Programmation de cobots : de l'apprentissage de trajectoires à leur acceptabilité

Amélie Legeleux

THESE DE DOCTORAT DE

L'UNIVERSITE BRETAGNE SUD

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

Par

Amélie LEGELEUX

« Programmation de cobots : de l'apprentissage de trajectoires à leur acceptabilité »

Thèse présentée et soutenue à Plouzané, le 31 Mai 2022
Unité de recherche : Lab-STICC, CNRS UMR 6285
Thèse N° : 629

Rapporteurs avant soutenance :

Olivier LY	Professeur des Universités, Université de Bordeaux
Alexandre PAUCHET	Maître de conférences HDR, INSA de Rouen

Composition du Jury :

Président :	Olivier LY	Professeur des Universités, Université de Bordeaux
Examinateurs :	Alexandre PAUCHET	Maître de conférences HDR, INSA de Rouen
	Sophie SAKKA	Maître de conférences HDR, École Centrale de Nantes
	Aurélie LANDRY	Maître de conférences, Université Grenoble Alpes
Dir. de thèse :	Dominique DUHAUT	Professeur des Universités, Université Bretagne Sud
Co-dir. de thèse :	Cédric BUCHE	Professeur des Universités, CNRS (Australie)

Invités :

Nathalie LE BIGOT	Maître de conférences, Université de Bretagne Occidentale
Maria KYRARINI	Assistant Professor, Santa Clara University (États-Unis)

« Persévérer, secret de tous les triomphes. »
Victor Hugo[1]

[1] L'homme qui rit, 1869

Remerciements

Je tiens à remercier, tout d'abord, Cédric BUCHE, mon co-directeur de thèse, pour m'avoir proposé ce sujet de thèse. Sans son soutien, son expertise, sa grande disponibilité, ces travaux n'auraient pu être aussi aboutis. La relation de confiance qu'il a su créer m'a permis d'échanger en toute franchise, ce qui m'a apporté une grande sérénité. Je le remercie également de m'avoir permis d'aller en Floride pour travailler sur un nouveau robot et de m'avoir donné la possibilité de rencontrer des chercheurs australiens.

Je ne peux aussi manquer de remercier Dominique DUHAUT, en sa qualité de directeur de thèse, pour son encadrement, sa disponibilité et son soutien. Grâce à sa confiance en mes capacités, je suis allée en Australie pour découvrir la compétition internationale RoboCup, expérience grâce à laquelle notre équipe RoboBreizh a enchainé diverses victoires.

J'exprime toute ma gratitude à Nathalie LE BIGOT, Maître de Conférences en psychologie, pour son soutien sans faille, ses conseils avisés et le temps qu'elle a su m'accorder bien que n'appartenant pas à l'équipe du projet « Prog4Yu », cadre de cette thèse. Son expertise fut indispensable puisque la psychologie n'était pas dans mon champ de compétence.

Cette thèse n'aurait pu exister sans le financement du projet Prog4Yu ANR-18-CE10-0008, conduit par Damien PELLIER. Ce projet vise à faire sortir le robot de sa cage et à se changer en cobot, robot collaboratif travaillant avec des opérateurs de production dans un espace de travail partagé.

Je tiens également à souligner le lourd travail de relecture et d'analyse de mes travaux effectués par les rapporteurs, Olivier Ly, Professeur des Universités et Alexandre Pauchet, Maître de Conférence HDR. Grâce à leur précieux conseils et les diverses remarques des membres du jury, cette thèse gagne en précision et en qualité. Parmi les membres du jury, je remercie Aurélie Landry pour son soutien et ses conseils ainsi qu'Olivier LY pour ses remarques très constructives lors des réunions avec le Comité de Suivi Individuel (CSI).

Je remercie grandement Fabrice HARROUET, Maître de Conférences, pour avoir enrichi mon savoir et mes connaissances techniques en matière de langage C++ et de réseau informatique. Je remercie également Laurent GAUBERT, Maître de Conférences, pour m'avoir apporté son expertise lors de mes doutes sur la partie mathématique du modèle d'apprentissage avec les données pondérées.

De nombreux étudiants ont contribué aux travaux présentés dans cette thèse. Je remercie vivement Jérémie DONJAT, excellent stagiaire en dernière année à l'ENIB. Sans lui, les travaux sur le pré-traitement des données pour l'apprentissage par démonstration n'auraient pu se concrétiser. Merci également à Paul CREPIN, Marie DUBOIS, Gaëtan LECLERCQ, Alexandre LOUIS, Arthur PASSET, Chloé MICHAUD, Maëlys TURPIN, étudiants en psychologie pour leur contribution à l'élaboration des expériences sur l'acceptabilité. Merci aussi à Anaïs RAISON pour ses conseils lors des mises en place des expériences et lors de l'analyse des résultats, en particulier pour l'utilisation du logiciel de statistiques R. Je tiens également à remercier Romain BIANNIC pour ses idées ingénieuses lors de la préparation de mes expériences et pour la modélisation 3D des supports pour verre.

Nicolas DUMINY utilisait le robot YuMi juste avant moi. Merci à lui pour ses conseils lors de la prise en main du robot. Je remercie ensuite les différents étudiants en projet PRI à l'ENIB pour leur recherche sur l'utilisation du robot YuMi. Je souhaite aussi remercier Axel LARIVIERE pour ses travaux sur le mode cartésien du robot Nao et celui du robot YuMi. Le robot YuMi n'aurait pu être utilisé sans l'aide et les conseils d'Ekhi LANIESSE, ingénieur robotique chez ABB. Je remercie également l'équipe du support technique d'ABB, notamment Philippe URIOT.

Je suis aussi reconnaissante envers François BETTANCOURT, Loïc LEROUX et les autres formateurs contactés qui n'ont pas hésité à inscrire leurs étudiants ou tout du moins, à relayer mon mail de demande de participants. Grâce à Floras DUROS, j'ai obtenu d'autres contacts de formateurs, intéressés par mon expérience sur les cobots.

I would like to thank Dr. Ubbo VISSER sincerely for his warm welcome at the University of Miami and for his advice on my research and his goodwill. Regarding my travel in the United States of America, I cannot help thinking about the indispensable assistance from Alban DELAMARRE regarding accommodation and providing me a reliable mean of transportation, a bike, which was very useful contrary to the late or missing buses. Many thanks to Stephanie LUNN who pick me up at the airport and for her kindness. I am grateful to all student for their support, their welcome and their cheerfulness: Joe MASTERJOHN, Elik KARTAL, Zishi WU, Shengxin LUO, Katarzyna WIKTORIA PASTERNAK, Pedro PEÑA.

I also thank Dr. Maria KYRARINI for the very instructive meting about the possible improvement of her thesis work. Many thanks to invite me at your « Women in Computer Science and Engineering: Mentoring Seminars ».

Travailler dans le Centre Européen de Réalité Virtuelle (CERV) m'a permis de rencontrer de nombreuses personnes et ce dès mon stage de dernière année à l'ENIB. Je remercie vivement Mihai POLCEANU pour ses nombreux conseils. Merci à tous mes collègues doctorants, qui vivent la même aventure que moi : Pierre MAHIEUX, Jean-Michel FAZZARI, Yann GLEMAREC, Maëlic NEAU, Antoine DIZET, Damien BOUCHABOU, Natnael ARGAW, Andreea-Oana PETAC, François LASSON, Serawork WALLELIGN. Je remercie également les trois premières docteures de l'ENIB, rencontrées au tout début de ma thèse : Cindy EVEN, Joanna TAOUM et Landy RAJAONARIVO. Je remercie tous les membres du Lab-STICC qui m'ont soutenue, accueillie et conseillée tout au long de ces trois années consacrées à ma thèse.

Cette thèse n'aurait abouti sans le soutien inconditionnel de ma famille et ma belle-famille. Grâce à eux, j'ai eu le courage de poursuivre alors que l'abandon était réellement envisagé. Merci à toute ma famille de m'encourager, de croire en moi à chaque instant. La rédaction de cette thèse n'aurait su s'achever sans la relecture minutieuse de Claire, merci à toi. Toutes mes pensées vont également à l'homme de ma vie, qui a su supporter mes sautes d'humeur en toutes circonstances.

Pour finir, je remercie Robin pour avoir enrichi ma vie personnelle afin de ne pas toujours trop travailler, du barbecue au cinéma en passant par les restaurants et les mangas, il a su m'offrir des moments de répit.

Sommaire

Introduction générale

« Le plus difficile n'est pas trouver la solution mais de se poser la bonne question. »

Cédric Villani[2]

Sommaire

[2] Propos recueillis lors de la Fête de la Science AuRA en 2016

Quand les ordinateurs ont commencé à émerger, leur utilisation nécessitait la connaissance du langage de programmation et du fonctionnement de ses composants. Par la suite, les ordinateurs portables se sont généralisés avec une interface plus simple à utiliser par le grand public. Il est maintenant courant d'utiliser un ordinateur pour effectuer de nombreuses tâches. Dans un environnement qui n'est plus seulement à deux dimensions, les robots, machines qui interagissent avec leur environnement, posent des problématiques similaires à l'utilisation des premiers ordinateurs. Sans connaissance en programmation et en robotique, il est actuellement impossible d'utiliser ce type de nouvelle technologie. L'apprentissage de nouvelles capacités sans nécessité de programmation devient alors un nouveau défi à relever (Calinon, 2009).

Aux débuts des machines autonomes, les automates se définissent comme des « machines qui imitent la forme et/ou le comportement d'un être vivant par le moyen des dispositifs mécaniques, pneumatiques, hydrauliques, électriques ou électroniques qu'elles contiennent » (Meyer, 2016). Avec l'arrivée des capteurs, les automates se transforment en robots capables de percevoir leur environnement et d'interagir avec en conséquence. Le mot **robot** est apparu pour la première fois dans une pièce de théâtre de 1920 grâce à l'écrivain tchèque Karel Čapek. De nombreux films, livres ou autres œuvres culturelles ont dépeint les robots sous diverses formes et avec des capacités et des objectifs très variés (Gelin & Guilhem, 2016). Au fur et à mesure du développement des capacités de calcul et de l'évolution des capteurs, les robots, devenus de plus en accessibles, s'imposent. Qui n'a pas déjà rencontré ou ne dispose pas d'un robot aspirateur de nos jours ?

L'industrie 4.0 se définit comme la quatrième révolution industrielle, après la mécanisation, l'industrialisation et l'automatisation. Cette industrie du futur cherche à moderniser les outils de production actuels à l'aide de nouvelles technologies pour plus de sureté et de flexibilité (Stäubli, 2021). Parmi les nouvelles technologies, on retrouve le cloud et la cybersécurité, l'Internet des Objets (IoT), la fabrication additive avec les imprimantes 3D, la réalité virtuelle et augmentée, la simulation et les jumeaux numériques, et enfin le big data qui permet l'émergence de l'intelligence artificielle et les décisions prises en temps réel par exemple.

La prochaine évolution des robots est très probablement liée aux cobots. Le **cobot**, ou robot collaboratif, est conçu afin d'interagir avec un opérateur humain, dans un espace partagé, pour la réalisation d'une tâche en collaboration de manière sécuritaire et à l'aide d'une programmation simplifiée (Michaelis et al., 2020). Si la collaboration nécessite un contact entre l'opérateur humain et le cobot, ce dernier dispose « de matériaux légers, des contours arrondis, un rembourrage, des peaux […] et des capteurs à la base du robot » (Stäubli, 2021).
(Bauer et al., 2016) définit cinq niveaux de coopération entre un opérateur humain et un robot : la cellule, la coexistence, la synchronisation, la coopération et la collaboration (Figure 1).

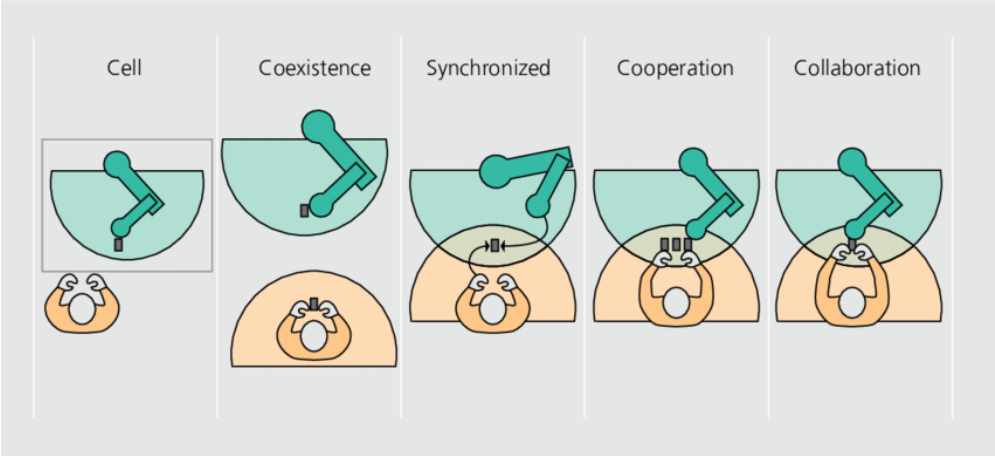


Figure 1 : Niveaux de coopération entre un humain et un robot (Bauer et al., 2016)

La cellule est le mode existant pour les robots industriels, c'est-à-dire qu'ils sont placés dans une cage pour fonctionner. A ce niveau, il n'y a pas vraiment de coopération.

La coexistence est similaire à la cellule avec la cage en moins. L'opérateur humain et le robot travaillent côte à côte sans partager d'espace de travail.

Dans les niveaux suivants, l'opérateur humain et le robot travaillent dans un espace commun. Le niveau synchronisé signifie que l'opérateur humain et le robot travaille à des moments différents, l'un puis l'autre.

Lorsque l'opérateur humain et le robot travaillent en même temps mais sur un objet différent, cela s'appelle la coopération.

La collaboration est le niveau supérieur, où la tâche s'effectue en simultané sur le même objet.

Selon les analyses de l'*International Federation of Robotics* (IFR, 2021), les robots s'imposent de plus en plus dans les industries (Figure 2). En 2020, dans le monde, plus de 3 000 robots était opérationnels dans les usines.

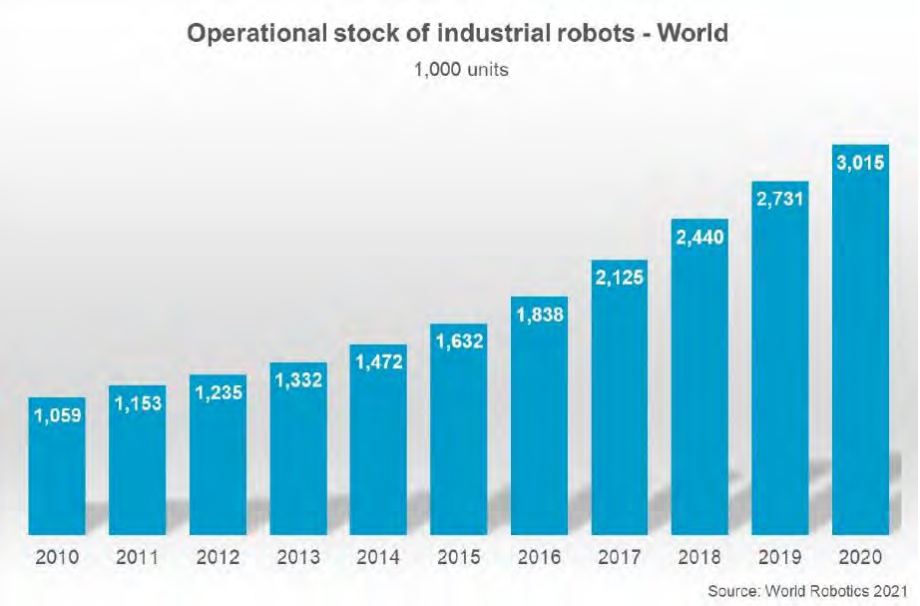


Figure 2 : Nombre de robots industriels opérationnels dans le monde (IFR, 2021)

Au niveau des marchés, le continent asiatique couvre la majorité des installations de robots industriels (Figure 3).

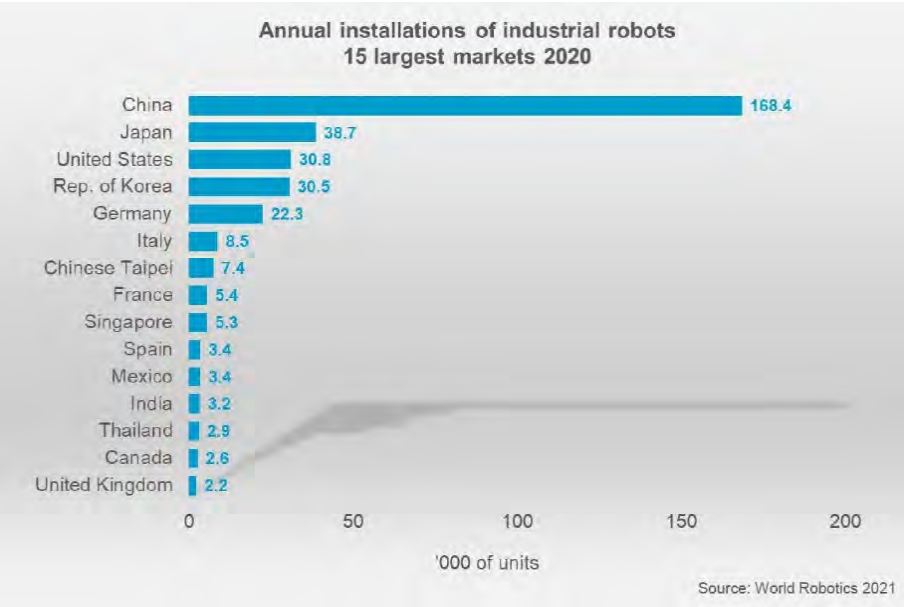


Figure 3 : Nombre d'installations annuelles de robots industriels dans le monde en 2020 en fonction des pays (IFR, 2021)

La plus grande part des robots industriels se retrouvent dans le domaine de l'électronique (Figure 4).

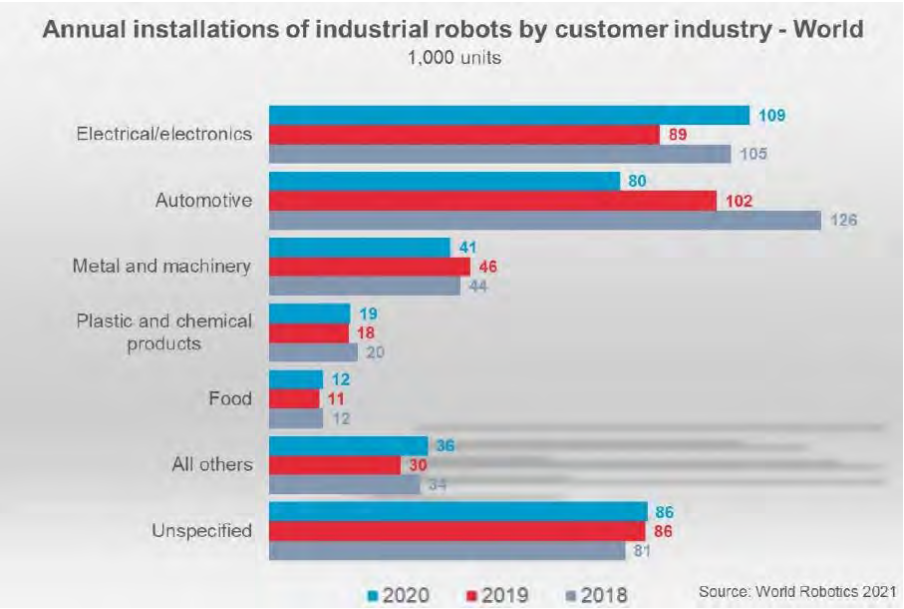


Figure 4 : Nombre d'installations annuelles de robots industriels dans le monde en fonction du type d'industrie (IFR, 2021)

Parmi les robots industriels mondiaux, la part des cobots a augmenté de 6% en 2020 (Figure 5).

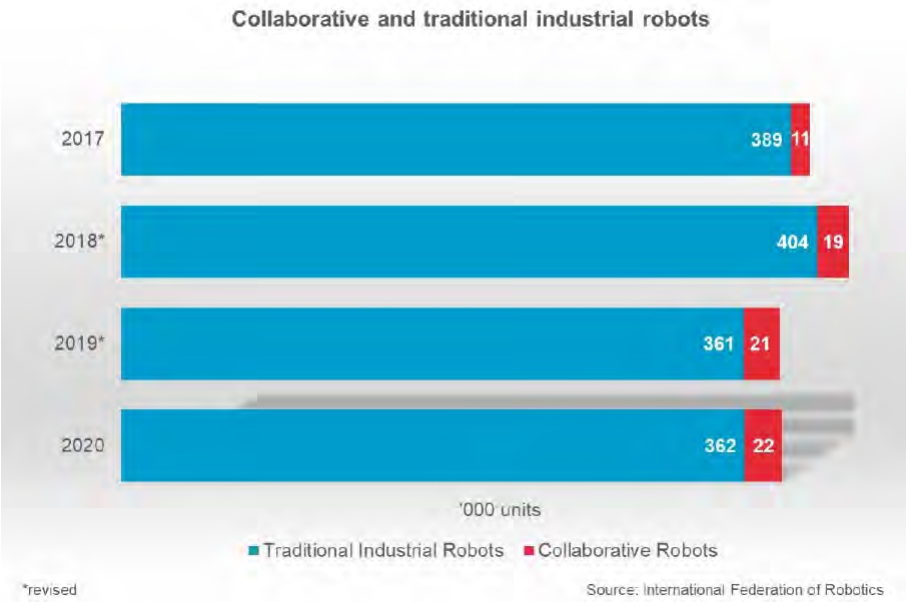


Figure 5 : Part des robots collaboratifs et traditionnels dans l'ensemble de robots industriels (IFR, 2021)

Les cobots, contrairement aux robots traditionnels, offrent plus de flexibilité et sont déployés plus rapidement grâce à leur programmation simplifiée et accessible par des utilisateurs non-experts. Selon (Stäubli, 2021), la robotique collaborative cherche à préserver l'opérateur humain en diminuant les troubles musculo-squelettiques (TMS) et la pénibilité des tâches répétitives. Le savoir-faire est également conservé puisque le cobot a besoin d'un opérateur pour lui apprendre le mouvement correct à réaliser. Un gain de production peut être observé puisque l'opérateur humain est vecteur de flexibilité lors d'un changement de production ; ce dernier peut se focaliser sur son expertise.

Cependant, pour garantir la sécurité des opérateurs, les cobots fonctionnent avec une vitesse plus faible que les robots et une charge embarquée également moins lourde. L'assimilation de ces cobots doit être prise en compte lors de leur installation afin d'améliorer leur acceptabilité par les futurs utilisateurs.

L'arrivée des cobots pose de nombreux défis parmi lesquels l'acceptabilité par les opérateurs, la sécurité et la programmation du cobot. Le cobot, contrairement au robot traditionnel, nécessite l'expertise d'un opérateur humain pour fonctionner.

Par conséquence, en plus du développement physique du cobot (mécanique, électronique et informatique), les sciences humaines et sociales deviennent indispensables pour l'intégration du cobot dans l'industrie (Calinon & Billard, 2007b ; Kleinpeter, 2015 ; Wang et al., 2019).

Le cobot, sorti de sa cage, peut poser des problèmes de sécurité comme le risque de chute de charge, de choc avec le robot, de coincement de membres de l'opérateur dans la pince du robot ou de conséquences liées à l'activité du robot telles des brûlures, coupures et autres activités dangereuses (Bauer et al., 2016 ; Stäubli, 2021).

La programmation du cobot doit être simple d'utilisation et compréhensible par des utilisateurs non-experts (Kragic et al., 2018 ; Wang et al., 2019). (El Zaatari et al., 2019) identifie trois composantes essentielles pour la programmation des cobots : la communication, l'optimisation et l'apprentissage. La communication peut être orale, non-verbale ou à l'aide d'une interface utilisateur. L'optimisation vise à l'amélioration continue autant du côté du geste du cobot en fonction de son environnement (temps, obstacle par exemple) que du côté utilisateur (confort, confiance, qualité produit par exemple). L'apprentissage du cobot est similaire à celui des êtres humains : avec des démonstrations, des essais-erreur, des retours de l'enseignant, par exemple.

L'objectif du projet Prog4Yu ANR-18-CE10-0008, projet qui définit le cadre de cette thèse, est de « développer une approche de « programmation par démonstrations » des cobots à l'intention des opérateurs de productions, non expert en langages de programmation mais ayant l'expertise de la réalisation des tâches, tout en garantissant que cette approche soit acceptable et intelligible par ces opérateurs ».

Ce projet, lié à l'industrie 4.0, cherche à développer la programmation par démonstration des cobots, tout en vérifiant des aspects psychologiques comme l'acceptabilité. La présente thèse prend comme base, les objectifs du projet Prog4Yu.

Problématique

Malgré les promesses liées aux cobots, leur utilisation actuelle est très loin d'être satisfaisante. Dans une étude de (Michaelis et al., 2020), neuf experts travaillant en industrie - ingénieurs et développeurs - ont été questionnés sur leur expérience avec les cobots.

Les résultats montrent que les applications actuelles des cobots sont liées à des tâches simples et de long terme, similaires aux utilisations des robots traditionnels. Ceci peut s'expliquer par la formation des ingénieurs très orientée robots traditionnels et automatisme. Les cobots sont alors pensés comme un cas spécifique des robots. Ils sont donc associés à des tâches similaires aux robots bien qu'ils puissent être employés pour des tâches complexes puisqu'ils sont liés à l'expertise des opérateurs humains.

Le second constat est la perception d'un besoin d'une formation complexe en programmation et en robotique pour pouvoir utiliser et programmer des cobots. Il est important de noter que cette constatation est contradictoire avec l'objectif principal des cobots : la programmation simple par des utilisateurs non-experts.

Au vu de la situation actuelle des cobots, comment peuvent-ils être programmés et apprendre simplement, avec des utilisateurs non-experts ? Seront-ils acceptés et compris par les opérateurs de production ? Cette thèse propose de répondre à ces questions via diverses contributions et expériences scientifiques réalistes.

Contributions scientifiques

Les diverses contributions scientifiques décrites dans cette thèse peuvent être présentées selon quatre différents challenges.

Challenge 1 : Comment obtenir des données qui seront ensuite utilisées pour l'apprentissage par démonstration ?

- **Acquisition des données via des démonstrations kinesthésiques** : le choix de ce type de démonstration est simple d'utilisation, intuitif et compréhensible par des utilisateurs non-experts (Introduction du Chapitre 1).
- **Étude et mise en place de méthodes de pré-traitement des données** : pour pouvoir être utilisées pour l'apprentissage par démonstration, les données doivent être alignées temporellement et filtrées (Section 1.1 et Chapitre 2).

Challenge 2 : L'apprentissage par démonstration peut-il s'adapter au souhait de l'utilisateur et fonctionner sur diverses plateformes robotiques ?

- **Mise en place d'une architecture générique** : cette architecture facilite l'ajout d'un nouveau robot (Section 3.1).
- **Proposition d'un apprentissage par démonstration avec des données pondérées** : la pondération des données est choisie par l'utilisateur et permet d'améliorer l'apprentissage sans modification des démonstrations (Section 1.2 et Chapitre 3).

Challenge 3 : La programmation par démonstration, développée pour des utilisateurs non-experts, est-elle comprise, utilisée et acceptée par ces derniers ?

- **Développement d'une interface utilisateur** : l'interface développée est conçue pour être simple d'utilisation et intuitive (Section 4.3.2.2).
- **Mise en place d'une formation sur la programmation par démonstration des cobots pour des utilisateurs non-experts** : la formation proposée a permis à tous les participants de comprendre et d'utiliser la programmation par démonstration (Section 4.3.2.3 et Section 4.3.2.4).
- **Expérience pour évaluer l'acceptabilité de la programmation par démonstration par de potentiels futurs utilisateurs des cobots** : l'expérience a montré que la programmation par démonstration est acceptée et comprise par les utilisateurs (Section 4.1 et Section 4.3).

Challenge 4 : Comment l'anthropomorphisme d'un robot est-il perçu ? A-t-il un impact sur leur utilisation et sur leur acceptabilité ?

- **Expérience sur l'anthropomorphisme perçue des robots** : un questionnaire en ligne a relevé des niveaux d'anthropomorphisme perçu des robots différents en fonction, seulement, de leur aspect visuel (Section 4.2).
- **Ajout de l'impact de l'anthropomorphisme dans l'expérience sur l'acceptabilité de la programmation par démonstration** : deux robots, de niveau d'anthropomorphisme différent, sont employés pour déterminer l'impact de l'anthropomorphisme (Section 4.1 et Section 4.3).

Ces contributions scientifiques sont reconnues par plusieurs publications scientifiques (voir Annexe 1). Les travaux de cette thèse montrent que les cobots peuvent apprendre simplement et être employés dans l'industrie à condition que leur environnement, leurs tâches et leur développement soient configurés selon ces robots d'un nouveau type et que les futurs utilisateurs soient associés dès le début du projet.

Organisation du manuscrit

Ce manuscrit s'organise autour de quatre chapitres :

- **Chapitre 1** : état de l'art autour des algorithmes pour le pré-traitement des données et pour l'apprentissage par démonstration ;
- **Chapitre 2** : étude et mise en œuvre de méthodes de pré-traitement des données dans le cadre de l'apprentissage par démonstration afin de déterminer la méthode la plus adaptée (Donjat et al., 2022) ;
- **Chapitre 3** : proposition d'une architecture générique pour l'apprentissage par démonstration avec un modèle d'apprentissage par démonstration adaptable selon le souhait de l'utilisateur (Legeleux et al., 2022) ;
- **Chapitre 4** : études et expériences autour de l'anthropomorphisme des robots et de l'acceptabilité de la programmation par démonstration par des utilisateurs non-experts (Fraune et al., 2022 ; Legeleux et al., 2021).

Chapitre 1

L’apprentissage par démonstration

« La connaissance scientifique possède en quelque sorte des propriétés fractales : nous aurons beau accroître notre savoir, le reste – si infime soit-il – sera toujours aussi infiniment complexe que l’ensemble de départ »

Isaac Asimov[3]

Sommaire

[3] I. Asimov: A Memoir, 1995 (trad. Hélène Collon, éd. Denoël,1996)

Les robots sont de plus en plus présents dans nos vies quotidiennes. Du robot tondeuse au robot de cuisine, ils arborent diverses formes et sont prédestinés à des tâches très variées. Cependant, chaque robot n'a généralement qu'une seule façon d'apprendre à bouger/agir de la manière souhaitée. Il faut pour cela écrire les lignes de code correspondant à la logique selon laquelle le robot devra se mouvoir. Cette implémentation, souvent réalisée par des experts, est parfois simplifiée à l'aide de blocs de fonction(s) qui masquent une logique de programmation utilisée par les experts informatiques. Lorsque le robot doit agir d'une autre manière, soit cela est prévu dans son code, soit il faut reprogrammer le robot. Cette reprogrammation demande du temps et de l'expertise. Dans le cas d'un besoin de reprogrammation d'un mouvement du robot, l'apprentissage par démonstration est une nouvelle manière de programmer le robot sans avoir besoin d'écrire une seule ligne de code (Zhu & Hu, 2018).

L'apprentissage par démonstration (LbD pour *Learning by Demonstration*), appelée également programmation par démonstration (PbD pour *Programming by Demonstration*) (Calinon, 2009) ou apprentissage par imitation, permet au robot d'apprendre une tâche à partir d'une ou plusieurs démonstration(s). Le robot apprend donc à partir de son expérience, comme un enfant. Un adulte montre à un enfant comment écrire en l'aidant à tenir un crayon et en lui montrant comment il est possible d'écrire. Inconsciemment, le fait de voir d'autres personnes écrire aide également l'enfant dans son apprentissage. Ces deux méthodes de démonstration existent dans l'apprentissage par démonstration. L'apprentissage par démonstration permet à un opérateur humain d'enseigner des mouvements/actions à un robot (Chernova & Thomaz, 2014 ; Ravichandar et al., 2020).

Les démonstrations peuvent se regrouper selon quatre types (Argall et al., 2009) :

- **La téléopération** : les mouvements du robot sont contrôlés par une personne, comme le montre la Figure 6. Soit le corps du robot est manipulé physiquement (**démonstration kinesthésique**), soit il est commandé par un contrôleur comme une manette. Ce type de démonstration permet au robot d'apprendre à partir de ses propres capteurs qui enregistrent le mouvement effectué.
- **Le mimétisme** : le robot imite un être humain ou un autre robot, ce qui implique un algorithme de suivi intégré dans le robot. De la même manière que pour la téléopération, le robot apprend à partir de ses propres capteurs.
- **L'observation externe** : un ou des capteur(s) externe(s) enregistre(nt) les mouvements réalisés par un opérateur humain, comme une caméra.
- **Les capteurs portables** : des capteurs portés par un opérateur humain enregistrent ses mouvements.

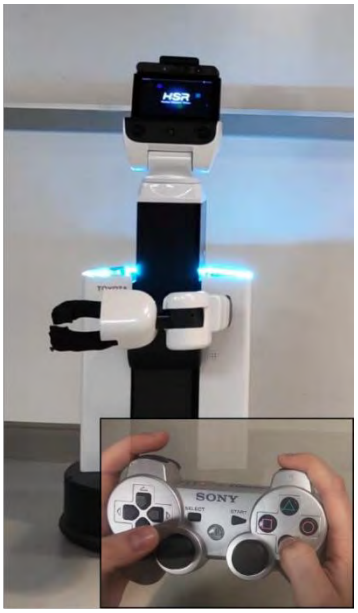
(a)   (b)

Figure 6 : Type de démonstration

(a) Démonstration kinesthésique avec le robot Pepper, (b) Téléopération avec une manette et le robot HSR

Selon le type de démonstration choisi, le lien entre les données enregistrées et les données qui commandent le robot n'est pas toujours trivial. Il se pose donc un problème de correspondance (Argall et al., 2009 ; Chernova & Thomaz, 2014). Un opérateur humain peut avoir plus de degré de liberté qu'un robot. Ainsi, la téléopération et le mimétisme offre un lien direct entre les démonstrations et le mouvement que peut faire le robot contrairement aux cas de l'observation externe et des capteurs portables. L'apprentissage kinesthésique sera employé par la suite pour faire les démonstrations. Ce type de démonstration est très intuitif pour un opérateur humain, puisqu'il s'agit d'un mode d'apprentissage utilisé dans sa vie de tous les jours pour enseigner un mouvement à un enfant ou une autre personne. Il est également simple à comprendre et correspond au contexte industriel pour enseigner des gestes techniques.

Une revue de la littérature sera effectuée dans la Section 1.1 afin de présenter les technologies existantes dans le pré-traitement des données qui seront utilisées ensuite pour les algorithmes d'apprentissage par démonstration (Section 1.2).

1.1 Pré-traitement des données pour l'apprentissage par démonstration

Les démonstrations fournissent des données qui peuvent être de diverses natures. Avec l'apprentissage kinesthésique, les données proviennent directement des capteurs du robot. Les données peuvent être obtenues en mode joint ou en mode cartésien (Calinon & Billard, 2008b). Le mode joint correspond directement à la position ou vitesse des articulations du robot. Les valeurs de position angulaire peuvent être exprimées en radian ou en degré.

Le mode cartésien correspond à la position de l'effecteur du robot, c'est-à-dire le bout de son bras, dans l'espace. Généralement, cette position est donnée selon un repère orthonormé défini par les axes x, y, z et les rotations respectives sont fournies en radian/degré ou sous forme de quaternion.

Ces données doivent être de bonne qualité car ce sont elles qui serviront de base pour l'apprentissage. Les démonstrations étant réalisées à l'aide d'une personne, leur qualité en est donc dépendante. L'opérateur humain peut commettre des erreurs, ajouter des mouvements parasites même sans le vouloir ou même sans s'en rendre compte, soit autant de possibilités d'impacter la fiabilité des données. Ainsi, deux phases se distinguent pour le pré-traitement des données, étape antérieure à l'apprentissage. Il y a l'alignement des données et la sélection des démonstrations.

L'alignement des données consiste à aligner temporellement les démonstrations d'une même tâche. En effet, un opérateur humain ne peut reproduire le même mouvement avec exactement la même vitesse, le même temps d'exécution.

La sélection des démonstrations permet de ne conserver que des démonstrations similaires, celles qui reproduisent le plus fidèlement le mouvement recherché. L'opérateur humain peut parfois faire une démonstration qui ne correspond pas du tout à la tâche ou rajouter un mouvement supplémentaire. Ces mauvaises données seront filtrées lors de la sélection des démonstrations. Ces deux phases sont indispensables pour favoriser un bon apprentissage.

Une méthode très utilisée pour traiter les données de séries temporelles est *Dynamic Time Warping* (DTW) (Berndt & Clifford, 1994 ; Müller, 2007 ; Sabbaghi et al., 2014). Cet algorithme compare deux séries de données, temporellement indépendantes.

Les couples de points de chacune des deux séries sont comparés à l'aide d'une mesure de distance comme la distance euclidienne ou la distance Manhattan (somme des valeurs absolues des différences de coordonnées). Tous ces écarts créent une matrice de coûts qui évalue la similarité entre chaque paire de points ; plus la similarité est grande, plus le coût est faible. Un coût de zéro représente deux points identiques.

Le *Warping Path* est ensuite calculé afin de trouver le chemin représentant le plus court chemin et ayant le plus faible coût dans la matrice de coût. Le score final se calcule en faisant la somme de la différence absolue entre chaque paire de points du *Warping Path*.

DTW permet donc d'aligner temporellement des séries temporelles. En définissant une référence, le *Warping Path* du DTW donne la possibilité de sélectionner des démonstrations (Kyrarini et al., 2016).

Néanmoins, la complexité du DTW entraîne un temps de calcul qui peut être très élevé en fonction de la taille des séries temporelles considérées (Movchan & Zymbler, 2015). Afin d'accélérer les calculs, plusieurs modifications de l'algorithme DTW sont proposées : contraindre le calcul du *Warping Path* selon une forme donnée (Geler et al., 2019 ; Sakoe & Chiba, 1978), faire varier graduellement la résolution des données lors des calculs (Choi et al., 2020), découper le *Warping Path* en plusieurs parties (Jambhale & Khaparde, 2014).

D'autres chercheurs ont également cherché à améliorer la qualité de l'algorithme : en ajoutant un poids sur les distances pour éviter les distorsions et les valeurs aberrantes (Jeong et al., 2011), en modifiant la méthode de calcul de distance entre chaque point (Muscillo et al., 2007) et en ajoutant artificiellement de nouveaux points via une interpolation pour un meilleur calcul de coût (Munich & Perona, 1999).

(Kyrarini & Graeser, 2017) propose un pré-traitement des données via 3 grandes étapes : une division de chaque démonstration en sous-parties, un alignement temporel des sous-parties avec l'algorithme de *Ramer-Douglas-Peucker* (RDP) appelé également *Douglas-Peucker* (Douglas & Peucker, 1973 ; Ramer, 1972) et une sélection des sous-parties à l'aide d'un calcul de coût utilisant la distance Manhattan. L'alignement temporel avec RDP est comparable à celui à l'aide de DTW pour un temps de calcul beaucoup plus faible.

Le pré-traitement des données avant l'utilisation d'algorithmes d'apprentissage par démonstration n'est pas beaucoup étudié dans la littérature. Généralement, DTW est utilisé pour l'alignement temporel des démonstrations qui sont ensuite utilisées pour l'apprentissage.
La qualité des données étant très importante pour l'apprentissage, il convient de déterminer le pré-traitement le plus adapté. Cet aspect sera abordé plus en détail dans le Chapitre 2 où sera présenté un comparatif détaillé des principales méthodes de pré-traitement. L'ensemble des techniques présentées précédemment seront mises en œuvre et évaluées pour sélectionner le pré-traitement le plus adapté. Celui-ci sera employé par l'algorithme d'apprentissage.

1.2 Algorithmes d'apprentissage par démonstration

Une fois les données traitées, le robot peut apprendre à l'aide d'algorithmes d'apprentissage par démonstration. Pourquoi ne pas apprendre à l'aide d'une seule démonstration d'une tâche ? Tout simplement car il n'est pas aisé pour un opérateur humain de réaliser directement le mouvement parfaitement. La généralisation, et donc l'apprentissage à partir de plusieurs démonstrations, extrait et lisse le mouvement souhaité en éliminant le bruit présent dans les données.

L'apprentissage peut s'effectuer à deux niveaux (Calinon, 2009 ; Chernova & Thomaz, 2014) : l'apprentissage haut niveau (symbolique) et l'apprentissage bas niveau (trajectoire).

L'apprentissage haut niveau consiste à extraire les différentes actions d'une tâche et à en comprendre la logique à travers une représentation symbolique. Le mouvement est décomposé en une séquence d'actions prédéfinies régies par des règles. Ce niveau d'apprentissage nécessite une base de données d'actions basiques ainsi que des données supplémentaires telles que la position et l'orientation des objets ou l'état de l'environnement par exemple.

L'apprentissage bas niveau est une représentation générique du mouvement, au niveau de la trajectoire. Ce niveau ne permet pas l'adaptation d'une trajectoire selon une nouvelle contrainte (nouvelle position d'objet, obstacles supplémentaires…) ni d'apprendre des mouvements très complexes. L'apprentissage bas niveau ne nécessite que les données des capteurs des articulations du robot contrairement à l'apprentissage haut niveau qui nécessite des capteurs supplémentaires comme une caméra par exemple.

Ces deux niveaux d'apprentissage peuvent être associés afin d'avoir un apprentissage complet (Kyrarini, 2019). Dans la suite des travaux, l'apprentissage par démonstration sera effectué au niveau de la trajectoire ; il est aussi appelé apprentissage bas niveau.

Une des méthodes largement utilisée pour encoder un mouvement est *Dynamic Movement Primitives* (DMP) (Ijspeert et al., 2002, 2013 ; Schaal, 2006 ; Schaal et al., 2007). Ce modèle convertit une démonstration en un jeu d'équations différentielles non-linéaires. Ces équations sont robustes aux perturbations externes, garantissent une convergence de l'algorithme d'apprentissage et permettent une correction dynamique. Ainsi, DMP peut s'adapter facilement à une nouvelle position d'arrivée en modifiant simplement le paramètre de but.

Un modèle de *DMP modifié* (Pastor et al., 2009) ajoute la possibilité de modifier la position de départ en plus de la position d'arrivée et l'évitement d'obstacle dans les équations différentielles apprises. *Mixture of Motor Primitives* (MoMP) (Mülling et al., 2013) associe plusieurs DMP pour généraliser un mouvement. Chaque démonstration est encodée par un DMP afin d'apprendre au robot comment jouer au tennis de table.

Une autre variante aux primitives de mouvement est *Stable Estimator of Dynamic Systems* (SEDS) (Khansari-Zadeh & Billard, 2011). SEDS encode statistiquement le mouvement à l'aide d'équations différentielles non-linéaires du premier ordre en s'inspirant de *Gaussian Mixture Model* (GMM). SEDS garantit la stabilité asymptotique globale du système dynamique.

Hidden Markov Model (HMM) est une machine à états probabiliste finie qui divise la trajectoire considérée en états. Chaque état est modélisé par une distribution de gaussiennes qui permettent de prendre en compte les contraintes de temps et d'espace. La génération d'une trajectoire à partir du modèle appris peut se faire à l'aide d'interpolation ou de spline (suite de courbes polynomiales du troisième degré) entre des points clés du mouvement (Aleotti & Caselli, 2005 ; Asfour et al., 2008 ; Billard et al., 2006 ; Brand & Hertzmann, 2000 ; Calinon et al., 2005 ; Calinon & Billard, 2004).

Un autre moyen de générer une trajectoire est d'utiliser *Gaussian Mixture Regression* (GMR) qui opère une régression sur les gaussiennes apprises (Calinon, D'halluin, et al., 2010 ; Calinon, Sauser, et al., 2010).

Task-Parameterized Hidden Semi-Markov Model (TP-HSMM) utilise un processus semi-Markovien couplé à un modèle de tâches paramétrées afin d'adapter le mouvement à diverses situations environnementales (Pignat & Calinon, 2017). Dans ce cas, les démonstrations devront représenter un large panel de conditions afin que le modèle de mouvement soit suffisamment générique.

La dernière principale catégorie de méthode d'apprentissage par démonstration est *Gaussian Mixture Model* (GMM). GMM encode la trajectoire des démonstrations à l'aide de distributions de gaussiennes pondérées. Le poids donné à chaque gaussienne créé une représentation du mouvement grâce à la combinaison d'un nombre fini de gaussiennes. L'encodage de ce modèle extrait les contraintes de temps et d'espace des démonstrations. *Gaussian Mixture Regression* (GMR) génère ensuite la trajectoire apprise (Calinon et al., 2007 ; Calinon & Billard, 2008b).

Task-Parameterized GMM (TP-GMM) ajoute un modèle de tâches paramétrées à GMM pour une adaptation du mouvement dans de nouvelles conditions (Calinon, 2016, 2018 ; Rozo et al., 2016).

GMM modification (m-GMM) ajoute une étape supplémentaire au processus GMM/GMR afin de modifier les moyennes du modèle GMM appris pour adapter le mouvement à de nouvelles positions d'objet ou éviter des obstacles (Kyrarini, Haseeb, et al., 2019 ; Kyrarini, Zheng, et al., 2019). Cette méthode ne nécessite pas de démonstrations dans diverses conditions contrairement au TP-GMM.

Le Tableau 1 répertorie les principaux algorithmes d'apprentissage par démonstration présentés précédemment. HMM ne représente que les points clés contrairement au GMM qui encode le mouvement de manière continue. Il a également été montré que GMM permettait d'encoder de manière automatique les contraintes d'une trajectoire (Calinon & Billard, 2007a). La génération à l'aide de GMR crée une trajectoire plus lisse.

Tableau 1 : Comparaison des algorithmes d'apprentissage par démonstration

Algorithmes d'apprentissage	Démons-trations		Adaptation à l'environnement			Références
	Unique	Multiple	Position de départ	Position d' arrivée	Obstacles	
DMP	X			X		(Ijspeert et al., 2002, 2013 ; Schaal, 2006 ; Schaal et al., 2007)
DMP modifié	X		X	X	X	(Pastor et al., 2009)
MoMP		X	X	X		(Mülling et al., 2013)
SEDS		X	X	X		(Khansari-Zadeh & Billard, 2011)
HMM/génération de trajectoire (ajustement par interpolation ou spline)		X	X	X		(Aleotti & Caselli, 2005 ; Asfour et al., 2008 ; Billard et al., 2006 ; Brand & Hertzmann, 2000 ; Calinon et al., 2005 ; Calinon & Billard, 2004)
HMM/GMR		X		X		(Calinon, D'halluin, et al., 2010 ; Calinon, Sauser, et al., 2010)
TP-HSMM/GMR		X	X	X		(Pignat & Calinon, 2017)
GMM/GMR		X				(Calinon et al., 2007 ; Calinon & Billard, 2008b)
GMM/TP-GMM+GMR		X	X	X	X	(Calinon, 2016, 2018 ; Rozo et al., 2016)
GMM/m-GMM+GMR	X	X	X	X	X	(Kyrarini, Haseeb, et al., 2019 ; Kyrarini, Zheng, et al., 2019)

Les travaux de (Kyrarini, 2019) sont donc les plus prometteurs dans le cadre d'un apprentissage par démonstration avec un nombre limité de démonstration et ne nécessitant pas de conditions environnementales différentes dans les démonstrations pour pouvoir s'adapter. Ces travaux présentent cependant des limites. Le framework développé utilise la technologie Robot Operating System (ROS) (Quigley et al., 2009). Ceci crée une dépendance vis-à-vis de cette technologie et limite donc les robots qui peuvent être utilisés (2 dans ses travaux). En effet, l'intégration de ROS lorsqu'elle n'est pas native est particulièrement compliquée et même impossible pour certains robots. De plus, il n'est pas évident de déterminer si une interface utilisateur est employée lors de son expérience avec un robot industriel. Enfin l'expérience sur l'apprentissage robotique de tâches d'assemblage industriel a eu lieu dans un laboratoire, avec des participants du laboratoire. L'opérateur humain devait être loin du robot lors de son mouvement et l'interaction simultanée avec les 2 bras du robot simultanément n'était pas possible.

Dans le Chapitre 3, une architecture générique va être présentée afin de permettre à n'importe quel robot, avec la technologie ROS ou non, d'être intégré. L'apprentissage à l'aide des 2 bras du robot sera également implémenté. Une version améliorée de GMM sera proposée afin d'améliorer l'apprentissage en fonction des préférences de l'opérateur. Dans le Chapitre 4, une expérience à l'aide de potentiels futurs utilisateurs des cobots sera proposée afin de valider ou non l'acceptabilité de la programmation par démonstration dans l'optique d'une industrie 4.0.

Conclusion

L'état de l'art des technologies de pré-traitement et des algorithmes d'apprentissage par démonstration met en évidence les atouts et les inconvénients de chacune des méthodes existantes. L'apprentissage par démonstration est une bonne méthode pour programmer un cobot par des utilisateurs non-experts. Le pré-traitement des données étant indispensable, une comparaison de méthodes utilisant des technologies existantes dans des situations diverses relèvera le pré-traitement le plus adapté pour l'apprentissage par démonstration (Chapitre 2). L'apprentissage par démonstration sera ensuite mis en place dans le Chapitre 3 en se basant sur le modèle *Gaussian Mixture Model* (GMM) et *Gaussian Mixture Regression* (GMR). Le Chapitre 4 déterminera si la programmation par démonstration est acceptée par les futurs utilisateurs des cobots. L'organisation des prochains chapitres est récapitulée par la Figure 7.

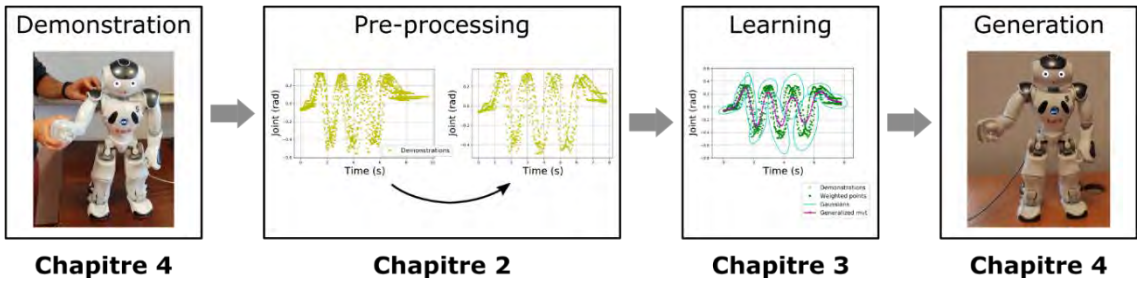


Figure 7 : Organisation du manuscrit

Chapitre 2

Pré-traitement des données pour l'apprentissage par démonstration

« L'observation recueille les faits ; la réflexion les combine ; l'expérience vérifie le résultat de la combinaison. »

Denis Diderot[4]

Sommaire

[4] Œuvres complètes de Denis Diderot Tome II, 1821, p152

Une bonne qualité des données pour l'apprentissage par démonstration est indispensable. Comme à l'école, si l'enseignement et les livres scolaires ne sont pas de bonne qualité ou présentent des données erronées, l'apprentissage sera dégradé. Ainsi, les données des démonstrations doivent montrer un mouvement similaire via un bon alignement temporel et ne pas comporter des valeurs aberrantes via la sélection des démonstrations. Ces deux étapes de pré-traitement doivent filtrer et améliorer la qualité des données. Les critères de choix des techniques ne sont pas clairement étudiés dans la littérature. C'est pourquoi, diverses méthodes vont être testées et comparées pour déterminer le pré-traitement le plus efficace à utiliser dans le cas d'un apprentissage par démonstration (Figure 8). Dans un premier temps, les méthodes choisies seront présentées dans la Section 2.1. Dans un second temps, ces méthodes seront évaluées selon diverses tâches, avec plusieurs robots et des données de qualité différente (Section 2.2).

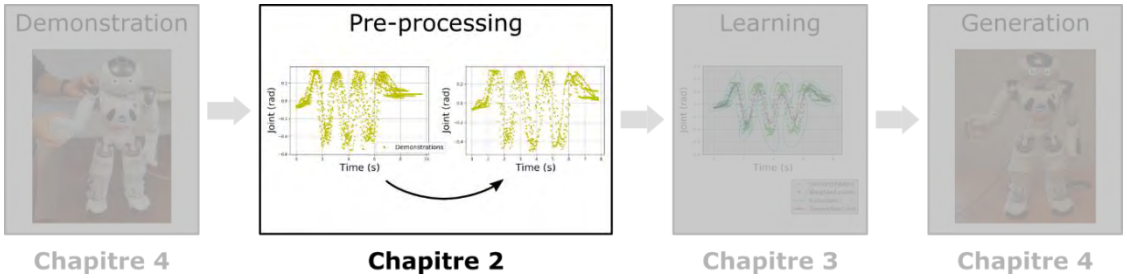


Figure 8 : Organisation du manuscrit - Chapitre 2

2.1 Méthodes de pré-traitement des données

2.1.1 Choix des techniques

Dans le cadre de l'apprentissage par démonstration, le pré-traitement des données est composé de deux étapes importantes : l'alignement temporel et la sélection des démonstrations. L'alignement temporel, étape indispensable pour l'apprentissage, transforme les démonstrations afin qu'elles correspondent temporellement entre elles.

La sélection des démonstrations supprime les démonstrations qui contiennent des valeurs aberrantes ou trop différentes des autres. La qualité des démonstrations est ainsi affinée, l'apprentissage s'en trouve donc amélioré.

Les techniques présentes dans la littérature (Section 1.1) sont soit spécialisées dans une étape du pré-traitement (alignement temporel ou sélection des démonstrations), soit le pré-traitement en entier. Afin de déterminer quel pré-traitement fonctionne le mieux pour l'apprentissage par démonstration, divers algorithmes sont comparés. Plusieurs techniques sont choisies pour chaque étape. Ces techniques sont sélectionnées en fonction de leur utilisation fréquente dans la littérature ainsi que de leur potentiel.

L'alignement temporel sera réalisé selon deux méthodes : soit à l'aide d'une variation N de l'algorithme de *Douglas-Peucker* (DP-N) (Elmar de Koning, 2011) soit de manière proportionnelle.

L'algorithme DP-N aligne toutes les démonstrations sur la plus petite en nombre de points. Ainsi, chaque démonstration aura le même nombre de points.

L'alignement proportionnel consiste simplement à utiliser une règle de trois, ce qui permet de conserver la distance entre les points. L'alignement peut se faire sur n'importe quelle démonstration. Si on veut aligner les données d'une série temporelle B sur une série temporelle de référence A, il suffit de multiplier chaque point t_B de la série B par le temps total T_A de la série A puis diviser par le temps total T_B de la série B, comme le montre l'Équation (1).

$$t_{new\ B} = t_B \times \frac{T_A}{T_B} \tag{1}$$

La sélection des démonstrations sera effectuée selon deux techniques. La première technique est appelée algorithme de sélection Point-à-Point et la seconde, appelé algorithme de sélection DTW, est basée sur l'algorithme DTW (Müller, 2007).

L'algorithme Point-à-Point ne peut être utilisé que si l'alignement est fait avec l'algorithme DP-N. Cette technique consiste à sélectionner les démonstrations à l'aide d'une classification des scores d'erreur par rapport à une démonstration de référence. Dans l'algorithme Point-à-Point, chaque démonstration obtient un score d'erreur, calculé à l'aide de la somme des différences absolues entre chaque point consécutif de la démonstration référence. La classification des scores d'erreur utilise l'algorithme K-means avec 1 000 itérations. L'algorithme classe les scores selon deux clusters. Les démonstrations conservées sont celles qui sont dans le même cluster que la démonstration référence. Si la référence est la seule démonstration dans le cluster, les démonstrations sont comparées à un seuil et seules celles étant en dessous de celui-ci seront sélectionnées.

L'algorithme de sélection DTW est similaire au précédent. La seule différence est dans le calcul des scores d'erreur. A la place du score d'erreur, DTW permet d'obtenir un score final à l'aide du *Warping Path* optimal. Cela assure le meilleur alignement entre la référence et chaque démonstration. La classification et ensuite la sélection des démonstrations se déroulent de la même manière que précédemment. Un seuil est également utilisé afin de sélectionner les démonstrations lorsque la référence est la seule démonstration dans le cluster.

2.1.2 Définition des méthodes

Le pré-traitement des données comporte l'alignement temporel et la sélection des démonstrations (Figure 9). L'association de ces deux étapes sera appelée méthode.

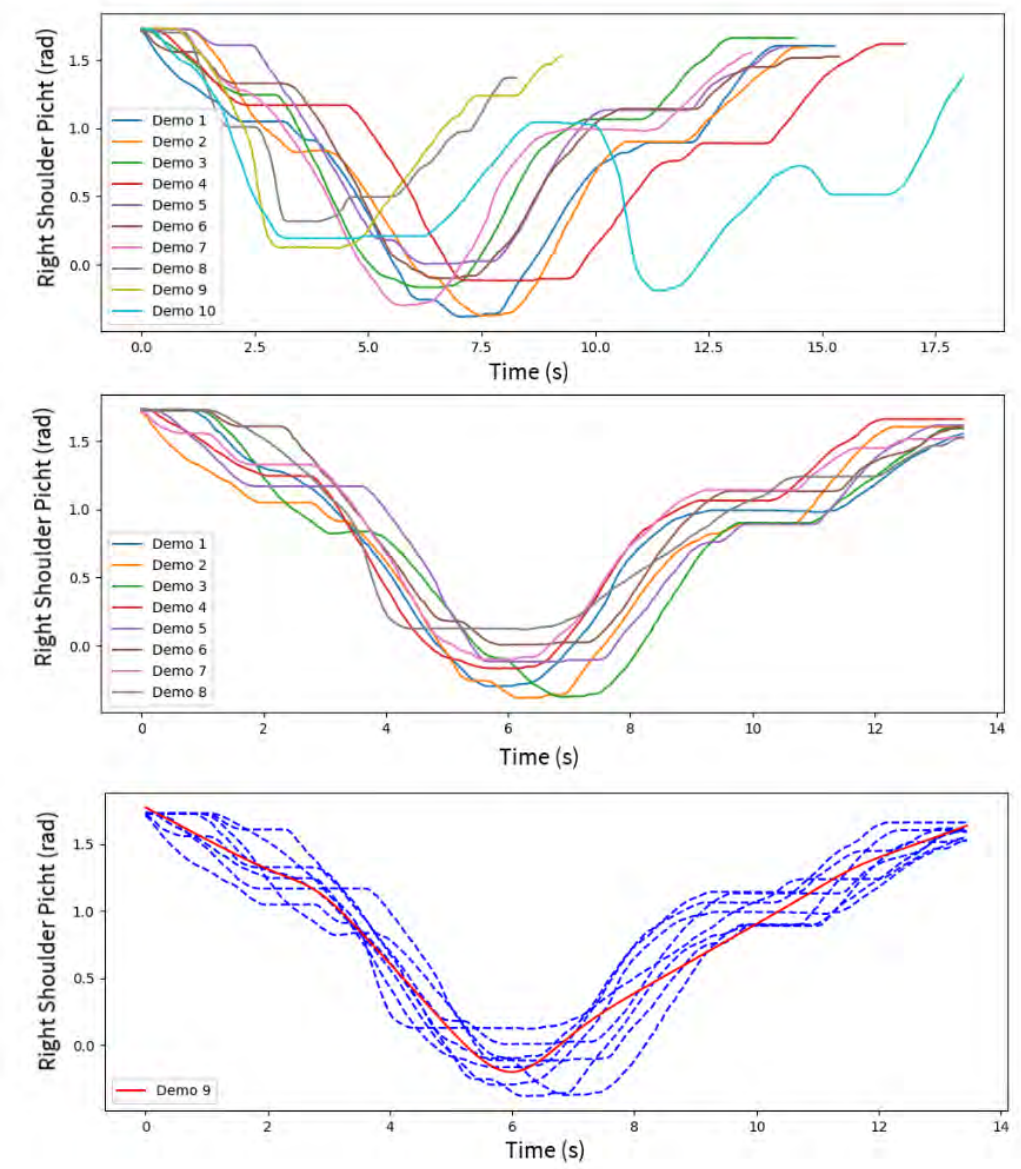


Figure 9 : Étapes de pré-traitement

De haut en bas : démonstrations originales, démonstrations alignées et sélectionnées, mouvement appris (en rouge) après la phase d'apprentissage avec les démonstrations alignées et sélectionnées (en pointillées bleu)

À l'aide des techniques sélectionnées précédemment, six combinaisons, ou méthodes, sont définies (Tableau 2).

Tableau 2 : Description des méthodes

Alignement temporel (en vert) : alignement DP-N (Al. DP-N), alignement proportionnel (Al. Prop). Sélection des démonstrations (en bleu) : sélection Point-à-Point (Sel. Ptp), sélection DTW (Sel. DTW). Sub-sec = algorithme de sous-sections de démonstrations (en orange)

Méthodes					
Alignement seulement		Démonstrations entières		Sous-section de démonstrations	
MA1	**MA2**	**M1**	**M2**	**M3**	**M4**
Al. DP-N	Al. Prop	Al. DP-N	Al. Prop	Sub-sec.	Sub-sec.
-	-	Sel. Ptp	Sel. DTW	Al. DP-N	Al. Prop
-	-	-	-	Sel. Ptp	Sel. DTW

Les méthodes MA1 et MA2, méthodes qui ne réalisent que l'alignement temporel, serviront de base pour comparer les autres méthodes. En effet, la phase d'apprentissage a besoin au minimum d'un alignement temporel pour fonctionner correctement.

Quatre autres méthodes M1, M2, M3 et M4 sont basées sur des combinaisons d'algorithmes d'alignement et de sélection présentés précédemment. Parmi ces quatre méthodes, deux catégories se distinguent. La première va impacter l'entièreté des démonstrations. La seconde va agir sur des portions de démonstrations, définies par des marqueurs temporels fournis par l'utilisateur.

La méthode M1 combine l'algorithme d'alignement DP-N et l'algorithme de sélection Point-à-Point. La première étape de cette méthode consiste à aligner temporellement les démonstrations avec l'algorithme DP-N. La démonstration référence est ensuite sélectionnée. La dernière étape réalise la sélection des démonstrations.

La méthode M2 utilise l'algorithme d'alignement proportionnel et l'algorithme de sélection DTW. La référence est d'abord sélectionnée. L'alignement entre la référence et les démonstrations est ensuite effectué. Enfin, les bonnes démonstrations sont sélectionnées.

Les méthodes M3 et M4 sont similaires aux méthodes M1 et M2 respectivement avec une étape de sous-division en plus. La sous-division utilisée divise les démonstrations selon des points temporels importants, pouvant représentés des étapes clés du mouvement. L'alignement temporel est réalisé sur les sous-divisions définies par ces points clés à la place de l'ensemble de la démonstration. La sélection des démonstrations est également faite sur les sous-sections des démonstrations. Ces sous-sections permettent la suppression d'une partie d'une démonstration plutôt que la suppression de l'entièreté d'une démonstration.

Le choix des combinaisons d'algorithmes pour la définition des méthodes est dépendant des caractéristiques et contraintes des algorithmes. L'algorithme d'alignement DP-N aligne les démonstrations sur celles disposant du plus faible nombre de points. L'algorithme d'alignement proportionnel peut aligner les démonstrations sur n'importe quelle démonstration sélectionnée comme étant la référence. L'algorithme de sélection Point-à-Point est utilisé avec l'algorithme d'alignement DP-N car il requière une fréquence d'échantillonnage constante et le même nombre de points pour chaque démonstration afin de pouvoir les comparer. L'utilisation des méthodes avec les sous-divisions signifie que des points importants sont connus. Si ce n'est pas le cas, ces méthodes ne seront pas appropriées et l'utilisation d'une des méthodes avec la totalité des démonstrations sera un meilleur choix.

2.2 Évaluation des méthodes

2.2.1 Protocole expérimental

Pour évaluer les méthodes, une expérience avec plusieurs robots, plusieurs tâches et plusieurs qualités de données est mise en place. Trois robots sont utilisés : YuMi, Pepper et Nao (Figure 10).

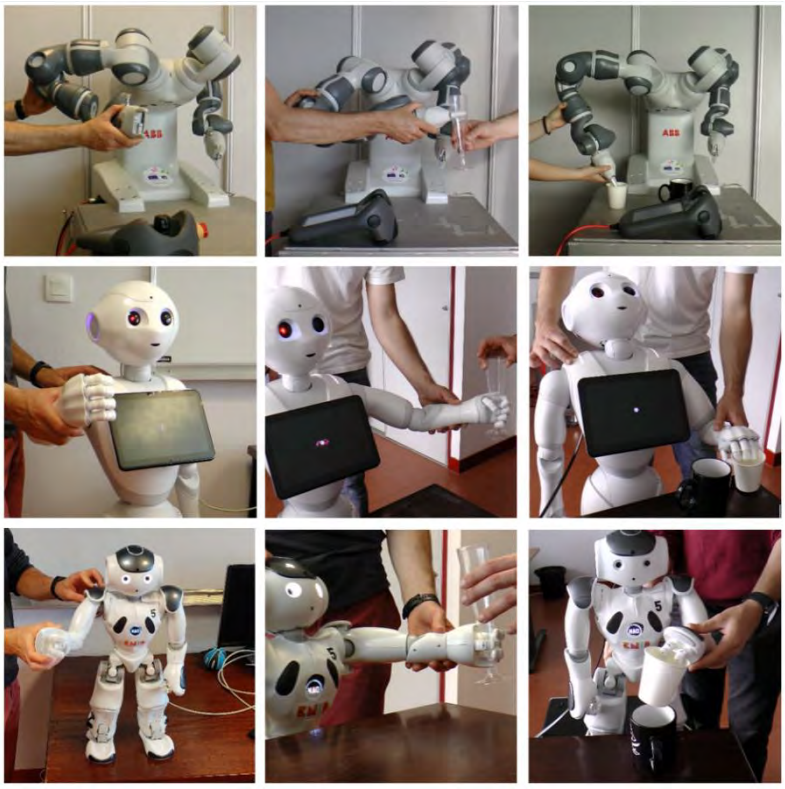


Figure 10 : Photos des 3 robots et des 3 tâches

En ligne, les robots (de haut en bas) : YuMi, Pepper et Nao. En colonne, les tâches (de gauche à droite) : Lettre D, Donner Verre, Gobelet dans Mug.

Les données collectées sur les robots sont les valeurs de leurs joints dans le temps. Ces valeurs constituent les démonstrations. Le robot YuMi dispose de sept joints, ou axes, et d'une pince par bras. Les robots Pepper et Nao ont cinq joints et une main par bras. Les tâches seront réalisées avec un seul bras pour chaque robot. La fréquence d'échantillonnage du robot YuMi est de 250 Hz et de 30 Hz pour les robots Pepper et Nao. Ainsi, comme le robot YuMi récupère plus de points, le temps de pré-traitement sera plus élevé que pour les autres robots. Aucun système de vision n'est utilisé pour la réalisation des tâches.

Chaque robot réalisera les trois tâches : « Lettre D » (LD pour *Letter D*), « Donner Verre » (GG pour *Give Glass*) et « Gobelet dans Mug » (GIM pour *Goblet Into Mug*) (Figure 10). La tâche « Lettre D » consiste à écrire la lettre D dans les airs. La tâche « Donner Verre » déplace un verre vers une personne. Un gobelet doit être placé dans un mug dans la tâche « Gobelet dans Mug ». Les tâches ont une difficulté croissante. Pour la tâche LD, l'écriture dans les airs est la tâche la plus simple car elle n'implique aucune contrainte matérielle ni spatiale. La tâche GG augmente la difficulté car le robot doit être précis pour la récupération du verre. La dernière tâche GIM est la plus complexe car elle inclut deux contraintes spatiales, contrairement à la tâche GG qui n'en comporte qu'une. Le robot doit être plus précis pour prendre le gobelet et le mettre dans le mug.

Pour chaque tâche, dix-huit démonstrations sont faites pour chaque robot. Parmi ces dix-huit démonstrations, il y a dix bonnes démonstrations, cinq démonstrations normales et trois mauvaises démonstrations. Les bonnes démonstrations permettent d'effectuer parfaitement la tâche. Les démonstrations normales atteignent l'objectif de la tâche mais des mouvements supplémentaires inutiles sont ajoutés. Les mauvaises démonstrations ne remplissent pas du tout l'objectif de la tâche. Les conditions de l'environnement (position du robot, position des objets, de l'utilisateur) sont les mêmes pour chaque démonstration.

À l'aide de ces démonstrations, cinq jeux de données contenant dix démonstrations sont créés (Tableau 3). Ces jeux de données ont une qualité décroissante, contenant de plus en plus de mauvaises démonstrations.

Tableau 3 : Combinaison des démonstrations des 5 jeux de données

Chaque jeu de données est composé de 10 démonstrations. Nb démos = nombre de démonstrations

Jeu de données	Nb de bonnes démos	Nb de démos normales	Nb de mauvaises démos
D1	10	-	-
D2	8	2	-
D3	7	2	1
D4	6	2	2
D5	4	3	3

2.2.2 Mesures

La démonstration idéale remplit parfaitement l'objectif de la tâche avec la trajectoire la plus adaptée. Dans la pratique, aucune mesure ne permet de vérifier si la démonstration est idéale. Par la suite, le terme de démonstration idéale sera utilisé pour la démonstration sélectionnée subjectivement par l'utilisateur et représentant le meilleur mouvement pour réaliser une tâche donnée. La démonstration idéale sera utilisée ensuite pour mesurer la performance de l'apprentissage lié au pré-traitement employé.

La démonstration de référence, utilisée par les algorithmes de sélection des démonstrations, est déterminée à l'aide des scores d'erreur définis dans la Section 2.1.1. Ces scores d'erreur sont calculés pour chaque association de démonstrations. La démonstration récoltant le score d'erreur cumulé le plus faible, sera la référence. Cette référence sera utilisée par les algorithmes de sélections des démonstrations.

Afin de comparer les méthodes, quatre mesures sont choisies : le temps de calcul (CT ou *Computing Time*), l'erreur absolue moyenne finale (FMAE ou *Final Mean Absolue Error*), l'erreur quadrique moyenne finale (FMSE ou *Final Mean Squared Error*) et le taux de succès (SR ou *Success Rate*). Le temps de calcul déterminera la méthode la plus rapide. Avant chaque mesure, une normalisation sera effectuée sur chaque joint afin de garantir un impact identique sur le score d'erreur de tous les joints.

FMAE et FMSE se basent respectivement sur l'erreur absolue moyenne (MAE ou *Mean Absolute Error*) et l'erreur quadratique moyenne (MSE ou *Mean Squared Error*).
La MAE compare deux séries et détermine un score de similarité entre elles. Deux séries identiques ont une valeur de MAE de zéro. La MSE est similaire à la MAE ; la différence au carré remplace la différence absolue.
Les scores de la MAE et la MSE quantifient l'écart entre les deux séries. L'information croisée à partir de la MAE et la MSE fournit des indications sur les similarités entre deux séries et comment les erreurs y sont distribuées. Pour chaque joint, la MAE et la MSE sont calculées entre la démonstration idéale et la démonstration courante.

À partir de ces valeurs, une moyenne est calculée pour obtenir une seule valeur de MAE et de MSE pour chaque démonstration. Une seconde moyenne est réalisée à partir de toutes les moyennes de MAE et MSE afin d'obtenir une valeur finale (FMAE et FMSE respectivement) entre la démonstration idéale et toutes les autres démonstrations, comme le montre les Équations (2) et (3).

$$FMAE = \frac{1}{D}\sum_{d=1}^{D}\frac{1}{J}\sum_{j=1}^{J}\frac{1}{Pt}\sum_{p=1}^{Pt}|Pt_{demo} - Pt_{ideal}| \quad (2)$$

$$FMSE = \frac{1}{D}\sum_{d=1}^{D}\frac{1}{J}\sum_{j=1}^{J}\frac{1}{Pt}\sum_{p=1}^{Pt}(Pt_{ideal} - Pt_{demo})^2 \quad (3)$$

avec : Pt le nombre de point dans la démonstration, Pt_{demo} la valeur angulaire de la démonstration évaluée, Pt_{ideal} la valeur angulaire de la démonstration idéale, J le nombre de joints et D le nombre de démonstrations.

Le taux de succès est déterminé après la phase d'apprentissage composée des algorithmes *Gaussian Mixture Model* (GMM) et *Gaussian Mixture Regression* (GMR) (Calinon et al., 2007 ; Calinon & Billard, 2008b). Le taux de succès est une valeur comprise entre 0 et 100 ; 0 correspondant à un échec total ou une absence de mouvement et 100 un succès parfait de la tâche. La valeur du taux de succès s'évalue selon plusieurs critères (Tableau 4).

Tableau 4 : Critères d'évaluation du taux de succès

%	Critères d'évaluation	Types de critère
50	L'objectif de la tâche est atteint.	Réussite de l'objectif (50%)
25	Aucun mouvement n'a interféré avec la tâche.	Qualité de la réalisation (50%)
10	La tâche est réalisée sans mouvement inutile.	
10	Le robot a fait sa trajectoire en entier sans erreur.	
5	Le mouvement est fluide.	

Le SR est divisé en deux parts égales ; la première moitié se focalise sur la réussite de l'objectif de la tâche et la seconde moitié évalue la qualité de la réalisation de la tâche.

Au niveau des critères portant sur la qualité de la réalisation, les mouvements qui interfèrent avec l'objectif de la tâche sont notés sur 25 points car ils peuvent conduire à l'échec de la tâche. Un mouvement qui interfère avec la tâche peut être, dans le cas de la tâche GG, un mauvais maintien du verre dans l'effecteur du robot voire, dans le pire des cas, sa chute.

Les mouvements inutiles sont des mouvements supplémentaires à l'exécution de la tâche mais n'ayant aucun impact sur sa bonne réalisation. Si la trajectoire est arrêtée avant la fin car les données contiennent une position inatteignable, alors le critère lié à la présence d'erreur aura comme valeur 0. Si le robot fait toute la trajectoire sans ne rien accomplir de la tâche, alors le taux de succès est alors de 10.

La fluidité a un impact moins important dans le taux du succès car elle est moins critique pour la réalisation de la tâche. Cependant, si la tâche consiste à déplacer un contenant rempli de liquide, le critère de fluidité doit avoir une valeur plus élevée dans le calcul du taux de succès car il est majeur dans la bonne réalisation de la tâche.

Ces mesures fournissent des informations sur la qualité de la génération de mouvement selon la qualité du pré-traitement choisi.

2.2.3 Résultats

Les diverses méthodes MA1, MA2, M1, M2, M3 et M4 sont comparées. Afin d'évaluer au mieux les méthodes, quatre prismes de lecture sont retenus : une évaluation de manière générale, une deuxième en fonction des jeux de données, une troisième selon les robots et une dernière au regard des tâches.

2.2.3.1 Évaluation générale

Pour déterminer la méthode de pré-traitement la plus adaptée pour l'apprentissage par démonstration de manière générale, une moyenne est effectuée sur les résultats selon les jeux de données. La Figure 11 représente le temps de calcul des méthodes selon les robots.

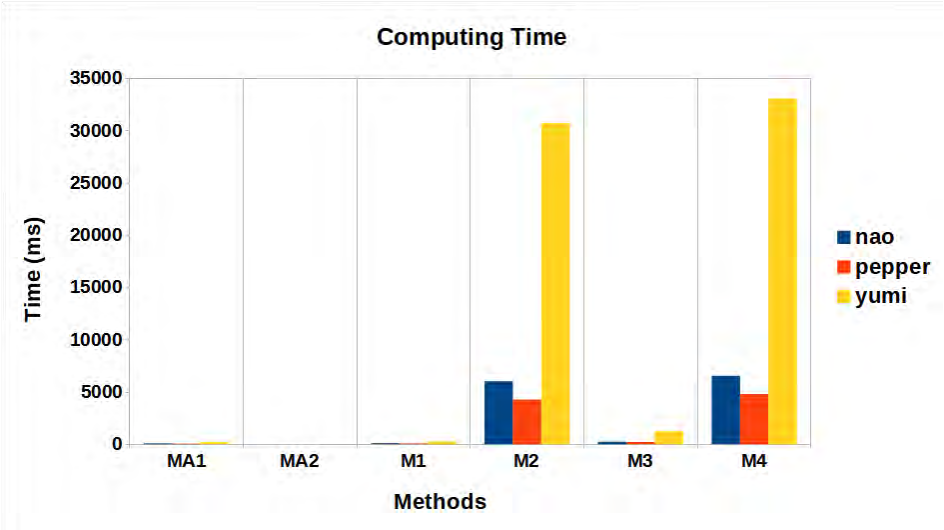


Figure 11 : Temps de calcul des méthodes

Les méthodes MA1 et MA2 ont un temps de calcul très faible, moins de 300 ms, car ces méthodes ne réalisent que l'alignement temporel. Le temps de calcul de la méthode MA2 est très faible comparé aux autres méthodes, environ 0,2 ms. La méthode M1 est plus rapide que la méthode M2 et il en est de même entre la méthode M3 et la méthode M4. Les méthodes utilisant les démonstrations entières sont plus rapides que leur équivalent utilisant les sous-divisions des démonstrations. Les méthodes utilisant l'algorithme de sélection DTW sont celles ayant le temps de calcul le plus long.

Toutes les démonstrations n'ont pas le même nombre de points. Même en déterminant le temps de calcul de chaque méthode pour cent points, les tendances restent les mêmes (Tableau 5).

Tableau 5 : Temps de calcul de chaque méthode pour cent points (en ms)

Méthodes	Nao	Pepper	YuMi
MA1	**21,37**	**19,50**	**30,11**
MA2	**0,08**	**0,10**	**0,07**
M1	**24,95**	**23,26**	**34,69**
M2	1555,41	1330,57	3943,00
M3	62,57	63,88	161,15
M4	1691,96	1494,79	4249,14

La méthode obtenant la plus faible valeur de FMAE est M1, suivie de près par M2 (Tableau 5).

Tableau 6 : FMAE pour chaque méthode

Méthodes	Nao			Pepper			YuMi		
	LD	GG	GIM	LD	GG	GIM	LD	GG	GIM
MA1	9,88e-2	4,56e-2	4,57e-2	4,94e-2	3,07e-2	3,86e-2	1,35e-1	3,56e-2	9,21e-2
MA2	9,50e-2	4,61e-2	4,61e-2	5,34e-2	3,27e-2	3,95e-2	1,41e-1	3,53e-2	9,08e-2
M1	**3,34e-2**	**2,05e-2**	**2,90e-2**	**2,61e-2**	**1,93e-2**	2,91e-2	**7,49e-2**	**1,41e-2**	**4,36e-2**
M2	4,58e-2	2,82e-2	2,98e-2	3,85e-2	2,09e-2	**2,67e-2**	9,93e-2	1,42e-2	4,73e-2
M3	1,27e-1	7,31e-2	6,39e-2	1,02e-1	4,91e-2	3,60e-2	1,48e-1	4,06e-2	1,10e-1
M4	4,58e-2	3,77e-2	2,98e-2	5,08e-2	2,09e-2	**2,67e-2**	9,93e-2	1,42e-2	4,73e-2

Pour les valeurs de FMSE, la méthode qui est généralement la meilleure est M1 et parfois M2 (Tableau 7).

Tableau 7 : FMSE pour chaque méthode

Méthodes	Nao			Pepper			YuMi		
	LD	GG	GIM	LD	GG	GIM	LD	GG	GIM
MA1	3,00e-2	3,01e-2	4,55e-3	1,45e-2	5,28e-3	7,99e-3	2,92e-1	1,22e-2	6,53e-2
MA2	3,01e-2	7,11e-3	9,10e-3	1,51e-2	5,83e-3	8,54e-3	3,21e-1	1,19e-2	6,44e-2
M1	**4,55e-3**	**1,43e-3**	4,17e-3	**3,22e-3**	2,19e-3	4,46e-3	**4,50e-2**	1,99e-3	**1,42e-2**
M2	9,11e-3	5,07e-3	**4,10e-3**	6,62e-3	**1,98e-3**	**4,06e-3**	1,06e-1	**1,85e-3**	1,46e-2
M3	1,14e-1	2,93e-2	2,57e-2	3,87e-2	7,02e-2	6,53e-3	1,25e-1	1,33e-2	5,90e-2
M4	9,11e-3	8,43e-3	**4,10e-3**	1,43e-2	**1,98e-3**	**4,06e-3**	1,06e-1	**1,85e-3**	1,46e-2

Les différences de FMAE et FMSE entre les méthodes M1, M2, M3 et M4 par rapport aux méthodes contrôles MA1 et MA2 représentent l'amélioration liée à la sélection des bonnes démonstrations (Tableau 8 et Tableau 9).

Tableau 8 : Différence de FMAE pour chaque méthode

Méthodes	Nao			Pepper			YuMi		
	LD	GG	GIM	LD	GG	GIM	LD	GG	GIM
MA1-M1	**6,54e-2**	**2,52e-2**	**1,66e-2**	**2,33e-2**	1,14e-2	9,49e-3	**6,02e-2**	**2,15e-2**	**4,85e-2**
MA2-M2	4,92e-2	1,78e-2	1,63e-2	1,49e-2	**1,18e-2**	**1,28e-2**	4,12e-2	2,11e-2	4,36e-2
MA1-M3	-2,79e-2	-2,75e-2	-1,82e-2	-5,24e-2	-1,84e-2	2,56e-3	-1,28e-2	-4,99e-3	-1,83e-2
MA2-M4	4,92e-2	8,37e-3	1,63e-2	2,66e-3	**1,18e-2**	**1,28e-2**	4,12e-2	2,11e-2	4,36e-2

Tableau 9 : Différence de FMSE pour chaque méthode

Méthodes	Nao			Pepper			YuMi		
	LD	GG	GIM	LD	GG	GIM	LD	GG	GIM
MA1-M1	**2,54e-2**	**2,87e-2**	0,38e-3	**1,13e-2**	3,09e-3	3,51e-3	**2,47e-1**	**1,03e-2**	**5,11e-2**
MA2-M2	2,10e-2	2,04e-3	**5,00e-3**	8,52e-3	**3,85e-3**	**4,49e-3**	2,15e-1	1,00e-2	4,98e-2
MA1-M3	-8,40e-2	0,84e-3	-2,11e-2	-2,42e-2	-6,49e-2	1,46e-3	1,67e-1	-1,02e-3	6,23e-3
MA2-M4	2,10e-2	-1,32e-3	**5,00e-3**	0,86e-3	**3,85e-3**	**4,49e-3**	2,15e-1	1,00e-2	4,98e-2

Les différences de FMAE et de FMSE entre la méthode M1 et MA1 obtiennent généralement les meilleurs résultats, suivies par celles entre les méthodes M2 et MA2 et celles entre les méthodes M4 et MA2.

Le Tableau 10 répertorie les taux de succès de chaque méthode. Selon le taux de succès, l'ordre des méthodes allant de la meilleure à la moins bonne est : M2, MA2, M4, M1, M3, MA1.

Tableau 10 : Taux de succès de chaque méthode

Méthodes	Nao			Pepper			YuMi			Moyenne générale	Rang
	LD	GG	GIM	LD	GG	GIM	LD	GG	GIM		
MA1	64,6	46,8	40,4	68,2	78,8	43,6	77,2	73,2	48,0	60,09	6
MA2	**98,0**	65,2	40,4	77,6	**88,6**	43,6	88,4	82,4	**82,4**	74,07	2
M1	70,6	56,8	56,4	75,0	83,8	52,4	**93,4**	59,8	53,0	66,80	4
M2	90,4	**68,2**	**79,6**	93,6	86,0	**58,4**	92,6	**99,2**	30,0	77,56	1
M3	73,4	54,8	72,8	79,0	86,8	40,2	73,8	57,8	39,6	62,24	5
M4	88,0	56,0	63,0	**96,4**	87,4	52,6	72,6	59,2	59,6	70,53	3

De manière générale, la méthode M2 est la meilleure car elle obtient le meilleur compromis entre l'obtention d'une bonne valeur de FMAE et FMSE et un bon taux de succès. La méthode M2 nécessite un temps de calcul élevé mais qui reste acceptable dans le cadre d'un apprentissage par démonstration.

2.2.3.2 Évaluation en fonction du jeu de données

Afin de déterminer la meilleure méthode selon les jeux de données, les résultats sont moyennés selon les robots et les tâches. Plus les jeux de données sont de mauvaise qualité, plus le temps de calcul diminue. Cela s'explique probablement par le fait que les mauvaises démonstrations sont plus courtes temporellement que les bonnes.

La Figure 12 représente les valeurs de FMAE et FMSE de chaque méthode selon les jeux de données. Les valeurs de FMAE et FMSE croissent en fonction de la dégradation des jeux de données. La méthode M1 obtient les meilleures valeurs suivies par les méthodes M2 et M4.

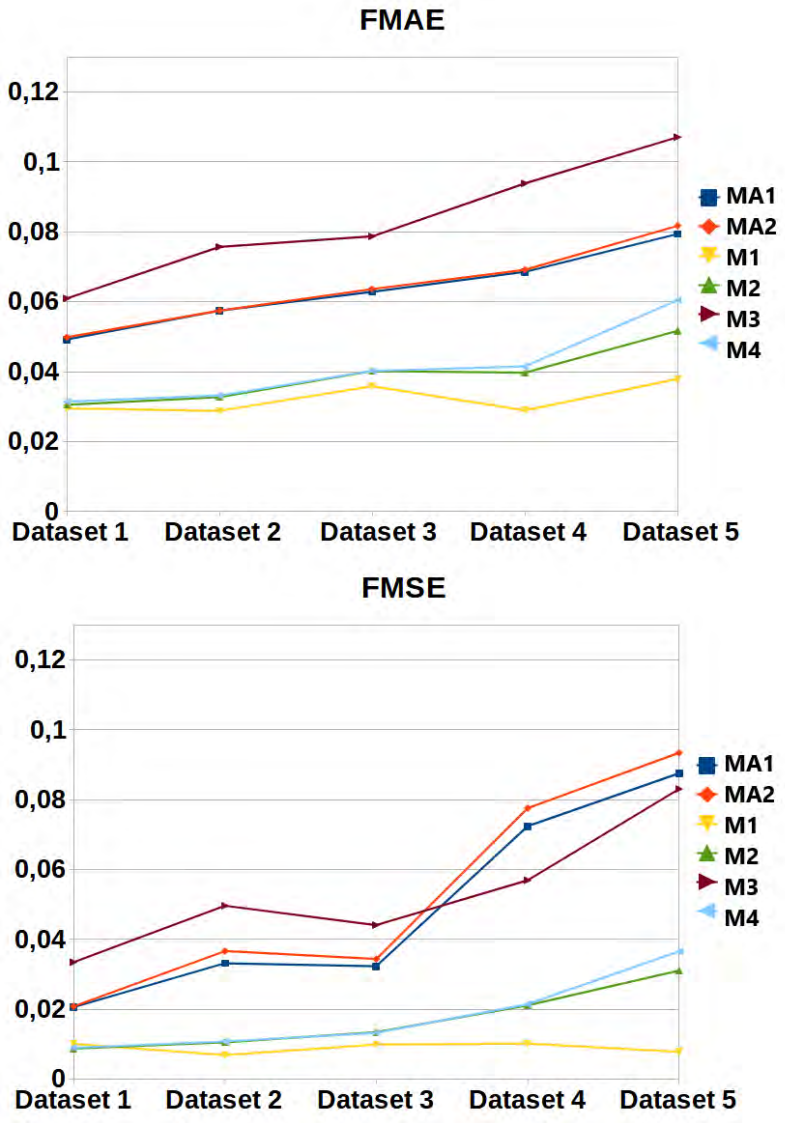


Figure 12 : FMAE et FMSE pour chaque méthode selon les jeux de données

Au niveau du taux de succès, la meilleure méthode est M2 et parfois MA2 (Tableau 11).

Tableau 11 : Taux de succès pour chaque méthode selon les jeux de données

Méthodes	Jeux de données				
	D1	D2	D3	D4	D5
MA1	78,7	78,9	62,9	45,3	34,7
MA2	**90,0**	**88,3**	75,1	67,2	49,7
M1	75,9	70,3	73,4	58,3	56,0
M2	84,3	82,8	**89,6**	**67,9**	**63,2**
M3	89,7	84,9	62,4	48,3	35,9
M4	72,4	86,0	77,9	67,8	48,6

2.2.3.3 Évaluation selon les robots

Les résultats selon les robots sont obtenus en moyennant les valeurs selon les tâches et les jeux de données. Au niveau du temps de calcul, les moins bons résultats sont obtenus avec le robot YuMi. Cela est dû à sa fréquence d'échantillonnage plus élevée que celle des robots Nao et Pepper, ce qui augmente le nombre de points à traiter. Le temps de calcul est légèrement plus rapide avec le robot Pepper qu'avec le robot Nao (Figure 11 et Tableau 5).

La Figure 13 contient les résultats des calculs de FMAE et FMSE. La valeur de FMAE est la meilleure avec le robot Pepper puis Nao et enfin YuMi, sauf pour les méthodes M3 et M4. Les méthodes M3 et M4 sont celles qui procèdent aux sous-divisions, ce qui peut expliquer les différences aux niveaux des résultats des valeurs de FMAE. Le robot YuMi est celui qui comporte le plus d'erreurs, probablement dû à son nombre supérieur de données à traiter. Les valeurs de FMSE sont mauvaises avec les méthodes MA1 et MA2 et grandement améliorées avec les méthodes M1, M2, M3 et M4. Parmi les méthodes M1 à M4, la méthode M3 est la moins bonne.

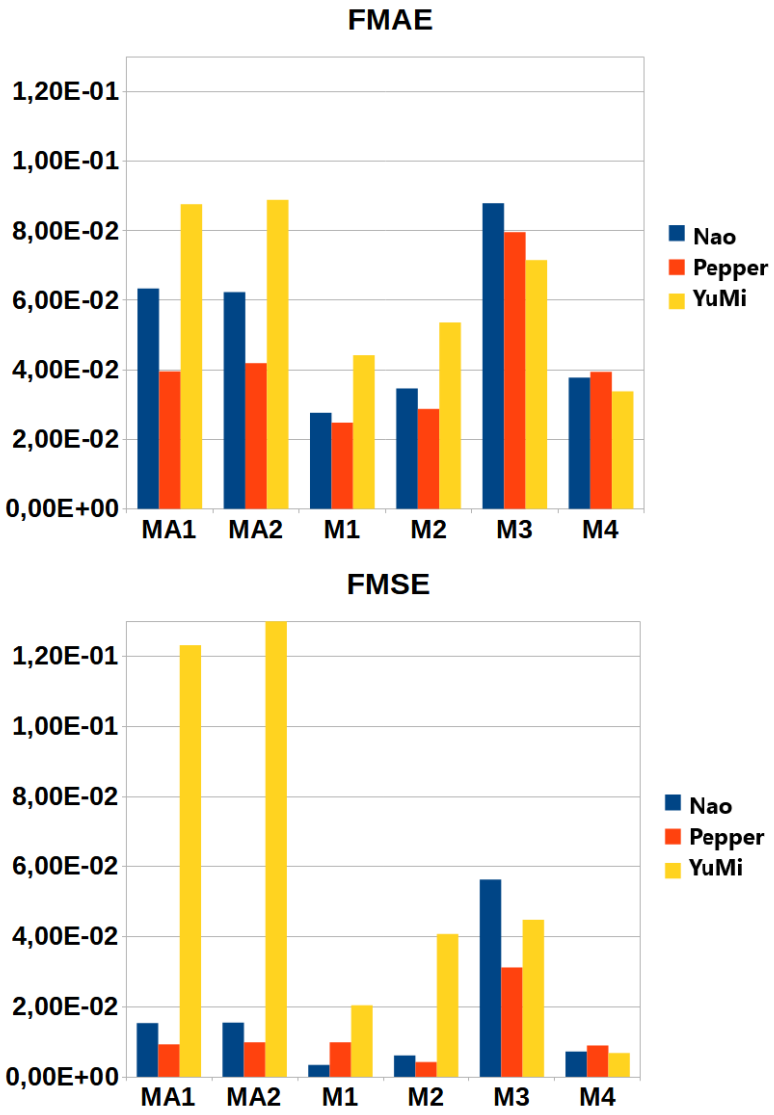


Figure 13 : FMAE et FMSE de chaque méthode selon les robots

Le taux de succès est le meilleur avec la méthode M2, sauf pour le robot YuMi qui est meilleur avec la méthode MA2 (Tableau 12).

Tableau 12 : Taux de succès de chaque méthode selon les robots

Méthodes	Robots		
	Nao	Pepper	YuMi
MA1	50,6	63,5	66,1
MA2	67,9	69,9	**84,4**
M1	61,3	70,4	68,7
M2	**79,4**	**79,3**	73,9
M3	67,0	68,7	57,1
M4	69,0	78,8	63,8

2.2.3.4 Évaluation au regard des tâches

Les méthodes sont également comparées selon les tâches en moyennant les résultats en fonction des jeux de données et des robots. Au niveau du temps de calcul, les résultats ne sont pas pertinents à analyser car ils dépendent de la durée des tâches.

Les valeurs de FMAE sont les meilleures avec la tâche GG suivies par la tâche GIM et enfin LD (Figure 14). Cela est surprenant car la difficulté de la tâche LD est plus faible que celles des autres tâches. Une possible raison réside dans le fait que la tâche LD n'a pas de contrainte spatiale et donc le mouvement qui consiste à écrire la lettre D dans l'espace peut être très différent d'une démonstration à une autre, même pour des bonnes démonstrations. Ces résultats sont corrélés avec les valeurs de FMSE qui montre des valeurs très élevées pour la tâche LD (Figure 14). La méthode M1 obtient de manière générale les meilleurs résultats au niveau FMAE et FMSE suivi de près par la méthode M2.

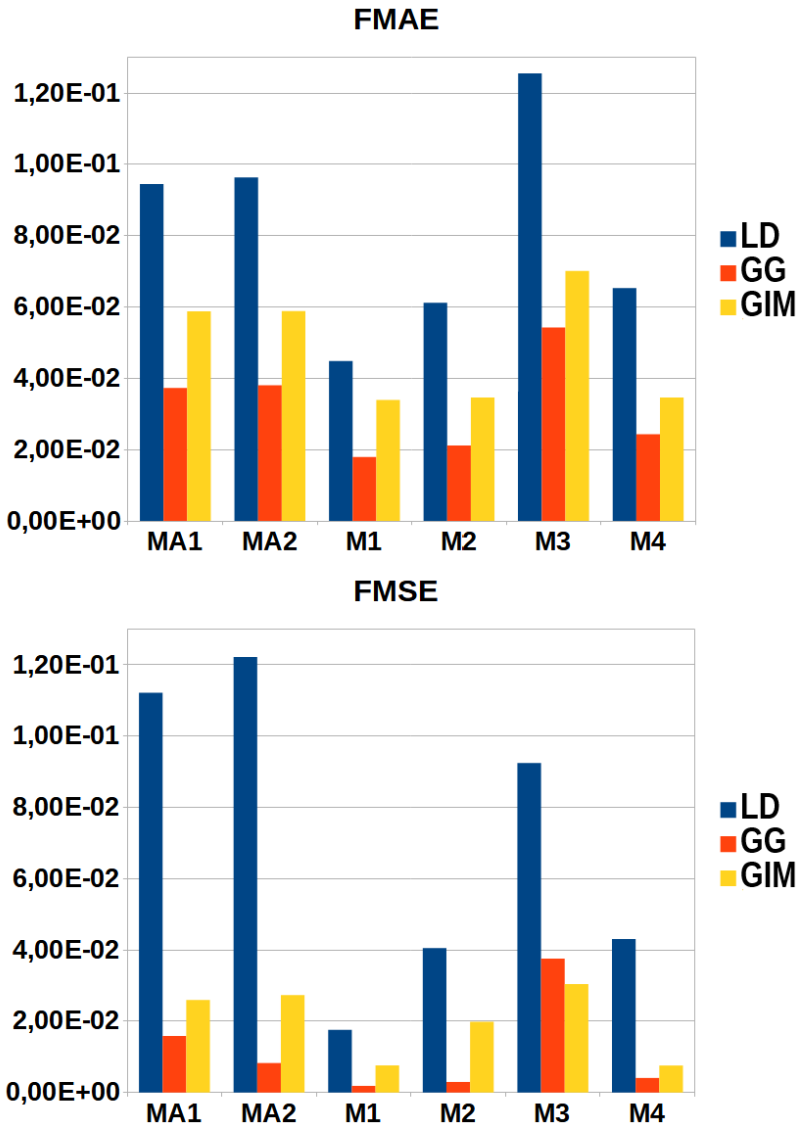


Figure 14 : FMAE et FMSE de chaque méthode selon les tâches

Au niveau du taux de succès, la meilleure méthode est M2 (Tableau 13).

Tableau 13 : Taux de succès de chaque méthode selon les tâches

Méthodes	Tâches		
	LD	GG	GIM
MA1	70,0	66,3	44,0
MA2	88,0	78,7	55,5
M1	79,7	66,8	53,9
M2	**92,2**	**84,5**	56,0
M3	75,4	66,5	50,9
M4	85,7	67,5	**58,4**

D'une manière générale, les niveaux élevés des erreurs de la tâche LD sont très probablement dus à l'aspect non-physique du mouvement. De plus, les tâches GG et GIM incluent des objets, là où la tâche LD est simplement basée sur un aspect visuel, qui est davantage enclin à des erreurs et à une évaluation plus subjective.

2.2.4 Discussion

Comme le montre les différents résultats, il y a deux méthodes qui surpassent les autres : M1 et M2. D'un côté, la méthode M1 réduit les erreurs (Tableau 6, Tableau 7) et obtient la plus faible erreur après la sélection des démonstrations (Tableau 8, Tableau 9). D'un autre côté, la méthode M2 présente les meilleurs résultats après la phase d'apprentissage (Tableau 10).

D'autres paramètres sont également à prendre en compte. La méthode M1, basée sur l'algorithme d'alignement DP-N, réduit le nombre de points en sélectionnant les meilleurs. Ceci peut expliquer le faible nombre d'erreurs mais aussi la détérioration de la forme du mouvement. À contrario, la méthode M2 ne supprime aucun point, ce qui a pour effet d'améliorer le taux de succès. La méthode M2 conserve la forme générale du mouvement. C'est pourquoi la génération de mouvement après l'apprentissage est meilleure.

Concernant les méthodes avec les sous-divisions de démonstrations, la méthode M4 est correcte mais pas optimale. La méthode M3 est celle qui présente les plus mauvais résultats de toutes les méthodes testées. Ces méthodes pourraient être améliorées si les points importants renseignés étaient plus précis selon la forme du mouvement.

Par conséquence, la méthode M2 est le meilleur compromis entre les valeurs de FMAE/FMSE et le taux de succès avec un temps de calcul raisonnable.

Conclusion

Afin de sélectionner le pré-traitement qui sera implémenté lors du développement de l'apprentissage par démonstration, plusieurs méthodes sont confrontées sur trois robots, trois tâches et cinq jeux de données. La méthode M2, qui utilise l'algorithme d'alignement proportionnel et l'alignement de sélection DTW, obtient les meilleurs résultats après la phase d'apprentissage. Que ce soient selon les robots, selon les tâches ou selon les jeux de données, et même de manière plus générale, la méthode M2 s'est révélée la meilleure pour l'apprentissage par démonstration. Par la suite, la méthode M2 est retenue pour le pré-traitement.

Chapitre 3

Apprentissage par démonstration générique avec des données pondérées

« […] les charmes enchanteurs de cette sublime science [la science des mathématiques] ne se décèlent dans toute leur beauté qu'à ceux qui ont le courage de l'approfondir. »

Carl Friedrich Gauss[5]

Sommaire

[5] Correspondance Germain-Gauss, lettre 7 rédigée en français, 1807

L'apprentissage par démonstration donne la possibilité à un robot d'apprendre à partir d'exemples donnés par une personne (Figure 15). De manière similaire, l'enseignement dispensé à l'école doit être suffisamment générique pour que n'importe quel élève puisse comprendre et apprendre. Si l'élève n'a pas correctement appris, l'enseignant fournit de nouvelles explications ou informations pour aider l'élève à mieux comprendre les nouvelles connaissances qui lui sont transmises. L'apprentissage par démonstration devrait pouvoir servir n'importe quel robot, peu importe sa forme ou sa programmation (Section 3.1). Un autre aspect important dans l'apprentissage est la possibilité d'orienter les connaissances selon le souhait de l'enseignant. Pour ce faire, les données des démonstrations vont être pondérées en fonction de l'importance que l'opérateur humain y apporte (Section 3.2).

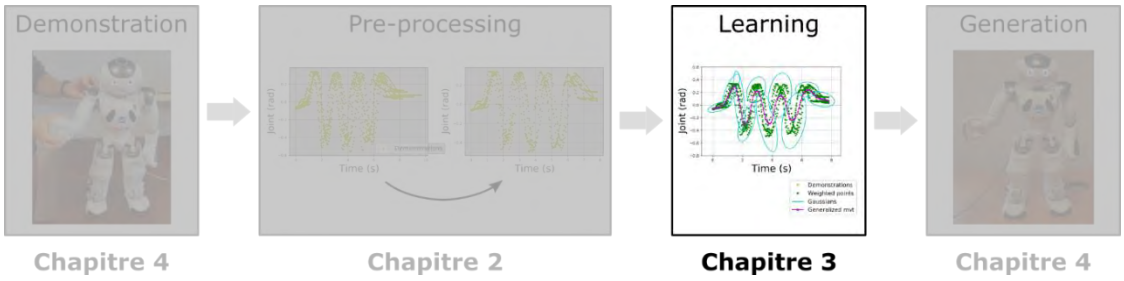


Figure 15 : Organisation du manuscrit - Chapitre 3

3.1 Généricité de l'apprentissage

3.1.1 Besoin de généricité ?

Le terme généricité est utilisé par la suite pour définir le fait que l'apprentissage peut s'appliquer sur des plateformes robotiques très variées. La généricité de l'apprentissage est un atout pour une réutilisation du programme dans un autre contexte, ici sur un autre robot.

Le programme doit donc autoriser l'ajout d'un robot quels que soient son nombre d'articulations, son langage de programmation (langage de programmation standard, langage propriétaire, technologie ROS, etc.) ou ses types de données (mode joint, mode cartésien, autre(s) donnée(s) supplémentaire(s)).

La généricité est malheureusement peu développée ou limitée dans un certain cadre (ex : avec ROS), ce qui crée une forme de dépendance du programme. Un programme générique favorise donc la possibilité de réemployer la technologie plutôt que de lui imposer une nouvelle implémentation.

3.1.2 Architecture générique et ajout d'un robot

L'architecture logicielle générique proposée se base sur l'héritage de classes comme le montre le diagramme de classes de la Figure 16. Plus la classe est haut dans l'héritage, plus elle est générique.

La classe Robot_base est utilisée uniquement par l'interface utilisateur afin de contrôler l'apprentissage du robot. La classe Robot est la classe principale qui définit tous les robots. Les robots n'utilisant pas la technologie ROS (Quigley et al., 2009) doivent hériter de cette classe. La classe Robot_ros regroupe les robots utilisant ROS. Elle se divise en deux sous-classes qui dépendent de la commande de mouvement : le robot peut être contrôlé soit à l'aide d'une action ROS (classe Robot_ros_action), soit avec un topic ROS (classe Robot_ros_topic). Chaque nouveau robot hérite donc d'une des classes suivantes : Robot, Robot_ros_action ou Robot_ros_topic. La plupart du temps, les robots simulés héritent de leur classe robot réel.

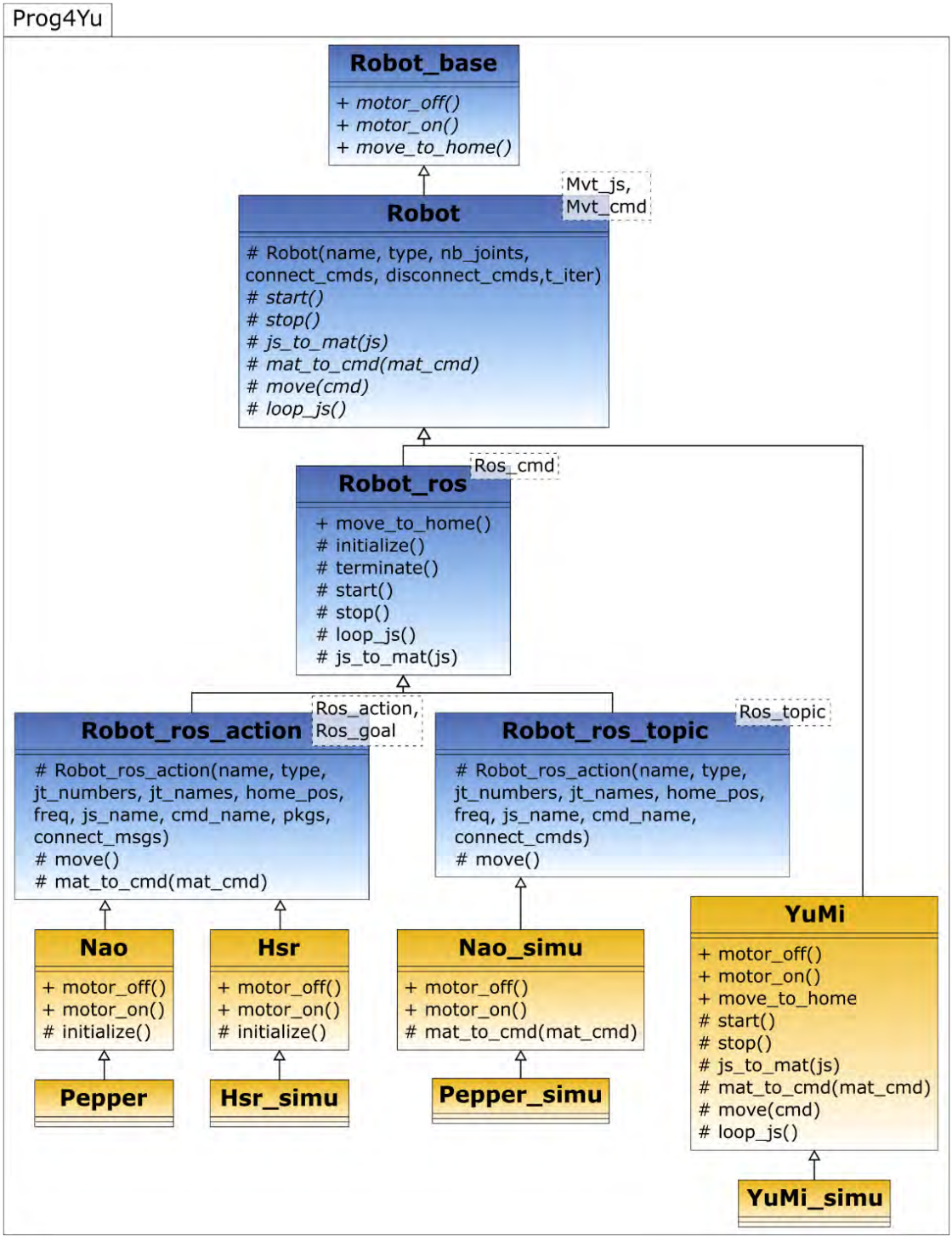


Figure 16 : Diagramme de classes de la généricité de l'apprentissage

Les robots ajoutés dans cette architecture sont les robots Nao, Pepper, YuMi et HSR (Figure 17).

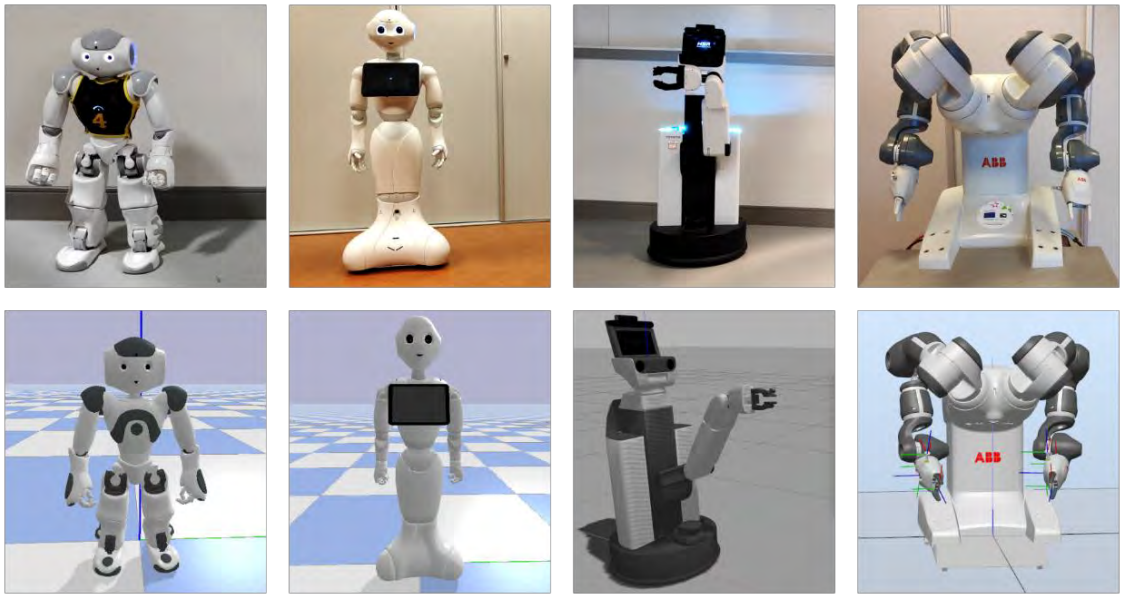


Figure 17 : Robots ajoutés et leur simulation

Liste des robots de gauche à droite : Nao, Pepper, HSR, YuMi. Première ligne : robots réels. Deuxième ligne : robots simulés (liste des logiciels de gauche à droite : qiBullet, qiBullet, Gazebo, RobotStudio).

Chaque nouveau robot est défini principalement par son nom et son type (réel ou simulé), sa manière de démarrer et s'arrêter, sa commande pour désactiver et activer ses moteurs, sa façon d'envoyer les données du robot et de le faire bouger.

Lors de la création du robot, le nombre de données possibles doit être défini. Il s'agit du nombre d'articulations dans le cas du mode joint, de la position et de l'orientation dans le cas du mode cartésien. Pour un robot disposant de deux bras ou plus, le nombre de données possibles associées à un bras doit être cumulé avec celui du ou des autres bras ; une seule valeur doit être donnée pour chaque robot. L'apprentissage s'adaptera à ce nombre.

Si le robot utilise ROS, l'ajout du robot sera alors simplifié ; moins de fonctions seront à instancier (voir Section 3.1.3). Dans le cas contraire, il peut y avoir également un lien entre les fonctions et le langage du robot, s'il est différent du langage C++, à créer. En effet, tous les robots ont un ou plusieurs langages de programmation, généralement pas toujours standard. Par exemple, le robot YuMi ne se contrôle qu'avec le langage propriétaire RAPID (voir Section 3.1.4).

L'architecture générique présentée donne la possibilité d'ajouter n'importe quel robot en utilisant l'apprentissage par démonstration. Cependant, pour pouvoir ajouter un robot, il faut être en mesure d'instancier un certain nombre de fonctions (voir Section 3.1.3 et Section 3.1.4). Le principal frein à l'ajout d'un robot est la possibilité d'avoir un mode passif des moteurs du robot. Dans le cas des robots Nao et Pepper, les moteurs sont simplement désactivés, c'est-à-dire à l'aide d'une absence de raideur, et l'utilisateur peut manipuler les bras du robot (Figure 6, page 15). L'absence de raideur fait tomber les bras vers le bas avec la gravité ; ils peuvent ensuite être manipulés librement.

Le robot YuMi dispose d'une fonction « Lead-Through » qui permet de manipuler le robot avec une compensation de la gravité. Ainsi les bras du robot peuvent facilement se manipuler, sans avoir besoin de maintenir toujours le robot en position contrairement aux robots Nao et Pepper.

Si le robot ne dispose d'aucune de ces fonctionnalités, il est possible d'utiliser un contrôleur pour faire les démonstrations, comme une manette. Cette option, utilisée avec le robot HSR (Figure 6b, page 15), nécessite un développement supplémentaire pour faire le lien entre le contrôleur et la commande du robot. Un contrôleur est aussi plus difficile à utiliser et moins intuitif que les démonstrations kinesthésiques.

Les robots simulés ne servent qu'à visualiser les mouvements appris. En effet, il est compliqué de contrôler un robot dans un espace à trois dimensions à l'aide d'une interface graphique ou d'une souris (deux dimensions) pour faire une démonstration, même si l'architecture générique le permettrait. Les démonstrations sont donc toujours réalisées sur des robots réels.

3.1.3 Exemples de robots contrôlables avec ROS : Nao, Pepper et HSR

Les robots contrôlables avec ROS sont les plus simples à ajouter. En effet, la technologie ROS (Quigley et al., 2009) standardise la communication avec le robot et le format des données. Le simulateur Gazebo est également commun à tous les robots utilisant ROS (voir Figure 17). Les robots Nao et Pepper ne disposent pas d'un modèle virtuel officiel du robot et les modèles non officiels existants ne sont pas complets et entièrement fonctionnels. Ces robots ont donc été simulés à l'aide du logiciel qiBullet (Figure 17) (Busy & Caniot, 2020).
Le robot HSR nécessite le package ROS officiel « tmc » (Toyota Motor Corporation, 2021) et les robots Nao et Pepper utilisent les packages ROS « naoqi_driver » (Knese, 2015), « naoqi_pose » (Hornung & Lemaignan, 2015) et « nao_apps » (Rabaud, 2015) (packages non officiels).

Le constructeur de classe des robots utilisant ROS dispose des arguments suivants (voir le diagramme de classes de la Figure 16) :

- le nom du robot et son type (réel ou simulé),
- le nombre et le nom de ses articulations (ou données en mode cartésien),
- les valeurs correspondant à la position de départ,
- la fréquence pour récupérer et envoyer les données,
- le nom du topic pour récupérer la position courante,
- le nom du topic ou de l'action pour envoyer la commande de position,
- la liste des packages à lancer au démarrage du robot et les messages de connexion qui

permettent de vérifier le bon démarrage des packages ou des commandes à lancer dans un terminal.

Tous les robots utilisant ROS doivent surcharger les 2 fonctions suivantes (Figure 16) :

- **fonction moteur on/off** (*motor_on()* et *motor_off()*) : la fonction « moteur on » doit permettre de mettre le robot dans un état où il peut recevoir une commande de position. La fonction « moteur off » met le robot dans un état où l'opérateur humain peut réaliser des démonstrations, appelé mode passif. Les moteurs doivent être dans un mode passif pour permettre leur manipulation et la commande de la pince/main du robot.

Si besoin, les robots utilisant ROS peuvent également surcharger les fonctions suivantes (Figure 16) :

- **fonction conversion matrice-commande** (*mat_to_cmd()*): la librairie Armadillo (Sanderson & Curtin, 2016, 2018) est utilisée pour effectuer les calculs pour l'apprentissage par démonstration (Section 3.2). Le format matriciel de cette librairie a ensuite besoin d'être converti en commande pour faire bouger le robot (format topic ou action ROS).
- **fonction d'initialisation et de fermeture** (*initialize()* et *terminate()*) : la classe Robot_ros gère automatiquement le lancement et la fermeture de ROS. Si les robots ont besoin de commandes particulières lors de leur démarrage et à la fin de leur utilisation, ces deux fonctions peuvent être surchargées.

Le mode passif pour les robots Nao et Pepper est obtenu en désactivant la raideur de leurs articulations dans leurs bras et leur tête. Les démonstrations peuvent donc utiliser les 2 bras et la tête des robots. Ces parties des robots sont choisies car elles ne perturbent pas la position globale du robot. Si la partie basse du robot était utilisée pour les démonstrations, la stabilité de la position debout du robot ne serait pas garantie.

Pour fermer la main des robots, une commande vocale « Gauche » ou « Droite » est associée respectivement à une commande pour la fermeture/ouverture de la main. Si la main du robot est fermée alors il l'ouvrira et inversement avec la même commande vocale.
Pour le robot HSR, il n'est pas possible de mettre le bras dans un mode passif. Le bras et la pince du robot étaient donc contrôlés à l'aide d'une manette lors des démonstrations (Figure 6, page 3).

Le mode joint a été utilisé avec les robots Nao, Pepper et HSR. Le mode cartésien est testé avec le package ROS « track_ik » (Beeson & Ames, 2015) pour le robot Nao. Ce mode n'est pas retenu car les solutions données par le solveur cinématique du package étaient très faibles. En effet, pour atteindre une position dans l'espace, le solveur peut obtenir soit plusieurs solutions, soit une seule solution, soit aucune. Le robot Nao ne dispose que de cinq degrés de liberté par bras et son amplitude de mouvement est faible, chaque axe étant assez limité. Dans un espace réel à six dimensions, le solveur ne trouvait presque aucune position pour une trajectoire donnée. Les points de la trajectoire définie sont proches les uns des autres. Le déplacement à réaliser est donc fin et nécessite de fait d'avoir un robot dont les mouvements peuvent être contrôlés très précisément, ce qui n'est pas le cas des robots cités précédemment.

3.1.4 Exemple d'un robot sans ROS : YuMi

Le robot YuMi ne dispose pas de packages ROS contenant toutes les fonctionnalités nécessaires dans notre contexte. Ce robot illustre donc l'ajout d'un robot ne disposant pas de package ROS. Le robot YuMi ne se contrôle qu'avec du code en langage propriétaire RAPID. Un lien entre les fonctions de l'architecture et ce langage a donc été réalisé afin d'utiliser ce robot de manière similaire aux autres. L'Annexe 1 décrit la création de ce lien avec l'aide des technologies External Guided Motion (EGM) et Robot Web service (RWS). Pour la simulation, le logiciel propriétaire RobotStudio est utilisé. Le robot simulé se contrôle de manière identique au robot réel, seul l'IP du robot change.

Le constructeur de la classe Robot nécessite les données suivantes (voir le diagramme de classes de la Figure 16) :

- le nom du robot et son type (réel ou simulé),
- le nombre de ses articulations (ou données en mode cartésien),
- les commandes à lancer dans un terminal pour connecter/déconnecter le robot,
- le temps d'intervalle entre deux données qui sera utilisé pour générer la trajectoire apprise.

En plus des fonctions « moteur on/off » vues précédemment, d'autres fonctions doivent être surchargées (Figure 16) :

- **fonction déplacement vers la position départ** (*move_to_home()*) : cette fonction doit bouger le robot vers la position de départ.
- **fonction start/stop** (*start()* et *stop()*) : permet de spécifier un démarrage/arrêt spécifique lors de la création/suppression du robot. Les fonctions d'initialisation et de fermeture décrites précédemment ne sont disponibles qu'avec les robots utilisant ROS car la classe Robot_ros gère automatiquement les fonctions start/stop.

- **fonction conversion position-matrice** (*js_to_mat()*) : les données de position (mode joint ou cartésien) doivent être converties en matrice de la librairie Armadillo (Sanderson & Curtin, 2016, 2018).
- **fonction déplacement selon commande** (*move()*) : le robot doit bouger selon la commande spécifiée.
- **fonction boucle position** (*loop_js()*) : une boucle, selon un booléen, doit récupérer la position du robot en continu.

Le mode passif du robot YuMi est disponible avec la fonctionnalité « Lead-Through » qui met le robot dans un état où l'utilisateur peut manipuler le robot simplement car le robot compense la gravité. L'utilisateur a la possibilité de bouger les 2 bras du robot. Par soucis de praticité, il est possible de manipuler le robot avec 2 personnes où chacune manipule un des bras du robot. Deux boutons de la tablette tactile du robot sont associés chacun à un bras. Chaque bouton déclenche alternativement la fermeture ou l'ouverture de la pince correspondante (pour plus de détails, voir Annexe 1).

3.2 Pondération des données

3.2.1 Ajout de l'humain dans la boucle de l'apprentissage

Comme mentionné dans le Chapitre 1, les algorithmes les plus prometteurs pour l'apprentissage par démonstration utilisent *Gaussian Mixture Model* (GMM) et *Gaussian Mixture Regression* (GMR) (Calinon et al., 2007 ; Calinon & Billard, 2008b).

Cependant, la trajectoire générée par ces algorithmes, même via l'évolution des algorithmes proposée par (Kyrarini, 2019), ne peut pas être améliorée une fois l'apprentissage effectué. En effet, l'apprentissage ne peut être modifié qu'en variant les données en entrées, c'est-à-dire en ajoutant ou en supprimant des démonstrations. Si toutes les démonstrations sont de bonne qualité et que la trajectoire générée par l'apprentissage n'est pas satisfaisante, il est impossible d'améliorer davantage l'apprentissage avec les algorithmes GMM/GMR et les travaux de (Kyrarini, 2019).

Nous proposons d'améliorer le modèle GMM/GMR en permettant l'ajout de poids aux données en fonction de leur importance. L'utilisateur peut ainsi améliorer l'apprentissage et donc la trajectoire générée sans avoir besoin de modifier les démonstrations.

Notre proposition consiste à remplacer l'algorithme GMM par une version de ce dernier compatible avec des données pondérées (Gebru et al., 2016), utilisée à l'origine pour faire de la classification de données audio-visuelles.

Afin de comprendre le fonctionnement de ces algorithmes, la Section 3.2.2 présente l'algorithme GMM, la Section 3.2.3 l'extension du modèle GMM avec des données pondérées et la Section 3.2.4 l'algorithme GMR. La Section 3.2.5 portera sur les 2 utilisations possibles du modèle proposé : le mode simple et le mode multiple.

3.2.2 Gaussian Mixture Model (GMM)

Chaque démonstration dispose de N_j données de dimension D. Un jeu de données ξ contenant J démonstrations est composé de N données avec $N = \sum_{j=1}^{J} N_j$ et $\xi = \{\xi_j\}_{j=1}^{N}$. Un mélange de gaussiennes a K distributions de gaussiennes. La fonction de densité de probabilité $P(\xi_j)$ du GMM (Calinon, 2009) est définie par l'Équation (4) :

$$P(\xi_j) = \sum_{k=1}^{K} P(k)\, P(\xi_j|k) \tag{4}$$

où les paramètres $P(k)$ et $P(\xi_j|k)$ sont définis par les Équations (5) et (6).

GMM se compose des paramètres $\{\pi_k, \mu_k, \Sigma_k\}_{k=1}^{K}$ où π_k est le vecteur de probabilités *a priori*, μ_k le vecteur de moyennes et Σ_k la matrice de covariance de la gaussienne k. Le vecteur de probabilités *a priori* π_k doit satisfaire deux conditions : $\pi_k \in [0, 1]$ et $\sum_{k=1}^{K} \pi_k = 1$.

$$P(k) = \pi_k \tag{5}$$

$$P(\xi_j|k) = \mathcal{N}(\xi_j; \mu_k, \Sigma_k)$$

$$= \frac{1}{\sqrt{(2\pi)^D det(\Sigma_k)}} e^{-\frac{1}{2}\left((\xi_j-\mu_k)^T \Sigma_k^{-1} (\xi_j-\mu_k)\right)} \tag{6}$$

Les paramètres d'un GMM sont appris à l'aide de l'algorithme Espérance-Maximisation (EM) (Dempster et al., 1977) à partir de valeurs approximatives obtenues avec l'algorithme K-means (MacQueen, 1967). À chaque pas u, l'étape E calcule :

$$p_{k,j}^{u} = \frac{\pi_k^u\, \mathcal{N}(\xi_j; \mu_k^u, \Sigma_k^u)}{\sum_{i=1}^{K} \pi_i^u\, \mathcal{N}(\xi_j; \mu_i^u, \Sigma_i^u)} \tag{7}$$

$$E_k^u = \sum_{j=1}^{N} p_{k,j}^u\, e^{-\frac{1}{2}\left((\xi_j-\mu_k)^T \Sigma_k^{-1} (\xi_j-\mu_k)\right)} \tag{8}$$

et l'étape M :

$$\pi_k^{u+1} = \frac{E_k^u}{N} \tag{9}$$

$$\mu_k^{u+1} = \frac{\sum_{j=1}^{N} p_{k,j}^u \, \xi_j}{E_k^u} \tag{10}$$

$$\Sigma_k^{u+1} = \frac{\sum_{j=1}^{N} p_{k,j}^u \left(\xi_j - \mu_k^{u+1}\right)\left(\xi_j - \mu_k^{u+1}\right)^T}{E_k^u} \tag{11}$$

jusqu'à ce que la convergence de la log-vraisemblance $\mathcal{L}(\xi_j)^u$ de l'Équation (12) augmente d'un seuil $C1$. L'algorithme EM est calculé pour chaque gaussienne k.

$$\mathcal{L}(\xi_j) = \sum_{j=1}^{N} \log(P(\xi_j)) \tag{12}$$

Le BIC score S_{BIC} (Schwarz, 1978) est utilisé pour définir le nombre optimal de gaussienne comme le montre l'Équation (13). Dans ce score, le premier membre de l'addition mesure à quel point le modèle suit les données. Le second membre est un facteur de pénalité afin de minimiser le nombre de paramètres du modèle. Plus le BIC score est faible, meilleur est le paramétrage de l'algorithme GMM.

$$S_{BIC} = -\mathcal{L}(\xi_j) + \frac{n_p}{2}\log(N) \tag{13}$$

avec : $n_p = (K-1) + K(D + (1/2)D(D+1)$

3.2.3 Gaussian Mixture Model avec des données pondérées (W-GMM)

L'algorithme GMM avec des données pondérées (W-GMM) (Gebru et al., 2016) est similaire à l'algorithme GMM avec un poids w_j sur chaque donnée. Un poids $w_j > 0$ représente l'importance d'un point de donnée ; comme si cette donnée était observée w_j fois. Si $w_j = 1$, la donnée a autant d'importance que si aucun poids ne lui était appliqué. Pour le modèle W-GMM, comme $\mathcal{N}\left(\xi_j; \mu_k, \Sigma_k\right)^{\boldsymbol{w_j}} \propto \mathcal{N}\left(\xi_j; \mu_k, \Sigma_k/\boldsymbol{w_j}\right)$, l'Équation (6) devient :

$$P\left(\xi_j|k\right) = \mathcal{N}\left(\xi_j; \mu_k, \Sigma_k/\boldsymbol{w_j}\right)$$

$$= \frac{1}{\sqrt{(2\pi)^D det\left(\Sigma_k/\boldsymbol{w_j}\right)}} e^{-\frac{1}{2}\left((\xi_j-\mu_k)^T\left(\frac{\Sigma_k}{\boldsymbol{w_j}}\right)^{-1}(\xi_j-\mu_k)\right)} \quad (14)$$

En conséquence, l'algorithme EM reçoit également quelques modifications (pour plus de détails, voir Annexe 3). Les Équations (7), (10) et (11) deviennent respectivement les Équations (15), (16) et (17).

$$p_{k,j}^u = \frac{\pi_k^u \, \mathcal{N}\left(\xi_j; \mu_k^u, \Sigma_k^u/\boldsymbol{w_j}\right)}{\sum_{i=1}^K \pi_i^u \, \mathcal{N}\left(\xi_j; \mu_i^u, \Sigma_i^u/\boldsymbol{w_j}\right)} \quad (15)$$

$$\mu_k^{u+1} = \frac{\sum_{j=1}^N \boldsymbol{w_j} \, p_{k,j}^u \, \xi_j}{\sum_{j=1}^N \boldsymbol{w_j} \, p_{k,j}^u} \quad (16)$$

$$\Sigma_k^{u+1} = \frac{\sum_{j=1}^N \boldsymbol{w_j} \, p_{k,j}^u \left(\xi_j - \mu_k^{u+1}\right)\left(\xi_j - \mu_k^{u+1}\right)^T}{E_k^u} \quad (17)$$

L'algorithme W-GMM n'a aucun impact sur la formulation des équations de l'algorithme GMR. Seuls les paramètres appris par GMM sont impactés par les poids de chaque donnée.

3.2.4 Gaussian Mixture Regression (GMR)

Les paramètres appris de l'algorithme GMM (ou W-GMM) μ_k et Σ_k sont utilisés par GMR (Calinon, 2009) afin de générer une trajectoire. Un jeu de données $\xi = \{\xi^I, \xi^O\}$ est composé d'un vecteur position ξ^O et d'un vecteur pas de temps ξ^I. L'Équation (18) décrit les composantes de la moyenne μ_k et de la covariance Σ_k.

$$\mu_k = \begin{bmatrix} \mu_k^I \\ \mu_k^O \end{bmatrix}, \qquad \Sigma_k = \begin{bmatrix} \Sigma_k^I & \Sigma_k^{IO} \\ \Sigma_k^{OI} & \Sigma_k^O \end{bmatrix} \tag{18}$$

La trajectoire $\hat{\xi}$ est générée à l'aide de l'algorithme GMR, comme le montre l'Équation (19).

$$\hat{\xi} = \sum_{k=1}^{K} h_k \, \hat{\xi}_k \tag{19}$$

avec : $\begin{cases} h_k = \frac{\pi_k \, \mathcal{N}(\xi^I; \mu_k^I, \Sigma_k^I)}{\sum_{i=1}^{K} \pi_i \, \mathcal{N}(\xi^I; \mu_i^I, \Sigma_i^I)} \\ \hat{\xi}_k = \mu_k^O + \Sigma_k^{OI} (\Sigma_k^I)^{-1} (\xi^I - \mu_k^I) \end{cases}$.

3.2.5 Mode simple et mode multiple

Les données provenant des démonstrations peuvent être en mode joint ou en mode cartésien. Les robots Nao, Pepper, HSR et YuMi permettent tous l'utilisation du mode joint. Ainsi, ce sera celui-ci qui sera utilisé par la suite.

Avec l'utilisation des algorithmes GMM et GMR, il est commun de donner toutes les données en entrée des algorithmes. Le nombre de gaussiennes est donc le même pour chaque dimension des données. En effet, dans le cas du mode joint, tous les joints obtiennent un nombre de gaussiennes identique. Cette utilisation des algorithmes sera appelée le **mode multiple**, tous les joints sont utilisés pour l'apprentissage.

Fournir tous les joints n'est pas forcément le plus efficace. Un joint peut nécessiter un nombre inférieur de gaussiennes par rapport à un autre joint, s'il n'est pas beaucoup utilisé dans le mouvement par exemple. Dans ce cas, l'utilisation d'un nouveau GMM pour chaque joint pourrait être plus efficace. L'utilisation de l'algorithme GMM, ou W-GMM, de manière individuelle pour chaque joint sera appelé **mode simple**. Ces deux modes (simple et multiple) sont comparés dans la Section 3.3 afin de déterminer si l'un des deux modes est plus efficace avec l'utilisation du modèle proposé : W-GMM et GMR.

3.3 Évaluation du modèle

3.3.1 Protocole expérimental

Le modèle proposé, composée de l'association du W-GMM et du GMR, a pu être testé avec différents robots grâce à l'architecture générique mise en place. Cette dernière standardise les démonstrations afin que l'algorithme d'apprentissage puisse traiter des démonstrations indépendamment du robot duquel sont issues les données d'entrée.
Ainsi, le modèle proposé est testé lors d'une expérience avec plusieurs robots et plusieurs tâches à réaliser afin de montrer diverses possibilités d'utilisation du modèle. Au cours de l'expérience, les valeurs angulaires des joints des robots sont utilisées pour créer les démonstrations.

Avant l'apprentissage, les données nécessitent d'être alignées temporellement et d'être filtrées. Quand une personne montre un mouvement à un robot, il est impossible de reproduire la même trajectoire à la même vitesse donc un alignement temporel est indispensable. L'opérateur humain peut également commettre des erreurs ou ajouter des mouvements parasites, souvent involontairement. Toutes ces erreurs doivent également être supprimées. Sans ces deux étapes de pré-traitement, le résultat de l'apprentissage sera impacté puisqu'il est directement lié à la qualité des démonstrations.

Le pré-traitement utilisé correspond à la méthode M2 du Chapitre 2 qui à l'issue de la comparaison des pré-traitements était le plus performant. L'alignement temporel de la méthode M2 est effectué de manière proportionnelle. Seul ce type d'alignement permet de gérer les possibles variations du pas de temps d'échantillonnage ainsi que les potentiels défaut d'acquisition de données. La filtration, ou sélection des démonstrations, de la méthode M2 utilise l'algorithme DTW.

Dans l'expérience, trois algorithmes sont testés :

- Référence : **GMM/GMR en mode multiple** (GG-mult),
- Proposition 1 : **GMM/GMR en mode simple** (GG-sing),
- Proposition 2 : **W-GMM/GMR en mode simple** (WGG-sing).

L'algorithme W-GMM/GMR en mode multiple n'est pas testé. En effet, ce mode n'est pas approprié en mode joint puisqu'il génère une trajectoire de mauvaise qualité.

L'apprentissage est compliqué quand les données comportent des évolutions brusques au regard de la fréquence d'échantillonnage, comme avec des mouvements sinusoïdaux. Les tâches choisies pour l'expérience sont : « Coucou » (CC) et « Gobelet dans Mug » (GiM), comme le montre la Figure 18. La tâche « Coucou » consiste à saluer de la main. La tâche « Gobelet dans Mug » a pour objectif de placer un gobelet dans un mug.

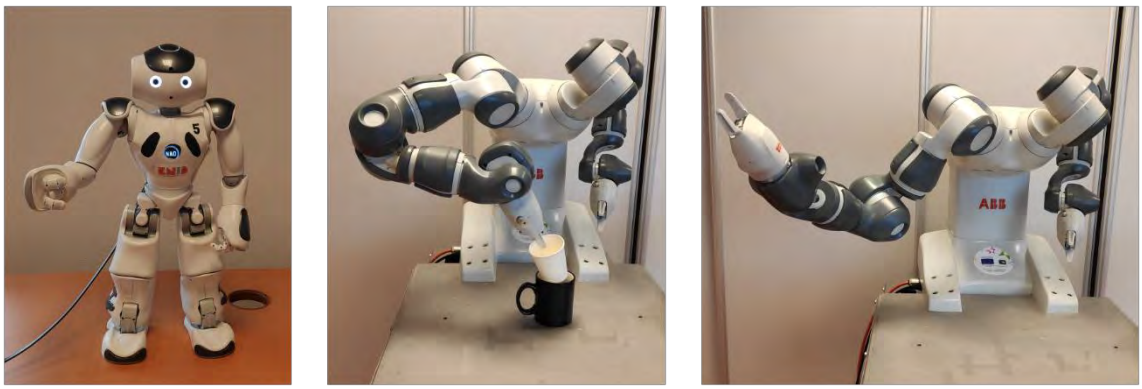


Figure 18 : Photos des deux robots et des deux tâches

De gauche à droite (robot-tâche) : Nao-CC, YuMi-CC, YuMi-GiM

La tâche « Coucou » est réalisée avec les robots Nao et YuMi. La seconde tâche est effectuée uniquement avec le robot YuMi. Les données du Nao sont enregistrées avec une fréquence d'échantillonnage de 15 Hz et de 250 Hz pour le robot YuMi.

Pour chaque tâche, dix bonnes démonstrations sont réalisées. Les données de ces démonstrations sont ensuite préparées avec les deux étapes de pré-traitement de la méthode M2, mentionnée précédemment, puis l'apprentissage est réalisé.

Pour le modèle WGG-sing, les poids attribués aux données sont représentés dans la Figure 19.

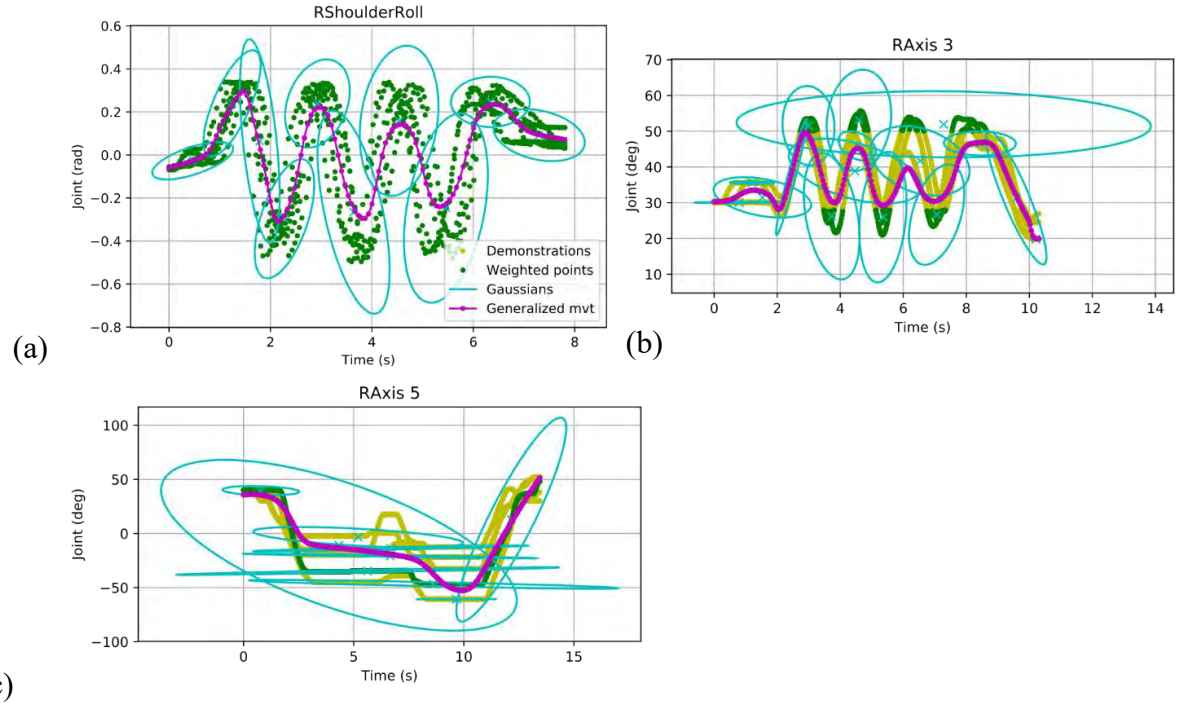


Figure 19 : Pondération des données de chaque tâche et la trajectoire générée avec le modèle WGG-sing

(a) Nao-CC, (b) YuMi-CC, (c) YuMi-GiM

La valeur des poids, déterminée de manière empirique, est différente pour chaque tâche. Les poids choisis illustrent certaines possibilités données par l'apprentissage proposée.

Avec Nao et la tâche CC, toutes les données sont pondérées à 1,15. La pondération de tous les points de données aide l'apprentissage lorsque la fréquence d'échantillonnage est faible. Bien que non intuitive, la pondération de toutes les données a un impact sur l'apprentissage.

Pour YuMi et la tâche CC, les pics de la sinusoïde sont pondérés à 9,5.

Pour YuMi et la tâche GiM, les données d'une démonstration sélectionnée ont un poids de 5. La pondération d'une seule démonstration, considérée comme une bonne démonstration à suivre, oriente l'apprentissage vers cette bonne trajectoire.

Excepté pour la pondération de toutes les données ou la pondération d'une seule démonstration, les points à pondérer sont facilement choisis en sélectionnant les points supérieurs ou inférieurs à une valeur spécifique de joint ou en définissant une ou des fenêtres temporelles.

3.3.2 Mesures

Le succès d'une trajectoire est compliqué à définir car il n'est pas possible de comparer la trajectoire générée avec la trajectoire parfaite car elle est non définie (voir la définition de la démonstration idéale dans la Section 2.2.2). Le choix des mesures dans le domaine de l'apprentissage par démonstration reste toujours un défi à relever (Argall et al., 2009 ; Chernova & Thomaz, 2014).

Dans cette expérience, la trajectoire générée sera donc comparée à toutes les démonstrations. Les mesures retenues pour comparer les modèles sont le taux de succès (SR), l'erreur absolue moyenne finale (FMAE) et l'erreur quadratique moyenne finale (FMSE). Le taux de succès est un score sur 100, défini avec le Tableau 4, page 38. FMAE et FMSE, définies par les Équations (2) et (3), sont calculées entre la trajectoire générée et toutes les démonstrations. Par conséquent, le meilleur modèle est celui ressemblant le plus aux démonstrations, c'est-à-dire celui récoltant les meilleures valeurs de FMAE/FMSE tout en ayant un bon taux de succès.

3.3.3 Résultats

Les résultats de l'expérience sont présentés dans le Tableau 14 pour le taux de succès et le Tableau 14 pour les valeurs de FMAE et FMSE.

Tableau 14 : Taux de succès de chaque modèle sur 100

Modèles	Robot-tâche		
	Nao-CC	YuMi-CC	YuMi-GiM
GG-mult	75	38	35
GG-sing	92	45	40
WGG-sing	**98**	**60**	**55**

Tableau 15 : FMAE / FMSE pour chaque modèle

Modèles	Robot-tâche		
	Nao-CC	YuMi-CC	YuMi-GiM
GG-mult	5,56e-1 / 1,84e-1	6,93e-1 / 4,87e-1	3,90e-1 / 0,84e-1
GG-sing	4,53e-1 / 0,94e-1	**6,68e-1 / 4,60e-1**	4,65e-1 / 1,71e-1
WGG-sing	**4,22e-1 / 0,77e-1**	7,28e-1 / 6,84e-1	**3,42e-1 / 0,77e-1**

Les trajectoires générées par chaque modèle sont représentées par la Figure 20 pour le robot Nao et la tâche CC, par la Figure 21 pour le robot YuMi et la tâche CC et par la Figure 22 pour le robot YuMi et la tâche GiM.

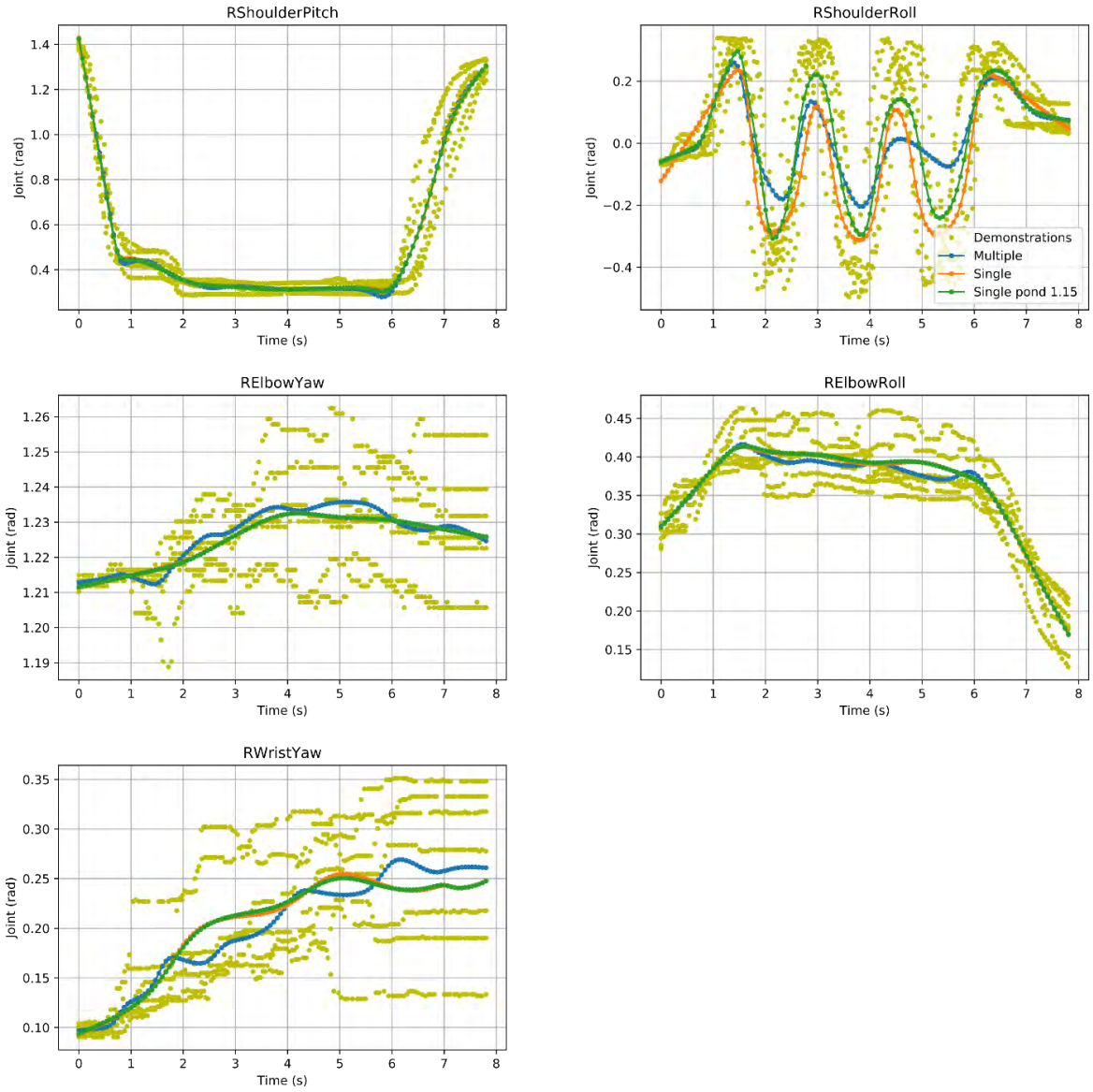


Figure 20 : Comparaison des modèles avec le robot Nao et la tâche CC

Multiple = GG-mult, Single = GG-sing, Single pond = WGG-sing

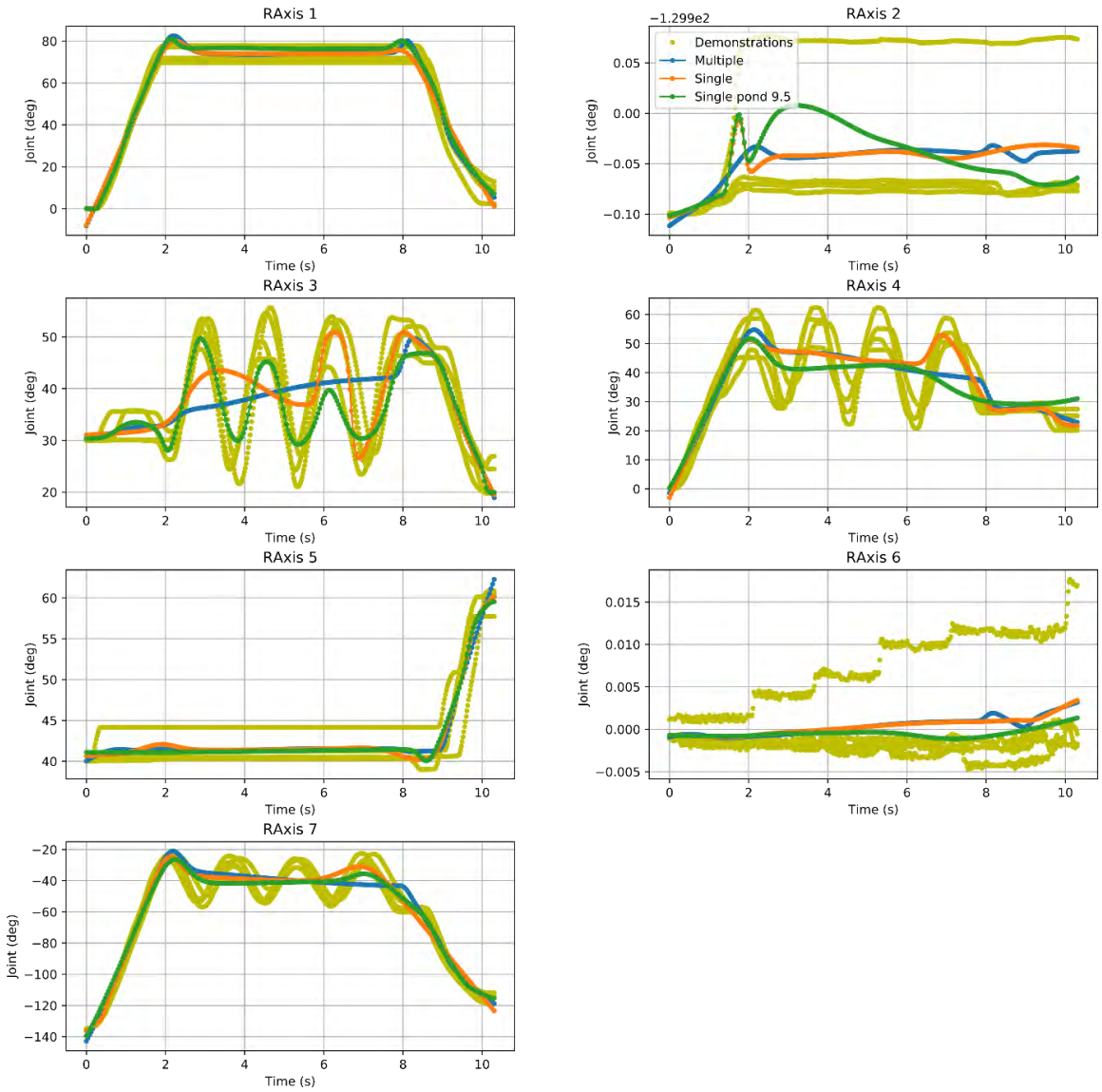


Figure 21 : Comparaison des modèles avec le robot YuMi et la tâche CC

Multiple = GG-mult, Single = GG-sing, Single pond = WGG-sing

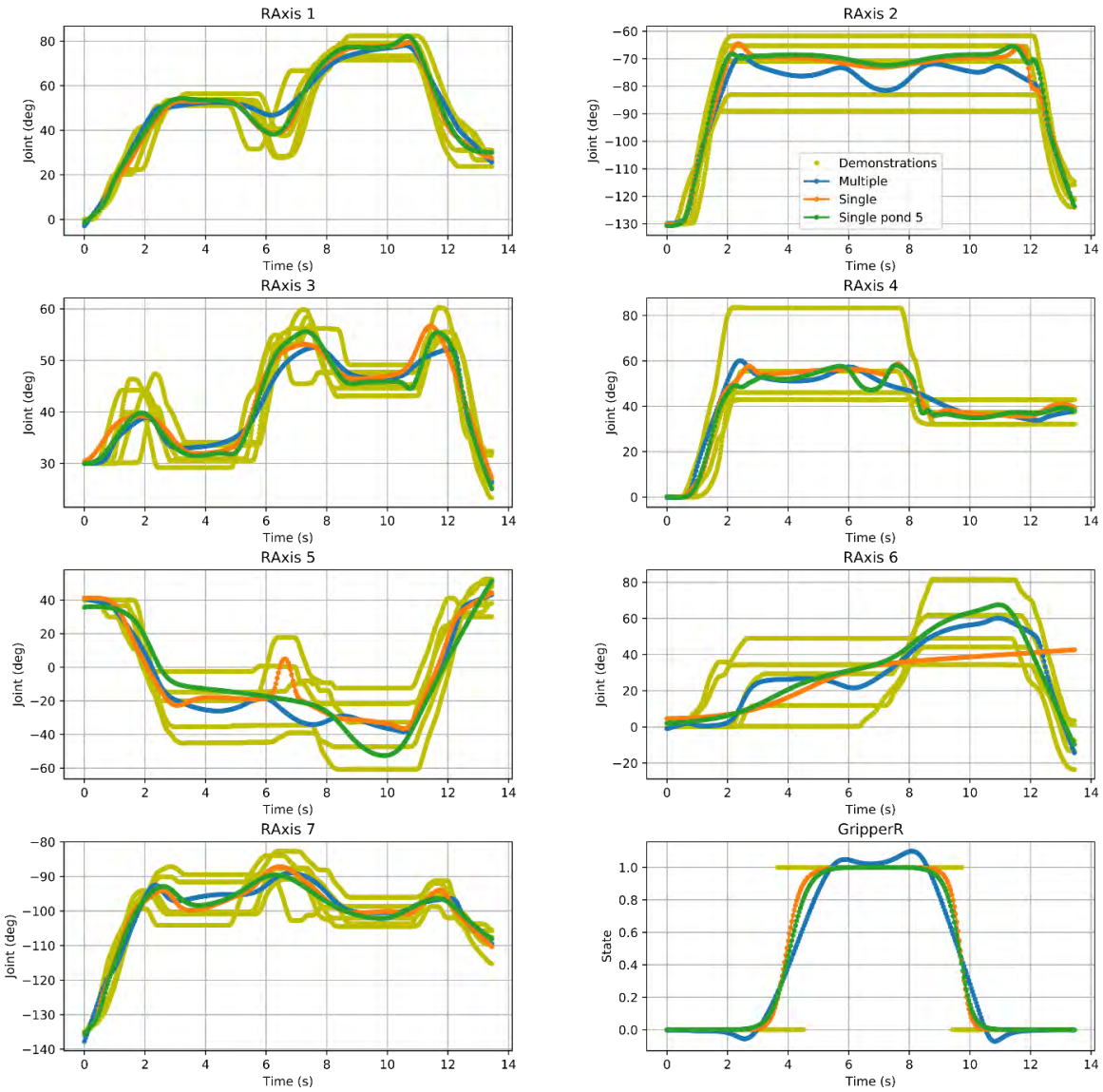


Figure 22 : Comparaison des modèles avec le robot YuMi et la tâche GiM

Multiple = GG-mult, Single = GG-sing, Single pond = WGG-sing

Le modèle GMM/GMR avec le mode multiple donne les moins bons résultats sur toutes les mesures. En effet, ce modèle attribue le même nombre de gaussiennes pour tous les joints, ainsi la trajectoire générée n'est pas optimale.

Le modèle GMM/GMR avec le mode simple améliore les résultats par rapport au même modèle en mode multiple.

Le modèle GMM/GMR avec les données pondérées fait progresser le modèle en mode simple en aidant l'apprentissage quand cela est nécessaire. Ce modèle donne les meilleurs résultats au niveau du taux de succès pour chaque tâche. Les valeurs de FMAE et FMSE sont les meilleurs avec le modèle utilisant des poids sauf pour le robot YuMi et la tâche CC. La Figure 21 montre que le modèle GMM/GMR avec les données pondérées est le seul modèle capable de suivre la sinusoïde de l'axe 3 du bras droit (RAxis 3). En conséquence, ce modèle est celui obtenant le meilleur taux de succès car il est le seul capable de répliquer la bonne séquence de mouvements pour effectuer un salut de la main.

Avec le robot Nao, le modèle W-GMM/GMR en mode simple avec des poids sur chaque donnée réduit l'impact de la faible fréquence d'échantillonnage qui complique l'apprentissage et en particulier dans le cas de variations soudaines dans le mouvement.

Ce modèle peut aussi pondérer une démonstration spécifique afin d'orienter l'apprentissage selon une bonne démonstration sélectionnée par l'utilisation, comme le montre le robot YuMi et la tâche GiM.

Dans le cas où les données des démonstrations sont la valeur des joints, le modèle GMM/GMR fonctionne mieux s'il est utilisé en mode simple. Ce mode simple peut être amélioré en pondérant certains points des données, choisis par l'utilisateur.

3.3.4 Discussion

Avec les données en mode joint, le modèle proposé, W-GMM en mode simple, obtient les meilleurs résultats. Ce modèle permet à l'utilisateur d'adapter la pondération en fonction de points qu'il estime importants dans la trajectoire.

L'expérience proposée montre plusieurs possibilités d'utilisation de la pondération.
Il est possible de pondérer tous les points des données lorsque la fréquence d'échantillonnage est faible. Cette pondération est encore plus efficace lorsque des changements brusques ont lieu avec une faible fréquence d'échantillonnage.
Une autre possibilité consiste à pondérer une seule démonstration afin d'influencer l'apprentissage selon cette dernière.
La dernière option de pondération proposée consiste à mettre des poids sur des points particuliers, ce qui permet d'attirer la génération de trajectoire vers ces points qui sont d'autant plus importants.

Conclusion

L'apprentissage par démonstration est un moyen facile et simple d'utilisation pour programmer un cobot. Le modèle GMM/GMR extrait les contraintes principales d'une tâche à partir d'un ensemble de démonstrations afin de générer une trajectoire.

Nous avons proposé d'améliorer les performances de ce modèle en permettant la pondération des données. L'utilisateur choisit quelles sont les données qui ont le plus d'importance afin d'améliorer la trajectoire générée. Le modèle proposé est testé sur différentes tâches et plusieurs robots.

Le modèle GMM/GMR standard peut avoir des problèmes à modéliser des données contenant des changements rapides. Ce défaut peut être diminué en utilisant des données pondérées qui permettent au mouvement généré de mieux suivre les variations des données d'entrée.

Le mode simple proposé permet d'améliorer de manière notable l'apprentissage en individualisant le traitement du mouvement pour chaque joint. Cependant, pour des données en mode cartésien, le mode multiple doit être privilégié car les dimensions des données sont liées entre elles. La vitesse de la trajectoire générée n'est pas un paramètre critique puisqu'elle peut être facilement ajustée à l'aide d'un facteur appliqué au vecteur du pas de temps de la trajectoire.

À l'aide de cet apprentissage, n'importe quelle tâche peut être enseignée à un robot à l'aide de démonstrations kinesthésiques. La seule contrainte au niveau des tâches est que la possible modification de la position des objets et de l'environnement du robot n'est pas prise en considération dans ce modèle. L'environnement du robot nécessitera donc d'être totalement figé.

L'apprentissage fonctionne également avec les 2 bras d'un robot. Pour faciliter les démonstrations avec 2 bras robotiques, deux personnes peuvent s'occuper chacune d'un bras lors de chaque démonstration.

Chapitre 4
Acceptabilité de l'apprentissage par démonstration

« Il faut parfois longtemps pour qu'une vérité démontrée devienne une vérité acceptée. »

Gustave Le Bon[6]

Sommaire

[6] Les incertitudes de l'heure présente, 1923

De nombreux robots nous aident au quotidien, comme le robot tondeuse ou le robot de cuisine. Les robots se développent de plus en plus et se retrouvent dans des domaines très variés comme la médecine, le divertissement, les compagnons sociaux ou l'industrie. Tous ces robots sont conçus pour des tâches spécifiques mais nécessitent des programmeurs en robotique et du temps de développement pour leur programmation (Figure 23). L'arrivée de la programmation par démonstration permet à un utilisateur non-expert en programmation ni en robotique de modifier facilement le comportement d'un robot, ce qui augmente la flexibilité des tâches que le robot peut accomplir. Cependant, cette nouvelle programmation est-elle acceptée par tout le monde ? L'utilisation et la compréhension de l'intérêt de ces robots est également très intimement liées à leur acceptabilité. Faire apprendre un mouvement à un robot via la programmation par démonstration implique directement son utilisateur et devrait augmenter son intérêt et son envie de l'utiliser.

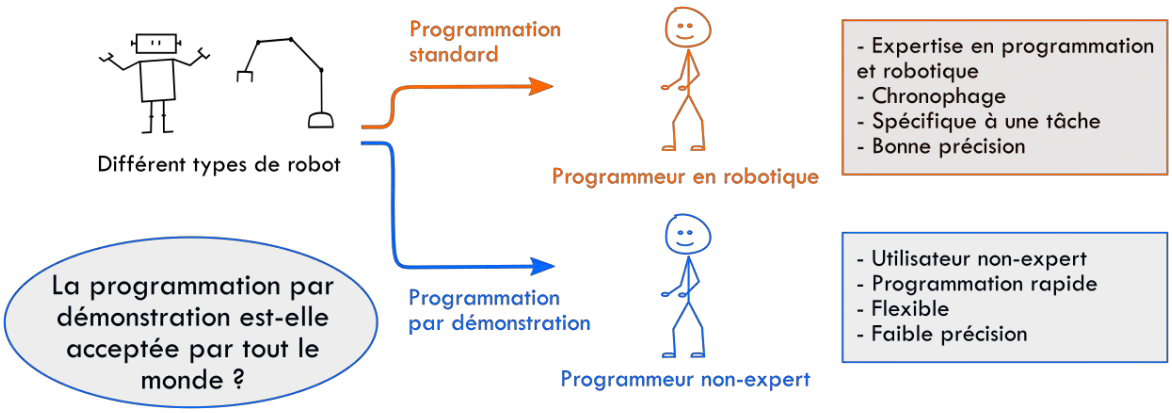


Figure 23 : Programmation standard vs Programmation par démonstration

Les cobots, ou robots collaboratifs, se développent principalement dans le domaine industriel. Ces cobots travaillent dans le même espace que les opérateurs humains et fonctionnent ensemble afin de réaliser une tâche. Pour faciliter leur utilisation, la programmation par démonstration permet à une seule personne de programmer le robot, le démarrer, l'arrêter et l'utiliser. Cette programmation est développée pour permettre à n'importe qui de pouvoir programmer un robot.

Or, cette programmation doit être comprise et acceptée par l'utilisateur. L'aspect humanoïde du cobot, ou anthropomorphisme, peut également impacter cette acceptabilité (Mori, 1970).

Ce chapitre cherchera donc à évaluer l'acceptabilité de la programmation par démonstration ainsi que l'impact de l'anthropomorphisme dans un contexte industriel (Figure 24). La Section 4.1 étudie la littérature au niveau de l'acceptabilité et de l'anthropomorphisme de manière générale. L'anthropomorphisme des robots est évalué dans la Section 4.2. La Section 4.3 présente une expérience sur l'acceptabilité de la programmation par démonstration et l'impact de l'anthropomorphisme.

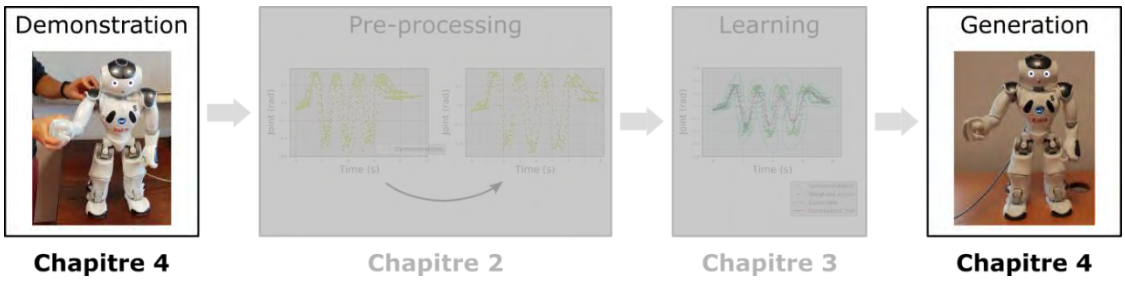


Figure 24 : Organisation du manuscrit - Chapitre 4

4.1 Acceptabilité et anthropomorphisme

L'acceptabilité d'une nouvelle technologie définit le niveau de tolérance de l'utilisation de cette dernière. Est-ce qu'une nouvelle technologie est acceptée par les utilisateurs ? De quelle manière les utilisateurs s'approprient un nouveau système ? Quelles sont les dimensions qui affectent cette acceptabilité ?

L'acceptabilité peut se mesurer à trois moments différents : a priori, après une utilisation, au quotidien (Terrade et al., 2009). L'acceptabilité a priori porte sur la représentation que l'on se fait de l'usage et de l'intérêt de la technologie. Après une utilisation de la technologie, l'utilisateur se fait une meilleure idée de l'utilité et de la facilité d'utilisation du dispositif. Lorsque la technologie est utilisée au quotidien, l'appropriation du système peut s'observer de manière très claire.

Selon (Nielsen, 1994), l'acceptabilité d'un système se compose de l'acceptabilité sociale et l'acceptabilité pratique.

L'acceptabilité sociale mesure la propension de la société à percevoir la technologie dans la vie de tous les jours (Terrade et al., 2009). Elle prend en compte le comportement d'usage qui inclut de nombreux facteurs, comme par exemple, les jugements de valeurs, les stéréotypes, la morale, l'éthique et la perception sociale dans le cadre de l'usage de la technologie.

L'acceptabilité pratique, quant à elle, est définie par la manière dont une personne perçoit la technologie lors de son utilisation. Elle s'appuie sur diverses dimensions, comme par exemple, l'utilité théorique, l'utilisabilité, l'efficience, la fiabilité, la facilité d'apprentissage et la satisfaction.

L'acceptabilité des robots, ou des cobots, se distingue de l'acceptabilité d'une nouvelle technologie par le fait que l'on puisse attribuer une personnalité à un robot, particulièrement en ce qui concerne les robots humanoïdes. Dans ce cadre, la forme du robot peut également jouer un rôle dans l'acceptation de ce dernier.

Souvent, l'acceptabilité des robots est entachée par certaines craintes, partagées par le grand public (Gelin & Guilhem, 2016) : le robot va me remplacer au travail ; il va diminuer les interactions humaines ; il va conserver des données personnelles sur moi et va me juger ; le robot peut-il différencier « le bien et le mal » ? La plupart des personnes se sentent davantage menacées par les robots autonomes que par les robots non-autonomes (Złotowski et al., 2017).

Si les attentes que l'on se fait des robots sont trop élevées par rapport aux capacités réelles des robots, leur acceptabilité sociale s'en trouvera affectée. Dans de nombreux cas, le simple fait d'expliquer ce dont est capable le robot, comment il fonctionne et à quoi il sert, suffit à faire disparaître certaines des appréhensions envers des robots.

Par exemple, les robots dans l'industrie aident les opérateurs humains, les soulagent dans des tâches risquées ou permettent d'éviter les troubles musculo-squelettiques (TMS) liés au travail répétitif. Il a été observé que l'introduction des robots dans l'industrie n'a pas augmenté le chômage (IFR, 2017). Prendre en considération toutes ces dimensions lors de la conception des robots permet d'améliorer leur acceptabilité (Salvini et al., 2010). L'arrivée des cobots dans l'industrie nécessite d'associer les futurs utilisateurs au processus de leur développement afin d'augmenter leur acceptabilité (Bauer et al., 2016). Une bonne équipe de travail humain-robot repose sur une bonne communication, sur une coordination harmonieuse et sur une collaboration efficace (Mingyue Ma et al., 2018).

De nombreux facteurs impactent l'acceptabilité d'une technologie et en particulier celle des robots. Moins la performance du robot est bonne, plus l'utilisateur perd confiance dans le robot (Hedlund et al., 2021). La fluidité lors de la collaboration revêt également toute son importance lors d'une interaction humain-robot (Hoffman, 2019). D'autres facteurs sont également à prendre en considération, comme la vitesse de réponse du robot, sa trajectoire, la proximité humain-robot et la charge de travail de l'utilisateur (Wang et al., 2019). Le contexte, l'environnement et la tâche de travail influencent également l'acceptabilité des robots (Mingyue Ma et al., 2018).

(Bobillier-Chaumon & Dubois, 2009) propose de répartir les facteurs d'acceptation d'une technologie en cinq catégories :

- Dimensions intra-individuelles : facteurs ayant un coût (cognitif, affectif…) pour l'individu ;
- Dimensions interindividuelles : évolutions des relations au travail, des collectifs ;
- Dimensions métapersonnelles : effets sur les organisations ;
- Dimensions transpersonnelles : conséquences sur l'identité et la profession de l'individu ;

- Dimensions impersonnelles : ressentiments en termes de confiance et de fiabilité de la technologie.

Contrairement à de nombreuses nouvelles technologies, les robots sont créés avec des caractéristiques physiques qui leur donnent une apparence ayant un impact certain sur leur acceptabilité. L'anthropomorphisme se définit comme le processus d'assignement de caractéristiques humaines - physiques ou comportementales - à un agent non-humain (Waytz et al., 2010).

En outre, associer des capacités mentales à un robot entraine une certaine attente des utilisateurs ainsi que des conséquences en termes d'identité, de morale, de responsabilité et d'influence sociale (Waytz et al., 2010). Souvent, on attribue des capacités mentales et comportementales à des robots alors que leur comportement, créé artificiellement, est bien moins développé que l'impression qu'ils nous en donnent (Duffy, 2003). Si les attentes que l'on a d'un robot sont trop éloignées de ses véritables capacités, le robot sera moins bien accepté (Duffy, 2003).

Généralement, on dit d'un robot qu'il est anthropomorphe lorsqu'il ressemble à un être humain. Augmenter l'anthropomorphisme des robots, comme dans le cadre du développement des voitures autonomes, pourrait améliorer la confiance que l'on attribue dans les machines (Waytz et al., 2014).

On appelle robots humanoïdes, les robots ayant des caractéristiques physiques humaines, comme une tête, des bras et des jambes. Des caractéristiques anthropomorphiques comme une tête avec des yeux et une bouche faciliteraient les interactions sociales car robots et êtres humains utiliseraient les mêmes mécanismes de communication. Le choix du design et des capacités cognitives des robots est donc très important et doit être pris en compte lors de la conception du robot. Il est ainsi possible d'éviter la crainte que peuvent susciter les robots et donc d'en faciliter leur utilisation (Duffy, 2003).

Des robots qui ressemblent trop à des êtres humains peuvent en effet provoquer de l'inconfort voir de la peur (Dinet & Vivian, 2015 ; Strait et al., 2017).

Il paraîtrait même qu'au niveau de la forme du robot, l'aspect plutôt masculin ou féminin aurait également un impact sur les capacités et comportements attendus du robot (Beraldo et al., 2018). Un aspect féminin du robot pourrait augmenter l'efficacité des interactions humain-robot (Strait et al., 2017).

Néanmoins, comme le montre la théorie de la « vallée de l'étrange » (ou *uncanny valley*) (Mori, 1970), l'apparence humaine des robots ne doit pas dépasser un certain seuil pour éviter de créer un malaise ou une attitude négative envers les robots (Figure 25). Un robot humanoïde doit donc conserver un certain aspect robotique afin d'être plus accepté par les usagers.

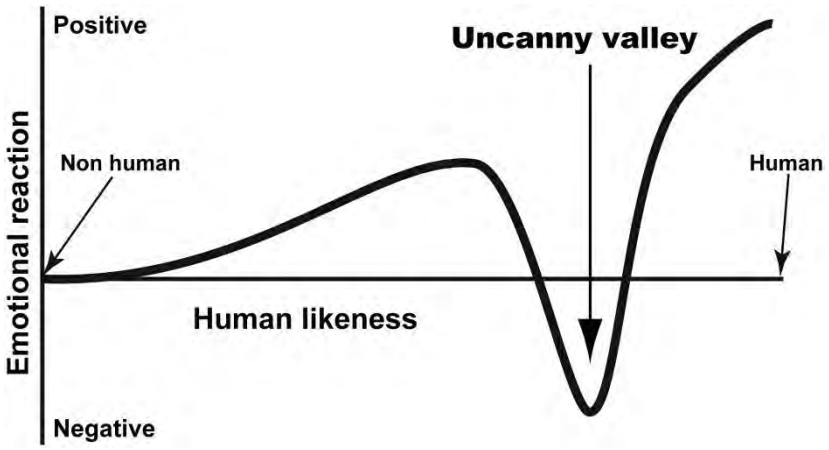


Figure 25 : Représentation simplifiée de la « vallée de l'étrange » (Sasaki et al., 2017)

Les études sur l'acceptabilité des robots dépendantes du genre, de la nationalité et de l'âge des personnes ainsi que la forme du robot sont difficilement généralisables à cause des limites méthodologiques (Dinet & Vivian, 2015). Les expériences doivent donc prendre en compte au maximum tous ces paramètres.

La forme humanoïde des robots est imaginée à l'origine afin de créer une machine à l'image de l'homme (Gelin & Guilhem, 2016) mais finalement, cette forme augmente-t-elle l'acceptabilité de ces robots ?

Divers modèle ou échelles proposent de théoriser l'acceptabilité des utilisateurs envers une nouvelle technologie, en s'appuyant, généralement, sur des données obtenues à partir d'échelles de Likert. Une échelle de Likert est constituée d'une ou plusieurs affirmations pour lesquelles un participant énonce son degré d'accord ou de désaccord. Pour chaque modèle ou échelle, divers facteurs sont relevés et chacun se compose généralement de plusieurs items.

Le modèle UTAUT (*Unified Theory of Acceptance and Use of Technology*), proposé par (Venkatesh et al., 2003), se base sur l'étude de nombreux modèles existant pour définir les facteurs les plus importants de l'acceptabilité. Ces facteurs sont les attentes en termes de performance, en termes d'effort, l'attitude envers l'utilisation de la technologie, l'influence sociale, les conditions facilitantes, l'efficacité de l'utilisateur, l'anxiété et l'intention d'utilisation. Tous ces facteurs sont également influencés par le genre, l'âge et l'expérience de l'utilisateur. Chaque facteur est composé de diverses affirmations.

L'échelle TAM (*Technology Acceptance Model*) (Davis, 1989) porte sur les perceptions de l'efficacité et de la facilité d'utilisation de la nouvelle technologie avec de nombreux items pour chaque facteur. Ces deux facteurs sont évalués avec une échelle de Likert sur 7.

D'autres échelles se fondent sur d'autres dimensions impactant également l'acceptabilité des robots, comme la charge de travail ou la confiance.

L'échelle NASA-TLX (*NASA-Task Load Index*) évalue la charge de travail d'un utilisateur lors d'une activité (Hart & Staveland, 1988). Les facteurs retenus dans cette échelle sont au nombre de six : la charge mentale, la charge physique, la pression temporelle, la performance, le niveau d'effort et le niveau de frustration. Toutes ces dimensions s'évaluent à l'aide d'une ligne dont chaque extrémité comporte un adjectif décrivant l'un des deux extrêmes de la dimension courante.

D'autres échelles se basent sur les ressentis des participants, comme l'échelle NARS (*Negative Attitudes Toward Robots*) (Nomura et al., 2006a) et l'échelle RAS (*Robot Anxiety Scale*) (Nomura et al., 2006b). Le NARS évalue l'attitude négative envers les interactions avec les robots, leur influence sociale et les interactions émotionnelles avec les robots. Le RAS mesure le niveau d'anxiété envers les capacités de communication, les caractéristiques comportementales et le discours avec les robots. Ces deux échelles s'appuient des échelles de Likert pour évaluer tous ces facteurs.

Le questionnaire QUEAD (*Questionnaire for the Evaluation of Physical Assistive Devices*) (Schmidtler et al., 2017) mélange divers aspects afin de créer un questionnaire pour évaluer au mieux des dispositifs d'assistance physique. Les facteurs retenus sont l'efficacité perçue, la facilité d'utilisation perçue, les émotions, l'attitude et le confort.

(Charalambous et al., 2016) propose une échelle afin d'évaluer la confiance dans le cadre d'une collaboration humain-robot au sein du secteur industriel. Les facteurs retenus sont principalement liés à la performance du robot : le mouvement du robot et la vitesse de préhension, la coopération en sécurité, la fiabilité du robot et de la pince.

Pour évaluer spécifiquement l'anthropomorphisme perçu, (Fussell et al., 2008) propose d'évaluer le ressenti des participants à l'aide d'une liste d'adjectifs, positifs et négatifs, regroupés en quatre catégories : adjectifs liés à la sociabilité humaine, aux traits de personnalité et d'état, à la robotique, et d'autres adjectifs.

Une autre possibilité consiste à évaluer des adjectifs avec deux extrêmes, répartis de part et d'une échelle de type Likert. L'échelle GodSpeed (Bartneck et al., 2009) utilise ce type d'évaluation et les adjectifs se classent selon les facteurs suivants : l'anthropomorphisme, la perception de vie, l'amabilité, l'intelligence perçue et la sécurité perçue.

Le Tableau 16 présente les facteurs pouvant influencer l'acceptabilité des nouvelles technologies / robots regroupés par modèle ou échelles.

Tableau 16 : Facteurs pouvant influencer l'acceptabilité regroupés par modèle / échelles

Facteurs = nombre d'items ou d'adjectif des extrêmes

Références	Nom du modèle ou de l'échelle	Nature de l'échelle	Facteurs
(Venkatesh et al., 2003)	UTAUT	Likert sur 3 à 7	- Espérance de performance (x4) - Espérance d'effort (x4) - Attitude envers l'utilisation de la technologie (x4) - Influence sociale (x4) - Conditions facilitantes (x4) - Efficacité de l'utilisateur (x4) - Anxiété (x4) - Intention d'utilisation (x3)
(Davis, 1989)	TAM	Likert sur 7	- Efficacité perçue (x6) - Facilité d'utilisation perçue (x6)
(Hart & Staveland, 1988)	NASA-TLX	Ligne graduée	- Charge mentale (faible / élevée) - Charge physique (faible / élevée) - Charge temporelle (faible / élevée) - Performance (faible / bonne) - Effort (faible / élevé) - Niveau de frustration (faible / élevé)
(Nomura et al., 2006a)	NARS	Likert sur 5	- Attitude négative envers l'interaction avec les robots (x6) - Attitude négative envers l'influence sociale des robots (x5) - Attitude négative envers les interactions émotionnelles avec les robots (x3)
(Nomura et al., 2006b)	RAS	Likert sur 6	- Anxiété sur les capacités de communication des robots (x3) - Anxiété sur les caractéristiques comportementales des robots (x4) - Anxiété sur le discours avec les robots (x4)

Suite du Tableau 16 : Facteurs pouvant influencer l'acceptabilité regroupés par modèle / échelles

Facteurs = nombre d'items ou d'adjectif des extrêmes

Références	Nom du modèle ou de l'échelle	Nature de l'échelle	Facteurs
(Schmidtler et al., 2017)	QUEAD	Likert sur 7	- Efficacité perçue (x6) - Facilité d'utilisation perçue (x9) - Émotions (x5) - Attitude (x3) - Confort (x3)
(Charalambous et al., 2016)	Echelle de confiance	Likert sur 5	- Mouvement du robot et vitesse de prise (x2) - Coopération en sécurité (x4) - Fiabilité du robot et de la pince (x4)
(Fussell et al., 2008)	Liste d'adjectifs	Liste d'adjectifs vrai-faux	- Sociabilité humaine (x10) - Traits de personnalité ou d'état (x10) - Robotique (x10) - Autres adjectifs (x10)
(Bartneck et al., 2009)	GodSpeed	Likert sur 5	- Anthropomorphisme (x5) - Perception de vie (x6) - Amabilité (x5) - Intelligence perçue (x5) - Sécurité perçue (x3)

Dans l'expérience décrite dans la Section 4.3, le questionnaire développé pour évaluer l'acceptabilité se basera sur les modèle et échelles existants. En effet, il n'existe aucun questionnaire permettant d'évaluer l'acceptabilité de la programmation par démonstration de cobots. Ainsi, le questionnaire créé s'appuiera sur les modèle et échelles évaluant l'acceptabilité de nouvelles technologies, des robots et l'influence de l'anthropomorphisme.

4.2 Pré-expérience : anthropomorphisme des robots

4.2.1 Objectif

Cette pré-expérience vise à évaluer l'anthropomorphisme de différents robots et en particulier leur aspect. A quel point la forme d'un robot ressemble-t-elle à un être humain ? Les résultats en termes d'anthropomorphisme perçu par les participants permettront, dans la Section 4.3, de poser les bases pour évaluer l'impact de l'anthropomorphisme. L'objectif initial de cette pré-expérience est de déterminer si le robot Pepper est considéré comme anthropomorphe contrairement au robot YuMi. Afin d'obtenir des résultats plus généraux, d'autres robots sont inclus dans cette pré-expérience. L'hypothèse de recherche est la suivante :

H0 : « Les robots humanoïdes sont considérés comme les plus anthropomorphes. ».

Les robots disposant de caractéristiques physiques humaines comme des bras, une tête et des jambes devraient être reconnus comme ceux ressemblant le plus à un être humain. Cette pré-expérience s'appuie sur l'étude d'une variable indépendante à vingt modalités dont chacune correspond à un robot.

4.2.2 Méthode

4.2.2.1 Participants

Les participants à cette étude étaient les étudiants de l'école d'ingénieurs ENIB (Ecole Nationale d'Ingénieur de Brest). 162 personnes ont participé, dont 78% d'hommes. Quant à l'âge des participants, 81% ont entre 19 et 25 ans (moyenne : 21 ans, écart-type : 1,650). La majorité des participants sont de nationalité française et sont inscrits à partir de la troisième et jusqu'à la cinquième année à l'ENIB. Pour davantage d'informations, les statistiques descriptives du profil des participants sont illustrées par la Figure 26.

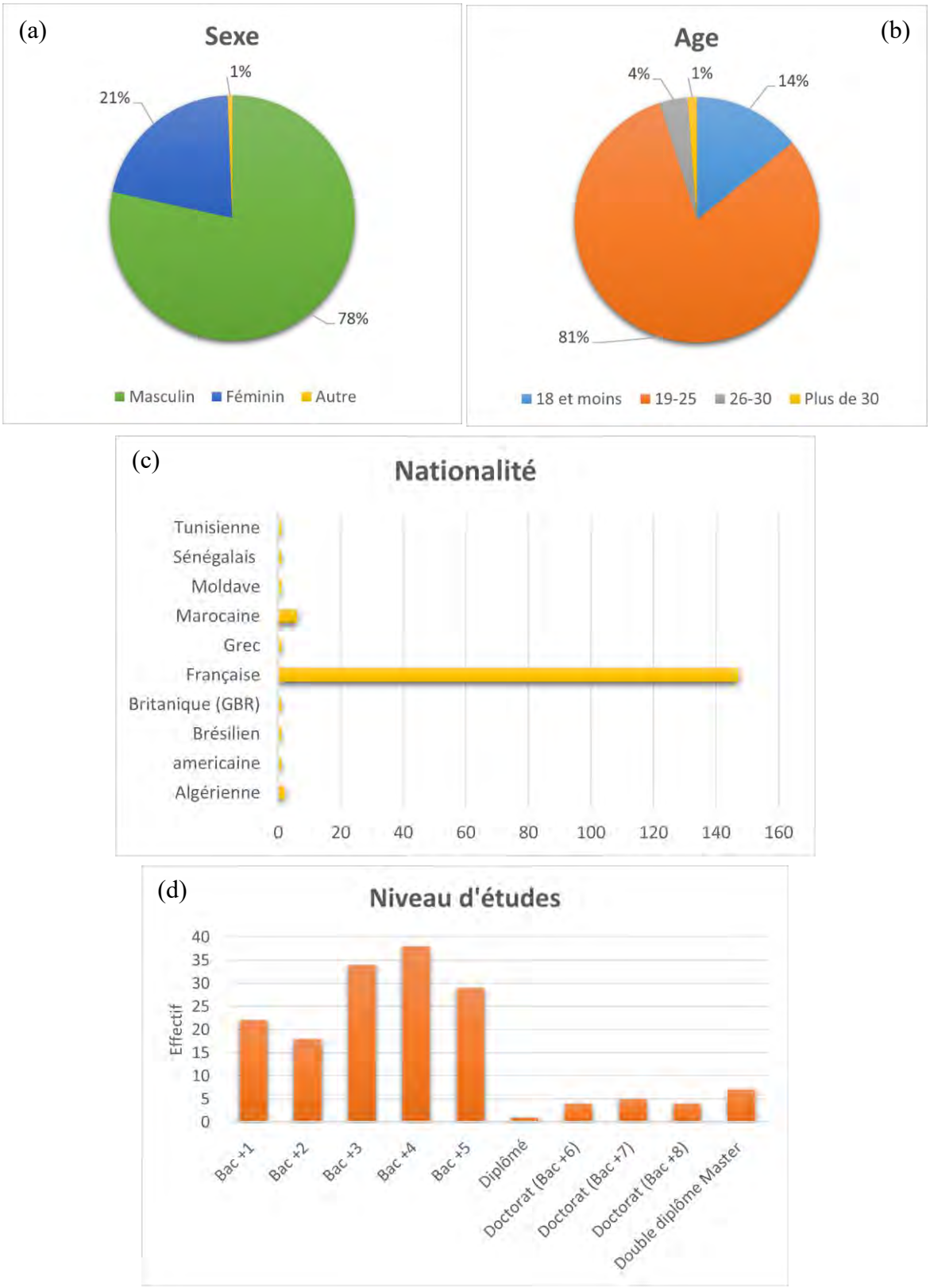


Figure 26 : Statistiques descriptives du profil des participants

(a) Sexe, (b) Age, (c) Nationalité, (d) Niveau d'études

4.2.2.2 Matériel

L'étude porte sur la perception du niveau d'anthropomorphisme de divers robots. A cette fin, un questionnaire est développé avec l'outil Google Forms. Il se compose de vingt questions, une par robot, et de quatre questions sur le profil des participants (Annexe 3).

Les vingt questions sur les robots se composent d'une unique question et d'une photo de robot. La question se compose d'une affirmation et d'une échelle de Likert de 1 à 7 (entre « Pas du tout d'accord » et « Tout à fait d'accord »). L'affirmation est la suivante : « Ce robot ressemble à un humain. ».

Le participant devait noter, pour chaque photo de robot, le niveau de ressemble avec un être humain (voir le questionnaire en Annexe 3). Les vingt robots sélectionnés représentent un large panel de robots existants : du robot tondeuse ou de cuisine à des robots plus humanoïdes comme Asimo ou Atlas. Les robots retenus sont les suivants, triés par ordre alphabétique : Aibo, araignée, Asimo, aspirateur, Atlas, Buddy, Cozmo, cuisine, drone, industriel, Kismet, Nabaztag, Nao, Pepper, ROV, Spot, TurtleBot, voiture, Winky, YuMi[7]. Chaque photo de robot est choisie avec un fond blanc afin de ne pas influencer l'utilisation que l'on peut lui attribuer. Sans fond, les participants n'ont également pas d'information sur la taille réelle du robot.

[7] Aibo : Sony, https://us.aibo.com/ ; araignée (Hexapod CR-6) : Robo-Soul, https://fr.aliexpress.com/item/32831830732.html ; Asimo : Honda, https://global.honda/innovation/robotics/ASIMO.html ; aspi-rateur (PCR-3100) : Sichler Haushltsgeräte, https://www.pearl.fr/article/ZX7075/robot-aspirateur-connecte-2500-mah-pcr-3100 ; Atlas : Boston Dynamics, https://www.bostondynamics.com/atlas ; Buddy : Blue Frog Robotics, https://www.bluefrogrobotics.com/robot/ ; Cozmo : Anki, https://www.digitaldreamlabs.com/pages/cozmo ; cuisine (KM-6618) : Rosenstein & Söhne, https://www.pearl.ch/de/kuechenmaschine-km-6618-inkl-umfangreiches-zubehoer.html ; drone (Kargu-2) : STM, https://www.stm.com.tr/en/kargu-autonomous-tactical-multi-rotor-attack-uav ; industriel (IRB-6620) : ABB, https://new.abb.com/products/robotics/industrial-robots/irb-6620 ; Kismet : Massachusetts Institute of Technology, http://www.ai.mit.edu/projects/sociable/baby-bits.html ; Nabaztag : Aldebaran Robotics, https://www.nabaztag.com/# ; Nao : SoftBank Robotics, https://www.softbankrobotics.com/emea/fr/nao ; Pepper : SoftBank Robotics, https://www.softbankrobotics.com/eea/fr/pepper ; ROV (Mini-ROV Observer) : Subsea Tech, https://www.subsea-tech.com/fr/mini-rov-observer/ ; Spot : Boston Dynamics, https://www.bostondynamics.com/products/spot ; TurtleBot : Willow Garage, https://www.turtlebot.com/turtlebot2/ ; voiture (Autonom Shuttle Evo): PME Navya, https://navya.tech/fr/solutions/transport-de-personnes/navette-autonome-pour-le-transport-de-passagers/ ; Winky : Mainbot, https://heywinky.com/ ; YuMi : ABB, https://new.abb.com/products/robotics/collaborative-robots/irb-14000-yumi.

Les vingt questions portant sur les robots apparaissaient dans un ordre aléatoire pour chaque participant afin d'éviter un effet d'ordre de présentation des robots.

4.2.2.3 Procédure expérimentale

Le questionnaire est envoyé à l'ensemble des étudiants de l'ENIB le 16 novembre 2019 et reste disponible en ligne pendant vingt-quatre jours. La durée estimée pour compléter le questionnaire est d'environ 5 minutes. Le questionnaire est entièrement anonymisé puisque seules quelques informations générales sur le profil des participants sont recueillies.

4.2.3 Résultats

Pour chaque robot, un score moyen est calculé en associant les réponses à des chiffres entre 1 et 7. Le chiffre 1 est associé à la réponse « Pas du tout d'accord » et 7 pour « Tout à fait d'accord ».

La moyenne de chaque robot est représentée à l'aide d'une frise (Figure 27) sur laquelle trois groupes se distinguent nettement.

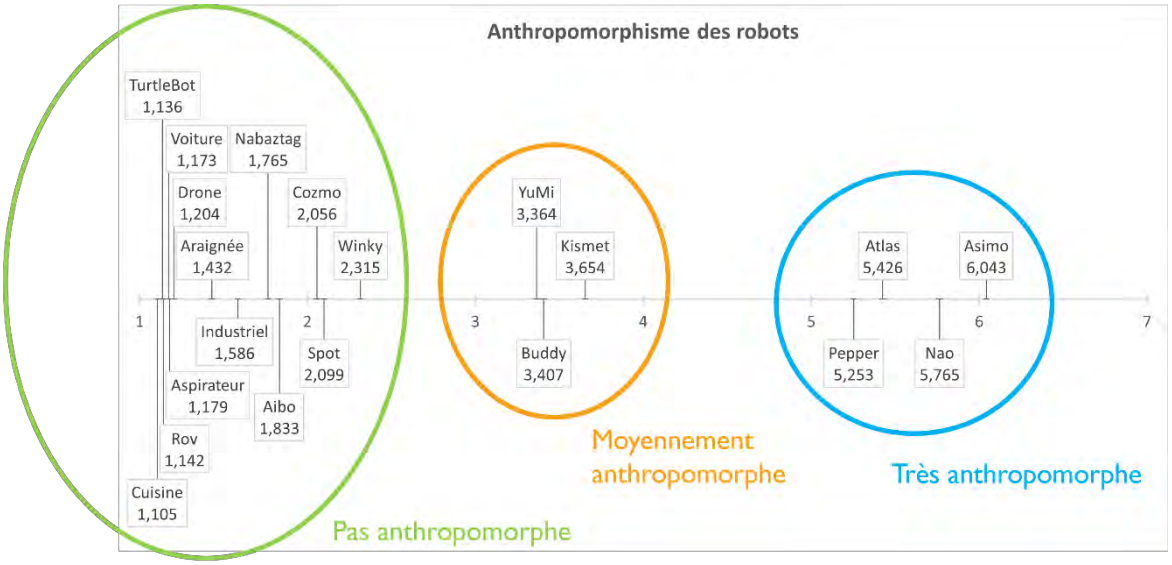


Figure 27 : Résultat de la pré-expérience sur l'anthropomorphisme des robots

Moyennes des résultats à la question « Ce robot ressemble à un humain. » (1 : Pas du tout d'accord, 7 : Tout à fait d'accord) pour chaque robot

Ces groupes sont liés au niveau d'anthropomorphisme perçu des robots : groupe « très anthropomorphe » (Figure 28), groupe « moyennement anthropomorphe » (Figure 29) et groupe « pas anthropomorphe » (Figure 30). Chaque groupe est analysé pour en comprendre les caractéristiques.

Le groupe « **très anthropomorphe** » (Figure 28) contient les quatre robots humanoïdes : Asimo, Nao, Atlas et Pepper. Asimo a obtenu le score le plus élevé parmi tous les robots. Ceci peut s'expliquer par le fait qu'il dispose d'une forme humaine complète avec des articulations bien dissimulées, ce qui crée une certaine harmonie au niveau de l'aspect extérieur. Le robot suivant est Nao, avec une forme humaine complète mais des caches extérieurs gris, ce qui ajoute un aspect plus robotique. Le robot Atlas obtient la troisième place. Atlas ne dispose pas de mains mais plutôt de boules et de nombreux câbles sont visibles sur l'ensemble du robot. Pepper se place en dernière position de ce groupe, surement dû au fait qu'il ne dispose pas de jambes. Le groupe « très anthropomorphe » peut donc se résumer aux caractéristiques suivantes : un corps, deux bras, une tête et éventuellement une paire de jambes.

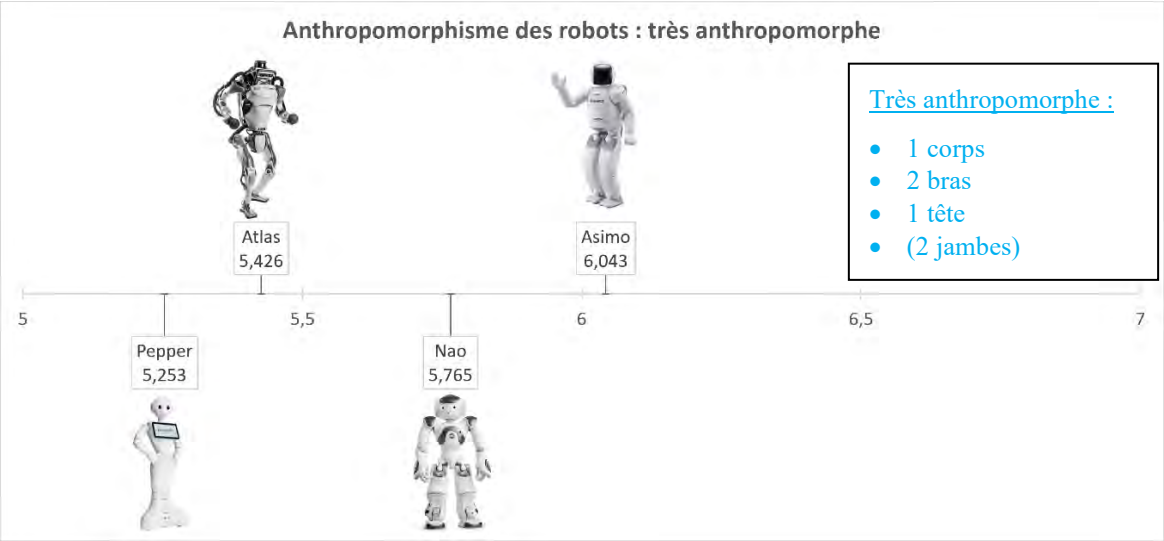


Figure 28 : Résultat de la pré-expérience sur l'anthropomorphisme des robots, groupe « très anthropomorphe »

Le deuxième groupe est qualifié de robots « **moyennement anthropomorphes** » (Figure 29). Ce groupe est composé, dans l'ordre décroissant, des trois robots suivants : Kismet, Buddy et YuMi. Le robot Kismet est composé d'une tête très expressive avec des moteurs qui permettent de faire bouger ses oreilles, sourcils, paupières, lèvres, mâchoire et tête. Le robot Buddy est mobile avec deux roues et une tablette qui lui fait office de tête très expressive avec des yeux et une bouche. Étonnamment, YuMi, robot sans tête, appartient à cette catégorie. Certes, il se retrouve à la dernière place mais ses bras et son torse lui offrent des caractéristiques que l'on associe aux êtres humains. Le groupe « moyennement anthropomorphe » regroupe les robots ayant une tête expressive ou deux bras.

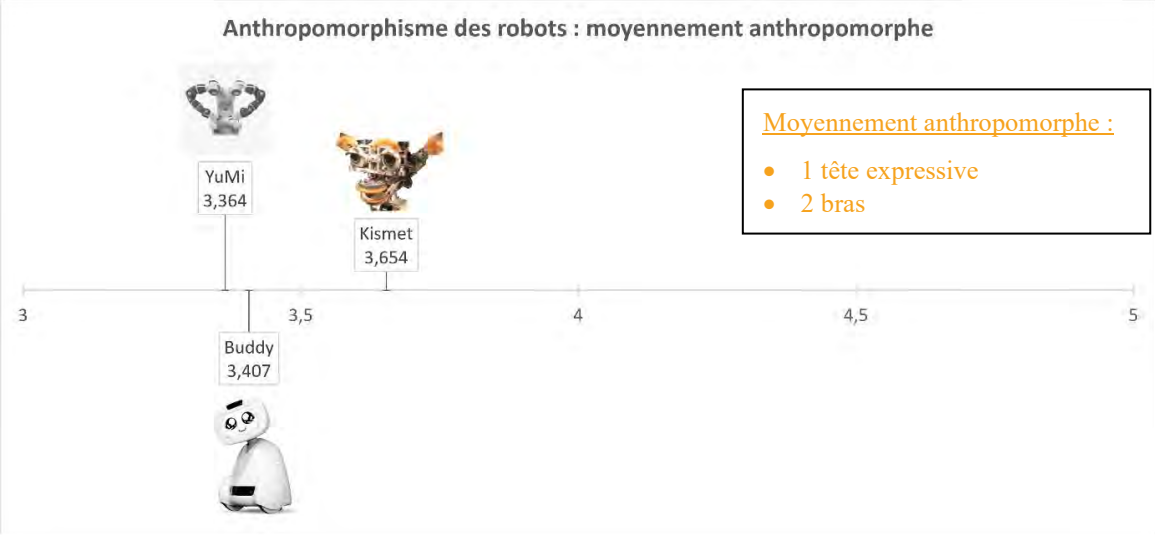


Figure 29 : Résultat de la pré-expérience sur l'anthropomorphisme des robots, groupe « moyennement anthropomorphe »

Le dernier groupe « **pas anthropomorphe** » est composé de tous les robots ayant le moins de caractéristiques physiques humaines (Figure 30). A la première place de ce groupe se trouve le robot Winky, robot plutôt expressif avec ses yeux et ses oreilles. Ce robot aurait pu également se trouver dans la catégorie des robots « moyennement anthropomorphes » car ayant une tête expressive. Peut-être que son manque d'expressivité lui a valu le déclassement dans le dernier groupe. Ensuite, on retrouve des robots soit de type « animal » comme le Spot, Aibo, araignée soit d'autres robots « séduisants » comme Cozmo, Nabaztag. Le robot industriel est également parmi ces robots car connu comme un robot utilisé dans l'industrie. Les derniers robots de ce groupe sont soit ceux que l'on peut retrouver dans la vie de tous les jours comme le robot aspirateur, le robot de cuisine, le drone ou la voiture autonome, soit des robots plutôt inconnus du grand public et dont l'utilisation est méconnue comme le ROV ou le TurtleBot. Il est pratiquement impossible de déterminer les caractéristiques particulières des robots placés dans le groupe « pas anthropomorphe ». Tous les robots ne rentrant pas dans les groupes « très anthropomorphe » ou « moyennement anthropomorphe » se retrouvent dans ce groupe.

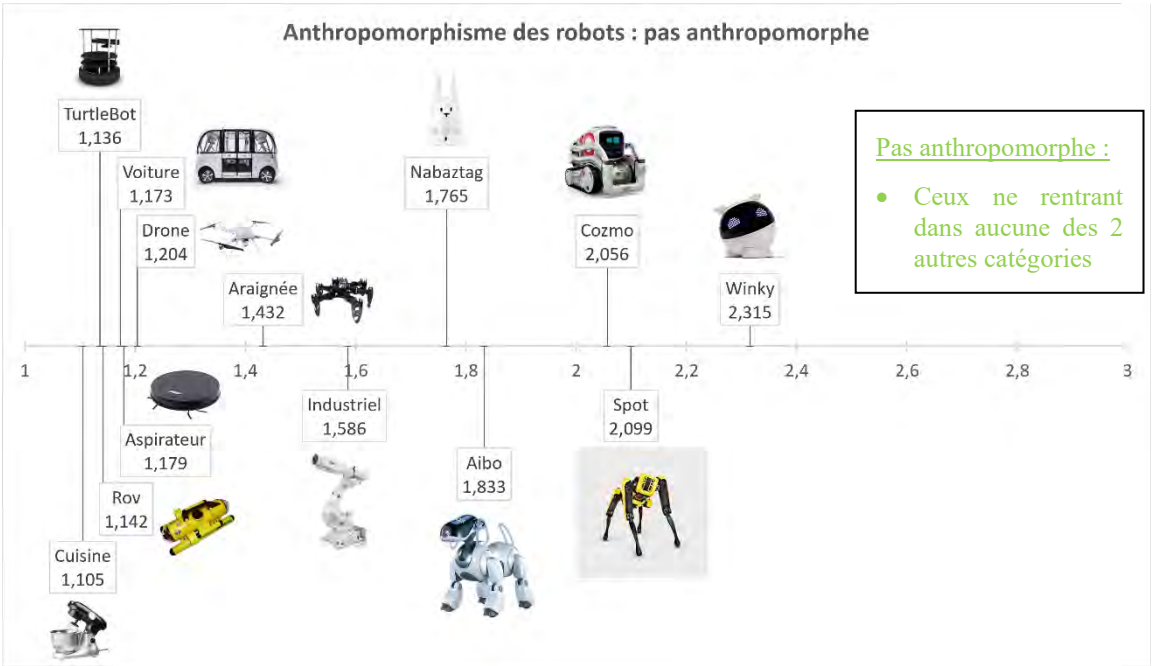


Figure 30 : Résultat de la pré-expérience sur l'anthropomorphisme des robots, groupe « pas anthropomorphe »

4.2.4 Discussion

Par rapport à l'hypothèse de départ H0, les robots humanoïdes ont effectivement obtenu les meilleurs résultats et sont, par conséquent, les robots les plus anthropomorphes. Le robot Pepper fait partie de ce groupe, ce qui permet par la suite de le considérer comme robot anthropomorphe par rapport, par exemple, au robot YuMi qui est en dernière position du groupe moyennement anthropomorphe.

4.3 Expérience : acceptabilité de la programmation par démonstration et impact de l'anthropomorphisme

4.3.1 Objectif

Après le développement d'algorithmes d'apprentissage par démonstration, on peut s'interroger sur la question suivante : « Est-ce que la programmation par démonstration est acceptée par les utilisateurs ? » En effet, ce type de programmation est développé pour être utilisé par des utilisateurs non-expert en programmation ni en robotique. Néanmoins, il est nécessaire que cette nouvelle programmation soit comprise et acceptée par les utilisateurs. La présente expérience vise à évaluer l'acceptabilité de la programmation par démonstration et à étudier l'impact de l'anthropomorphisme sur cette celle-ci.

L'algorithme d'apprentissage par démonstration, déjà évalué dans le Chapitre 3, l'expérience présentée ici se n'appuiera sur aucun algorithme d'apprentissage par démonstration. La démonstration effectuée par l'utilisateur sera reproduite telle quelle par le robot lors de la reproduction du mouvement. Ce choix est privilégié dans le but de ne pas surcharger la formation des utilisateurs novices ni d'influencer la performance de reproduction du mouvement. En effet, celle-ci est dépendante de l'algorithme d'apprentissage et des capacités intrinsèques du robot employé.

L'acceptabilité de la programmation par démonstration sera évaluée dans un contexte où un utilisateur non-expert collabore avec un cobot pour réaliser une certaine tâche. En vue d'étendre la portée des résultats à un contexte industriel, les participants devront se rapprocher le plus possible des utilisateurs finaux des cobots et la tâche à réaliser devra être réaliste par rapport à ce qui se fait concrètement dans les industries (Chernova & Thomaz, 2014).

Pour mener à bien cette expérience, deux facteurs expérimentaux (variables indépendantes) sont sélectionnés : le type de programmation et le type de cobot. Le **type de programmation** est soit la « programmation par démonstration » (PbD), soit une « programmation par mouvements préprogrammés » (MPp). La programmation par démonstration nécessite de réaliser une démonstration kinesthésique. L'utilisateur manipule le bras du robot afin de lui apprendre le mouvement à réaliser et peut ensuite rejouer le mouvement appris. Dans le cas des mouvements préprogrammés, le robot connaît déjà des mouvements et l'utilisateur peut les démarrer. Deux **types de cobots** sont utilisés pour l'expérience : un robot de type industriel, YuMi, et un robot humanoïde, Pepper.

Les trois hypothèses de recherche sont les suivantes :

H1 : « L'acceptabilité de la programmation sera plus importante lorsque qu'elle se fait par démonstration plutôt que par des mouvements préprogrammés. » ;

H2 : « Les cobots ont davantage de chance d'être acceptés quand ils sont anthropomorphes. » ;

H3 : « L'anthropomorphisme améliore l'acceptabilité de la programmation par démonstration d'un cobot. ».

La mise en place et l'exécution de ce type d'expérience ainsi que l'analyse des résultats nécessitent une bonne préparation en amont. (Chernova & Thomaz, 2014) propose un guide pour la conception et l'évaluation d'algorithmes d'apprentissage par démonstration lors d'une expérience incluant une interaction humain-robot.

4.3.2 Méthode

4.3.2.1 Participants

Selon (Stadler et al., 2014), l'expérience en programmation n'a d'impact ni sur le nombre de démonstrations effectuées ni sur la charge de travail. Peu importe si l'utilisateur maîtrise la programmation informatique, il lui faut juste réussir à manipuler le robot pour lui montrer le mouvement correct.

Dans le but de correspondre au contexte industriel visé par les cobots, les participants sélectionnés sont les utilisateurs finaux, ceux qui utiliseront les cobots et l'apprentissage par démonstration pour leur travail (Chernova & Thomaz, 2014). Avoir des participants déjà en poste est compliqué autant en termes de recrutement que de disponibilité.

Ainsi, les participants sont des étudiants issus de formations liées à l'industrie et par conséquent peut-être de potentiels futurs utilisateurs. Parmi ces étudiants, 16 ont participé à cette expérience. Trois différentes formations, toutes effectuées en alternance, sont représentées :

- Baccalauréat Professionnelle « Métiers de l'ÉLectricité et de ses Environnements Connectés » (MELEC) : 5 étudiants sont inscrits en Terminale ;
- Licence Professionnelle « Production et Gestion Industrielle en MÉCAnique » (PGI MECA) : 10 étudiants sont inscrits en Licence 3 ;
- Licence Professionnel « Systèmes Automatisées, Réseaux et Informatique Industrielle » (SARII) : 1 étudiant est inscrit en Licence 3.

Les participants recrutés ont au maximum le niveau Bac +3 ; leur profil professionnel est ainsi plus proche de celui des utilisateurs finaux. Les étudiants inscrits dans d'avantage d'années d'études supérieures correspondent plutôt à des profils d'ingénieur ou de cadre.

A l'exception d'un participant, tous mentionnent avoir déjà travaillé en entreprise. Tous les participants sont des hommes, majeurs et francophones. La moyenne d'âge est de 21 ans (écart-type : 3,828).

Chaque participant utilise un seul robot et un seul type de programmation. Cela permet d'éliminer le biais lié à l'apprentissage, à l'expérience et à la fatigue. Les participants sont donc divisés en quatre groupes indépendants, selon le croisement des facteurs expérimentaux type de programmation et type de cobot (Figure 31). Chaque combinaison des facteurs expérimentaux est testée par 4 participants.

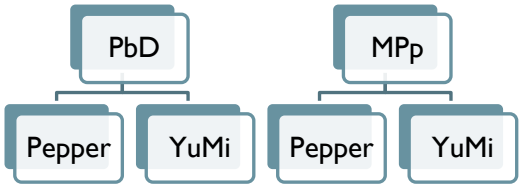


Figure 31 : Création des groupes en fonction des facteurs expérimentaux type de programmation et type de cobot

PbD : programmation par démonstration, MPp : programmation avec des mouvements préprogrammés

Le profil des participants est également analysé par rapport à leur connaissance et leur utilisation de la programmation informatique et des robots. Les participants doivent renseigner leur niveau de connaissance en programmation informatique et en robotique (Figure 32 et Figure 33).

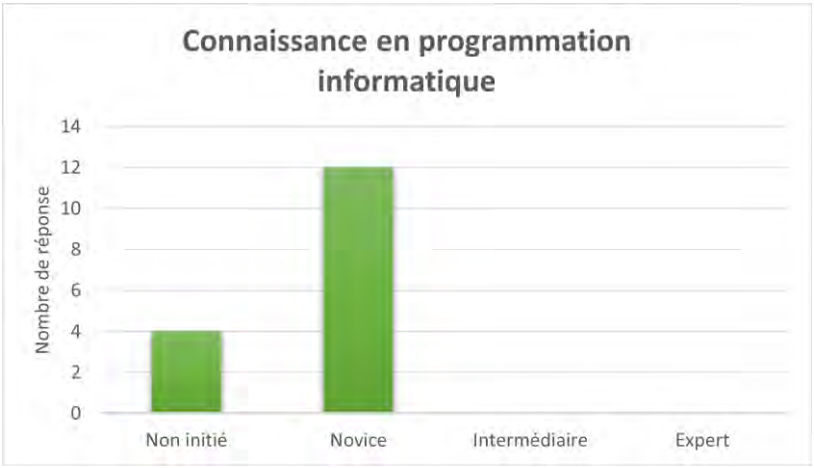


Figure 32 : Connaissance en programmation informatique des participants

Quatre réponses possibles : expert, intermédiaire, novice, non-initié

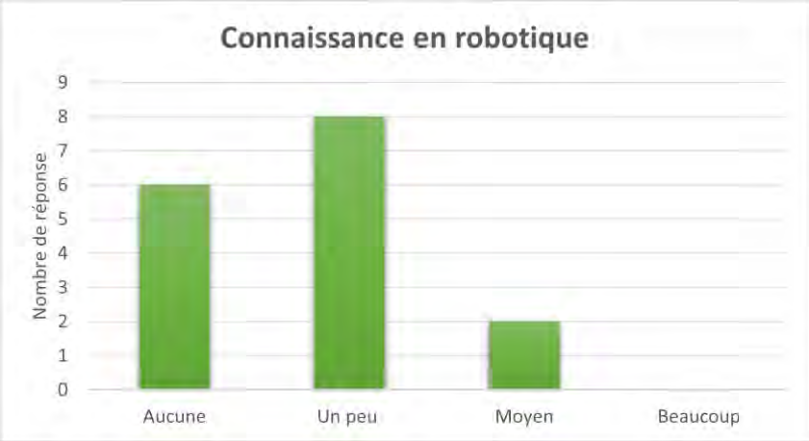


Figure 33 : Connaissance en robotique des participants

Quatre réponses possibles : beaucoup, moyen, un peu, aucune

Les résultats montrent que 75% d'entre eux estiment qu'ils sont novices en programmation informatique et les 25% restant se disent non initié. La moitié des participants jugent avoir peu de connaissance en robotique. 13% pensent avoir un niveau moyen et 38%, aucune connaissance.

Pour connaître l'expérience des participants avec les robots, deux questions leur sont posées : « Avez-vous déjà rencontré un robot ? » (Figure 34) et « Avez-vous déjà utilisé un robot ? » (Figure 35).

Figure 34 : Réponses des participants à la question « Avez-vous déjà rencontré un robot ? »

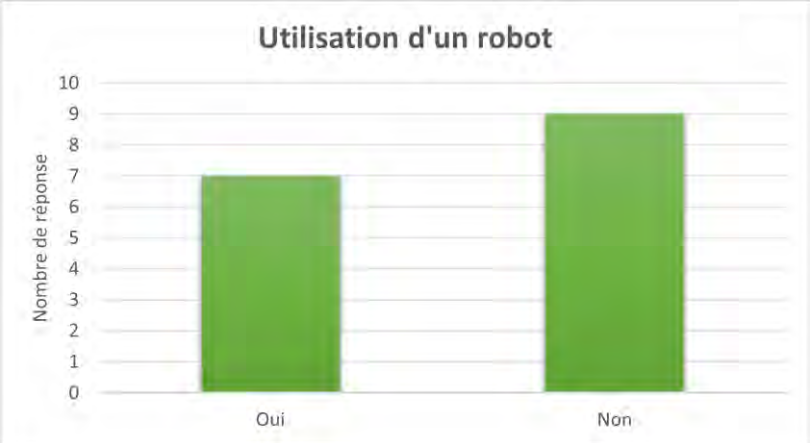


Figure 35 : Réponses des participants à la question « Avez-vous déjà utilisé un robot ? »

75% des participants ont déjà rencontré un robot et 56% n'ont jamais utilisé de robot.

Des participants ayant une grande expérience avec les robots pourraient influencer les résultats au niveau de l'acceptabilité car déjà familiers avec ces derniers. Dans notre cas, les résultats peuvent être considérés comme cohérents puisque les participants sont formés par le biais de l'alternance, ce qui augmente d'autant plus la proportion de personnes ayant déjà rencontré ou utilisé un robot. L'autre raison donnée par les participants est la rencontre de robot(s) lors de visite de musée ou d'opérations portes ouvertes d'écoles.

Il est à noter que le terme de robot n'a volontairement pas été défini au préalable des questions ci-dessus. Ainsi les participants doivent répondre selon leur propre définition du terme robot.

La dernière question posée avant le début de l'expérience porte sur leur ressenti par rapport à l'expérience. Cinq adjectifs leur sont proposés : impatient, content, neutre, stressé et anxieux. Les participants peuvent choisir plusieurs adjectifs et même en ajouter d'autres. Sur la Figure 36, les résultats montrent que 42% étaient contents et la même proportion d'entre eux étaient neutres. Aucun participant n'a estimé être anxieux. 11% des participants s'estimaient impatients et 5% se sentaient stressés.

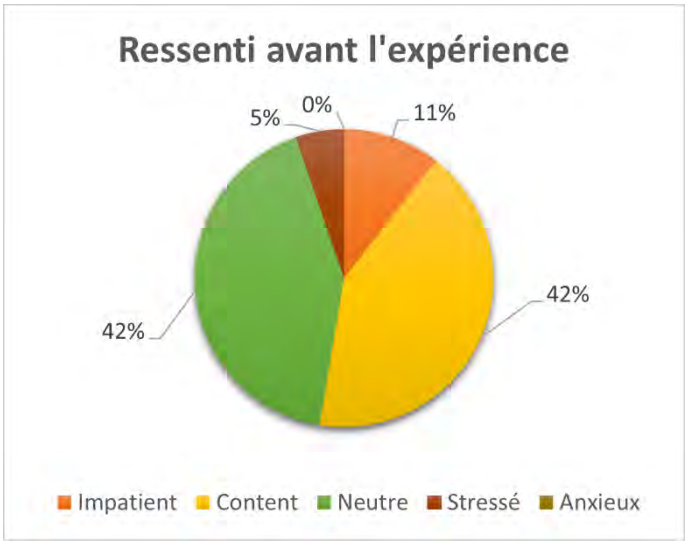


Figure 36 : Ressenti des participants avant l'expérience

Cinq adjectifs proposés (impatient, content, neutre, stressé, anxieux) plus la possibilité d'ajouter d'autres adjectifs

4.3.2.2 Matériel

Le matériel utilisé pour l'expérience se compose de deux questionnaires, un formulaire de consentement, des vidéos des tâches à faire réaliser par le robot, deux ordinateurs, deux robots et deux salles d'un centre de recherche.

Le premier questionnaire, complété avant l'expérience, porte sur le profil du participant (Annexe 7). Le second questionnaire, à la fin de l'expérience, contient les questions qui permettent d'évaluer l'acceptabilité (Annexe 8).

Un formulaire de consentement est présenté, complété puis signé avant de démarrer l'expérience (Annexe 5). Pour les mesures et questionnaires, seul le numéro associé au participant apparaît. Ce numéro est reporté sur le formulaire de consentement.

Pour présenter les tâches à réaliser, des vidéos sont présentées aux participants (Figure 38 et Figure 37). La vidéo d'une tâche donnée est de durée identique pour chaque robot. Le point de vue de la caméra est choisi afin de bien visualiser le robot et l'ensemble de la scène.

Un premier ordinateur est utilisé pour le remplissage des questionnaires sous format numérique avec l'outil Google Forms. Le second ordinateur est utilisé pour la programmation du robot.

Une salle est consacrée à l'accueil du participant et au remplissage des questionnaires. La seconde salle est dédiée à l'expérience avec l'un des deux robots disponibles : le robot Pepper ou le robot YuMi (Figure 17, page 55). Le robot non utilisé est caché à l'aide d'un drap afin que le participant ne se concentre que sur le robot qu'il va utiliser.

Selon le type de condition, le programme permet soit de faire de la « programmation par démonstration », soit la « programmation par mouvements préprogrammés ».

Les mesures prises en plus des questionnaires (décrites dans la Section 4.3.2.4) sont notées manuellement par l'instructeur dans un tableau.

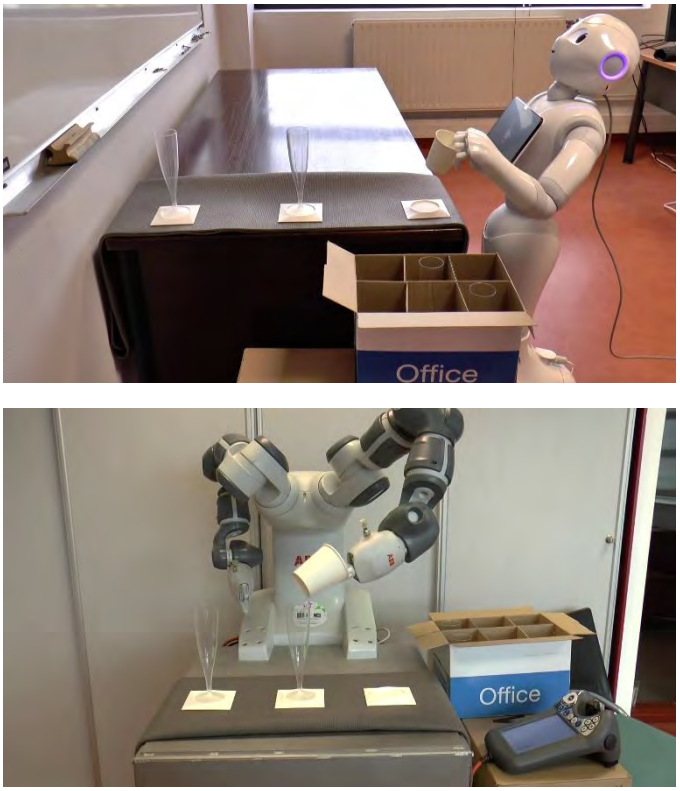


Figure 38 : Image de la vidéo de la tâche à réaliser lors de la formation

En haut avec le robot Pepper et en bas avec le robot YuMi

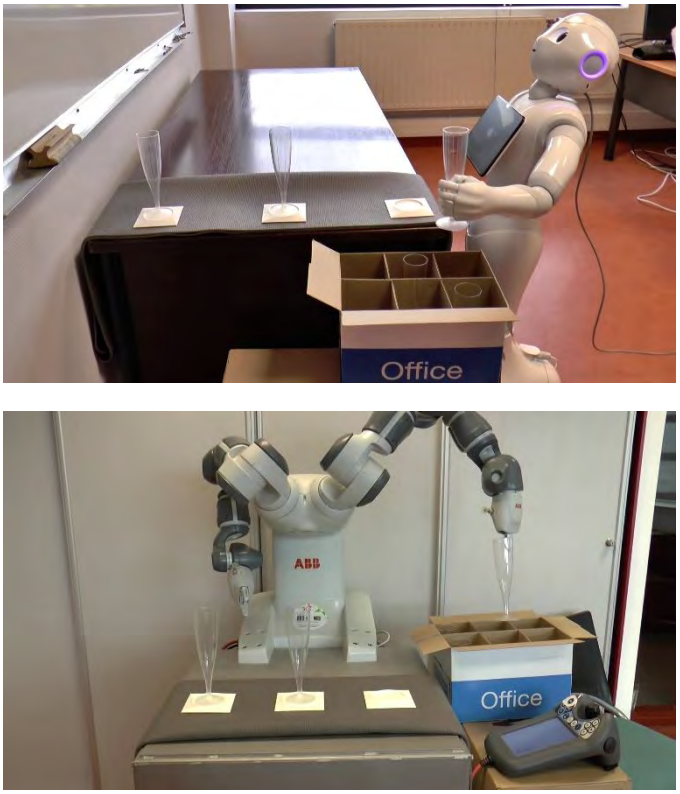


Figure 37 : Image de la vidéo de la tâche à réaliser lors l'expérience principale

En haut avec le robot Pepper et en bas avec le robot YuMi

La mise en scène et les tâches choisies pour l'expérience correspondant le plus possible à une situation réelle d'entreprise industrielle. La tâche à réaliser lors de l'expérience vise à reconstituer au maximum le processus de palettisation[8] industrielle. A dessein, un convoyeur[9] est reproduit pour le remplissage d'un carton (Figure 37). Le robot doit remplir le carton avec un verre. Pour rendre le convoyeur plus réaliste, des supports sont fabriqués afin d'apporter les verres. Ces supports, modélisés à l'aide d'un logiciel de Dessin Assisté par Ordinateur (DAO), sont obtenus grâce à une imprimante 3D (Figure 39).

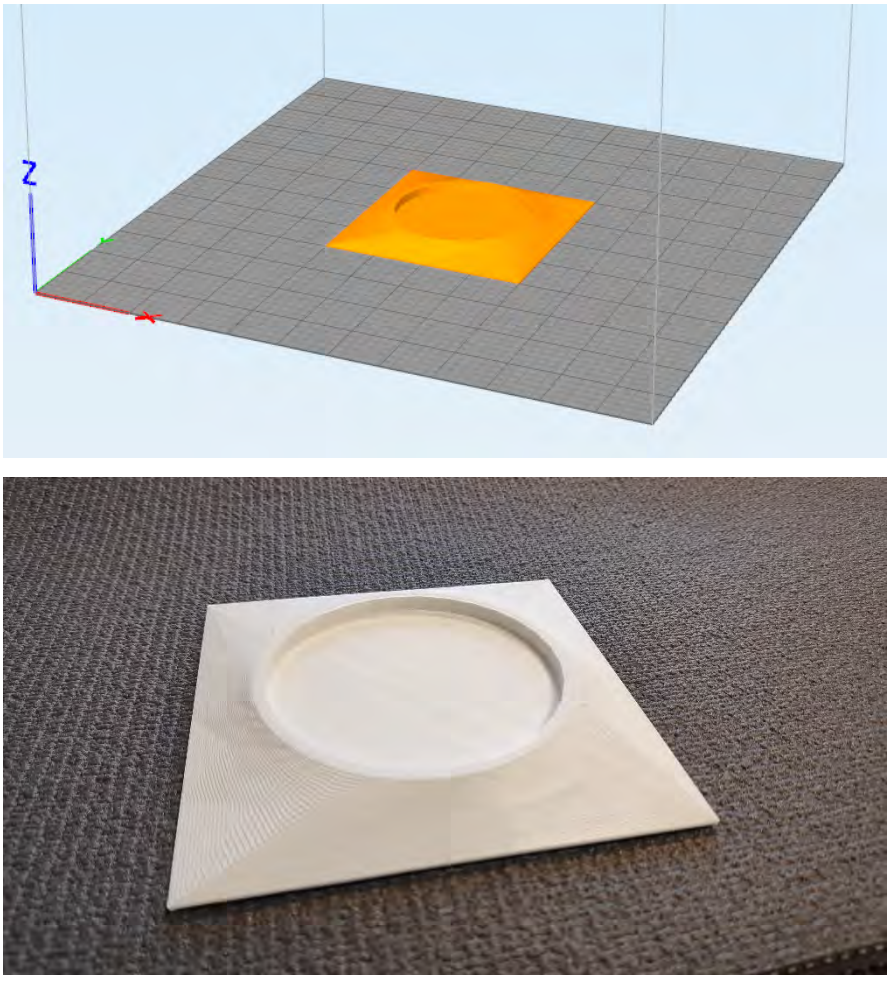


Figure 39 : Support pour verre

De haut en bas : modèle numérique, support imprimé en 3D

[8] Définition du dictionnaire Le Robert : « Action de palettiser (une marchandise), d'organiser par l'emploi de palettes. »
[9] Définition du dictionnaire Le Robert : « Transporteur automatique de marchandise. *Tapis roulant servant de convoyeur*. »

L'interface utilisateur pour programmer le robot est développée à l'aide de la bibliothèque pour interface graphique (GUI) multi-plateforme wxWidget. L'interface graphique est simple d'utilisation et intuitive (Figure 40).

Figure 40 : Interface utilisateur

(a) : aucun robot connecté ; (b) : exemple de message ; (c) : condition programmation par démonstration ; (d) : condition programmation avec mouvements préprogrammés ; (e) : fenêtre enregistrer ; (f) : fenêtre bouger

L'utilisateur est guidé par des messages (Figure 40b) et l'ordonnancement des boutons permet à l'utilisateur de ne pas se tromper dans l'ordre des actions à effectuer. Cet ordonnancement est indispensable pour le bon déroulement de l'expérience puisque le robot se trouve parfois dans un mode passif – l'utilisateur peut manipuler le robot – et parfois dans un mode actif – le robot bouge de lui-même. Quand l'utilisateur souhaite faire une démonstration, le robot doit se trouver dans le bon mode et inversement lors de la reproduction du mouvement. Cette organisation permet d'éviter d'endommager le robot lors des démonstrations.

Cette interface permet aussi simplement de connecter un nouveau robot et de changer de robot sans avoir besoin de la redémarrer.

Après le lancement de l'interface, l'utilisateur doit connecter un robot (Figure 40a). Cette étape est réalisée avant l'arrivée du participant. Ensuite, le robot doit être dans un état connecté avant de pouvoir l'utiliser. Dans le cas du robot Pepper, les packages ROS sont démarrés (voir Section 3.1.3) et dans le cas du robot YuMi, divers états d'initialisation sont effectués ainsi que le démarrage de la machine à états (voir Section 3.1.4).

Le bouton « Position de départ » fait bouger le robot dans sa position de départ. Par défaut, cette position est atteinte lors de la connexion du robot.

Sur la gauche de l'interface est affichée la liste des mouvements que le robot connaît. Dans la condition « programmation par démonstration », la liste est vide car l'utilisateur doit apprendre au robot les mouvements à effectuer (Figure 40c). Avec la condition « programmation par mouvements préprogrammés », deux mouvements sont disponibles, un pour la tâche lors de la formation et un second pour la tâche lors de l'expérience principale (Figure 40d). Ces différentes étapes ainsi que les tâches à réaliser sont décrites dans la Section 4.3.2.3.

Ensuite, selon le type de programmation, un ou deux boutons sont disponibles.

Le premier bouton « Enregistrer » lance la procédure d'enregistrement d'une démonstration.
Une nouvelle fenêtre apparaît afin de permettre de renseigner le nom du mouvement qui va être montré ainsi qu'un bouton qui déclenche le démarrage de l'enregistrement du mouvement, terminé par un nouvel appui sur ce bouton (Figure 40e). Avec la condition « programmation par mouvements préprogrammés », ce bouton n'apparait pas (Figure 40d).
Le deuxième bouton « Bouger » offre la possibilité de faire bouger le robot selon un mouvement appris. Une nouvelle fenêtre s'ouvre, l'utilisateur choisit un mouvement puis le robot le réalise (Figure 40f). Ce bouton apparaît dans les deux types de programmation.
L'interface a été comprise par tous les participants qui n'ont pas eu de soucis lors de son utilisation en total autonomie.

4.3.2.3 Procédure expérimentale

Le cadrage précis du processus d'une expérience est essentiel à son bon déroulement et à la fiabilité des résultats. En effet, une expérience où l'utilisateur a mal compris ce qu'il devait faire ou qui a été mal formé va influencer sa performance et ses retours sur l'expérience. (Chernova & Thomaz, 2014) formule de nombreux conseils pour la préparation d'un protocole expérimental dans le cadre d'une étude sur l'apprentissage par démonstration.

Au regard de l'âge des participants, le tutoiement mutuel est proposé aux participants avant l'expérience. Ainsi, les textes de consignes prévus pour l'expérience utilisent donc la deuxième personne du singulier. Afin que les conditions expérimentales soient identiques pour chaque participant, tous les textes de consigne prononcés par l'instructeur sont écrits au préalable et lus, toujours par le même instructeur, au cours de chaque expérience (Annexe 6).

La procédure développée pour cette expérience se base sur les travaux de (Weiss et al., 2009). Elle se divise en quatre étapes : le briefing, la formation, l'expérience et le questionnaire (Figure 41).

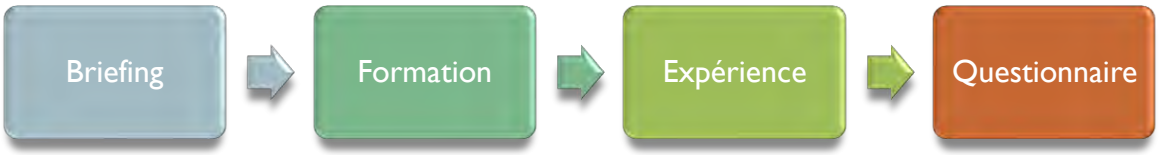


Figure 41 : Procédure expérimentale

La phase de **briefing** se déroule dans une salle sans la présence du robot qui va être utilisé. L'objectif est multiple : préparer le participant à l'expérience, lui expliquer la procédure de l'expérience et le rassurer.

Pour commencer, le formulaire de consentement est présenté à chaque participant en deux exemplaires, dont un qu'il conservera (Annexe 5). Le texte prononcé par l'instructeur est le suivant :

> « *Voici un formulaire de consentement avant ta participation.*
> *Tu n'es pas obligé de faire l'expérience. Tu as choisi de la faire de plein gré.*
> *Tu peux arrêter l'expérience à tout moment.*
> *Dans le formulaire, il est indiqué que les risques éventuels liés à l'expérience sont des chocs. Les risques sont les mêmes que pour toute manipulation d'objet du quotidien. Par exemple, se coincer les doigts lorsque l'on ferme une porte ou avec le capot d'un ordinateur portable.*
> *Merci de lire le formulaire attentivement et de signer en bas de la page sur les deux exemplaires. Un exemplaire est pour toi et le second pour moi.* »

Le formulaire, dûment rempli, est signé par le participant. Il est ensuite amené à compléter un questionnaire socio-démographique, sous forme numérique à l'aide de l'ordinateur prévu à cet effet (Annexe 7). Ensuite, l'instructeur décrit les étapes de l'expérience avec ces mots :

> « *Tu vas interagir avec un cobot, c'est-à-dire un robot collaboratif, afin de réaliser une tâche.*
> *Dans un premier temps, une formation te sera proposée afin de comprendre le fonctionnement du robot ainsi que l'interface de programmation. La tâche à réaliser te sera présentée à l'aide d'une vidéo. Tu pourras solliciter mon aide et poser des questions lors de cette phase.*
> *La deuxième phase est consacrée à l'expérience. Tu devras réaliser une autre tâche en autonomie. Elle te sera également présentée avec une vidéo.*
> *Un second questionnaire clôturera l'expérience.*
> *Un temps est réservé à la fin de l'expérience si tu souhaites discuter. Est-ce que tout est clair ?* »

L'étape suivante est la **formation** où l'instructeur décrit le fonctionnement du robot et de l'interface de programmation avant la réalisation d'une tâche simple qui valide la bonne compréhension du participant (Figure 42). Dans la condition « programmation par démonstration », le participant est formé, en plus, à la manipulation du robot avant de réaliser la tâche simple. L'objectif de cette phase est de rassurer le participant, de le mettre en confiance et de le laisser manipuler et découvrir le robot pour finir sur une tâche pratique. Les connaissances des participants sont ainsi normalisées afin d'être capable de réaliser l'expérience principale.

La tâche à réaliser lors de la formation consiste à prendre, soulever et reposer un gobelet (Figure 38). La tâche est choisie de telle sorte que sa réalisation soit simple et équitable entre les robots Pepper et YuMi. La formation prend fin lorsque le participant a réussi la tâche ou estime avoir suffisamment compris.

Figure 42 : Configuration de la salle pour la formation et l'expérience

De haut en bas : avec le robot Pepper, avec le robot YuMi

Le texte de consigne de l'instructeur est le suivant, avec en (bleu) le texte de la condition « programmation par démonstration » et en [vert] pour la « programmation par mouvements préprogrammés » :

> *« La formation commence maintenant. Voici le robot que tu vas utiliser ainsi que son interface de programmation.*
> *Tu vas programmer le robot [afin de lui faire réaliser une tâche] (en lui montrant physiquement comment réaliser une tâche). L'objectif est de [faire] (lui apprendre) la tâche « prendre_gobelet » (montrer la vidéo).*
> *Pour cela, (tu vas manipuler le bras du robot pour lui montrer le mouvement. Dans un premier temps,) je vais t'expliquer le fonctionnement de l'interface. (Ensuite, je te montrerai la manipulation du robot. Dans un dernier temps, tu devras réaliser la tâche « prendre_gobelet » avec mon aide.) »*

« L'interface te permet de positionner le robot dans sa position de départ, c'est-à-dire la position dans laquelle le robot se trouve actuellement.
Sur la gauche, tu as la liste des tâches que le robot a apprises. (Ici, le robot ne sait rien faire.)
Ensuite, tu as [1] (2) bouton(s) : (enregistrer et) bouger.
(Le bouton enregistrer te permet de faire une démonstration d'une tâche au robot. Tu peux choisir le nom que tu souhaites pour la tâche. Le nom ne peut contenir que des lettres, chiffres et tiret du bas. Si tu te trompes lors de la réalisation de la tâche, tu peux la refaire avec le même nom et le robot ne retiendra que la dernière.)
Le bouton bouger te permet de faire bouger le robot en fonction de la tâche que tu auras choisi.
Le robot ne dispose pas d'un système de vision qui permet de connaître la position des objets qu'il manipule. C'est pourquoi les objets devront toujours se trouver à la même position. Pense à positionner les objets avant de démarrer le mouvement d'une tâche. »

« Je vais te montrer comment manipuler le robot. La tâche 'test' permet de tester la manipulation du robot. Voici comment le robot peut bouger. Il faut faire attention à ne pas bouger trop rapidement le robot. Il ne faut pas non plus forcer sur ses articulations. A la fin du mouvement, il faut positionner le bras du robot vers sa position de départ. Afin d'ouvrir ou fermer la main (Pepper) / pince (YuMi) du robot, il faut dire 'gauche' (Pepper) / appuyer sur le bouton suivant (YuMi). Vas-y, entraine-toi à manipuler le robot. »

« Tu vas devoir maintenant programmer le robot afin qu'il réalise la tâche « prendre_gobelet » que tu as vu dans la vidéo. Tu peux revoir la vidéo quand tu le souhaites. Vas-y, je vais te guider quand tu auras besoin. »

La prochaine étape est l'**expérience** en elle-même, où le participant doit réaliser une autre tâche en autonomie (Figure 42).

La tâche à réaliser consiste à prendre un verre et le ranger dans un carton (Figure 37). Cette tâche est choisie car les deux robots peuvent l'effectuer sans que Pepper, robot le moins précis, ne soit trop désavantagé. Cette tâche est également réaliste car elle représente le déchargement de pièce provenant d'un convoyeur sur une palette contenant un carton.

Le participant est en totale autonomie. L'instructeur ne peut intervenir que si un problème logiciel ou matériel important se déclare.

Le texte de l'instructeur est le suivant :

> « *La formation est maintenant terminée. L'expérience commence. Tu vas devoir réaliser une nouvelle tâche en autonomie. Voici la vidéo de la tâche à réaliser. Comme lors de la formation, tu peux revoir la vidéo de la tâche à réaliser quand tu le souhaites. Le verre à utiliser se trouve ici. Je resterai dans la salle afin de m'assurer du bon déroulement de l'expérience. Je ne pourrais plus répondre à tes questions ni t'aider, exception si le robot ou le programme plante. Lorsque tu estimes avoir terminé, merci de me le signaler.* »

La dernière étape, celle du **questionnaire**, se déroule à l'aide du même ordinateur que celui utilisé lors de l'étape du briefing.

Une fois le document complété, le participant dispose d'un temps pour poser toutes les questions qu'il souhaite. A ce moment-là, les participants posent souvent des questions sur l'objectif final de l'expérience, comment le programme fonctionne, quel est mon parcours. De nombreux commentaires et retours très positifs des participants sont également constatés à ce moment.

4.3.2.4 Questionnaires et mesures

Le rôle de l'enseignant lors de l'apprentissage par démonstration est très important (Calinon & Billard, 2007b). En plus du design et de la procédure de l'expérience où l'enseignant est au cœur du système, le type d'évaluation doit être défini en fonction des besoins.

De nombreuses manières permettent d'évaluer une interaction humain-robot (Calinon & Billard, 2007b ; Steinfeld et al., 2006). Ces dernières peuvent se classer en deux catégories : les mesures subjectives et les mesures objectives (Chernova & Thomaz, 2014).

Les mesures subjectives sont généralement obtenues en interrogeant les participants soit à l'oral, soit à l'aide de questionnaires avec généralement une échelle de Likert. Pour éviter une surcharge cognitive et obtenir des résultats fiables au questionnaire, ce dernier ne doit pas être trop long.

Les mesures objectives sont liées à des mesures automatiques ou liées à un barème bien défini afin que l'avis du participant ou de l'instructeur ne rentre pas en compte dans cette mesure. Une mesure de temps, un taux de succès ou le nombre de démonstrations effectuées sont des exemples de mesures objectives.

L'évaluation de l'acceptabilité de la programmation par démonstration et l'impact de l'anthropomorphisme sont effectués à l'aide d'un questionnaire créé à cet effet. En effet, il n'existe aucun questionnaire dans la littérature pour ce type d'évaluation.

Le questionnaire développé se base sur les questionnaires existants qui évaluent les interactions humain-robot ou humain-système (voir Section 4.1). Le Tableau 17 répertorie les questions constituant ce questionnaire.

Tableau 17 : Questionnaire pour évaluer l'acceptabilité de la programmation par démonstration et l'impact de l'anthropomorphisme

*[1] : Echelle de confiance (Charalambous et al., 2016) ; [2] : TAM (Davis, 1989) ; [3] : NASA-TLX (Hart & Staveland, 1988) ; [4] : RAS / NARS (Nomura et al., 2006b, 2006a) ; [5] : QUEAD (Schmidtler et al., 2017) ; [6] : UTAUT (Venkatesh et al., 2003) ; * : aucune référence ; ~ : inspiré de*

Questionnaire sur le dispositif	
Modalité	**Questions**
Charge de travail *	CT1 : « Pour réaliser l'expérience, quel degré d'activité physique était nécessaire ? » [3] CT2 : « Pour réaliser l'expérience, quel degré d'activité mentale était nécessaire ? » [3]
Satisfaction *	S1 : « J'ai aimé utiliser ce dispositif. » [6] S2 : « La tâche du robot a été réalisée correctement. » [3]
Utilité perçue [2]	UP1 : « Ce dispositif me serait très utile pour mon travail. » [2, 5] UP2 : « Ce dispositif diminuerait ma fatigue dans mon travail. » * UP3 : « Ce dispositif diminuerait mes erreurs dans mon travail. » * UP4 : « Ce dispositif me permettrait d'accomplir mes tâches plus rapidement dans mon travail. » [2, 5]
Facilité d'utilisation perçue [5]	FUP1 : « Ce dispositif est facile à apprendre. » [5, 6] FUP2 : « Ce dispositif est intuitif. » [5] FUP3 : « Ce dispositif est facile à utiliser. » [5, 6]
Questionnaire sur le robot	
Modalité	**Questions**
Attitude [5]	AT1 : « J'aime collaborer avec ce robot. » [5] AT2 : « L'utilisation de ce robot est une bonne idée. » [5] AT3 : « Je souhaiterais utiliser ce robot dans mon travail. » [5]
Confiance [1]	C1 : « Je me suis senti(e) en sécurité lors de l'interaction avec le robot. » [1] C2 : « La taille du robot ne m'a pas intimidé. » [1] C3 : « Je sentais que je pouvais compter sur le robot pour faire ce qu'il avait à faire. » [1]
Anxiété [4]	AX1 : « Je sens que le robot va me remplacer au travail. » [~4] AX2 : « Le travail avec un robot me stresse. » [3] AX3 : « Le travail avec un robot me rend anxieux. » [5]

L'Annexe 8 présente le questionnaire rempli par les participants avec les questions regroupées par thématique afin de faciliter la complétion des participants.

Le questionnaire développé est composé de questions portant sur le dispositif (programmation et interface) et sur le robot. Toutes les questions sont réunies selon sept modalités : la « charge de travail », la « satisfaction », l'« utilité perçue », l'« attitude », la « confiance » et l'« anxiété ».

La combinaison de questions selon une modalité sera évaluée à l'aide de l'Alpha de Cronbach. Les questions n'obtenant pas une valeur d'Alpha de Cronbach convenable seront évaluées indépendamment.

Parmi les questions, certaines sont à la forme négative et toutes se basent sur une échelle de Likert sur 5.

Les questions CT1 et CT2 comportent les items suivants : « Très faible », « Faible », « Normal », « Élevé » et « Très élevé ». Toutes les autres questions ont les items habituels : « Pas du tout d'accord », « Pas d'accord », « Neutre », « D'accord », « Tout à faire d'accord ».

Toutes les modalités sont évaluées pour chaque hypothèse. L'existence d'un effet principal du facteur type de programmation, obtenu à l'aide de l'ANOVA (*ANalysis Of VAriance*), est examinée pour vérifier l'hypothèse H1. L'éventuel effet principal du facteur type de cobot permet de tester l'hypothèse H2. L'effet d'interaction entre les deux facteurs éclaireront eux, la dernière hypothèse H3. Les effets d'interaction significatif seront analysés également avec un test t de Student.

En plus du questionnaire, des mesures objectives sont prises afin d'évaluer d'autres paramètres de l'expérience.

Lors de la formation et lors de l'expérience, six mesures sont effectuées : leur durée, le nombre de démonstration(s) (dans le cas de la programmation par démonstration), le nombre de reproduction(s) du mouvement, le nombre de visionnage(s) de la vidéo de la tâche à réaliser, le nombre de problème(s) liés au robot / programme et le taux de succès défini dans le Tableau 4, page 38.
Toutes ces mesures sont notées à la main par l'instructeur dans un tableau pendant la formation et pendant l'expérience. Ces mesures sont évaluées à l'aide d'un test t-Student.

Le questionnaire sur l'acceptabilité est complété par des questions sur l'intelligence du robot perçue et la ressemblance du robot avec un être humain. L'objectif est d'évaluer à quel point les utilisateurs de la condition programmation par démonstration estiment que le robot est « intelligent » au regard de la condition programmation par mouvements préprogrammés.
Les questions sur l'intelligence sont au nombre de quatre, avec des affirmations et une échelle de Likert sur 5, identique à celle utilisée dans le questionnaire sur l'acceptabilité.
Les affirmations sont les suivantes :

- I1 : « Le robot ressemblait à un humain. » ;
- I2 : « Le robot a appris comme un humain. » ;
- I3 : « Le robot pensait comme un humain. » ;
- I4 : « Le robot était intelligent. ».

Pour terminer le questionnaire, les participants sont invités à noter à quel point ils ont trouvé que l'expérience était « amusante » sur 5. Un champ est également à leur disposition s'ils souhaitent laisser un commentaire.

Concernant la première hypothèse H1 (« L'acceptabilité de la programmation sera plus importante lorsque qu'elle se fait par démonstration plutôt que par des mouvements préprogrammés. »), les participants de la condition PbD devraient être plus satisfaits, trouver une meilleure « utilité perçue », avoir une bonne « attitude », être en « confiance » et moins « anxieux » qu'avec la condition MPp. Pourtant, les résultats au niveau de la « charge de travail » et de la « facilité d'utilisation » pourraient être plus faibles dans la condition PbD. En effet, les utilisateurs de cette condition sont sensés réaliser une étape supplémentaire pour enseigner le mouvement au robot.

La deuxième hypothèse H2 (« Les cobots ont davantage de chance d'être acceptés quand ils sont anthropomorphes. ») serait vérifiée si le robot Pepper obtenait des résultats plus élevés sur le questionnaire portant sur le robot que le robot YuMi. Les résultats sur le questionnaire lié au dispositif ne devraient pas montrer de différence entre les robots au niveau des modalités « charge de travail » et « facilité d'utilisation », puisqu'elles sont principalement liées au type de programmation. Les modalités « satisfaction », « utilité perçue », « attitude », « confiance » devraient obtenir les scores les plus élevés avec le robot Pepper et la modalité « anxiété » devraient récoltés des scores plus faibles avec le robot Pepper.

La troisième hypothèse H3 (« L'anthropomorphisme améliore l'acceptabilité de la programmation par démonstration d'un cobot. ») promettrait une différence de scores d'acceptabilité entre la condition PbD et MPp d'autant plus importante avec le robot Pepper qu'avec le robot YuMi.

4.3.3 Résultats

Avant d'analyser les résultats des participants, la cohérence des modalités choisies est vérifiée à l'aide de l'Alpha de Cronbach (Tableau 18).

Tableau 18 : Alpha de Cronbach sur le questionnaire sur l'acceptabilité

Modalité	α	Questions	α (si l'item est retiré)
Charge de travail	0,552	CT1 CT2	
Satisfaction	0,567	S1 S2	
Utilité perçue	**0,880**	UP1 UP2 UP3 UP4	
Facilité d'utilisation perçue	**0,831**	FUP1 FUP2 FUP3	
Attitude	0,559	AT1 AT2 AT3	**0.737** 0.324 0.155
Confiance	0,282	C1 C2 C3	-0,229 0,254 0,636
Anxiété	**0,707**	AX1 AX2 AX3	

Les modalités « utilité perçue », « facilité d'utilisation perçue » et « anxiété » ont une valeur d'Alpha comprise entre 0,7 et 0,95, ce qui valide la cohérence de leurs items. Les modalités « charge de travail » et « satisfaction » n'obtiennent pas une valeur d'Alpha suffisante ce qui amène à analyser leurs deux items respectifs indépendamment. L'Alpha de la modalité « confiance » dispose de 3 items indépendants. Au niveau de l'« attitude », si on enlève la question AT1, la modalité devient cohérente avec une valeur d'Alpha de 0,737.

Les hypothèses sont vérifiées à l'aide du calcul de l'ANOVA sur les résultats obtenus pour chaque modalité / items indépendants. L'effet du facteur type de programmation teste l'hypothèse H1 (Tableau 19) et celui du facteur type de cobot, l'hypothèse H2 (Tableau 20). L'hypothèse H3 peut être vérifiée avec l'interaction entre les deux facteurs, type de programmation (PbD et MPp) et type de cobot (Pepper et YuMi) (Tableau 21).

Les résultats présentés dans le Tableau 19 montrent des effets principaux significatifs du type de programmation sur les items CT1 et S1 ($p < .05$) et une tendance sur l'item CT2 ($p < .1$). Toutes les autres modalités ou items n'ont pas de données significatives ($p > .05$).

Tableau 19 : ANOVA, effet du facteur type de programmation

*Moyenne et écart type pour chaque modalité/item. - : p < .1 ; * : p < .05 ; ** : p < .01 ; *** : p < .001*

	CT1	**CT2**	**S1**	**S2**	**UP**	**FUP**
MPp	1,000 (0,000)*	1,857 (0,835)-	3,875 (0,835)*	4,250 (1,035)	3,531 (0,860)	4,667 (0,436)
PbD	1,500 (0,535)	2,750 (0,707)	4,625 (0,518)	4,000 (0,926)	3,188 (0,952)	4,583 (0,496)
	AT1	**AT23**	**C1**	**C2**	**C3**	**AX**
MPp	3,875 (0,991)	4,000 (1,035)	4,500 (0,535)	4,750 (0,463)	4,125 (0,991)	1,583 (0,684)
PbD	4,375 (0,744)	3,688 (0,651)	4,500 (0,535)	4,875 (0,354)	4,125 (0,641)	1,667 (0,756)

La « charge de travail » physique, item CT1, est plus élevée dans le cas de la « programmation par démonstration » ($F(1/12) = 6{,}000, p < .05$).

Les résultats de l'item CT2, « charge de travail » mentale, ont tendance à être plus élevés avec la « programmation par démonstration » ($F(1/12) = 4{,}742, p = .0501$). Ces premiers résultats peuvent s'expliquer par le fait que les participants doivent apprendre au robot comment faire le mouvement dans la condition PbD.

Le deuxième effet du type de programmation s'observe sur l'item S1 (« J'ai aimé utiliser ce dispositif. »). Les participants ont préféré la « programmation par démonstration » par rapport à la « programmation par mouvements préprogrammés » ($F(1/12) = 6{,}000, p < .05$).

L'impact du type de cobot est illustré par le Tableau 20 montrant plusieurs effets principaux significatifs.

Tableau 20 : ANOVA, effet du facteur type de cobot

*Moyenne et écart type pour chaque modalité/item. Pep. : Pepper ; Yu. : YuMi ; - : p < .1 ; * : p < .05 ; ** : p < .01 ; *** : p < .001*

	CT1	CT2	S1	S2	UP	FUP
Pep.	1,250 (0,463)	2,250 (0,707)	3,875 (0,641)*	3,375 (0,744)***	2,844 (0,823)*	4,417 (0,556)-
Yu.	1,250 (0,463)	2,375 (1,061)	4,625 (0,744)	4,875 (0,354)	3,875 (0,655)	4,833 (0,178)
	AT1	**AT23**	**C1**	**C2**	**C3**	**AX**
Pep.	3,875 (0,835)	3,563 (0,678)	4,500 (0,535)	4,875 (0,354)	3,500 (0,535)***	1,667 (0,735)
Yu.	4,375 (0,916)	4,125 (0,954)	4,500 (0,535)	4,750 (0,463)	4,750 (0,463)	1,583 (0,707)

Au niveau de la modalité « satisfaction », les deux items indépendants ont un effet significatif selon le type de cobot. Le robot YuMi obtient un score plus élevé avec l'item S1 « J'ai aimé utiliser ce dispositif. » ($F(1/12) = 6{,}000, p < .05$) ainsi qu'avec l'item S2, lié à la bonne réalisation de la tâche ($F(1/12) = 24{,}000, p < .001$). Les participants sont donc plus satisfaits lors de l'utilisation du robot YuMi.

Les résultats de la modalité « utilité perçue » montre un effet significatif du type de cobot ($F(1/12) = 7{,}342, p < .05$). La projection des participants sur l'utilité du dispositif est plus forte avec le robot YuMi qu'avec le robot Pepper.

Au niveau de la « facilité d'utilisation perçue », le robot YuMi a tendance à être plus simple d'utilisation que le robot Pepper ($F(1/12) = 3{,}571, p = .083$).

Le dernier effet significatif du type de cobot s'observe sur l'item C3, lié à la confiance du participant dans le robot pour qu'il fasse ce qu'il doit effectuer ($F(1/12) = 30{,}000$, $p < .001$). Les participants ont plus confiance dans le robot YuMi que le robot Pepper.

Aucun n'autre effet significatif du type de cobot se distingue sur les autres modalités et items indépendants ($p > .05$).

Globalement, le robot Pepper obtient des résultats significatifs moins bons que le robot YuMi. Une possible explication peut venir du fait que les utilisateurs avaient plus de mal à utiliser le robot Pepper puisqu'étant moins précis.

La dernière hypothèse H3 est vérifiée avec les effets d'interaction entre les deux facteurs type de programmation et type de robot (Tableau 21).

Tableau 21 : ANOVA, effet d'interaction entre les facteurs type de programmation et type de cobot

*Moyenne et écart type pour chaque modalité/item. MPe : MPp Pepper ; MYu : MPp YuMi ; PPe : PbD Pepper ; PYu : PbD YuMi ; - : p < .1 ; * : p < .05 ; ** : p < .01 ; *** : p < .001*

	CT1	CT2	S1	S2	UP	FUP
MPe	1,000 (0,000)	2,000 (0,816)	3,500 (0,577)	3,500 (1,000)	2,875 (0,144)	4,500 (0,577)
MYu	1,000 (0,000)	1,750 (0,957)	4,250 (0,957)	5,000 (0,000)	4,188 (0,747)	4,833 (0,192)
PPe	1,500 (0,577)	2,500 (0,577)	4,250 (0,500)	3,250 (0,500)	2,813 (1,248)	4,333 (0,609)
PYu	1,500 (0,577)	3,000 (0,816)	5,000 (0,000)	4,750 (0,500)	3,563 (0,427)	4,833 (0,192)
	AT1	**AT23**	**C1**	**C2**	**C3**	**AX**
MPe	3,500 (1,000)	3,625 (0,750)	4,250 (0,500)-	4,750 (0,500)	3,250 (0,500)*	2,000 (0,720)*
MYu	4,250 (0,957)	4,375 (1,250)	4,750 (0,500)	4,750 (0,500)	5,000 (0,000)	1,167 (0,333)
PPe	4,250 (0,500)	3,500 (0,707)	4,750 (0,500)	5,000 (0,000)	3,750 (0,500)	1,333 (0,667)
PYu	4,500 (1,000)	3,875 (0,629)	4,250 (0,500)	4,750 (0,500)	4,500 (0,577)	2,000 (0,770)

Les résultats de l'item C1 « Je me suis senti(e) en sécurité lors de l'interaction avec le robot » montrent une tendance au niveau de la sécurité perçue. Le score est plus élevé avec le robot YuMi dans la condition MPp et avec le robot Pepper dans la condition PbD.

Les deux effets d'interaction significatifs se situent au niveau de l'item C3 (« Je sentais que je pouvais compter sur le robot pour faire ce qu'il avait à faire. ») et la modalité « anxiété » AX. Pour analyser les résultats, la Figure 43 et la Figure 44 représentent graphiquement l'interaction entre les facteurs type de programmation et type de cobot entre l'item C3 et la modalité AX respectivement. Chaque effet est étudié séparément.

Sur la Figure 43, l'item C3, portant sur la confiance en ce que réalise le robot, obtient un score plus élevé avec le robot YuMi qu'avec le robot Pepper dans la condition « programmation par mouvements préprogrammés » ($t(3) = -7{,}000$, $p < .01$). Il n'y a aucun effet simple significatif entre les robots dans la condition « programmation par démonstration » ($t(5{,}88) = -1{,}964$, $p > .05$).

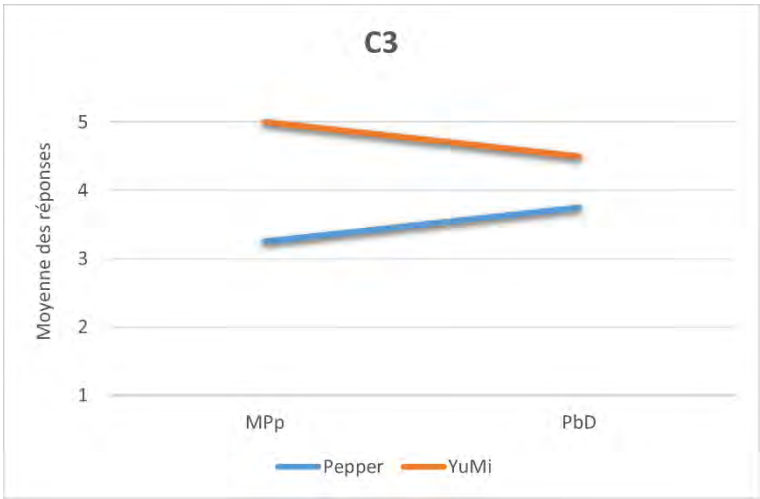


Figure 43 : Graphique de l'interaction entre le facteur type de programmation et le facteur type de cobot sur l'item C3 ($p < .05$)

Item C3 : « Je sentais que je pouvais compter sur le robot pour faire ce qu'il avait à faire. » ; type de programmation : MPp ou PbD ; type de cobot : Pepper ou YuMi ; réponse entre 1 (Pas du tout d'accord) et 7 (Tout à fait d'accord)

La Figure 44 montre un effet d'interaction croisé entre les deux facteurs étudiés. Cependant, il n'y a aucun effet simple significatif du type de robot autant dans la condition « programmation par mouvements préprogrammés » ($t(4{,}23) = 2{,}100$, $p > .05$) que dans la condition « programmation par démonstration » ($t(5{,}88) = -1{,}309, p > .05$).

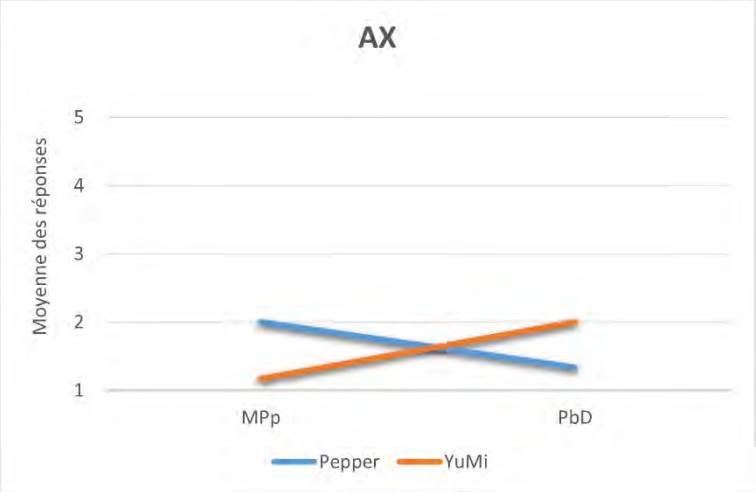


Figure 44 : Graphique de l'interaction entre le facteur type de programmation et le facteur type de cobot sur la modalité AX ($p < .05$)

Modalité AX : « anxiété » ; type de programmation : MPp ou PbD ; type de cobot : Pepper ou YuMi ; réponse entre 1 (Pas du tout d'accord) et 7 (Tout à fait d'accord)

Aucune autre donnée n'est significative pour les effets d'interaction entre facteur ($p > .05$).

Les résultats des mesures subjectives sont résumés dans l'Annexe 9. Seuls les résultats significatifs ($p > .05$) ou ayant un intérêt pour comprendre l'utilisation de la « programmation par démonstration » sont développés.

Selon la condition type de programmation, la durée de la formation et celle de l'expérience sont différentes de manière significative. En moyenne, la durée de la formation est de 1 minutes 42 pour la « programmation par mouvements préprogrammés » contre 6 minutes 42 pour la « programmation par démonstration » ($t(7{,}52) = -4{,}709, p < .01$). La durée de l'expérience obtient des moyennes similaires ($t(7{,}17) = -2{,}617, p < .05$). Les différences de durées entre les robots sont légèrement défavorables pour le robot Pepper mais non significatives ($p > .05$).

Le nombre moyen de démonstrations dans le cas de la condition « programmation par démonstration » n'obtient pas de différence significative entre la formation et l'expérience ($p > .05$). Ce résultat prouve que la formation était suffisante pour comprendre le fonctionnement de la « programmation par démonstration ». Une tendance s'observe au niveau du type de robot. Avec le robot Pepper, les participants ont réalisé plus de démonstration qu'avec le robot YuMi ($t(5{,}80) = 2{,}333, p = .060$).

Pour le nombre de reproduction du mouvement, aucun effet simple significatif n'est mesuré par rapport au type de cobot ni au moment – formation ou expérience ($p > .05$). Lors de la formation, les participants ont tendance à démarrer plus souvent le mouvement dans la condition « programmation par démonstration » ($t(7) = -2{,}198, p = .064$).

Lors de la formation et lors de l'expérience, les participants pouvaient visionner la vidéo de la tâche à réaliser autant de fois que nécessaire. L'effet simple significatif sur cette mesure est obtenu avec un score plus élevé dans la condition « programmation par démonstration » lors de la formation ($t(7) = -3{,}416, p < .05$). En effet, lors de la formation, les participants de la condition PbD ont plus de consignes entre le moment où la vidéo de la tâche à réaliser leur est présentée et l'instant où ils doivent réaliser la tâche.

Les scores de taux de succès ne montrent aucun effet significatif ($p > .05$) selon le type de programmation, le type de robot ou le moment (formation/expérience). Les scores sont légèrement plus élevés avec le robot YuMi, robot plus précis que le robot Pepper.

Les questions liées à l'intelligence sont étudiées indépendamment puisque l'Alpha de Cronbach est de 0,216. Même en retirant un des items, la valeur de l'Alpha n'est pas améliorée.

Le Tableau 22 regroupe les moyennes obtenues pour chaque item lié à l'« intelligence ». Les résultats sont présentés selon chaque combinaison type de programmation et type de cobot. Aucun effet principal n'est relevé pour le facteur type de programmation et le facteur type de cobot ($p > .05$).

Tableau 22 : ANOVA sur les items liés à l'intelligence

*Moyenne et écart type pour chaque item. - : p-value significative à 0.1 ; * : p-value significative à 0.05 ; ** : p-value significative à 0.01 ; *** : p-value significative à 0.001*

	I1	I2	I3	I4
MPp Pepper	3,500 (0,577)	3,250 (0,957)*	2,500 (0,577)**	3,000 (1,155)
MPp YuMi	3,000 (0,816)	3,750 (0,500)	2,750 (0,957)	3,750 (0,500)
PbD Pepper	3,000 (0,816)	4,250 (0,500)	1,500 (1,000)	3,000 (0,816)
PbD YuMi	3,250 (0,957)	4,250 (0,500)	1,250 (0,500)	2,750 (1,258)

Les items I1 (« Le robot ressemblait à un humain. ») et I4 (« Le robot était intelligent. ») n'ont aucun effet d'interaction significatif ($p > .05$). Les deux autres items I2 et I3 ont un effet d'interaction significatif, illustré par les Figure 45 et Figure 46.

Comme le montre la Figure 45, l'item I2 (« Le robot a appris comme un humain. ») obtient un score plus élevé avec la condition PbD qu'avec la condition MPp mais cette différence n'est pas significative ($p > .05$). Il n'y a aucun effet du type de robot dans la condition PbD ($p > .05$).

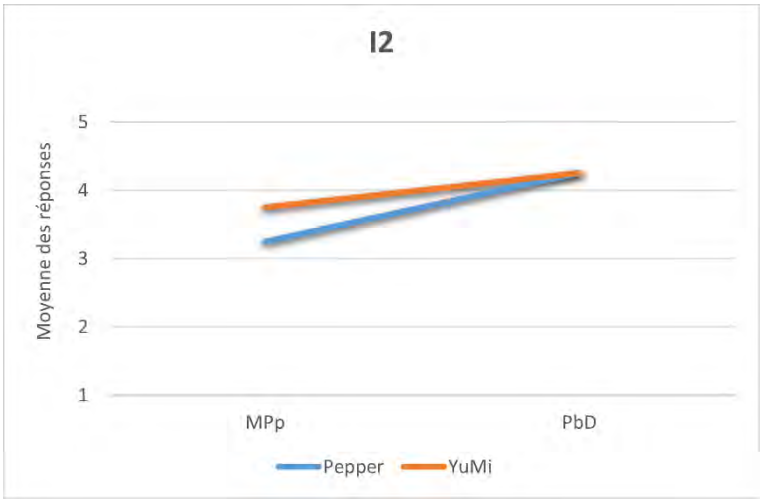


Figure 45 : Graphique de l'interaction entre le facteur type de programmation et le facteur type de cobot sur l'item I2 ($p < .05$)

Item I2 : « Le robot a appris comme un humain » ; type de programmation : MPp ou PbD ; type de cobot : Pepper ou YuMi ; réponse entre 1 (Pas du tout d'accord) et 7 (Tout à fait d'accord)

Illustré par la Figure 46, le score de l'item I3 (« Le robot pensait comme un humain. ») est plus faible dans la condition PbD mais non significatif ($p > .05$). Le type de cobot a un effet simple significatif seulement avec le robot YuMi. Le robot YuMi a un score plus faible dans la condition PbD ($t(4{,}52) = 2{,}778, p < .05$).

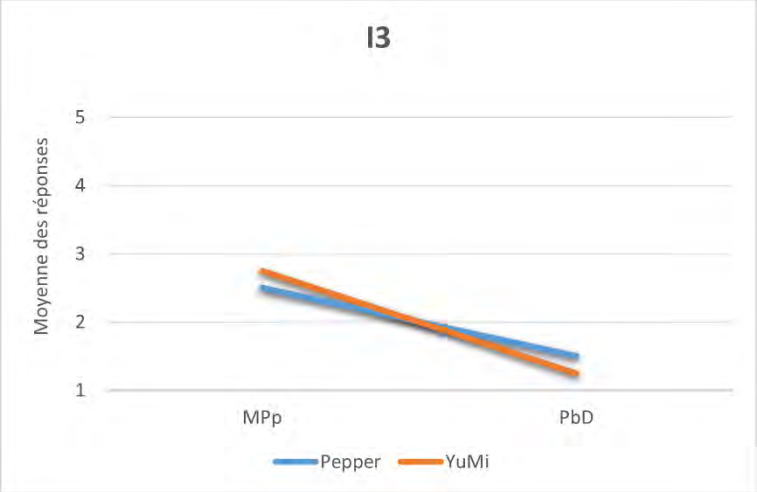


Figure 46 : Graphique de l'interaction entre le facteur type de programmation et le facteur type de cobot sur l'item I3 ($p < .05$)

Item I2 : « Le robot pensait comme un humain » ; type de programmation : MPp ou PbD ; type de cobot : Pepper ou YuMi ; réponse entre 1 (Pas du tout d'accord) et 7 (Tout à fait d'accord)

Le Tableau 23 regroupe les résultats de la dernière question du questionnaire : « Avez-vous trouvez l'expérience amusante ? ». Les scores montrent un effet significatif du type de programmation. Les participants ont trouvé l'expérience plus amusante dans la condition PbD par rapport à la condition MPp ($F(1/12) = 5{,}000, p < .05$).

Tableau 23 : Résultats à la question « Avez-vous trouvé l'expérience amusante ? »

	Expérience amusante
MPp Pepper	4,000 (0,816)
MPp YuMi	4,500 (0,577)
PbD Pepper	5,000 (0,000)
PbD YuMi	4,750 (0,500)

4.3.4 Discussion

Les hypothèses de départ sont les suivantes :

H1 : « L'acceptabilité de la programmation sera plus importante lorsque qu'elle se fait par démonstration plutôt que par des mouvements préprogrammés. » ;

H2 : « Les cobots ont davantage de chance d'être acceptés quand ils sont anthropomorphes. » ;

H3 : « L'anthropomorphisme améliore l'acceptabilité de la programmation par démonstration d'un cobot. ».

Par rapport à l'hypothèse H1, la charge de travail, mentale et physique, sont, comme prévu, plus élevé dans la condition PbD que MPp. Les durées de formation et d'expérience, le nombre de reproduction et le nombre de visionnage de la vidéo de la tâche à réaliser ont également des scores meilleurs par rapport à la condition MPp. Ces résultats s'expliquent par la phase d'apprentissage, phase supplémentaire dans la condition PbD par rapport à la condition MPp. Aucune différence significative n'est observée sur les modalités « utilité perçue », « attitude », « confiance » et « anxiété ». Il est important de noter que les participants ont préféré la condition PbD : meilleur score au niveau de l'item S1 « J'ai aimé utiliser ce dispositif » et au niveau de la question sur l'amusement. Les participants ont donné une note légèrement plus élevée dans la condition Pbd à l'affirmation « Le robot apprenait comme un humain. ».

L'hypothèse H2 a tendance à être invalidée à cause des mauvaises performances du robot Pepper. Comme prévu, il n'y a pas de différence au niveau de la modalité « charge de travail » selon le type de cobot. La « satisfaction », l'« utilité perçue », la « facilité d'utilisation perçue » et l'item C3, lié à la confiance sur ce que le robot doit faire, obtiennent des scores plus élevés avec le robot YuMi. Les participants ont tendance à avoir besoin de faire plus de démonstration dans la condition PbD avec le robot Pepper. Avec le taux de succès, légèrement plus faible avec le robot Pepper, les mauvais résultats obtenus se comprennent. Les participants ont noté l'affirmation « Le robot pensait comme un humain. » avec une note plus faible dans la condition PbD et d'autant plus avec le robot YuMi.

L'hypothèse H3 est invalidée par les résultats de l'expérience. Que ce soit au niveau de l'item C3 lié à la confiance dans l'exécution du robot et la modalité « anxiété », aucune différence significative n'est observée dans la condition PbD selon le type de robot. De manière globale, le robot YuMi dispose de résultats légèrement plus élevés puisque son taux de succès et sa « facilité d'utilisation » sont plus grandes. Ces conclusions sont très intéressantes puisqu'elles montrent que le niveau d'anthropomorphisme n'est pas très important contrairement à la performance du robot.

De manière générale, l'expérience, peu importe le type de programmation, donne de bonnes moyennes en termes de satisfaction, utilité perçue, facilité d'utilisation, attitude, confort. Les niveaux d'anxiété obtenus sont faibles. La programmation par démonstration a bien été comprise et acceptée par les utilisateurs. Elle nécessite un temps de programmation plus élevé que la « programmation par mouvements préprogrammés » sans que cette augmentation diminue l'acceptabilité de manière générale. La différence observée entre les robots est davantage liée à la performance du robot qu'à son aspect.

Cela est très encourageant puisque l'aspect humanoïde du robot n'est pas mieux ou mal perçu par les utilisateurs. Au regard des résultats obtenus, l'industrie du futur ne nécessite pas l'ajout d'aspect humanoïde à partir du moment où le cobot effectue son travail et le réussit.

Un autre aspect important et non mesuré lors de l'expérience est la compréhension de la programmation par démonstration. Les utilisateurs, via la tâche à effectuer lors de la formation, prouvent leur compréhension du dispositif. Tous les participants qui ont effectué la programmation par démonstration ont correctement réussi à montrer les mouvements au robot, à utiliser l'interface et à imaginer le potentiel du dispositif. De nombreuses remarques orales de ces participants sont également allées dans ce sens : facilité d'utilisation ressentie, très bonne idée pour les utilisateurs novices et appréciation du dispositif, sont des exemples de ces commentaires.

Conclusion

La programmation par démonstration est développée pour permettre à tout un chacun d'enseigner une action à un robot. Cependant, l'assimilation d'une telle technologie nécessite son acceptation et sa compréhension par l'ensemble des utilisateurs.

De manière à répondre à ces questions, une expérience est mise en place. Une pré-expérience a validé le niveau d'anthropomorphisme des robots humanoïdes. L'expérience consiste à comparer le niveau d'acceptation de la programmation par démonstration par rapport à la programmation utilisant des mouvements préprogrammés en y incluant le facteur anthropomorphique.

Les résultats montrent que la programmation par démonstration est bien acceptée et comprise par les utilisateurs même si cette dernière nécessite une charge de travail supérieure, autant physique que mentale.

Le niveau d'anthropomorphisme n'influence pas l'acceptabilité. Dans notre expérience, les résultats liés au niveau d'anthropomorphisme s'expliquent par les mauvaises performances du robot Pepper.

Tous ces résultats montrent que la programmation par démonstration est acceptée si elle est bien comprise pas les utilisateurs et que l'aspect humanoïde du robot n'entre pas en ligne de compte. L'ajout de cobots au sein de processus industriels peut ainsi profiter de ces conclusions.

Conclusion générale

« Regardez vers les étoiles et pas vers vos pieds. Essayez de donner un sens à ce que vous voyez et demandez-vous ce qui fait l'univers existe. Soyez curieux. Même si la vie peut parfois paraître difficile, il y a toujours quelque chose que vous pouvez faire, et que vous pouvez réussir. L'important, c'est de na pas abandonner. Tant qu'il y a de la vie, il y a de l'espoir. »

Stephen Hawking[10]

Sommaire

[10] Extrait d'un message audio que Stephen Hawking a enregistré peu avant sa mort.

L'objectif de cette thèse était de développer l'apprentissage par démonstration pour des cobots et d'évaluer l'acceptabilité de la programmation par démonstration par des utilisateurs non-experts. Dans un premier temps, une étude sur le pré-traitement des données a abouti à la sélection de la méthode de pré-traitement la plus adaptée pour l'apprentissage par démonstration. Dans un second temps, une architecture générique est proposée afin de fonctionner sur de nombreux robots. Un modèle d'apprentissage par démonstration avec pondération est développé pour être adaptable selon les choix de l'utilisateur. Dans un dernier temps, une étude sur l'acceptabilité de la programmation par démonstration a montré que cette nouvelle méthode de programmation d'un cobot est bien acceptée par l'utilisateur et que l'anthropomorphisme du cobot n'a pas d'influence sur l'acceptabilité générale, contrairement à la performance du robot. La programmation par démonstration développée dans cette thèse fonctionne sur de nombreux robots pour des tâches complexes et est acceptée, comprise et utilisée par des utilisateurs non-experts qui sont les potentiels futurs utilisateurs des cobots. Le code développé et richement documenté est disponible, sur simple demande à l'auteure. Ces divers travaux sont prometteurs pour le développement des cobots dans l'industrie du futur.

Périmètre du domaine d'étude

L'architecture proposée est compatible avec tous les robots disposant d'un mode passif pour la manipulation de leurs articulations. Dans le cas contraire, la commande du robot à l'aide d'une manette est également possible mais moins intuitive et moins précise qu'avec des démonstrations kinesthésiques (voir Section 3.1).

Les démonstrations kinesthésiques peuvent être compliquées si le robot dispose d'un nombre élevé de degrés de liberté. Dans notre expérience sur l'acceptabilité de la programmation par démonstration, les participants utilisant le robot YuMi, robot avec 7 joints par bras, ont réussi à le manipuler à la suite de la formation. Un robot disposant de deux bras peut être programmé par démonstration, avec un utilisateur par bras pour faciliter la manipulation.

La vérification de la trajectoire générée par l'apprentissage par démonstration n'a pas été développée dans cette thèse afin de conserver la généricité de l'architecture proposée. Cette vérification est une étape supplémentaire qui est généralement développée par les industriels. Elle est donc directement liée au robot employé.

Une attention toute particulière devra être portée à la gestion des collisions entre les bras et les objets de l'environnement. La prise en compte de ces potentielles collisions pourrait être réalisée via l'implémentation d'une phase de test de la trajectoire générée. La modélisation de l'environnement, mis à jour, ainsi que la connaissance de la structure du robot permettent de tester les collisions en amont de l'exécution du mouvement. Cette phase, dépendante à la fois du robot et de son environnement de travail, n'a pas été développée. En effet, cette étape supplémentaire n'appartient pas au champ d'étude car elle est fortement liée à la plateforme robotique considérée, contrairement à l'aspect générique de l'architecture proposée.

Ce type de programmation est simple d'utilisation mais moins précis que la programmation standard car directement lié à la précision de la démonstration faite par l'utilisateur. Dans le cas d'une tâche demandant beaucoup de précision, comme le suivi d'une ligne par exemple, la programmation standard sera plus adaptée.

Les résultats liés à l'étude sur l'acceptabilité sont effectués sur un nombre limité de participants, ce qui pourrait être amélioré avec un recrutement plus conséquent.

Perspectives

La programmation par démonstration ouvre de nouvelles perspectives pour la programmation des cobots ainsi que pour leurs futures applications. Divers axes d'amélioration peuvent être envisagés à différents niveaux : l'apprentissage par démonstration, les robots, les canaux de communications humain-robot et l'aspect juridique des cobots.

Axe 1 : perspectives d'amélioration de l'apprentissage par démonstration proposée

- **Algorithme pour la sélection automatique de la pondération** : le modèle d'apprentissage proposé pourrait être développé en ajoutant un algorithme de sélection automatique de la pondération à appliquer en fonction des points à pondérer choisis par l'utilisateur. Cet algorithme devra inclure une ou plusieurs mesure(s) de performance de la trajectoire générée afin d'affiner la valeur de la pondération. Cette amélioration simplifierait le processus de programmation d'un cobot avec la méthode proposée pour les utilisateurs non-experts.
- **Etape de vérification de la trajectoire générée** : afin de pouvoir être déployé dans les industries, le modèle proposé devra inclure une étape de vérification de la trajectoire générée et pouvoir s'adapter à la position des objets dans l'environnement du cobot.
- **Nouvelles données en entrée pour l'apprentissage par démonstration** : le modèle proposé pourrait également avoir en entrée des données d'une autre nature comme la force en plus des valeurs des joints (Gao et al., 2019 ; Nematollahi et al., 2019 ; Pignat & Calinon, 2019 ; Silvério et al., 2018).

Axe 2 : développement des possibilités des robots

- **Amélioration du mode passif « Lead-through » du robot YuMi** : le robot YuMi pourrait avoir son mode passif amélioré en augmentant la résistance du robot lorsque l'utilisateur se rapproche d'une limite de joint. Cela aura pour effet l'amélioration des démonstrations des utilisateurs.
- **Développement de packages ROS pour le robot YuMi** : la majorité des robots industriels sont liés à un langage propriétaire, ce qui complexifie leur programmation. Le développement de packages ROS pour les robots industriels, comme le robot YuMi, favoriserait l'intégration de ces robots.

Axe 3 : amélioration de l'acceptabilité des robots via l'ajout de nouveaux modes de communication

- **Ajout de nouveaux modes de communications** : pour améliorer l'acceptabilité des cobots, d'autres techniques de communication pourraient être développées comme le regard, qui permet de montrer ou vérifier la position d'un objet, ou la parole, qui permet de donner des consignes ou des retours sur ce qui se passe (Calinon & Billard, 2008a).

Axe 4 : réflexion autour des règles juridiques des cobots

- **Etude sur les droits et responsabilités des cobots** : le cobot, nouveau type de robot, doit également être étudié au niveau du droit et des responsabilités qu'on peut lui accorder (Gelin & Guilhem, 2016).

Liste des figures

Liste des tableaux

Bibliographie

Aleotti, J., & Caselli, S. (2005). Trajectory clustering and stochastic approximation for robot programming by demonstration. *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1029-1034. https://doi.org/10.1109/IROS.2005.1545365

Argall, B. D., Chernova, S., Veloso, M., & Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, *57*(5), 469-483. https://doi.org/10.1016/j.robot.2008.10.024

Asfour, T., Azad, P., Gyarfas, F., & Dillmann, R. (2008). Imitation learning of dual-arm manipulation tasks in humanoid robots. *International Journal of Humanoid Robotics*, *05*(02), 183-202. https://doi.org/10.1142/S0219843608001431

Bartneck, C., Kulić, D., Croft, E., & Zoghbi, S. (2009). Measurement Instruments for the Anthropomorphism, Animacy, Likeability, Perceived Intelligence, and Perceived Safety of Robots. *International Journal of Social Robotics*, *1*, 71-81. https://doi.org/10.1007/s12369-008-0001-3

Bauer, W., Bender, M., Braun, M., Rally, P., & Scholtz, O. (Éds.). (2016). *Lightweight robots in manual assembly—Best to start simply ! : Examining companies' initial experiences with lightweight robots*. Fraunhofer IAO, Stuttgart. http://publica.fraunhofer.de/documents/N-415111.html

Beeson, P., & Ames, B. (2015). TRAC-IK : An Open-Source Library for Improved Solving of Generic Inverse Kinematics. *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, 928-935. https://doi.org/10.1109/HUMANOIDS.2015.7363472

Beraldo, G., Di Battista, S., Badaloni, S., Menegatti, E., & Pivetti, M. (2018). Sex differences in expectations and perception of a social robot. *2018 IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO)*, 38-43. https://doi.org/10.1109/ARSO.2018.8625826

Berndt, D. J., & Clifford, J. (1994). Using Dynamic Time Warping to Find Patterns in Time Series. *AAA1-94 Workshop on Knowledge Discovery in Databases*, 359-370.

Billard, A. G., Calinon, S., & Guenter, F. (2006). Discriminative and Adaptive Imitation in Uni-Manual and Bi-Manual Tasks. *Robotics and Autonomous Systems*, *54*(5), 370-384. https://doi.org/10.1016/j.robot.2006.01.007

Bobillier-Chaumon, M.-É., & Dubois, M. (2009). L'adoption des technologies en situation professionnelle : Quelles articulations possibles entre acceptabilité et acceptation ? *Le Travail Humain*, *72*(4), 355-382. https://doi.org/10.3917/th.724.0355

Brand, M., & Hertzmann, A. (2000). Style Machines. *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, 183-192. https://doi.org/10.1145/344779.344865

Busy, M., & Caniot, M. (2020). *QiBullet, a Bullet-based simulator for the Pepper and NAO robots*. https://arxiv.org/abs/1909.00779v3

Calinon, S. (2009). *Robot Programming by Demonstration : A Probabilistic Approach*. EPFL/CRC Press.

Calinon, S. (2016). A tutorial on task-parameterized movement learning and retrieval. *Intelligent Service Robotics*, *9*(1), 1-29. https://doi.org/10.1007/s11370-015-0187-9

Calinon, S. (2018). Robot Learning with Task-Parameterized Generative Models. In A. Bicchi & W. Burgard (Éds.), *Robotics Research* (Vol. 3, p. 111-126). Springer International Publishing. https://doi.org/10.1007/978-3-319-60916-4_7

Calinon, S., & Billard, A. (2004). Stochastic Gesture Production and Recognition Model for a Humanoid Robot. *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, *3*, 2769-2774. https://doi.org/10.1109/IROS.2004.1389828

Calinon, S., & Billard, A. (2007a). Incremental Learning of Gestures by Imitation in a Humanoid Robot. *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction*, 255-262. https://doi.org/10.1145/1228716.1228751

Calinon, S., & Billard, A. (Éds.). (2008a). A framework integrating statistical and social cues to teach a humanoid robot new skills. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Workshop on Social Interaction with Intelligent Indoor Robots*.

Calinon, S., & Billard, A. (2008b). A Probabilistic Programming by Demonstration Framework Handling Constraints in Joint Space and Task Space. *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 367-372. https://doi.org/10.1109/IROS.2008.4650593

Calinon, S., & Billard, A. G. (2007b). What is the teacher's role in robot programming by demonstration? : Toward benchmarks for improved learning. *Interaction Studies*, *8*(3), 441-464. https://doi.org/10.1075/is.8.3.08cal

Calinon, S., D'halluin, F., Sauser, E. L., Caldwell, D. G., & Billard, A. G. (2010). Learning and Reproduction of Gestures by Imitation. *IEEE Robotics & Automation Magazine*, *17*(2), 44-54. https://doi.org/10.1109/MRA.2010.936947

Calinon, S., Guenter, F., & Billard, A. (2005). Goal-Directed Imitation in a Humanoid Robot. *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 299-304. https://doi.org/10.1109/ROBOT.2005.1570135

Calinon, S., Guenter, F., & Billard, A. (2007). On Learning, Representing, and Generalizing a Task in a Humanoid Robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *37*(2), 286-298. https://doi.org/10.1109/TSMCB.2006.886952

Calinon, S., Sauser, E. L., Billard, A. G., & Caldwell, D. G. (2010). Evaluation of a probabilistic approach to learn and reproduce gestures by imitation. *2010 IEEE International Conference on Robotics and Automation*, 2671-2676. https://doi.org/10.1109/ROBOT.2010.5509988

Charalambous, G., Fletcher, S., & Webb, P. (2016). The Development of a Scale to Evaluate Trust in Industrial Human-robot Collaboration. *International Journal of Social Robotics*, *8*, 193-209. https://doi.org/10.1007/s12369-015-0333-8

Chernova, S., & Thomaz, A. L. (2014). Robot Learning from Human Teachers. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, *8*(3), 1-121. https://doi.org/10.2200/S00568ED1V01Y201402AIM028

Choi, W., Cho, J., Lee, S., & Jung, Y. (2020). Fast Constrained Dynamic Time Warping for Similarity Measure of Time Series Data. *IEEE Access*, *8*, 222841-222858. https://doi.org/10.1109/ACCESS.2020.3043839

Davis, F. D. (1989). Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *MIS Quarterly*, *13*(3), 319-340. https://doi.org/10.2307/249008

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data Via the EM Algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, *39*(1), 1-22. https://doi.org/10.1111/j.2517-6161.1977.tb01600.x

Dinet, J., & Vivian, R. (2015). Perception et attitudes à l'égard des robots anthropomorphes en France : Validation d'une échelle d'attitudes. *Psychologie Française*, *60*, 173-189. https://doi.org/10.1016/j.psfr.2015.05.002

Donjat, J., Legeleux, A., Buche, C., & Duhaut, D. (2022). Temporal Alignment and Demonstration Selection as Pre-processing Phase for Learning by Demonstration. *The International FLAIRS Conference Proceedings*, *35*. https://doi.org/10.32473/flairs.v35i.130649

Douglas, D. H., & Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, *10*(2), 112-122. https://doi.org/10.3138/FM57-6770-U75U-7727

Duffy, B. R. (2003). Anthropomorphism and the social robot. *Robotics and Autonomous Systems*, *42*, 177-190. https://doi.org/10.1016/S0921-8890(02)00374-3

El Zaatari, S., Marei, M., Li, W., & Usman, Z. (2019). Cobot programming for collaborative industrial tasks : An overview. *Robotics and Autonomous Systems*, *116*, 162-180. https://doi.org/10.1016/j.robot.2019.03.003

Elmar de Koning. (2011). *psimpl—Douglas-Peucker simplification*. http://psimpl.sourceforge.net/douglas-peucker.html

Fraune, M. R., Leite, I., Karatas, N., Amirova, A., Legeleux, A., Sandygulova, A., Neerincx, A., Dilip Tikas, G., Gunes, H., Mohan, M., Abbasi, N. I., Shenoy, S., Scassellati, B., de Visser, E. J., & Komatsu, T. (2022). Lessons Learned About Designing and Conducting Studies From HRI Experts. *Frontiers in Robotics and AI*, *8*. https://doi.org/10.3389/frobt.2021.772141

Fussell, S. R., Kiesler, S., Setlock, L. D., & Yew, V. (2008). How People Anthropomorphize Robots. *2008 3rd ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 145-152. https://doi.org/10.1145/1349822.1349842

Gao, X., Ling, J., Xiao, X., & Li, M. (2019). Learning Force-Relevant Skills from Human Demonstration. *Complexity*, *2019*. https://doi.org/10.1155/2019/5262859

Gebru, I. D., Alameda-Pineda, X., Forbes, F., & Horaud, R. (2016). EM Algorithms for Weighted-Data Clustering with Application to Audio-Visual Scene Analysis. *IEEE Transactions on*

Pattern Analysis and Machine Intelligence, *38*(12), 2402-2415. https://doi.org/10.1109/TPAMI.2016.2522425

Geler, Z., Kurbalija, V., Ivanović, M., Radovanović, M., & Dai, W. (2019). Dynamic Time Warping : Itakura vs Sakoe-Chiba. *2019 IEEE International Symposium on INnovations in Intelligent SysTems and Applications (INISTA)*, 1-6. https://doi.org/10.1109/INISTA.2019.8778300

Gelin, R., & Guilhem, O. (2016). *Le robot est-il l'avenir de l'homme ?* La Documentation française.

Hart, S. G., & Staveland, L. E. (1988). Development of NASA-TLX (Task Load Index) : Results of Empirical and Theoretical Research. In P. A. Hancock & N. Meshkati (Éds.), *Human Mental Workload* (Vol. 52, p. 139-183). North-Holland. https://doi.org/10.1016/S0166-4115(08)62386-9

Hedlund, E., Johnson, M., & Gombolay, M. (2021). The Effects of a Robot's Performance on Human Teachers for Learning from Demonstration Tasks. *Proceedings of the 2021 ACM/IEEE International Conference on Human-Robot Interaction*, 207-215. https://doi.org/10.1145/3434073.3444664

Hoffman, G. (2019). Evaluating Fluency in Human–Robot Collaboration. *IEEE Transactions on Human-Machine Systems*, *49*(3), 209-218. https://doi.org/10.1109/THMS.2019.2904558

Hornung, A., & Lemaignan, S. (2015). *Naoqi_pose ROS package*. http://wiki.ros.org/naoqi_pose

IFR. (2017). *The Impact of Robots on Productivity, Employment and Jobs*. International Federation of Robotics. https://ifr.org/papers

IFR. (2021, octobre 28). *World Robotics 2021*. International Federation of Robotics. https://ifr.org/downloads/press2018/2021_10_28_WR_PK_Presentation_long_version.pdf

Ijspeert, A. J., Nakanishi, J., Hoffmann, H., Pastor, P., & Schaal, S. (2013). Dynamical Movement Primitives : Learning Attractor Models for Motor Behaviors. *Neural Computation*, *25*(2), 328-373. https://doi.org/10.1162/NECO_a_00393

Ijspeert, A. J., Nakanishi, J., & Schaal, S. (2002). Movement Imitation with Nonlinear Dynamical Systems in Humanoid Robots. *Proceedings 2002 IEEE International Conference on Robotics and Automation*, *2*, 1398-1403. https://doi.org/10.1109/ROBOT.2002.1014739

Jambhale, S. S., & Khaparde, A. (2014). Gesture Recognition Using DTW & Piecewise DTW. *2014 International Conference on Electronics and Communication Systems (ICECS)*, 1-5. https://doi.org/10.1109/ECS.2014.6892646

Jeong, Y.-S., Jeong, M. K., & Omitaomu, O. A. (2011). Weighted dynamic time warping for time series classification. *Pattern Recognition*, *44*(9), 2231-2240. https://doi.org/10.1016/j.patcog.2010.09.022

Khansari-Zadeh, S. M., & Billard, A. (2011). Learning Stable Nonlinear Dynamical Systems With Gaussian Mixture Models. *IEEE Transactions on Robotics*, *27*(5), 943-957. https://doi.org/10.1109/TRO.2011.2159412

Kleinpeter, É. (2015). Le Cobot, la coopération entre l'utilisateur et la machine. *Multitudes*, *58*, 70-75. https://doi.org/10.3917/mult.058.0070

Knese, K. (2015). *Naoqi_driver ROS package*. http://wiki.ros.org/naoqi_driver

Kragic, D., Gustafson, J., Karaoguz, H., Jensfelt, P., & Krug, R. (2018). Interactive, Collaborative Robots : Challenges and Opportunities. *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, 18-25. https://doi.org/10.24963/ijcai.2018/3

Kyrarini, M. (2019). *Robot Learning from Human Demonstrations for Human-Robot Synergy* [Doctoral dissertation, Universität Bremen]. http://nbn-resolving.de/urn:nbn:de:gbv:46-00107477-17

Kyrarini, M., & Graeser, A. (2017). Automatic Selection of Human Demonstrations for Robot Learning of Industrial Scenario. *2017 IEEE/RSJ IROS WORKSHOP Human in-the-loop robotic manipulation: on the influence of the human role*. https://doi.org/10.13140/RG.2.2.17733.76002

Kyrarini, M., Haseeb, M. A., Ristić-Durrant, D., & Gräser, A. (2019). Robot learning of industrial assembly task via human demonstrations. *Autonomous Robots*, *43*(1), 239-257. https://doi.org/10.1007/s10514-018-9725-6

Kyrarini, M., Leu, A., Ristić-Durrant, D., Gräser, A., Jackowski, A., Gebhard, M., Nelles, J., Bröhl, C., Brandl, C., Mertens, A., & Schlick, C. M. (2016). Human-Robot Synergy for Cooperative Robots. *Facta Universitatis, Series: Automatic Control and Robotics*, *15*(3), 187-204. https://doi.org/10.22190/FUACR1603187K

Kyrarini, M., Zheng, Q., Haseeb, M. A., & Gräser, A. (2019). Robot Learning of Assistive

Manipulation Tasks by Demonstration via Head Gesture-based Interface. *2019 IEEE 16th International Conference on Rehabilitation Robotics (ICORR)*, 1139-1146. https://doi.org/10.1109/ICORR.2019.8779379

Legeleux, A., Buche, C., & Duhaut, D. (2022). Gaussian Mixture Model with Weighted Data for Learning by Demonstration. *The International FLAIRS Conference Proceedings*, *35*. https://doi.org/10.32473/flairs.v35i.130559

Legeleux, A., Le Bigot, N., Buche, C., & Duhaut, D. (2021, mars). Acceptability of the Programming by Demonstrations of Cobots by Non-Experts. *Workshop YOUR study design! Participatory critique and refinement of participants' studies*. 2021 ACM/IEEE International Conference on Human-Robot Interaction (HRI '21), New York, NY, USA.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, *5.1*, 281-297.

Meyer, J.-A. (2016). La préhistoire de l'IA. Visite guidée des automates et proto-robots conçus depuis l'Antiquité jusqu'au milieu du XXe siècle. *Technique et Sciences Informatiques*, *35*(1), 75-113. https://doi.org/10.3166/tsi.35.75-113

Michaelis, J. E., Siebert-Evenstone, A., Shaffer, D. W., & Mutlu, B. (2020). Collaborative or Simply Uncaged? Understanding Human-Cobot Interactions in Automation. *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 1-12. https://doi.org/10.1145/3313831.3376547

Mingyue Ma, L., Fong, T., Micire, M. J., Kim, Y. K., & Feigh, K. (2018). Human-Robot Teaming : Concepts and Components for Design. In M. Hutter & R. Siegwart (Éds.), *Field and Service Robotics* (Vol. 5, p. 649-663). Springer International Publishing. https://doi.org/10.1007/978-3-319-67361-5_42

Mori, M. (1970). The Uncanny Valley. *Energy*, *7*(4), 33-35.

Movchan, A., & Zymbler, M. (2015). Time Series Subsequence Similarity Search Under Dynamic Time Warping Distance on the Intel Many-core Accelerators. In G. Amato, R. Connor, F. Falchi, & C. Gennaro (Éds.), *Similarity Search and Applications* (p. 295-306). Springer International Publishing. https://doi.org/10.1007/978-3-319-25087-8_28

Müller, M. (Éd.). (2007). Dynamic Time Warping. In *Information Retrieval for Music and Motion*

(p. 69-84). Springer. https://doi.org/10.1007/978-3-540-74048-3_4

Mülling, K., Kober, J., Kroemer, O., & Peters, J. (2013). Learning to Select and Generalize Striking Movements in Robot Table Tennis. *The International Journal of Robotics Research*, *32*(3), 263-279. https://doi.org/10.1177/0278364912472380

Munich, M. E., & Perona, P. (1999). Continuous Dynamic Time Warping for translation-invariant curve alignment with applications to signature verification. *Proceedings of the Seventh IEEE International Conference on Computer Vision*, *1*, 108-115. https://doi.org/10.1109/ICCV.1999.791205

Muscillo, R., Conforto, S., Schmid, M., Caselli, P., & D'Alessio, T. (2007). Classification of Motor Activities through Derivative Dynamic Time Warping applied on Accelerometer Data. *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 4930-4933. https://doi.org/10.1109/IEMBS.2007.4353446

Nematollahi, I., Kuhner, D., Welschehold, T., & Burgard, W. (2019). Augmenting Action Model Learning by Non-Geometric Features. *2019 International Conference on Robotics and Automation (ICRA)*, 7769-7775. https://doi.org/10.1109/ICRA.2019.8794153

Nielsen, J. (1994). *Usability Engineering*. Morgan Kaufmann.

Nomura, T., Suzuki, T., Kanda, T., & Kato, K. (2006a). Measurement of negative attitudes toward robots. *Interaction Studies*, *7*(3), 437-454. https://doi.org/10.1075/is.7.3.14nom

Nomura, T., Suzuki, T., Kanda, T., & Kato, K. (2006b). Measurement of Anxiety toward Robots. *ROMAN 2006 - The 15th IEEE International Symposium on Robot and Human Interactive Communication*, 372-377. https://doi.org/10.1109/ROMAN.2006.314462

Pastor, P., Hoffmann, H., Asfour, T., & Schaal, S. (2009). Learning and Generalization of Motor Skills by Learning from Demonstration. *2009 IEEE International Conference on Robotics and Automation*, 763-768. https://doi.org/10.1109/ROBOT.2009.5152385

Pignat, E., & Calinon, S. (2017). Learning adaptive dressing assistance from human demonstration. *Robotics and Autonomous Systems*, *93*, 61-75. https://doi.org/10.1016/j.robot.2017.03.017

Pignat, E., & Calinon, S. (2019). Bayesian Gaussian Mixture Model for Robotic Policy Imitation. *IEEE Robotics and Automation Letters*, *4*(4), 4452-4458. https://doi.org/10.1109/LRA.2019.2932610

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., & Ng, A. (2009). ROS : An open-source Robot Operating System. *ICRA Workshop on Open Source Software*.

Rabaud, V. (2015). *nao_apps—ROS package*. http://wiki.ros.org/nao_apps

Ramer, U. (1972). An Iterative Procedure for the Polygonal Approximation of Plane Curves. *Computer Graphics and Image Processing*, *1*(3), 244-256. https://doi.org/10.1016/S0146-664X(72)80017-0

Ravichandar, H., Polydoros, A. S., Chernova, S., & Billard, A. (2020). Recent Advances in Robot Learning from Demonstration. *Annual Review of Control, Robotics, and Autonomous Systems*, *3*(1), 297-330. https://doi.org/10.1146/annurev-control-100819-063206

Rozo, L., Calinon, S., Caldwell, D. G., Jiménez, P., & Torras, C. (2016). Learning Physical Collaborative Robot Behaviors From Human Demonstrations. *IEEE Transactions on Robotics*, *32*(3), 513-527. https://doi.org/10.1109/TRO.2016.2540623

Sabbaghi, E., Bahrami, M., & Ghidary, S. S. (2014). Learning of Gestures by Imitation using a Monocular Vision System on a Humanoid Robot. *2014 Second RSI/ISM International Conference on Robotics and Mechatronics (ICRoM)*, 588-594. https://doi.org/10.1109/ICRoM.2014.6990966

Sakoe, H., & Chiba, S. (1978). Dynamic Programming Algorithm Optimization for Spoken Word Recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *26*(1), 43-49. https://doi.org/10.1109/TASSP.1978.1163055

Salvini, P., Laschi, C., & Dario, P. (2010). Design for Acceptability : Improving Robots' Coexistence in Human Society. *International Journal of Social Robotics*, *2*, 451-460. https://doi.org/10.1007/s12369-010-0079-2

Sanderson, C., & Curtin, R. (2016). Armadillo : A template-based C++ library for linear algebra. *Journal of Open Source Software*, *1*, 26. https://doi.org/10.21105/joss.00026

Sanderson, C., & Curtin, R. (2018). A User-Friendly Hybrid Sparse Matrix Class in C++. In J. H. Davenport, M. Kauers, G. Labahn, & J. Urban (Éds.), *International Congress on Mathematical Software* (Vol. 10931, p. 422-430). Springer International Publishing. https://doi.org/10.1007/978-3-319-96418-8_50

Sasaki, K., Ihaya, K., & Yamada, Y. (2017). Avoidance of Novelty Contributes to the Uncanny

Valley. *Frontiers in Psychology*, *8*, 1792. https://doi.org/10.3389/fpsyg.2017.01792

Schaal, S. (2006). Dynamic Movement Primitives—A Framework for Motor Control in Humans and Humanoid Robotics. In H. Kimura, K. Tsuchiya, A. Ishiguro, & H. Witte (Éds.), *Adaptive Motion of Animals and Machines* (p. 261-280). Springer. https://doi.org/10.1007/4-431-31381-8_23

Schaal, S., Mohajerian, P., & Ijspeert, A. (2007). Dynamics systems vs. Optimal control—A unifying view. In P. Cisek, T. Drew, & J. F. Kalaska (Éds.), *Progress in Brain Research* (Vol. 165, p. 425-445). Elsevier. https://doi.org/10.1016/S0079-6123(06)65027-9

Schmidtler, J., Bengler, K., Dimeas, F., & Campeau-Lecours, A. (2017). A Questionnaire for the Evaluation of Physical Assistive Devices (QUEAD) : Testing Usability and Acceptance in physical Human-Robot Interaction. *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 876-881. https://doi.org/10.1109/SMC.2017.8122720

Schwarz, G. (1978). Estimating the Dimension of a Model. *The Annals of Statistics*, *6*(2), 461-464.

Silvério, J., Huang, Y., Rozo, L., Calinon, S., & Caldwell, D. G. (2018). Probabilistic Learning of Torque Controllers from Kinematic and Force Constraints. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1-8. https://doi.org/10.1109/IROS.2018.8594103

Stadler, S., Weiss, A., & Tscheligi, M. (2014). I Trained this Robot : The Impact of Pre-Experience and Execution Behavior on Robot Teachers. *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, 1030-1036. https://doi.org/10.1109/ROMAN.2014.6926388

Stäubli. (2021). *Le Guide de la Robotique*.

Steinfeld, A., Fong, T., Kaber, D., Lewis, M., Scholtz, J., Schultz, A., & Goodrich, M. (2006). Common Metrics for Human-Robot Interaction. *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction*, 33-40. https://doi.org/10.1145/1121241.1121249

Strait, M. K., Aguillon, C., Contreras, V., & Garcia, N. (2017). The Public's Perception of Humanlike Robots : Online Social Commentary Reflects an Appearance-Based Uncanny Valley, a General Fear of a "Technology Takeover", and the Unabashed Sexualization of Female-Gendered Robots. *2017 26th IEEE International Symposium on Robot and Human*

Interactive Communication (RO-MAN), 1418-1423. https://doi.org/10.1109/ROMAN.2017.8172490

Terrade, F., Pasquier, H., Reerinck-Boulanger, J., Guingouain, G., & Somat, A. (2009). L'acceptabilité sociale : La prise en compte des déterminants sociaux dans l'analyse de l'acceptabilité des systèmes technologiques. *Le travail humain*, *72*, 383-395. https://doi.org/10.3917/th.724.0383

Tjerngren, J., & ABB, A. (2018). Ease-of-use Packages between ROS and ABB Robots. *ROS-Industrial Conference*.

Toyota Motor Corporation. (2021). *Tmc ROS package*. https://docs.hsr.io/hsr_develop_manual_en/index.html

Venkatesh, V., Morris, M. G., Davis, G. B., & Davis, F. D. (2003). User Acceptance of Information Technology : Toward a Unified View. *MIS Quarterly*, *27*(3), 425-478. https://doi.org/10.2307/30036540

Wang, W., Chen, Y., Li, R., & Jia, Y. (2019). Learning and Comfort in Human–Robot Interaction : A Review. *Applied Sciences*, *9*(23), 5152. https://doi.org/10.3390/app9235152

Waytz, A., Cacioppo, J., & Epley, N. (2010). Who Sees Human? : The Stability and Importance of Individual Differences in Anthropomorphism. *Perspectives on Psychological Science*, *5*(3), 219-232. https://doi.org/10.1177/1745691610369336

Waytz, A., Heafner, J., & Epley, N. (2014). The mind in the machine : Anthropomorphism increases trust in an autonomous vehicle. *Journal of Experimental Social Psychology*, *52*, 113-117. https://doi.org/10.1016/j.jesp.2014.01.005

Weiss, A., Igelsbock, J., Calinon, S., Billard, A., & Tscheligi, M. (2009). Teaching a Humanoid : A User Study on Learning by Demonstration with HOAP-3. *RO-MAN 2009 - The 18th IEEE International Symposium on Robot and Human Interactive Communication*, 147-152. https://doi.org/10.1109/ROMAN.2009.5326274

Zhu, Z., & Hu, H. (2018). Robot Learning from Demonstration in Robotic Assembly : A Survey. *Robotics*, *7*(2), 17. https://doi.org/10.3390/robotics7020017

Złotowski, J., Yogeeswaran, K., & Bartneck, C. (2017). Can we control it? Autonomous robots threaten human identity, uniqueness, safety, and resources. *International Journal of Human-Computer Studies*, *100*, 48-54. https://doi.org/10.1016/j.ijhcs.2016.12.008

Liste des annexes

Annexe 1

Publications par l'auteure

Les travaux de recherche menés dans cette thèse sont reconnus par plusieurs publications scientifiques (conférences, journal, workshop) et deux conférences scientifiques – une internationale et une nationale.

Articles scientifiques

- **Publications dans un journal**

Fraune, M. R., Leite, I., Karatas, N., Amirova, A., **Legeleux, A.**, Sandygulova, A., Neerincx, A., Dilip Tikas, G., Gunes, H., Mohan, M., Abbasi, N. I., Shenoy, S., Scassellati, B., de Visser, E. J., & Komatsu, T. (2022). Lessons Learned About Designing and Conducting Studies From HRI Experts. *Frontiers in Robotics and AI, 8*. https://doi.org/10.3389/frobt.2021.772141

- **Publications dans une conférence**

Donjat, J., **Legeleux, A.**, Buche, C., & Duhaut, D. (2022). Temporal Alignment and Demonstration Selection as Pre-processing Phase for Learning by Demonstration. *The International FLAIRS Conference Proceedings*, *35*. https://doi.org/10.32473/flairs.v35i.130649

Legeleux, A., Buche, C., & Duhaut, D. (2022). Gaussian Mixture Model with Weighted Data for Learning by Demonstration. *The International FLAIRS Conference Proceedings*, *35*. https://doi.org/10.32473/flairs.v35i.130559

- **Acceptation dans un workshop**

Legeleux, A., Le Bigot, N., Buche, C., & Duhaut, D. (2021, mars). Acceptability of the Programming by Demonstrations of Cobots by Non-Experts. *Workshop YOUR study design! Participatory critique and refinement of participants' studies*. 2021 ACM/IEEE International Conference on Human-Robot Interaction (HRI '21), New York, NY, USA.

- **Autre article**

Dizet, A., Le Bono, C., **Legeleux, A.**, Neau, M., & Buche, C. (2021). RoboCup@Home Education 2020 Best Performance : RoboBreizh, a modular approach. *arXiv:2107.02978*. http://arxiv.org/abs/2107.02978

Conférences scientifiques

- **Internationale**

Dizet, A., Le Bono, C., **Legeleux, A.**, Neau, M., & Buche, C. (2020, septembre 30). *Design Of A Modular Architecture Using ROS* [Invited Lecture Series]. RoboCup@Home Education. https://www.robocupathomeedu.org/learn/online-classroom/invited-lecture-series

- **Nationale**

Buche, C., Desmeulles, G., & **Legeleux, A**. (2020, juin 19). *Enjeux industriels et sociétaux de l'Intelligence Artificielle* [Conférence invitée]. Petit-Déjeuner Débat, Concarneau, France.

Annexe 2

Processus d'ajout du robot YuMi

ABB a développé des packages ROS pour l'utilisation de leurs robots (Tjerngren & ABB, 2018). Cependant, ces packages ne disposent pas de toutes les fonctionnalités nécessaires à l'ajout du robot dans l'architecture proposée.

Ainsi, l'ajout du robot YuMi a nécessité l'utilisation des technologies **Robot Web Service** (RWS) et **External Guided Motion** (EGM).

RWS est un service mettant à disposition une interface web pour établir une communication entre deux machines. RWS utilise HTTP comme protocole de communication grâce à un serveur REST implanté dans le robot et par lequel n'importe quelle machine extérieure, connectée via un câble Ethernet, peut se connecter pour communiquer avec. Les librairies POCO C++ (*POrtable COmponents C++ Libraries*) sont utilisées pour gérer les requête RWS.

EGM est une option disponible sur certains robots afin de contrôler le mouvement du robot à l'aide d'un dispositif externe. EGM comporte trois principales caractéristiques : le flux de position (envoi en continu de la position du robot), le guidage en position (mouvement du robot en fonction de la position envoyée par le dispositif externe) et la correction de chemin (correction du mouvement du robot en fonction des données envoyées par le dispositif externe). Dans cette thèse, seules les fonctionnalités flux de position et guidage en position sont utilisées. Pour fonctionner, l'option EGM a nécessité l'utilisation des librairies Boost C++ (*Boost C++ Libraries*) et du Google Protocol Buffers.

Afin de créer un lien entre l'état de l'architecture proposée et l'état du robot, une machine à états est mise en place.

Au départ, le système est initialisé à **l'état 0**. Après chaque changement d'état, le système retourne à l'état 0, qui est un état intermédiaire afin que le robot attende le prochain changement d'état.

L'état 1 correspond à la désactivation des moteurs, c'est-à-dire à l'activation du mode « Lead-through ».

L'état 2 intervient à la fin de l'enregistrement d'une démonstration et stoppe le mode « Lead-through ».

L'état 3 permet l'enregistrement d'une démonstration en démarrant la communication EGM.

L'état 4 permet de faire bouger le robot avec EGM, terminée avec **l'état 5**.

L'état 6 bouge le robot vers la position de départ.

L'état 7 termine le programme.

Tous ces états sont écrits dans un programme RAPID. Les changements d'état du robot sont commandés par le programme principal écrit en C++. Le programme RAPID est transféré sur le contrôleur du robot et démarré avec RWS. Le programme principal, proposé dans cette thèse, s'exécute sur un ordinateur sous Ubuntu relié par un câble Ethernet au robot.

Afin de gérer les différentes machines, des IP fixes sont utilisées. Pour le robot, le port service est utilisé car son IP est fixe (192.168.125.1). Le PC sous Windows ayant le simulateur RobotStudio a comme IP : 192.168.125.2. Le PC contenant le programme principal a pour IP fixe 192.168.125.3.

Pour démarrer la connexion avec le robot (**fonction start**), plusieurs requêtes RWS mettent le robot dans un état initial et démarrent les programmes RAPID.

La première requête gère les cookies avec le robot. En effet, la connexion se limite à soixante-dix requêtes si les cookies ne sont pas gérés correctement.

Ensuite, deux autres requêtes mettent le robot en mode automatique et activent les moteurs.

Un fichier en RAPID pour chaque bras est transféré sur le contrôleur du robot puis chargé à l'aide de quatre autres requêtes.

La requête suivante active les tâches RAPID. Il y a une tâche par bras (un bras correspondant techniquement à un robot). Chaque tâche peut comporter un ou plusieurs programme(s) RAPID. Le pointeur de programme est réinitialisé à l'aide d'une nouvelle requête afin de démarrer les programmes au début.

Une nouvelle requête démarre les programmes RAPID.

Une autre requête attend le changement d'état d'une variable appelée « done » afin d'attendre la fin de l'initialisation de chaque bras. Cette variable change d'état avant le démarrage de la machine à états dans les programmes RAPID. Cette variable est utilisée dans le programme principal afin d'attendre la fin d'un changement d'état dans le programme RAPID. Le robot (ou les robots pour chaque bras) est maintenant prêt à être utilisé.

La fin de la connexion au robot (**fonction stop**) termine la machine à états en modifiant l'état en 7 à l'aide d'une requête qui change la valeur de la variable correspondant à l'état. Les programmes RAPID sont également stoppés avec une autre requête. Les sockets ouverts dans le programme principal pour la communication EGM sont également fermés.

L'activation (**fonction moteur on**) et la désactivation des moteurs (**fonction moteur off**) se fait à l'aide d'une requête RWS qui modifie l'état de la machine à états vers la valeur 2 et 1 respectivement. L'état 2, utilisé après la fin d'une démonstration, coupe le mode « Lead-through » et la communication EGM. L'état 1 active le mode « Lead-through ».

La position du robot est enregistrée dans une structure contenant la position du bras gauche et celle du bras droit (mode joint ou cartésien), l'état de la pince gauche et la pince droite et le pas de temps courant. La conversion entre cette structure et une matrice (**fonction conversion position-matrice**) est donc simple à mettre en place. La conversion matrice vers commande (**fonction conversion matrice-commande**) se fait également avec cette structure.

Le déplacement vers la position de départ (**fonction déplacement vers la position départ**) correspond à l'état 6. Le mouvement est réalisé à l'aide d'une procédure en RAPID et d'une position prédéfinie.

Le déplacement du robot selon le programme principal (**fonction déplacement selon commande**) correspond à l'état 4 dans la machine à états. Du côté du robot, la communication EGM est démarrée.

Cette communication se composent de deux sockets UDP, une par bras du robot. Cette communication peut fonctionner avec soit une position en mode joint, soit en mode cartésien. Dans le cadre de cette thèse, le mode joint est utilisé. En effet, le mode cartésien aurait pu convenir mais ce mode ne prend en compte que six axes sur les sept de chaque bras et ne garantit pas la fidélité de la position de l'ensemble du bras dans l'espace par rapport aux démonstrations.

Pour pouvoir bouger les pinces du robot, ces dernières sont reliées à l'état de variables modifiables avec une requête RWS. Du côté du programme principal, la communication vers les mêmes sockets est activée.

Deux threads sont démarrés afin d'envoyer simultanément la position des bras et des pinces. Chaque thread gère la position d'un bras et la pince correspondante. La fin du déplacement entraine l'activation de l'état 5 qui coupe la communication EGM.

L'état 3 permet la récupération de la position courante du robot (**fonction boucle position**). Du côté du robot, la communication EGM est démarrée et la position courante du robot est envoyée dans les deux sockets UDP. La position des pinces du robot est corrélée à une variable, indiquant leur état, et peut être lue à l'aide d'une requête RWS.

La commande, ouverture et fermeture de chaque pince, est associée à des boutons de la tablette tactile du robot. L'utilisateur peut ainsi manipuler les bras du robot lors des démonstrations et utiliser les boutons de la tablette pour fermer ou ouvrir les pinces du robot. Chaque pince est associée à un bouton qui modifie l'état de la pince. Si la pince est ouverte, elle se ferme et inversement.

Du côté du programme principal, trois threads sont démarrés. Le premier thread récupère la position des pinces du robot à l'aide de deux requêtes RWS. Le deuxième thread récupère la position du bras gauche à l'aide d'un socket UDP et le dernier thread celle du bras droit. Toutes les données sont stockées à l'aide de la structure de position décrite précédemment.

Les deux bras sont synchronisés à l'aide de threads qui parallélisent la récupération des positions et la commande en position. Cependant, il est possible qu'un thread soit plus rapide qu'un autre et inversement. Par exemple, une requête RWS prend plus de temps que la réception ou l'envoi de données sur un socket UDP. Une synchronisation est donc nécessaire pour la réception et l'envoi des positions du robot.

Pour la récupération des positions, la synchronisation se déroule de la manière suivante. Chaque donnée est récupérée toutes les 4 millisecondes. La boucle de la fonction boucle position, à chaque pas de temps, attend que la structure de position contienne une position du bras gauche et une position du bras droit. Si le thread modifiant la valeur de l'état des pinces n'a pas eu le temps de mettre à jour les valeurs, les valeurs retenues correspondent aux dernières valeurs connues de l'état des pinces. La valeur complète de la position du robot (la position des deux bras ainsi que l'état des deux pinces) est sauvegardée. Ensuite, la structure contenant la position courante du robot est vidée sauf au niveau de l'état des pinces du robot.

Pour l'envoi des positions, une synchronisation est également nécessaire. Il faut que les bras et les pinces soient synchronisés lors de leur mouvement. Chaque thread, un pour le mouvement des bras et un pour le mouvement des pinces, a comme paramètre le mouvement complet à réaliser. Dans une boucle prenant fin lorsque le dernier temps du mouvement est atteint, le temps est scruté. Si le temps courant est en avance par rapport à la prochaine commande, le thread attend. Si le thread est en retard, le retard est affiné afin d'envoyer la bonne commande. Si le thread est à temps, la commande en cours est envoyée. Ainsi, les threads se synchronisent en fonction du temps réel courant et du temps de la commande.

Annexe 3

Algorithme EM avec des données pondérées

Le Tableau 24 répertorie les notations mathématiques employées.

Tableau 24 : Notations mathématiques

J	Le nombre de démonstrations	K	Le nombre de gaussiennes du modèle GMM
N	Le nombre total de données	$P(\xi_j)$	La fonction de densité du modèle GMM
N_j	Le nombre de données de la démonstration j	π_k	Le vecteur de probabilités *a priori* de la $k^{\text{ième}}$ gaussienne du modèle GMM
ξ	Un jeu de données	μ_k	Le vecteur de moyenne de la $k^{\text{ième}}$ gaussienne du modèle GMM
ξ_j	La $j^{\text{ième}}$ donnée	Σ_k	La matrice de covariance de la $k^{\text{ième}}$ gaussienne du modèle GMM
$\boldsymbol{w_j}$	Le poids associé à la $j^{\text{ième}}$ donnée		

Dans notre cas, le **modèle GMM avec données pondérées** (W-GMM) se définit avec les équations suivantes :

$$P(\xi_j) = \sum_{k=1}^{K} P(k)\, P(\xi_j|k)$$

$$P(k) = \pi_k$$

$$P(\xi_j|k) = \mathcal{N}(\xi_j; \mu_k, \Sigma_k/\boldsymbol{w_j}) = \frac{1}{\sqrt{(2\pi)^D det(\Sigma_k/\boldsymbol{w_j})}} e^{-\frac{1}{2}\left((\xi_j-\mu_k)^T\left(\frac{\Sigma_k}{\boldsymbol{w_j}}\right)^{-1}(\xi_j-\mu_k)\right)}$$

où l'on notera par la suite $\Theta = \{\pi_k, \mu_k, \Sigma_k/\boldsymbol{w_j}\}_{k=1}^{K}$ et $\widehat{\Theta}$ son estimation.

L'estimation des paramètres du modèle GMM peut s'obtenir à l'aide du principe du maximum de vraisemblance.

$$\widehat{\Theta} = argmax_{\Theta} \prod_{j=1}^{N} P(\xi_j; \Theta)$$

L'**algorithme Espérance-Maximisation**[11] (EM) (Dempster et al., 1977) propose une méthode pour l'estimation des paramètres du modèle $\widehat{\Theta}$. L'algorithme EM est itératif et son objectif est de déterminer les paramètres d'un modèle probabilité ayant des variables latentes non observables à l'aide du maximum de vraisemblance. Cet algorithme se décompose en deux temps : l'étape E avec le calcul de l'espérance et l'étape M avec la maximisation.

L'**étape E** correspond à la formulation de la fonction Q à l'itération u. On note $\xi = \{\xi_j\}_{j=1}^{N}$ le jeu de données et $\mathrm{Z} = \{\mathrm{z}_j\}_{j=1}^{N}$ l'ensemble des données latentes. Dans notre cas, ces données latentes spécifient la gaussienne associée à chaque donnée d'entrée.

$$Q_u(\Theta) = \mathbb{E}_{Z|\xi;\Theta_u}[\log P(\xi, Z; \Theta)]$$

Avec la linéarité de l'espérance, la fonction Q devient :

$$Q_u(\Theta) = \sum_{j=1}^{N} \mathbb{E}_{z_j|\xi_j;\Theta_u}\big[\log P\big(\xi_j, z_j; \Theta\big)\big].$$

La définition de l'espérance amène à l'équation suivante :

$$Q_u(\Theta) = \sum_{j=1}^{N} \sum_{k=1}^{K} P(z_j = k|\xi_j; \Theta) \cdot \log P\big(\xi_j, z_j; \Theta\big).$$

Le théorème de Bayes sur les probabilités permet d'obtenir :

$$\begin{aligned} Q_u(\Theta) &= \sum_{j=1}^{N} \sum_{k=1}^{K} \frac{P(\xi_j|z_j = k; \Theta) \cdot P(z_j = k; \Theta)}{P\big(\xi_j; \Theta\big)} \cdot \log P\big(\xi_j, z_j; \Theta\big) \\ &= \sum_{j=1}^{N} \sum_{k=1}^{K} \frac{P(\xi_j|z_j = k; \Theta) \cdot P(z_j = k; \Theta)}{\sum_{h=1}^{K} P\big(z_j = k; \Theta\big) \cdot \ P(\xi_j|z_j = k; \Theta)} \cdot \log P\big(\xi_j, z_j; \Theta\big). \end{aligned}$$

[11] La présente démonstration s'appuie sur celle de Matthew N. Bernstein (https://mbernste.github.io/posts/gmm_em/) pour le modèle GMM.

Pour simplifier les équations, les notations suivantes sont être adoptés :

$\pi_{u,k} = P(\xi_j | z_j = k; \Theta)$

$\pi_{u,h} = P(\xi_j | z_j = h; \Theta)$

$\phi(\xi_j; \mu_{u,k}, \Sigma_{u,k}/\boldsymbol{w_j}) = P(\xi_j | z_j = k; \Theta)$

$\phi(\xi_j; \mu_{u,h}, \Sigma_{u,h}/\boldsymbol{w_j}) = P(\xi_j | z_j = h; \Theta)$

$P(\xi_j, z_j; \Theta) = \pi_k \cdot \phi(\xi_j; \mu_k, \Sigma_k/\boldsymbol{w_j})$.

L'équation devient alors :

$$Q_u(\Theta) = \sum_{j=1}^{N} \sum_{k=1}^{K} \frac{\pi_{u,k} \cdot \phi(\xi_j; \mu_{u,k}, \Sigma_{u,k}/\boldsymbol{w_j})}{\sum_{h=1}^{K} \pi_{u,h} \cdot \phi(\xi_j; \mu_{u,h}, \Sigma_{u,h}/\boldsymbol{w_j})} \cdot \log(\pi_k \cdot \phi(\xi_j; \mu_k, \Sigma_k/\boldsymbol{w_j}))$$

$$Q_u(\Theta) = \sum_{j=1}^{N} \sum_{k=1}^{K} p_{k,j}^{u} \cdot \log(\pi_k \cdot \phi(\xi_j; \mu_k, \Sigma_k/\boldsymbol{w_j}))$$

Avec $p_{k,j}^{u} = \frac{\pi_{u,k} \cdot \phi(\xi_j; \mu_{u,k}, \Sigma_{u,k}/\boldsymbol{w_j})}{\sum_{h=1}^{K} \pi_{u,h} \cdot \phi(\xi_j; \mu_{u,h}, \Sigma_{u,h}/\boldsymbol{w_j})}$.

L'**étape M** consiste à maximiser la fonction $Q_u(\Theta)$. La notation π_k correspond aux probabilités qu'une donnée appartienne à la gaussienne k. Ainsi la somme des probabilités doit être égale à 1. L'objectif est donc de résoudre :

$$\Theta_{u+1} = argmax_{\theta} Q_u(\Theta)$$

avec $\sum_{k=1}^{K} \pi_k = 1$.

Grâce à cette contrainte et à la continuité de l'objectif et de la contrainte, le problème d'optimisation peut être résolu avec les multiplicateurs de Lagrange. La fonction lagrangien L est définie ainsi :

$$L(\Theta, \lambda) = \sum_{j=1}^{N} \sum_{k=1}^{K} p_{k,j}^{u} \cdot \log(\pi_k \cdot \phi(\xi_j; \mu_k, \Sigma_k/\boldsymbol{w_j})) + \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right)$$

La solution pour chaque paramètre de Θ s'obtient en rendant la dérivée partielle de la fonction lagrangien égale à 0. Tout d'abord, **la dérivée partielle par rapport au paramètre $\boldsymbol{\pi_k}$** donne, pour un k donné :

$$\frac{\partial L(\Theta, \lambda)}{\partial \pi_k} = \frac{1}{\pi_k} \sum_{j=1}^{N} p_{k,j}^{u} + \lambda.$$

La valeur de π_k, correspondant à un extremum, peut être déterminé ainsi :

$$0 = \frac{1}{\pi_k} \sum_{j=1}^{N} p_{k,j}^{u} + \lambda$$

$$\Rightarrow \pi_k = -\frac{1}{\lambda} \sum_{j=1}^{N} p_{k,j}^{u}.$$

En ajoutant ce résultat à la contrainte $\sum_{k=1}^{K} \pi_k = 1$, la valeur de λ peut être obtenue :

$$\sum_{k=1}^{K} -\frac{1}{\lambda} \sum_{j=1}^{N} p_{k,j}^{u} = 1$$

$$\Rightarrow \lambda = -\sum_{j=1}^{N} \sum_{k=1}^{K} p_{k,j}^{u}$$

$$\Rightarrow \lambda = -n \text{ car } \sum_{k=1}^{K} p_{k,j}^{u} = 1.$$

En ajoutant ce résultat à l'équation de la dérivée partielle de la fonction lagrangien, la valeur finale de π_k peut être déterminée :

$$0 = \frac{1}{\pi_k} \sum_{j=1}^{N} p_{k,j}^{u} + \lambda$$

$$\Rightarrow 0 = \frac{1}{\pi_k} \sum_{j=1}^{N} p_{k,j}^{u} - n$$

$$\boxed{\Rightarrow \pi_k = \frac{1}{n} \sum_{j=1}^{N} p_{k,j}^{u}.}$$

La dérivée partielle de la fonction lagrangien **par rapport à $\boldsymbol{\mu_k}$** pour un k donné est la suivante :

$$\frac{\partial L(\Theta,\lambda)}{\partial \mu_k} = \sum_{j=1}^{N} p_{k,j}^{u} \cdot \frac{\partial}{\partial \mu_k} \cdot \log(\pi_k \cdot \phi(\xi_j; \mu_k, \Sigma_k/\boldsymbol{w_j}))$$

$$= \sum_{j=1}^{N} \frac{p_{k,j}^{u}}{\pi_k \cdot \phi(\xi_j; \mu_k, \Sigma_k/\boldsymbol{w_j})} \cdot \frac{\partial}{\partial \mu_k} \pi_k \cdot \phi(\xi_j; \mu_k, \Sigma_k/\boldsymbol{w_j})$$

$$= \sum_{j=1}^{N} \frac{p_{k,j}^{u} \cdot \frac{\partial}{\partial \mu_k} \phi(\xi_j; \mu_k, \Sigma_k/\boldsymbol{w_j})}{\phi(\xi_j; \mu_k, \Sigma_k/\boldsymbol{w_j})}$$

$$= \sum_{j=1}^{N} \frac{p_{k,j}^{u} \cdot \frac{\partial}{\partial \mu_k} e^{-\frac{1}{2}\left((\xi_j-\mu_k)^T \left(\frac{\Sigma_k}{\boldsymbol{w_j}}\right)^{-1} (\xi_j-\mu_k)\right)}}{e^{-\frac{1}{2}\left((\xi_j-\mu_k)^T \left(\frac{\Sigma_k}{\boldsymbol{w_j}}\right)^{-1} (\xi_j-\mu_k)\right)}}$$

$$= \sum_{j=1}^{N} p_{k,j}^{u} \cdot \frac{\partial}{\partial \mu_k} \left(-\frac{1}{2} \left((\xi_j - \mu_k)^T \left(\frac{\Sigma_k}{\boldsymbol{w_j}}\right)^{-1} (\xi_j - \mu_k) \right) \right).$$

Comme la matrice Σ_k est symétrique, la propriété $\frac{\partial}{\partial y}(x-y)\Sigma(x-y) = -2\Sigma(x-y)$ peut s'appliquer :

$$\frac{\partial L(\Theta,\lambda)}{\partial \mu_k} = \sum_{j=1}^{N} p_{k,j}^{u} \cdot \left(-\frac{1}{2} \left(-2 \left(\frac{\Sigma_k}{\boldsymbol{w_j}}\right)^{-1} (\xi_j - \mu_k) \right) \right)$$

$$= \sum_{j=1}^{N} p_{k,j}^{u} \cdot \left(\mathbf{w_j} \cdot \Sigma^{-1} (\xi_j - \mu_k) \right).$$

La valeur de $\mu_k$correspond à un extremum, sa dérivée partielle est donc nulle.

$$0 = \sum_{j=1}^{N} p_{k,j}^{u} \cdot \left(\mathbf{w_j} \cdot \Sigma^{-1} (\xi_j - \mu_k) \right)$$

$$\Rightarrow \Sigma \cdot 0 = \sum_{j=1}^{N} p_{k,j}^{u} \cdot \left(\mathbf{w_j} \cdot \Sigma \cdot \Sigma^{-1} \cdot (\xi_j - \mu_k) \right)$$

$$\Rightarrow \sum_{j=1}^{N} \mu_k \cdot \mathbf{w_j} \cdot p_{k,j}^{u} = \sum_{j=1}^{N} \mathbf{w_j} \cdot p_{k,j}^{u} \cdot \xi_j$$

$$\boxed{\Rightarrow \mu_k = \frac{\sum_{j=1}^{N} \mathbf{w_j} \cdot p_{k,j}^{u} \cdot \xi_j}{\sum_{j=1}^{N} \mathbf{w_j} \cdot p_{k,j}^{u}}}$$

Le dernier paramètre de Θ s'obtient en **dérivant partiellement** la fonction lagrangien **par rapport au paramètre $\boldsymbol{\Sigma_k}$**. Pour un k donné, la dérivée partielle est définie ainsi :

$$\begin{aligned}\frac{\partial L(\Theta,\lambda)}{\partial \Sigma_k} &= \sum_{j=1}^{N} p_{k,j}^{u} \cdot \frac{\partial}{\partial \Sigma_k} \cdot log(\pi_k \cdot \phi(\xi_j; \mu_k, \Sigma_k/\boldsymbol{w_j})) \\ &= \sum_{j=1}^{N} \frac{p_{k,j}^{u}}{\pi_k \cdot \phi(\xi_j; \mu_k, \Sigma_k/\boldsymbol{w_j})} \cdot \frac{\partial}{\partial \Sigma_k}\left(\pi_k \cdot \phi(\xi_j; \mu_k, \Sigma_k/\boldsymbol{w_j})\right) \\ &= \sum_{j=1}^{N} p_{k,j}^{u} \frac{\frac{\partial}{\partial \Sigma_k}\left(\phi(\xi_j; \mu_k, \Sigma_k/\boldsymbol{w_j})\right)}{\phi(\xi_j; \mu_k, \Sigma_k/\boldsymbol{w_j})}.\end{aligned}$$

Avec $\phi(\xi_j; \mu_k, \Sigma_k/\boldsymbol{w_j}) = \frac{1}{\sqrt{(2\pi)^D det(\Sigma_k/\boldsymbol{w_j})}} e^{-\frac{1}{2}\left((\xi_j-\mu_k)^T\left(\frac{\Sigma_k}{\boldsymbol{w_j}}\right)^{-1}(\xi_j-\mu_k)\right)}$ et la propriété $\left(\frac{u}{v}\right)' = u'v - uv'$, l'équation devient :

$$\frac{\partial L(\Theta,\lambda)}{\partial \Sigma_k} = \sum_{j=1}^{N} p_{k,j}^{u} \cdot \left[\underbrace{\frac{\frac{\partial}{\partial \Sigma_k} e^{-\frac{1}{2}\left((\xi_j-\mu_k)^T\left(\frac{\Sigma_k}{\boldsymbol{w_j}}\right)^{-1}(\xi_j-\mu_k)\right)}}{e^{-\frac{1}{2}\left((\xi_j-\mu_k)^T\left(\frac{\Sigma_k}{\boldsymbol{w_j}}\right)^{-1}(\xi_j-\mu_k)\right)}}}_{A} - \underbrace{\frac{\frac{\partial}{\partial \Sigma_k}\sqrt{(2\pi)^D det(\Sigma_k/\boldsymbol{w_j})}}{\sqrt{(2\pi)^D det(\Sigma_k/\boldsymbol{w_j})}}}_{B}\right].$$

Pour simplifier les calculs, les notations A et B sont adoptées. Calcul pour déterminer A :

$$A = \frac{\frac{\partial}{\partial \Sigma_k} e^{-\frac{1}{2}\left((\xi_j-\mu_k)^T\left(\frac{\Sigma_k}{\boldsymbol{w_j}}\right)^{-1}(\xi_j-\mu_k)\right)}}{e^{-\frac{1}{2}\left((\xi_j-\mu_k)^T\left(\frac{\Sigma_k}{\boldsymbol{w_j}}\right)^{-1}(\xi_j-\mu_k)\right)}} = -\frac{1}{2} \cdot \boldsymbol{w_j} \cdot \frac{\partial}{\partial \Sigma_k}\left((\xi_j-\mu_k)^T \Sigma_k^{-1}(\xi_j-\mu_k)\right).$$

La propriété $\frac{\partial}{\partial X} x^T X^{-1} y = -(X^{-1})^T x y^T (X^{-1})^T$ et le fait que l'inverse d'une matrice symétrique est également symétrique, donne :

$$A = -\frac{1}{2} \cdot \boldsymbol{w_j} \cdot \left(-\Sigma_{\mathrm{k}}^{-1} \cdot (\xi_j - \mu_k) \cdot (\xi_j - \mu_k)^T \cdot \Sigma_{\mathrm{k}}^{-1}\right).$$

Calcul pour déterminer B :

$$B = \frac{\frac{\partial}{\partial \Sigma_k} \sqrt{(2\pi)^D det(\Sigma_k / \boldsymbol{w_j})}}{\sqrt{(2\pi)^D det(\Sigma_k / \boldsymbol{w_j})}}$$

$$= \frac{\partial}{\partial \Sigma_k} log \sqrt{(2\pi)^D det(\Sigma_k / \boldsymbol{w_j})}$$

$$= \frac{1}{2} \cdot \frac{\partial}{\partial \Sigma_k} log \left((2\pi)^D det(\Sigma_k / \boldsymbol{w_j})\right)$$

$$= \frac{1}{2} \cdot \frac{\partial}{\partial \Sigma_k} log \left(det\left(\frac{\Sigma_k}{\boldsymbol{w_j}}\right)\right).$$

La propriété $\frac{d}{dX} log|det(X)| = (X^T)^{-1}$ devient $\frac{d}{dX} log(det(X)) = (X^T)^{-1}$ dans notre cas. En effet, la matrice de covariance Σ_k est définie semi-positive ainsi son déterminant n'est pas négatif. De plus, la matrice de covariance est symétrique, donc sa transposée est égale à elle-même.

$$B = \frac{1}{2} \cdot \Sigma_{\mathrm{k}}^{-1}$$

En réinjectant A et B, l'équation devient :

$$\frac{\partial L(\Theta, \lambda)}{\partial \Sigma_k} = \sum_{j=1}^{N} p_{k,j}^u \cdot \left[-\frac{1}{2} \cdot \boldsymbol{w_j} \cdot \left(-\Sigma_{\mathrm{k}}^{-1} \cdot (\xi_j - \mu_k) \cdot (\xi_j - \mu_k)^T \cdot \Sigma_{\mathrm{k}}^{-1}\right) - \frac{1}{2} \cdot \Sigma_{\mathrm{k}}^{-1}\right]$$

$$= -\frac{1}{2} \cdot \sum_{j=1}^{N} p_{k,j}^u \cdot \left[\boldsymbol{w_j} \cdot \left(-\Sigma_{\mathrm{k}}^{-1} \cdot (\xi_j - \mu_k) \cdot (\xi_j - \mu_k)^T \cdot \Sigma_{\mathrm{k}}^{-1}\right) + \Sigma_{\mathrm{k}}^{-1}\right]$$

$$= -\frac{1}{2} \cdot \left[\left(\sum_{j=1}^{N} \boldsymbol{w_j} \cdot p_{k,j}^{u} \right) \Sigma_{\mathrm{k}}^{-1} - \Sigma_{\mathrm{k}}^{-1} \left(\sum_{j=1}^{N} \boldsymbol{w_j} \cdot p_{k,j}^{u} \cdot \left(\xi_j - \mu_k\right) \cdot \left(\xi_j - \mu_k\right)^T \right) \cdot \Sigma_{\mathrm{k}}^{-1} \right]$$

Finalement, il reste à trouver la solution de l'équation quand elle est égale à 0 :

$$0 = \left(\sum_{j=1}^{N} \boldsymbol{w_j} \cdot p_{k,j}^{u} \right) \Sigma_{\mathrm{k}}^{-1} - \Sigma_{\mathrm{k}}^{-1} \left(\sum_{j=1}^{N} \boldsymbol{w_j} \cdot p_{k,j}^{u} \cdot \left(\xi_j - \mu_k\right) \cdot \left(\xi_j - \mu_k\right)^T \right) \cdot \Sigma_{\mathrm{k}}^{-1}.$$

La matrice I correspond à la matrice identité :

$$0 = \Sigma_{\mathrm{k}}^{-1} \left[\left(\sum_{j=1}^{N} \boldsymbol{w_j} \cdot p_{k,j}^{u} \right) \cdot \mathrm{I} - \left(\sum_{j=1}^{N} \boldsymbol{w_j} \cdot p_{k,j}^{u} \cdot \left(\xi_j - \mu_k\right) \cdot \left(\xi_j - \mu_k\right)^T \right) \Sigma_{\mathrm{k}}^{-1} \right]$$

$$\Rightarrow 0 = \left(\sum_{j=1}^{N} \boldsymbol{w_j} \cdot p_{k,j}^{u} \right) \cdot \mathrm{I} - \left(\sum_{j=1}^{N} \boldsymbol{w_j} \cdot p_{k,j}^{u} \cdot \left(\xi_j - \mu_k\right) \cdot \left(\xi_j - \mu_k\right)^T \right) \Sigma_{\mathrm{k}}^{-1}$$

$$\Rightarrow \left(\sum_{j=1}^{N} \boldsymbol{w_j} \cdot p_{k,j}^{u} \right) \cdot \mathrm{I} = \left(\sum_{j=1}^{N} \boldsymbol{w_j} \cdot p_{k,j}^{u} \cdot \left(\xi_j - \mu_k\right) \cdot \left(\xi_j - \mu_k\right)^T \right) \Sigma_{\mathrm{k}}^{-1}$$

$$\Rightarrow \left(\sum_{j=1}^{N} \boldsymbol{w_j} \cdot p_{k,j}^{u} \right) \cdot \Sigma_{\mathrm{k}}^{-1} = \sum_{j=1}^{N} \boldsymbol{w_j} \cdot p_{k,j}^{u} \cdot \left(\xi_j - \mu_k\right) \cdot \left(\xi_j - \mu_k\right)^T$$

$$\boxed{\Rightarrow \Sigma_{\mathrm{k}}^{-1} = \frac{\sum_{j=1}^{N} \boldsymbol{w_j} \cdot p_{k,j}^{u} \cdot \left(\xi_j - \mu_k\right) \cdot \left(\xi_j - \mu_k\right)^T}{\sum_{j=1}^{N} \boldsymbol{w_j} \cdot p_{k,j}^{u}}}$$

Les trois résultats obtenus pour l'étape M ainsi que celui de l'étape E permettent de calculer les paramètres du modèle GMM avec des données pondérées (W-GMM).

Annexe 4

Questionnaire pré-expérience

Le questionnaire développé pour la pré-expérience sur l'anthropomorphisme des robots est utilisé sous format numérique à l'aide de l'outil Google Forms. Les Figure 47 et Figure 48 illustrent l'aspect graphique du questionnaire, complété par les participants.

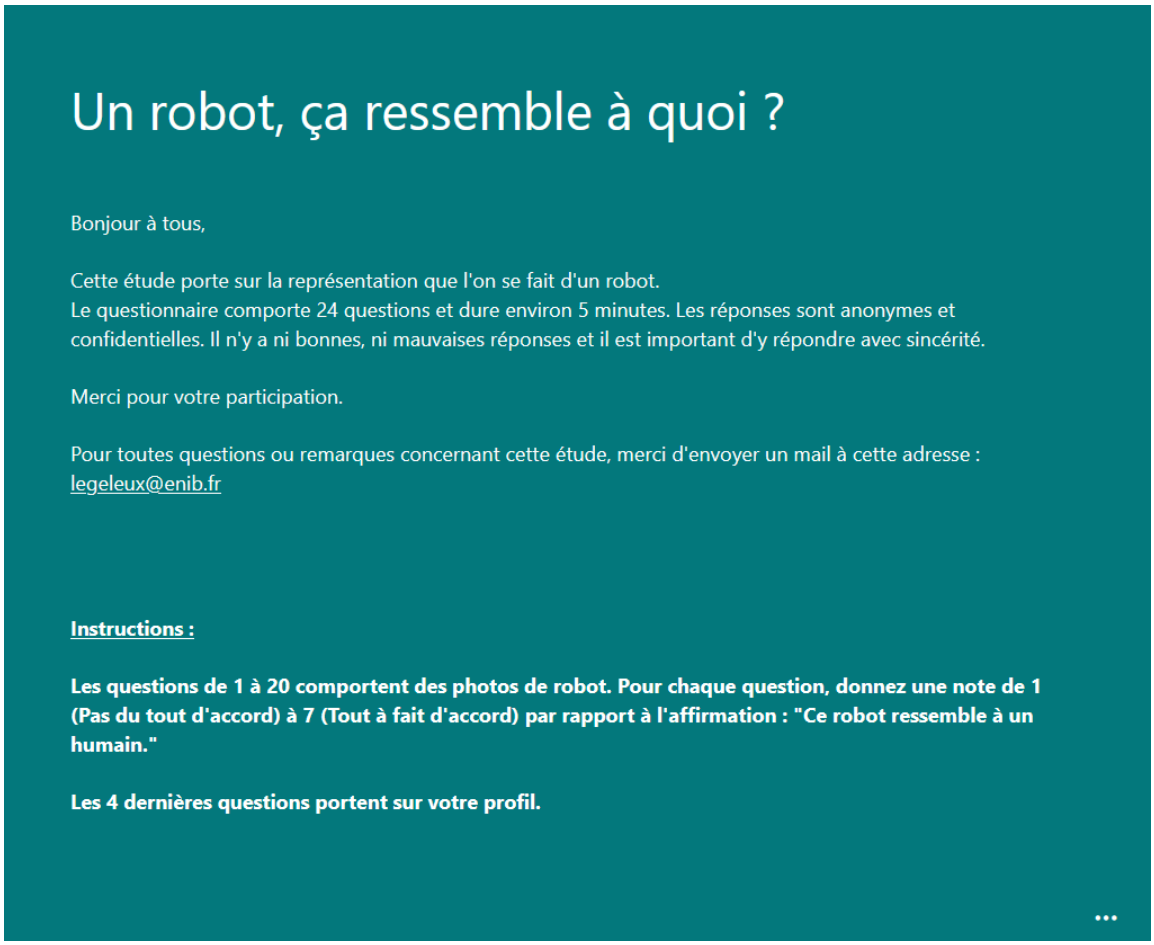


Figure 47 : Aspect visuel du questionnaire de la pré-expérience, texte de consigne

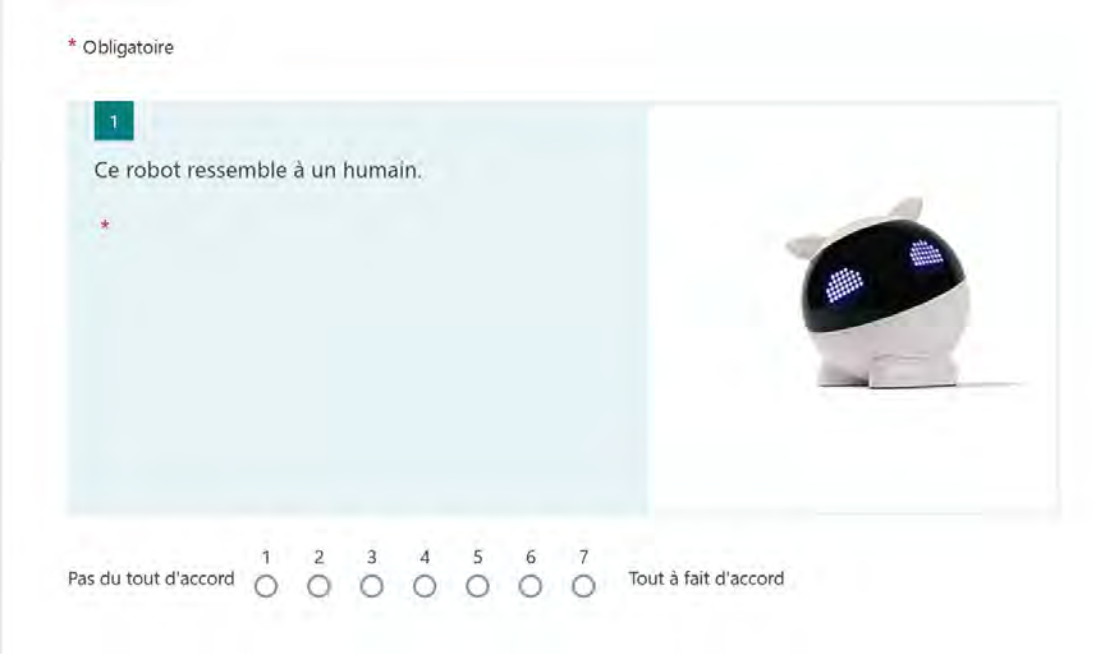


Figure 48 : Aspect visuel du questionnaire de la pré-expérience, exemple d'une question

Le questionnaire est composé de vingt questions portant sur des photos de robot et quatre questions sur le profil des participants. Les questions comportant les photos de robots apparaissent dans un ordre aléatoire pour chaque participant. La question identique pour toutes les photos de robot est la suivant : « Ce robot ressemble à un humain. ». Le participant doit ensuite noter l'affirmation de 1 (Pas du tout d'accord) à 7 (Tout à fait d'accord). Les vingt photos de robot utilisées pour ce questionnaire sont indiquées par Figure 49[12]. Les participants doivent répondre à chacune des vingt questions.

[12] Aibo : Sony, https://us.aibo.com/ ; araignée (Hexapod CR-6) : Robo-Soul, https://fr.aliexpress.com/item/32831830732.html ; Asimo : Honda, https://global.honda/innovation/robotics/ASIMO.html ; aspi-rateur (PCR-3100) : Sichler Haushltsgeräte, https://www.pearl.fr/article/ZX7075/robot-aspirateur-connecte-2500-mah-pcr-3100 ; Atlas : Boston Dynamics, https://www.bostondynamics.com/atlas ; Buddy : Blue Frog Robotics, https://www.bluefrogrobotics.com/robot/ ; Cozmo : Anki, https://www.digitaldreamlabs.com/pages/cozmo ; cuisine (KM-6618) : Rosenstein & Söhne, https://www.pearl.ch/de/kuechenmaschine-km-6618-inkl-umfangreiches-zubehoer.html ; drone (Kargu-2) : STM, https://www.stm.com.tr/en/kargu-autonomous-tactical-multi-rotor-attack-uav ; industriel (IRB-6620) : ABB, https://new.abb.com/products/robotics/industrial-robots/irb-6620 ; Kismet : Massachusetts Institute of Technology, http://www.ai.mit.edu/projects/sociable/baby-bits.html ; Nabaztag : Aldebaran Robotics, https://www.nabaztag.com/# ; Nao : SoftBank Robotics, https://www.softbankrobotics.com/emea/fr/nao ; Pepper : SoftBank Robotics, https://www.softbankrobotics.com/eea/fr/pepper ; ROV (Mini-ROV Observer) : Subsea Tech, https://www.subsea-tech.com/fr/mini-rov-observer/ ; Spot : Boston Dynamics, https://www.bostondynamics.com/products/spot ; TurtleBot : Willow Garage, https://www.turtlebot.com/turtlebot2/ ; voiture (Autonom Shuttle Evo): PME Navya, https://navya.tech/fr/solutions/transport-de-personnes/navette-autonome-pour-le-transport-de-passagers/ ; Winky : Mainbot, https://heywinky.com/ ; YuMi : ABB, https://new.abb.com/products/robotics/collaborative-robots/irb-14000-yumi.

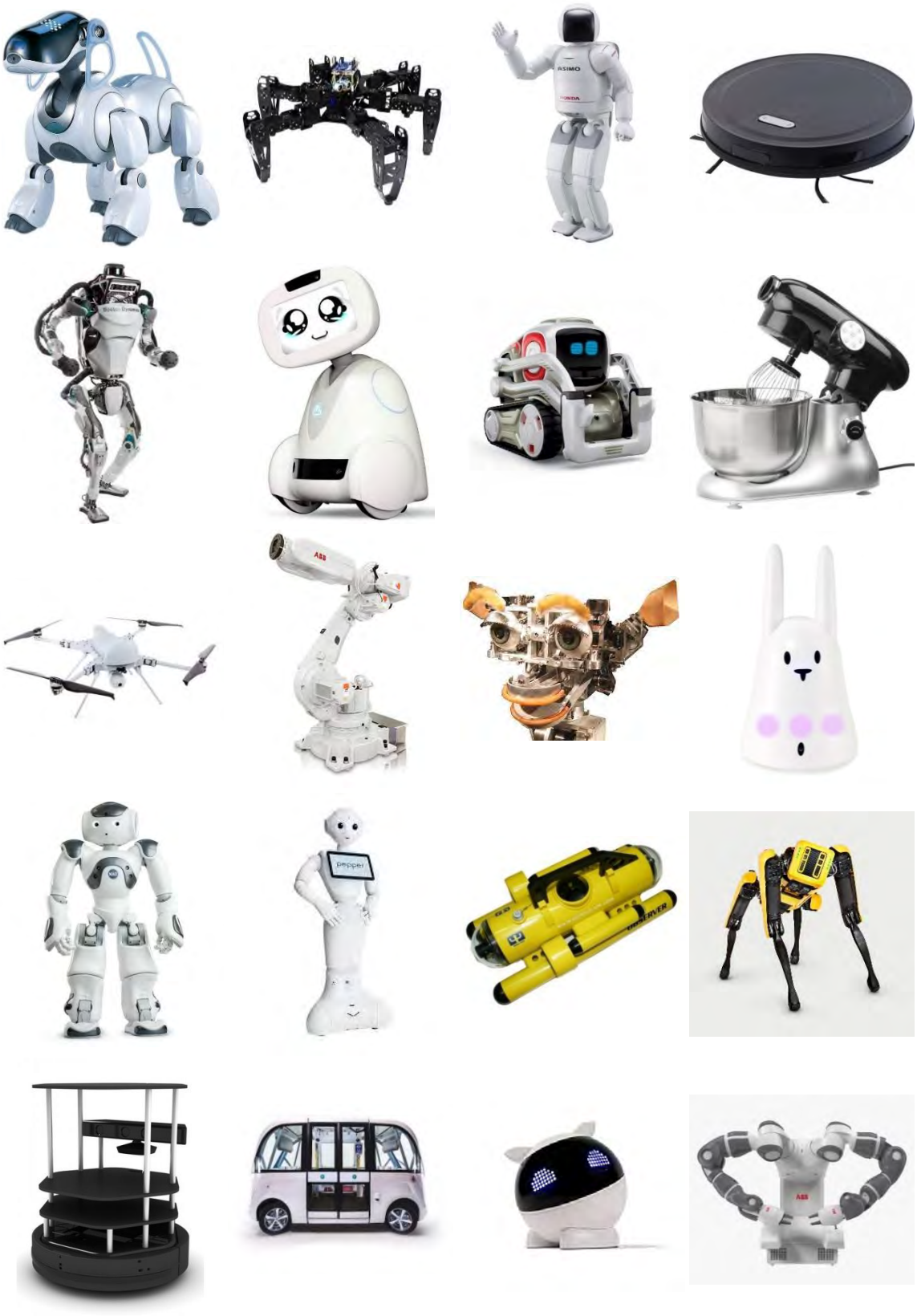


Figure 49 : Photos des robots du questionnaire de la pré-expérience

Ligne par ligne et de gauche à droite : Aibo, araignée, Asimo, aspirateur / Atlas, Buddy, Cozmo, cuisine / drone, industriel, Kismet, Nabaztag / Nao, Pepper, ROV, Spot / TurtleBot, voiture, Winky, YuMi

Les questions portant sur le profil des participants – celles obligatoires sont indiquées avec une astérisque « * » – sont les suivantes :

- « Veuillez indiquer votre sexe : *
  - Féminin
  - Masculin
  - Autre (champ à compléter) » ;
- « Veuillez indiquer votre âge : *
  - (champ à compléter avec un nombre, supérieur ou égal à 18) » ;
- « Veuillez indiquer votre nationalité : *
  - Française
  - Autre (champ à compléter) » ;
- « En quelle année d'étude êtes-vous actuellement à l'ENIB ? *
  - Bac +1
  - Bac +2
  - Bac +3
  - Bac +4
  - Bac +5
  - Double diplôme Master
  - Doctorat (Bac +6)
  - Doctorat (Bac +7)
  - Doctorat (Bac +8)
  - Autre (champ à compléter) ».

Annexe 5

Formulaire de consentement

Formulaire de consentement

Titre du projet : acceptabilité de l'apprentissage par démonstration	
Promoteur : Lab-STICC, Université Bretagne Sud – sous la direction de Dominique DUHAUT **Coordonnée** : dominique.duhaut@univ-ubs.fr	**Investigateur** : Amélie LEGELEUX, doctorante **Coordonnée** : amelie.legeleux@univ-ubs.fr

Avant d'accepter de participer à ce projet de recherche, veuillez prendre le temps de lire et de comprendre les renseignements qui suivent.

Nature et objectif de l'expérience :
Ce travail de recherche s'inscrit dans le projet Prog4Yu visant à développer une méthode de programmation de cobots (robots collaboratifs) pour des utilisateurs non-experts. Le but de cette étude est d'évaluer l'acceptabilité la programmation par démonstration.

Déroulement de la participation :
Cette expérience aura lieu au Centre Européen de Réalité Virtuelle et comportera 4 phases : briefing, formation, tâche, questionnaire. Lors du briefing, une description de l'expérience et de ses phases vous sera proposée. Une formation à la programmation du robot vous permettra de vous familiariser avec le matériel. Vous devrez ensuite réaliser, seul, une tâche qui vous sera présentée. L'expérience se terminera par un questionnaire. La durée de l'expérience est d'environ 30 minutes.

Risques éventuels liés à l'expérience :
Une mauvaise manipulation du robot ou de l'interface de programmation pourrait vous causer des chocs légers (comme n'importe quelle manipulation d'objets du quotidien) ou endommager le robot. Veuillez suivre attentivement la formation et poser toutes les questions que vous souhaitez. Lors de l'expérience, l'investigateur sera dans la salle afin de garantir son bon déroulement.

Je soussigné(e) ..... certifie donner librement mon accord pour participer à l'expérience décrite ci-dessus.

Je comprends que j'ai le droit de refuser de participer à cette recherche ou de retirer mon consentement à tout moment sans encourir aucune responsabilité ni aucun préjudice de ce fait. Mon consentement ne décharge pas les organisateurs de la recherche de leurs responsabilités et je conserve tous mes droits garantis par la loi.

A cet effet, j'autorise expressément le Lab-STICC à conserver les données recueillies durant cette étude. J'ai été informé(e) que mon identité n'apparaîtra dans aucun rapport ou publication et que toute information me concernant sera traitée de façon confidentielle. J'accepte que les données enregistrées à l'occasion de cette étude puissent faire l'objet d'un traitement informatisé non nominatif par le promoteur ou pour son compte. J'ai bien noté que mon droit d'accès prévu par la loi informatique et liberté s'exerce à tout moment auprès de l'unité de recherche.

A Plouzané, le .....

Signature de l'investigateur :

Signature du volontaire :
(précédée de la mention « lu et approuvé »)

Annexe 6

Textes de consigne pour l'expérience

Accueil participant

« Bonjour, je suis Amélie Legeleux. Je te remercie pour ta participation. Merci de me suivre. »

Déplacement vers la salle d'accueil *(sans robot)*

« (Tu peux déposer tes affaires ici.) Voici un formulaire de consentement avant ta participation. Tu n'es pas obligé de faire l'expérience. Tu as choisi de la faire de plein gré. Tu peux arrêter l'expérience à tout moment.
Dans le formulaire, il est indiqué que les risques éventuels liés à l'expérience sont des chocs. Les risques sont les mêmes que pour toute manipulation d'objet du quotidien. Par exemple, se coincer les doigts lorsque l'on ferme une porte ou avec le capot d'un ordinateur portable.
Merci de lire le formulaire attentivement et de signer en bas de la page sur les deux exemplaires. Un exemplaire est pour toi et le second pour moi. »

Formulaire de consentement

« Avant de démarrer l'expérience, merci d'éteindre ou mettre en sourdine ton téléphone portable puis compléter ce questionnaire. En bas de la page, tu ne cliqueras sur le bouton suivant qu'une seule fois et le questionnaire sera fini. »

Questionnaire pré-expérience

« Tu vas interagir avec un cobot, c'est-à-dire un robot collaboratif, afin de réaliser une tâche. Dans un premier temps, une formation te sera proposée afin de comprendre le fonctionnement du robot ainsi que l'interface de programmation. La tâche à réaliser te sera présentée à l'aide d'une vidéo. Tu pourras solliciter mon aide et poser des questions lors de cette phase.

La deuxième phase est consacrée à l'expérience. Tu devras réaliser une autre tâche en autonomie. Elle te sera également présentée avec une vidéo.

Un second questionnaire clôturera l'expérience.

Un temps est réservé à la fin de l'expérience si tu souhaites discuter. Est-ce que tout est clair ? Merci de me suivre, nous allons changer de salle. »

Déplacement vers la salle d'expérience *(avec robot)*

(bleu) : PbD / [vert] : MPp

« *La formation commence maintenant. Voici le robot que tu vas utiliser (**montrer avec la main**) ainsi que son interface de programmation (**montrer avec la main**).*

*Tu vas programmer le robot [afin de lui faire réaliser une tâche] (en lui montrant physiquement comment réaliser une tâche). L'objectif est de [faire] (lui apprendre) la tâche « prendre_gobelet » (**montrer la vidéo de la tâche à réaliser**).*

Pour cela, (tu vas manipuler le bras du robot pour lui montrer le mouvement. Dans un premier temps,) je vais t'expliquer le fonctionnement de l'interface. (Ensuite, je te montrerai la manipulation du robot. Dans un dernier temps, tu devras réaliser la tâche « prendre_gobelet » avec mon aide.)

*L'interface te permet de positionner le robot dans sa position de départ (**montrer le bouton**), c'est-à-dire la position dans laquelle le robot se trouve actuellement.*

*Sur la gauche (**montrer**), tu as la liste des tâches que le robot a apprises. (Ici, le robot ne sait rien faire.)*

*Ensuite (**montrer**), tu as [1] (2) bouton(s) : (enregistrer et) bouger.*

(Le bouton enregistrer te permet de faire une démonstration d'une tâche au robot. Tu peux choisir le nom que tu souhaites pour la tâche. Le nom ne peut contenir que des lettres, chiffres et tiret du bas. Si tu te trompes lors de la réalisation de la tâche, tu peux la refaire avec le même nom et le robot ne retiendra que la dernière.)

Le bouton bouger te permet de faire bouger le robot en fonction de la tâche que tu auras choisi.

Le robot ne dispose pas d'un système de vision qui permet de connaître la position des objets qu'il manipule. C'est pourquoi les objets devront toujours se trouver à la même position. Pense à positionner les objets avant de démarrer le mouvement d'une tâche.

*Je vais te montrer comment manipuler le robot. La tâche 'test' (**montrer**) permet de tester la manipulation du robot. Voici comment le robot peut bouger (**montrer**). Il faut faire attention à ne pas bouger trop rapidement le robot. Il ne faut pas non plus forcer sur ses articulations. A la fin du mouvement, il faut positionner le bras du robot vers sa position de départ. Afin d'ouvrir ou fermer la main (Pepper) / pince (YuMi) du robot, il faut dire 'gauche' (Pepper) / appuyer sur le bouton suivant (YuMi) (**montrer**). Vas-y, entraine-toi à manipuler le robot. »*

Entrainement manipulation du robot

*« Tu vas devoir maintenant programmer le robot afin qu'il réalise la tâche « prendre_gobelet » que tu as vu dans la vidéo. Tu peux revoir la vidéo quand tu le souhaites. Vas-y, je vais te guider quand tu auras besoin. (**démarrer chronomètre, prendre mesures**) »*

Entrainement programmation du robot

*« Tu as maintenant réussi à faire la tâche « prendre_gobelet ». As-tu des questions avant de terminer la formation ? (**arrêter chronomètre**) »*

Fin de la formation

« *La formation est maintenant terminée. L'expérience commence. Tu vas devoir réaliser une nouvelle tâche en autonomie. Voici la vidéo de la tâche à réaliser (**montrer la vidéo**). Comme lors de la formation, tu peux revoir la vidéo de la tâche à réaliser quand tu le souhaites. Le verre à utiliser se trouve ici (**montrer**). Je resterai dans la salle afin de m'assurer du bon déroulement de l'expérience. Je ne pourrais plus répondre à tes questions ni t'aider, exception si le robot ou le programme plante. Lorsque tu estimes avoir terminé, merci de me le signaler. (**démarrer chronomètre, prendre mesures, arrêter chronomètre**)* »

Fin de la tâche

« Merci de me suivre, on va changer de salle.

Merci maintenant de compléter la suite du questionnaire. De la même manière que précédemment, ne clique qu'une seule fois sur le bouton suivant. Merci de répondre le plus sincèrement possible, il n'y a pas de bonnes ou mauvaises réponses. »

Questionnaire post-expérience

« L'expérience est terminée. Je te remercie d'avoir pris part à cette expérience ainsi que pour ton sérieux. Merci de ne pas communiquer sur l'expérience à tes amis qui peuvent être de futurs participants afin de ne pas biaiser les résultats. Si tu as des questions, souhaite discuter, je peux maintenant te répondre. »

Discussion

Annexe 7

Questionnaire socio-démographique

Au début du l'expérience sur l'acceptabilité de la programmation par démonstration et sur l'impact de l'anthropomorphisme, les participants remplissent un questionnaire socio-démographique. Ce questionnaire, disponible sur format numérique avec l'outil Google Forms, disposait du design illustré par la Figure 50.

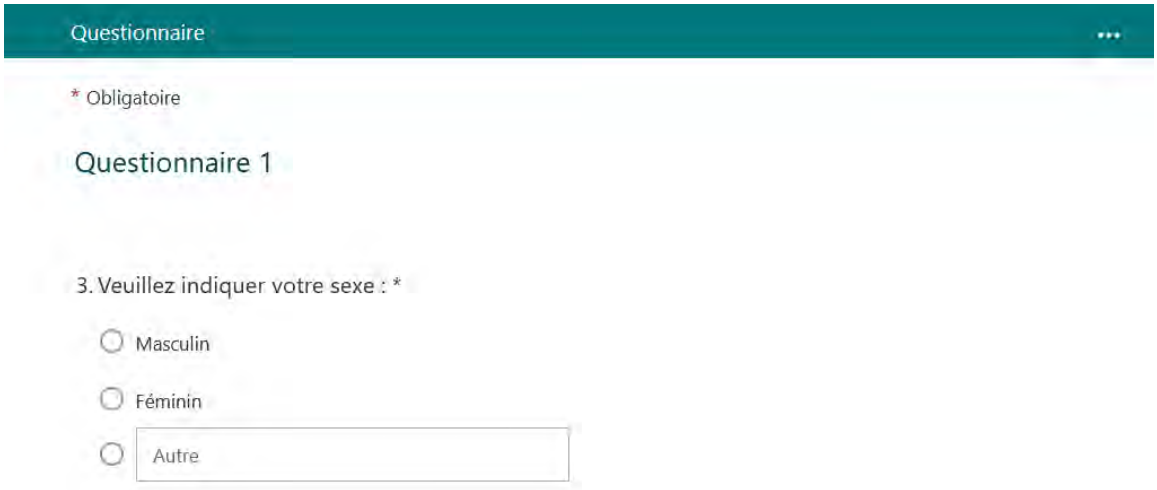


Figure 50 : Aspect visuel du questionnaire socio-démographique de l'expérience sur l'acceptabilité de la programmation par démonstration et l'impact de l'anthropomorphisme, exemple d'une question

Les questions du questionnaire, dont celles avec une astérisque « * » étant obligatoires, sont regroupées ci-dessous. La question sur le ressenti du participant accepte plusieurs réponses, d'où l'utilisation de la puce « □ ».

- « Veuillez indiquer votre sexe : *
  - Féminin
  - Masculin
  - Autre (champ à compléter) » ;

- « Veuillez indiquer votre âge : *
  - (champ à compléter avec un nombre) » ;
- « Veuillez indiquer votre catégorie socio-professionnelle : *
  - Étudiant
  - Étudiant en alternance
  - Salarié en reprise d'étude
  - Autre (champ à compléter) » ;
- « Quel diplôme êtes-vous en train de préparer et quelle est sa spécialité ? *
  - Bac Pro – MELEC
  - Licence Pro – PGI MECA
  - Licence Pro – SARII
  - Autre (champ à compléter) » ;
- « En quelle année d'étude êtes-vous actuellement ? *
  - Seconde
  - Première
  - Terminale
  - Bac +1
  - Bac +2
  - Bac +3
  - Autre (champ à compléter) » ;
- « Avez-vous déjà travaillé en entreprise ? *
  - Oui
  - Non » ;

- « Si oui, combien de temps ? Quel(s) poste(s) et quelle(s) mission(s) ?
  - (champ à compléter) » ;
- « Quel est votre niveau de connaissances en programmation informatique ? *
  - Expert
  - Intermédiaire
  - Novice
  - Non initié » ;
- « Avez-vous déjà rencontré un robot ? *
  - Oui
  - Non » ;
- « Si oui, dans quel(s) contexte(s) ?
  - (champ à compléter) » ;
- « Avez-vous déjà utilisé un robot ? *
  - Oui
  - Non » ;
- « Si oui, dans quel(s) contexte(s) ?
  - (champ à compléter) » ;
- « Avez-vous des connaissances en robotique ? *
  - Beaucoup
  - Moyen
  - Un peu
  - Aucune » ;

- « Quel est votre ressenti à l'idée de rencontrer et d'utiliser un robot ? *
  - □ Impatient
  - □ Content
  - □ Neutre
  - □ Stressé
  - □ Anxieux
  - □ Autre (champ à compléter) » ;
- « Si vous souhaitez être contacté pour connaître les résultats de l'expérience, vous pouvez laisser vos coordonnées :
  - o (champ à compléter) ».

Annexe 8

Questionnaire acceptabilité

Le questionnaire pour l'évaluation de l'acceptabilité de la programmation par démonstration et de l'impact de l'anthropomorphisme est présenté dans la Section 4.3.2.4. Les participants ont complété le questionnaire sous format numérique, développé avec l'outil Google Forms. La Figure 51 montre l'aspect visuel du questionnaire.

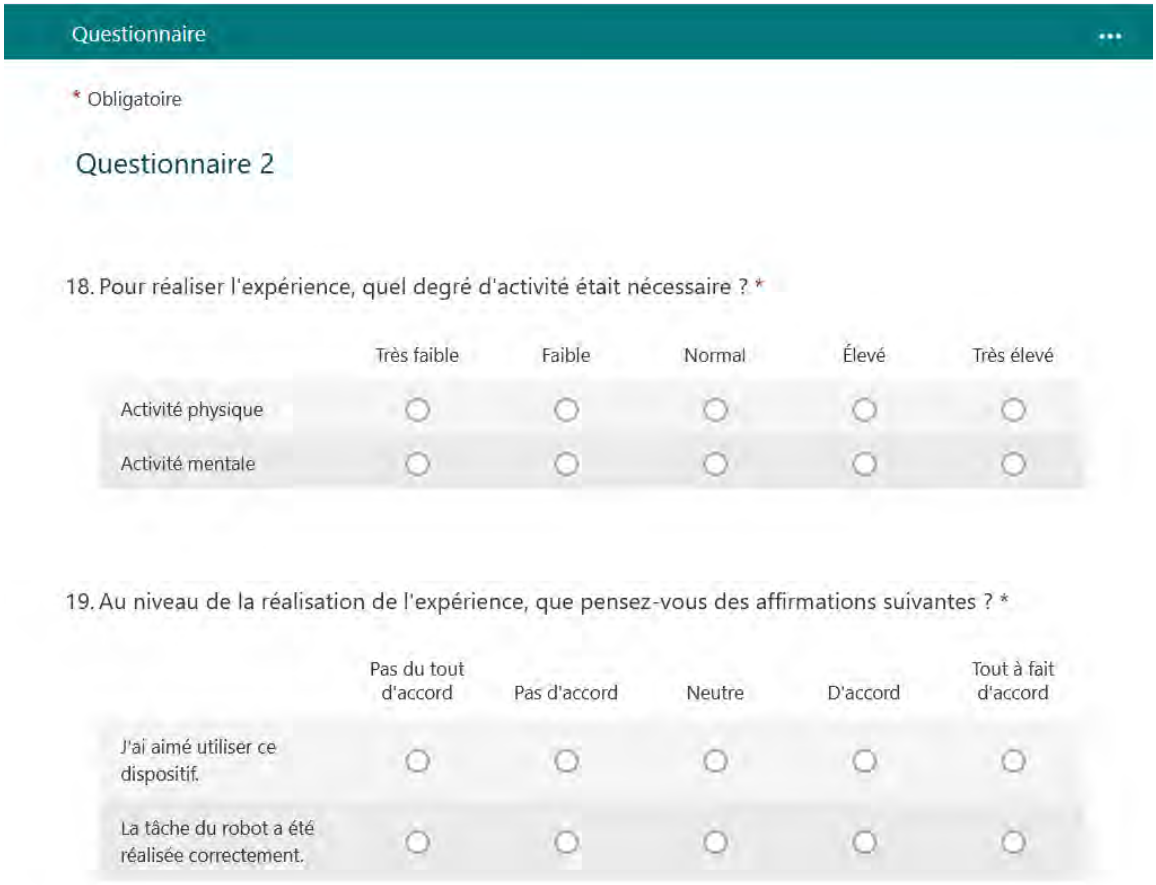


Figure 51 : Aspect visuel du questionnaire de l'expérience sur l'acceptabilité de la programmation par démonstration et l'impact de l'anthropomorphisme, exemple de questions

Afin de faciliter le remplissage du questionnaire, les questions sont regroupées par thématiques. Toutes les questions sont composées d'une affirmation que le participant doit noter sur une échelle de 5 items (« Pas du tout d'accord », « Pas d'accord », « Neutre », « D'accord », « Tout à fait d'accord »). Seules les deux premières affirmations disposent des items suivants : « Très faible », « Faible », « Normal », « Élevé », « Très élevé ». Les questions, toutes obligatoires avec l'astérisque « * », sont présentées ainsi :

- « Pour réaliser l'expérience, quel degré d'activité était nécessaire ? *
  - Activité physique
  - Activité mentale » ;
- « Au niveau de la réalisation de l'expérience, que pensez-vous des affirmations suivantes ? *
  - J'ai aimé utiliser ce dispositif.
  - La tâche du robot a été réalisée correctement. » ;
- « Que diriez-vous de la facilité d'utilisation du dispositif ? *
  - Ce dispositif est facile à apprendre.
  - Ce dispositif est intuitif.
  - Ce dispositif est facile à utiliser » ;
- « Quel est votre ressenti par rapport au robot ? *
  - J'aime collaborer avec ce robot.
  - Je me suis senti en sécurité lors de l'interaction avec le robot.
  - La taille du robot ne m'a pas intimidé.
  - Je sentais que je pouvais compter sur le robot pour faire ce qu'il avait à faire ;
  - Le travail avec un robot me stresse.
  - Le travail avec un robot me rend anxieux. » ;

- « Si vous projetez l'utilisation du dispositif dans votre futur travail, que pensez-vous des affirmations suivantes ? *
  - Ce dispositif me serait très utile pour mon travail.
  - Ce dispositif diminuerait ma fatigue dans mon travail.
  - Ce dispositif diminuerait mes erreurs dans mon travail.
  - Ce dispositif me permettrait d'accomplir mes tâches plus rapidement au travail.
  - L'utilisation de ce robot est une bonne idée.
  - Je souhaiterais utiliser ce robot dans mon travail.
  - Je sens que le robot va me remplacer au travail. ».

En plus de ces questions, d'autres affirmations sur l'intelligence et la ressemblance du robot avec un humain sont également proposées. Une dernière question porte sur le niveau d'amusement ressenti lors de l'expérience. Ces dernières questions sont présentées en ces termes :

- « Lors de l'expérience, vous estimez que : *
  - Le robot ressemblait à un humain.
  - Le robot a appris comme un humain.
  - Le robot pensait comme un humain.
  - Le robot était intelligent. » ;
- « Avez-vous trouvé l'expérience amusante ? *
  - (échelle avec cinq étoile) » ;
- « Si vous souhaitez laisser un commentaire :
  - (champ à compléter) » ;

Annexe 9

Résultats des mesures subjectives

Le Tableau 25 regroupe les résultats des mesures subjectives de l'expérience portant sur l'acceptabilité de la programmation par démonstration et l'impact de l'anthropomorphisme.

Tableau 25 : Résultats des mesures subjectives

Moyenne et écart-type pour chaque mesure ; durée en minutes et secondes

	Nombre de démonstration		Nombre de reproduction		Nombre de visionnage	
	Formation	Expérience	Formation	Expérience	Formation	Expérience
MPp	-	-	1,000 (0,000)	1,250 (0,463)	0,000 (0,000)	0,125 (0,354)
PbD	2,125 (1,356)	3,625 (3,926)	1,875 (1,126)	3,375 (4,340)	0,625 (0,518)	0,125 (0,354)

	Durée		Taux de succès	
	Formation	Expérience	Formation	Expérience
MPp	01 : 42 (0,000)	01 : 20 (0,000)	91,0% (24,66)	96,9% (8,839)
PbD	06 : 42 (0,002)	06 : 39 (0,004)	90,6% (24,52)	87,5% (35,36)

	Nombre de démonstration		Nombre de reproduction		Nombre de visionnage	
	Formation	Expérience	Formation	Expérience	Formation	Expérience
Pep.	3,000 (1,155)	6,000 (4,546)	1,750 (1,165)	3,500 (4,276)	0,250 (0,463)	0,000 (0,000)
Yu.	1,250 (0,957)	1,250 (0,500)	1,125 (0,354)	1,125 (0,354)	0,375 (0,518)	0,250 (0,463)

	Durée		Taux de succès	
	Formation	Expérience	Formation	Expérience
Pep.	03 : 53 (0,002)	05 : 27 (0,005)	81,9% (32,04)	84,4% (35,20)
Yu.	04 : 31 (0,003)	02 : 32 (0,001)	99,8% (0,707)	100% (0,000)

Table des matières

Titre : Programmation de cobots : de l'apprentissage de trajectoire à leur acceptabilité

Mots-clés : Apprentissage par démonstration, Cobots, Interaction Humain-Robot, Acceptabilité

Résumé : Les robots, nouvelles machines présentes dans nos vies quotidiennes, se diversifient. De récents progrès ont permis l'émergence des cobots, robots qui collaborent avec les êtres humains. Ce nouveau type de robots, contrairement aux robots traditionnels, nécessite l'expertise des opérateurs pour fonctionner. L'apprentissage par démonstration offre alors une méthode inédite de programmation. L'opérateur peut manipuler le bras du robot pour lui apprendre le mouvement à réaliser.

Cette thèse propose de développer cet apprentissage par démonstration à travers plusieurs prismes de lecture : les données, l'apprentissage et l'acceptabilité.

Avant d'être utilisées par l'apprentissage, les données sont obtenues à l'aide de démonstrations kinesthésiques puis alignées temporellement et filtrées afin d'en améliorer la qualité. Un nouvel algorithme d'apprentissage avec des données pondérées est proposé avec une architecture logicielle générique, fonctionnant sur de nombreuses plateformes robotiques. Finalement, l'acceptabilité de la programmation par démonstration est vérifiée par une expérience avec des participants, potentiels futurs utilisateurs des cobots. L'impact de l'anthropomorphisme est également pris en considération.

Les différents résultats obtenus permettent d'envisager l'implantation des cobots dans l'industrie du futur : de l'acquisition des données en passant par l'apprentissage tout en vérifiant l'acceptabilité et la bonne compréhension de la programmation par les utilisateurs.

Title: Cobot's programming: from trajectory learning to their acceptability

Keywords: Learning from Demonstration, Cobots, Human-Robot Interaction, Acceptability

Abstract: Robots, new machines in our daily lives, diversify. Recent progress has made the rising of cobots possible. Cobots are robots which collaborate with human beings. Contrary to traditional robots, this new type of robot requires the expertise of an operator to run. The Learning from Demonstration creates an original way of programming. The operator can manipulate the robot's arm in order to teach it the movement to realize.

The present thesis proposes an improvement of this learning through these three axes: the data processing, the learning, and the acceptability.

Before being used by the learning, data is retrieved during the kinesthetic demonstration, then temporally aligned, and filtered to improve its quality. A novel learning algorithm with weighted data is proposed with generic software architecture allowing it to run on multiple robotics platforms. Finally, the acceptability of the Programming by Demonstration is evaluated with an experiment whose participants are potentially future users of cobots. The impact of the anthropomorphism is also considered.

The different outcomes permit to consider the introduction of cobots in the industry of the future: from the data acquisition to the learning while evaluating the acceptability as well as the understanding of this type of programming by users.