recensé plusieurs axes de recherche qui semblent pouvoir être menés à court ou moyen terme. Le défi à long terme sera d'intégrer toutes ces améliorations dans une architecture unique qui permettra un transfert efficace en prenant en compte un grand panel de robots et de tâches.

Bibliographie

[1] https://thedatafrog.com/en/articles/image-recognition-transfer-learning/.

[2] Edward J Anderson. Infinite programming. In *Proceedings of an International Symposium on Infinite Dimensional Linear Programming*, pages 7–10. Springer Science & Business Media, September 1984.

[3] Joaquim Armengol, Josep Vehí, Louisé Travé-Massuyès, and Miguel Ángel Sainz. Application of modal intervals to the generation of error-bounded envelopes. *Reliable Computing*, 7 :171–185, 2001.

[4] Tamim Asfour, Julian Schill, Heiner Peters, Cornelius Klas, Jens Bücker, Christian Sander, Stefan Schulz, Artem Kargov, Tino Werner, and Volker Bartenbach. Armar-4 : A 63 dof torque controlled humanoid robot. In *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 390–396, 2013.

[5] Jean Pierre Aubin and A. Cellina. *Differential Inclusions : Set-Valued Maps and Viability Theory*. Springer-Verlag, Berlin, Heidelberg, 1984.

[6] Amir Massah B, Arman Sharifi K., Yaser Salehinia, and Farid Najafi. An open loop walking on different slopes for nao humanoid robot. *Procedia Engineering*, 41 :296–304, 2012. International Symposium on Robotics and Intelligent Sensors 2012 (IRIS 2012).

[7] Samuel Barrett, Matthew E. Taylor, and Peter Stone. Transfer learning for reinforcement learning on a physical robot. In *Adaptive Agents and Multi-Agent Systems*, 2010.

[8] Samuel Beaussant. *Transfer Learning between robots with State Abstraction*. PhD thesis, 2023.

[9] Samuel Beaussant, Sébastien Lengagne, Benoit Thuilot, and Olivier Stasse. Towards zero-shot cross-agent transfer learning via latent-space universal notice network. *submitted to Robotics and Autonomous Systems*, 2024.

[10] Mehdi Benallegue, Adrien Escande, Sylvain Miossec, and Abderrahmane Kheddar. Fast c1 proximity queries using support mapping of sphere-torus-patches bounding volumes. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 483 –488, may 2009.

[11] Mourad Benoussaad. *Protocole d'identification sous FES et synthèse des séquences de stimulation chez le blessé médullaire*. Theses, Université Montpellier II - Sciences et Techniques du Languedoc, December 2009.

[12] Martin Berz, Georg Hoffstätter, and Georg Hoffst Atter. Computation and application of taylor polynomials with interval remainder bounds. *Reliable Computing*, 4 :83–97, 1998.

[13] James E. Bobrow. Optimal robot plant planning using the minimum-time criterion. *Robotics and Automation, IEEE Journal of*, 4(4) :443 –450, aug 1988.

[14] D. Boërio, J-Y. Hogrel, A. Créange, and J-P. Lefaucheur. Méthodes et intérêt clinique de la mesure de la période réfractaire nerveuse périphérique chez l'homme. *Neurophysiologie Clinique/Clinical Neurophysiology*, 34(6) :279–291, 2004.

[15] R. Bohlin and L. E. Kavraki. Path planning using lazy PRM. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 1, pages 521–528, San Francisco, CA, USA, 2000.

[16] Karim Bouyarmane, Adrien Escande, Florent Lamiraux, and Abderrahmane Kheddar. Potential field guide for humanoid multicontacts acyclic motion planning. In *IEEE Int. Conf. on Robotics and Automation*, may 2009.

[17] Karim Bouyarmane and Abderrahmane Kheddar. Humanoid robot locomotion and manipulation step planning. *Advanced Robotics (Int. J. of the Robotics Society of Japan), Special Issue on the Cutting Edge of Robotics in Japan*, 26(10) :1099–1126, July 2012.

[18] Mary D. Klein Breteler, Stan C.A.M. Gielen, and Ruud G.J. Meulenbroek. Endpoint constraints in aiming movements effects of approach angle and speed. *Biological Cybernetics*, 85(1) :65 – 75, july 2001.

[19] Bernard Brogliato and Aneel Tanwani. Dynamical systems coupled with monotone set-valued operators : Formalisms, applications, well-posedness, and stability. *SIAM Review*, 62(1) :3–129, February 2020.

[20] Stanislas Brossette and Pierre-Brice Wieber. Collision avoidance based on separating planes for feet trajectory generation. In *Humanoids 2017 - IEEE RAS International Conference on Humanoid Robots* , pages 509–514, Birmingham, United Kingdom, November 2017. IEEE.

[21] José Luis Carrillo-Medina and Roberto Latorre. Influence of the refractory period on neural networks based on the recognition of neural signatures. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9, 2015.

[22] Gilles Chabert and Luc Jaulin. computing the pessimism of inclusion functions. *Reliable Computing*, 13 :489–505, 2007.

[23] Gilles Chabert and Luc Jaulin. Contractor Programming. *Artificial Intelligence*, 173 :1079–1100, 2009. WOS.

[24] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan D. Ratliff, and Dieter Fox. Closing the sim-to-real loop : Adapting simulation randomization with real world experience. *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979, 2018.

[25] Mohamad Chehadeh, Igor Boiko, and Yahya Zweiri. The role of time delay in sim2real transfer of reinforcement learning for cyber-physical systems, 09 2022.

[26] Xuechao Chen, Wenxi Liao, Zhangguo Yu, Haoxiang Qi, Xinyang Jiang, and Qiang Huang. Motion coordination for humanoid jumping using maximized joint power. *Advances in Mechanical Engineering*, 13(6) :16878140211028448, 2021.

[27] Matthew Chignoli, Donghyun Kim, Elijah Stanger-Jones, and Sangbae Kim. The mit humanoid robot : Design, motion planning, and control for acrobatic behaviors. In *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*, pages 1–8, 2021.

[28] Benjamin Chrétien. *GPU-based Semi-Infinite Optimization for Whole-Body Robot Control*. PhD thesis, 07 2016.

[29] Benjamin Chrétien, Adrien Escande, and Abderrahmane Kheddar. Gpu robot motion planning using semi-infinite nonlinear programming. *IEEE Transactions on Parallel and Distributed Systems*, 27 :1–1, 01 2016.

[30] Josep-Arnau Claret, Gentiane Venture, and Luis Basañez. Exploiting the robot kinematic redundancy for emotion conveyance to humans as a lower priority task. *International Journal of Social Robotics*, 9 :277–292, 04 2017.

[31] W.G. Cochran. *Sampling Techniques*. Wiley Series in Probability and Statistics. Wiley, 1977.

[32] Marco Cuturi and Mathieu Blondel. Soft-dtw : a differentiable loss function for time-series. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 894–903. JMLR.org, 2017.

[33] Hongkai Dai, Andrés Valenzuela, and Russ Tedrake. Whole-body motion planning with centroidal dynamics and full kinematics. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 295–302, 2014.

[34] Lisandro Dalcin and Yao-Lung L. Fang. mpi4py : Status update after 12 years of development. *Computing in Science & Engineering*, 23(4) :47–54, 2021.

[35] David Daney, Nicolas Andreff, Gilles Chabert, and Yves Papegay. Interval method for calibration of parallel robots : Vision-based experiments. *Mechanism and Machine Theory*, 41(8) :929–944, 2006. Special issue on CK 2005, International Workshop on Computational Kinematics.

[36] David Daney, Yves Papegay, and Arnold Neumaier. Interval methods for certification of the kinematic calibration of parallel robots. volume 2, pages 1913–1918, 04 2004.

[37] Mélodie Daniel. *Optimizing Decision-Making for Human-Robot Collaboration*. PhD thesis, 2022.

[38] Mélodie Daniel, Aly Magassouba, Miguel Aranda, Laurent Lequièvre, Juan Antonio Corrales Ramón, Roberto Iglesias Rodriguez, and Youcef Mezouar. Multi actor-critic ddpg for robot action space decomposition : A framework to control large 3d deformation of soft linear objects. *IEEE Robotics and Automation Letters*, 9(2) :1318–1325, 2024.

[39] Ewen Dantec, Maximilien Naveau, Pierre Fernbach, Nahuel Villa, Guilhem Saurel, Olivier Stasse, Michel Taix, and Nicolas Mansard. Whole-body model predictive control for biped locomotion on a torque-controlled humanoid robot. In *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, pages 638–644, 2022.

[40] Tilak de Alwis. Maximizing or minimizing polynomials using algebraic inequalities. In *Proc. 9th Asian Technological Conference on Mathematics*, pages 88–97, Dec. 2004. Singapore.

[41] Carl de Boor. Package for calculating with b-splines. *SIAM Journal on Numerical Analysis*, 14 :57, 10 1973.

[42] Carl De Boor. *A Pratical Guide to Splines*, volume 27. Springer-Verlag, New York, 1978.

[43] Andrea Del Prete, Nicolas Mansard, Oscar Ramos, Olivier Stasse, and Francesco Nori. Implementing torque control with high-ratio gear boxes and without joint-torque sensors. *International Journal of Humanoid Robotics*, 13, 03 2015.

[44] Coline Devin, Abhishek Gupta, Trevor Darrell, P. Abbeel, and Sergey Levine. Learning modular neural network policies for multi-task and multi-robot transfer. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2169–2176, 2017.

[45] Moritz Diehl, Hans Georg Bock, Holger Diedam, and Pierre-Brice Wieber. Fast Direct Multiple Shooting Algorithms for Optimal Robot Control. In *Fast Motions in Biomechanics and Robotics*, Heidelberg, Germany, 2005.

[46] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1 :269–271, 1959.

[47] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf : A deep convolutional activation feature for generic visual recognition. pages 647–655, 10 2013.

[48] Hassan El Makssoud. *Modélisation et Identification des Muscles Squelettiques
sous Stimulation Electrique Fonctionnelle*. Theses, Université Montpellier II - Sciences et Techniques du Languedoc, December 2005.

[49] Benjamin Ellenberger. Pybullet gymperium. `https://github.com/benelot/pybullet-gym`, 2018–2019.

[50] Adrien Escande and Abderrahmane Kheddar. Contact planning for acyclic motion with tasks constraints. In *IEEE/RSJ International Conference on Intelligent RObots and Systems (IROS 2009)*, Oct. 11-15 2009.

[51] Adrien Escande, Abderrahmane Kheddar, and Sylvain Miossec. Planning support contact-points for humanoid robots and experiments on HRP-2. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 2974–2979, Beijing, October 2006.

[52] Adrien Escande, Sylvain Miossec, and Abderrahmane Kheddar. Continuous gradient proximity distance for humanoids free-collision optimized-postures. In *IEEE-RAS 7th International Conference on Humanoid Robots*, 2007.

[53] Gerald E. Farin, Josef Hoschek, and Myung-Soo Kim. Handbook of computer aided geometric design. 2002.

[54] Mohammadhadi Farzanehkaloorazi, Mehdi Masouleh, and Stéphane Caro. Interval-analysis-based determination of the singularity-free workspace of gough-stewart parallel robots. pages 1–6, 05 2013.

[55] R. A. Fisher. *Statistical Methods for Research Workers*, pages 66–70. Springer New York, New York, NY, 1992.

[56] Albert Gidon and Idan Segev. Principles governing the operation of synaptic inhibition in dendrites. *Neuron*, 75(2) :330–341, 2012.

[57] C. Giorgi and S. Marinelli. Roles and transcriptional responses of inhibitory neurons in learning and memory. *Frontiers in Molecular Neuroscience*, 14, 2021.

[58] Ian J. Goodfellow. NIPS 2016 tutorial : Generative adversarial networks. *CoRR*, abs/1701.00160, 2017.

[59] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2672–2680, Cambridge, MA, USA, 2014. MIT Press.

[60] Klaus Greff, Rupesh K. Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. Lstm : A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10) :2222–2232, 2017.

[61] Abhishek Gupta, Coline Devin, Yuxuan Liu, Pieter Abbeel, and Sergey Levine. Learning invariant feature spaces to transfer skills with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[62] Rémy Guyonneau, Sebastien Lagrange, Laurent Hardouin, and Philippe Lucidarme. Guaranteed interval analysis localization for mobile robots. *Journal on Advanced Robotics*, 28, 07 2014.

[63] Hossein Habibi, Chenghao Yang, Rongjie Kang, Ian D. Walker, Isuru S. Godage, Xin Dong, and David T. Branson. Modelling an actuated large deformation soft continuum robot surface undergoing external forces using a lumped-mass approach. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5958–5963, 2018.

[64] Mélodie Hani Daniel Zakaria, Sébastien Lengagne, Juan Antonio Corrales Ramón, and Youcef Mezouar. General framework for the optimization of the human-robot collaboration decision-making process through the ability to change performance metrics. *Frontiers in Robotics and AI*, 8, 2021.

[65] E. Hansen and G.W. Walster. *Global optimization using interval analysis*. Marcel Dekker, 2nd edition, 2004.

[66] Eldon R. Hansen. Global optimization using interval analysis — the multi-dimensional case. *Numerische Mathematik*, 34 :247–270, 1980.

[67] Jianxiang He, Yanzi Li, Yingtian Liu, Jiyang Chen, Chaoqun Wang, Rui Song, and Yibin Li. The development of spiking neural network : A review. In *2022 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 385–390, 2022.

[68] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[69] Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae : Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2016.

[70] Hirohisa Hirukawa, Shizuko Hattori, Kensuke Harada, Shuuji Kajita, Kenji Kaneko, Fumio Kanehiro, Kiyoshi Fujiwara, and Mitsuharu Morisawa. A universal stability criterion of the foot contact of legged robots - adios zmp. In *IEEE International Conference on Robotics and Automation (ICRA).*, pages 1976– 1983, may 2006.

[71] Ronald A. Howard. Dynamic programming and markov processes. 1960.

[72] Xiaowei Huang, Daniel Kroening, Wenjie Ruan, James Sharp, Youcheng Sun, Emese Thamo, Min Wu, and Xinping Yi. A survey of safety and trustworthiness of deep neural networks : Verification, testing, adversarial attack and defence, and interpretability. *Computer Science Review*, 37 :100270, 2020.

[73] K. Ichida. Constrained optimization using interval analysis. *Computers & Industrial Engineering*, 31(3) :933–937, 1996. 18th International Conference on Computers and Industrial Engineering.

[74] Rishabh Jangir, Guillem Alenyà, and Carme Torras. Dynamic cloth manipulation with deep reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4630–4636, 2020.

[75] Noémie Jaquier, Michael C. Welle, Andrej Gams, Kunpeng Yao, Bernardo Fichera, Aude Billard, Aleš Ude, Tamim Asfour, and Danica Kragic. Transfer learning in robotics : An upcoming breakthrough ? a review of promises and challenges, 2023.

[76] Luc Jaulin. Robust set-membership state estimation ; application to underwater robotics. *Automatica*, 45(1) :202–206, 2009.

[77] Luc Jaulin and Éric Walter. Guaranteed tuning, with application to robust control and motion planning. *Automatica*, 32(9) :1217–1221, 1996.

[78] Luc Jaulin, Michel Kieffer, Olivier Didrit, and Eric Walter. *Applied interval analysis*. Springer, 2001.

[79] Luc Jaulin and Fabrice Le Bars. An interval approach for stability analysis : Application to sailboat robotics. *IEEE Transactions on Robotics*, 29(1) :282–287, 2013.

[80] Sarah A. Jessup and Tamera R. Schneider. Chapter 22 - the role of emotions in human-robot interactions. In Chang S. Nam and Joseph B. Lyons, editors, *Trust in Human-Robot Interaction*, pages 515–530. Academic Press, 2021.

[81] Arthur Juliani, Vincent-Pierre Berges, Ervin Teng, Andrew Cohen, Jonathan Harper, Chris Elion, Chris Goy, Yuan Gao, Hunter Henry, Marwan Mattar, and Danny Lange. Unity : A general platform for intelligent agents, 2020.

[82] Bassel Kaddar, Yannick Aoustin, and Christine Chevallereau. Arm swing effects on walking bipedal gaits composed of impact, single and double support phases. *Robotics and Autonomous Systems*, 66 :104–115, 2015.

[83] H. Kadone, H. Hicheur, J. Grezes, and A.Berthoz. Nature of the kinematic cues underlying the perception of emotions during human gait. In *the Int. Soc. for Posture and Gait Research*, 2009.

[84] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kensuke Harada, Kazuhito Yokoi, and Hirohisa Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1620 – 1626, september 2003.

[85] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kazuhito Yokoi, and Hirohisa Hirukawa. A realtime pattern generator for biped walking. In *IEEE Int. Conf. on Robotics and Automation (ICRA).*, may 2002.

[86] Shuuji Kajita, Takashi Nagasaki, Kenji Kaneko, Kazuhito Yokoi, and Kazuo Tanie. A running controller of humanoid biped HRP-2LR. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 616–622, April 2005.

[87] Rawan Kalawoun. *Motion planning of multi-robot system for airplane stripping*. Theses, Université Clermont Auvergne [2017-2020], April 2019.

[88] Rawan Kalawoun, Sébastien Lengagne, Francois Bouchon, and Youcef Mezouar. Bsplines properties with interval analysis for constraint satisfaction problem application in robotics. In *15th International Conference on Intelligent Autonomous Systems IAS-15*, june 2018.

[89] Kenji Kaneko, Fumio Kanehiro, Shuuji Kajita, Hirohisa Hirukawa, Toshikazu Kawasaki, Masaru Hirata, Kazuhiko Akachi, and Takakatsu Isozumi. Humanoid robot

HRP-2. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 2, pages 1083–1090, April/May 2004.

[90] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *CoRR*, abs/1710.10196, 2017.

[91] Oussama Khatib, Luis Sentis, and Jae-Heung Park. A unified framework for whole-body humanoid robot control with multiple constraints and contacts. In *Springer Tracts in Advanced Robotics - STAR Series, European Robotics Symposium (EURON)*, 2008.

[92] M. Kieffer, L. Jaulin, and E. Walter. Guaranteed recursive nonlinear state estimation using interval analysis. In *Decision and Control, 1998. Proceedings of the 37th IEEE Conference on*, volume 4, pages 3966–3971, Tampa, FL, USA, December 1998.

[93] Kuno Kim, Yihong Gu, Jiaming Song, Shengjia Zhao, and Stefano Ermon. Domain adaptive imitation learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5286–5295. PMLR, 13–18 Jul 2020.

[94] J. Koenemann, A. Del Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard. Whole-body model-predictive control applied to the hrp-2 humanoid. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3346–3351, 2015.

[95] Adrien Koessler, Alexandre Goldsztejn, Sébastien Briot, and Nicolas Bouton. Certified detection of parallel robot assembly mode under type 2 singularity crossing trajectories. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6073–6079, 2017.

[96] Jens Kotlarski, Roderick de Nijs, Houssem Abdellatif, and Bodo Heimann. New interval-based approach to determine the guaranteed singularity-free workspace of parallel robots. In *2009 IEEE International Conference on Robotics and Automation*, pages 1256–1261, 2009.

[97] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60 :84 – 90, 2012.

[98] J. J. Jr. Kuffner and S. M. LaValle. RRT-connect an efficient approach to single-query path planning. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 2, pages 995–1001, San Francisco, CA, USA, 2000.

[99] Solomon Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22 :79–86, 1951.

[100] Michael Kutzer, Sean Segreti, Christopher Brown, Russell Taylor, and Simon Mears. Design of a new cable-driven manipulator with a large open lumen : Preliminary applications in the minimally-invasive removal of osteolysis. pages 2913–2920, 05 2011.

[101] Jacques Laffont and Sébastien Lengagne. Mise en place de projets industriels pour des élèves ingénieurs. In *Colloque de l'Enseignement des Technologies et des Sciences de l'Information et des Systèmes (CETSIS 2023)*, 2023.

[102] Craig Lawrence, Jian L. Zhou, and Andre L. Tits. *User's Guide for CFSQP Version 2.5 A C Code for Solving (Large Scale) Constrained Nonlinear (Minimax) Optimization Problems, Generating Iterates Satisfying All Inequality Constraints.* Electrical

Engineering Department, Institute for Systems Research University of Maryland, College Park, MD 20742.

[103] Sung-Hee Lee, Junggon Kim, F.C. Park, Mumsang Kim, and James E. Bobrow. Newton-type algorithms for dynamics-based robot movement optimization. In *IEEE Transactions on robotics*, volume 21, pages 657– 667, 2005.

[104] Youngeun Lee, Sébastien Lengagne, Abderrahmane Kheddar, and Young J. Kim. Accurate evaluation of a distance function for optimization-based motion planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, 2012.

[105] Sébastien Lengagne. *Planification et re-planification de mouvements sûrs pour les robots humanoïdes. (in French)*. PhD thesis, Université Montpellier II - Sciences et Techniques du Languedoc, october, 21 2009.

[106] Sébastien lengagne, Jovana Jovic, Camilla Pierella, Philippe Fraisse, and Christine Azevedo Coste. Generation of multi-contact motions with passive joints improvement of sitting pivot transfer strategy for paraplegics. In *IEEE/RAS-EMBS Int. Conf. on Biomedical Robotics and Biomechatronics (BioRob)*, 2012.

[107] Sebastien Lengagne, Rawan Kalawoun, François Bouchon, and Youcef Mezouar. Reducing pessimism in Interval Analysis using Bsplines Properties Application to Robotics. *Reliable Computing*, 27 :63–87, July 2020.

[108] Sébastien Lengagne, Abderrahmane Kheddar, Sébastien Druon, and Eiichi Yoshida. Emulating human leg impairments and disabilities in walking with humanoid robots. In *IEEE Int. Conf. on RObotics & BIOmimetics*, 2011.

[109] Sébastien Lengagne, Paul Mathieu, Abderrahmane Kheddar, and Eiichi Yoshida. Generation of dynamic motions under continuous constraints efficient computation using b-splines and taylor polynomials. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010.

[110] Sébastien Lengagne, Nacim Ramdani, and Philippe Fraisse. Planning and fast replanning safe motions for humanoid robots. *IEEE Transactions on Robotics*, 27(6) :1095 –1106, dec. 2011.

[111] Sébastien Lengagne, Joris Vaillant, and Eiichi Yoshida. Generation of Whole-body Optimal Dynamic Multi-Contact Motions. *The International Journal of Robotics Research*, page 17, April 2013.

[112] Sébastien Lengagne. Comparison of the basis-functions for interval-based optimization. *submitted to International Journal of Approximate Reasoning*, –.

[113] Sébastien Lengagne. Detailed results of the comparison of basis functions used in interval-based optimization process https//uca.hal.science/hal-04320488.

[114] Sébastien Lengagne. Mise en pratique de compétences en robotique à partir d'un robot réel et de sa simulation. In *Colloque de l'Enseignement des Technologies et des Sciences de l'Information et des Systèmes (CETSIS 2023)*, 2023.

[115] Timothée Lesort, Natalia Díaz-Rodríguez, Jean-Franois Goudou, and David Filliat. State representation learning for control : An overview. *Neural Networks*, 108 :379–392, 2018.

[116] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4-5) :421–436, 2018.

[117] Sergey Levine, Peter Pastor, Alex Krizhevsky, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37 :421 – 436, 2016.

[118] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[119] Zhi-Song Liu, Vicky Kalogeiton, and Marie-Paule Cani. Multiple style transfer via variational autoencoder. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 2413–2417, 2021.

[120] Jan Willem Loeve. Finding time-optimal trajectories for the resonating arm using the rrt* algorithm. 2012.

[121] Xin Luo and Min Sun. Development of modal interval algorithm for solving continuous minimax problems. *Applied Mathematics and Computation*, 422 :126970, 2022.

[122] Ndivhuwo Makondo, Benjamin Rosman, and Osamu Hasegawa. Accelerating model learning with inter-robot knowledge transfer. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2417–2424, 2018.

[123] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, N. Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym : High performance gpu-based physics simulation for robot learning. *ArXiv*, abs/2108.10470, 2021.

[124] Nicolas Mansard, Olivier Aycard, and Carla Koike. Hierarchy of behaviours. In *IEEE Int. Conf. on Robotics and Biomimetics, ROBIO'05*, pages 583–588, Hong-Kong, China, July 2005.

[125] Jan Matas, Stephen James, and Andrew J. Davison. Sim-to-real reinforcement learning for deformable object manipulation. In Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 734–743. PMLR, 29–31 Oct 2018.

[126] Christophe Michel. *Modélisation mathématique de l'activité électrophysiologique des neurones auditifs primaires.* Theses, Université Montpellier II - Sciences et Techniques du Languedoc, December 2012.

[127] Sylvain Miossec, Sébastien Lengagne, Abderrahmane Kheddar, and Kazuhito Yokoi. Motion optimization of robotic systems and validation of HRP-2 robot. In *RSJ National Conference*, 2008.

[128] Sylvain Miossec, Kazuhito Yokoi, and Abderrahmane Kheddar. Development of a software for motion optimization of robots– application to the kick motion of the HRP-2 robot. In *IEEE International Conference on Robotics and Biomimetics*, pages 299–304, 2006.

[129] R.E. Moore. *Interval Analysis.* Prentice-Hall, Englewood Cliffs, 1966.

[130] Mehdi Mounsif. *Exploration of Teacher-Centered and Task-Centered paradigms for efficient transfer of skills between morphologically distinct robots.* PhD thesis, 2020.

[131] Mehdi Mounsif, Sebastien Lengagne, Benoit Thuilot, and Lounis Adouane. BAM ! Base Abstracted Modeling with Universal Notice Network : Fast Skill Transfer Between Mobile Manipulators. In *7th International Conference on Control, Decision and Information Technologies (CoDIT 2020)*, volume 1, pages 926–932, Prague, Czech Republic, June 2020.

[132] Mehdi Mounsif, Sébastien Lengagne, Benoit Thuilot, and Lounis Adouane. Task-specific loss : A teacher-centered approach to transfer learning between distinctly structured robotic agents. In Oleg Gusikhin, Kurosh Madani, and Janan Zaytoon, editors, *Informatics in Control, Automation and Robotics*, pages 166–186, Cham, 2022. Springer International Publishing.

[133] Mehdi Mounsif, Sébastien Lengagne, Benoit Thuilot, and Lounis Adouane. Coach-gan : Fast adversarial transfer learning between differently shaped entities. In *Proceedings of the 17th International Conference on Informatics in Control, Automation and Robotics - ICINCO*, pages 89–96. INSTICC, SciTePress, 2020.

[134] Mohamed Mustafa, Alexandru Stancu, Nicolas Delanoue, and Eduard Codres. Guaranteed slam—an interval approach. *Robotics and Autonomous Systems*, 100 :160–170, 2018.

[135] Meinard Müller. *Information Retrieval for Music and Motion*. 01 2007.

[136] Laura Bella Naumann and Henning Sprekeler. Presynaptic inhibition rapidly stabilises recurrent excitation in the face of plasticity. *PLOS Computational Biology*, 16(8) :1–28, 08 2020.

[137] Lukas Netz. Using horner schemes to improve the efficiency and precision of interval constraint propagation. 2015.

[138] A. Neumaier. *Interval methods for systems of equations*. Cambridge university press, Cambridge, 1990.

[139] Arnold Neumaier. Taylor forms - use and limits. *Reliable Computing*, 2003.

[140] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving rubik's cube with a robot hand, 2019.

[141] Gaël Pages. *Estimation de la posture d'un sujet paraplégique en vue d'une rééducation des membres inférieurs sous stimulation électrique fonctionnelle*. Theses, Université Montpellier II - Sciences et Techniques du Languedoc, December 2006.

[142] Gaël Pages, Nacim Ramdani, Philippe Fraisse, and David Guiraud. Upper body posture estimation for standing function restauration. In *IEEE International Conference on Robotics and Automation (ICRA).*, 2007.

[143] Gael Pages, Nacim Ramdani, Philippe Fraisse, and David Guiraud. Upper body posture estimation for standing function restoration. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3742–3747, 2007.

[144] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8, 2017.

[145] N. Perrin, O. Stasse, L. Baudouin, F. Lamiraux, and E. Yoshida. Fast humanoid robot collision-free footstep planning using swept volume approximations. *IEEE Transactions on Robotics*, 28(2) :427–439, 2012.

[146] Aurelio Piazzi and Antonio Visioli. Global minimum-time trajectory planning of mechanical manipulators using interval analysis. *International Journal of Control*, 71 :631–652(22), November 1998.

[147] Aurelio Piazzi and Antonio Visioli. Global minimum-jerk trajectory planning of robot manipulators. In *IEEE Transactions on Industrial Electronics*, volume 47, pages 140–149, febru 2000.

[148] Rosalind W. Picard. Affective computing : Challenges. *International Journal of HumanComputer Studies*, 59 :2003, 2003.

[149] Michael Posa, Cecilia Cantu, and Russ Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *International Journal of Robotics Research*, 33 :69–81, 10 2014.

[150] Jai Rajyaguru, Mario Villanueva, Boris Houska, and Benoit Chachuat. Chebyshev model arithmetic for factorable functions. *Journal of Global Optimization*, 68, 06 2017.

[151] Noëlie Ramuzat, Olivier Stasse, and Sebastien Boria. Benchmarking whole body controllers on the talos humanoid robot. *Frontiers in Robotics and AI*, 9, 03 2022.

[152] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf : An astounding baseline for recognition. In *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 512–519, 2014.

[153] Joseph Redmon and Ali Farhadi. Yolov3 : An incremental improvement. *arXiv*, 2018.

[154] R. Reemtsen. Semi-infinite programming discretization methods. 1998.

[155] Jose Sanchez, Juan Antonio Corrales Ramon, Belhassen-Chedli Bouzgarrou, and Youcef Mezouar. Robotic Manipulation and Sensing of Deformable Objects in Domestic and Industrial Applications : A Survey. *The International Journal of Robotics Research*, 37(7) :688 – 716, June 2018.

[156] Sandeep Singh Sandha, Luis Garcia, Bharathan Balaji, Fatima Anwar, and Mani Srivastava. Sim2real transfer for deep reinforcement learning with stochastic state transition delays. In Jens Kober, Fabio Ramos, and Claire Tomlin, editors, *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 1066–1083. PMLR, 16–18 Nov 2021.

[157] Terence Sanger and Pallavi N. Baljekar. The perceptron : a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6 :386–408, 1958.

[158] Edgar Schönfeld, Sayna Ebrahimi, Samarth Sinha, Trevor Darrell, and Zeynep Akata. Generalized zero- and few-shot learning via aligned variational autoencoders. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8239–8247, 2018.

[159] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017.

[160] Tanmay Shankar, Yixin Lin, Aravind Rajeswaran, Vikash Kumar, Stuart Anderson, and Jean Oh. Translating robot skills : Learning unsupervised skill correspondences across robots. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International*

Conference on Machine Learning, volume 162 of *Proceedings of Machine Learning Research*, pages 19626–19644. PMLR, 17–23 Jul 2022.

[161] Bruno Siciliano and Oussama Khatib. *Springer Handbook of Robotics*. Springer-Verlag, Berlin, Heidelberg, 2007.

[162] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv 1409.1556*, 09 2014.

[163] Lingfeng Sun, Haichao Zhang, Wei Xu, and Masayoshi Tomizuka. Efficient multi-task and transfer reinforcement learning with parameter-compositional framework. *IEEE Robotics and Automation Letters*, 8(8) :4569–4576, 2023.

[164] T. Sunaga. Theory of interval algebra and its application to numerical analysis. *RAAG Memoirs, Ggujutsu Bunken Fukuy-kai*, 2 :547–564, 1958.

[165] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[166] Oemer Terlemez, Stefan Ulbrich, Christian Mandery, Martin Do, Nikolaus Vahrenkamp, and Tamim Asfour. Master motor map (mmm) - framework and toolkit for capturing, representing, and reproducing human motion on humanoid robots. volume 2015, pages 0–0, 11 2014.

[167] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco : A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012.

[168] Jesus Tordesillas and Jonathan P. How. Minvo basis finding simplexes with minimum volume enclosing polynomial curves. *Computer-Aided Design*, 151 :103341, 2022.

[169] Nikolaus F. Troje. 12. Retrieving Information from Human Movement Patterns. *Understanding Events How Humans See, Represent, and Act on Events.*, pages 308–335, March 2008.

[170] Y. Uno, M. Kawato, and R. Suzuki. Formation and control of optimal trajectory in human multijoint arm movement. *Biological Cybernetics*, 6(2) :89–101, juin 1989.

[171] Joris Vaillant, Abderrahmane Kheddar, Hervé Audren, François Keith, Stanislas Brossette, Adrien Escande, Karim Bouyarmane, Kenji Kaneko, Mitsuharu Morisawa, Pierre Gergondet, Eiichi Yoshida, Shuuji Kajita, and Fumio Kanehiro. Multi-contact vertical ladder climbing with an hrp-2 humanoid. *Autonomous Robots*, 40, 03 2016.

[172] Josep Vehí, Inées Ferrer, and Miguel Ángel Sainz. A survey of applications of interval analysis to robust control. *IFAC Proceedings Volumes*, 35(1) :389–400, 2002. 15th IFAC World Congress.

[173] Miomir Vukobratović and Branislav Borovac. Zero-moment point thirty five years of its life. *International Journal of Humanoid Robotics*, 1(1) :157–173, 2004.

[174] Andreas Wächter and Lorenz T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106 :22–57, 2006.

[175] Bin Wang, Longhua Wu, Kangkang Yin, U. Ascher, Libin Liu, and Hui Huang. Deformation capture and modeling of soft objects. volume 34, 08 2015.

[176] Yang Wen, Leiting Chen, Yu Deng, and Chuan Zhou. Rethinking pre-training on medical imaging. *Journal of Visual Communication and Image Representation*, 78 :103145, 2021.

[177] Daniel Wolf, Tristan Payer, Catharina Silvia Lisson, Christoph Gerhard Lisson, Meinrad Beer, Michael Götz, and Timo Ropinski. Self-supervised pre-training with contrastive and masked autoencoder methods for dealing with small datasets in deep learning for medical imaging. *Scientific Reports*, 13(1) :20260, 2023.

[178] Markus Wulfmeier, Ingmar Posner, and P. Abbeel. Mutual alignment transfer learning. In *Conference on Robot Learning*, 2017.

[179] Yiting Xie and David Richmond. Pre-training on grayscale imagenet improves medical image classification. In Laura Leal-Taixé and Stefan Roth, editors, *Computer Vision – ECCV 2018 Workshops*, pages 476–484, Cham, 2019. Springer International Publishing.

[180] Peipei Xu, Wenjie Ruan, and Xiaowei Huang. Quantifying safety risks of deep neural networks. *Complex & Intelligent Systems*, 9, 07 2022.

[181] Xiao Xu, You Lu, and Qiming Fu. Applying graph neural network in deep reinforcement learning to optimize wireless network routing. pages 218–223, 03 2022.

[182] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 818–833, Cham, 2014. Springer International Publishing.

[183] Jin Zhang and Jingyue Li. Testing and verification of neural-network-based safety-critical control software : A systematic literature review. *Information and Software Technology*, 123 :106296, 2020.

[184] Kai Zhang, Eric Lucet, Julien Alexandre Dit Sandretto, Selma Kchir, and David Filliat. A review of classical and learning based approaches in task and motion planning. In Giuseppina Gini, Henk Nijmeijer, Wolfram Burgard, and Dimitar Filev, editors, *Informatics in Control, Automation and Robotics*, pages 83–99, Cham, 2023. Springer International Publishing.

[185] Xinyu Zhang, Stephane Redon, Minkyoung Lee, and Young J. Kim. Continuous collision detection for articulated models using taylor models and temporal culling. *ACM Trans. Graph.*, 26(3) :15, 2007.

[186] Pu Zheng, Pierre-Brice Wieber, and Olivier Aycard. Online optimal motion generation with guaranteed safety in shared workspace. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9210–9215, 2020.

Annexe A

Description des activités d'enseignement

Cet annexe présente en détail, les enseignements que j'ai pu mener depuis le début de ma carrière. Certains sont des enseignements récurrents d'autres sont plus ponctuels et ont été menés seulement pendant une ou deux années.

A.1 Enseignements récurrents

Chaque année, je suis responsable de plusieurs enseignements que j'ai repris, fait évoluer ou montés.

A.1.1 Automatique :

Depuis ma prise de poste, j'assure les cours de base en automatique aux **GE3** et **GP4**. Ces cours/TD/TP portent sur l'identification, la modélisation et le contrôle des systèmes linéaires une entrée/une sortie dans le domaine continu et dans le domaine discret. J'y présente notamment les fonctions de transfert et transmittance, les méthodes d'identification des systèmes continus du premier et deuxième ordre, les correcteurs continus classiques (PID) ainsi que des correcteurs discrets simples (méthode polynomiale). J'ai repris, fait évoluer et monter de nouveaux TPs mettant en oeuvre des moteurs à courant continu, des régulations de hauteur d'eau et de position d'une bille sur un rail incliné.

A.1.2 Automatique numérique :

Je dispense également, depuis 2019 (et en 2014), un cours avancé d'identification et de commande dans le domaine discret, via les méthodes de ZDAN et celles basées sur les structures de type RST (PID numérique,...) aux **GE3**. Dans cet enseignement, je relie les concepts théoriques à leur implémentation pratique sur microcontrôlleur en proposant l'implémentation d'un régulateur de vitesse adaptatif sur Arduino.

A.1.3 Initiation à ROS :

ROS est un outil de plus en plus utilisé pour la simulation et le contrôle des robots. J'ai donc logiquement proposé d'ajouter cet outil à la formation des étudiants de **GE**. Dans le cadre de ce cours d'initiation, j'introduis, depuis 2018, la notion de noeud, topic, message

et demande aux étudiants de réaliser un programme de sortie de labyrinthe pour un robot mobile. Pour cela j'ai créé un paquet disponible via le lien github. Initialement prévu pour les **GE3**, cette introduction est aujourd'hui présentée également aux étudiants de la prépa **HUAT** [1] ainsi qu'aux étudiants du Master 1 Automatique Robotique. En 2024, cet enseignement a été confié à un intervenant industriel.

A.1.4 C++, CMAKE et versionning :

Depuis 2017, je suis en charge d'une partie des cours de programmation avancée aux **GE4B**. J'y introduis l'outil cmake d'aide à la compilation et à l'installation dans l'objectif de sensibiliser les étudiants au partage et à la pérénnité d'un code. J'aborde également le polymorphisme, le chargement dynamique de librairie ainsi que les licences et le versionning.

A.1.5 Robots manipulateurs et introduction à l'Intelligence Artificielle :

Depuis 2020, ce cours à destination des **GE4A** aborde les bases de la robotique des bras manipulateurs sériels : modélisation, contrôle, planification. A la fin de cet enseignement, les élèves sont confrontés à plusieurs petits challenges de pick and place. Les paquets ROS des challenges sont disponibles sur le lien github, cet enseignement a donné lieu à une publication pédagogique [114]. Depuis 2023, une introduction à l'Intelligence Artificielle est faite dans le cadre du cours par Samuel Beaussant (en tant que doctorant, puis intervenant extérieur).

A.1.6 Introduction à la robotique humanoïde :

Depuis 2020, j'ai l'opportunité de faire une introduction à la robotique humanoïde en Master 2 **PAR**. J'y présente les différences de modélisation entre les robots à pattes et les robots sériels, j'introduis la notion d'équilibre ainsi que la génération de mouvements de marche. J'en profite pour initier les étudiants à la présentation d'article scientifique via des séances de classe inversée.

A.1.7 Projet de Synthèse :

Le département **GE** a une pédagogie orientée vers les projets. Le projet de synthèse est un enseignement que je fais tous les trois ans [2] qui a pour but de donner la possibilité aux **GE3**, d'appliquer les concepts qu'ils ont vus durant leur première année du cursus ingénieur. En 2018 et 2022, le thème était une course de robots mobiles. Vous pouvez trouver une petite vidéo de présentation ici.

A.2 Enseignements ponctuels

En plus de ces enseignements récurrents, j'ai pu assurer quelques enseignements ponctuels en fonction des besoins du département **GE**.

1. Ces étudiants viennent de l'université chinoise *Hubei University of Automotive Technology* et peuvent intégrer l'**ISIMA**, le département **GP**, **GE** et plus récemment **IMDS**.
2. L'objectif est de changer l'équipe encadrante afin de renouveler plus facilement les sujets proposés.

A.2.1 Robotique mobile :

En 2022, j'ai assuré le cours de robotique mobile pour les **GE5A**, en présentant la notion de modèle, de planification et de contrôle des robots mobiles à roues. Les travaux pratiques ont été menés sur le robot Limo et j'ai proposé un environnement de simulation disponible ici.

A.2.2 Réseaux et html

En 2012-2013, j'ai eu l'occasion de donner quelques cours de réseaux et html aux étudiants de première et deuxième année du DUT Réseaux et Télécoms de l'IUT de Beziers.

A.3 Projets

Le département **GE** porte une grande importance à la formation de ses étudiants via des mises en situation concrètes à travers des projets industriels. Dès mon arrivée à Polytech Clermont, je me suis impliqué dans le cadre des projets à plusieurs niveaux.

A.3.1 Projet industriel en GE :

Ces projets se déroulent en binôme entre la quatrième année (∼50h) et la cinquième année (∼150h). L'encadrement des projets peut regrouper plusieurs aspects selon les projets : client (donne un sujet), tuteur technique (aide technique) et tuteur académique (suivi de projet). L'enseignement des projets s'adapte régulièrement aux attentes des industriels et aux profils des étudiants qui évoluent. Ayant contribué à l'évolution des projets ces dernières années, nous avons publié leur fonctionnement actuel dans [101].

En plus des projets industriels où j'avais le rôle de tuteur, voici quelques exemples de sujets que j'ai encadrés en tant que client :

— Pepper fait du labyrinthe à bille : L'objectif est de rendre le robot Pepper capable de se saisir d'un labyrinthe à billes et de le résoudre en dirigeant la bille vers la sortie,

— Robot guide : Dans un environnement structuré, un robot mobile (Pepper ou Limo) doit trouver une personne spécifiée et la guider jusqu'à l'interlocuteur.

— PLAMOBO : PLAteforme MOdulaire pour les roBOts : Ce projet vise à développer une plateforme générique pour la conception d'un robot et la réalisation de tâches à partir d'élément de base (capteurs, actionneurs, ...) qui soit compatible avec ROS.

— Robot dessinateur : L'objectif est de permettre à un robot de dessiner une image pouvant venir de la caméra (dessin d'un portrait) ou d'une banque de données.

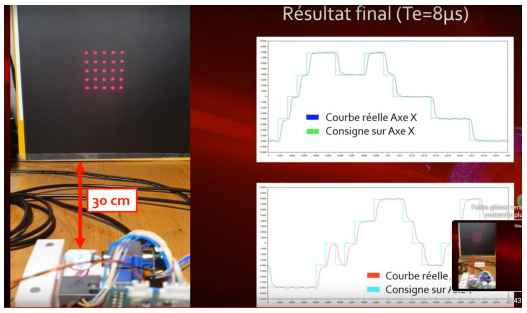
La liste complète des projets encadrés est disponible ici et quelques projets sont illustrés sur la Figure A.1. Dans le cadre de l'évolution de cet enseignement et afin d'accroître la visibilité de nos activités dans le département **GE**, j'ai proposé la réalisation de courtes vidéos en fin de projet. La production des élèves est disponible ici.

A.3.2 Sous-traitance :

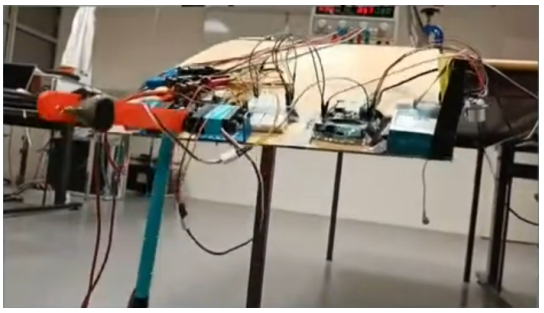
La notion de sous-traitance est de plus en plus importante dans la mission d'un ingénieur. Dans le cadre des projets industriels, les étudiants de cinquième année peuvent

(a) Trieur de M&M's

(b) Labyrinthe à billes



(c) Asservissement Laser



(d) Compensation des oscillations

FIGURE A.1 – Autres exemples de projet étudiants.

sous-traiter certaines tâches aux étudiants de quatrième année. Cet enseignement permet aux **GE5** de spécifier efficacement leur besoin et aux **GE4** d'apprendre l'importance de fournir un résultat de qualité afin d'avoir de nouvelles demandes. Je co-encadre cette activité pédagogique pour laquelle nous avons adapté la méthode agile [3] en affectant les étudiants directeurs de produit ou développeur.

A.3.3 Autres projets

J'ai également encadré des projets pour des étudiants de **IMDS** ou de l'ISIMA en lien avec les activités de recherche sur l'estimation d'extremum de fonction par fonctions de base, comme présenté dans le Chapitre 4 et également des sujets de **PEIP** sur des sujets plus génériques (Google peut-il devenir Skynet ?, Comment communiquerait la Résistance aujourd'hui ?, l'inversion des pôles magnétiques, ...).

A.4 Enseignements à l'étranger

A.4.1 Djibouti :

Suite aux compétences que j'ai acquises par l'acquisition du robot Pepper et principalement de sa programmation sous ROS, l'université de Djibouti m'a invité (en 2018 et 2019) à assurer un enseignement d'une semaine aux étudiants de LAEER (Licence Appliquée en Énergétique et Énergies Renouvelables) à l'Institut Universitaire de Technologie Industrielle. J'en ai également profité pour former les enseignants qui ont pu reprendre cet enseignement depuis.

3. Présentation de la méthode agile ici

A.4.2 Norvège :

En octobre 2022, dans le cadre de la mise en place d'une collaboration avec l'Université arctique de Norvège, j'ai pu faire deux séances de cours concernant les concepts d'équilibre des robots humanoïdes appliqués à l'homme (issues de mon cours sur la robotique humanoïde du Master) ainsi que sur le contrôle des systèmes complexes appliqué à un galvanomètre pour l'asservissement de visée laser en tant que retour d'expérience des activités de projet.

Annexe B

Sélection d'articles

Les articles suivants illustrent, en détails, certains travaux de recherche présentés dans ce manuscrit :

— 2013 : **S. Lengagne**, Joris Vaillant, and Eiichi Yoshida. *Generation of Whole-body Optimal Dynamic Multi-Contact Motions*. The International Journal of Robotics Research, page 17, April 2013.

— 2024 : **S. Lengagne**. *Comparison of several basis functions for interval-based optimization*, soumis à International Journal of Approximate Reasoning.

— 2024 : S. Beaussant, **S. Lengagne**, B. Thuilot, O. Stasse. *Towards Zero-Shot Cross-Agent Transfer Learning via Latent-Space Universal Notice Network.* soumis à Robotics and Autonomous Systems.

B.1 Génération de mouvements complexes

Article : **S. Lengagne**, Joris Vaillant, and Eiichi Yoshida. *Generation of Whole-body Optimal Dynamic Multi-Contact Motions*. The International Journal of Robotics Research, page 17, April 2013.

Generation of Whole-body Optimal Dynamic Multi-Contact Motions

Sébastien Lengagne[*,†] **Joris Vaillant**[*] **Eiichi Yoshida**[*]
Abderrahmane Kheddar[*,‡]

Abstract: We propose a method to plan optimal whole-body dynamic motion in multi-contact non-gaited transitions. Using a B-Spline time-parameterization for the active joints, we turn the motion-planning problem into a semi-infinite programming formulation that is solved by non-linear optimization techniques. Our main contribution lies in producing constraint-satisfaction guaranteed motions for any time-grid. Indeed, we use Taylor series expansion to approximate the dynamic and kinematic models over fixed successive time-intervals, and transform the problem (constraints and cost functions) into time-polynomials which coefficients are function of the optimization variables. The evaluation of the constraints turns then into computation of extrema (over each time-interval) that are given to the solver. We also account for collisions and self-collisions constraints that have not a closed-form expression over the time. We address the problem of the balance within the optimization problem and demonstrate that generating whole-body multi-contact dynamic motion for complex tasks is possible and can be tractable, although still time consuming. We discuss thoroughly the planning of a sitting motion with the HRP-2 humanoid robot and assess our method with several other complex scenarios.

Keywords and phrases: Multi-contact motions, optimization, balance, Taylor series approximation, time-interval discretization, humanoid robots.

Introduction

Planning dynamic motion for complex robotic systems such as humanoids, under various constraints is known to be a complex problem in robotic research. For example, locomotion in cumbersome environments is non-gaited and approaches developed for biped cyclic locomotion do not apply. We propose a method that can generate gaited or non-gaited, optimal and dynamic multi-contact motions of the robotic system (in the sense of integrating a preview of what the following support contacts are).

Classical approaches usually solve the problem in two steps: the first step plans the motion of a reduced simple model [25] that encompasses the main

[*]CNRS-AIST Joint Robotics Laboratory (JRL), UMI3218/CRT, Tsukuba, Japan.
[†]Karlsruhe Institute of Technology, Institute for Anthropomatics, Humanoids and Intelligence Systems Lab, Adenauerring 2, 76131 Karlsruhe, Germany.
[‡]CNRS-Université Montpellier 2, LIRMM, Interactive Digital Human group, 161 rue Ada, F-34095, Montpellier cedex 5, France.

invariants of the motion (center of mass, centers of pressure, etc.); the second step consists in generating the whole body dynamic motion that tracks at best the first step plan but where additional constraints (joint limits, torque limits, collision detection...) are checked a posteriori. Hence, the first step assumes viability of the second step. In some typical tasks (e.g. walking) precaution are taken for the first step planning to generate very likely feasible plans. However, for more general tasks such as moving in cumbersome situations it is hard to both plan based on reduced/simple models and make plausible assumptions on the feasibility of the plan for the whole-body motion.

In this paper, we solve whole-body planning using non-linear optimization techniques and a full body dynamic model. Optimization techniques were used as early as the 1980th in robotic motion planning. The starting point of our work is the result obtained in [30]; Lee et al. proposed a method to generate motions for robots with redundant or exactly actuated closed chains. This method is based on the optimization of the parameters of the joint trajectories and computes analytically the model of the robot and its derivative. Their algorithm was illustrated with 2D simulation examples and we considered whether it is possible to extend the approach for the 3D complex case of multi-contact motions for humanoid robots. That is, how to impose unilateral constraints on the contact forces, ensure balance, include collision avoidance and guarantee constraint satisfaction not only at sample time but also all along the motion whatever time-grid size is chosen? Our present work tackles these issues. The work by [30] is certainly the root of our approach. However, we propose several novel contributions in order to generate complex and dynamic multi-contact motions for humanoid robots.

In Section 1, we present the related work about motion generation for humanoid robots and emphasize the lack of efficient methods to generate complex and dynamic multi-contact motions. Then, we present the mathematical model of the robot and the analytical formulation of the contact forces in Section 2. Section 3 is about formulating the motion generation problem as a non-linear optimization. To solve this problem and produce motions with guaranteed constraint satisfaction all over the time, including non-desired collisions avoidance, Section 4 introduces our second contribution: the time-interval discretization. This discretization is based on Taylor series expansion approximation for the continuous state variables of the robot. Finally, in Section 5, we discuss the technical details and validate our method with several complex scenarios such as a sitting motion for the HRP-2 [27], where both hands, both feet and the waist are involved in contact transitions during motion.

1. Related work

Previous work proposed several approaches in order to generate sequences of contact stances, to compute optimal motions, to characterize its balance and to perform experiments of multi-contact motions.

Recently, there is an increasing interest in planning non-gaited motion where a humanoid robot is allowed to contact the environment with any possible part of its body to support motions in bulky and uneven spaces. For example, [6, 7, 13, 14, 20, 21] developed a contact-before-motion planning algorithms to plan sequence of contacts for such problems. Complex examples were illustrated on the HRP-2 robot [13, 15]. However, in these experiments, the motion between successive contacts, namely the transitions, was not fully dynamic. Our present contribution aims at filling this gap. That is, generating optimal dynamic motions in multi-contact transitions.

Optimal control methods are used to generate (periodic) complex motions such as one-legged hopping, walking [39] and running [45]. They are usually categorized into three different types [12]: (i) dynamic programming methods are based on the principle that the optimality of sub-arcs is restricted to small state dimensions; (ii) indirect methods use the Pontryagin's maximum principle, but have a small domain of convergence; (iii) direct methods that transform the infinite problem into a finite Nonlinear Programming Problem (NLP) are commonly used for real systems such as humanoid robots.

The direct methods are composed of the direct single shooting, collocation and direct multiple shooting methods for which the control and/or the state variables are discretized and use the direct dynamic model of the robot. A fourth direct method uses parameterization. It finds a local solution of the optimal control through optimal values of the trajectories' parameters; as an example of parameterization B-Splines are widely used, see for example [30, 38]. Parameterization proved to be particularly efficient in robotics [47, 49]. In this case, inverse dynamic model can be used that allows decomposing the motion into several independent parts, which can be parallelizing using multi-thread software. Using state-of-the-art parameterization based methods, continuous time-dependent constraints are still discretized into a finite set of discrete constraints over a time-grid. This kind of method was used in [1] to realize the lifting motion of 23.4kg with the HRP-2 humanoid [27]. Each time an experiment crashed during trials, investigation of log data revealed that the torque constraint was violated in-between two points of the chosen time-grid, albeit the optimization process ended successfully. In that sense, this method is not safe, since the user cannot trust the results of the optimization process. At that time, the optimization problem was reformulated with smaller time-grid, recomputed and re-experimented until success.

This observation motivates the work we did and present here. Our new formulation, guarantees that, under model-match hypothesis, in case of suc-

cessful termination of the optimization process, the planned motion will not violate any constraints all along the time-motion whatever discretization range is. This is the reason why we call it safe. One can use the time-interval discretization proposed in [37] but at the price of a substantially much longer computation time. We can also proceed in a recursive way by running the optimization using a time-grid, and then compute the violation of the constraints using the time-interval while penalizing the bounds of the constraints until no violation occurs as presented in [37]. Yet, those methods cannot deal with continuous equality constraints that describe the constant position of a body during a contact stance.

Specific to humanoids robots, balance is a very challenging and a difficult issue. During walking motions, the balance of the robot is usually characterized by the Zero Moment Point (ZMP) [52]. Unfortunately, the use of the ZMP assumes that there are only co-planar contacts and cannot apply to general multi-contact motions. One can also consider the Generalized-ZMP (GZMP) [19] or the Foot Rotation Index [18]. To characterize the balance of the robot, we rather monitor if the Contact Wrench Sum (CWS), due to gravity and dynamic effects, remains in the Contact Wrench Cone (CWC) as presented in [23]. From this definition of the balance, we propose an analytical formulation of the contact forces.

The use of reduced models during the optimization can lead to some constraints violation a posteriori. We better plan the motion with the most complete model and constraints possible in order to guarantee that whole-body motion is highly likely feasible. The choice of the full-body model impacts on the complexity of the computation, and hence on the computation time; but is the only way to ensure a priori a viable plan of the robot.

Collisions with the environment and the self-collisions are also critical constraints to handle during motion planning. However, collision avoidance requires the evaluation of the distance to obstacle. In order to have good performances of the optimization process, it is highly suggested to consider convex bounding shapes to describe the bodies of the robot such as the Sphere-Torus-Patch Bounding Volume (STP-BV) presented in [16]. This solution allows having continuous gradient of the distances and we are using it in our problem formulation. To ensure the collision avoidance over the entire motion's duration, some methods such as the Continuous Collision Detection [54] use Taylor models and temporal culling.

We assume a perfect full-body dynamic rigid model of the HRP-2 robot and a perfect knowledge of its environment. However, to perform real experiments, the last issue is subject to the control approach of the robot during multi-contact stances. Work in [24, 48] proposed balance controller based on the control of the forces, whereas [29] is based on the control of both linear and angular momentum of the robot. Interestingly [46] presents a methodology to control the internal forces and Center Of Mass (COM) during a multi-contact interaction between a humanoid robot and the environment

using a virtual linkage model. This method was validated in simulations of manipulation tasks without locomotion, such as cleaning a board or manipulating a panel. Eventually, a controller used during locomotion of robots is presented in [44]; it uses the redundancy of the multi-contact motion to create optimal distribution of the contact force and was successfully applied to a quadruped robot during actual experiments. In our work, we assume that the robot's controller tracks the generated joint trajectories perfectly under little model uncertainties.

2. Multi-contact motion definition

In this section, we present the model and the constraints that are needed to generate a multi-contact motion.

2.1. Kinematics

First, we deal with the kinematic constraints. A contact is defined simply as the relative configuration of a given robot's link regarding the environment, excluding sliding. This can be defined with equality constraints involving two Cartesian frames: one on the contact body $\mathbf{X}_i$ and the other one on the environment at the expected desired position of the contact $\mathbf{X}_i^e$ (Figure 1). Where $\mathbf{X} \in \mathbb{R}^6$ denotes the six-component-contact-vector of a frame position and orientation ($\mathbf{X} = [x, y, z, \theta_x, \theta_y, \theta_z]^T$). In order to consider N_c contacts during a time interval $[\Delta t]$, we must ensure a set of n_i continuous equality constraint for each contacts as follows:

$$\forall t \in [\Delta t], \forall i \in \{1, \dots, N_c\}, \forall j \in \{1, \dots, n_i\} \quad \text{kin}_{i,j}(\mathbf{X}_i(t), \mathbf{X}_i^e) = 0 \tag{2.1}$$

where n_i is the number of constraints for each type of contact. For a planar rigid contact there are three rotations and three positions constraints ($n_i = 6$). The general formulation of Equation 2.1 allows to consider planar, edge or point contacts. Moreover, we consider fully and partially defined contacts: the value of some components of the contact position and/or orientation can be unknown (but must stay constant). For example, for a planar rigid contact, one considers:

$$\forall t \in [\Delta t], \forall i \in \{1, \dots, N_c\}, \forall j \in \{1, \dots, 6\} \quad \begin{cases} \text{if } \kappa_{i,j} = 1 \rightarrow & \mathbf{X}_{i,j}(t) = \mathbf{X}_{i,j}^e \\ \text{if } \kappa_{i,j} = 0 \rightarrow & \frac{d}{dt}\mathbf{X}_{i,j}(t) = 0 \end{cases} \tag{2.2}$$

where $\kappa_{i,j}$ is a Boolean attached to the j^{th} component of the i^{th} contact vector. This Boolean is equal to 1 if the component value is known and is equal to 0 if the component value is unknown, in the case of partially defined contacts. In the latter case, the actual value will be a result of the optimization process. Equations (2.1) will impose the kinematic behavior of the motion. The balance of the motion will depend on the dynamic of the motion as defined hereafter.

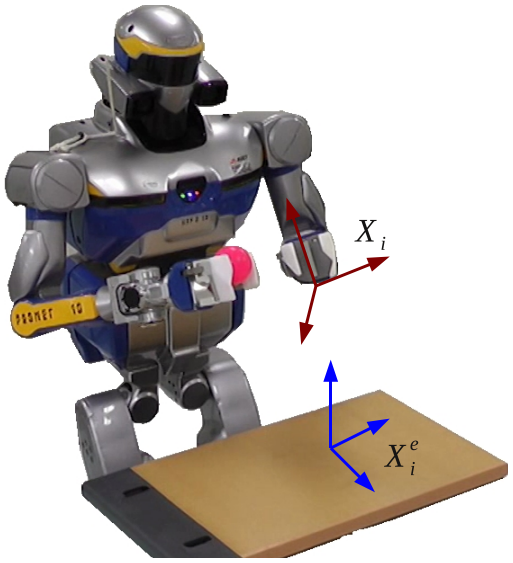


FIG 1. *Description of the contact of the left hand with the table. The frame* $\mathbf{X}_i$ *is linked to the body and must coincide the expected frame* $\mathbf{X}_i^e$ *on the table.*

2.2. Inverse dynamic model

We plan motion with the full-body dynamic model of the humanoid robot:

$$\begin{bmatrix} \Gamma \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{M}_1(q) \\ \mathbf{M}_2(q) \end{bmatrix} \ddot{q} + \begin{bmatrix} \mathbb{H}_1(q,\dot{q}) \\ \mathbb{H}_2(q,\dot{q}) \end{bmatrix} + \begin{bmatrix} \mathbf{J}_1^T(q) \\ \mathbf{J}_2^T(q) \end{bmatrix} \mathbb{F} \tag{2.3}$$

where, $q(t) \in \mathbb{R}^{N_{dof}}$ is a vector containing the N_{dof} joint position $(q_i(t))$, $\Gamma(t) \in \mathbb{R}^{N_{dof}}$ is the vector of the joint torques, $\mathbb{F}(t)$ is the vector of the N_f active linear contact forces, $\mathbf{M}_1 \in \mathbb{R}^{N_{dof} \times N_{dof}}$ $\mathbf{M}_2 \in \mathbb{R}^{6 \times N_{dof}}$ are the two components of the inertial matrix, $\mathbb{H}_1 \in \mathbb{R}^{N_{dof}}$ and $\mathbb{H}_2 \in \mathbb{R}^6$ the two components of vector due to gravity, centrifugal and Coriolis effects, $\mathbf{J}_1 \in \mathbb{R}^{N_{dof} \times 3N_f}$ and $\mathbf{J}_2 \in \mathbb{R}^{6 \times 3N_f}$ are the components of the Jacobian matrix. Equation (2.3) assumes that the first contact (from which the waist trajectory is computed, see [36]) is always maintained. This assumption is enforced by adding constraints on the balance of the robot as presented in Section 2.6. Equation (2.3) emphasizes the link between the contact forces and the joint torques. To know the joint torques, it is necessary to know the joint trajectories and the contact forces. Regarding the known joint trajectories, there is an infinite set of combination joint torques–contact forces, depending on the internal forces. In the following, we present our method to get rid of this indetermination and compute the joint torques and contact forces from the joint trajectories and a few additional variables, in a way that enforces the balance of the robot.

2.3. Balance

To characterize the balance of the motion, we constrain the Contact Wrench Sum, due to gravity and dynamic effects, to remain within the Contact Wrench Cone as presented in [23]. In fact, we ensure that the contact forces that counterparts the dynamic effect $\mathbb{D}_2 = \mathbf{M}_2(q)\ddot{q} + \mathbb{H}_2(q, \dot{q})$ must be feasible; i.e., hold a desired contact state, prohibiting from any undesired sliding of the link on the surface or contact removal.

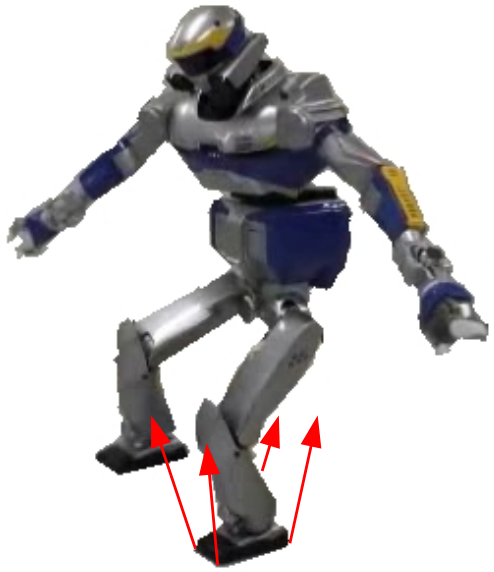


FIG 2. *Description of the contact forces for the left foot. We consider four forces: one for each corner of the foot sole.*

We define a set of N_f linear forces $\mathbb{F} = \{F_1, F_2, \ldots, F_{N_f}\}(F_i \in \mathbb{R}^3)$, for all the contacting bodies, as presented in Figure 2 and their normal and tangent spaces. To avoid unexpected sliding or taking-off of any part of the robot, each linear contact force F_i must fulfill:

$$\forall t \in [\Delta t], \forall i \in \{1, \ldots, N_f\} \quad \begin{cases} F_i^n(t) > 0 \\ ||F_i^t(t)||^2 \leq \mu_i^2 F_i^n(t)^2 \end{cases} \tag{2.4}$$

where F_i^n is the normal component of the contact force, F_i^t is the 2D vector of the tangent force and μ_i is the friction coefficient at contact i. For each contact forces i, those two constraints can be merge into the equivalent equation:

$$\text{fric}(F_i(t)) = F_{i,x}(t)^2 + F_{i,y}(t)^2 - \text{sign}(F_{i,z}) \times \mu_i^2 F_{i,z}(t)^2 \leq 0 \tag{2.5}$$

Note that we consider the z-axis as the normal axis at the contact space. Moreover, those contact forces must counterparts dynamic effect by satisfying the second part of the dynamic Equation (2.3), that is:

$$\mathbb{D}_2(q, \dot{q}, \ddot{q}) + \mathbf{J}_2^T(q)\mathbb{F} = 0 \tag{2.6}$$

2.4. Contact forces

In our previous work [36], a 2D-model study of this problem revealed that Equation (2.6) cannot be satisfied if the contact forces are parameterized using B-Spline functions. Hence, we compute the value of the contact forces $\mathbb{F} = \{F_1, F_2, \ldots, F_{N_f}\}$ from the joint trajectories that exactly counterpart the dynamics effects while fulfilling the constraint of Eq. (2.4) as best as possible, i.e. the contact forces that solve:

$$\begin{gathered} \min \tfrac{1}{2} \textstyle\sum_i \beta_i(\alpha_i ||F_i^t||^2 + F_i^{n2}) \\ \mathbb{D}_2(q, \dot{q}, \ddot{q}) + \mathbf{J}_2^T(q)\mathbb{F} = 0 \end{gathered} \tag{2.7}$$

where F_i^n is the normal component of the contact force F_i; F_i^t the 2D vector component of the tangent component of the contact force F_i; β_i and α_i are weight coefficients as detailed in Appendix B. The exact satisfaction of these constraints is left to the optimization solver. The Appendix B show how to obtain the analytical formulation of the contact forces such as:

$$F_i = \mathbb{W}_i^{-1} \left[\hat{P}_i A_i \; A_i\right] \Omega^{-1} \mathbb{D}_2 \tag{2.8}$$

with:

$$\Omega = \sum_i \left(\begin{bmatrix} \hat{P}_i A_i \\ A_i \end{bmatrix} \mathbb{W}_i^{-1} \begin{bmatrix} \hat{P}_i A_i & A_i \end{bmatrix} \right) \tag{2.9}$$

and:

- $\mathbb{W}_i = \mathrm{diag}(\beta_i\alpha_i, \beta_i\alpha_i, \beta_i)$ is a weight matrix,
- $\beta_i > 0$ weights the contact forces i regarding the other forces,
- $\alpha_i \gg 1$ weights the tangential components regarding the tangential ones,
- P_i, A_i are the position and orientation of the contact forces.

The matrix Ω is a 6×6 matrix that we invert using the Gauss-Jordan algorithm, to compute the value of the contact forces $\mathbb{F}$.

2.5. Collision avoidance

During multi-contact motion transitions, there are parts of the robot for which we seek a sustained collision with the environment (desired motion

supporting contacts), and the remaining part that should avoid (non desired) collisions. Therefore, planning motion considers collision avoidance as a constraint. Let $\mathcal{L}_i$ be the i-th link of the robot, $i \in [1, N_l]$ for a robot having N_l identified links or all the sub-part that are convex composing all links, and let $\mathcal{O}_i$ be a set obstacles among N_o ones. Avoiding collision can be simply written as:

$$d(\mathcal{L}_i, \mathcal{O}_j)(t) \geq \epsilon \tag{2.10}$$

where $d(X, Y)(t)$ is the distance separating body X from body Y, the pair i, j is the list of the defined links/obstacles to be checked and ϵ is a security margin. This constraint means that the separating distance between two any body must be kept positive (at ϵ margin) all along the motion. Self-collisions are enforced similarly:

$$d(\mathcal{L}_i, \mathcal{L}_j)(t) \geq \epsilon \;\; i \neq j \text{ and } j \neq i - 1 \tag{2.11}$$

There are several methods to compute the distance among pairs of geometrical objects. The simple the geometry the faster is the computation. This is the reason why in robotics we use bounding volume for the links (e.g. capsules, oriented bounding boxes, etc.). Since we are using the distance in the frame of an optimization problem, many solvers require computing the gradients of the constraints. In order to fulfill existence and continuity of the distance function gradients, while keeping fast computation, we need to have strictly convex bounding volume of the links. We devised in our previous papers such new bounding volume operator, called STP-BV that achieved bulged convex hulls so as the distance computation is fast and its gradient C^1 continuous [2, 16]. Notice that computing the distance δ, is the outcome of a local optimization process.

2.6. Constraints

We define a multi-contact motion by a sequence of multi-contact phases. One phase is an amount of time for which all the contacts hold (no creation or release of contacts). The transition between two phases is done when at least one contact is created or released.

The generation of the multi-contact motions is equivalent to finding the joint trajectories, the contact forces and the duration over each contact phase. This motion must validate a set of equality (Eq. 2.12a) and inequality Eq. (2.12b-2.12f) continuous constraints:

$$\forall t \in [\Delta t], \forall i \in \{1, \ldots, N_c\}, \forall j \in \{1, \ldots, n_i\} \;\; \text{kin}_{i,j}(\mathbf{X}_i(t), \mathbf{X}_i^e) = 0 \tag{2.12a}$$

$$\forall t \;\; \in [0, T], \forall i \;\; \in \{1, \ldots, N_{dof}\} \;\; \underline{q_j} \leq q_j(t) \leq \overline{q}_j \tag{2.12b}$$

$$\forall t \;\; \in [0, T], \forall i \;\; \in \{1, \ldots, N_{dof}\} \;\; \underline{\dot{q}_j} \leq \dot{q}_j(t) \leq \overline{\dot{q}_j} \tag{2.12c}$$

$$\forall t \in [0,T], \forall i \in \{1,\dots,N_{dof}\} \quad \underline{\Gamma}_j \leq \Gamma_j(t) \leq \overline{\Gamma}_j \tag{2.12d}$$

$$\forall t \in [0,T], \forall i \in \{1,\dots,N_f\} \quad \text{fric}(F_i(t)) \leq 0 \tag{2.12e}$$

$$\forall t \in [0,T], \forall i \in \{1,\cdots,N_d\} \quad \delta_i \geq \epsilon \tag{2.12f}$$

where $\delta_i = \text{sign}(d_i)d_i^2$ is the signed square distance between two bodies, this value is returned by the algorithm presented in [2]. One can also consider a set of k discrete constraints on any state x_k of the robot in order to specify desired property of the motion.

$$\forall k \quad z_k(x_k, t_k) \leq 0 \tag{2.13}$$

3. Problem formulation

In the previous Section, we detailed the model and the constraints of a multi-contact motion. Now, we introduce the basics in terms of terminology, notations, and the general formulation of the non-linear optimization problem.

3.1. Motion planning problem

The motion generation process solves the following optimization problem:

$$\begin{aligned} \min_{q(t),\Gamma(t),\mathbb{F}(t),\mathbf{T}_f} \quad & C(q(t),\dot{q}(t),\ddot{q}(t),\Gamma(t),\mathbb{F},\mathbf{T}_f) \\ \text{subject to} \quad & \\ \{\Gamma,\mathbb{F}\} &= \text{IDM}(q,\dot{q},\ddot{q}) \\ c_{eq}(q,\dot{q},\ddot{q},\Gamma,\mathbb{F}) &= 0 \\ c_{ineq}(q,\dot{q},\ddot{q},\Gamma,\mathbb{F}) &\leq 0 \\ c_{teq}(q(t_d),\dot{q}(t_d),\ddot{q}(t_d),\Gamma(t_d),\mathbb{F}(t_d)) &\leq 0 \end{aligned} \tag{3.1}$$

This formalism is used in [38]. Usually, the constraint $\{\Gamma,\mathbb{F}\} = \text{IDM}(q,\dot{q},\ddot{q})$ is replaced by the direct dynamic model $\dot{x}(t) = f(x(t),u(t))$ where $x(t)$ is the state variable and $u(t)$ the control variables [45], but both representations are mathematically equivalent and differ only from the implementation viewpoint. Thus, the optimal control problem (3.1) can be considered as an Infinite Programming (IP) problem that consists in finding the joint trajectories $q(t)$, the force coefficient β and the phase durations $\mathbf{T}_f$:

$$\begin{aligned} \underset{q(t),\beta,T_f}{\text{argmin}} \quad & C(q(t),\beta,\mathbf{T}_f) \\ \forall i, \forall t \in [\Delta_i] \quad & g_i(q(t),\beta,\mathbf{T}_f) \leq 0 \\ \forall j, \forall t \in [\Delta_j] \quad & h_j(q(t),\beta,\mathbf{T}_f) = 0 \\ \forall t_k \in \{t_1,t_2,\dots\} \quad & z_k(q(t_k),\beta,\mathbf{T}_f) \leq 0 \end{aligned} \tag{3.2}$$

where C is the cost function, which accounts for quantitative and/or qualitative robot motion in given application contexts. Well known cost functions

in robotics are: motion duration [41, 5], minimum torque [8], global energy consumption [38], jerk for smooth motion [42], torque change [50], or any weighted combination of the above and more. In the equation, g, h and z are respectively the continuous inequality and equality and the discrete inequality constraints as presented in Section 2.6.

We voluntary omit the discrete constraints z, since they do not lead to any kind of problem in the state-of-the-art methods.

3.2. Semi-Infinite Programming

The problem (3.2) is an Infinite Programming (IP) since the solution to be found is a set of continuous functions $q(t)$ that fulfill a set of continuous constraints all along the time t (both can be seen as infinite sets of discrete values). In order to make an IP computationally solvable, most of the methods [43] reduce such complexity by defining a parameters set $\mathbf{P} \in \mathbb{R}^N$ that gives at least a parametric shape for the functions to be found (the optimal trajectories $q(t)$).

$$q(t) = f(\mathbf{P}, t) \tag{3.3}$$

Based on prior work in robotics, e.g. [30], we chose to shape the joint trajectories with clamped uniform cubic B-Splines curves. A B-Spline function is the weighted sum of $k-$order basis functions defined by m number of control points:

$$\forall t \in [\Delta t] \quad q(t) = \sum_{i=1}^{m} b_i^k(t) p_i \tag{3.4}$$

The basis-functions $b_i^k(t)$ are computed using the Cox-de-Boor recursion [11] and the control points ρ_i are part of the parameters of the motion. This parameterization was already used for motion generation in [30, 38]. Subsequently, the motion planning problem turns to finding the best parameter set $\mathbf{P} \in \mathbb{R}^N$ such as:

$$\begin{array}{ll} \underset{\mathbf{P}}{\operatorname{argmin}} & C(\mathbf{P}) \\ \forall i, \forall t \in [\Delta_i] & g_i(\mathbf{P}, t) \leq 0 \\ \forall j, \forall t \in [\Delta_j] & h_j(\mathbf{P}, t) = 0 \end{array} \tag{3.5}$$

This problem is a Semi-Infinite Programming (SIP) [43] since it deals with a finite set of optimization variables $\mathbf{P} \in \mathbb{R}^N$, which must satisfy a set of continuous constraints. The detail of the optimization parameter vector $\mathbf{P}$ in our case in presented in Section 5.1.

4. Dealing with continuous constraints

Enforcing joint position and velocity limits is straightforward, see Appendix C. For more complex constraints such as torques, we use a time-interval discretization based on a polynomial approximation.

4.1. Time-interval discretization

Most of the solvers cannot deal with continuous constraints and require a finite number of constraints. Although the time-grid discretization is widely used [22, 43, 51], it assumes (but does not guarantee) that if a constraint holds at times t_k and t_{k+1}, then it holds $\forall t \in [t_k, t_{k+1}]$ [37]. We experienced in our previous attempts [1, 37] and highlight that violations of the limits may happen, even when the optimization algorithm ended with success status. To solve this issue and generate motions in a safe way, the time-interval discretization [37] replaces the continuous inequality constraints of the optimization problem by:

$$\forall i, \forall [t] \in \mathbb{IT} \quad \sup([g]_i(\mathbf{P}, [t])) \leq 0 \tag{4.1}$$

where $\mathbb{IT} = \{[t]_1, [t]_2, \cdots, [t]_{q-1}, [t]_q\}$ and $[t]_n = [t_{n-1}, t_n]$ is the time interval decomposition of the phase Δt. In [37], we compute a conservative value of the extrema over each time interval $[t]_i$, and the optimization solver can produce a motion that guarantees all the constraint validity over the entire motion duration. Unfortunately, this method has a huge computation time. In this paper, we consider the same time-interval discretization, but we rather use Taylor approximation, which reduces the computation time substantially and allows us to deal with continuous inequality and equality constraints.

4.2. Taylor approximation

4.2.1. Principle

In order to perform a time-interval discretization (Eq. (4.1)) that considers continuous inequality and equality constraints, we use a polynomial Taylor approximation of the constraints. Taylor expansion was also used in [54] to tie the bounds of the bounding box hierarchies for collision detection.

The original computations [37], based on Interval Analysis, do not keep track of the correlation between the different sub-functions which leads to an overestimation of the bounds [54]. To overcome this drawback, we define each function over a time interval $\Delta t = [t_s, t_e]$ as a $n-$order polynomial function:

$$\forall t \in [\Delta t] \; f(t) = \sum_{i=0}^{n} a_i(\mathbf{P}) \times t^i \tag{4.2}$$

where n is the order of the approximation, $\{a_0(\mathbf{P}), a_1(\mathbf{P}), \ldots, a_n(\mathbf{P})\} \in \mathbb{R}^{n+1}$ are the coefficient of the polynomial depending on the optimization parameters $\mathbf{P}$. Compared to previous work [4], we voluntary omit to compute the remaining error interval $[\epsilon]$, assuming that we choose the order n of the polynomial approximation so that we can neglect it [35].

4.2.2. Inequality

Since we get a polynomial function of the constraints (cf. Eq. (4.2)), we want to know the extrema of this polynomial over the considered among of time $[\Delta t]$. Some work approximate the extrema of a polynomial based on the Inequality of the Means [10], but cannot find it for a given interval. Several methods find the extrema based on polynomial root finding of the derivatives of the original polynomial. Analytical approaches fail for polynomial having an order $n > 5$. In order to evaluate this extrema for any order n, we use the property of the B-Splines explained in Appendix C. That is, given a polynomial, the function $f(t)$, and hence its extrema, are contained within the convex hull of the equivalent control points ρ_i. Therefore, we get:

$$\forall t \in [\Delta_t] \ \underline{f} \leq f(t) \leq \overline{f} \implies \forall i \ \underline{f} \leq \rho_i \leq \overline{f} \tag{4.3}$$

With:

$$[\rho_0, \rho_1, \ldots, \rho_n] = [a_0, a_1, \ldots, a_n] \times \mathbf{B}^{-1} \tag{4.4}$$

and where $\mathbf{B} \in \mathbb{R}^{(n+1)\times(n+1)}$ is a square matrix of the polynomial parameters of the n-order B-Splines basis functions[1] $b_i^n(t)$ and relies on the time interval $[\Delta t]$. Using a time-scaled implementation of the polynomial, we can compute the inverse $\mathbf{B}^{-1}$ only once and use it all along the optimization process. As for the polynomial coefficient, the equivalent B-Splines control point ρ_i rely on the optimization parameter: $\rho_i(\mathbf{P})$. Eventually, constraining the equivalent control point ρ_i to remain within two bounds enforces the continuous function to stay within those bounds.

4.2.3. Equality

As shown in Section 2.6, multi-contact motions involve equality constraints that must hold all along the motion, or at some time intervals in order to define fully or partially the contacts. These constraints are the $h_j(\mathbf{P}, t)$ in Eq. (3.5). Using Taylor approximations of h_j, we end up with the polynomial of Eq. (4.2) and impose the coefficients $a_i(\mathbf{P})$ to fulfill the constraint:

$$\forall i \in \{0, \cdots, N_e\} \ \ a_i(\mathbf{P}) = 0 \tag{4.5}$$

The continuous equality constraint being of order n, $N_e \leq n$ is the order that we set for the solver to avoid an over-constrained formulation. As a result, the equality constraint is fulfilled only under small variation tolerance. The compromise between the precision and the optimization convergence rate is tuned heuristically with the choice of $N_e = 2$. Actually, this highlights a deeper fundamental problem linked to the choice of the joint trajectory parameterization for closed kinematics chains. There is certainly a venue for a deeper study specific to this issue.

[1] Note that the n-order basis functions used to evaluate extrema are different from the cubic basis function used to define the joint trajectories in Equation (3.4).

4.3. Evaluation of collision avoidance constraints

We need to compute the minimum of the distance constraint for a range of time intervals (e.g. for $t \in [t_k, t_{k+1}]$). Contrarily to the other constraints, the distance function does not have a closed-form formula, neither an approximate one that allows it to be written as a polynomial in terms of the optimization variables. We therefore need to approach the evaluation of $\min_{t\in[t_k,t_{k+1}]} \delta(t)$ with a numerical method.

In our recent work [31], we devised a fast method, which evaluates the min distance along a given time interval. It combines the well-known Golden search method with the conservative advancement technique used in computer graphics simulation to find the first time of collision (see more details in [31]). Yet, in [31], we used capsules as bounding volume: not only they make fast computation of the distance, but the capsule shape formula allows fast computation of the conservative advancement bounds. However, the capsule is not a strictly convex shape. In this work, since we use the STP-BV bounding volume, we applied only the Golden search method to seek for the min value of the distance function on a given time interval with a posteriori check.

5. Experimental validation

5.1. Description of the motion

We validated our method by generating a sitting motion for the humanoid robot HRP-2 [27]. This motion alternates contacts with both hands and feet to finally contacts the waist of the robot with the chair, see Figure 4. The contact of the feet and of the waist are fully defined by their position and orientation on the ground, whereas the contact of the hand are partially defined, i.e., the hands are asked to contact the table whatever the exact position.

This motion was decomposed into thirteen phases that start from the half-sitting motion and end when the robot has both feet contacting the floor, the left hand contacting the table and the waist contacting the chair. Each phase is decomposed into six time intervals and involves nine control points per joints as illustrated on Figure 3. We ensure joint trajectory continuity by computing the three first control points of the next phase from the three last ones of the current phases. Hence, the Cartesian trajectories are continuous having smooth transitions between the contact phases[2].

For one contacting body over every contact phase, we consider the same linear evolution of the coefficient β for all its contact forces. We do not impose the continuity of the coefficient β over the whole motion duration. We

[2]We neglect the impacts at contact creation because we constrain the landing speed at contact creations.

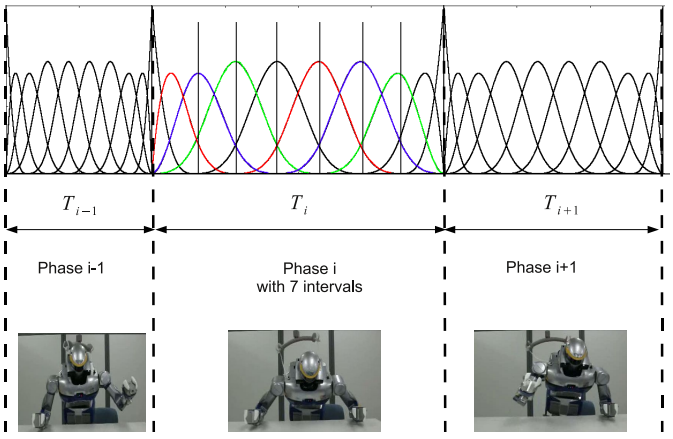


FIG 3. *Illustration of the decomposition of the motion into several phases and time interval. At each time instant the joint trajectory relies only on four parameters.*

set $\forall i, \alpha_i = 1000$, this allows to avoid redundant and unnecessary optimization parameters. Eventually, the parameter vector $\mathbf{P} \in \mathbb{R}^n$ contains all the control points, the linear coefficient of β and the phase durations.

During the sitting motion, the degrees of freedom of the head and of the wrists are set to a constant value, since they do not impact significantly the balance of the robot. Due to the fragility of the forces sensor at the endpoints of the arms, we choose to contact the table with the wrist of the robot. We consider seventy different collisions (thirty six between the robot and its environment and thirty four self collisions). We also consider the constraints of the contact forces, the velocity and torque limits (the joint position limits are set by limits on the variables).

We are considering the following cost function C as the weighted sum of joint torques, joint jerks and motion duration, as presented:

$$C(q) = a \int_0^T \sum_i \Gamma_i^2 dt + b \int_0^T \sum_i \dddot{q}_i{}^2 dt + cT \tag{5.1}$$

where $a = 1e-2$, $b = 1e-5$ and $c = 4$ are the value we set heuristically to have human-like walking motions with the HRP-2 [33]. We choose the order $n = 5$ for the polynomial approximation of the constraints. Eventually, the optimization gets 2339 variables (1883 free plus 556 constant variables) and 33852 constraints.

5.2. Technical implementation

Our method was programmed in the C++ language and executed on the following hardware and software: CPU Intel(R) Xeon(R) E5620, 8 cores, 2.4

GHz, Cache 12 Mo: Linux Gentoo 3.0.6 64bits.

We do not use the Lagrangian formalism of the IDM of Equation (2.3), but rather use a recursive formulation like in [30, 40], which is fast and requires a fewer number of operations [28].

Our software uses template classes for the model and specialized types to compute the derivative [3] and the Taylor form of the constraints and cost functions. The model computation needs a lot of creation and destruction of those types, which induce many memory allocations. To lower the memory allocations, we use the TC-Malloc library that gives better malloc performance than the glibc library.

Regarding the optimization solver, we believe[3] that it is better to use off-the-shelf available solvers since the optimization theory provides the fundamentals, but optimization is also a matter of tricks and recipes of tuning and numerical robustness. We also apply this policy by using the IPOPT solver [53]. IPOPT is free; it handles large non-linear optimization problems; it has a C++ interface, and was used in [37, 38].

5.3. Experimental validation

The sequence of the contacts is the output of the method presented in [7], and is given as an input to our motion generation method. However, we formulate our problem to allow adjusting these contacts. Not all collisions and self-collisions are relevant to check and only pertinent ones are accounted for a given multi-contact motion problem. We performed several optimizations, in order to define the appropriate collisions and self-collisions that deserve to be considered. Often, first solutions are not achieved by the robot because they are too fast or result in high impacts or simply too risky to try. In these cases we add artificial constraints to drive the solution to our desired behavior or tune the weights of the cost function. The final collision-free optimization process spent 610 iterations and 3h 26mn 37s of computation time to generate the optimal motion on the entire sequence of contacts, i.e. the entire sitting motion starting from the initial posture to the final one.

Any experimental validations of this motion is successfully performed by the HRP-2 Robot without over-passing its physical limits, without any unexpected collision or self-collisions and keeping its balance, see Figure 4 and Multimedia Extension 1. The records of the time history of the feet contact forces and joint knee torques are given in Figures 5 and 6. Moreover, this experiment proved to be repeatable at will since it is often reproduced during VIP visits to our lab.

The cost function and the constraints are model-based. However, the ankle of the robot is equipped with a shock absorbing mechanism, which is equivalent to having a non-controllable flexible joint. This non-modeled flexible

[3]It is also very strongly suggested in all the optimization textbooks, e.g. [17], page 5.

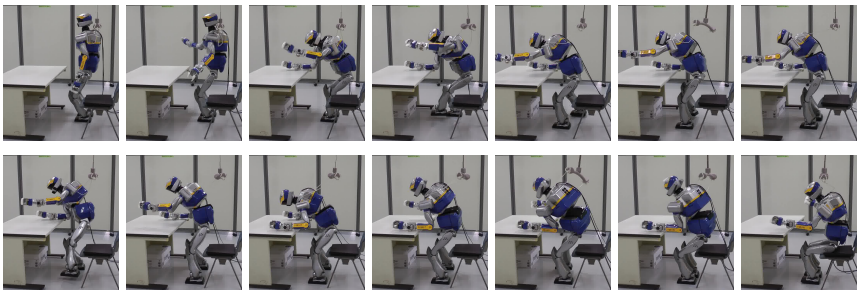
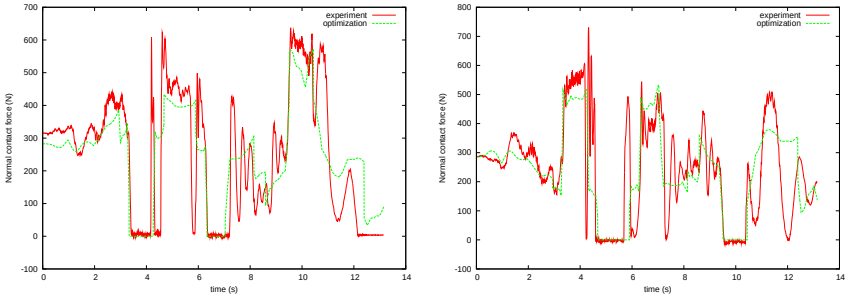


FIG 4. *Experimental validation of the sitting motion.*

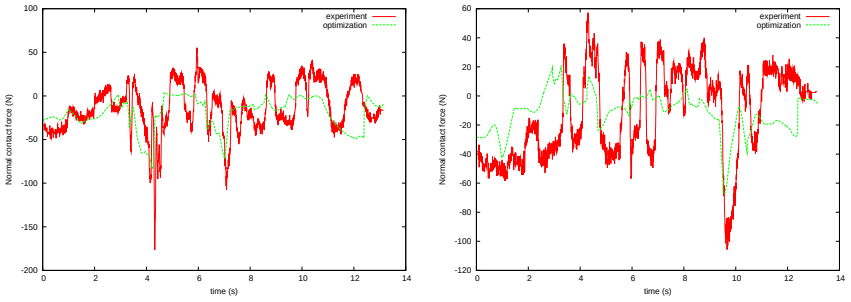


(a) left foot (b) right foot

FIG 5. *Experimental validation of the sitting motion, plots of the feet contact forces.*

part produces small hopping of the foot at the beginning of the contacting phase. We think that this flexibility is also the cause of the release of the contact between the left hand and the table at the end of the motion (cf. submitted video). Although we use only local joint PD control, without attitude or torque feedbacks, the motion was achieved with good balance within the physical limits of the robot. Yet, because of the existing discrepancies (e.g. we do not include flexibilities in the dynamic model, we might have different friction coefficients at contact, presence of small unexpected perturbations, etc.) joint torques differ from the model-computed ones in the planning process; see for example, the computed and actual torques of the knees in Fig. 6. Despite this, contact forces shown on Fig. 5 are quite similar to the computed ones, and makes it possible to identify the contact and non-contact phases. Indeed, as far as the model used in the optimization does not differ much, induced errors can still be absorbed by the PD tracking controller. But for considerable perturbations, discrepancies or uncertainties, re-planning is unavoidable.

In general, it doesn't make sense to include an error-tracking controller as part of the model-base optimization process. But we need to distinguish the low-level controller (that is based only on error tracking, which is assumed



(a) left foot (b) right foot

FIG 6. *Experimental validation of the sitting motion, plots of the knee-joint torques.*

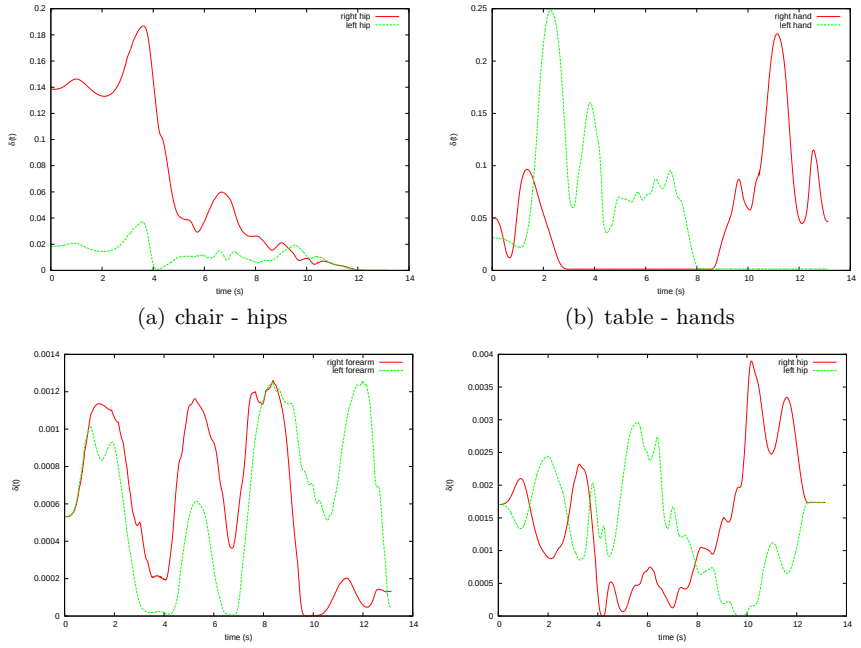
null in simulation, since no perturbation) and the controller strategy that may use robot states or contact forces, etc. A control strategy must be part of the model whereas the error tracking controller not.

In Figure 7, we present the evaluation of the constraint for the collision and self-collisions avoidance. On the Figure 7(a), we see that the hips never contact the chair before the final contact stance at $t = 12s$, the contact stance when both hands contact the table can easily be seen on Figure 7(b). Figures 7(c) and 7(d) represent two challenging self-collisions. It appears that self-collisions are very close to occur but will not (Note that $\delta(t)$ represents the signed square of the distance hence a distance of one millimeter will result in $\delta = 1e - 6$ on the plots).

5.4. Additional experiments

The chair-sitting scenario is chosen for its complexity and also because we have a ground-truth on running it with different control strategies, see e.g. [15]. We however assess our multi-contact motion generator with several other experiments involving the HRP-2 humanoid robot. In [35], we perform a kicking motion while making one hand stay still at a constant position, see snapshots Figure 8. We also generate a kicking motion with the right arm gripper uses a table to support the motion. The contact between the gripper and the table is put at the same position as the previous experiment in Figure 9. These experiments reveal also the difference in posture and force distribution when the humanoid's hand uses an additional contact support for the kicking. Using our method, we are certainly able to produce regular walking motions. Since there are some non-modeled flexibilities we consider additional constraints to account for the dynamic effects during motion [34]. Hence, we are able to generate normal walking (cf Fig. 11) and also a walking motion stepping on a fifteen-centimeter platform, see Figure 12 and [34].

We are aware that more efficient state-of-the-art gaited walking pattern



(a) chair - hips (b) table - hands

(c) chest - forearms (d) waist - hips

FIG 7. *Plots of the constraint δ for several collisions and self-collisions avoidance.*

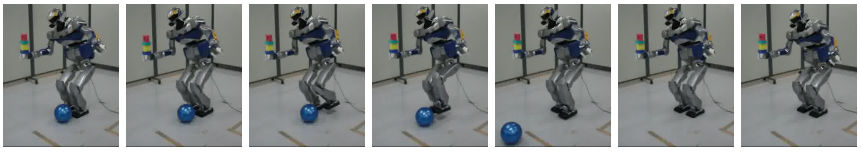


FIG 8. *The robot kicks the ball with the right gripper at a constant position.*

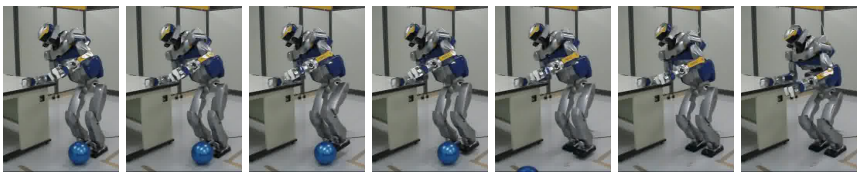


FIG 9. *The robot leans on a table in order to kick the ball.*

planners/controllers are able to generate this kind of walking motions. We believe that our method can be used to design whole-body trajectories for COM-COP reduced-model-based walking pattern generators. But more importantly, our method is able to consider (without any change in the software

FIG 10. *With the right arm, the HRP-2 robot takes an additional support on the desk, which allows it to stably lean and put a ball into the trash-box located under the desk.*

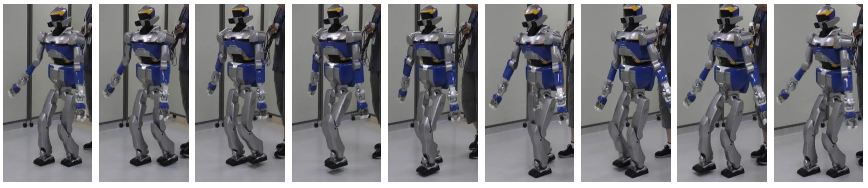


FIG 11. *Walking motion without additional constraints.*

FIG 12. *The HRP-2 over crossing a fifteen-centimeter platform: the HRP-2 takes steps the platform with one foot and over-cross it with the other.*

framework) some additional constraints in order to reproduce leg impairments [33] by setting a constant knee-position to emulate a humanoid knee sprained and wearing a splint (Figure 13), or by setting maximal contact forces on one foot to emulate a broken or painful foot as illustrated by Figure 14.

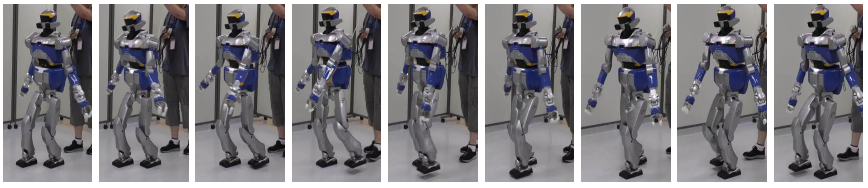


FIG 13. *Walking motion with locked knee.*

We can also consider contact around the edge. We implemented a walking motion that reproduce the rotation around the heels and toes as presented in Figure 15. Eventually, we performed a throwing motion with the HRP-2 robot. Figure 16 shows the execution of the motion for which the foot steps were determined heuristically in order to mimic a baseball player throwing.

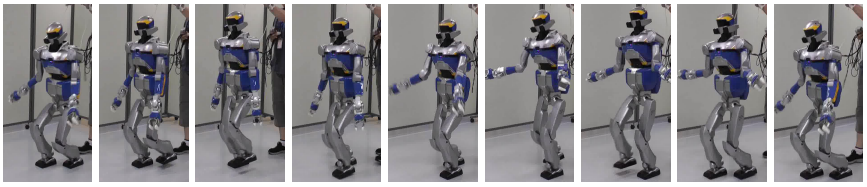


FIG 14. *Walking motion with sore leg.*

FIG 15. *The HRP-2 humanoid rotates around the edges during a walking motion.*

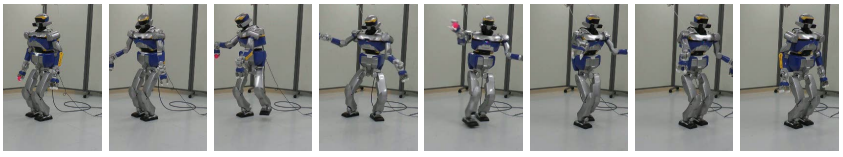


FIG 16. *The robot throws a ball mimicking a baseball player.*

Note that for all these experiments are made open-loop (i.e. without attitude of force feedbacks), we use only the local joint position PD controller when at least one hand is contacting the environment during the motion (Figs. 9 and 10). Otherwise, for the walking motions we use the HRP-2 embedded stabilizer presented in [26] in order to deal with the non-modeled flexibilities.

We also used our method to generate motions for the HOAP-3 humanoid robot and for a simulated model of the human. We built a motion database for the HOAP-3 robot that was used to perform walking motion during a fMRI based robotic embodiment [9]. We extended this method to study the impact of passive or stimulated knee joints on the fatigue of the upper limbs of a paraplegic person during a sitting pivot transfer motion [32]. All those experiments and simulations illustrate the versatility and efficiency of our method.

6. Discussions

We propose a versatile and effective method to generate dynamic and optimal multi-contact motions for humanoid robots. We highlight through seve-

ral typical complex scenarios, its ability to generate complex motions that cannot be computed using existing geometric planning methods or task-function approaches.

Since we compute the entire motion over all the phases during the same optimization process, we have an optimal motion for the considered sequence of contact stances. The generation of the sequence is based on quasi-static postures in the multi-contact planner that provide us with the contact stances. We believe that it is possible to merge the computation of the sequence of contact stances and the generation of the dynamic motion.

Our method still requires a large amount of computation time. To make it more attractive, we must decrease the computation time. There are several venues for that. The first trivial one is to reduce the horizon of the preview window by feeding the solver with only the upcoming newt two of three transitions rather than giving the entire sequence of contacts. This option is interesting only after reaching good computation performances since the method can then be transformed into a model-based nonlinear preview controller.

We currently use the L-BFGS approximation of the Hessian provided by the IPOPT solver [53] that considers the Hessian matrix full. Indeed, regarding the decomposition of the motion into several phases and time-interval, it appears that Hessian matrix is sparse which is not taken into account for the moment. An implementation of a sparse L-BFGS method should help to decrease the number of iterations, and hence the computation time. Another solution should be to include parallel architectures such as the General Purpose Graphics Processing Unit (GPGPU) in order to make use of parallel computation of the constraint and cost functions and of their gradients. It is possible to parallelize at least the computation of each time interval $[t]_i$ that are independent from each other. By considering these issues, it could be interesting to design a new solver that can use the computational efficiency of the GPGPU and exploit the sparse properties of the problem under study.

Albeit the presence of the flexible parts and assuming a perfectly know environment, the motion is currently performed without any balance controller. Future work should focus on the motion adaptation as presented in [37] or/and on a torque control law similar to [46] that can deal with uncertainties regarding the position and orientation of the contact points, i.e. on the position and dimension of the table and chair in our case.

7. Conclusion

In this paper, we solve multi-contact transitions motions using non-linear optimization techniques. Computed trajectories guarantee constraint satisfaction all over the time-motion whatever the time-grid size and accounts for all classical robotic constraints including non desired collisions avoidance. We experimented the computed trajectories on complex multi-contact

scenarios involving the HRP-2 humanoid robot. As far as we know, there is no existing method that can produce such complex multi-contact motions. This is mainly due to the difficulties to model the dynamic balance and unilateral forces, to deal efficiently with the continuous constraints and to integrate the collision and self-collision in a safe way. We presented an analytical formulation of the contact forces that counterpart the dynamic effect and encompass the balance of the robot. To ensure the validity of the continuous constraints we make use of Taylor series expansion approximations for the models of the robot.

Our future work is now focuses toward building and improving performance and robustness strategies in using effectively such a method in close interchange with low-level controllers.

Appendix A: Index to Multimedia Extensions

The multimedia extensions to this article are at: http://www.ijrr.org

Extension	Type	Description
1	Video	Experimental Validations

Appendix B: Analytical formulation of the contact forces

B.1. Problem

Knowing the joint trajectories, we analytically compute the contact forces in Eq. (2.6), which can also be written as:

$$\mathbb{F} = -(\mathbf{J}_2^T(q))^{-1}\mathbb{D}_2 \tag{B.1}$$

Since the matrix $\mathbf{J}_2$ is not necessarily square, the pseudo inverse: $(\mathbf{J}_2^T)^+ = (\mathbf{J}_2\mathbf{J}_2^T)^{-1}\mathbf{J}_2$ can be computed instead of $(\mathbf{J}_2^T)^{-1}$. By definition, the pseudo-inverse solves an equation system that has more degrees of freedom than equations, which is our case, by minimizing the Euclidean norm of the solution. Yet, our primary interest is not to minimize the Euclidean norm but to find a function $f(q, \mathbb{D}_2)$ that solves the contact forces which satisfy both Eq. (2.6) and as much as possible Eq. (2.4), that is:

$$\mathbb{F} = f(q, \mathbb{D}_2) \tag{B.2}$$

We propose to find the analytic solution for the contact forces that reflects the balance of the robot, cf. Eq. (2.6), and the friction constraints, Eq. (2.4); i.e., the analytic solution to the following problem:

$$\begin{array}{c} \min \frac{1}{2} \sum_i \beta_i (\alpha_i ||F_i^t||^2 + F_i^{n2}) \\ \sum_i \left(\left[\begin{array}{c} \hat{P}_i A_i \\ A_i \end{array} \right] [F_i] \right) + [\mathbb{D}_2] = 0 \end{array} \tag{B.3}$$

In this formulation, we decompose $\mathbf{J}_2^T$ through (i) $\hat{P}_i$, the screw operator of the contact position, and (ii) A_i the orientation of the contact framework; $[\mathbb{D}_2] = [M_x, M_y, M_z, F_x, F_y, F_z]^T$ is the effort due to the free dynamics, β_i a coefficient to equilibrate (or not) the repartition of the forces, α_i is used to weight the tangential forces in regards to its corresponding normal force. The solution to this problem ensures the dynamic equality constraint Eq. 2.6, but does not guarantee the feasibility of the contact forces that must be checked by the optimization solver. Note that defining $\forall i \ \beta_i = 1$, $\alpha_i = 1$ is equivalent to solve the pseudo-inverse problem.

B.2. Analytic solution

The resolution of this problem starts by writing the Lagrangian:

$$L = \sum_i \left[\begin{array}{c} \beta_i \alpha_i F_{x,i}^2 \\ \beta_i \alpha_i F_{y,i}^2 \\ \beta_i F_{z,i}^2 \end{array} \right] + \left(\sum_i \left[\begin{array}{c} \hat{P}_i A_i \\ A_i \end{array} \right] [F_i] + [\mathbb{D}_2] \right) [\lambda] \tag{B.4}$$

Here, we assume that the z-axis is the normal direction of the contact forces and define $[\lambda]$ as the vector of the Lagrange multipliers. The optimal solution fulfills the optimality condition:

$$\frac{\partial L}{\partial (F_{\circ,i}, \lambda_j)} = 0 \tag{B.5}$$

From the derivative with respect to F_i we have:

$$F_i = -\mathbb{W}_i^{-1} \left[\begin{array}{cc} \hat{P}_i A_i & A_i \end{array} \right] [\lambda] \tag{B.6}$$

with $\mathbb{W}_i = \text{diag}(\beta_i \alpha_i, \beta_i \alpha_i, \beta_i)$. Then, we replace the contact forces expression in the set of the equality constraint and we get:

$$\Omega [\lambda] = - [\mathbb{D}_2] \tag{B.7}$$

With:

$$\Omega = \sum_i \left(\left[\begin{array}{c} \hat{P}_i A_i \\ A_i \end{array} \right] \mathbb{W}_i^{-1} \left[\begin{array}{cc} \hat{P}_i A_i & A_i \end{array} \right] \right) \tag{B.8}$$

where $\Omega \in \mathbb{R}^{6 \times 6}$ is a 6×6 matrix that we can easily invert using the Gauss-Jordan algorithm to find the value of the Lagrange multipliers:

$$[\lambda] = -\Omega^{-1} [\mathbb{D}_2] \tag{B.9}$$

Then, we put the Lagrange multipliers $[\lambda]$ in Eq. (B.6) to get the value of the contact forces. Eventually, we are able to compute the contact forces from the joint trajectories and, using Equation (2.3), we can compute the joint torques.

Appendix C: Properties of the B-Spline

We briefly present the properties of the B-Spline that are relevant during the motion generation process.

C.1. Control points

The first nicety is related to the computation of the derivative of the state variables regarding the optimization parameters. Since, at any time instants, the joint values rely only on four control points, we choose to define intervals so that, all along a time interval, the joint value relies only on four control points, see Figure 3. This avoids to compute the derivative regarding all the parameters since only a few of them have a non-null value.

C.2. Convex hull

B-Splines ($q(t)$) is entirely contained in the convex hull of its control points; that yields:

$$\begin{array}{ll} \text{if} & \forall i \in [1,m] \;\; \underline{q} \leq p_i \leq \overline{q} \\ \text{then} & \forall t \in [\Delta t] \;\; \underline{q} \leq q(t) \leq \overline{q} \end{array} \tag{C.1}$$

We use this property to ensure the joint limits without implementing them as sets of continuous inequality constraints, but rather considering bounds directly on the optimization parameters.

C.3. Derivative of a B-Spline

The time-derivate of a B-Spline with m control points is another B-Spline parameterized with $m-1$ control points. This is obtained by derivation of the Cox-de-Boor recursion [11] with respect to time t:

$$\dot{q}(t) = \sum_{i=1}^{m} \dot{b}_i^k(t) p_i = \sum_{i=1}^{m-1} b_i^{k-1}(t) r_i \tag{C.2}$$

with:

$$r_i = \frac{k}{u_{i+k+1} - u_{i+1}} (p_{i+1} - p_i) \tag{C.3}$$

here k is the order of the B-Spline basis functions, u_i is the i^{th} component of the nodal vector as defined in [11]. It is then possible to obtain a system

of $(m-1)$ linear inequalities to impose joint speed limits, as we enforced joint position limits as bounds on the B-Spline parameters. Thus, we add the following constraints:

$$\underline{\dot{q}} \leq r_i \leq \overline{\dot{q}} \tag{C.4}$$

Extensions to upper derivatives follow the same way.

C.4. Optimality

Hence, the obtained solution is very likely sub-optimal as there is no reciprocal to Eq. (C.1). We already raised this issue in [35]. Yet, the parameterization of the joint trajectories (by any way) leads already to a sub-optimal motion and we believe that there is a venue for a thorough study on other possible parameterizations. Indeed, any kind of parameterization restricts the IP problem's solution for the chosen parameterization, which is de facto sub-optimal.

Acknowledgement

This work is partially supported by grants from the German Research Foundation (DFG: Deutsche Forschungsgemeinschaft), from the Japan Society for the Promotion of Science (JSPS) Grant-in-Aid for JSPS Fellows (P09809) and for Scientific Research (B), 22300071, 2010, and from the RoboHow.Cog FP7 `www.robohow.eu`. Authors would like to thank Dr Francois Keith and Pierre Gergondet for the help and assistance in making the simulation and experiments.

References

[1] Hitoshi Arisumi, Sylvain Miossec, Jean-Rémy Chardonnet, and Kazuhito Yokoi. Dynamic lifting by whole body motion of human robots. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 668–675, september 2008.

[2] Mehdi Benallegue, Adrien Escande, Sylvain Miossec, and Abderrahmane Kheddar. Fast c1 proximity queries using support mapping of sphere-torus-patches bounding volumes. In Robotics and Automation, 2009. ICRA '09. IEEE International Conference on, pages 483 –488, may 2009.

[3] Claus Bendtsen and Ole Stauning. FADBAD, a flexible c++ package for automatic differentation.

[4] Martin Berz, Georg Hoffsttter, and Georg Hoffst Atter. Computation and application of taylor polynomials with interval remainder bounds. Reliable Computing, 4:83–97, 1998.

[5] James E. Bobrow. Optimal robot plant planning using the minimum-time criterion. Robotics and Automation, IEEE Journal of, 4(4):443 –450, aug 1988.

[6] Karim Bouyarmane, Adrien Escande, Florent Lamiraux, and Abderrahmane Kheddar. Potential field guide for humanoid multicontacts acyclic motion planning. In IEEE Int. Conf. on Robotics and Automation, may 2009.

[7] Karim Bouyarmane and Abderrahmane Kheddar. Humanoid robot locomotion and manipulation step planning. Advanced Robotics (Int. J. of the Robotics Society of Japan), Special Issue on the Cutting Edge of Robotics in Japan, 26(10):1099–1126, July 2012.

[8] Mary D. Klein Breteler, Stan C.A.M. Gielen, and Ruud G.J. Meulenbroek. End-point constraints in aiming movements : effects of approach angle and speed. Biological Cybernetics, 85(1):65 – 75, july 2001.

[9] Ori Cohen, Sébastien Druon, Sébastien lengagne, Avi Mendelsohn, Malach Rafael, Abderrahmane Kheddar, and Doron Friedman. fmri based robotic embodiment: a pilot study. In IEEE/RAS-EMBS Int. Conf. on Biomedical Robotics and Biomechatronics (BioRob), 2012.

[10] Tilak de Alwis. Maximizing or minimizing polynomials using algebraic inequalities. In Proc. 9th Asian Technological Conference on Mathematics, pages 88–97, Dec. 2004. Singapore.

[11] Carl De Boor. A Pratical Guide to Splines, volume 27. Springer-Verlag, New York, 1978.

[12] Moritz Diehl, Hans G. Bock, Holger Diedam, and Pierre-Brice Wieber. Fast direct multiple shooting algorithms for optimal robot control. In Moritz Diehl and Katja Mombaur, editors, Fast Motions in Biomechanics and Robotics, volume 340 of Lecture Notes in Control and Information Sciences, pages 65–93. Springer Berlin / Heidelberg, 2006.

[13] Adrien Escande and Abderrahmane Kheddar. Contact planning for acyclic motion with tasks constraints. In IEEE/RSJ International Conference on Intelligent RObots and Systems (IROS 2009), Oct. 11-15 2009.

[14] Adrien Escande, Abderrahmane Kheddar, and Sylvain Miossec. Planning support contact-points for humanoid robots and experiments on HRP-2. In Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on, pages 2974–2979, Beijing, October 2006.

[15] Adrien Escande, Abderrahmane Kheddar, Sylvain Miossec, and Sylvain Garsault. Planning support contact-points for acyclic motions and experiments on HRP-2. In Oussama Khatib, Vijay Kumar, and George Pappas, editors, Internation Symposium on Experimental Robotics, number 54 in Springer tracts in advanced robotics, pages 293–302, Athens, Greece, 14-17 July 2008. Springer-Verlag.

[16] Adrien Escande, Sylvain Miossec, and Abderrahmane Kheddar. Con-

tinuous gradient proximity distance for humanoids free-collision optimized-postures. In IEEE-RAS 7th International Conference on Humanoid Robots, 2007.

[17] Philip E. Gill, Walter Murray, and Margaret H. Wright. Practical optimization. Academic Press, 1982.

[18] Ambarish Goswami. Postural stability of biped robots and the foot-rotation indicator (FRI) point. International Journal of Robotic Research, 18(6):523–533, 1999.

[19] Kensuke Harada, Shuuji Kajita, Kenji Kaneko, and Hirohisa Hirukawa. Dynamics and balance of a humanoid robot during manipulation tasks. IEEE Transactions on Robotics, 22(3):568–575, June 2006.

[20] Kris Hauser, Tim Bretl, and Jean-Claude Latombe. Non-gaited humanoid locomotion planning. In Humanoid Robots, 2005 5th IEEE-RAS International Conference on, pages 7–12, Tsukuba,, December 2005.

[21] Kris Hauser and Jean-Claude Latombe. Multi-modal motion planning in non-expansive spaces. The International Journal of Robotics Research June vol. no. 7, 29(7):897–915, June 2010.

[22] R. Hettich and K. O. Kortanek. Semi-infinite programming: theory, methods, and applications. SIAM Rev., 35(3):380–429, 1993.

[23] Hirohisa Hirukawa, Shizuko Hattori, Kensuke Harada, Shuuji Kajita, Kenji Kaneko, Fumio Kanehiro, Kiyoshi Fujiwara, and Mitsuharu Morisawa. A universal stability criterion of the foot contact of legged robots - adios zmp. In IEEE International Conference on Robotics and Automation (ICRA)., pages 1976– 1983, may 2006.

[24] Sang-Ho Hyon, Joshua G. Hale, and Gordon Cheng. Full-body compliant humanhumanoid interaction: Balancing in the presence of unknown external forces. IEEE Transactions on Robotics, 23(5):884 – 898, 2007.

[25] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kensuke Harada, Kazuhito Yokoi, and Hirohisa Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. In IEEE International Conference on Robotics and Automation, volume 2, pages 1620 – 1626, september 2003.

[26] Shuuji Kajita, Takashi Nagasaki, Kenji Kaneko, Kazuhito Yokoi, and Kazuo Tanie. A running controller of humanoid biped HRP-2LR. In Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on, pages 616–622, April 2005.

[27] Kenji KANEKO, Fumio KANEHIRO, Shuuji KAJITA, Hirohisa HIRUKAWA, Toshikazu KAWASAKI, Masaru HIRATA, Kazuhiko AKACHI, and Takakatsu ISOZUMI. Humanoid robot HRP-2. In Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on, volume 2, pages 1083–1090, April/May 2004.

[28] Wisama Khalil and Etienne Dombre. Modeling, Identification &

Control of Robots. Hermes Sciences Europe, 3 edition, march 2002.

[29] Sung-Hee Lee and Ambarish Goswami. A momentum-based balance controller for humanoid robots on non-level and non-stationary ground. Journal of Autonomous Robots, 33(4):399–414, november 2012.

[30] Sung-Hee Lee, Junggon Kim, F.C. Park, Mumsang Kim, and James E. Bobrow. Newton-type algorithms for dynamics-based robot movement optimization. In IEEE Transactions on robotics, volume 21, pages 657–667, 2005.

[31] Youngeun Lee, Sébastien Lengagne, Abderrahmane Kheddar, and Young J. Kim. Accurate evaluation of a distance function for optimization-based motion planning. In IEEE/RSJ International Conference on Intelligent Robots and Systems IROS, 2012.

[32] Sébastien lengagne, Jovana Jovic, Camilla Pierella, Philippe Fraisse, and Christine Azevedo Coste. Generation of multi-contact motions with passive joints: Improvement of sitting pivot transfer strategy for paraplegics. In IEEE/RAS-EMBS Int. Conf. on Biomedical Robotics and Biomechatronics (BioRob), 2012.

[33] Sébastien Lengagne, Abderrahmane Kheddar, Sébastien Druon, and Eiichi Yoshida. Emulating human leg impairments and disabilities in walking with humanoid robots. In IEEE Int. Conf. on RObotics & BIOmimetics, 2011.

[34] Sébastien Lengagne, Abderrahmane Kheddar, and Eiichi Yoshida. Considering floatting contact and un-modeled effects for multi-contact motion generation. In Workshop on Humanoid Service Robot Navigation in Crowded and Dynamic Environments at the upcoming IEEE Humanoids Conference, 2011.

[35] Sébastien Lengagne, Paul Mathieu, Abderrahmane Kheddar, and Eiichi Yoshida. Generation of dynamic motions under continuous constraints: Efficient computation using b-splines and taylor polynomials. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2010.

[36] Sébastien Lengagne, Paul Mathieu, Abderrahmane Kheddar, and Eiichi Yoshida. Generation of dynamic multi-contact motions: 2d case studies. In IEEE-RAS Internationnal conference on Humanoid robots, 2010.

[37] Sébastien Lengagne, Nacim Ramdani, and Philippe Fraisse. Planning and fast replanning safe motions for humanoid robots. IEEE Transactions on Robotics, 27(6):1095 –1106, dec. 2011.

[38] Sylvain Miossec, Kazuhito Yokoi, and Abderrahmane Kheddar. Development of a software for motion optimization of robots– application to the kick motion of the HRP-2 robot. In IEEE International Conference on Robotics and Biomimetics, pages 299–304, 2006.

[39] Katja D. Mombaur, Hans G. Bock, Johannes P. Schlöder, and Richard W. Longman. Open-loop stable solutions of periodic optimal control problems in robotics. ZAMM - Journal of Applied

Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik, 85(7):499–515, 2005.

[40] F. C. Park, J. E. Bobrow, and S. R. Ploen. A lie group formulation of robot dynamics. Int. J. Rob. Res., 14(6):609–618, December 1995.

[41] Aurelio Piazzi and Antonio Visioli. Global minimum-time trajectory planning of mechanical manipulators using interval analysis. International Journal of Control, 71:631–652(22), November 1998.

[42] Aurelio Piazzi and Antonio Visioli. Global minimum-jerk trajectory planning of robot manipulators. In IEEE Transactions on Industrial Electronics, volume 47, pages 140–149, febru 2000.

[43] Rembert Reemtsen and Jan-J. Rückmann, editors. Nonconvex optimization optimization and its applications: Semi-infinite Programming. Kluwer Academic Publishers, 1998.

[44] Ludovic Righetti, Jonas Buchli, Michael Mistry, and Stefan Schaal. Control of legged robots with optimal distribution of contact forces. In IEEE-RAS International Conference on Humanoid Robots, pages 318 – 324, 2011.

[45] Gerrit Schultz and Katja D. Mombaur. Modeling and optimal control of human-like running. Mechatronics, IEEE/ASME Transactions on, 15(5):783 –792, oct. 2010.

[46] Luis Sentis, Jaeheung Park, and Oussama Khatib. Compliant control of multicontact and center-of-mass behaviors in humanoid robots. IEEE transactions on robotics, 26:483–501, 2010.

[47] Marc C. Steinbach. Optimal motion design using inverse dynamics. Technical report, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 1997.

[48] Benjamin J. Stephens and Christopher G. Atkeson. Dynamic balance force control for compliant humanoid robots. In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1248 – 1255, 2010.

[49] Oskar Von Stryk. Optimal control of multibody systems in minimal coordinates. ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik, 78(S3):1117–1120, 1998.

[50] Y. Uno, M. Kawato, and R. Suzuki. Formation and control of optimal trajectory in human multijoint arm movement. Biological Cybernetics, 6(2):89–101, juin 1989.

[51] Oskar Von Stryk. Numerical solution of optimal control problems by direct collocation, 1993.

[52] Miomir Vukobratović and Branislav Borovac. Zero-moment point : Thirty five years of its life. International Journal of Humanoid Robotics, 1(1):157–173, 2004.

[53] Andreas Wächter and Lorenz T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale

nonlinear programming. Mathematical Programming, 106:22–57, 2006.
[54] Xinyu Zhang, Stephane Redon, Minkyoung Lee, and Young J. Kim. Continuous collision detection for articulated models using taylor models and temporal culling. ACM Trans. Graph., 26(3):15, 2007.

B.2 Utilisation des fonctions de base dans un processus d'optimisation.

Article : **S. Lengagne**. *Comparison of several basis functions for interval-based optimization*, soumis à International Journal of Approximate Reasoning.

Comparison of several basis functions for interval-based optimization

Sébastien Lengagne
Université Clermont Auvergne, Clermont Auvergne INP, CNRS, Institut Pascal, ISPR, Clermont-Ferrand 63000, France

Abstract

Interval Analysis proves valuable for tackling optimization and constraint satisfaction problems, offering the ability to ascertain the absence of a solution or identify the global optimal solution while considering uncertainties. Nevertheless, it encounters challenges stemming from an overestimation of functions, referred to as pessimism. Previous work used the properties of the Bernstein basis functions and of the Kronecker product to achieve a more optimistic evaluation of mathematical functions. In this paper, we consider and introduce several basis functions and compare their performances on different robotic-based optimization processes.

Keywords: Interval Analysis, basis-functions, pessimism, optimization

1. Introduction

In many fields, obtaining optimal parameters is a critical concern. In robotics, the identification process, where optimization enables the discovery of parameters aligning the model results with experimental outcomes is challenging issue [7]. Optimization techniques empower robots to execute motions or static poses with minimal energy consumption or perform tasks as swiftly as possible [11]. These techniques yield optimal parameters that must adhere to a set of constraints, such as joint limits, joint torques, balance, ensuring the integrity of the robotic system, task completion, and optimization of specified criteria. The constraints and criteria functions can range from simple (continuous, linear, monotonic, etc.) to very complex (discontinuous, non-linear, with local extrema), depending on the application.

For straightforward cases (quadratic criteria without constraints), optimal parameters can be easily found using Lagrange multipliers, active set methods [19], or other similar approaches. In more intricate scenarios, heuristic algorithms or iterative algorithms [9, 26] are often employed to produce solutions within a finite time frame. In cases of increased complexity, especially with discontinuous or non-differentiable functions, Genetic Algorithms [24] may provide a solution without guaranteeing optimality.

Optimization techniques grounded in Interval Analysis deliver globally optimal results, ensuring constraint satisfaction within a finite space, and can identify infeasible problems [5]. However, these algorithms suffer from prohibitive computation times due to a large number of iterations caused by the pessimism induced by Interval Analysis. Pessimism leads to an overestimation of functions and is intrinsically linked to the inclusion function used in Interval Analysis. Although the given interval remains conservative (containing the actual solution), it is larger than the actual solution, potentially increasing the number of iterations in optimization algorithms. To mitigate pessimism, the natural inclusion function can be replaced by the centred or Taylor-centred inclusion [18]. Previous work [10] considered Bernstein and Kronecker product properties to evaluate functions in a less conservative manner, aiming to reduce pessimism and subsequently decrease the number of iterations and computation time in optimization processes.

In this paper, we introduce and compare different basis functions to evaluate their performances and their benefits during optimization process. We consider well-known Bernstein and BSplines functions and other basis functions based on optimization process such as the Minimum Volume basis function and it approximation counterpart presented in [25]. We also introduce the Minimum Norm and the Minimum Variance optimized based basis function and use recursives basis functions that presents interesting structure.

Section 2 presents the optimization process and the method to solve it based on Interval Analysis or on the basis-functions-based evaluation. Section 3 details the different basis functions. Eventually, Section 4 compares the performances during simple evaluations and optimization processes. Note that the code is available at [1] and all the results are deeply presented in [12].

2. Problem formulation and resolution

2.1. Optimization problem

In this paper, we compare the effectiveness of different basis functions used in the evaluation process of criteria and constraint function over multi-dimensional interval through the resolution of optimization problems such as:

$$\begin{array}{rr} \text{Find} & q \in \mathbb{Q} \subset \mathbb{R}^n \\ \text{such as} & \min\limits_{q} \mathcal{F}(q) \\ \text{with } \forall j \in \{1,\ldots,m\} & \mathcal{G}_j(q) \in [\underline{g}_j, \overline{g}_j] \end{array} \tag{1}$$

Where n is the number of parameters, $\mathbb{Q}$ the finite search space, $\mathcal{F}(q)$ the criteria function and $\mathcal{G}_j(q)$ the set of j constraints that must remain within the interval $[\underline{g}_j, \overline{g}_j]$ with the lower bound $\underline{g}_j$ and the upper bound $\overline{g}_j$ limits.

2.2. Interval Analysis and pessimism

Interval Analysis (IA) was initially developed to take into account the quantification errors introduced by the floating point representation of real numbers with computers [15, 17, 23]. Several works showed that IA is competitive compared to the classic optimization solvers since it provides guaranteed solutions respecting the constraints [6, 14, 20].

Let us define an interval $[a] = [\underline{a}, \bar{a}]$ as a connected and closed subset of $\mathbb{R}$, with $\underline{a} = Inf([a])$, $\bar{a} = Sup([a])$ and $Mid([a]) = \frac{\underline{a}+\bar{a}}{2}$. The set of all real intervals of $\mathbb{R}$ is denoted by $\mathbb{IR}$. A vector of interval is defined as a box. Real arithmetic operations are extended to intervals. Consider an operator $\circ \in \{+, -, *, \div\}$ and $[a]$ and $[b]$ two intervals. Then:

$$[a] \circ [b] = [inf_{u\in[a],v\in[b]}\, u \circ v, \quad sup_{u\in[a],v\in[b]}\, u \circ v] \tag{2}$$

Consider a function $\mathbf{m} : \mathbb{R}^n \longmapsto \mathbb{R}^m$; the range of this function over a box [a] is given by:

$$\mathbf{m}([\mathbf{a}]) = \{\mathbf{m}(\mathbf{u}) \mid \mathbf{u} \in [\mathbf{a}]\} \tag{3}$$

Considering and interval function $[\mathbf{m}] : \mathbb{IR}^n \longmapsto \mathbb{IR}^m$, we can define it as an inclusion function for $\mathbf{m}$ if:

$$\forall [\mathbf{a}] \in \mathbb{IR}^n,\ \mathbf{m}([\mathbf{a}]) \subseteq [\mathbf{m}]([\mathbf{a}]) \tag{4}$$

The natural inclusion function of $\mathbf{m}$ is evaluated by replacing each occurrence of a real variable by the corresponding interval and each standard function

by its interval counterpart. Hence, the optimization problem of Equation 1 may be turned into its interval analysis form, such:

$$\begin{array}{rl} \text{Find} & [q] \in \mathbb{IQ} \subset \mathbb{IR}^n \\ \text{such as} & \min\limits_{[q]} [\mathcal{F}]([q]) \\ \text{with } \forall j \in \{1, \ldots, m\} & [\mathcal{G}_j]([q]) \in [\underline{g}_j, \overline{g}_j] \end{array} \tag{5}$$

With $\mathbb{IR}$ and $\mathbb{IQ}$ represent the set of all the possible interval in $\mathbb{R}$ and $\mathbb{Q}$.

Nevertheless, the choice of the inclusion function will greatly impact the accuracy of the computation due to an overestimation called *pessimism*. This overestimation is mainly caused by the multi-occurrence of variables in equations [2, 16, 27]. To illustrate this property, we can consider $x \in [-1; 1]$ and the function $y = f(x) = x - x$. Obviously $y = 0$ but using interval analysis and the natural inclusion function we get $[y] = [x] - [x] = [-2; 2]$ that contains the solution but with a huge over estimation. However, accuracy impacts the performanes of the optimization methods as presented in Section 2.3. The work presented in [10] shows that the use of basis functions may reduce the pessimism, hence speed up the computation.

2.3. Branch and bound algorithm

To demonstrate the effectiveness of our approach, we will compare the performance (i.e., number of iterations and the computation time) of a basic optimization algorithm using the natural inclusion function (using Interval Analysis) with several basis-function based inclusion function.

Branch and bound algorithms, as the one proposed in in the Algorithm 1, is employed to solve optimization problems defined by Equation (5) through Interval Analysis. They are based on the evaluation of the constraint and objective functions for a current box (set of interval). These evaluations determine whether the box should be discarded (due to constraint violation, excessive objective function value or too small box regarding a given threshold) or split into two smaller boxes. The splitting process, also called bisection, aims to obtain a more precise evaluation of the constraints and the objective function. The constraint $\mathcal{G}_j([q])$ is deemed violated if its interval evaluation has no intersection with the bounds $[g_j]$. Various splitting methods can be applied [15]. The Algorithm 1 can be modified to include the contraction of the considered box such as shown in [10]. For simplicity (as this paper focuses on the basis function evaluation), we do not consider more advanced bisection processes and we split the box by dividing the largest interval in the middle.

2.4. Basis function Properties

In this section, we present the main properties of basis functions we use to evaluate the extrema of a function over a given box. We consider a set of basis function $b^k(q)$ of order k that fits the following constraints:

$$\forall q \in [\underline{q}, \overline{q}] \quad \sum_{i=1}^{m} b_i^k(q) = 1 \tag{6}$$

$$\forall i, \forall q \in [\underline{q}, \overline{q}] \quad 0 \leq b_i^k(q) \leq 1 \tag{7}$$

Those two constraints imposes that if a function can be defined as:

$$F(q) = \sum_{i=1}^{m} b_i^k(q) p_i \tag{8}$$

hence, this yields:

$$\text{if} \quad \forall i \in [1, m] \quad \underline{F} \leq p_i \leq \overline{F} \quad \text{then} \quad \forall q \in [\underline{q}, \overline{q}] \quad \underline{F} \leq F(q) \leq \overline{F} \tag{9}$$

Eventually, knowing the equivalent control points p_i of a function $F(q)$ for a given box $[q]$ allows to have a conservative evaluation of the bound of $[f([q])]$.

2.5. Function evaluation

Equation (8) presents the case of a variable of one dimension $q \in \mathbb{R}$. To deal with multivariable $q \in \mathbb{R}^n = \{q_1, q_2, \ldots, q_n\}$, we assume that any function $\mathcal{F}(q)$ or $\mathcal{G}_j(q)$ can be represented in a polynomial form such as:

$$F(q) = [a_1, a_2, a_3, \ldots, a_i, \ldots].[1, q_1, q_2, q_1 q_2, \ldots, \mu_i, \ldots]^T \tag{10}$$

where μ_i represents the i-th monomial. We define the set of all the polynomial coefficient a_i as the vector X, and the set of all the corresponding monomial μ_i such as M. We reformulate Equation (10) into:

$$F(q) = X.M^T \tag{11}$$

By extrapoling Equation (8), Equation (10) can also be expressed regarding its basis function form:

$$F(q) = \sum_{i=1}^{m_1} \sum_{j=1}^{m_2} \ldots \sum_{z=1}^{m_n} \left(b_i^{m_1}(q_1) b_j^{m_2}(q_2) \ldots b_z^{m_n}(q_n) \right) \times p_{i,j,\ldots,z} \tag{12}$$

With $b_i^{m_k}(q_k)$ the m_k-order-basis function relative to the variable q_k and $p_{i,j,\ldots,z}$ a control point Defining P the set of control points, we can use the compact representation:

$$F(q) = \mathbb{B}.P.M^T \tag{13}$$

where $\mathbb{B}$ can be written using the Kronecker product $\otimes$ as presented in [13, 21] such as:

$$\mathbb{B} = \mathbf{B}_1 \otimes \mathbf{B}_2 \otimes \ldots \otimes \mathbf{B}_i \otimes \ldots \otimes \mathbf{B}_n \tag{14}$$

Where, the matrix $\mathbf{B}_i$ links the control points to the coefficient of the polynomial expression of the basis functions of input q_i. From Equations (11) and (13), we define the following relation:

$$X = \mathbb{B}.P \tag{15}$$

Using the Kronecker product inversibility property we turn Equation (15) into:

$$P = \left(\mathbf{B}_1^{-1} \otimes \mathbf{B}_2^{-1} \otimes \ldots \otimes \mathbf{B}_n^{-1}\right) X \tag{16}$$

Eventually, starting from a given box $[q]$, on can compute the polynomial vector X and the control point vector P. Then, considering the basis function properties defined in Equation (9), it makes possible to estimate the bounding value of the function $F([q])$ for the given box $[q]$.

2.6. Determining intersection

During each iteration of the optimization process, the goal of the evaluation of the functions $\mathcal{F}(q)$ and $\mathcal{G}_j(q)$ is to determine if the projection of a given box $[q]$. This bound can be constant for the constraint evaluation (Line 12 of Algorithm 1) or depending of the previous evaluation (Lines 5 and 23 of Algorithm 1) for the criteria evaluation. If this projection is inside or overlapping a given bound $[\underline{b}, \overline{b}]$, then the optimization process may continue performing a bisection of the box $[q]$. If this projection has no intersection with $[\underline{b}, \overline{b}]$, the current box $[q]$ is discarded. Hence, considering a less pessimistic evaluation will reduction the number of bisection, hence may reduce the computation time.

Using the basis function based evaluation, i.e. computing the control point P, to know if the projection is fully inside (Line 18 of Algorithm 1) or outside (Line 13 of Algorithm 1) the bound $[\underline{b}, \overline{b}]$, all the control points must be computed. However, the main advantage of the formulation proposed in Equation (16) is that only a few part of the control points may highlight

overlapping (Line 15 of Algorithm 1), hence removing the need to compute the other part of the control point. Thus, each control point in the vector P is computed sequentially.

Finally, in order to reduce the computation time, we can :

- reduce the pessimism using the best fitted basis function, hence reducing the number of iterations,
- reduce the computation time of each iterations by considering sparse of repetitive parts of the matrices $\mathbf{B}_i^{-1}$

Moreover, it is important to highlight that only a few value of X are different from 0. Eventually, a smart implementation of Equation (16) speeds up the computation.

2.7. Implementation tricks

In order to deal with the technical implementation, some issue must be adressed.

2.7.1. Input Normalization

The first issue is a bout the implementation of the Equation(16). In [8], the author consider that the vector X remains constant over whole the computation process and update the matrix $\mathbf{B}_i$ regarding the current bounds of $[q_i]$ at each iterations. The main drawback of this solution is the numerical errors while computing ${\mathbf{B}_i}^{-1}$ for narrow intervals on $[q_i]$ due to ill conditioning. To solve this issue, we use the inputs normalization with the following affine change of variables:

$$[q_i] = m_i + [q_i^{ref}]\frac{d_i}{2} \tag{17}$$

with m_i and d_i are the middle and the diameter of interval $[q_i]$ and $[q_i^{ref}]$ is set as the reference interval of $[q_i]$ and is initialized at $[q_i^{ref}] = [-1; 1]$. By doing so, the matrix $\mathbf{B}_i$ rely on the reference interval $[q_i^{ref}]$, hence remains constant over whole the computation process. The polynomial parameter vector X relies on the middle and diameter of all the inputs.

2.8. Non linear functions

The proposed method in this paper is based on a polynomial formulation of the constraint and criteria functions. In case of non-linear functions $s([q_i])$, we use a Taylor approximation:

$$s([q_i]) = s(m_i) + \frac{ds(m_i)}{dq}\frac{d_i}{2}[q_i^{ref}] + \ldots + [\varepsilon_s(q_i)]) \tag{18}$$

Where $[\varepsilon_s(q_i)]$ is the approximation error that we normalize using ${[\varepsilon_{s,i}}^{ref}]$, $m_{s,i}$ and $d_{s,i}$ as proposed in Equation (17). The reference error interval $[\varepsilon_{s_i}^{ref}]$ is considered as a new input in Equation (14). In our work, we consider the first order approximation of non-linear functions. Since, we aim at dealing with robotics (non-linear) problems in Section 4, we consider the following non-linear functions: $s([q_i]) \in \{\sin([q_i]), \cos([q_i])\}$, but the method can be applied to any continuous function.

The interval value $[\varepsilon_s(q_i)]$, hence $m_{s,i}$ and $d_{s,i}$, is updated as soon as the original interval $[q_i]$ is modified and ${[\varepsilon_{s,i}}^{ref}]$ is set to $[-1; 1]$. $[\varepsilon_s(q_i)]$ is not considered in the bisection process. The computation of this interval error is quite long and represent a significant part of X hence as soon as this error is small enough (width inferior to 10^{-9}) we do not compute in the next bissected box and consider it as the last computed value to speed up the computation.

2.9. Dealing with memory space

The size of the matrix $\mathbb{B} \in \mathbb{R}^{s\times s}$, hence the number of control point to compute P, relies on the order of Equation (12) and is equal to $s = \prod_{i=1}^{n} m_n$. This can lead to huge value of s and complex computation of the value of X. To avoid this huge number of control point to compute, we create an intermediate variable as soon as $s > 50$.

3. Basis functions

In this Section, we present and introduce several basis functions to compare their performances in Section 4. Those basis functions must fit the properties defined in Equations (6) and (7). Some of the functions (Bernstein, BSplines, MinVolume and ApproxMinVolume) were already compared in [25] in the case of finding the simplexes enclosing 3D objects with minimum volume. A visual representations of those basis function is presente in Table 1, whereas the values of the matrices $\mathbf{b}^{-1}$ is presented in Table 2. Complete values and results are available in [1].

3.1. Bernstein

The Bernstein basis functions are usually defined as follow:

$$b_{i,k}(q) = \frac{k!}{i!(k-i)!}.q^i(1-q)^{k-i} \tag{19}$$

Here, k is the degree of the Bézier curve, q is the parameter, and i ranges from 0 to k. The Bernstein basis functions provide a convenient way to interpolate between control points and define the shape of Bézier curves [4]. Equation (8) is defined for an interval $q \in [0:1]$ and we adapt it in our case for reference intervals $[-1:1]$.

3.2. B-Splines

The B-spline basis functions, can be recursively defined using the Cox-de Boor recursion formula [3]. The base case for $k = 0$ is given by:

$$b_i^0(q) = \begin{cases} 1 & \text{if } \kappa_i \leq q < \kappa_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

For higher degrees k, the recursive definition is as follows:

$$b_i^k(q) = \left(\frac{q-\kappa_i}{\kappa_{i+k}-\kappa_i}\right) b_i^{k-1}(q) + \left(\frac{\kappa_{i+k+1}-q}{\kappa_{i+k+1}-\kappa_{i+1}}\right) b_{i+1}^{k-1}(q) \tag{21}$$

Where $\mathbf{K} = \{\kappa_0, \kappa_1, ..., \kappa_{n+k+1}\}$ is defined as the knot vector that play an important role in determining the continuity and behavior of the B-spline. It was used to compute continuous joint trajectory in motion generation for humanoid robots [11]. In this paper, since we normalize the input as explained in Subsection 2.7, we defined the BSplines basis function based on the knot vector:

$$\mathbf{K} = \{-1-2k, -1-2(k-1), \ldots, -1, 1, \ldots, 1+2(k-1), 1+2k\} \tag{22}$$

Note that if we consider the knot vector $K = \{-1, -1, \ldots, 1, 1\}$, we get the Bernstein functions that was defined in [10].

3.3. MinVolume

Whereas Bernstein and BSplines functions are computed through an iterative process, the authors of [25], proposes to build the basis functions that minimizes the volume of the convex hull defined by the control points. In order to define the basis function, they optimize the roots $q_{i,j}$ and the coefficient α_i such as:

$$\begin{array}{lll}
\underline{\text{if } k \text{ is odd}} & b_i(q) = -\alpha_i(q-1)\prod_{j=1}^{\frac{k-1}{2}}(q-q_{i,j})^2 & i = 0, 2, \dots k-1 \\
& b_i(q) = b_{n-1}(-q) & i = 1, 3, \dots, k \\
\underline{\text{if } k \text{ is even}} & b_i(q) = -\alpha_i(q+1)(q-1)\prod_{j=1}^{\frac{k-2}{2}}(q-q_{i,j})^2 & i \text{ odd integer} \in [0, k/2-1] \\
& b_i(q) = \alpha_i \prod_{j=1}^{\frac{k}{2}}(q-q_{i,j})^2 & i \text{ even integer} \in [0, k/2-1] \\
& b_i(q) = b_{k-1}(-q) & i = k/2+1, \dots, k
\end{array} \tag{23}$$

Their proposed method optimize the coefficients α_i and $q_{i,j}$ in order to ensure the constraint of Equation (6) and (7) and minimize the following Equation to ensure minimal volume of the convex hull:

$$\text{minimize} - det(|\mathbf{B}|)^2 \tag{24}$$

Note that the optimization process performed in [25] does not guarantee global minimum for degrees higher than 4.

3.4. Approximation of MinVolume

The resolution of the optimization problem defined in the previous section is not trivial and considered as impossible for $k\rangle 7$. So, the authors of [25] proposed an algorithm to recursively approximate the roots $(q_{i,j})$, such as:

$$q_{i,j} = sin\left(\frac{c_0\left(i - \frac{s_{j,k}-1}{2}\right) + c_1\left(j - \frac{k-1}{2}\right)}{k + c_2}\right) \tag{25}$$

Where the value of $c_0 \approx 0.2735$, $c_1 \approx 3.0385$ and $c_2 \approx 0.4779$ were found by interpolating the values from the values for the order $k < 7$. The values of α_i are obtained solving the linear constraint of Equation (6).

3.5. MinNorm

In [25], they propose the MinVolume basis function in order to reduce the volume of the convex hull composed by the control point. In this paper, we

propose to use the same optimization process as presented in Subsection 3.3 but trying to minimize the norm of P. Since the polynomial vector X defined in Equation (15) is not constant and not known in advance, we try to find the roots $q_{i,j}$ and the linear coefficient α_i that to ensure the constraint of Equation (6) and (7) and minimize:

$$\text{minimize}||\mathbf{B}^{-1}||^2 \tag{26}$$

3.6. MinVariance

Minimizing the norm of the contol point vector P is relevant if we want to have the minimal value (as closed to zero as possible). However, in our case we are interested into finding the smallest difference between the minimal and maximal value of the control. Hence, we propose to minimize the variance of the vector P in order to minimize the difference between the maximal and minimal value of P, hence we define the basis functions such as it minimize:

$$\text{minimize}||\mathbf{B}^{-1} - Average\left(\mathbf{B}^{-1}\right)||^2 \tag{27}$$

Considering a matrix A :

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,k-1} & a_{1,k} \\ a_{2,1} & a_{2,2} & \dots & a_{2,k-1} & a_{2,k} \\ \dots & \dots & \dots & \dots & \dots \\ a_{k-1,1} & a_{k-1,2} & \dots & a_{k-1,k-1} & a_{k-1,k} \\ a_{k,1} & a_{k,2} & \dots & a_{k-1,k-1} & a_{k,k} \end{bmatrix} \tag{28}$$

we define $Average\,(A)$ such as:

$$Average\,(\mathbf{A}) = \begin{bmatrix} \langle a_1 \rangle & \langle a_1 \rangle & \dots & \langle a_1 \rangle & \langle a_1 \rangle \\ \langle a_2 \rangle & \langle a_2 \rangle & \dots & \langle a_2 \rangle & \langle a_2 \rangle \\ \dots & \dots & \dots & \dots & \dots \\ \langle a_{k-1} \rangle & \langle a_{k-1} \rangle & \dots & \langle a_{k-1} \rangle & \langle a_{k-1} \rangle \\ \langle a_k \rangle & \langle a_k \rangle & \dots & \langle a_k \rangle & \langle a_k \rangle \end{bmatrix} \tag{29}$$

With $\langle a_i \rangle = \frac{1}{k} \sum_{j=1}^{k} a_{i,k}$.

3.7. Recursive

In this part, we present basis functions that could speed up the computation, by producing sparse or repetitive part of the matrix $\mathbf{B}_i^{-1}$. Considering

the following functions for a normalized input:

$$\beta_1(q) = 0.5 - 0.5q \tag{30}$$
$$\beta_2(q) = 0.5 + 0.5q \tag{31}$$
$$\delta_0(q) = 1.0 \tag{32}$$

we built the following recurrence and define the Recursive function such as:

$$\forall i \in \{1, \ldots, k\} \begin{cases} b_i(q) = \delta_{i-1}(q).\beta_1(q) \\ \delta_i(q) = \delta_{i-1}(q).\beta_2(q) \end{cases} \tag{33}$$

A variant of this basis function, called Recursive2, can be defined as:

$$\forall i \in \{1, \ldots, k\} \begin{cases} b_i(q) = \delta_{i-1}(q).\beta_2(q) \\ \delta_i(q) = \delta_{i-1}(q).\beta_1(q) \end{cases} \tag{34}$$

4. Results

4.1. Values and performances of the basis functions matrices

The visual representation of the basis functions can be found on Figure 1 for the order 2 to 6 and the matrix $\mathbf{B}^{-1}$ for order 1 to 3 is available on Table 2. The numerical values of the matrix $\mathbf{B}$ and $\mathbf{B}^{-1}$ from order 1 to 6 are available in [12]. One can notice that the Bernstein, MinVolume, ApproxMinVolume and the MinNorm basis functions produce quiet small values (less than 2) in the $\mathbf{B}^{-1}$ component contrary to the BSplines or Recursives basis function that may produce larger evaluations (this point is clearly evident for higher order). However, the Recursive and Recursive2 basis functions have a triangular part of the matrix $\mathbf{B}^{-1}$ (with only 1 or -1) (lower left part for Recurisve avec upper left part for Recursive2), that will allow to capitalize the computations, thus to speed up the computation. Moreover, those recursive functions also contains integer values, that will avoid rounding error for this matrix.

Table 3 presents the results of the criteria of MinVolume, MinNorm and MinVariance for all the basis functions for ordre 1 to 3 (order 1 to 6 is available in [12]). One can see that the Bernstein, MinVariance, MinNorm, MinVolume and ApproxMinVolume have quite similar performances, whereas Recursive, Recursive 2 and mainly the BSplines have very poor performances regarding the minimized criteria. These results suggest that the evaluationn using those 3 basis function will be the more pessimistic ones.

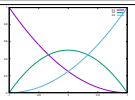
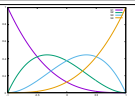
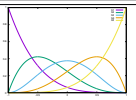
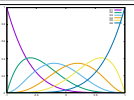
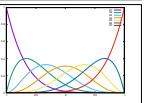
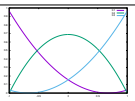
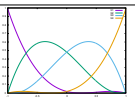
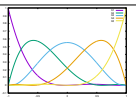
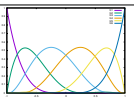
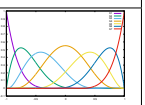
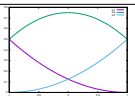
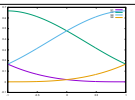
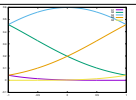
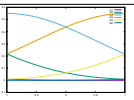
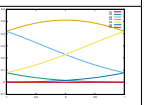
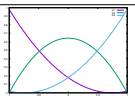
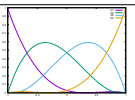
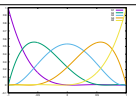
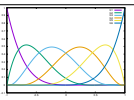
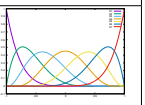
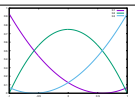
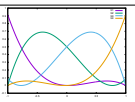
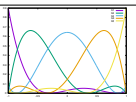
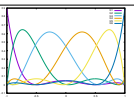
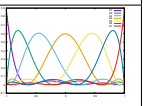
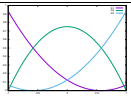
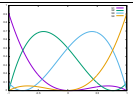
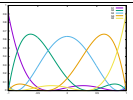
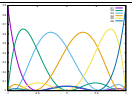
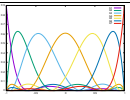
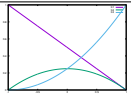
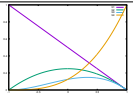
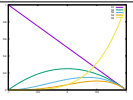
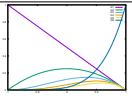
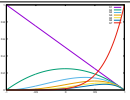
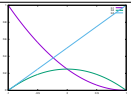
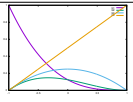
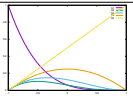
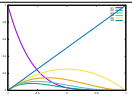
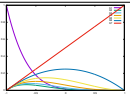
order :	2	3	4	5	6
Bernstein					
MinVariance					
BSplines					
MinNorm					
MinVolume					
ApproxMinVo					
Recursive1					
Recursive2					

Table 1: Representation of the basis functions for the order 2 to 6

4.2. Evaluation on 3D Robotics system

Here, we compare the evaluation performances with the complex model of a 3D robot (KUKA LWR shown in Figure 1). We aim at computing the end effector position and the joint torque given a static posture and studying the diameter of the evaluations regarding the diameter of the input $[q]$. To do so, we use the forward kinematic model and the inverse dynamics model as presented in [22].

In Figure 2, we compare the evaluation performance of those functions regarding several width of the input box $[q]$ by comparing the width of the sum of square torques. The complete results with detail values are available in [12].

type	$\mathbf{B}^{-1}$		
	1	2	3
Bernstein	$\begin{bmatrix} 1.000000 & -1.000000 \\ 1.000000 & 1.000000 \end{bmatrix}$	$\begin{bmatrix} 1.000000 & -1.000000 & 1.000000 \\ 1.000000 & 0.000000 & -1.000000 \\ 1.000000 & 1.000000 & 1.000000 \end{bmatrix}$	$\begin{bmatrix} 1.000000 & -1.000000 & 1.000000 & -1.000000 \\ 1.000000 & -0.333333 & -0.333333 & 1.000000 \\ 1.000000 & 0.333333 & -0.333333 & -1.000000 \\ 1.000000 & 1.000000 & 1.000000 & 1.000000 \end{bmatrix}$
BSplines	$\begin{bmatrix} 1.000000 & -1.000000 \\ 1.000000 & 1.000000 \end{bmatrix}$	$\begin{bmatrix} 1.000000 & -2.000000 & 3.000000 \\ 1.000000 & 0.000000 & -1.000000 \\ 1.000000 & 2.000000 & 3.000000 \end{bmatrix}$	$\begin{bmatrix} 1.000000 & -3.000000 & 7.666667 & -15.000000 \\ 1.000000 & -1.000000 & -0.333333 & 3.000000 \\ 1.000000 & 1.000000 & -0.333333 & -3.000000 \\ 1.000000 & 3.000000 & 7.666667 & 15.000000 \end{bmatrix}$
MinVolume	$\begin{bmatrix} 1.000000 & -1.000000 \\ 1.000000 & 1.000000 \end{bmatrix}$	$\begin{bmatrix} 1.000000 & -1.154701 & 1.000000 \\ 1.000000 & 0.000000 & -0.333333 \\ 1.000000 & 1.154701 & 1.000000 \end{bmatrix}$	$\begin{bmatrix} 1.000000 & -1.149096 & 1.093732 & -1.090170 \\ 1.000000 & -0.592096 & -0.000898 & 0.037136 \\ 1.000000 & 0.592096 & -0.000898 & -0.037136 \\ 1.000000 & 1.149096 & 1.093732 & 1.090170 \end{bmatrix}$
ApproxMinVo	$\begin{bmatrix} 1.000000 & -1.000000 \\ 1.000000 & 1.000000 \end{bmatrix}$	$\begin{bmatrix} 1.000000 & -1.156628 & 1.000000 \\ 1.000000 & -0.000000 & -0.331122 \\ 1.000000 & 1.156628 & 1.000000 \end{bmatrix}$	$\begin{bmatrix} 1.000000 & -1.160380 & 1.102051 & -1.097218 \\ 1.000000 & -0.573815 & -0.001433 & 0.046001 \\ 1.000000 & 0.573815 & -0.001433 & -0.046001 \\ 1.000000 & 1.160380 & 1.102051 & 1.097218 \end{bmatrix}$
MinNorm	$\begin{bmatrix} 1.000000 & -1.000000 \\ 1.000000 & 1.000000 \end{bmatrix}$	$\begin{bmatrix} 1.000000 & -1.043194 & 1.000000 \\ 1.000000 & 0.000000 & -0.556693 \\ 1.000000 & 1.043194 & 1.000000 \end{bmatrix}$	$\begin{bmatrix} 1.000000 & -1.016520 & 1.011659 & -1.006966 \\ 1.000000 & -0.502965 & -0.060748 & 0.366237 \\ 1.000000 & 0.502965 & -0.060748 & -0.366237 \\ 1.000000 & 1.016520 & 1.011659 & 1.006966 \end{bmatrix}$
MinVariance	$\begin{bmatrix} 1.000000 & -1.000000 \\ 1.000000 & 1.000000 \end{bmatrix}$	$\begin{bmatrix} 1.000000 & -1.000000 & 1.000000 \\ 1.000000 & -0.000000 & -0.999900 \\ 1.000000 & 1.000000 & 1.000000 \end{bmatrix}$	$\begin{bmatrix} 1.000000 & -1.003617 & 1.002520 & -1.001329 \\ 1.000000 & -0.536651 & -0.070682 & 0.435290 \\ 1.000000 & 0.536651 & -0.070682 & -0.435290 \\ 1.000000 & 1.003617 & 1.002520 & 1.001329 \end{bmatrix}$
Recursive1	$\begin{bmatrix} 1.000000 & -1.000000 \\ 1.000000 & 1.000000 \end{bmatrix}$	$\begin{bmatrix} 1.000000 & -1.000000 & 1.000000 \\ 1.000000 & 1.000000 & -3.000000 \\ 1.000000 & 1.000000 & 1.000000 \end{bmatrix}$	$\begin{bmatrix} 1.000000 & -1.000000 & 1.000000 & -1.000000 \\ 1.000000 & 1.000000 & -3.000000 & 5.000000 \\ 1.000000 & 1.000000 & 1.000000 & -7.000000 \\ 1.000000 & 1.000000 & 1.000000 & 1.000000 \end{bmatrix}$
Recursive2	$\begin{bmatrix} 1.000000 & -1.000000 \\ 1.000000 & 1.000000 \end{bmatrix}$	$\begin{bmatrix} 1.000000 & -1.000000 & 1.000000 \\ 1.000000 & -1.000000 & -3.000000 \\ 1.000000 & 1.000000 & 1.000000 \end{bmatrix}$	$\begin{bmatrix} 1.000000 & -1.000000 & 1.000000 & -1.000000 \\ 1.000000 & -1.000000 & 1.000000 & 7.000000 \\ 1.000000 & -1.000000 & -3.000000 & -5.000000 \\ 1.000000 & 1.000000 & 1.000000 & 1.000000 \end{bmatrix}$

Table 2: Values of the matrices for the different basis functions $\mathbf{B}^{-1}$ for order 1 to 3.

For the largest input interval (with a diameter of 2), we notice that the state-of-the-art interval implementation produces better (or at least similar) results that the basis-function based implementation. However, for smaller intervals the results of Interval Analysis is always larger than the basis functions evaluation, except for the BSplines basis functions. From those results, we can deduce that the BSplines functions will produces a huge over estimation that may lead to extra number of iterations during the optimization processes (hence larger computation time). The other basis functions may induce similar performances regarding the number of iterations.

4.3. Optimization of 2D and 3D Robotics system

In this part, we evaluate how interesting the use of the basis functions is regarding the computation time and the number of iterations for several input dimensions. We consider two king of problem to generate static posture for robotics system:

- the first one considers a planar robot with 2 to 5 degrees of freedom,

order	type	MinVolume $-det(\|\mathbf{B}\|)^2$	MinNorm $\|\|\mathbf{B}^{-1}\|\|^2$	MinVariance $\|\|\mathbf{B}^{-1} - Average\,(\mathbf{B}^{-1})\,\|\|^2$
1	all	-0.25	4	2
2	Bernstein	-0.0625	8	4.66667
	MinVariance	-0.101637	7.53553	**3.73773**
	BSplines	-0.00390625	30	18.6667
	MinNorm	-0.094799	**7.48641**	3.79204
	MinVolume	**-0.105469**	7.77778	3.85185
	ApproxMinVo	-0.105467	7.78522	3.85684
	Recursive	-0.015625	17	13.3333
	Recursive2	-0.015625	17	13.3333
3	Bernstein	-0.0197754	12.4444	8
	MinVariance	-0.0776098	10.9429	**5.99574**
	BSplines	-1.69542e-06	609.778	552
	MinNorm	-0.0703759	**10.9231**	6.01885
	MinVolume	**-0.110146**	12.1142	6.91991
	ApproxMinVo	-0.110036	12.1925	6.98117
	Recursive	-0.000244141	96	90
	Recursive2	-0.000244141	96	90

Table 3: Values of the different criteria from the basis function from order 1 to 3.

- the second one considers a 3D robot inspired for the KUKA-LWR robot with 4 or 6 degrees of freedom.

The results presented in this paper are based on the average of several cases (different position of the end effector and minimizing sum of square joint position or torques), the full results are presented in [12].

As suggested previously, the BSplines basis function produces the worst results regarding the other basis functions, we exclude it from the description of the results presented hereafter.

Figure 3 presents the ratio of the number of iterations of the optimization process regarding the use of natural inclusion function. We can see that all the basis functions produce a lower number of iterations. The gain is less important for simple (with low number of degree of freddom) problems. Regarding the different basis function, we can state that they produce nearly the same results, even if the Bernstein basis function seems to be the best.

Figure 4 presents the results regarding the computation time (without

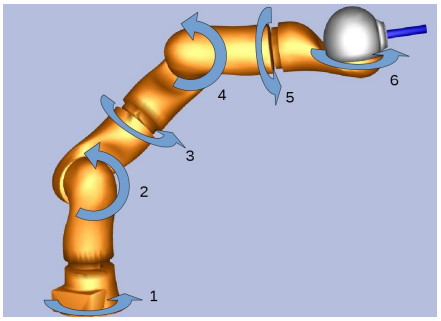


Figure 1: The 3D robot: KUKA LWR

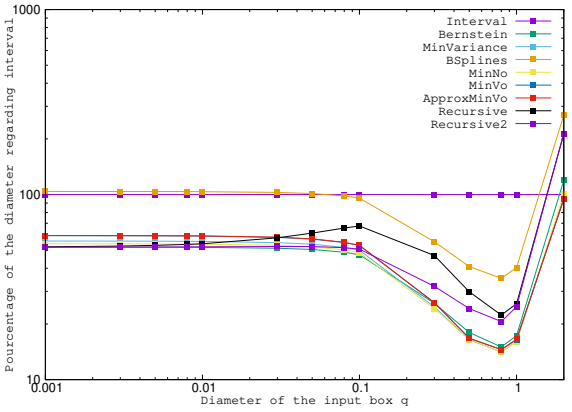
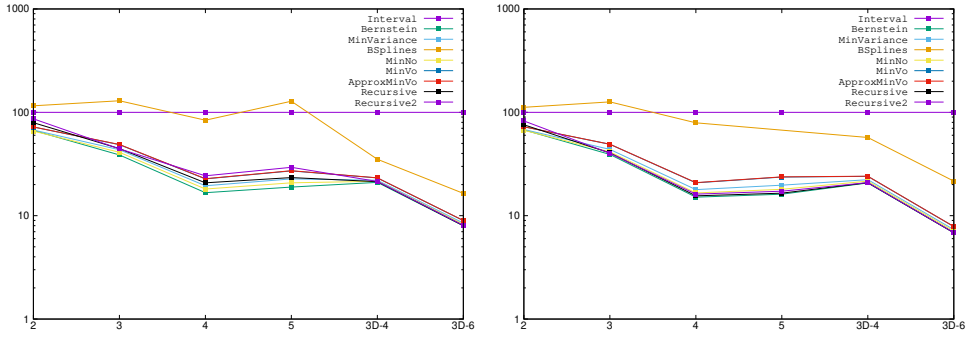


Figure 2: Representation of the diameter of the evaluation using basis function regarding interval analysis.

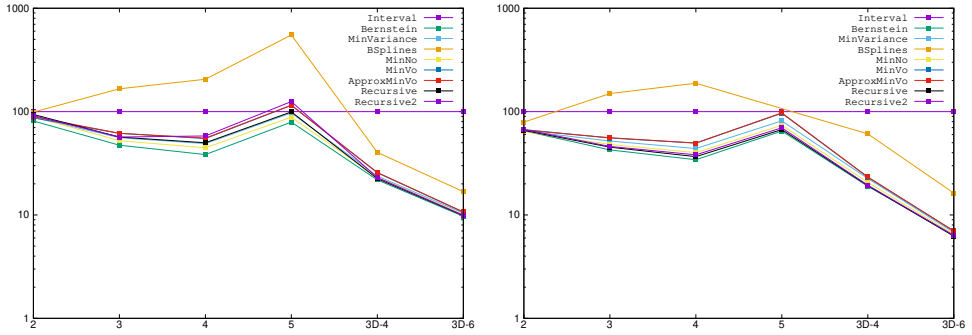
considering the preparation phase). We ca also notive that the performances are enhanced with the problem complexity (problem 3D-4 and 3D-6 are the most complex problems). We can also see that the basis function produce better computation time with a lower precision. This can be explained by Figure 2, for which we can see that the evaluation for large interval is at the advantage of the natural inclusion functions.

Figure 5 presents the results regarding the computation time taking into account the preparation phase that goes from a few seconds for 2D case to several minutes for the 3D cases. As depicted previsouly, the advantage of using basis function is more important when considering complex systems (3D-6), considering that the average computation for the 3D-4 case is of



(a) precision of 0.01 (b) precision of 0.001

Figure 3: Comparison of the number of iterations



(a) precision of 0.01 (b) precision of 0.001

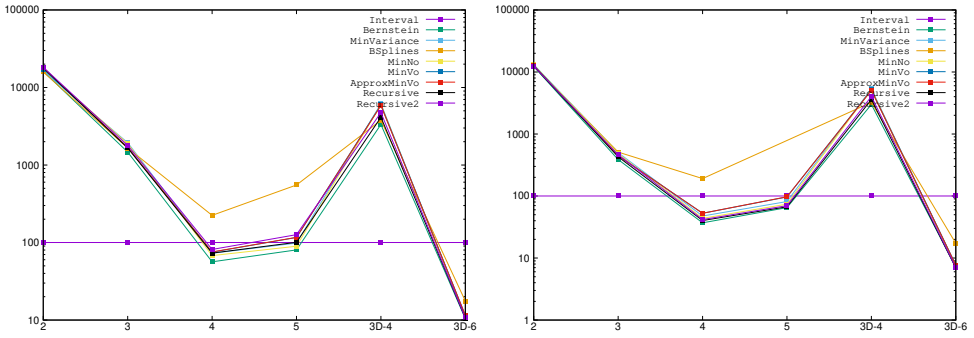
Figure 4: Comparison of the computation time.

several hours and several days for the 3D-6 case.

Regarding the computation time of Figure 4 and 5, it appears that all the basis function produce similar results even if we can notice that the Bernstein basis function is the best one.

5. Conclusion and future works

We present a method to evaluate function over multi-dimension interval using basis function. We introduce and compare several basis function performance to solve optimization problem. It seems that the initial choice of [10] of using Bernstein function produces the best results in terms of number of iterations and of computation time. However, this conclusion is not intuitive



(a) precision of 0.01 (b) precision of 0.001

Figure 5: Comparison of the total computation time (including preparation phase).

regarding the comparison of the criteria (min volume, norm or variance). However, it seems that the Bernstein basis function is the best compromise between those three criteria.

In order to decrease the pessimism, it could be interesting to consider different kind of evaluation during the same optimization process: using natural inclusion function for large interval, using MinNorm to compute the errors due to the polynomial approximal and Bernstein or MinVariance in the other cases.

In this paper, we focussed on finding the best basis function for the evalution function, however, to get efficient and tractable optimization algorithm regarding the computation time we must also take part of a multithreading implemtation, consider more advanced bisection process and use warm start. Some of this bisection process may be oriented by the computation of the control point to guess the part of the search space containing the optimal solution.

Acknowledgment

We are grateful to the Mésocentre Clermont-Auvergne of the Université Clermont Auvergne for providing help, computing and storage resources facilities.

[1] https//github.com/lengagne/polynomial-interval-optimization.

[2] Gilles Chabert and Luc Jaulin. computing the pessimism of inclusion functions. *Reliable Computing*, 13:489–505, 2007.

[3] Carl de Boor. Package for calculating with b-splines. *SIAM Journal on Numerical Analysis*, 14:57, 10 1973.

[4] Gerald E. Farin, Josef Hoschek, and Myung-Soo Kim. Handbook of computer aided geometric design. 2002.

[5] E. Hansen and G.W. Walster. *Global optimization using interval analysis*. Marcel Dekker, 2nd edition, 2004.

[6] Chaoem Jiang, Xue Han, Fengjiao Guan, and Yonghong Li. An uncertain structural optimization method based on nonlinear interval number programming and interval analysis method. *Engineering Structures*, 29(11):3168 – 3177, 2007.

[7] Jovana Jovic, Adrien Escande, Ko Ayusawa, Eiichi Yoshida, Aberrahmane Kheddar, and Gentiane Venture. Humanoid and human inertia parameter identification using hierarchical optimization. *IEEE Transactions on Robotics*, 32(3):726–735, June 2016.

[8] Rawan Kalawoun, Sébastien Lengagne, Francois Bouchon, and Youcef Mezouar. Bsplines properties with interval analysis for constraint satisfaction problem application in robotics. In *15th International Conference on Intelligent Autonomous Systems IAS-15*, june 2018.

[9] Craig Lawrence, Jian L. Zhou, and Andre L. Tits. *User's Guide for CFSQP Version 2.5 A C Code for Solving (Large Scale) Constrained Nonlinear (Minimax) Optimization Problems, Generating Iterates Satisfying All Inequality Constraints*. Electrical Engineering Department, Institute for Systems Research University of Maryland, College Park, MD 20742.

[10] Sebastien Lengagne, Rawan Kalawoun, François Bouchon, and Youcef Mezouar. Reducing pessimism in Interval Analysis using Bsplines Properties Application to Robotics. *Reliable Computing*, 27:63–87, July 2020.

[11] Sébastien Lengagne, Joris Vaillant, and Eiichi Yoshida. Generation of Whole-body Optimal Dynamic Multi-Contact Motions. *The International Journal of Robotics Research*, page 17, April 2013.

[12] Sébastien Lengagne. Detailed results of the comparison of basis functions used in interval-based optimization process https//uca.hal.science/hal-04320488.

[13] Charles F.Van Loan. The ubiquitous kronecker product. *Journal of Computational and Applied Mathematics*, 123(1):85 – 100, 2000. Numerical Analysis 2000. Vol. III Linear Algebra.

[14] Hongliang Ma, Shijie Xu, and Yuying Liang. Global optimization of fuel consumption in j2 rendezvous using interval analysis. *Advances in Space Research*, 59(6):1577 – 1598, 2017.

[15] Ramon E. Moore and Fritz Bierbaum. *Methods and Applications of Interval Analysis (SIAM Studies in Applied and Numerical Mathematics) (Siam Studies in Applied Mathematics, 2.)*. Soc for Industrial & Applied Math, 1979.

[16] Lukas Netz. Using horner schemes to improve the efficiency and precision of interval constraint propagation. 2015.

[17] A. Neumaier. *Interval methods for systems of equations*. Cambridge university press, Cambridge, 1990.

[18] Arnold Neumaier. Taylor forms - use and limits. *Reliable Computing*, 2003.

[19] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, August 2000.

[20] Carlos Pérez-Galván and I. David L. Bogle. Global optimisation for dynamic systems using interval analysis. *Computers and Chemical Engineering*, 2017.

[21] Kathrin Schäcke. On the kronecker product, 2013.

[22] Bruno Siciliano and Oussama Khatib. *Springer Handbook of Robotics*. Springer-Verlag, Berlin, Heidelberg, 2007.

[23] T. Sunaga. Theory of interval algebra and its application to numerical analysis. *RAAG Memoirs, Ggujutsu Bunken Fukuy-kai*, 2:547–564, 1958.

[24] Mohammad Taherdangkoo, Mahsa Paziresh, Mehran Yazdi, and Mohammad Bagheri. An efficient algorithm for function optimization: modified stem cells algorithm. *Open Engineering*, 3(1):36–50, 2013.

[25] Jesus Tordesillas and Jonathan P. How. Minvo basis finding simplexes with minimum volume enclosing polynomial curves. *Computer-Aided Design*, 151:103341, 2022.

[26] Andreas Wächter and Lorenz T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106:22–57, 2006.

[27] Jinglai Wu. *Uncertainty analysis and optimization by using the orthogonal polynomials*. PhD thesis, 2015.

Appendix A. Optimization algorithm

The detail of the branch and bound algorithm is detailed in Algorithm 1. One can notice that the objective function $F(q)$ is evaluated at the beginning as soon as we found at least one feasible solution.

Algorithm 1 Algorithm for optimization process using bisection process.

Require: Initial research space $\mathbf{Q}$, desired precision ε

1: initialization: $\mathcal{C}$: ($\forall j$ $\mathcal{C}_j$ = **false**), $\tilde{f} = \infty$, Q.push($\mathbf{Q}, \mathcal{C}$)
2: **while** Q is not empty **do**
3: get element $[q, \mathcal{C}] = Q$.pop()
4: **if** $\tilde{f} \neq \infty$ **then**
5: evaluate objective $[f] = [\underline{f}; \overline{f}] = \mathcal{F}([q])$
6: **if** $\underline{f} > \tilde{f}$ **then**
7: Stop and go to line 2
8: **end if**
9: **end if**
10: **for** $\forall j$ **do**
11: **if** (C_j = **false**) **then**
12: evaluate constraints $[g_j] = \mathcal{G}_j([q])$
13: **if** ($[g_j] \cap [\underline{g}_j : \overline{g}_j] = \emptyset$) **then**
14: Stop and go to line 2
15: **else if** $[g_j] \not\subset [\underline{g}_j : \overline{g}_j]$ **then**
16: BISECTION($[q], \mathcal{C}$) and go to line 2
17: **else**
18: $\mathcal{C}_j$ = **true**
19: **end if**
20: **end if**
21: **end for**
22: **if** $\tilde{f} = \infty$ **then**
23: evaluate objective $[f] = [\underline{f} : \overline{f}] = \mathcal{F}([q])$
24: **end if**
25: **if** $\overline{f} < \tilde{f}$ **then**
26: $[\tilde{f}, \tilde{q}] = [\overline{f}, q]$
27: BISECTION($[q], \mathcal{C}$)
28: **end if**
29: **end while**
30: **Return** Optimal box $\tilde{q}$
31:
32: **function** BISECTION($[q], \mathcal{C}$)
33: **if** $diam([q]) \rangle \varepsilon$ **then**
34: $\{q_1, q_2\} = cut(q)$
35: Q.push($q_1, \mathcal{C}$)
36: Q.push($q_2, \mathcal{C}$)
37: **end if**
38: **end function**

B.3 Transfert de compétences

Article : S. Beaussant, **S. Lengagne**, B. Thuilot, O. Stasse. *Towards Zero-Shot Cross-Agent Transfer Learning via Latent-Space Universal Notice Network.* Robotics and Autonomous Systems, In press, 184, pp.104862.

Towards Zero-Shot Cross-Agent Transfer Learning via Latent-Space Universal Notice Network

Samuel Beaussant[a,*], Sebastien Lengagne[a], Benoit Thuilot[a], Olivier Stasse[b]

[a]*Université Clermont Auvergne, Clermont Auvergne INP, CNRS, Institut Pascal, F-63000 Clermont-Ferrand, France*
[b]*LAAS-CNRS Université de Toulouse, CNRS, Toulouse, France*

Abstract

Despite numerous improvements regarding the sample-efficiency of Reinforcement Learning (RL) methods, learning from scratch still requires millions (even dozens of millions) of interactions with the environment to converge to a high-reward policy. This is usually because the agent has no prior information about the task and its own physical embodiment. One way to address and mitigate this data-hungriness is to use Transfer Learning (TL). In this paper, we explore TL in the context of RL with the specific purpose of transferring policies from one agent to another, even in the presence of morphology discrepancies or different state-action spaces. We propose a process to leverage past knowledge from one agent (source) to speed up or even bypass the learning phase for a different agent (target) tackling the same task. Our proposed method first leverages Variational Auto-Encoders (VAE) to learn an agent-agnostic latent space from paired, time-aligned trajectories collected on a set of agents. Then, we train a policy embedded inside the created agent-invariant latent space to solve a given task, yielding a task-module reusable by any of the agents sharing this common feature space. Through several robotic tasks and heterogeneous hardware platforms, both in simulation and on physical robots, we show the benefits of our approach in terms of improved sample-efficiency. More specifically we report zero-shot generalization in some instances, where performances after transfer are recovered instantly. In worst case scenarios, performances are retrieved after fine-tuning on the target robot for a fraction of the training cost required to train a policy with similar performances from scratch.

Keywords: Reinforcement Learning, Transfer Learning, Robot manipulation

1. Introduction

Deep Reinforcement Learning came to light thanks to the works on Deep Q-Networks by [1, 2], multiple general and versatile RL solvers have been proposed to improve the performance of RL agents [3, 4, 5]. These algorithms demonstrated impressive achievements on a vast range of robotic tasks such as in-hand manipulation for rubik's cube solving [6], control of quadrupedal gaits [7], cloth manipulation [8] or swing-peg-in-hole manipulation [9]. However, these spectacular results overshadow the often massive cost required to train these models. Indeed, despite great achievements, RL methods still suffer from low sample efficiency, which means that a large amount of interactions (at least in the order of millions) with the environment is needed to obtain a high-performance policy [10]. One of the main explanations behind such a high computational cost, is the fact that most agents are trained from scratch, without any pre-training or prior knowledge of the task or environment.

Consequently, each time the agent is required to learn a task, it has to first discover how to articulate its body appropriately and avoid undesirable joint configurations, while also trying to act optimally with respect to a reward function. Finding a practical and general approach for pre-training and transfer learning in the context of RL policies, is still an open research problem. Analogous to deep learning in Computer Vision (CV), it would be interesting to be able to pre-train a model on a given domain (i.e large image dataset for CV or agent for RL) such that it performs well or can be efficiently fine-tuned on another domain for the same task. For instance, previous work in CV has demonstrated that pre-training on ImageNet can enhance performance when fine-tuning for a new domain, such as tumor detection. In RL, this concept could involve pre-training a model on a 6-DoF (Degrees of Freedom) UR10 robot and then transferring it to a 7-DoF Panda robot for the same manipulation task.

In this paper, we specially focus on cross-robot transfer learning and wish to enable differently shaped robots to share knowledge for a more sample-efficient RL training. Our current approach is based on a functional and hierarchical decomposition of the policy, with at the center, a high-level task-specific module, and on each side, acting as an interface, agent-specific modules, as depicted in Figure 1. This functional decomposition approach was first introduced and explored in [11]. The agent-agnostic task module is learned via RL on a given task while the agent's modules, representing its kinematic or dynamic models, are obtained with analytical methods or learned. [11] only considered explicit agent-agnostic information such

*Corresponding author.
Email addresses: samuel.beaussant@gmail.com (Samuel Beaussant), sebastien.lengagne@uca.fr (Sebastien Lengagne), benoit.thuilot@uca.fr (Benoit Thuilot), olivier.stasse@laas.fr (Olivier Stasse)

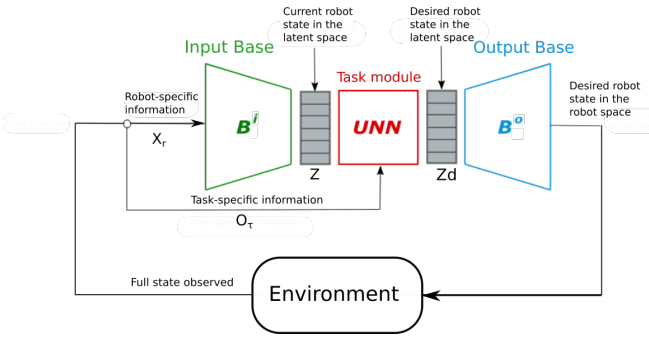


Figure 1: Latent space UNN pipeline. The state provided by the environment is decomposed into robot-specific and task-specific information, which are fed to the corresponding modules. The task module (in red) is operating inside a latent space provided by the two outter modules which are robot-specific.

as effector position or velocity for interfacing the task-module with the robot. In this work, we go a step further and generalize this method by introducing an agent-agnostic latent space in place of the Cartesian space for knowledge transfer. Thanks to our main contribution, namely, the Latent-Space Universal Notice Network the different modules come together to enable efficient cross-robot transfer learning. We demonstrate the effectiveness of our proposed TL method through 3 increasingly complex robotic tasks on 3 dissimilar robots in simulation. We also provide results for the medium difficulty task in the physical domain on 2 of the considered robots.

Practical use cases of cross-robot task transfer may include for instance :

- Changing an old and worn out robot on an assembly line with a new but different one. As they ultimately achieve the same task, we can transfer the task module from the old to the new robot to speed up the replacement process.
- Learn and prototype a task on an unreliable but cheap robot before transferring it to an industrial-grade but more expensive robot. This way we prevent potential hazard due to the inherent unpredictability of RL training on the target robot.
- Similarly, we could learn and prototype a task on a simulated robot before transferring it to any physical robot equipped with the appropriate agent's modules.

In the next section, we formalize the cross-agent transfer learning problem using prior works. Following this, we introduce our contribution, the Latent Space Universal Notice Network, which is a framework for cross-agent transfers. Then, we contextualize our work with relevant related literature in Section 4. The implementation and training of our method are then detailed in Section 5. Empirical results supporting our claims are discussed and analyzed in Section 6 and Section 7. Lastly, we end this paper by discussing some the current limitations, including over-fitting issues and the state-paring requirement, before concluding.

2. Problem formulation

2.1. Markov Decision Process

We begin by formally defining the mathematical framework used to model our sequential decision-making problems. They can be framed as a Markov Decision Process (MDP) which is a discrete-time stochastic control process typically defined by a 5-tuple $MDP =< S, A, P, R, \rho_0 >$:

- S is the set of valid (continuous) states (the state space),
- A is the set of valid (continuous) actions the decision-maker can take (the action space),
- $P(s_{t+1}|s_t, a_t)$ is the transition probability distribution describing the MDP dynamic. It dictates its evolution conditioned on both the present MDP state s_t and the agent's action a_t.
- $R : S \times A \times S \mapsto \mathbb{R}$ is the reward function giving a feedback g_t to the decision-maker after transitioning from state s_t to state s_{t+1}, due to action a_t,
- ρ_0 is the initial state distribution of the MDP.

In a MDP, the goal of the agent is to find a policy that maximizes over an horizon T its cumulative discounted reward G_t defined as :

$$G_t = \sum_t^T \gamma^t g_t \tag{1}$$

where $\gamma \in [0; 1]$ is a hyper-parameter weighting distant rewards.

2.2. Cross Agent Transfer Learning

To frame our transfer learning problem, we take inspiration from a recent survey on TL in robotics [12]. Three core concepts are needed to formally define our objective : robot, environment and task. A robot r embodies the learning agent and enables it to act on its environment. It is defined by its dynamic, its morphology and sensor modalities. The environment E can be defined as the physical or virtual local space containing everything the robot can reach or interact with. Together, these two notions form a domain D which can be written as a 4-tuple:

$$D_{r,E} =< S_D, A_D, P_D, \rho_{0_D} > \tag{2}$$

also following the formalization of [13]. In other words, a domain fully describes the robot r, its embodiment and its dynamic, as well as its surroundings. But it does not imply any particular behavior as no reward function is specified. This is where the concept of a task $\mathcal{T}$ comes into place. It defines the goals we want the robot to achieve as well as the overall desired behavior, and can be represented by the appropriate reward function $R_\mathcal{T}$. Finally, we can define a MDP by combining both concepts:

$$D_{r,E} \bigcup R_\mathcal{T} =< S_D, A_D, P_D, R_\mathcal{T}, \rho_{0_D} > \tag{3}$$

In our case, we consider 3 different domains and 3 different tasks which combine to 9 MDPs as depicted in Figure 2. Our goal is to transfer task-specific knowledge to a target robot, enabling it to perform a task already known by other source robots. We define more precisely the cross-agent transfer problem as follows, slightly adapted from [14] :

Definition 1: Cross Agent Transfer Learning Given a set of n source domains $\{D_{R_s^i,E}\}_{i=1:n}$, a target domain $D_{R_t,E}$ and a task $\mathcal{T}$, Cross-Agent Transfer Learning (CATL) aims to derive an optimal policy $\pi^*_{D_{R_t,E},\mathcal{T}}$ for the target MDP more efficiently than learning it from scratch, by leveraging information I_s^i from source MDPs as well as information I_t from the target MDP.

Definition 1 is rather general but conveys the idea that we transfer knowledge from source domains to the target domain with respect to a given task. It also makes no assumption about the nature of the information I transferred. As such it could be a teacher neural network's hidden state as [15], Q-Values as in [16], expert demonstrations or a partial/complete policies. As discussed and analyzed in [10, 12], it is not trivial to define what kind of knowledge should be transferred and through which medium.

A recent but popular and successful approach to address CATL, is to find a single policy that can manage a variety of robot hardware configurations for a given task $\mathcal{T}$. In this case, what we are transferring is the policy. More formally, if we denote by $\pi^*_{\mathcal{T}}$ such a policy and $\Re$ the set of domains (or robot morphologies) considered, we wish to solve :

$$\pi^*_{\mathcal{T}} = \arg\max_{\pi} \mathbb{E}_{r\sim\Re}[G^{\mathcal{T}}(r)] \tag{4}$$

where $G^{\mathcal{T}}(r)$ is the discounted return for robot r on task $\mathcal{T}$. In other words, $\pi^*_{\mathcal{T}}$ should be optimal and robot-agnostic.

In this work, we will consider that $\Re$ is composed of robots functionally and morphologically different. In other words, they differ both by their segment length and by their number of joints. Let D_s represent the source domain and D_t the target domain. To provide a general CATL framework, we make minimal assumptions and consider the possibility that $dim(A_s) \neq dim(A_t)$ and $dim(S_s) \neq dim(S_t)$. This CATL setting is challenging but also more realistic. It involves developing algorithms that can abstract relevant features from the source domain, map these features onto the target domain, and adapt the knowledge accordingly. In the next section, we present our method to generalize learned behaviors across various platforms.

3. Latent Space UNN (LS-UNN)

In this section, we describe our CATL method : the Latent Space Universal Notice Network (LS-UNN) and the different parts involved. It is based on two key concepts: modularity and latent space alignment. We start by giving a conceptual explanation of both notions to provide a high-level overview of our framework and how each part fits together. Finally, we explain how the different modules come together to enable efficient cross-agent transfer learning.

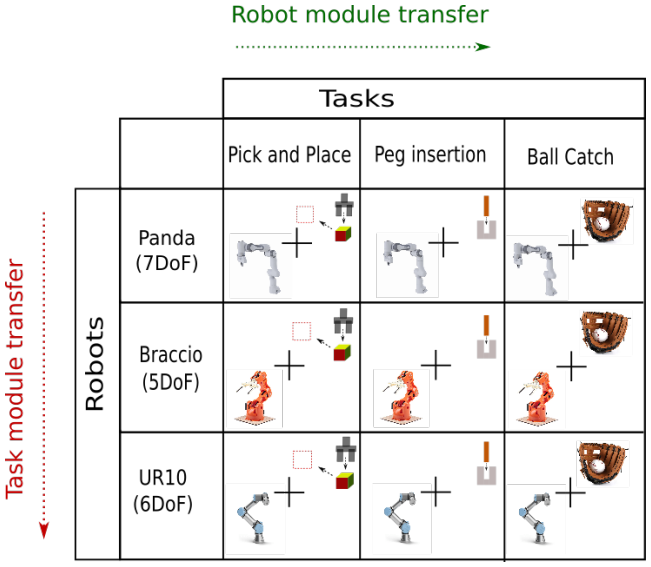


Figure 2: Considered Markov Decision Processes. The setup considers 3 robots with varying DoF in our pool each of them performing 3 tasks. This amounts to a combination of 9 MDPs in total.

3.1. Motivations

Prior works [17, 18, 19, 20] solved Equation (4) and enforced robot-agnosticity by training alternatively $\pi^*_{\mathcal{T}}$ on each robot of a training-set contained within $\Re$. Then, the remaining robots of $\Re$ were used as a test-set to assess the generalizing abilities of the policy $\pi^*_{\mathcal{T}}$. However, this potentially requires a large amount of training on a multitude of robot morphologies, leading to a training overhead that might overshadow the benefits of transfer. If one spends several hours designing robots and training $\pi^*_{\mathcal{T}}$ on a large set of source robots instead of training from scratch for a couple of hours directly on the target robot, it is worth wondering if these methods can have a practical usefulness.

Instead of training a policy on each robot, hoping that it will generalize to unseen hardwares, we draw inspiration from [21] and create a latent space shared by the considered robots that will serve as a support for the knowledge transfer. A latent space is a compressed but meaningful representation of some high-dimensional data, which encodes enough information to re-generate the original data. In our case, the embedded data will be the robots state. By design, this common feature space Z will be robot-agnostic, which implies that a policy trained within will effectively generalize to all the robots sharing it. Consequently, we avoid the costly requirement of training the policy on a large robots set and solve Equation (4) by training on a single robot. Inspired by prior works on CATL [11, 18], we also take a modular approach and decompose the policy $\pi^*_{\mathcal{T}}$ into a task-specific part and a robot-specific part as depicted in Figure 1. This hierarchical modularity makes it possible to decouple the high-level task resolution from the low-level robot control for efficient cross-robot transfer.

3.2. The UNN task module

The UNN task module is the transferable part of the modular policy (depicted in red in Figure 1). The task module operates in the robot-agnostic latent space Z described in the previous section. By doing so, we make sure that the task-module is not concerned with the details of actuation and focus on the high-level decisions. As such, the task-solving process is largely agnostic to the robot morphology and suits any of the targeted robots for transfer. For example, solving a pick and place task may require low-level robot-specific motor commands, but the high level process will roughly be the same : first move the effector close to the pick target, grasp the object and get it at the desired location. By training in a robot-agnostic latent space, we abstract away the robot-specific part during training, resulting in a transferable task-specific module. There is no predefined semantic meaning to the latent variables as it is an abstracted representation of the robots states. Instead, the latent features are learned to best represent the robots state-action space.

3.3. The robot modules B_i and B_o

As explained earlier, the task module is not aware of what kind of low-level robot-specific motor commands will actually move the robot or even what is the exact robot state. This part is handled by two robot-specific modules which are responsible for interfacing the task-module with the actual robot. We repurpose the semantic of [11] and call the robot-specific modules the *bases*. They serve two main purposes :

- Output base B^o : Translate the high-level and latent commands of the task module into robot's actuation.
- Input base B^i : Embed the robot state into its corresponding latent representation to feed the task module.

Similarly, the robot modules are task-agnostic and can be reused for other tasks. Moreover, since we are dealing with transfer for robots with heterogeneous state and action dimensions, we cannot have a single policy fitting every state-action space dimension. This issue is solved by having each robot represented by its own specific modules adapted to its control dimensionality.

3.4. Combining task and robot modules for efficient transfer

A complete and functional policy for a specific MDP requires combining the corresponding robot modules and task module as depicted in Figure 2. When varying the task, we use the same robot modules. Similarly when varying the robot we transfer the task module. Figure 1 illustrates the full policy architecture with the corresponding input/output of each module. We first split the full observed state s into robot-specific and task-specific information:

$$s = \{x_r, o_\tau\} \tag{5}$$

The robot-specific information x_r holds the observation regarding the state of the robot itself such as joint position or joint velocity. On the other hand, everything the agent needs to know about the state of the task is contained in the task-specific information o_τ. It could include objects location and velocity and more broadly any robot-agnostic data. The robot specific vector x_r is translated into a latent and robot agnostic vector z by means of the input base $B^i : X_r \mapsto Z$

$$z = B^i(x_r) \tag{6}$$

The task module is called the Universal Notice Network (UNN) and is responsible for solving the task. It operates inside the shared latent space $U : Z \times O^\mathcal{T} \mapsto Z$ and maps the current robot latent state z as well as the task specific observations o_τ, to a desirable robot-agnostic latent state z_d :

$$z_d = U(z, o_\tau) \tag{7}$$

Finally, we get the effective desired robot-specific state x_r^d by projecting z_d back to the robot space through the use of the output base $B^o : Z \mapsto X_r$:

$$x_r^d = B^o(z_d) \tag{8}$$

The robot-specific desired state is then used to derive an action using a controller. In summary, the modular policy π_τ^* is the composition of the 3 modules:

$$\pi_\tau^* = B^o \circ U \circ B^i \tag{9}$$

or more explicitly:

$$\pi_\tau^* = B^o(U(B^i(x_r), o_\tau)) \tag{10}$$

Each of the three mappings discussed above is learned by a feed-forward neural network. For the rest of the paper, we denote by B^i_ϕ,B^o_θ,U_ψ, the three modules respectively parameterized by the weights ϕ,θ,ψ. Now that our framework is properly explained, we proceed to the next section where we re-contextualize our work in relation to related works.

4. Related works

Cross-agent transfer is concerned with reusing and leveraging knowledge and past experience from a morphologically distinct agent in order to speed up the learning of a target agent. It is a very promising avenue of research in the quest for mitigating the notorious sample-inefficiency of RL. Despite its appealing practical applications, it is still an understudied subfield of transfer learning and only a handful of prior works has tackled the cross-agent transfer learning problem. In this section, we review relevant works on the topic of transfer learning in robotics. For the sake of clarity, we propose to classify the methods discussed in two broad categories that we introduce in the next sub-sections and we analyze how this paper fits into that body of literature.

4.1. Learning a robot-agnostic policy

In this section, we present methods that achieve cross-agent transfer learning by training a single policy on a large variety of robot morphologies which can be thought as a form of domain randomization. Their main goal is to learn an agent-agnostic

policy, that is a policy that can control a wide range of robots, regardless of their shapes.

One of the earliest attempt was a work by Colin Devin and Abhishek Gupta [18]. They proposed a novel modular architecture to hierarchically decouple low-level robot-specific actuation from the robot-invariant and high-level task resolution. This results in robot-specific and task-specific modules that are combined and trained together until a latent space common to all modules is found. If the training set is large enough, the learned representation may enable zero-shot transfer to unseen robots or task modules. Similar in spirit to previous work, Hardware Conditioned Policies (HCP) from [17] trains a single policy on a variety of body configurations in order to achieve agent-agnostic control by conditioning the policy on a vector representing the robot hardware. Their method successfully learned a policy able to control 9 different variations of the same robot but achieves low success rate on the peg-in hole task. An effective and meaningful extension to their work is [19] which added an adversarial network responsible for sampling morphology for the policy network to train on. They report improved results both on learning speed and zero-shot performance when compared to HCP. A different but related approach [20] frames the challenging problem of hardware-agnostic policy training as a few-shot learning problem where the goal is to adapt an action-selection policy to a new robot from small amounts of data. They address this problem using a gradient-based meta-learning method [22]. However, their method was not designed to adapt policies for robots with different numbers of DoFs. More recently, [23] managed to train a high-capacity transformer-based model to learn a "generalist" robot policy from a large-scale robot-learning dataset. Their method demonstrates impressive results on 9 different robots for a wide array of tasks. However, running inference or fine-tuning their model on out-of-distribution robots, could be prohibitively costly for entities with less resources.

All the previously mentioned methods require training to convergence a policy on multiple possible robots of the training set before a shared representation can emerge and be used for transfer on an unseen robot/task. In practice, it is therefore not clear if their approaches exhibit an actual learning speed up over a simple training from scratch on the target robot or task, as the amount of training needed prior to transfer could be significant. Furthermore, designing variations of the robots in the training set to help the policy generalize is not a trivial task as explained in [19]. It requires expert knowledge to determine which physical parameters should vary and careful crafting of their distributions. Poorly handled parameters randomization could even be detrimental for learning. In contrast, our method leverages the creation of a robot-agnostic latent space ensuring instant generalization to new morphologies by encoding time-aligned trajectories of a primitive task execution for the robots considered. There is no need for any physical parameters randomization and we only need to train the task module once, on one robot.

4.2. Using a common feature space

Our proposed robot-agnostic latent space is similar to the invariant feature space introduced in [21]. However, their work is focused on transferring knowledge between robots through reward shaping by adding an "alignment" term to guide the student agent during training. As a consequence, their method does not enable zero-shot transfer unlike ours. Policy stitching [24] is a novel approach building on [18] to address its over-fitting and lack of zero-shot generalization issues. They introduce the use of anchor states to align the representations of policy modules. Their method improves both zero-shot and few-shot performance on challenging 3D manipulation tasks compared to prior works but transfers are limited to similar tasks. Recent work [25] proposes a framework to cross-robot skill transfer by learning a cycle generative model to map skill representations between robot domains. They further improve policy learning efficiency by incorporating a relative entropy term, regularizing the RL objective using skill priors. Their method enhances the sample efficiency of training policies in the target domain but does not support zero-shot transfers. Another line of works [15] proposed to augment representations in the layers of the student network with useful representations from the layers of a teacher network, by using lateral connections between their respective policy and value networks. As such, knowledge can "flow" between a teacher pre-trained on a related task and a single student agent. While significantly improving sample-efficiency, their method does not exhibit zero-shot performance. Authors of [26] proposed a method to hierarchically decouple high-level transferable knowledge from low-level robot-specific knowledge. Their method is conceptually very similar to ours and [11], but fundamental differences exist:

- They consider a goal space (i.e a robot-agnostic space) which needs to be defined by an expert (effector position/velocity, torso position etc..) instead of learned features.
- They minimize the mutual information between the morphology and the behavior to promote robot-agnosticity of the high-level policy. In our case, robot-agnosticity is ensured by leveraging a shared latent space.

To address some of the limitations presented in this section, the LS-UNN builds upon the strengths of previous works and incorporates new strategies to overcome the shortcomings. In the following section, we go into details on the training pipeline used.

5. LS-UNN : modules training

So far we have detailed the full policy pipeline as well as each component assuming access to a shared latent space Z. However, this robot-agnostic space needs to be obtained prior to training and transferring the UNN. In this section, we explicit the latent space creation and alignment process as well as the task-module training process. First, we consider the case

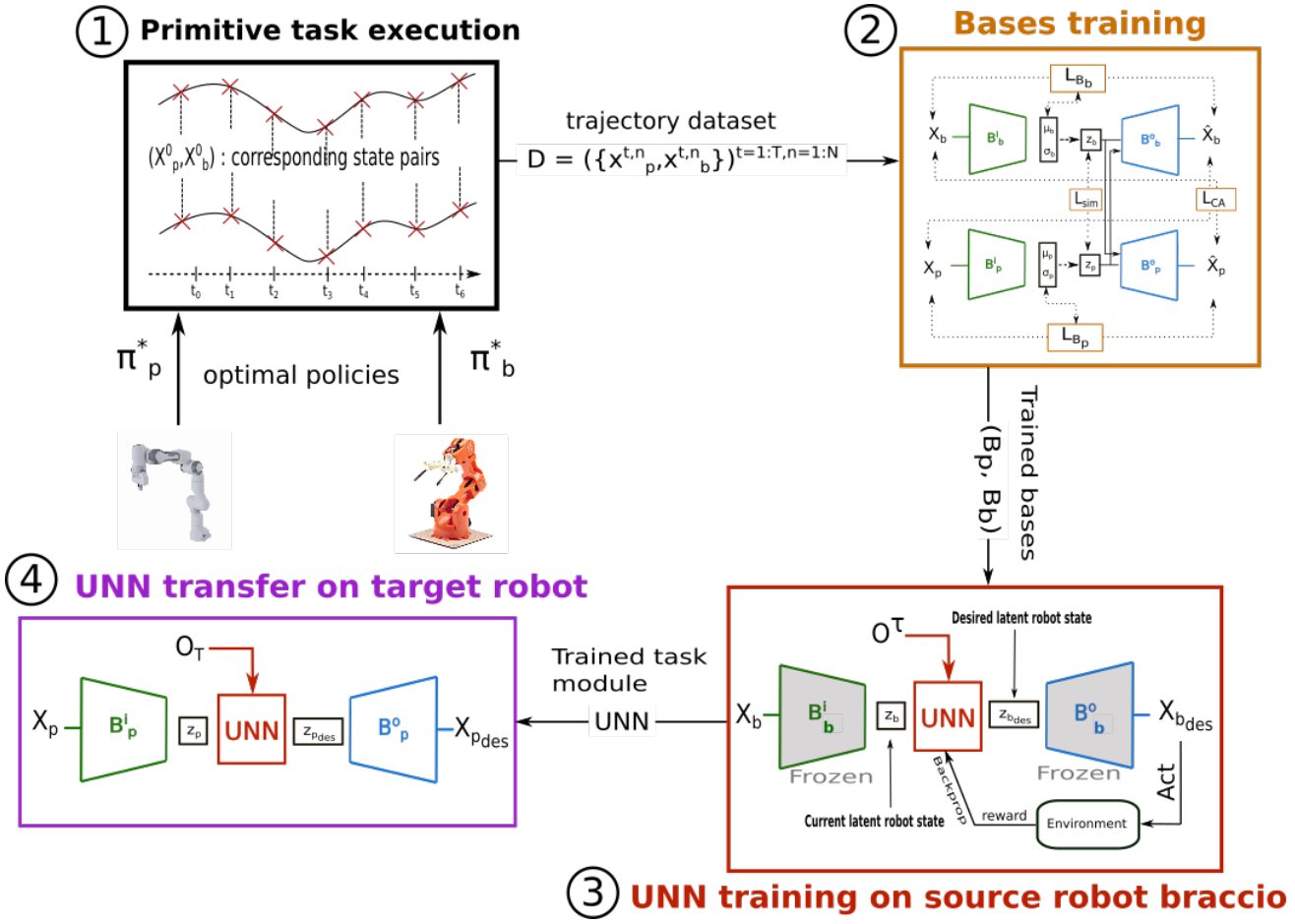


Figure 3: Latent space UNN learning process, from left to right using the Braccio and Panda robot as illustrative example. The first step is the collection of time-aligned pairs of corresponding states along primitive task execution. The next step is the bases training to obtain B_p and B_b. Using the trained bases, it is possible to train a UNN module inside the shared latent space. Finally, the trained task module is transferred to another robot.

where we only have two robots at our disposal. Then, in Section 5.3, we describe the procedure to "plug" any extra robot to the common latent space already created so that it can act as a source or a target robot for UNN transfers. Please note that any of the considered robots can be either source or target once the latent space is aligned. Finally, we detail the UNN training process with the PPO algorithm [4], but any RL algorithm could fit with our method. We specifically chose PPO because it provides a stable and monotonous policy improvement along with SOTA asymptotic performance. The full latent space UNN learning process is presented in Figure 3. Additional details can be found in Appendix A.

5.1. State pairing

We construct Z such that a pair of similar robot states from two different robots r_1 and r_2 $(x_{r_1}, x_{r_2}) \in X_{r_1} \times X_{r_2}$ maps to the same point in the latent space. As such, we first need to get corresponding pairs of states across robots morphologies (as illustrated in Figure 3 step1). In this work, we considered *primitive* tasks performed by both robots at the same speed with optimal policies. The optimal policies can be obtained through standard analytic methods or can be learned. We then assume that two states visited at the same timestep during the execution of the optimal policies for the primitive task can be matched. In other words, if we denote by $\pi^{*,\mathcal{T}}_{r_1}$ and $\pi^{*,\mathcal{T}}_{r_2}$ the optimal policy for primitive task $\mathcal{T}$ on respectively, robot r_1 and robot r_2, then

$$x^{t_1}_{r_1} \approx x^{t_2}_{r_2} \iff t_1 = t_2 \tag{11}$$

with t_1 and t_2 two timesteps occurring during the same task execution and "$\approx$" denotes similarity between two states. As such, we make use of the simple but effective time-alignment process described above, already used in prior works [21, 27, 28] to find correspondences between states.

In practice, we can consider several distinct primitive tasks, each with its optimal policies, to generate a large variety of trajectories along which states are paired. This would promote generalization for the bases, even if they have not been trained the entire robot state-space. As a consequence, these primitive tasks should be carefully crafted in order to be representative of how the robots can move, but also fairly simple to obtain opti-

mal policies easily. One can also include prior knowledge about desirable robot configurations in the form of a constraint (e.g effector orientation). For maximum efficiency, the bases training should focus on the state-space regions that will be visited during the UNN tasks training. Once a dataset of corresponding states defined as :

$$D = (x_{r_1}^{t,n}, x_{r_2}^{t,n})^{t=1:T, n=1:N} \tag{12}$$

where T is the length of a trajectory and N the number of trajectories, is available, the bases can be trained in an unsupervised setting to build the shared latent space.

5.2. Bases Training

The base training is the second step of Figure 3. This section explains how to create and align the robots latent space Z given D, the dataset of time-aligned corresponding states described at the previous step. As mentioned earlier, the two mappings $B^i : X_r \rightarrow Z$ and $B^o : Z \rightarrow X_r$ are respectively called the input base and the output base (depicted as the green and blue modules in Figure 1). They are represented using a variational auto-encoder (VAE) structure as described in [29] to benefit from its dimension reduction and generative capabilities. The VAE model is a very powerful approach to learn a low-dimensional latent representation $z \in Z$ of some data $x \in D$, where D is a dataset containing data of interest such as high-dimensional images or, in our case, vectors of robots state. It is made up of two complementary models, each with a specific role. The probabilistic encoder $B^i_{\phi_r}(z|x_r)$ parameterized by ϕ_r is tasked to relate x_r to its corresponding z on the data manifold. In other words, it maps data from the input space to a compressed, but meaningful, representation in the latent space. Most of the time, we assume a simple multivariate standard normal distribution $p(z)$ over latent vectors and $B^i_{\phi_r}(z|x_r) = \mathcal{N}(z|\mu_{\phi_r}(x_r), \sigma^2_{\phi_r}(x_r))$ where $\mu_{\phi_r}(x_r)$ and $\sigma^2_{\phi_r}(x_r)$ are parameterized by a neural network with weights ϕ_r. The generative part is handled by the probabilistic decoder $B^o_\theta(x_r|z)$ which remaps a given $z \in Z$ to its corresponding $x \in X_r$ by means of a neural network with weights θ_r. Both networks are trained by stochastic gradient descent mainly to maximize the likelihood of the training data :

$$L_{B_r} = -\mathbb{E}_{z\sim B^i_{\phi_r}(\cdot|x_r)}[\log B^o_{\theta_r}(x_r|z)] + \beta\, D_{KL}(B^i_{\phi_r}(z|x_r)\|\mathcal{N}(0, I)) \tag{13}$$

where β is the hyper-parameter introduced in [30] to adjust the trade-off between reconstruction and the KL-divergence. The KL-term in the loss function acts as a regularizer to ensure that the latent space is continuous (two close points in the latent space give two similar outputs when decoded) and complete (a point sampled from the latent space should produce an output that makes sense).

We wish to find a shared latent space between the robots by aligning their respective latent spaces to enable cross-robot transfer. In other words, if $x_{r_1} \approx x_{r_2}$, $z_{r_1} \sim B^i_{\phi_{r_1}}(\cdot|x_{r_1})$ and $z_{r_2} \sim B^i_{\phi_{r_2}}(\cdot|x_{r_2})$, then ideally we want $z_{r_1} = z_{r_2}$. We enforce this condition through the use of a similarity loss defined as:

$$L_{sim} = \mathbb{E}_{z_{r_1}\sim B^i_{\phi_{r_1}}(\cdot|x_{r_1}),\, z_{r_2}\sim B^i_{\phi_{r_2}}(\cdot|x_{r_2})}[\|z_{r_1} - z_{r_2}\|^2] \tag{14}$$

which encourages their encoding distances to be small. Inspired by [31], we also use a cross-alignment loss to improve cross-domains transfers. Each decoder reconstructs its input by using the latent encoding of the paired similar state sampled from the other robot's encoder:

$$\begin{aligned} L_{CA} = {} & \mathbb{E}_{z_{r_1}\sim B^i_{\phi_{r_1}}(\cdot|x_{r_1})}\left[\left\|B^o_{\theta_{r_2}}(z_{r_1}) - x_{r_2}\right\|^2\right] \\ & + \mathbb{E}_{z_{r_2}\sim B^i_{\phi_{r_2}}(\cdot|x_{r_2})}\left[\left\|B^o_{\theta_{r_1}}(z_{r_2}) - x_{r_1}\right\|^2\right] \end{aligned} \tag{15}$$

Gathering equations (13),(14) and (15), we train the bases of both robots at the same time with the following full objective:

$$\min_{\theta_{r_1},\theta_{r_2},\phi_{r_2},\phi_{r_1}} \sum_{(x_{r_1},x_{r_2})\in D} L_{B_{r_1}} + L_{B_{r_2}} + \delta L_{sim} + \lambda L_{CA} \tag{16}$$

where δ and λ are constants, respectively weighting the contributions of L_{sim} and L_{CA} to the full training loss. We chose $\delta = 2/3$, $\lambda = 1/3$ and $\beta = 0.00015$. See Appendix B for an in-depth discussion on how to adjust these hyper-parameters. . We trained for 100 epochs with a learning rate of 5.10^{-4} and a batch size of 100.

5.3. Adding a new robot to the set

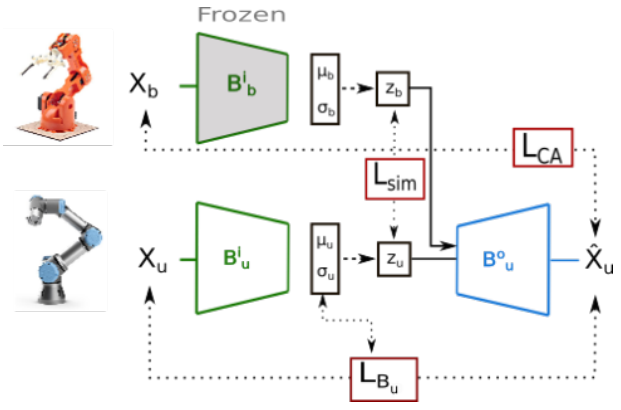


Figure 4: Bases fitting process. This figure describes the methodology to align the latent space of a new robot using the Braccio (reference) and UR10 (new) robots as illustration. The similarity and cross- alignment loss are minimized with respect to a ground truth provided by the frozen input base of a robot already sharing the latent space.

It is possible to transfer a UNN module to and/or from an arbitrary number of robots that were not initially considered during the shared latent space building. It is simply required to align the new robot's latent space to the already existing common latent space. The alignment process is depicted in Figure 4. The steps are very similar to the previously described workflow:

- Enlarge the dataset D of paired similar states with collected states on the new robot by following section 5.1.
- Use the input base $B^i_{\phi_{ref}}$ of one of the robots already contributing to Z as a reference to align the latent space of the newly added robot. The weights ϕ_{ref} are frozen and only ϕ_{new} and θ_{new}, respectively the input and output base weights of the added robot, are adjusted. More formally, the following loss is minimized

$$\min_{\theta_{new},\phi_{new}} \sum_{(x_{ref},x_{new})\in D} L_{B_{new}} + \delta L_{sim} + \lambda L_{CA} \tag{17}$$

Once the new robot's latent space is aligned, it can act as a source or as a target robot for any UNN module training or transfer. This process can be repeated to add as many extra robots as needed.

5.4. UNN training

Once the bases training is done, we can proceed to the training of a UNN module U_ψ, approximated by a neural network with weights ψ on a chosen task. The UNN training is illustrated in the third step of Figure 3. We parametrize the UNN policy as a Gaussian distribution i.e

$$U_\psi = \mathcal{N}(\mu_\psi(z, o_\tau), \Sigma_\psi) \tag{18}$$

where $\mu_\psi(z, o_\tau)$ is the neural network that maps from observations to mean actions and Σ_ψ is the covariance diagonal matrix whose parameters are independent of the state. For an intuitive and visual understanding of how the UNN module fits into the complete policy architecture, please refer to Figure 1. During task training, the bases weights θ_r and ϕ_r are frozen and the gradient is backpropagated through the UNN module only to adjust ψ (see Figure 3). As input to the UNN module, we sample $z \sim \mathcal{N}(z|\mu_{\phi_r}(x_r), \sigma^2_{\phi_r}(x_r))$ from $B^i_{\phi_r}(\cdot|x_r)$, rather than taking the mean, to keep stochasticity in the observed latent state as a form of domain randomization. We empirically found that it improves UNN robustness against domain shift that may appear when transferring the UNN modules from one robot to another, which in turn improves zero-shot performance after transfer. For the sake of clarity, we write the PPO objective [4] optimized by the UNN module, with the relevant notations and by using Equations (6) and (7). Denoting $s_u = (z, o_\tau)$, the concatenation of the current latent state z and the task-specific observation o_τ:

$$\begin{aligned} L(s_u, z_d, \psi_{old}, \psi) = \min \Big(& r(\psi) A^{U_{\psi_{old}}}(s_u, z_d), \\ & clip(r(\psi), 1-\epsilon, 1+\epsilon) A^{U_{\psi_{old}}}(s_u, z_d) \Big) \end{aligned} \tag{19}$$

with

$$r(\psi) = \frac{U_\psi(z_d|s_u)}{U_{\psi_{old}}(z_d|s_u)} \tag{20}$$

and A^{U_ψ} is the advantage function computed using the Generalized Advantage Estimation (GAE) algorithm [32]. Since the UNN operates on the shared latent space, it is by definition robot-agnostic and thus can be transferred to any robot of the set by plugging it between the corresponding pair of bases.

6. Experiments

To validate our TL approach, we conducted experiments in simulation and also on physical robots. The control loop executes at 10 Hz, which means the agent was observing the environment state and acting every 0.1 second. Simulation is run on the Unity physic simulator. For the real world experiments, we use ROS to operate the robots and the regular joint velocity controllers from the ros-control package. All robots are velocity controlled. A summary of the robots used and tasks learned is depicted in Figure 2.

6.1. Considered robots

To highlight the benefits of our cross-robot transfer method, we consider robots with distinct morphology traits (number of joints and the links length) but similar working capabilities such that they can perform the same tasks. More specifically, our pool of robots is composed of a Panda robot with 7 DoF, a UR10 with 6DoF and a Braccio robot with 5 DoF. On the simulation side, we normalize the total length such that all considered robots have roughly the same reaching capabilities for practical reasons. For the experiments done on the physical robots, we include the size difference into the task and scale the experimental setup according to the reach of each robot. This means that every position in the work space is defined relatively to the robot size . As shown in Figure 5, the 3 kinds of robots have different numbers of degrees of freedom and morphologies.

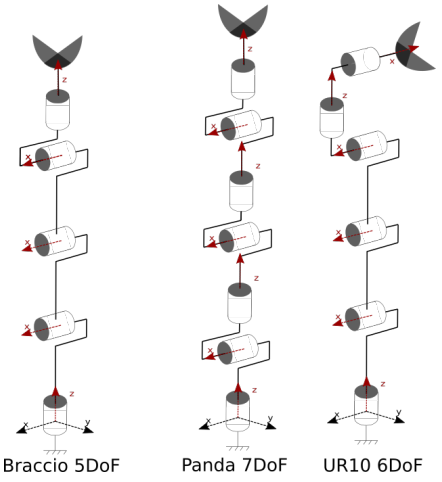


Figure 5: Kinematic diagrams of the considered robots

6.2. Considered tasks

We evaluate the efficiency of our proposed transfer method on three 3D tasks: pick and place, ball catcher and peg insertion. In all tasks, the robot-specific observations that the input bases encode into the latent spaces are the joints position $q \in \mathbb{R}^n$ and joints velocity $\dot{q} \in \mathbb{R}^n$, where n is the number of joints, so we have $X \in \mathbb{R}^{2n}$. As such, the decoded (i.e provided by the UNN and translated by the output base) robot state applied to the robot also contains the joints position. However, in all our

experiments the robots are velocity controlled, so we ignore the joints position given by the output base when working with the UNN module (see Appendix C for a discussion on the motivations).

- **Pick and Place**: The goal of the agent is to pick a cube from a table and place it at a desired location. While conceptually simple, it requires the agent to solve a sequential problem: first picking, then placing. The picking part is controlled by a simple boolean set by the agent. If this boolean is true and the suction part of the gripper is colliding with the cube, it sticks. The position of the cube and the desired location are chosen randomly at the beginning of each episode from a range of possible coordinates. The task-related observations for this task are $O^{\mathcal{T}} = \mathbb{R}^{10}$ with current cube position $\in \mathbb{R}^3$, target cube location $\in \mathbb{R}^3$, effector position $\in \mathbb{R}^3$ and a flag *hold* to indicate to the agent if a cube is held or not. The actions applied to the robot are desired joints velocity $\in \mathbb{R}^n$ and a boolean indicating whether the effector should stick or not. The reward function used for training is the following:

$$r_t = \begin{cases} \frac{a}{d_{z,c}} & \text{if } hold = True \\ -b.d_{e,c} & \text{else} \end{cases} \tag{21}$$

  where $d_{z,c}$ and $d_{e,c}$ are respectively the distance cube-drop zone and the distance effector-cube. The constant a and b are small scaling constants. In our settings $a = 1.2$ and $b = 0.3$.

- **Peg insertion**: This task is particularly challenging in the context of cross-robot transfer because it requires a precise control policy. The effector of the robot is replaced by a stick and the agent task is to insert it into a hole of appropriate dimensions. The position of the hole is randomly sampled at the beginning of each episode from a box region. It will highlight the UNN ability to accurately operate the robots it is transferred on, even when their kinematic structures vary significantly. The task-related observations for this task are $O^{\mathcal{T}} = \mathbb{R}^6$ with the tip of the peg position $\in \mathbb{R}^3$ and the hole position $\in \mathbb{R}^3$. Like previous task, the actions applied to the robot are desired joints velocity $\in \mathbb{R}^n$. The reward function used for training simply incentivizes the agent to get the peg closer to the bottom of the hole:

$$r_t = \begin{cases} -s.d_{p,h} & \text{if } d_{p,h} \geq \varepsilon \\ f & \text{else} \end{cases} \tag{22}$$

  where $d_{p,h}$ is the distance between the peg tip and the hole bottom. The constant $s = 0.15$ is a small scaling constant. The threshold ε was set to 0.001 and triggers a small positive reward f.

- **Ball Catcher**: This task is used to demonstrate that the UNN is able to cope with dynamic tasks. The effector of the robot is replaced by a basket and the goal of the agent is to catch a ball thrown at it before it touches the ground. The ball is thrown from the same location, but the trajectory is varied by randomly selecting the ball velocity for each of the 3 spatial axes. The task-related observations for this task are $O^{\mathcal{T}} = \mathbb{R}^{13}$ with ball position $\in \mathbb{R}^3$, ball velocity $\in \mathbb{R}^3$, basket position $\in \mathbb{R}^3$ and basket velocity $\in \mathbb{R}^3$ and a flag *catch* to indicate to the agent if the ball is inside the basket. Once again the actions applied to the robot are the desired joints velocity $\in \mathbb{R}^n$. The reward function used for training is the following:

$$r_t = \begin{cases} e - |\theta_e| & \text{if } catch = True \\ -c.d_{b,t} - |\theta_e| & \text{else} \end{cases} \tag{23}$$

  where $d_{b,t}$ is the distance between the basket and the target receiving position. The constant c is a small scaling constant and e is a small positive constant reward. In our settings, $c = 0.15$ and $e = 0.15$. The term $|\theta_e|$ is the angle between the effector pose and the vertical plane which guides the agents towards suitable body configurations. Its contribution to the overall reward is weighted by the constant β. Interestingly, this reward term is required by the regular RL agents to learn a successful policy but not for our UNN agents, already imbued with prior knowledge about adequate effector orientation through the bases as discussed in Appendix A.

For the real world experiments, we transfer the UNNs and the RL agents corresponding to the peg insertion task trained in simulation, to the physical Panda and UR10 only. The physical Braccio robot was not reliable enough to attempt a peg insertion task due to significant mechanical play and offsets in the joints. The position of the peg hole is measured in real time using a motion capture system from qualisys.

7. Results

In this section, we present our results both on training and transferring on the chosen manipulation tasks. We focus on zero-shot transfer as it is the most sample efficient. We therefore compare zero-shot performance of the transferred UNN to the asymptotic performance of regular RL baselines trained from scratch with PPO to emphasize the training-time saved. However, in some instances, source performances cannot be recovered directly after transferring the UNN agent. To correct this, the UNN module can be fine-tuned on the target robot to recover near maximum performance. We show that fine-tuning can be achieved with a fraction of the training time required by the pure RL baselines. Finally, we also demonstrate that the prior knowledge included in the bases can speed up the learning of the tasks on the source robot (i.e the UNN training), when compared to standard PPO training.

7.1. Performance metrics and baseline

Inspired by early work on TL for RL agents [33], we analyze three metrics when comparing both methods:

- **Zero-shot performance**: Initial performance of the UNN agent right after transfer. It will measure how useful the knowledge transfer is to jumpstart learning on the target robot.
- **Asymptotic performance**: The final performance obtained after training from the Zero-shot performance of the UNN agent (fine-tuning) or from scratch. This will show whether or not our method falls into local minima or enables similar asymptotic performance as baselines.
- **Sample efficiency**: Number of samples needed for the agent to reach its final performance. This is, arguably, the most appropriate metric for evaluating a transfer learning method, especially in RL which is notoriously sample inefficient.

The PPO baselines serve as an upper bound for the performance and corresponding amount of training steps needed when using state-of-the-art RL algorithms. Our method also leverages PPO to fit the UNN module, but inside a pre-trained and shared latent space. We train one UNN module per robot for each task (i.e 9 UNN modules in total) before transferring it to the other robots of the pool. As explained earlier, when varying the task, the robot bases are kept the same. We then measure performance as the task success rate, i.e the percentage of successful episodes over 1000 trials. For the pick and place task, an episode is considered successful if the distance between the cube and the place location is smaller than a threshold set to 0.15 Unity units (approximately 3 cm for an environment scaled for the UR10 robot). For the ball catching task, we define success as simply catching the ball, i.e the ball does not bounce outwards and is kept inside the basket. In the case of the peg insertion task, we also define task success with a distance threshold between the peg tip and its desired position inside the hole set to 0.05 Unity units or 1 cm. For the execution on the real robot, the success threshold is set according to the scale of the robot. Regarding the experiments on the physical robots, we test 28 different hole positions uniformly spread out on the working area. Its size is scaled proportionally to the robot total length. The square hole was 2.25 cm x 2.25 cm and the peg diameter was 1.9 cm. The bases used on the physical robots are the same as the ones used on their virtual counterparts. Videos demonstrating some of our experiments are available here. Code can be found here.

7.2. Zero-shot transfers

Here we discuss the performance obtained on the target robot right after transfer from a different source robot, without fine tuning the UNN module. In particular we compare immediate performance on the target robot after transfer to the performance of the corresponding RL baseline which serves as an upper bound. In other words, we answer the question "What kind of performance can be expected with a simple plug-and-play of the UNN module on a target robot ?". As we are ultimately interested in reducing the training time on the target robot, it could be interesting to see if a UNN transfer recovers performance close to the source robot or even RL baseline only with zero-shot transfer.

7.2.1. Pick and place

On the pick and place task which is arguably the simplest, nearly all transfers are zero-shot. Except for Braccio → Panda transfer, maximum performance is recovered after direct transfer without the need for further training on the target robot as shown in Table 1. It means that we get results competitive with those of the corresponding RL baselines, without any fine tuning or training on the target robot, saving us an average of 1.8 millions training steps required to reach convergence.

Source\Target	**Braccio**	**Panda**	**UR10**
Braccio	100	93 / **100**	100
Panda	100	100	100
UR10	100	100	100

(a) Performance obtained for UNN agents (zero-shot/**fine-tuned**).

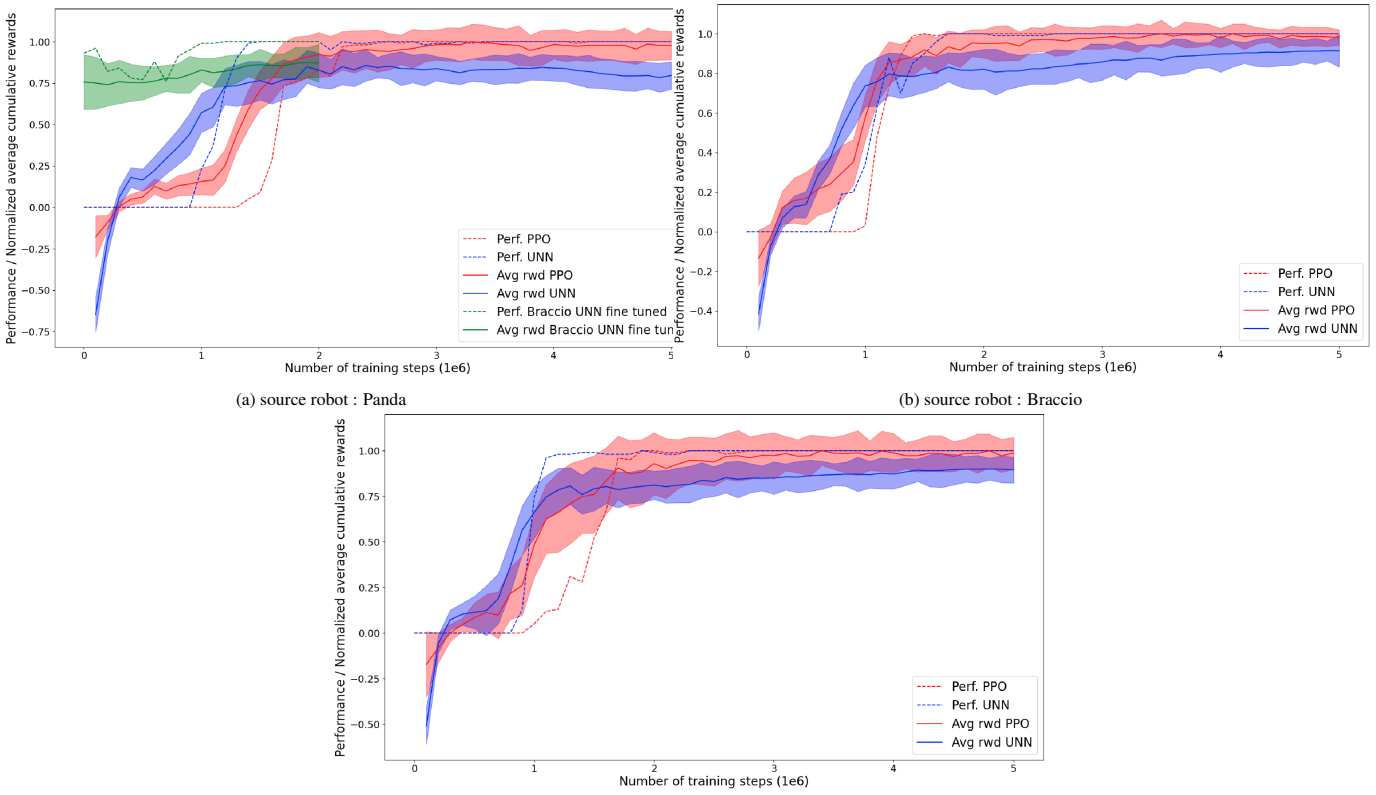
Braccio	**Panda**	**UR10**
100	100	100

(b) Performance obtained for RL agents.

Table 1: Pick and Place task : performances on the simulated robots. Results are reported as percentage of task success over 1000 trials.

7.2.2. Ball catcher

Regarding the zero-shot performance on the ball catcher task shown in Table 2, we can see that half of the transfers exhibits zero-shot performance close to the RL baselines performance : Braccio → UR10 (98%), Braccio → Panda (98%) and UR10 → Panda (98%). However, some of the zero-shot transfers experience a performance drop: Panda → Braccio (86%), Panda → UR10 (90%) and UR10 → Braccio (89%). This can be explained by the fact this task requires much more precise movements to catch the ball. If, for instance, the basket is tilted too much on the left or is leaning a little too far forward, the ball will bounce back outward, hit the basket's edge or miss it, resulting in task failure. The reconstruction error occurring during the bases training inevitably induces these small parasitic movements. As a consequence, this task is less tolerant towards imprecision and more challenging when it comes to transfer. Moreover, results seem to suggest that UNN policies learned on a higher DoF robot tend to not transfer well to robots with less DoF. All transfers from the Panda robot (7 DoF) need further fine tuning on the target robot. The same phenomenon can be observed from UR10 (6 DoF) to Braccio (5 DoF). We hypothesize that the UNN will somehow take advantage of the extra joints and learn policies that leverage this additional flexibility , yielding a policy that may not be adapted to less "expressive" robots. When transferring to a robot with fewer DoFs, such as the Braccio, the policy may be used to the additional expressiveness and this mismatch would lead to a performance drop. Though, performance is still high enough to demonstrate that transfer is beneficial and as will be presented in Section 7.3, maximum performance can be recovered with a fraction of the required training for RL baselines.



(a) source robot : Panda

(b) source robot : Braccio

(c) source robot : UR10

Figure 6: Pick and place task. Evolution of the average reward (solid line) and performance (dotted line) during training for the three robots considered. The y-axis is the normalized average reward / average performance and x-axis is the number of training steps. The PPO agent refers to our RL baseline trained from scratch. The UNN agents are trained using our framework. Fine-tuned agents metrics are shown on the plots which display the training curve of the source agent.

Source\Target	**Braccio**	**Panda**	**UR10**
Braccio	98	98	98
Panda	86 / **98**	98	90 / **98**
UR10	89 / **98**	98	99

(a) Performance obtained for UNN agents (zero-shot/**fine-tuned**).

Braccio	**Panda**	**UR10**
100	99	99

(b) Performance obtained for RL agents.

Table 2: Ball Catcher task : performances on the simulated robots. Results are reported as percentage of task success over 1000 trials.

7.2.3. Peg insertion

This task is also very challenging for transfer between robot's morphologies as it requires accurate movements and careful control of the robot to successfully insert the peg. Nevertheless, success rates right after transfer of the UNNs are significantly high, indicating a beneficial and efficient transfer of knowledge between the considered agents as shown in Table 3. But once again, fine tuning is required to recover from performance loss after transfer. We also show results from transferring the UNN policies on the physical UR10 and Panda robots. The UNN were trained in simulation and the bases used are the same as their virtual counterparts. The fine-tuned UNNs are also the ones obtained in simulation, they are not fine-tuned on the physical robots. Results are reported in Table 4. All the considered agents experience a performance drop after the sim2real transfer. In particular UR10 → Panda decreases from 90% to 71% success rate in zero-shot. Likewise, Braccio → UR10 suffers from the sim2real transfer, but its fine-tuned version performs well. Despite the large reality gap, performance on the physical robots is still above 80% after fine-tuning on the virtual robots.

Source\Target	**Braccio**	**Panda**	**UR10**
Braccio	100	95 / **100**	92 / **100**
Panda	95 / **100**	100	82 / **100**
UR10	99 / **100**	90 / **100**	100

(a) Performance obtained for UNN agents (zero-shot/**fine-tuned**).

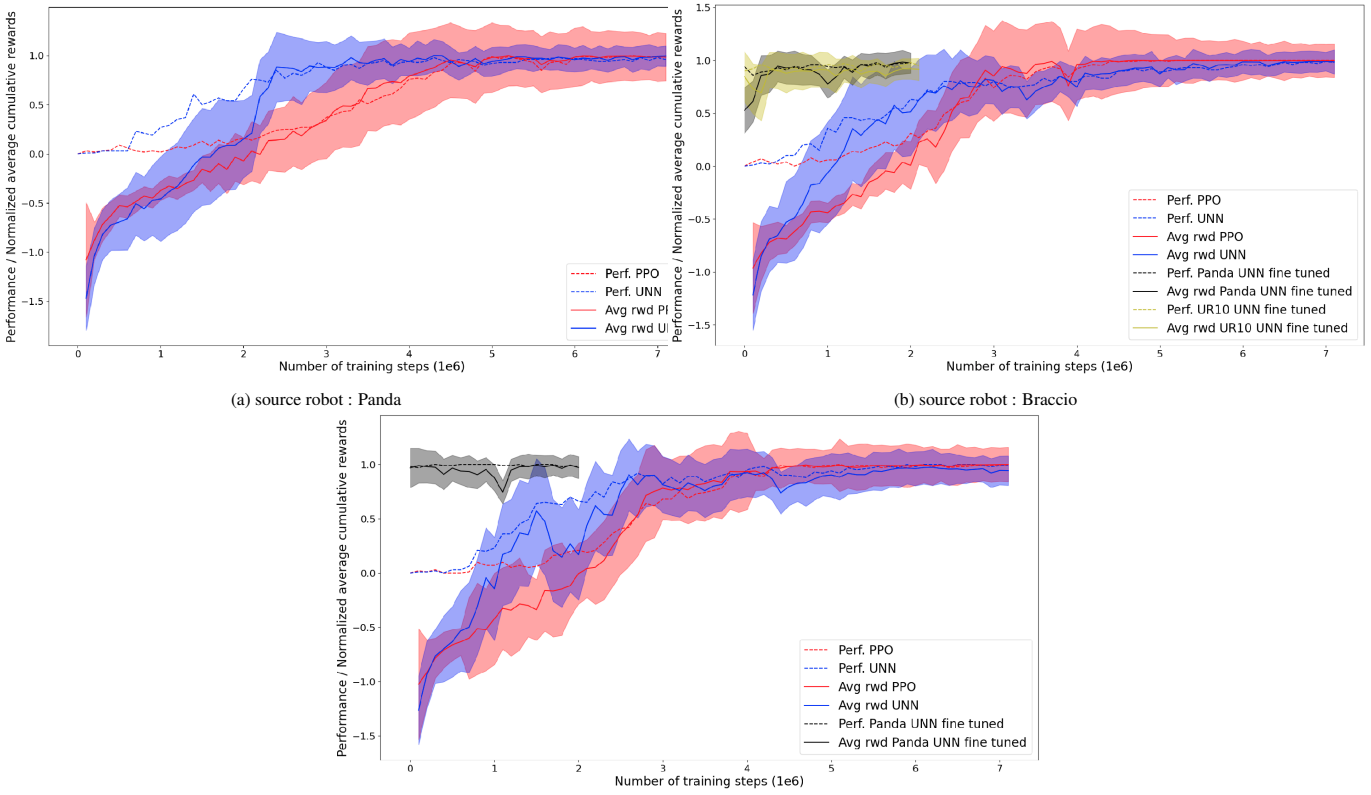
Braccio	**Panda**	**UR10**
100	100	100

(b) Performance obtained for RL agents.

Table 3: Peg Insertion task : performances on the simulated robots. Results are reported as percentage of task success over 1000 trials.

7.3. LS-UNN fine tuning

In this section, we analyze the performance of the transferred UNN after fitting it on the target robot. When fine-tuning



(a) source robot : Panda

(b) source robot : Braccio

(c) source robot : UR10

Figure 7: Ball catcher task. Evolution of the average reward (solid line) and performance (dotted line) during training for the three robots considered. The y-axis is the normalized average reward / average performance and x-axis is the number of training steps. The PPO agent refers to our RL baseline trained from scratch. The UNN agents are trained using our framework. Fine-tuned agents metrics are shown on the plot which display the training curve of the source agent.

Source\Target	**Panda (physical)**	**UR10 (physical)**
Braccio (virtual)	79 / **93**	79 / **93**
Panda (virtual)	93	79 / **96**
UR10 (virtual)	71 / **85**	96

(a) Performance obtained for UNN agents (zero-shot/**fine-tuned**).

Panda	**UR10**
89	100

(b) Performance obtained for RL agents.

Table 4: Peg Insertion task : performances on the physical robots. Results are reported as percentage of task success over 28 trials.

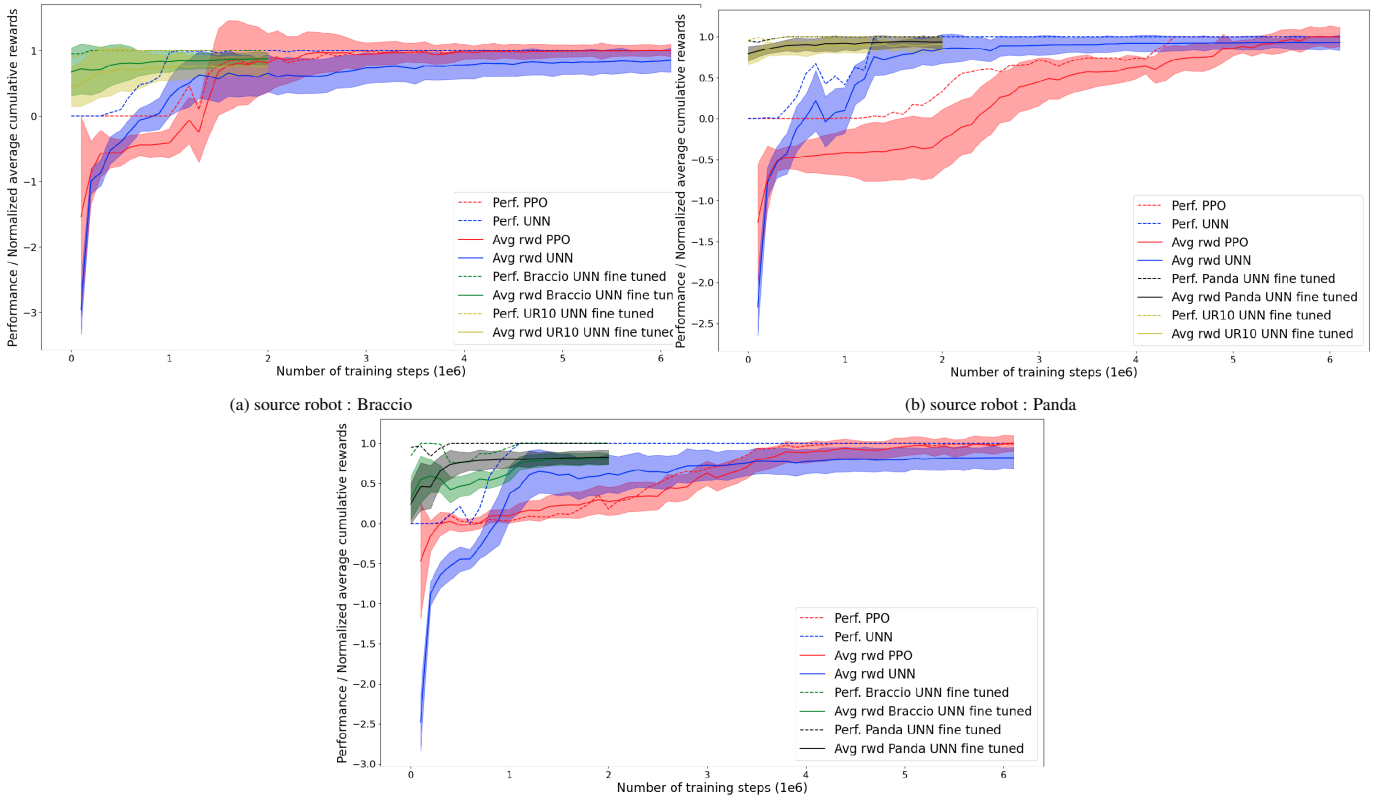
the UNN agent on a target robot, we remove stochasticity in the observed latent state, i.e $z = \mu_{\phi_r}(x_r)$. We are especially interested in the time gained by simply fine tuning the UNN module, as opposed to training a RL agent from scratch. As such, we will discuss the asymptotic performance and the sample efficiency metrics described in section 7.1. As shown in Figures 6, 7 and 8, most of the fine tuned UNN agents start from an already high jumpstart performance and converge faster to the highest performance than their RL counterparts. It is worth noting that PPO is a trust-region based RL algorithm which theoretically provides monotonous improvement by solving a KL-constraint. As such it is well suited for fine tuning because it mitigates catastrophic forgetting and improves training stability.

7.3.1. Pick and Place

For the pick and place task (see Figure 6), most of the transfers were already close to baseline, so fine tuning was performed only for the Braccio → Panda transfer. While the UNN experiences a performance drop, it recovers competitive results significantly faster than what was needed for its RL counterpart to reach the same performance. More precisely, it achieves roughly a 2x speed up over a regular PPO training.

7.3.2. Ball Catcher

For the more challenging ball catcher task, fine tuning is necessary on half of the transfers. However, all considered transfers achieve competitive performance provided some quick fine tuning. The Panda → Braccio and UR10 → Braccio transfers, which had the lowest zero-shot performance (86% and 89%), reach near baseline performance in respectively 1.6 millions and 2 millions steps of fine-tuning against the 4.5 millions required by the regular RL agent, yielding a 2.8x and 2.25x speed up. Regarding the UR10 → Braccio transfer starting at a 90% performance, the fine-tuning required only 14% of the amount of training samples needed by its corresponding RL baseline to achieve similar performance.



(a) source robot : Braccio

(b) source robot : Panda

(c) source robot : UR10

Figure 8: Peg insertion task. Evolution of the average reward (solid line) and performance (dotted line) during training for the three robots considered. The y-axis is the normalized average reward / average performance and x-axis is the number of training steps. The PPO agent refers to our RL baseline trained from scratch. The UNN agents are trained using our framework. Fine-tuned agents metrics are shown on the plot which display the training curve of the source agent.

7.3.3. Peg Insertion

None of the transfers retrieves full performance after zero-shot transfer. Nevertheless, success rates are already close to baseline. Braccio → UR10 has the lowest zero-shot success rate amongst all transfers, successfully achieving the task 82% of the time. Figure 8c shows that the Braccio UNN can quickly adapt to the UR10, reaching maximum performance with as little as 2% (100000 steps against 3.8 millions) of the experience required by the corresponding baseline. Other fine-tuned transfers depicted in Figure 8 exhibit similar recovery speed and improved sample efficiency.

7.4. Ablation study

We perform an ablation study on the UNN task space to better understand the contribution of the latent features to the transfer success. Three variants of the proposed framework are benchmarked:

- LS-UNN: the task module is trained in the latent space and receives task observations containing end-effector related information (the gripper, the peg, the basket).
- LS-UNN-no-FK: We remove end-effector related information, but keep the latent features. This baseline is trained with the same setup as LS-UNN (same bases, hyper-parameters etc...)
- UNN: The UNN modules operates in the end-effector cartesian space without latent features. This is equivalent to the UNN formulation in [11]. The UNN baseline is trained with the Base Abstracted Modeling robot where the task is learned on a free-flying end-effector to remove any bias that might emerge from a particular robot morphology

For simplicity of exposition, we average transfer performance over robots and display mean results for each environment.

	PnP	Ball Catcher	Peg Insertion
UNN	96	82	84
LS-UNN-no-FK	77	70	51
LS-UNN	**98**	**93**	**92**

Table 5: Zero-shot performance for each UNN variant after task-module transfer on Braccio, UR10 and Panda. Results represents task sucess rate over 100 trials and are averaged over robots for each environments: PnP (Pick and Place), Ball Catcher and Peg Insertion.

As shown in Table 5, a UNN trained with end-effector state/action space is a strong baseline and performs well across all environments which confirms the suitability of the cartesian space to transfer policies. An ablated LS-UNN however, underperforms

all other variants by a significative margin which indicates that relying on latent features only leads to more over-fitting and hinder zero-shot generalization to new morphologies. We hypothesize that this is due to a slight misalignment between latent spaces which causes compounding errors when executing the task on a target robot. However, when used in conjunction with end-effector related information, the performance significantly increases as demonstrated by the superior performance of LS-UNN. We believe that the task-module can use the additional information contained in the latent features to perform the task, and the end-effector information to self-correct and mitigate over-fitting/misalignment between source and target robot, leading to effective transfer.

8. Discussions and Limitations

Previous results have showcased the benefits of transferring UNN modules between agents. Our proposed approach significantly reduces the cost of training RL agents by providing efficient fine-tuning or even zero-shot generalization. Nevertheless, as is often the case, there is no free lunch. In its current form, leveraging the LS-UNN efficient transfers comes with some trade-offs that we discuss in this section.

8.1. State-pairing requirement

Our framework heavily relies on state-pairing to align the latent representations of the robots and enable beneficial transfer. Multiple solutions are available to find similarities between states depending on the setup and constraints. In this work, we used a simple time-alignment scheme to focus on how to perform efficient transfer of skills given an aligned latent space, rather than how to find correspondence between states. However, the assumption that the robots perform the same primitive task at the same rate may be too stringent or inadequate in some settings. For instance, it may not be robust enough when both robots are performing the primitive task at somewhat different speeds, or if the sampling loop does not run at a perfectly fixed rate. To address this shortcoming, more practical and sophisticated pairing methods can be explored such as dynamic time-warping [34, 35] or cycle-consistency [36, 37] to deal with temporally non-aligned states. A straight-forward extension to trajectory-pairing to alleviate the burden of time-alignment, is briefly studied in Appendix D. Another promising line of research [38, 39] focuses on encoding and aligning entire skill trajectories using a Conditional Neural Movement Primitive (CNMP) network [40] to achieve CATL. A valuable direction for future exploration is evaluating whether the shared feature space generated by this approach can serve as an effective learning space for the task module.

8.2. LS-UNN agents training

In this section, we analyze and compare the convergence speed and asymptotic performance reached by the UNN agents during training against vanilla RL baselines. We trained the agents for 6.10^6 steps on the pick and place task, 7.10^6 steps on the ball catching task and 6.10^6 steps on the peg insertion task and monitored their improvement through the average cumulative reward obtained per episode.

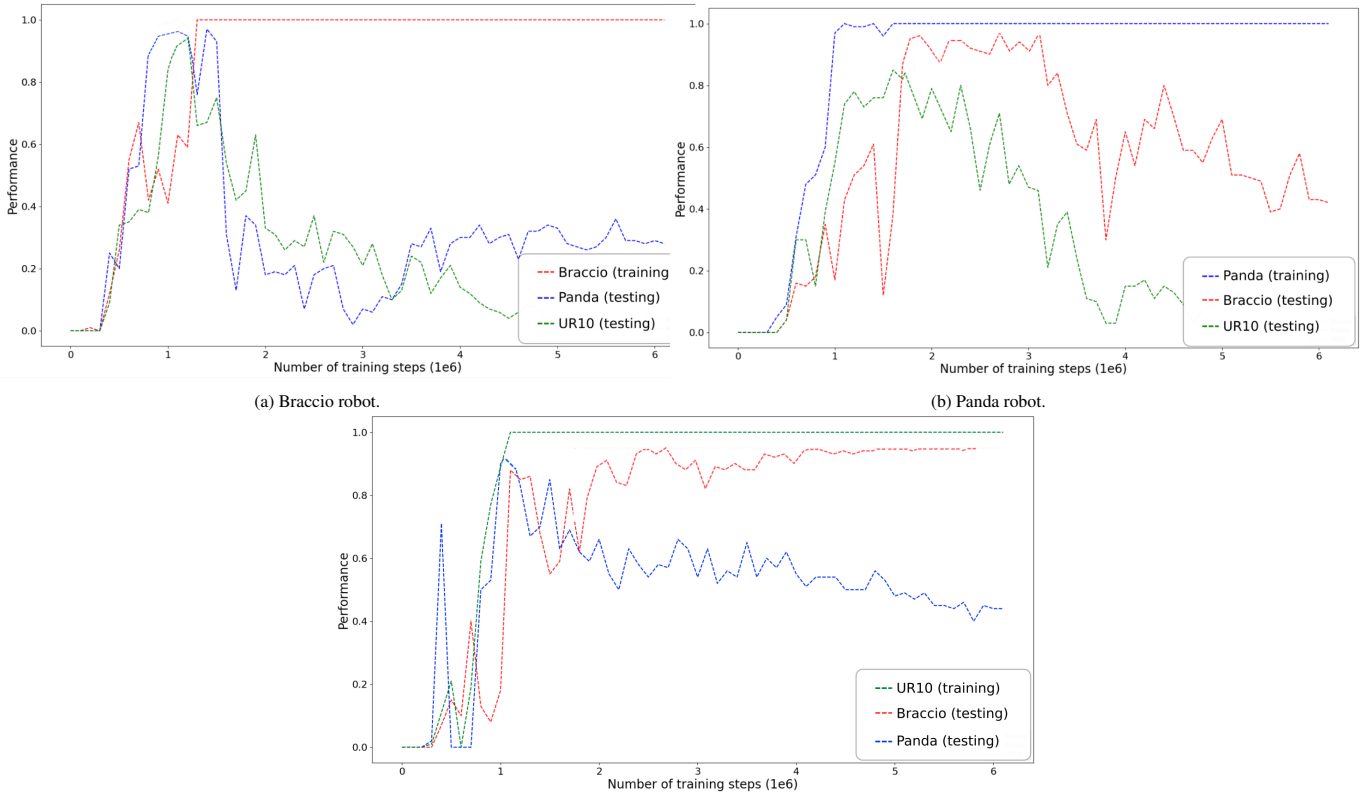
As shown in Figures 6, 7 and 8, except for the Braccio robot on the pick and place and ball catcher tasks, the UNN agents trained from scratch slightly outperform the Vanilla RL agents in terms of convergence speed. We believe that this is due to the constraint imposed on the end effector's orientation during state collection (see Appendix A). Consequently, the latent space embeds states are well suited for the tasks, in contrast to the original state-action space containing all kinds of inappropriate joints configurations. This potentially makes exploration more efficient within a compressed and meaningful latent space.

However, despite similar asymptotic performance on the pick and place and peg insertion tasks (depicted as dashed curves), UNN agents systematically converge to lower asymptotic average cumulative rewards. We hypothesize that it is due to the nature of the latent space. It is a compressed representation of the state-action space of each robot constructed from a finite number of trajectories. As such, it may not include the optimal policies reached by RL agents, which results in slower and sub-optimal trajectories for task execution. Hence, as the reward functions considered penalize slow behaviors, UNN agents will end up with a lower reward even though they complete the task. This is not the case in the ball catcher task because the ball cannot be catched faster or slower than its throwing speed. Overall, these results seem to indicate that a UNN agent trained from scratch will learn slightly faster than the RL agent on the same robot. Still, it may be unable to obtain similar rewards depending on the task and reward function shape, which indicates a less optimal policy with respect to the RL objective.

8.3. Over-fitting

In essence, our methodology promotes robot-agnosticity of the UNN module, by training within a latent space that is common among the robots considered for transfer. By doing so, our goal is to develop a module usable consistently by our pool of robots, regardless of their physical characteristics. However, in practice, depending on the robot platform, we observe significant over-fitting when learning a UNN module. To demonstrate this phenomenon, UNN modules have been trained on the source robots (training domain), transferred to the other robots (testing domain) every 50000 training steps and evaluated. Figure 9 depicts the performance curves of the UNN trained on each possible source robot and evaluated on the rest of the available robots for the peg insertion task. As shown, the UNN performance is relatively homogeneous at the beginning of training, both on source and target robots. However, after the UNN has converged on the training robot, performance starts to degrade on the test robots in most cases (except for Figure 9c with the braccio robot). It seems to indicate that the UNN module finds and exploits robot-specific strategies when trying to keep optimizing the RL objective.

As a workaround, we have check-pointed the UNN models every 50000 training steps and take the best performing one. All the results presented in the previous sections were obtained using this simple method. Moreover, as mentioned in Section 5.4, we empirically found that sampling z from $B^i_{\phi_r}(\cdot|x_r)$ instead



(a) Braccio robot.

(b) Panda robot.

(c) UR10 robot.

Figure 9: Peg insertion task. Performance curves on considered robots: y-axis is the performance and x-axis is the number of training steps. The UNN agent was trained on the (a) Panda robot, (b) Braccio robot, (c) UR10 robot, and tested on the rest of the available robots. Performance on the test robots is recorded every 50000 steps.

of taking $z = \mu_{\phi_r}(x_r)$ improves overall transfers by acting as a form of domain randomization. We let the design of a more robust and sophisticated method to future works.

8.4. Perspectives

While our approach shows promising results in transferring policies across robots with different morphologies, it remains an early-stage and proof-of-concept methodology with several open avenues for refinement and enhancement. Here, we outline a number of critical future research directions aimed at overcoming current limitations and exploring the broader applicability of our method in more complex scenarios.

One key challenge that needs to be addressed is the method robustness when dealing with limited sensor inputs or environments plagued by noise. The quality of the latent space representation is likely influenced by the fidelity of sensory data, and in practice, robots may have varying sensor capabilities. To address this issue, future work should involve experimenting with robots that have degraded or limited sensory inputs in noisy environments, to assess whether our agent-agnostic latent space can effectively handle variability while still enabling efficient transfer. One potential solution to mitigate the impact of noise and improve robustness would be to include denoising in the VAE training process, as done in denoising autoencoders [41]. Moreover practical scenarios may also include complex sensory inputs such as images for state estimation. In this setting, separating the robot's internal state from the "object-relevant" state in the environment can be less straightforward. For instance, visual inputs might inadvertently capture parts of the robot itself, making it harder to distinguish between robot-specific and environment-specific information. In future work, we will investigate how to extend our approach to handle cases where sensor data (such as images) cannot be easily decoupled from the robot's state. Techniques like self-supervised learning or attention mechanisms could be explored to disentangle robot states from environmental observations in high-dimensional input spaces. Another possibility is to learn an extra module that performs task-state estimation from raw images using a perception foundation model [42]

Future work should also explore more complex transfer scenarios involving intricate dynamic interactions and constraints. For instance, we did not consider scenarios where the robot must avoid obstacles by adopting specific joint configurations that challenge its morphology, or tasks requiring different end-effector trajectories based on the distinct kinematics of the robots involved (e.g., situations where one robot's optimal trajectory is infeasible for another due to space limitations or kinematic reach). Similarly, extending the method to tasks involving dynamic interactions and complex force control (e.g., tasks re-

quiring fine manipulation with significant contact forces) is an essential next step. Theoretically, our latent space can be extended to incorporate force trajectories, as there is no conceptual barrier to doing so. However, this will necessitate collecting paired demonstrations that include force information. Future work will explore this extension by focusing on tasks such as a dynamic peg-in-hole insertions with tighter tolerances, to validate the method's robustness in handling force interactions. Such experiments will further illuminate the method's adaptability to real-world constraints and varied robot designs.

9. Conclusion

In this paper, we discussed Transfer Learning as a promising avenue to mitigate the infamous sample-inefficiency that currently plagues state-of-the-art Reinforcement Learning algorithms. Motivated and inspired by several prior works, we introduced a simple and yet efficient methodology to transfer knowledge between agents, regardless of their body morphology. At the core of our approach: a robot-agnostic latent space constructed from time-aligned demonstrations on the considered robots, followed by the training of a task-module within it. Our experimental results showcased (i) close to baselines zero-shot performance in most of the transfers considered, (ii) significantly less required training samples than baselines, when fine-tuning is needed to recover performance and (iii) slight sample-efficiency increase when training an agent from scratch thanks to the prior knowledge included in the bases. Overall, our approach demonstrates an undeniable superiority in terms of sample-efficiency over regular RL training and does not require a large additional computation cost to setup.

However, several challenges remain to be solved. Despite very convincing results, directions for future improvements are obvious when looking at the data. Our approach suffers from: (i) occasional lower asymptotic cumulative reward which indicates a less optimal policy with respect to the RL objective and (ii) over-fitting to the source domain which requires a thorough testing of the model's checkpoints on the targeted robots. From a practical point of view, using unpaired and/or unaligned trajectories (instead of time-aligned demonstrations) for latent space building may prove a valuable addition to our framework. Addressing all these issues will motivate future works. Additionally, it could also be interesting to investigate more sophisticated types of VAE models to study the impact of the structure of the latent space for downstream model optimization and transfer. We believe this work highlighted the major potential of Transfer Learning applied to RL and hope that it will inspire more research efforts in this very promising, yet under-studied field of Reinforcement Learning.

10. Acknowledgements

This research was supported by the French Research Agency ANR through the AIM project. We gratefully thank Melodie Daniel and Laurent Lequievre for their assistance and insightful suggestions regarding the operation of the motion capture system. Olivier Stasse was funded by the European Project euROBIN (HORIZON-CL4-2021-DIGITAL-EMERGING-01).

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, et al., Playing atari with deep reinforcement learning, in: Advances in neural information processing systems (NeurIPS) deep learning workshop, 2013.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, et al., Human-level control through deep reinforcement learning, in: Nature, n°518, pp529-533, 2015.

[3] T. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, et al., Continuous control with deep reinforcement learning, in: 4th International Conference on Learning Representations (ICLR), 2016.

[4] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, in: arXiv preprint arXiv:1707.06347, 2017.

[5] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, in: International conference on machine learning (ICLR), 2018.

[6] I. Akkaya, M. Andrychowicz, et al., Solving rubik's cube with a robot hand, in: arXiv preprint arXiv:1910.07113, 2019.

[7] V. Tsounis, M. Alge, J. Lee, DeepGait: Planning and Control of Quadrupedal Gaits using Deep Reinforcement Learning, in: IEEE Robotics and Automation Letters, vol 5 n°2 pp 3699-3706, 2020.

[8] R. Jangir, G. Alenya, C. Torras, Dynamic cloth manipulation with deep reinforcement learning, in: IEEE International Conference on Robotics and Automation (ICRA), 2020.

[9] Y. Chebotar, A. Handa, V. Makoviychuk, et al., Closing the sim-to-real loop: Adapting simulation randomization with real world experience, in: International Conference on Robotics and Automation (ICRA), 2019.

[10] F. L. Da Silva, A. H. R. Costa, A survey on transfer learning for multiagent reinforcement learning systems, in: Artificial Intelligence Research, 2019.

[11] M. Mounsif, S. Lengagne, B. Thuilot, A. Lounis, BAM ! base abstracted modeling with universal notice network : Fast skill transfer between mobile manipulators, in: 7th 2020 International Conference on Control, Decision and Information Technologies (IEEE-CoDIT), 2019.

[12] N. Jaquier, M. C. Welle, A. Gams, K. Yao, B. Fichera, A. Billard, A. Ude, T. Asfour, D. Kragić, Transfer learning in robotics: An upcoming breakthrough? a review of promises and challenges, arXiv preprint arXiv:2311.18044 (2023).

[13] K. Kim, Y. Gu, J. Song, S. Zhao, S. Ermon, Domain Adaptive Imitation Learning, Technical Report, 2020. ArXiv:1910.00105.

[14] Z. Zhu, K. Lin, J. Zhou, Transfer learning in deep reinforcement learning: A survey, in: IEEE Transactions on Pattern Analysis and Machine Intelligence, 2020.

[15] M. Wan, T. Gangwani, J. Peng, Mutual information based knowledge transfer under state-action dimension mismatch, in: Conference on Uncertainty in Artificial Intelligence (UAI), 2020.

[16] N. Beck, A. Rajasekharan, H. Tran, Transfer reinforcement learning for differing action spaces via q-network representations, in: arXiv preprint: arXiv:2202.02442, 2022.

[17] T. Chen, A. Murali, A. Gupta, Hardware conditioned policies for multi-robot transfer learning, in: Advances in neural information processing systems (NeurIPS), 2018.

[18] C. Devin, A. Gupta, T. Darrell, P. Abbeel, S. Levine, Learning modular neural network policies for multi-task and multi-robot transfer, in: IEEE International Conference on Robotics and Automation (ICRA), 2017.

[19] L. Jackson, S. Eckersley, P. Senior, S. Hadfield, HARL-A: Hardware Agnostic Reinforcement Learning Through Adversarial Selection, in: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021.

[20] A. Ghadirzadeh, X. Chen, P. Poklukar, C. Finn, et al., Bayesian Meta-Learning for Few-Shot Policy Adaptation Across Robotic Platforms, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021.

[21] A. Gupta, C. Devin, Y. Liu, P. Abbeel, S. Levine, Learning Invariant Feature Spaces to Transfer Skills with Reinforcement Learning, in: International Conference Learning Representation (ICLR), 2017.

[22] C. Finn, P. Abbeel, S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in: International conference on machine learning (ICML), 2017.
[23] Q. Vuong, S. Levine, H. R. Walke, K. Pertsch, A. Singh, R. Doshi, C. Xu, J. Luo, L. Tan, D. Shah, et al., Open X-Embodiment: Robotic learning datasets and RT-X models, in: 2nd Workshop on Language and Robot Learning: Language as Grounding, 2023.
[24] P. Jian, E. Lee, Z. Bell, M. M. Zavlanos, B. Chen, Policy stitching: Learning transferable robot policies, in: Conference on Robot Learning, PMLR, 2023, pp. 3789–3808.
[25] Q. Yang, J. A. Stork, T. Stoyanov, Learn from robot: Transferring skills for diverse manipulation via cycle generative networks, in: 2023 IEEE 19th International Conference on Automation Science and Engineering (CASE), IEEE, 2023, pp. 1–6.
[26] D. Hejna, L. Pinto, P. Abbeel, Hierarchically decoupled imitation for morphological transfer, in: International Conference on Machine Learning (ICML), 2020.
[27] N. Makondo, B. Rosman, O. Hasegawa, Accelerating Model Learning with Inter-Robot Knowledge Transfer, in: IEEE International Conference on Robotics and Automation (ICRA), 2018.
[28] N. Makondo, B. Rosman, O. Hasegawa, Knowledge transfer for learning robot models via local procrustes analysis, in: 2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids), 2015.
[29] D. P. Kingma, M. Welling, Auto-encoding variational Bayes, in: 2nd International Conference Learning Representation (ICLR), 2014.
[30] I. Higgins, L. Matthey, A. Pal, C. P. Burgess, et al., beta-VAE: Learning basic visual concepts with a constrained variational framework, in: International Conference Learning Representation (ICLR), 2017.
[31] E. Schönfeld, S. Ebrahimi, S. Sinha, T. Darrell, Z. Akata, Generalized zero-shot learning via aligned variational autoencoders, in: Conference on Computer Vision and Pattern Recognition (CVPR), 2019.
[32] J. Schulman, P. Moritz, S. Levine, M. Jordan, P. Abbeel, High-Dimensional Continuous Control Using Generalized Advantage Estimation, in: Proceedings of the International Conference on Learning Representations (ICLR), 2016.
[33] M. E. Taylor, P. Stone, Y. Liu, Transfer learning via inter-task mappings for temporal difference learning., in: Machine Learning Research (JMLR), vol 8, pp 2125-2167, 2007, 2007.
[34] M. Cuturi, M. Blondel, Soft-dtw: a differentiable loss function for time-series, in: International conference on machine learning, PMLR, 2017, pp. 894–903.
[35] M. Müller, Dynamic time warping, in: Information retrieval for music and motion, Publisher Springer Berlin, Heidelberg, 2007, 2007.
[36] J.-Y. Zhu, T. Park, P. Isola, A. A. Efros, Unpaired image-to-image translation using cycle-consistent adversarial networks, in: Proceedings of the IEEE international conference on computer vision (ICCV), 2017.
[37] Q. Zhang, T. Xiao, A. A. Efros, L. Pinto, X. Wang, Learning cross-domain correspondence for control with dynamics cycle-consistency, in: International Conference on Learning Representations (ICLR), 2021.
[38] M. Akbulut, E. Oztop, M. Y. Seker, X. Hh, A. Tekden, E. Ugur, Acnmp: Skill transfer and task extrapolation through learning from demonstration and reinforcement learning via representation sharing, in: Conference on Robot Learning, PMLR, 2021, pp. 1896–1907.
[39] H. Aktas, Y. Nagai, M. Asada, E. Oztop, E. Ugur, Correspondence learning between morphologically different robots via task demonstrations, IEEE Robotics and Automation Letters (2024).
[40] M. Y. Seker, M. Imre, J. H. Piater, E. Ugur, Conditional neural movement primitives., in: Robotics: Science and Systems, volume 10, 2019.
[41] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, L. Bottou, Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion., Journal of machine learning research 11 (2010).
[42] R. Firoozi, J. Tucker, S. Tian, A. Majumdar, J. Sun, W. Liu, Y. Zhu, S. Song, A. Kapoor, K. Hausman, et al., Foundation models in robotics: Applications, challenges, and the future, The International Journal of Robotics Research (2023) 02783649241281508.
[43] S. Starke, N. Hendrich, D. Krupke, J. Zhang, Evolutionary multi-objective inverse kinematics on highly articulated and humanoid robots, in: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017.
[44] M. Blondel, A. Mensch, J.-P. Vert, Differentiable divergences between time series, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2021, pp. 3853–3861.

Appendix A. Building the latent space

We arbitrarily chose to use the Braccio-Panda pair for the latent space creation. We aligned the bases of the UR10 robot using the Braccio robot input module as reference. The **primitive task** used for the bases training was a simple **reaching task** as it is the basic building block of many more sophisticated skills . A target is randomly moving in the robot work space and the goal of the agent is to put the robot's effector at the desired pose.

The optimal policies for the primitive task were obtained with analytical methods [43]. As the bases represent the robots from the UNN's point of view, it is possible to include a priori knowledge of how the robot should behave. For example, as the tasks that we want our robots to achieve require the effector to remain vertical, we constrained our inverse kinematic solutions accordingly. As a consequence, the output bases will mostly generate trajectories suited for the given tasks.

We determined the optimal size of the latent space based on the reconstruction error (as measured by the mean squared error between predicted state $\hat{x}_r$ and true state x_r) for a fixed KL-divergence coefficient β (see Figure A.10). The reconstruction error is a good metric in this case, because it measures how much information about the input data is contained in the latent variables. Naturally, a low dimensional bottleneck (a dimension of 2 for instance) has not enough capacity to encode all the required information for proper reconstruction by the decoder. On the contrary, increasing the latent space size too much results in over-fitting as the network starts memorizing the data rather than finding a robust and meaningful low-dimensional representation, thus increasing reconstruction error on the test set. As shown in Figure A.10, a latent space of dimension 6 achieves the best results for an input size of 10 (Braccio) and 14 (Panda). This intuitively makes sense as the latent features should at least encode the 6 DoFs of the effector in space. As such, we expect this dimension to work well with most robot arms. More complex robots or primitive tasks may require a larger latent space. As such, mobile bases or legged locomotion might indeed require to search for the hyper-parameters that minimize the reconstruction error as these systems probably require additional dimensions to encode the greater variety of tasks and control requirements, and their latent space could be problem-dependent.

The recorded trajectories will ultimately define the latent space, so it is paramount to ensure that we visit a large variety of states in the work space to obtain a sufficiently rich latent space. In other words, it should have a large enough data coverage (with respect to the UNN tasks) because at inference-time, the output base will most likely generate interpolation of the trajectories seen during training. We chose to record a dataset consisting of 100000 pairs of similar states. In order to get these, the collected datasets were time-aligned as discussed in section 5.1. To do so, the target was moving at constant speed

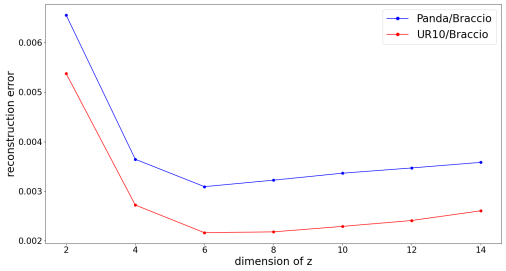


Figure A.10: Averaged reconstruction error (on the test set) as a function of the dimension of the latent variable z, for the bases training.

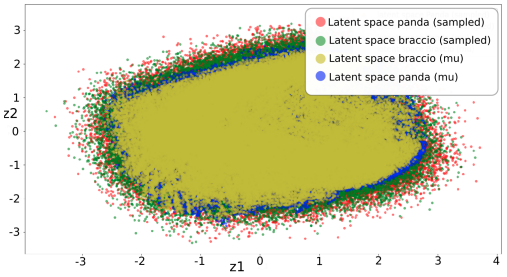


Figure B.11: Latent space for the braccio/panda pair after applying PCA.

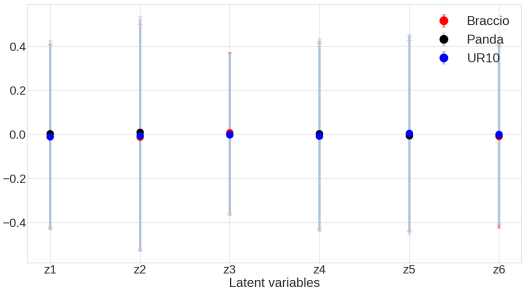


Figure B.12: Mean and standard deviation for each latent variable.

and visiting the same locations during the primitive task execution, for all robots considered. The robot state (joints position and velocity) was sampled at 40 Hz. In total, the trajectory recording took approximately 45 minutes per robot. As for the bases training, it took around 15 to 20 mins to complete on a single CPU without GPU parallelization.

Appendix B. Regularizing the latent space

When training the bases, a standard normal distribution prior over the latent vectors is applied, because it is easy to sample from when generating new data. In our case, this prior is all the more convenient because it places the latent space at a known location (around the origin) and encourages it to remain inside a known boundary thanks to the regularization effect of the KL-penalty in the cost function. These properties are desirable because it lets us know how to constrain the UNN module's action space for efficient exploration during training, as it will eventually act as the sampler. Without regularization, the bases training could result in an unbound and sparse latent space with clusters scattered very far apart from each other and "holes" without meaningful decoded states in between. Moreover, it is crucial for a successful transfer that the UNN does not wander outside of the latent space region seen during bases training as the output bases for both robots would not be able to decode similar states in this case. Regarding the weighting constant of Equation 13, the exact numerical values of δ and λ do not matter much but their magnitude should be similar to the reconstruction losses. However, the KL-divergence weight β should be carefully chosen as too much regularization would create an uninformative latent space (a standard gaussian). Too little would lead to the decoder over-fitting and prevent a useful transfer. Striking the right balance is crucial. In our experiment a significantly smaller weight gave the best results. This is the most sensitive hyperparameter in our framework. The Figure B.11 depicts a healthy latent space after applying principal component analysis (PCA) for dimension reduction (*mu* components are the mean of the gaussian distributions for the latent features). As seen, the latent space, of both robots are well aligned and regularized (clustered around zero, see Figure B.12). The latent features have not collapsed (i.e they are not indiscriminately mapped to 0) and they are well spread to encode as much information as possible.

Appendix C. Velocity control vs position control

During our early experiments, we initially chose to only encode and decode joints position when solving a task with the UNN. However, generating the exact joints positions suited to solve the task is extremely hard to get. It may not even be possible to find a set of latent points $z \subset Z$ such that $B^o(z) = x_r{}^*$, with $x_r{}^* \subset X_r$, one set of possible joint configuration solving the task. In other words, B^o is not surjective. One solution could be to enlarge the trajectories dataset used to create the latent space and use a more powerful decoder to increase the likelihood of generating x^*. A more practical solution is simply to generate the velocities leading to x^*.

Appendix D. Dealing with misaligned trajectories

While in practice, time-based alignment can be achieved straightforwardly on simple settings, it may not be robust enough when both robots are performing the primitive task at somewhat different speeds, or if the sampling loop does not run at a perfectly fixed rate. To address this issue and guaranty a reasonable alignment and state-pairing regardless of of possible desynchronization, we propose to rely on trajectory pairing rather than state-pairing. More formally, if we denote by $\pi^{*,\mathcal{T}}_{r_1}$ and $\pi^{*,\mathcal{T}}_{r_2}$ the optimal policies for primitive task $\mathcal{T}$ on respectively, robot r_1 and robot r_2, then:

$$\tau^{T_1}_{r_1}(s^{\mathcal{T}}_{g_1}) \approx \tau^{T_2}_{r_2}(s^{\mathcal{T}}_{g_2}) \iff s^{\mathcal{T}}_{g_1} = s^{\mathcal{T}}_{g_2} \tag{D.1}$$

where $\tau^{T}_{r_i}(s_g) = (s^{t1}_{r_i}, s^{t2}_{r_i}, ..., s^{T}_{r_i}) \in \mathbb{R}^{T \times N}$ is a trajectory of length T with N features, landing on goal state $s^{\mathcal{T}}_{g}$. In other words, if two trajectories reach the same goal state s_g, they are deemed

equivalent and can be matched. We replace the state-wise similarity loss with a trajectory-wise SoftDTW divergence [44]:

$$L_{sim}(\tau^{T_1}_{r_1}, \tau^{T_2}_{r_2}) = \text{SoftDTW}^{\gamma}(\tau^{T_1}_{r_1}, \tau^{T_2}_{r_2}) - \frac{1}{2}\,\text{SoftDTW}^{\gamma}(\tau^{T_1}_{r_1}, \tau^{T_1}_{r_1}) - \frac{1}{2}\,\text{SoftDTW}^{\gamma}(\tau^{T_2}_{r_2}, \tau^{T_2}_{r_2}) \tag{D.2}$$

Additionally, we also remove the L_{CA} cross-alignment term as it requires paired states and cannot be adapted for paired trajectories of different lengths. Finally, the overall objective function is the following:

$$\min_{\theta_{r_1},\theta_{r_2},\phi_{r_2},\phi_{r_1}} \sum_{(x_{r_1},x_{r_2})\in D} L_{B_{r_1}} + L_{B_{r_2}} + \delta L_{sim} \tag{D.3}$$

We evaluate this alignment approach on the same benchmark. We enforce misaligned trajectories by setting different task execution speeds. More specifically, taking the UR10 robot as reference, the Panda robot is moving 20% faster and the Braccio robot 40% faster. We generate 2500 equivalent trajectories on a Reaching task across robots and proceed to train a UR10-Panda pair of bases. The final preliminary step is to fit the Braccio bases to align with the Panda ones. The neural networks architecture is the same as before and we set δ, the weighting constant for L_{sim}, to 0.0025. In the table below, we average the transfer performance over environments for a more concise display. Overall, transfer performances reported

Source\Target	**Braccio**	**Panda**	**UR10**
Braccio	99	81	93
Panda	90	96	96
UR10	80	92	99

Table D.6: Performance obtained for zero-shot transfers on Pick and Place, Ball Catcher and Peg Insertion when using trajectory-pairing aligned with SoftDTW.

in this preliminary experiment with misaligned trajectories are lower than those reported in Section 7. Performance seems to drop as the misalignment increases. However, the misalignment considered were voluntarily exaggerated to test the robustness and limitations of this alignment approach. In practice, a desynchronization of 40 %, as studied for the Braccio robot, is unlikely when working with industrial-grade robots. Naturally, future work will investigate how efficiently full performance can be recovered when fine-tuning on the target robots but we believe that this simple modification shows promising robustness to time-shifts.