Prédiction de l'intention lors d'interaction physique avec le robot iCub, à l'aide de primitives de mouvements probabilistes

Oriane Dermy [1,*], **Alexandros Paraschos** [2,4], **Marco Ewerton** [2], **Jan Peters** [2,3], **François Charpillet** [1] **and Serena Ivaldi** [1]

[1] *Inria, Villers-lés-Nancy, 54600, France;*
Université de Lorraine, Loria, UMR7503, Vandoeuvre, 54500, France;
CNRS, Loria, UMR7503, Vandoeuvre, 54500, France
name.surname@inria.fr
[2] *TU Darmstadt, Darmstadt, Germany*
[3] *Max Planck Institute for Intelligent Systems, Tubingen, Germany*
[4] *Data Lab, Volkswagen Group, 80805 Munich, Germany*

Correspondence*:
Corresponding Author
oriane.dermy@inria.fr

ABSTRACT

Ce papier présente un logiciel open-source permettant de prédire l'intention d'un utilisateur interagissant avec le robot humanoid iCub. Notre but est de permettre au robot d'inférer l'intention de son partenaire humain lors de tâche collaborative, en prédisant la continuation de trajectoire attendu par le partenaire. Cette capacité est nécessaire pour que le robot développe des comportements d'anticipation, ce qui est crucial dans les scénarios de collaboration homme-robot, tel que la co-manipulation, l'assemblement ou le transport coopératif. Pour cela, nous proposons une approche permettant au robot d'acquérir des capacités basique de reconnaissance de l'intention, à l'aide de primitive de mouvement probabilistes (ProMPs). Cette méthode (ProMPs) permet de représenter, généraliser et reproduire des mouvements moteurs compliqués. Dans cette étude, le robot apprend un ensemble de primitives de mouvements à partir de plusieurs démonstrations fournies par le partenaire qui guide le robot physiquement. Lors de l'apprentissage, le scénario collaboratif est modélisé à partir des démonstrations fournies par le partenaire. Lors de la reproduction de la tache collaborative, le partenaire initie le mouvement du robot physiquement, et celui-ci utilise son apprentissage pour reconnaître l'intention de son partenaire. Dès les premières observations de ce mouvement, le robot peut en plus d'inférer l'intention de son partenaire (but), compléter le mouvement et ce, même si le partenaire arrête l'interaction physique avec le robot. Notre approche a été évalué en simulation et sur le robot réel. En simulation, le iCub est guidé par le partenaire à l'aide d'un appareil haptique nomé Géomagic Touch. Dans l'expérience réelle, le partenaire interagit avec le robot en lui attrapant ses mains et en les guidant. Deux expériences ont été effectués sur le robot réel : l'une où le robot effectue de simples trajectoires pour attraper des objects, et l'autre inspiré de tâche de tri collaboratif. Le logiciel implémentant notre approche est accessible sur la plateforme GitHub. De plus, des tutoriels et des vidéos sont fournis.

Mots clefs : robot, prédiction, intention, interaction, modèles probabilistes

1 INTRODUCTION

Actuellement, les industries n'utilisent que des robots industriels à base fixe, pour des tâches répétitives ne nécessitant pas d'interaction avec l'humain. Dans le future, nous imaginons que les robots humanoïdes pourront aussi effectuer ces tâches pour assister les travailleurs là où des robots à base fixe ne peuvent pas être installés, et où les robots ont besoin de s'adapter aux mouvements des travailleurs, tel que pour la fabrication d'avions (**?**). Pour collaborer avec des humains, les robots ont besoin de pouvoir prédire l'intention de leur partenaire. En effet, les travailleurs aimeraient n'avoir qu'à guider le commencement du mouvement des robots ou lorsqu'il s'agit de lever un objet en collaboration, de commencer eux-même le mouvement sans avoir à commander/programmer les robots. Le robots devineraient alors l'intention de leur partenaire le plus vite possible, afin de compléter la tâche attendu sans avoir besoin d'assistance supplémentaire. Plus précisément, les travailleurs n'auraient qu'à guider les bras des robots dans la direction du but, puis de les lâcher dès que ceux-ci montrent qu'ils sont capables d'atteindre le but voulu par eux-mêmes.

Ce scénario est particulièrement intéressant pour les industries (**?**), où les travailleurs n'auraient qu'à initier sans effort un mouvement avec les robots, puis ceux-ci seraient capable de déplacer correctement des objets lourds ou peu maniables, et de les placer à la position désirée. Il y a une multitude de scénarios où ces tâches collaboratives peuvent être initié par un humain et finalisé par un robot, que ce soit dans des services à la personne ou pour la production : assemblage de pièces, tris d'objets, soudure, déplacement d'objets nécessitant ou non d'être à plusieurs, etc. Dans tous ces exemples, le robot a besoin de prédire le but de chaque action et la trajectoire correspondante que le partenaire souhaite exécuter.P Pour effectuer cette prédiction, les robots doivent utiliser toutes qu'ils ont à leur disposition en utilisant leurs senseurs. Ces informations proviennent aussi de leurs expériences passées (a priori), venant par exemple de session d'apprentissage passé, que ce soit en imitant leur partenaire ou en collaborant avec eux. En plus d'être capable d'utiliser toutes ces données, les robots doivent comprendre et modéliser le comportement des humains, afin de développer cette capacité de prédiction de l'intention et l'autonomie liée à cette compréhension (**?**). La prédiction de l'intention du partenaire, passe par l'identification de la tâche courante et la prédiction de la trajectoire à effectuer pour atteindre ce but. Dans la littérature sur l'interaction homme-robot, beaucoup de mots-clefs sont associés à cette habilité de prédiction : inférence, estimation du but, lisibilité, reconnaissance de l'intention et prédiction. L'anticipation correspond à la capacité du robot à choisir correctement la tâche à effectuer dans une situation donnée (**?**). Pour cela, le robot doit prédire l'effet de son action, comme étudié avec le concept d'affordance (**???**). Il doit aussi être capable de prédire l'intention du partenaire, ce qui revient à estimer le but du partenaire (**??**). Finalement, il doit prédire les futures événements ou états, c'est à dire être capable de simuler l'évolution du système couplé homme-robot, comme cela est fait dans le cas de modèles de contrôle prédictifs (**??**) ou de la planification human-aware (**??**).

Il a été montré que faire des mouvement lisibles (**??**) aide l'interaction entre les personnes, puisque cela permet d'améliorer l'estimation mutuelle de l'intention et donc d'améliorer l'efficacité de la collaboration.

L'anticipation necessite donc l'habilité de visualiser ou de prédire le future état désiré, c'est à dire là ou le partenaire souhaite aller. Prédire l'intention de l'utilisateur est souvent formulé comme la prédiction du but de l'action humaine, ce qui signifie que le robot doit être capable de prédire au moins le but du partenaire quand deux partenaires s'engage dans une action conjointe. Pour faire une telle prédiction, une approche classique est de considérer chaque mouvement comme une instance d'une action particulière, ou d'une primitive de mouvement dirigé vers un but.

Dans la dernière décennie, beaucoup de méthodes ont été proposées afin de représenter des primitives de mouvements, tel que les "Gaussian Mixture Models" (GMM, les modèles de mixture gaussiennes) (**??**), les "Dynamic Movement Primitives" (DMP, les primitives de mouvements dynamiques) (**?**), ou encore récemment les "Probabilistic Dynamic Movement Primitive" (PDMP **?**, les primitives de mouvement dynamiques probabilistes) et les "Probabilistic Movement Primitives" (ProMP, primitives de mouvements probabilistes) (**?**).

Une review plus approfondis sur ces différentes techniques est disponible dans (**?**). Les différentes techniques d'apprentissage ont été appliqué dans beaucoup de scénario, tel que permettre à un robot à jouer au ping-pong, écrire des nombres, éviter des obstacles pour des mouvement de saisie-placement (pick and place), etc. Dans tous ces scénarios, les partenaires humains effectuent généralement les démonstrations (i.e., ils réalisent eux même les mouvements permettant d'effectuer les différentes tâches) soit en guidant manuellement le robot, soit en utilisant la téléopération, en suivant le paradigme classique d'apprentissage par démonstration. Certaines études de ce type ont aussi été appliqué au robot humanoïde iCub. Par exemple, dans **?** ils utilisent les DMPs pour adapter en ligne le mouvement dirigé vers un but afin d'éviter les obstacles variables rencontré par le bras robotique, tandis que dans **?**, ils utilisent les ProMPs afin d'apprendre comment le robot doit incliner un râteau en utilisant les informations des forces et moments.

Parmi toutes les techniques précédemment énoncées, les ProMPs nous semble être le plus prometteur afin de permette la reconnaissance de l'intention et l'anticipation de mouvement pour des scénario de collaboration homme-robot. Ils ont l'avantage, par rapport aux autres méthodes, d'avoir une modélisation de mouvements qui capture la variabilité des démonstrations humaines. Elles ont aussi différentes propriétés décrites dans **?**, telles que la co-activation et le couplage de primitives, ainsi que le redimensionnement temporel. Les ProMPs ont déjà été utilisés dans la coordination homme-robot pour générer des trajectoires robotiques appropriés en réponse à des trajectoires humaines initiées. (**?**). Contrairement aux DMPs, les ProMPs ne contiennent pas d'attracteur sur la position finale de la trajectoire, donc n'ont pas besoin d'information sur cette position finale et peuvent faire varier la position finale du mouvement tout en respectant les primitives apprises. [1] De plus, la méthode ProMPs est plus performante lorsque les données reçues par les capteurs sont bruitées ou variables, comme montré dans (**?**). [2] Une étude récente **?** présente une méthode appelée PDMP (Probabilistic Dynamic Movement Primitive). Cette méthode améliore les DMPs en y ajoutant des propriétés probabilistes afin de mesurer la vraisemblance qu'une primitive de mouvement est exécutée correctement et afin d'inférer des mouvements à partir de mesures de capteurs. Malgré cette amélioration, les PDMPs n'ont pas la propriété de généralisation des données **/*(? ?)*/** et les mouvements peuvent dévier de manière arbitraire hors des démonstrations. Cette dernière différence est critique lorsque l'on utilise des robots humanoïdes. Par exemple, à cause d'une telle déviation, le robot pourrait heurter quelque chose durant un mouvement, ou faire tomber un objet dans le cas de mouvement précis durant lequel le robot transporte un objet. C'est pourquoi, nous avons considéré que l'utilisation de ProMPs est plus adaptés à nos applications.

Dans ce papier, nous présentons notre approche afin de prédire, dans le cadre d'interaction et de collaboration physique homme-robot, l'intention du partenaire humain, basée sur les primitives de mouvement probabilistes (Probabilistic Movement Primitives, ProMPs) (**?**). Nous présentons aussi le logiciel open source associé qui permet d'implémenter la méthode à notre robot iCub.

1. Il y a des applications dans lesquelles la convergence des mouvements du robot vers un but unique et précis est une propriété désirable voir nécessaire. Cependant, il s'agit d'une hypothèse qui empêche de généraliser la méthode à différentes actions, ce qui est une autre raison pour laquelle nous préférons l'utilisation des ProMPs.

2. Pour avoir une étude détaillée des différences entre ProMPs et DMPs dans le cadre d'apprentissage de primitives d'interaction et de prédiction, voir (**?**).

Pour illustrer la technique utilisée, nous présentons dans ce papier un exemple de problématique qui consiste à permettre au robot de finir un mouvement initié par un utilisateur par guidage physique du bras du robot. à partir de peu d'observation de ce mouvement commun entre l'homme et le robot, mouvement supposé appartenir à une primitive de mouvement d'une certaine tâche, le robot doit reconnaître quel type de tâche est guidé par l'utilisateur et prédire la continuation de la trajectoire à effectuer afin d'accomplir le mouvement de manière autonome dès que l'utilisateur relâche le bras robotique. [3]

Pour effectuer ces tâches, le robot doit tout d'abord apprendre les primitives de mouvements qui y sont associées. S l'aide de la méthode ProMPs, nous décrivons ces primitives à l'aide de distributions de démonstrations décrites par un modèle probabiliste, plutôt que d'apprendre une trajectoire "moyenne" unique. Durant l'interaction, l'utilisateur commence par guider physiquement le robot afin qu'il effectue la tâche. En même temps, le robot collecte des données d'observation. Il utilise alors sa connaissance a-priori provenant des ProMPs qu'il a apprises, afin de calculer une prédiction de la tâche désirée ainsi que de la trajectoire future qu'il doit effectuer afin d'atteindre le but. Le concept de ce problème est représenté dans la Figure 1. Dans la partie supérieur, on représente l'étape d'apprentissage d'une primitive : le robot y est guidé physiquement par le partenaire durant tout le mouvement permettant d'effectuer une certaine tâche, et ce plusieurs fois. Ainsi, le robot observe plusieurs démonstrations de ce même mouvement. Durant ces mouvements, le robot collecte à la fois des mesures cinématiques (*e.g.*, positions Cartésienne) et dynamiques (*e.g.*, couple). Ces N trajectoires constituent une base pour l'apprentissage des primitives. Cet apprentissage consiste à calculer la distribution de paramètres $\boldsymbol{\omega}$ qui modélisent la trajectoire. On appelle la distribution ainsi apprise la *distribution à priori*. Dans le cas où le robot apprend différentes tâches, ce processus d'apprentissage est répliqué afin d'avoir une ProMP pour chaque tâche.

Le bas de la figure représente l'étape d'inférence. A partir d'un mouvement initié et guidé par l'utilisateur humain, le robot commence par identifier quelle ProMP correspond le plus (*i.e.*, quelle primitive l'utilisateur exécute parmis les différentes apprises). Puis, il estime la trajectoire future à partir du mouvement initié et de la distribution préalable. Pour ce faire, il calcule les paramètres $\boldsymbol{\omega}^*$ correspondant à la *distribution postérieur*. La trajectoire correspondante est alors utilisée par le robot afin de finir le mouvement puis la tâche de manière autonome, sans necessité l'aide de l'utilisateur.

Dans ce papier, nous décrivons à la fois la théorie et le logiciel sur lesquels se base la capacité du robot à prédire et effectuer les tâches et mouvements associés de manière autonome. Le logiciel est actuellement implémenté en Matlab et C++ ; il est open-source, accessible sur `github` :

`https://github.com/inria-larsen/icubLearningTrajectories`

De plus, il a été testé à la fois en simulation (à l'aide du simulateur Gazebo contenant une simulation du robot iCub) et sur le robot réel. En simulation, pour que l'utilisateur puisse guider physiquement le robot simulé, nous utilisons un appareil haptique nommé "Geomagic Touch", capable de faire ressentir à l'utilisateur une simulation des forces que le robot simulé exerce. [4] ; sur le robot réel, l'utilisateur attrape physiquement l'avant bras du robot réel pour le guider.

3. Ici, les tâches sont encodés par les primitives, primitives qui correspondent à des trajectoires : il s'agit d'une approche classique de technique d'apprentissage robotique et en générale des techniques basés sur des primitives. Il s'agit d'une hypothèse simplificatrice, mais cela permet de représenter plusieurs tâches différentes telles que : pointage, déplacement vers un but, attrapage d'objets, mouvement de regard, etc.

4. Dans notre expérience, nous n'utilisons pas cette fonctionalité de retour de forces. Nous utilisons cet appareil afin de diriger manuellement le bras du robot dans le simulateur. Ainsi, nous utilisons cet appareil comme un joystick plus naturellement manipulable.

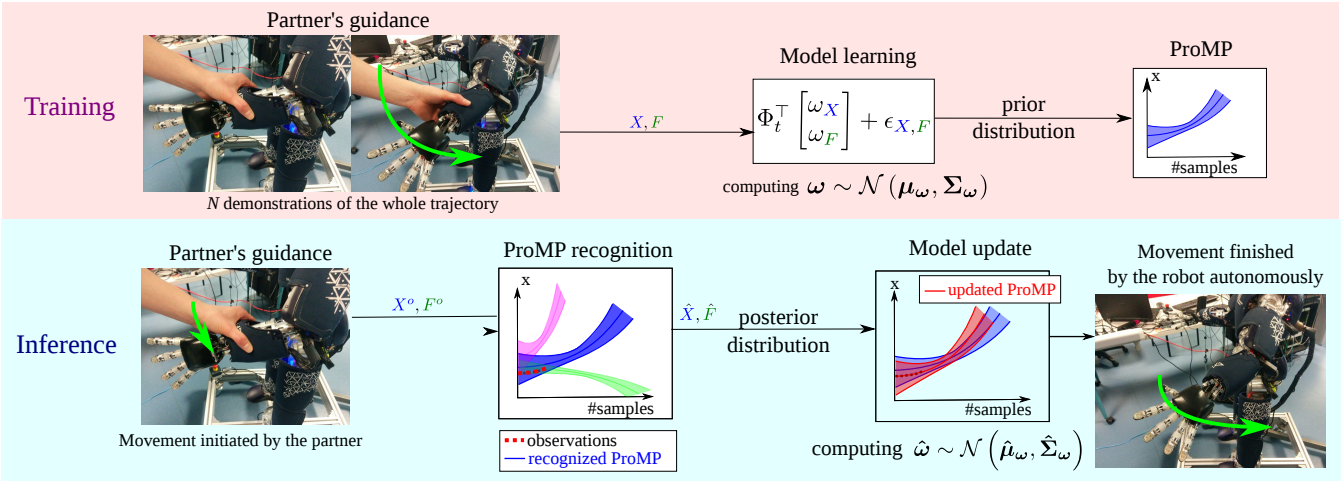

FIGURE 1. Concepte d'utilisation des ProMPs dans lequel le robot prédit la trajectoire a effectuer dans le cadre de tâche collaborative. En haut : phase d'apprentissage, où les ProMPs sont apprises à partir de plusieurs démonstrations guidés par un humain. Bas : phase d'inférence phase (en line), où le robot reconnaît, à partir d'un mouvement initié par son partenaire, la ProMP courante et où il prédit l'intention de son partenaire, *i.e.*, l'évolution future de la trajectoire initiée.

Nous fournissons aussi un exemple pratique du logiciel, permettant de répondre à des problèmes classiques. Dans l'exemple, les mesures des trajectoires correspondent à la position cartésienne de l'actuateur ainsi que les forces qui y sont exercées.

Notons notamment que ce type de mesures diffère de celles utilisées dans les autres études basées sur les ProMPs (**?**), où il s'agissait des valeurs articulaires. L'utilisation de positions Cartésiennes plutôt que des valeurs articulaires nous permet d'exploiter la redondance du bras robotique afin d'effectuer une certaine tâche dans un espace 3D, ainsi que de permettre au robot de bouger son bras de manière plus naturelle pour effectuer la trajectoire voulue (en effet, lors du déplacement du bras robotique, le partenaire ne bouge que l'avant bras du robot, ce qui limite les possibilités de mouvement du robot). Du point de vu contrôle du iCub, ce choix implique que le iCub définisse ses couples articulaires afin d'effectuer les mouvements à l'aide d'un contrôleur cartésien (**?**), plutôt que d'utiliser un contrôleur articulaire. En ce qui concerne les forces, on se base sur une estimation basée sur une modélisation de la dynamique du robot, et qui utilise les valeurs des capteurs de couple 6D (**??**). Tous les détails des expériences sont présentés dans ce papier et dans le tutoriel du logiciel.

Pour résumer, les contributions de ce papier sont les suivants :

—une description de la théorie utilisée dans ce logiciel basée sur la méthode ProMPs et permettant de prédire la trajectoire et le but désirés par l'utilisateur du robot lors d'interaction physique entre le partenaire et le robot. Elle fournie notamment les fonctionnalités suivantes : reconnaissance de la tâche courante, estimation de la durée de la tâche, prédiction de la trajectoire future. ;

—une étude expérimentale concernant comment l'information des forces et moments peuvent améliorer l'estimation de la durée/vitesse de la trajectoire initiée ;

—un software open-source permettant la reconnaissance de l'intention et son application avec le robot iCub, que ce soit en simulation ou avec le robot réel.

Ce papier est organisé en plusieurs sections. Dans la Section 2, nous résu we review the literature about intentions in Human-Robot Interaction (HRI), probabilistic models for motion primitives and their related software. In Section 4 we describe the theoretical tools that we use to formalize the problem of predicting

the intention of the human during interaction. Particularly, we describe the ProMPs and their use for predicting the evolution of a trajectory given early observations. In Section **??** we overview the software organization and the interconnection between our software and the iCub's main software, both for the real and simulated robot. The following sections are devoted to presenting our software and its use for predicting intention. We choose to present three examples of increasing complexity, with the simulated and real robot. We provide and explain in detail a software example for a 1-DOF trajectory in Section **??**. In Sections **??** and **??** we present the intention recognition application with the simulated and real iCub, respectively. In the first examples with the robot, the "tasks" are exemplified by simple reaching movements, to provide simple and clear trajectories that help the reader understand the method, whereas the last experiment with the robot is a collaborative object sorting task. Section **??** provides the links to the videos showing how to use the software in simulation and on the iCub. Finally, in Section **??** we discuss our approach, its limitations and outline our future developments.

2 RELATED WORK

In this paper we propose a method to recognize the intention of the human partner collaborating with the robot, formalized as the target and the "future" trajectory associated to a skill, modeled by a goal-directed Probabilistic Movement Primitive. In this section, we briefly overview the literature about intention recognition in human-robot interaction and motion primitives for learning of goal-directed robotic skills.

2.1 Intention during human-robot interaction

Lors de collaboration homme-robot, la compréhension mutuelle de ces deux agents est primordiale pour assurer la réussite de leur tâche commune. La compréhension mutuelle signifie que chaque agent est au courant de l'action courante de l'autre, de son statut, de son but, des informations disponibles qu'il peut estimer ou prédire. La reconnaissance de l'intention n'est qu'une pièce de ce problème, mais joue un rôle cruciale dans l'acquisition de compétence anticipative.

Formaliser l'intention s'avère être une tâche compliquée, notamment parce que fournir une représentation unique qui explique à la fois l'intention moteur (mouvement actuel) ; l'intention à bas niveau c'est à dire des actions dirigés vers un but (*e.g.* atteindre un objet ciblé et l'attraper) ; et l'intention à haut niveau, c'est à dires des actions complexes, abstraites, cognitives (*e.g.* changer une ampoule au plafond, en utilisant une échelle, en la grimpant pour afin atteindre l'ampoule, etc.). Dans **?**, un récapitulatif est fait sur les différentes approches existantes de la reconnaissance de l'action et de la prédiction de l'intention.

Du point de vue de l'humain, comprendre l'intention du robot signifie qu'il doit déterminer les mouvements ou actions du robot dirigés vers un but, de manière intuitive et non-ambiguë, et qu'il doit comprendre ce que le robot est entrain de faire, ou ce qu'il va faire (**?**). Dans **?**, on formalise les différence entre *prédiction* et *lisibilité* : un mouvement est lisible si un observateur peut rapidement inférer son but, tandis qu'un mouvement est prédictible lorsqu'il correspond aux attentes d'un l'observateur connaissant le but. De plus, quand une trajectoire peut être prédite par un observateur à partir d'observation antérieure de celle-ci, on peut dire que la trajectoire n'est pas seulement lisible, mais aussi prédictible.

Les difficultés lors de la génération de mouvements robotiques *lisibles* ont été étudiées dans des études récentes. Par exemple, dans **?** on utilise des techniques d'optimisation afin de générer des mouvements à la fois lisible et prédictibles. Dans **?**, on applique une méthode d'apprentissage par renforcement inverse à des voitures autonomes afin de sélectionner les mouvements du robot qui contiennent le maximum d'information pour les humains et qui facilite ainsi la compréhension des objectifs du robot.

Du point de vue du robot, comprendre l'intention du partenaire humain signifie que le robot doit être capable de déchiffrer l'ensemble des indices verbaux et non-verbaux naturellement générés par le comportement de l'humain afin d'identifier, selon la tâche et le contexte courant, quelle est son intention. Plus le robot utilise d'informations (*e.g.* signaux mesurables provenant du partenaire humain et de son environnement), plus l'estimation peut être précis. La forme la plus simple de la reconnaissance de l'intention du partenaire est d'estimer le but d'une action en cours, sous la condition que chaque action correspond à un mouvement dirigé vers un but. **/*intention*/** Dans **?**, on montre que les humains attribuent implicitement des intentions dirigées vers un but aux mouvements robotiques, ce qui montre que les humains effectuent un regard d'anticipation vers l'objectif visé. **/* ? phrase originel : showed that humans implicitly attribute intentions in form of goals to robot motions, proving that humans exhibit anticipatory gaze towards the intended goal */** Le regard a aussi été utilisé dans un jeu homme-robot avec le iCub **?**, où le robot (respectivement l'humain) suivait le regard de l'humain (respectivement du robot) pour identifier l'objet visé. Dans **?**, on propose un algorithme Bayésien permettant de prédire l'intention du mouvement humain, en calculant géométriquement le but le plus probable à partir de la méthode Espérance-Maximisation et d'un simple classificateur Bayésien. Dans (**?**), on propose une méthode appelée modèle dynamique dirigé vers le but (Intention-Driven Dynamics model). Cette méthode est basée sur les Modèles Dynamiques de Processus Gaussiens (GPDM (**?**)) et elle permet d'inférer l'intention de l'adversaire humain du robot lors d'un match de ping-pong. Cette intention à estimer correspond à la position ciblée avec la balle, et la méthode se base sur le mouvement entier du partenaire humain, avant même que celui-ci frappe la balle.

/*comprendre la phrase : More generally, modeling and descriptive approaches can be used to match predefined labels with measured data (?)*/.

Une forme plus complexe de reconnaissance de l'intention correspond à l'estimation de la continuation d'une trajectoire à partir des observations antérieures. C'est à dire que l'on veut estimer $[x_{t+1}, \ldots, x_{t+T_{future}}] = f(x_t, x_{t-1}, \ldots, x_{t-T_{past}})$. Ce problème, similaire à l'estimation des modèles prédictif de la dynamique d'un systeme **/*c'est bien ça forward dynamics ?*/**, est fréquemment adressé par les chercheurs en ce qui concerne les modèles de contrôle prédictifs, où être capable de faire évoluer le système dans les temps est la base du controle robotique. ~~This problem, very similar to the estimate of the forward dynamics model of a system, is frequently addressed by researchers in model predictive control, where being able to "play" the system evolving in time is the basis for computing appropriate robot controls.~~

Quand une trajectoire peut être prédite par un observateur à partir d'observation antérieure de celle-ci, on peut dire que la trajectoire n'est pas seulement *lisible*, mais aussi *prédictible*. Une approche systématique permettant de prédire une trajectoire est de raisonner en terme de primitives de mouvements, de façon à ce que la séquence de points d'une trajectoire peut être générée par un modèle temporel paramétrique ou par un système dynamique paramétrique. Par exemple, **?** planifie des trajectoires dirigées vers des buts permettant de transporter des objets. Cette méthode de planification permet notamment de connaître les informations sur le poids des objets transportés. Plus généralement, dans les approches génératives (**?**), des variables latentes sont utilisées pour modéliser par apprentissage les primitives et ainsi être capable de générer et d'inférer des actions. La section suivante donne plus de détails concernant l'état de l'art des techniques permettant de générer les primitives de mouvements.

Dans (**?**), le robot commence par apprendre une primitive d'interaction en regardant deux humains effectuer eux-même la tâche interactive, à l'aide de capture du mouvement. Cette primitive d'Interaction encapsule les dépendances entre deux mouvements humains. Puis, le robot utilise des primitives d'interaction

afin d'adapter son comportement aux mouvements de son partenaire. La méthode utilisé est basée sur les Primitives Dynamique de Moteur (Dynamics Motor Primitives (**?**)), où une distribution sur les paramètres des DMP est apprise. Notons que dans notre papier, nous n'utilisons pas la même approche pour apprendre les primitives d'interaction puisque nous modélisons les trajectoires du robot et de l'utilisateur comme une trajectoire unique conjointe, dû au fait que le l'utilisateur et le robot sont en contact physique. De plus, il n'y a pas de latence entre le mouvement initié par le partenaire et celui du robot, puisque le bras du robot est guidé physiquement par le partenaire jusqu'à ce que celui-ci stop le contact.

Beaucoup d'exemples dans la littérature se focalise sur des trajectoires cinématiques, correspondant aux gestes utilisés dans les interactions dyadiques caractérisés par une coordination des actions et des réactions. D'autres s'intéressent à des trajectoires plus complexes, qui correspondent aux trajectoires effectués lors d'interaction physique entre l'homme et le robot, permettant d'effectuer des tâches nécessitant la collaboration des agents et des échanges de forces. En effet, les informations cinématiques fournies par de telles trajectoires ne peuvent pas être analyser sans prendre en compte l'échange haptique et l'estimation des rôles des partenaires (*i.e.*, qui est le leader qui est le follower).

Estimer le rôle actuel du partenaire humain (maître/esclave ou leader/follower) est cruciale puisque l'information des rôle est nécessaire pour adapter correctement la compliance et l'impédance du robot au niveau des échanges de forces de contact.

Plus important, adapter l'interaction haptique peut permettre au robot d'exprimer qu'il a compris l'intention du partenaire et qu'il est capable de finir la tâche de manière autonome, en imitant le même type de communication non verbale qui est typique chez l'humain.

Par exemple, dans (**?**), le robot infère l'intention de l'humain en utilisant les mesures des forces humaines et en utilisant les modèles de mixture gaussiennes. Dans (**?**), l'impédance des bras est adapté en utilisant un modèle basé sur les modèles de mixtures de Gaussienne, sur les forces mesurées et sur les informations visuelles. Beaucoup d'études se focalisent sur l'abilité du robot à agir seulement quand et selon comment l'utilisateur le souhaite (**?**)(**?**) , d'autres sur la capacité au robot de ne pas interférer contre les forces de son partenaire (**?**) ou de ses actions (**?**).

In this paper, we describe our approach to the problem of recognizing the human intention during collaboration by providing an estimate of the future intended trajectory to be performed by the robot. In our experiments, the robot does not adapt its role during the physical interaction, but simply switch from follower to leader when the human breaks contact with it.

3 MA VIEILLE PARTIE

Dans cer papier, on souhaite que notre robot prédise l'intention de son partenaire afin de finaliser un mouvement et une action par lui même. Beaucoup d'autres travaux de recherche necessite cette abilité.

Dans **?**, il s'agit de prédire les mouvements d'un humain en utilisant une méthode appelée Prédiction Bayesienne de l'Intention du Mouvement Humain (BHMIP). Cette méthode se base sur des calculs géométriques permettant de trouver la destination la plus probable de l'humain ainsi que sur l'algorithme Esperance-Maximisation Puis, pour trouver la prédiction la plus performante, ils utilisent un classifieur Bayesien. Cette méthode a l'avantage d'être indépendant du type d'environnement et qu'il ne nécessite pas de calculs intensifs.

D'autres études utilisent la capacité d'inférence pour adapter l'impédance des robots. Dans (**?**), le robot infère l'intention de l'humain grâce a la mesure des forces de celui ci et en utilisant des modèles de mixtures

de gaussiennes. Dans (**?**), ils utilisent des Hidden Markov Models pour apprendre comment compenser des forces physique potentielles. Leur robot utilise cette méthode afin de serrer la main de son partenaire, en mesurant l'impédance de ce dernier et en reconnaissant quel type de mouvement il doit suivre. Dans (**?**), un bras robotique mesure les forces et les informations visuelles afin d'adapter son impédance, en utilisant ici aussi des modèles de mixtures gaussiennes. Dans notre étude, le robot utilise l'information des forces et des moments afin d'estimer l'intention de son partenaire ainsi que de détecter la vitesse du mouvement.

Dans une autre thématique de recherche, les robots utilisent la capacité de prédiction de l'intention humaine afin d'adapter leur comportement (maître/esclave). En effet, pour collaborer efficacement avec les humains, les robots doivent être rigide quand ils effectuent eux même la tache (maître), tout en restant alerte à la volonté de leur partenaire (c'est à dire qu'il doit rester suffisamment compliant pour permettre à l'utilisateur de contrôler l'action). C'est pourquoi beaucoup d'études se focalisent sur l'habilité du robot à agir uniquement quand et comment son utilisateur le souhaite (**?**)(**?**) et sur sa capacité à ne pas interférer avec les forces de son partenaire (**?**) ou ses actions (**?**). Dans notre étude, le robot est "esclave" lorsqu'il est guidé par son partenaire (au début du mouvement), puis devient maître lorsqu'il finit le mouvement par lui même.

Dans certaines études, le robot observe d'abord le mouvement du partenaire dans sa totalité avant d'agir.Dans (**?**), une méthode appelé Intention-Driven Dynamics model permet d'inférer l'intention du partenaire du robot durant un match de ping pong. Cette méthode est une amélioration des Gaussian Process Dynamical Models (GPDM (**?**)) et permet au robot d'inférer où la balle va être envoyé, avant même que le partenaire humain ait frappé la balle, grâce à l'analyse du mouvement de celui-ci.

Pour obtenir cette aptitude d'inférence, différentes méthodes existent. Dans(**?**), un récapitulatif est fait sur les différentes approches concernant la reconnaissance de l'action et de la prédiction de l'intention. La premiers possibilité consiste à faire coïncider les informations mesurées avec des modèles. Pour cela, une première possibilité consiste à utiliser des approches descriptives, où l'on fait correspondre à des labels prédéfinis, les données récoltées. On peut trouver un récapitulatif de ces méthodes descriptives dans (**?**). Une deuxième possibilité consiste à utiliser des approches génératives, où l'on utilise des variables latentes afin d'apprendre les informations utiles à l'inférence de l'action. Ce type d'approche se base sur l'utilisation de distribution probabilistes. Un récapitulatif de ces approches génératives peut être trouvé dans (**?**).

Dans **/*citer Yannis Demiris Prediction of intent in robotics and multi agent systems 2007*/**, les différentes approches permettant la reconnaissance de l'action et la prédiction de l'intentionnalité sont définis et expliqués. Dans ce document, une architecture générative est mise en évidence, appelé HAMMER (Hierarchical Attentive Multiple Models for Excecution and Recognition). Cette architecture utilise un modele inverse de prédiction **/*inverse-forward model*/** afin d'executer une action ou de reconnaitre une action faite par un demonstrateur.

Un probleme sous jacent à la reconnaissance de mouvement est que le robot doit estimer la durée de la trajectoire afin de l'aligner avec les trajectoires qu'il a apprit. Dans notre cas, au début de l'interaction physique entre l'homme et le robot, ce dernier observe un mouvement partiellement guidé par son utilisateur. À partir de ce mouvement partiel, le robot doit d'abord estimer l'etat courant du mouvement afin de comprendre l'intention de son partenaire. Ainsi, il a besoin d'estimer la vitesse du mouvement.

Mathématiquement, la loi de Fitt modélise la durée d'um mouvement. Une assumption de ce modèle est que la durée du mouvement est fonction linéaire de la difficulté à atteindre la cible**?**. Dans **?**, ils montrent qu'en modifiant la taille de la cible, la forme du mouvement change. Ainsi, il est difficile d'appliquer la théorie de la loi de Fitt quand la taille de la cible peut varier. Dans **?** et **?**, ils confirment cette idée en

montrant que la même forme du mouvement change avec la précision necessaire pour atteindre la position but du mouvement.

Une autre méthode, souvent utilisé en informatique est appelée Dynamics Time Warping [5] (DTW). Cette méthode permet de trouver la corrélation entre deux trajectoires dont la durée diffère, de maniere plus robuste qu'en utilisant la distance euclidienne. Dans **?**, cet algorithme est modifié afin de permettre de faire coïncider des mouvements partiels avec une trajectoire référence. Beaucoup d'améliorations et de variations de cette méthode existe. Dans **?**, ils proposent des méthodes permettant d'améliorer l'indexation, et ainsi d'accélérer la vitesse de calcul de l'algorithme. On trouve ainsi les méthodes FastDTW, Lucky Time Wraping ou encore FTW. Dans**?**, ils commencent par expliquer et comparer ces méthodes, puis ils ajoutent leur propre méthode appelé Pruned Warping Path, qui accélère encore plus la vitesse calculatoire. De plus, cette méthode permet de supprimer les données improbables. Mais le principale désavantage de toutes ces méthodes basés sur DTW est qu'elles ne préservent pas la forme globale de la trajectoire (la trajectoire est déformée).

Dans **?**, des primitives de mouvements sont apprises à l'aide d'apprentissage probabiliste. Ils améliorer l'estimation des mouvements en utilisant une méthode différente. Celle-ci utilise aussi une modélisation à base de gaussiennes de la fonction de déformation du temps, et au lieu d'utiliser la méthode DTW, cette méthode force l'alignement locale entre les deux mouvements, sans "sauter" des indexes. Ainsi, les trajectoires obtenues sont plus réalistes, plus régulière, et cette méthode préserve la forme globale des trajectoires.

3.1 *Movement primitives* / Primitives de mouvement

Movement Primitives (MPs) is a well established paradigm for representing complex motor skills.

Les modèles utilisant des primitives de mouvement (MPs) sont une référence quand il s'agit de représenter des capacités motrices complexes.

The most known method for representing movement primitives is probably the Dynamic Movement Primitives (DMPs) **???**. DMPs use a stable non-linear attractor in combination with a forcing term to represent the movement. The forcing term enables to follow specific movement, while the attractor asserts asymptotic stability. In a recent paper, **?** proposed an extension to DMPs, called PDMP (Probabilistic Dynamic Movement Primitive). This method improves DMP with probabilistic properties to measure the likelihood that the movement primitive is executed correctly and to perform inference on sensor measurement. However, The PDMPs do not have a data-driven generalization and can deviate arbitrarily from the demonstrations. This last difference can be critical for our applications with the humanoid robot iCub, since uncertainties are unavoidable and disturbances may happen frequently and de-stabilize the robot movement (for example, an unexpected collision during the movement). Thus, the ProMPs method is more accurate for our software.

?, **?** and **?** compared ProMPs and DMPs for learning primitives and specifically interaction primitives. With the DMP model, at the end of the movement, only a dynamic attractor is activated. Thus, it always reach a stable goal. The properties allowed by both methods are temporal scaling of the movement, learning from a single demonstration, and generalizing to new final position. With ProMPs, we have in addition the ability to do inference (thanks to the distribution), to force the robot to pass by several initial via-points (the early observations), to know the correlation between the input of the model, and to co-activate some

5. Déformation temporelle dynamique

ProMPs. In our study, we need these features, because the robot must determine a trajectory that passes by the early observations (beginning of the movement where the user guides physically the robot).

A Recurrent Neural Networks (RNN) approach **?** used a hierarchy of neural networks to simulate the activation of areas in human brain. The network can be trained to infer the state of the robot at the next point in time, given the current state. The authors propose to train the RNN by minimizing the error between the inferred position of the next time step and the ground-truth obtained from demonstrations.

Hidden Markov Models (HMMs) for movement skills were introduced by **?**. This method is often used to categorize movements, where a category represents a movement primitive. This method also allows to represent the temporal sequence of a movement. In **?** they use learned Hierarchical Hidden Markov Model (HHMMs) to recognize human behaviors efficiently. In **?** they present the Primitive based Coupled-HMM (CHMM) approach, for human natural complex action recognition. In this approach, each primitive is represented by a Gaussian Mixture Model.

Adapting Gaussian Mixture Models is another method used to learn physical interaction with learning. In **?** they use GMMs and Gaussian Mixture Regression to learn, in addition to the position (joint information), force information. Using this method, a humanoid robot is able to collaborate in one dimension with its partner for a lifting task. In our paper, we will also use (Cartesian) position and force information to allow our robot to interact physically with its partner.

A sub-problem of movement recognition is that robots need to estimate the duration of the trajectory to align a current trajectory with learned movements. In our case, at the beginning of the physical Human-Robot Interaction (pHRI), the robot observes a partial movement guided by its user. Given this partial movement, the robot must first estimate what the current state of the movement is to understand what its partner intent is. Thus, it needs to estimate the partial movement's speed.

Fitts' law models the movement duration for goal-directed movements. This model is based on the assumption that the movement duration is a linear function of the difficulty to achieve a target**?**. In **?**, they show that by modifying the target's width, the shape of the movement changes. Thus, it is difficult to apply Fitt's law when the size of the target can change. In **?** and **?**, they confirm this idea by showing that the shape of the movement changes with the accuracy required by the goal position of the movement.

Dynamics Time Warping (DTW) is a method to find the correlation between two trajectories that have different durations, in a more robust way than the Euclidean distance. In **?**, they modify the DTW algorithm to match a partial movement with a reference movement. Many improvements over this method exist. In **?**, they propose a robust method to improve the indexation. The calculation speed of DTW is improved using different methods, such as FastDTW, Lucky Time Warping or FTW. An explanation and comparison of these methods is presented in **?**, where they add their own computation speed improvement by using a method called Pruned Warping Paths. This method allows the deletion of unlikely data. However, a drawback of this well-known DTW method is they don't preserve the global trajectory's shape.

In **?**, where they use a probabilistic learning of movement primitives, they improve the duration estimation of movements by using a different time warping method. This method is based on a Gaussian basis model to represent a time warping function and, instead of DTW, it forces a local alignment between the two movements without "jumping" some index. Thus, the resulting trajectories are more realistic, smoother, and this method preserves the global trajectories' shapes.

For inferring the intention of the robot's partner, we use Probabilistic Movement Primitives (ProMPs), **?**). Specifically, we use the ProMP's conditioning operator to adapt the learned skills according to observations.

Software/library	Method	Code link	Language	Robot	Reference(s)
Dynamical System Modulation for Robot Adaptive Learning via Kinesthetic Demonstrations	GMR	?	Matlab	Hoap3	?
pbdlib-matlab	HMM, GMM, and others	?	Matlab	Baxter	?
DMP learning with GMR	DMP and GMR	?	Matlab or C	Coman	?
Stochastic Machine Learning Toolbox	Kernel Functions, Gaussian Processes, Bayesian Optimization	?	C++ or Python	—	
pydmps	DMP	?	Python	Sarcos	?
Dynamical Systems approach to Learn Robot Motions	GMM, SEDS	?	Matlab	iCub	?, ?
Function Approximation, DMP, and Black-Box Optimization (dmpbbo)	DMP	?	Python or C++	iCub	??
Learning Motor Skills from Partially Observed Movements Executed at Different Speeds	ProMP	?	Matlab or Python	—	?
icubLearningTrajectories	ProMP	?	Matlab and C++	iCub	—

Table 1. Open-source software libraries implementing Movement Primitives and their application to different known robots.

The ProMPs can encode the correlations between forces and positions and allow better prediction of the partner's intention. Further, the phase of the partner's movement can be inferred and therefore the robot can adapt to the partner's velocity changes. ProMPs are more efficient for collaborative tasks, as shown in **?**, where in comparison to DMPs, the root-mean square error of the predictions is lower.

3.2 Related open-source software

One of the goals of this paper is to introduce an open-source software for the iCub (but potentially for any other robot), where the ProMP method is used to recognize human intention during collaboration, so that the robot can execute initiated actions autonomously. This is not the first open-source implementation for representing movement primitives : however, it has a novel application and a rationale that makes it easy to use with the iCub robot.

In Table 1 we report on the main software libraries that one can use to learn movement primitives. Some have been also used to realize learning applications with iCub, *e.g.*, **??** or to recognize human intention. However, the software we propose here is different : it provides an implementation of ProMPs used explicitly for intention recognition and prediction of intended trajectories. It is interfaced with iCub, both real and simulated, and addresses in the specific case of physical interaction between the human and the robot. In short, it is a first step towards adding intention recognition ability to the iCub robot.

4 THEORETICAL FRAMEWORK

In this section we present the theoretical framework that we use to tackle the problem of intent recognition : we describe the ProMPs and how they can be used to predict trajectories from early observations.

Dans cette section nous présentons le cadre théorique que nous utilisons pour nous attaquer au problème de la reconnaissance d'intention. C'est-à-dire que nous allons détailler le fonctionnement des méthodes ProMPs et comment elles peuvent être utilisées pour prédire des trajectoires à partir des observations initiales.

*In Section **??** we formulate the problem of learning a primitive for a simple case, where the robot learns the distribution from several demonstrated trajectories. In Section **??** we formulate and provide the solution to the problem of predicting the "future" trajectory from early observations (i.e., the initial data points). In Section **??** we discuss the problem of predicting the time modulation, i.e., predicting the global duration of the predicted trajectory. This problem is non-trivial, as by construction the demonstrated trajectories are "normalized" in duration when the ProMP is learned.* [6] *In Section **??** we explain how to recognize, from the early observations, to which of many known skills (modeled by ProMPs) the current trajectory belongs. In all these sections we tried to present the theoretical aspects related to the use of ProMPs for the intention recognition application. Practical examples of these theoretical problems are presented and explained later in sections **??** - **??**. Section **??** explains how to use our software, introduced in Section **??**, for learning one ProMP for a simple set of 1-DOF trajectories. Section **??** presents an example with the simulated iCub in Gazebo, while Section **??** presents an example with the real iCub.*

Dans la Section **??** nous formulons le problème d'apprentissage de primitives à l'aide de la méthode ProMP pour un cas simple. Dans ce cas, le robot apprend la distribution à partir de plusieurs trajectoires de démonstration. Dans la Section **??** nous formulons et fournissons la solution au problème de prédiction de la trajectoire « future » à partir des premières observations sur la trajectoire. Dans la Section **??** nous discutons du problème de prédiction de la modulation du temps, c'est-à-dire la prédiction de la durée globale de la trajectoire prédite. Ce problème n'est pas trivial parce que les trajectoires de démonstration sont « normalisées » par rapport au temps pendant l'apprentissage des primitives avec la méthode ProMP. Pour certaines taches, comme des taches d'atteignabilité par exemple, il est raisonnable de supposer que la différence de durée des trajectoires est négligeable ; cependant d'autres taches exigent davantage de précision et la durée des trajectoires de démonstration peut significativement varier. Dans la Section **??** nous expliquons comment reconnaitre, à partir des premières observations, laquelle des nombreuses compétences apprises (modélisées par des méthodes ProMP) doit être sollicitée. Ainsi dans ces trois sections nous présentons les aspects théoriques liés à l'utilisation de méthodes ProMP dans le but de reconnaitre des intentions.

Nous illustrons ces problèmes théoriques dans les Sections **??** à **??**. Dans la Section **??** nous détaillons un exemple avec un robot iCub simulé avec Gazebo**??** et dans la Section **??** nous présentons un exemple cette fois-ci avec le robot iCub réel.

4.1 Notation

To facilitate understanding of the theoretical framework, we first introduce the notations we use in this section and throughout the remainder of the paper.

Pour faciliter la compréhension du cadre théorique nous introduisons tout d'abord les notations que nous utilisons tout au long de cette thèse.

Trajectories :

—$X(t) \in \mathbb{R}^3, X(t) = [x(t), y(t), z(t)]^\top$: the x/y/z-axis Cartesian coordinate of the robot's end-effector.

6. In some tasks, *e.g.*, reaching, it is reasonable to assume that the difference of duration of the demonstrated trajectories is negligible ; however, in other tasks the duration of the demonstrated trajectories may vary significantly.

—$F(t) \in \mathbb{R}^6, F(t) = [f_x, f_y, f_z, m_x, m_y, m_z]^\top$: the wrench contact forces, *i.e.* the external forces and moments measured by the robot at the contact level (end-effector).
—$\xi(t) \in \mathbb{R}^D$: the generic vector containing the current value or state of the trajectories at time t. It can be mono-dimensional (*e.g.* $\xi(t) = [z(t)]$), or multi-dimensional (*e.g.* $\xi(t) = [X(t), F(t)]^\top$), depending on the type of trajectories that we want to represent with the ProMP.
—$\Xi = \Xi_{[1:t_f]} = [\xi(1), \dots, \xi(t_f)]^\top \in \mathbb{R}^{D \cdot t_f}$ is an entire trajectory, consisting of t_f samples or data points.
—$\Xi_{i[1:t_{fi}]}$ is the i-th demonstration (trajectory) of a task, consisting of t_{fi} samples or data points.

Movement Primitives :

—$k \in [1 : K]$: the k-th ProMP, among a set of K ProMPs that represent different tasks/actions.
—n_k : number of recorded trajectories for each ProMP.
—$S_k = \{\Xi_{\{k,1\}}, \dots, \Xi_{\{k,n_k\}}\}$: set of n_k trajectories for the k-th ProMP.
—$\xi(t) = \Phi_t \boldsymbol{\omega} + \epsilon_\xi$ is the model of the trajectory with :
—$\epsilon_\xi \sim \mathcal{N}(0, \beta)$: expected trajectory noise.
—$\Phi_t \in \mathbb{R}^{D \times D \cdot M}$: radial basis functions (RBFs) used to model trajectories. It is a block diagonal matrix.
-M : number of RBFs.
-$\psi_{ji}(t) = \dfrac{e^{\frac{-(t-c_i)^2}{2h}}}{\sum_{m=1}^{M} e^{\frac{-(t-c_m)^2}{2h}}}$: i-th RBF for all inputs $j \in [1 : D]$.
It must be noted that the upper term comes from a Gaussian $\frac{1}{\sqrt{2\pi h}} e^{\frac{-(t-c_i)^2}{2h}}$, where c_i, h are respectively the center and variance of the i-th Gaussian. In our RBF formulation, we normalize all the Gaussians. ˆ
—$\boldsymbol{\omega} \in \mathbb{R}^{D \cdot M}$: time-independent parameter vector weighting the RBFs, *i.e.*, the parameters to be learned.
—$p(\boldsymbol{\omega}) \sim \mathcal{N}(\mu_{\boldsymbol{\omega}}, \Sigma_{\boldsymbol{\omega}})$: normal distribution computed from a set $\{\boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_n\}$. It represents the distribution of the modeled trajectories, also called *prior* distribution.

Time modulation :

—$\bar{s}$: number of samples used as reference to rescale all the trajectories to the same duration.
—$\Phi_{\alpha_i t} \in \mathbb{R}^{D \times D \cdot M}$: the RBFs rescaled to match the Ξ_i trajectory duration.
—$\alpha_i = \frac{\bar{s}}{t_{fi}}$: temporal modulation parameter of the i-th trajectory .
—$\alpha = \Psi_{\delta_{n_o}} \boldsymbol{\omega}_\alpha + \epsilon_\alpha$ is the model of the function mapping δ_{n_o} into the temporal modulation parameter α, with :
-Ψ : a set of RBFs used to model the mapping between δ_{n_o} and α ;
-δ_{n_o} is the variation of the trajectory during the first n_o observations (data points) ; it can be $\delta_{n_o} = \xi(n_o) - \xi(1)$ if the entire trajectory variables (*e.g.*, Cartesian position, forces, etc.) are considered, or more simply $\delta_{n_o} = X(n_o) - X(1)$ if only the variation in terms of Cartesian position is considered ;
-$\boldsymbol{\omega}_\alpha$: the parameter vector weighting the RBFs of the Ψ matrix.

Inference :

—$\Xi^o = [X^o, F^o]^\top = [\xi^o(1), \dots, \xi^o(n_o)]^\top$: early-trajectory observations, composed of n_o data points.
—Σ_ξ^o : noise of the initiated trajectory observation.
—$\hat{\alpha}$: estimated time modulation parameter of a trajectory to infer.
—$\hat{t}_f = \frac{\bar{s}}{\hat{\alpha}}$: estimated duration of a trajectory to infer.
—$\Xi^* = [\xi^o(1), \dots, \xi^o(n_o), \xi^*(n_o + 1), \dots, \xi^*(t_f)]$: ground truth of the trajectory for the robot to infer.

—$\hat{\Xi} = [\hat{X}, \hat{F}]^\top = [\xi^o(1), \ldots, \xi^o(n_o), \hat{\xi}(n_o+1), \ldots, \hat{\xi}(\hat{t}_f)]^\top$: the estimated trajectory.
—$p(\hat{\boldsymbol{\omega}}) \sim \mathcal{N}(\hat{\mu}_{\boldsymbol{\omega}}, \hat{\sigma}_{\boldsymbol{\omega}})$: posterior distribution of the parameter vector of a ProMP using the observation Ξ^o.
—$\hat{k}$: index of the recognized ProMP from the set of K known (previously learned) ProMPs.

Trajectoires :

—$X(t) \in \mathbb{R}^3, X(t) = [x(t), y(t), z(t)]^\top$: l'axe x/y/z pour représenter les coordonnées cartésiennes de l'effecteur du robot.
—$F(t) \in \mathbb{R}^6, F(t) = [f_x, f_y, f_z, m_x, m_y, m_z]^\top$: la wrench des forces de contact, c'est-à-dire les forces externes et les moments mesurées par le robot au niveau du contact (effecteur).
—$\xi(t) \in \mathbb{R}^D$: le vecteur générique contenant la valeur courante ou l'état des trajectoires au temps t. Il peut être mono-dimensionnel (*e.g.*, $\xi(t) = [z(t)]$), ou multidimensionnel(*e.g.* $\xi(t) = [X(t), F(t)]^\top$), dépendant du type de trajectoires qui peut être représenté avec la méthode ProMP.
—$\Xi = \Xi_{[1:t_f]} = [\xi(1), \ldots, \xi(t_f)]^\top \in \mathbb{R}^{D \cdot t_f}$ is an entire trajectory, consisting of t_f samples or data points.
—$\Xi_{i[1:t_{fi}]}$ is the i-th demonstration (trajectory) of a task, consisting of t_{fi} samples or data points.

Movement Primitives :

—$k \in [1:K]$: the k-th ProMP, among a set of K ProMPs that represent different tasks/actions.
—n_k : number of recorded trajectories for each ProMP.
—$S_k = \{\Xi_{\{k,1\}}, \ldots, \Xi_{\{k,n_k\}}\}$: set of n_k trajectories for the k-th ProMP.
—$\xi(t) = \Phi_t \boldsymbol{\omega} + \epsilon_\xi$ is the model of the trajectory with :
—$\epsilon_\xi \sim \mathcal{N}(0, \beta)$: expected trajectory noise.
—$\Phi_t \in \mathbb{R}^{D \times D \cdot M}$: radial basis functions (RBFs) used to model trajectories. It is a block diagonal matrix.
-M : number of RBFs,
-$\psi_{ji}(t) = \dfrac{e^{\frac{-(t-c_i)^2}{2h}}}{\sum_{m=1}^{M} e^{\frac{-(t-c_m)^2}{2h}}}$: i-th RBF for all inputs $j \in [1:D]$.
It must be noted that the upper term comes from a Gaussian $\frac{1}{\sqrt{2\pi h}} e^{\frac{-(t-c_i)^2}{2h}}$, where c_i, h are respectively the center and variance of the i-th Gaussian. In our RBF formulation, we normalize all the Gaussians. ˆ
—$\boldsymbol{\omega} \in \mathbb{R}^{D \cdot M}$: time-independent parameter vector weighting the RBFs, *i.e.*, the parameters to be learned.
—$p(\boldsymbol{\omega}) \sim \mathcal{N}(\mu_{\boldsymbol{\omega}}, \Sigma_{\boldsymbol{\omega}})$: normal distribution computed from a set $\{\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_n\}$. It represents the distribution of the modeled trajectories, also called *prior* distribution.

Time modulation :

—$\bar{s}$: number of samples used as reference to rescale all the trajectories to the same duration.
—$\Phi_{\alpha_i t} \in \mathbb{R}^{D \times D \cdot M}$: the RBFs rescaled to match the Ξ_i trajectory duration.
—$\alpha_i = \frac{\bar{s}}{t_{fi}}$: temporal modulation parameter of the i-th trajectory .
—$\alpha = \Psi_{\delta_{n_o}} \boldsymbol{\omega}_\alpha + \epsilon_\alpha$ is the model of the function mapping δ_{n_o} into the temporal modulation parameter α, with :
-Ψ : a set of RBFs used to model the mapping between δ_{n_o} and α ;
-δ_{n_o} is the variation of the trajectory during the first n_o observations (data points) ; it can be $\delta_{n_o} = \xi(n_o) - \xi(1)$ if the entire trajectory variables (*e.g.*, Cartesian position, forces, etc.) are considered, or more simply $\delta_{n_o} = X(n_o) - X(1)$ if only the variation in terms of Cartesian position is considered ;

-$\boldsymbol{\omega}_\alpha$: the parameter vector weighting the RBFs of the Ψ matrix.

Inference :

—$\Xi^o = [X^o, F^o]^\top = [\xi^o(1), \ldots, \xi^o(n_o)]^\top$: early-trajectory observations, composed of n_o data points.

—Σ_ξ^o : noise of the initiated trajectory observation.

—$\hat{\alpha}$: estimated time modulation parameter of a trajectory to infer.

—$\hat{t}_f = \frac{\bar{s}}{\hat{\alpha}}$: estimated duration of a trajectory to infer.

—$\Xi^* = [\xi^o(1), \ldots, \xi^o(n_o), \xi^*(n_o+1), \ldots, \xi^*(t_f)]$: ground truth of the trajectory for the robot to infer.

—$\hat{\Xi} = [\hat{X}, \hat{F}]^\top = [\xi^o(1), \ldots, \xi^o(n_o), \hat{\xi}(n_o+1), \ldots, \hat{\xi}(\hat{t}_f)]^\top$: the estimated trajectory.

—$p(\hat{\boldsymbol{\omega}}) \sim \mathcal{N}(\hat{\mu}_{\boldsymbol{\omega}}, \hat{\sigma}_{\boldsymbol{\omega}})$: posterior distribution of the parameter vector of a ProMP using the observation Ξ^o.

—$\hat{k}$: index of the recognized ProMP from the set of K known (previously learned) ProMPs.

4.2 Learning a Probabilistic Movement Primitive (ProMP) from demonstrations

Our toolbox to learn, replay and infer the continuation of trajectories is written in Matlab and available at :

`https://github.com/inria-larsen/icubLearningTrajectories/tree/master/MatlabProgram`

Let us assume the robot has recorded a set of n_1 trajectories : $\{\Xi_1, \ldots, \Xi_{n_1}\}$, where the i-th trajectory is $\Xi_i = \{\xi(1), \ldots, \xi(t_{f_i})\}$. $\xi(t)$ is the generic vector containing all the variables to be learned at time t, with the ProMP method. It can be mono-dimensional (*e.g.* $\xi(t) = [z(t)]$ for the z-axis Cartesian coordinate), or multi-dimensional (*e.g.* $\xi(t) = [X(t), F(t)]^\top$). Note that the duration of each recorded trajectory (*i.e.* t_{f_i}) may be variable. To find a common representation in terms of primitives, a time modulation is applied to all trajectories, such that they have the same number of samples $\bar{s}$ (see details in Section **??**). Such modulated trajectories are then used to learn a ProMP.

A ProMP is a Bayesian parametric model of the demonstrated trajectories in the form :

$$\xi(t) = \Phi_t \boldsymbol{\omega} + \epsilon_\xi \tag{1}$$

where $\boldsymbol{\omega} \in R^M$ is the time-independent parameter vector weighting the RBFs, $\epsilon_\xi \sim \mathcal{N}(0, \beta)$ is the trajectory noise, and Φ_t is a vector of M radial basis functions evaluated at time t :

$$\Phi_t = [\psi_1(t), \psi_2(t), \ldots, \psi_M(t)]$$

with

$$\begin{cases} \psi_i(t) & = & \frac{1}{\sum_{j=1}^{M} \psi_j(t)} exp\{\frac{-(t-c(i))^2}{2h}\} \\ c(i) & = & i/M \\ h & = & 1/M^2 \end{cases} \tag{2}$$

Note that all the ψ functions are scattered across time.

For each Ξ_i trajectory, we compute the ω_i parameter vector to have $\xi_i(t) = \Phi_t \omega_i + \epsilon_\xi$. This vector is computed to minimize the error between the observed $\xi_i(t)$ trajectory and its model $\Phi_t \omega_i + \epsilon_\xi$. This is done using the Least Mean Square algorithm, *i.e.* :

$$\omega_i = (\Phi_t^\top \Phi_t)^{-1} \Phi_t^\top \xi_i(t). \tag{3}$$

To avoid the common issue of the matrix $\Phi_t^\top \Phi_t$ in Equation **??** not being invertible, we add a diagonal term and perform Ridge Regression :

$$\omega_i = (\Phi_t^\top \Phi_t + \lambda)^{-1} \Phi_t^\top \xi_i(t). \tag{4}$$

where $\lambda = 10^{-11} \cdot \mathbb{1}_{D \cdot M \times D \cdot M}$ is a parameter that can be tuned by looking at the smallest singular value of the matrix $\Phi_t^\top \Phi_t$.

Thus, we obtain a set of these parameters : $\{\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_n\}$, upon which a distribution is computed. Since we assume Normal distributions, we have :

$$p(\boldsymbol{\omega}) \sim \mathcal{N}(\mu_{\boldsymbol{\omega}}, \Sigma_{\boldsymbol{\omega}}) \tag{5}$$

$$\text{with } \mu_{\boldsymbol{\omega}} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{\omega_i} \tag{6}$$

$$\text{and } \Sigma_{\boldsymbol{\omega}} = \frac{1}{n-1} \sum_{i=1}^{n} (\boldsymbol{\omega_i} - \mu_{\boldsymbol{\omega}})^\top (\boldsymbol{\omega_i} - \mu_{\boldsymbol{\omega}}) \tag{7}$$

The ProMP captures the distribution over the observed trajectories. To represent this movement primitive, we usually use the movement that passes by the mean of the distribution Figure **??** shows the ProMP for a 1-DOF lifting motion, with a number of reference samples $\bar{s} = 100$ and number of basis functions $M = 5$.

This example is included in our Matlab toolbox as `demo_plot1DOF.m`. The explanation of this Matlab script is presented in Section **??**. More complex examples are also included in the scripts `demo_plot*.m`.

4.3 Predicting the future movement from initial observations

Once the ProMP $p(\boldsymbol{\omega}) \sim \mathcal{N}(\mu_{\boldsymbol{\omega}}, \Sigma_{\boldsymbol{\omega}})$ of a certain task has been learned [7], we can use it to predict the evolution of an initiated movement. An underlying hypothesis is that the observed movement follows to this learned distribution.

Suppose that the robot measures the first n_o observations of the trajectory to predict (*e.g.*, lifting the arm). We call these observations $\Xi^o = [\xi^o(1), \ldots, \xi^o(n_o)]$. The goal is then to predict the evolution of the trajectory after these n_o observations, *i.e.* find $\{\hat{\xi}(n_o+1), \ldots, \hat{\xi}(\hat{t}_f)\}$, where $\hat{t}_f$ is the estimation of the trajectory duration (see Section **??**). This is equivalent to predicting the entire $\hat{\Xi}$ trajectory where the first n_o samples are known and equal to the observations : $\hat{\Xi} = \{\xi^o(1), \ldots, \xi^o(n_o), \hat{\xi}(n_o+1), \ldots, \hat{\xi}(t_{\hat{t}_f})\}$. Therefore, our prediction problem consists of predicting $\hat{\Xi}$ given the Ξ^o observations.

To do this prediction, we start from the learned prior distribution $p(\boldsymbol{\omega})$, and we find the $\hat{\boldsymbol{\omega}}$ parameter within this distribution that generates $\hat{\Xi}$. To find this $\hat{\boldsymbol{\omega}}$ parameter, we update the learned distribution $p(\hat{\boldsymbol{\omega}}) \sim \mathcal{N}(\hat{\mu}_{\boldsymbol{\omega}}, \hat{\Sigma}_{\boldsymbol{\omega}})$ using the formulae :

$$\begin{cases} \hat{\mu}_{\boldsymbol{\omega}} = \mu_{\boldsymbol{\omega}} + K(\Xi^o - \Phi_{[1:n_o]}\, \mu_{\boldsymbol{\omega}}) \\ \hat{\Sigma}_{\boldsymbol{\omega}} = \Sigma_{\boldsymbol{\omega}} - K(\Phi_{[1:n_o]}\, \Sigma_{\boldsymbol{\omega}}) \end{cases} \tag{8}$$

7. *i.e.*, we computed the $p(\boldsymbol{\omega})$ distribution from the dataset $\{\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_n\}$, where each $\boldsymbol{\omega}_i$ is an estimated parameter computed from the trajectory demonstrations.

where K is a gain computed by :

$$K = \Sigma_{\boldsymbol{\omega}} \Phi_{[1:n_o]}^{\top} (\Sigma_{\xi}^{o} + \Phi_{[1:n_o]} \Sigma_{\boldsymbol{\omega}} \Phi_{[1:n_o]}^{\top})^{-1} \tag{9}$$

Equation **??** and **??** can be computed through the marginal and conditional distributions **??**, as detailed in Appendix **??**.

Figure **??** shows the predicted trajectory for the lifting motion of the left arm of iCub. The different graphs show inferred trajectories when the robot observed $n_o = 10, 30, 50, 80\%$ of the total trajectory duration. This example is also available in the toolbox as `demo_plot1DOF.m`. The `nbData` variable changes the percentage of known data. Thus, it will be visible how the inference improves according to this variable. An example of predicted trajectories of the arm lifting in Gazebo can be found in a provided video (see Section **??**).

4.4 Predicting the trajectory time modulation

In the previous section, we presented the general formulation of ProMPs, which makes the implicit assumption that all the observed trajectories have the same duration and thus the same sampling. [8] That is why the duration of the trajectories generated by the RBF is fixed and equal to $\bar{s}$. Of course, this is valid only for synthetic data and not for real data.

To be able to address real experimental conditions, we now consider the variation of the duration of the demonstrated trajectories. To this end, we introduce a time modulation parameter α that maps the actual trajectory duration t_f to $\bar{s}$: $\alpha = \bar{s}/t_f$. The normalized duration $\bar{s}$ can be chosen arbitrarily ; for example it can be set to the average of the duration of the trajectories, *e.g.*, $\bar{s} = mean(t_{f1}, \ldots, t_{fK})$. Notably, in the literature sometimes α is called *phase* **??**. The effect of α is to change the phase of the RBFs, that are scaled in time.

The time modulation of the i-th trajectory Ξ_i is computed by $\alpha_i = \frac{\bar{s}}{t_{fi}}$. Thus, we have $\alpha \cdot t \in [1 : \bar{s}]$. Thus, the improved ProMP model is :

$$\xi_t = \Phi_{\alpha t} \boldsymbol{\omega} + \epsilon_t \,, \tag{10}$$

where $\Phi_{\alpha t}$ is the RBFs matrix evaluated at time αt. All the M Gaussian functions of the RBFs are spread over the same number of samples $\bar{s}$. Thus, we have :

$$\Phi_{\alpha t} = [\psi_1(\alpha t), \psi_2(\alpha t), \ldots, \psi_M(\alpha t)].$$

During the learning step, we record a set of α parameters : $S_\alpha = \{\alpha_1, \ldots, \alpha_n\}$. Then, using this set, we can replay the learned ProMP with different speeds. By default (*e.g.* when $\alpha = 1$), the speed allows to finish the movement in $\bar{s}$ samples.

During the inference, the time modulation α of the partially observed trajectory is not known. Unless fixed a priori, the robot must estimate it. This estimation is critical to ensure a good recognition, as shown in Figure **??** : the inferred trajectory (represented by the mean of the posterior distribution in red) does not have the same duration as the "real" intended trajectory (which is the ground truth). This difference is due to the estimation error of the time modulation parameter. This estimation $\hat{\alpha}$ by default is computed as the

8. Actually, we call here duration what is in fact the total number of samples for the trajectory.

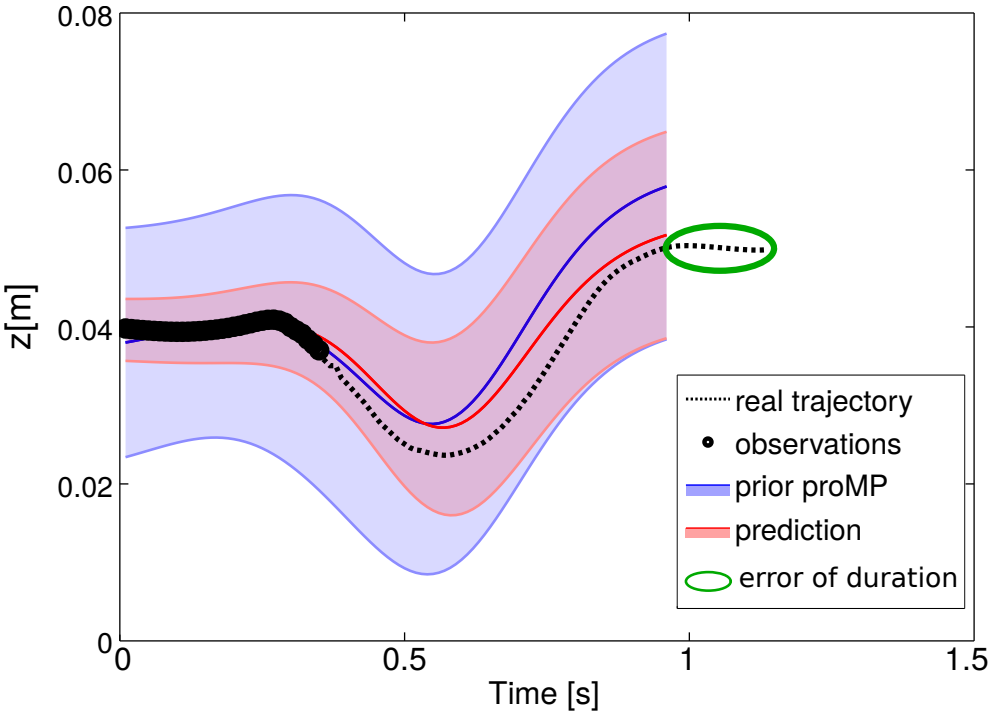

FIGURE 2. This plot shows the predicted trajectory given early observations (data points, in black), compared to the ground truth (*e.g.*, the trajectory that the human intends to execute with the robot). We show the prior distribution (in light blue) and the posterior distribution (in red), which is computed by conditioning the distribution to match the observations. Here, the posterior simply uses the average α computed over the $\alpha_1, \ldots, \alpha_K$ of the K demonstrations. Without predicting the time modulation from the observations and using the average α, the predicted trajectory has a duration that is visibly different from the ground truth.

mean of all the α_k observed during the learning :

$$\hat{\alpha} = \frac{\sum \alpha_k}{n_k}. \tag{11}$$

However, using the mean value for the time modulation is an appropriate choice only when the primitive represents goal-directed motions that are very regular, or for which we can reasonably assume that differences in the duration can be neglected (which is not a general case). In many applications this estimation may be too rough.

Thus, we have to find a way to estimate the duration of the observed trajectory, which corresponds to accurately estimating the time modulation parameter $\hat{\alpha}$. To estimate $\hat{\alpha}$, we implemented four different methods. The first is the **mean of all the** α_k, as in Equation **??**. The second is the **maximum likelihood**, with

$$\hat{\alpha} = \text{argmax}_{\alpha \in S_{\alpha k}} \{\text{loglikelihood}(\Xi^o, \mu_{\boldsymbol{\omega}_k}, \sigma_{\boldsymbol{\omega}_k}, \alpha_k)\}. \tag{12}$$

The third is the **minimum distance** criterion, where we seek the best $\hat{\alpha}$ that minimizes the difference between the observed trajectory Ξ_t^o and the predicted trajectory for the first n_o data points :

$$\hat{\alpha} = \text{argmin}_{\alpha \in S_{\alpha k}} \{\sum_{t=1}^{n_o} |\Xi_t^o - \Phi_{\alpha t} \mu_{\boldsymbol{\omega}_k}|\}. \tag{13}$$

The fourth method is based on a **model** : we assume that there is a correlation between α and the variation of the trajectory δ_{n_o} from the beginning until the time n_o. This "variation" δ_{n_o} can be computed as the variation of the position, *e.g.*, $\delta_{n_o} = X(n_o) - X(1)$, or the variation in the entire trajectory, $\delta_{n_o} = \Xi(n_o) - \Xi(1)$, or any other measure of progress, if this hypothesis is appropriate for the type of task trajectories of the application. [9] Indeed, the α can be linked also to the movement speed, which can be roughly approximated by $\dot{X} = \frac{\delta X}{t_f}$ ($\dot{\Xi} = \frac{\delta \Xi}{t_f}$). We model the mapping between δ_{n_o} and α by :

$$\alpha = \Psi(\delta_{n_o})^\top \boldsymbol{\omega}_\alpha + \epsilon_\alpha, \tag{14}$$

where Ψ are RBFs, and ϵ_α is a zero-mean Gaussian noise. During learning, we compute the $\boldsymbol{\omega}_\alpha$ parameter, using the same method as in Equation **??**. During the inference, we compute $\hat{\alpha} = \Psi(\delta_{n_o})^\top \boldsymbol{\omega}_\alpha$.

A comparison of the four methods for estimating α on a test study with iCub in simulation is presented in Section **??**.

There exist other methods in the literature for computing α. For example, **?** propose a method that models local variability in the speed of execution. In **?** they use a method that improves Dynamic Time Warping by imposing a smooth function on the time alignment mapping using local optimization. These methods will be implemented in the future works.

4.5 Recognizing one among many movement primitives

Robots should not learn only one skills, but many : different skills for different tasks. In our framework, tasks are represented by movement primitives, precisely ProMP. So it is important for the robot to be able to learn K different ProMPs and then be able to recognize from the early observations of a trajectory which of the K ProMPs the observations belong to.

During the learning step of a movement primitive $k \in [1 : K]$, the robot observes different trajectories $S_k = \{\Xi_1, \ldots, \Xi_n\}$. For each ProMP, it learns the distribution over the parameters vector $p(\boldsymbol{\omega}) \sim \mathcal{N}(\mu_{\boldsymbol{\omega}_k}, \Sigma_{\boldsymbol{\omega}_k})$, using Equation **??**. Moreover, the robot records the different phases of all the observed trajectories : $S_{\alpha k} = \{\alpha_{1k}, \ldots, \alpha_{nk}\}$.

After having learned these K ProMPs, the robot can use this information to autonomously execute a task trajectory. Since we are targeting collaborative movements, performed together with a partner at least at the beginning, we want the robot to be able to recognize from the first observations of a collaborative trajectory which is the current task that the partner is doing and what is the intention of the partner. Finally, we want the robot to be able to complete the task on its own, once it has recognized the task and predicted the future trajectory.

Let $\Xi^o = [\Xi_1 \ldots \Xi_{n_o}]^\top$ be the early observations of an initiated trajectory.

From these partial observations, the robot can recognize the "correct" (*i.e.*, most likely) ProMP $\hat{k} \in [1 : K]$. First, for each ProMP $k \in [1 : K]$, it computes the most likely phase (time modulation factor) $\hat{\alpha}_k$ (as explained in Section **??**), to obtain the set of ProMPs with the most likely duration : $S_{[\mu_{\boldsymbol{\omega}_k}, \hat{\alpha}_k]} = \{(\mu_{\boldsymbol{\omega}_1}, \hat{\alpha}_1), \ldots, (\mu_{\boldsymbol{\omega}_K}, \hat{\alpha}_K)\}$

Then we compute the most likely ProMP $\hat{k}$ in $S_{[\mu_{\boldsymbol{\omega}_k}, \hat{\alpha}_k]}$ according to some criterion. One possible way is to minimize the distance between the early observations and the mean of the ProMP for the first portion of

9. In our case, this assumption can be appropriate, because the reaching trajectories in our application are generally monotonic increasing/decreasing.

691 the trajectory :

$$\hat{k} = \arg\min_{k \in [1:K]} \left[\frac{1}{n_o} \sum_{t=1}^{n_o} |\Xi_t - \Phi_{\hat{\alpha}_k t}\, \mu_{\boldsymbol{\omega}_k}| \right] \tag{15}$$

692 In Equation **??**, for each ProMP $k \in [1 : K]$, we compute the average distance between the observed
693 early-trajectory Ξ_t and the mean trajectory of the ProMP $\Phi_{\hat{\alpha}_k t}\mu_{\omega_k}$, with $t = [1 : n_o]$. The most likely
694 ProMP $\hat{k}$ is selected by computing the minimum distance (arg min). Other possible methods for estimating
695 the most likely ProMPs could be inspired by those presented in the previous section for estimating the time
696 modulation, *i.e.* maximum likelihood or learned models.

697 Once identified the $\hat{k}$-th most likely ProMP, we update its posterior distribution to take into account the
698 initial portion of the observed trajectory, using Equation **??** :

$$\begin{cases} \hat{\mu}_{\boldsymbol{\omega}_{\hat{k}}} & = \mu_{\boldsymbol{\omega}_{\hat{k}}} + K(\Xi^o - \Phi_{\hat{\alpha}_{\hat{k}}[1:n_o]}\mu_{\boldsymbol{\omega}_{\hat{k}}}) \\ \hat{\Sigma}_{\boldsymbol{\omega}_{\hat{k}}} & = \Sigma_{\boldsymbol{\omega}_{\hat{k}}} - K(\Phi_{\hat{\alpha}_{\hat{k}}[1:n_o]}\Sigma_{\boldsymbol{\omega}_{\hat{k}}}) \\ K & = \Sigma_{\boldsymbol{\omega}_{\hat{k}}} \Phi^\top_{\hat{\alpha}_{\hat{k}}[1:n_o]} (\Sigma_{\xi^o} + \Phi_{\hat{\alpha}_{\hat{k}}[1:n_o]}\Sigma_{\boldsymbol{\omega}_{\hat{k}}}\Phi^\top_{\hat{\alpha}_{\hat{k}}[1:n_o]})^{-1} \end{cases} \tag{16}$$

699 with $\hat{\alpha}_{\hat{k}}[1 : n_o] = \hat{\alpha}_{\hat{k}}\, t$ (in matrix form), with $t \in [1 : n_o]$.

Finally, the inferred trajectory is given by :

$$\forall t \in [1 : \hat{t}_f], \hat{\xi}(t) = \Phi_t\, \hat{\mu}_{\boldsymbol{\omega}_{\hat{k}}}$$

700 with the expected duration of the trajectory $\hat{t}_f = \hat{\alpha}_k \bar{s}$. The robot is now able to finish the movement
701 executing the most-likely "future" trajectory $\hat{\Xi} = [\hat{\xi}_{n_o+1} \dots \hat{\xi}_{\hat{t}_f}]^\top$.

5 SOFTWARE OVERVIEW

702 In this section, we introduce our open-source software with an overview of its architecture. This software
703 is composed of two main modules, represented in Figure **??**.

704 While the robot is learning the Probabilistic Movement Primitives (ProMPs) associated to the different
705 tasks, the robot is controlled by its user. The user's guidance can be either manual for the real iCub, or
706 through a haptic device for the simulated robot.

707 A Matlab module allows replaying movement primitives or finishing a movement that has been initiated
708 by its user. By using this module, the robot can learn distributions over trajectories, replay movement
709 primitives (using the mean of the distribution), recognize the ProMP that best matches a current trajectory,
710 and infer the future evolution (until the end target) of this trajectory.

711 A C++ module forwards to the robot the control that comes either from the user or from the Matlab
712 module. Then, the robot is able to finish a movement initiated by its user (directly or through a haptic
713 device) in an autonomous way, as shown in Figure 1.

714 We present the C++ module in Section **??** and the theoretical explanation of the Matlab module algorithms
715 in Section 4. A guide to run this last module is first presented in Section **??** for a simple example, and
716 in Section **??** for our application, where a simulated robot learns many measured information of the
717 movements. Finally, we present results on the real iCub application in Section **??**.

718 Our software is available through the GPL licence, and publicly available at :

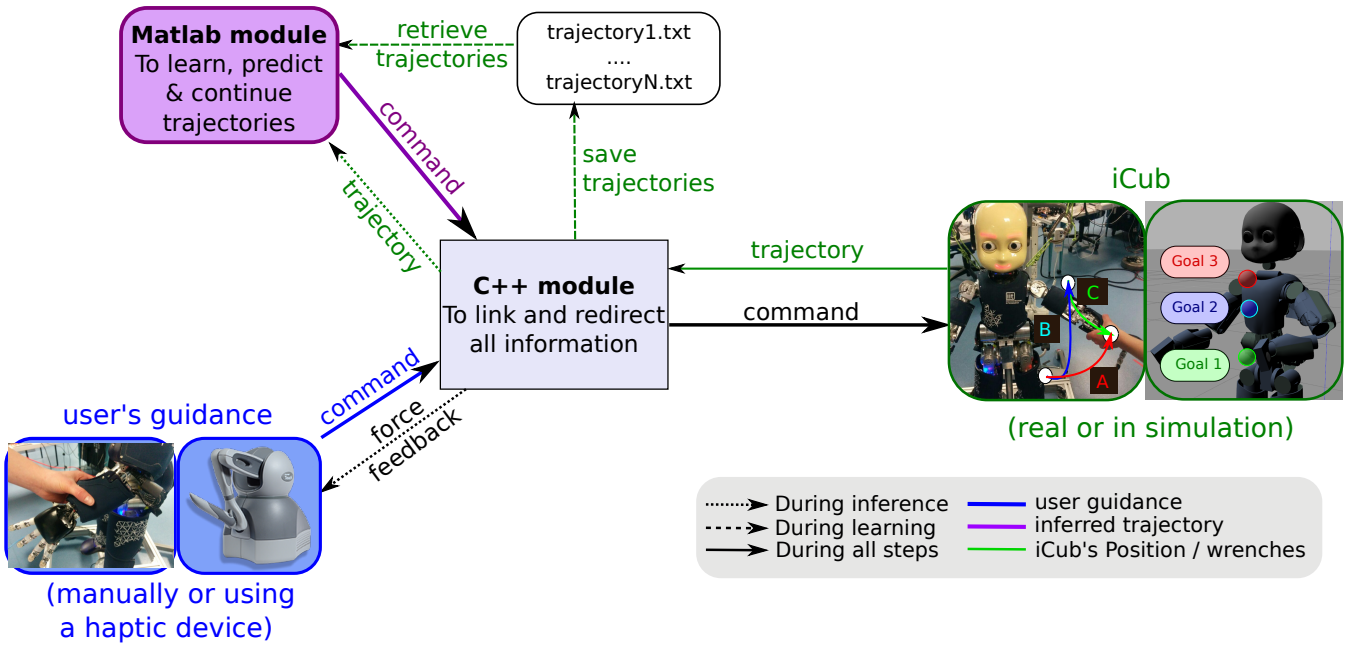

FIGURE 3. Software architecture and data-flows.
The robot's control is done either by the user's guidance (manually or through a haptic device) represented in blue, or by the Matlab module, in purple. The C++ module handles the control source to command the robot, as represented in black. Moreover, this module forwards information that comes from the iCub.

`https://github.com/inria-larsen/icubLearningTrajectories`.

Tutorial, readme and videos can be found in that repository. First, the readme file describes how to launch simple demonstrations of the software. Videos present these demonstrations to simplify the understanding. In the next sections, we detail the operation of the demo program for a first case of 1DOF primitive, followed by the presentation of the specific applications on the iCub (first simulated and then real).

6 SOFTWARE EXAMPLE : LEARNING A 1-DOF PRIMITIVE

In this section, we present the use of the software to learn ProMPs in a simple case of 1-DOF primitive. This example only uses the *MatlabProgram* folder, composed of :

—A sub-folder called "Data", where there are trajectory sets used to learn movement primitives. These trajectories are stored in text files with the following information :
-**input parameters** : # $input_1$ # $input_2$ [...]
-**input parameters with time-step** : # timeStep # $input_1$ # $input_2$ [...]
-***recordTrajectories.cpp* program recording** : See Section **??** for more information.

—A sub-folder called "used_functions". It contains all the functions used to retrieve trajectories, compute ProMPs, infer trajectories, and plot results. Normally, using this toolbox does not require understanding these functions. The first lines of these functions give an explanation of their functioning and precise what are the input(s) and output(s) parameters.

—Matlab scripts called "demo_*.m". They are simple examples of how to use this toolbox.

¡¡¡¡¡¡¡ HEAD The script `demo_plot1DOF.m`, can be used to compute a ProMP and to continue an initiated movement. The ProMP is computed from a dataset stored in a ".mat" file, called

traj1_1DOF.mat. In this script, variables are first defined to make the script specific to the current dataset :

Variable assignation	Commentary
DataPath'Datatraj1_1DOF.mat' ;	Can be either ".mat" or ".txt". In the current demo, you can also write DataPath = 'Data/traj1' if you want to use the text files of this dataset.
typeRecover= '.mat'	Or .txt, it depends on your choice of data file.
inputName = 'z[m]' ;	Label of your input(s). Here z represents the z-axis Cartesian coordinate.
s_ref=100 ;	Number of samples used as reference to rescale all the trajectories. to the same duration.
nbInput = 1 ;	Dimension of the generic vector containing the state of the trajectory.
M = 5 ;	Number of radial basis functions per input.
expNoise = 0.00001 ;	Expected trajectory noise.
percentData = 20 ;	Percent of observed data during the inference.

The variables include :

—`DataPath` is the path to the recorded data. If the data are stored in text files, this variable contains the folder name where text files are stored. These text files are called "record$X$.txt", with $X \in [0 : n-1]$ if there are n trajectories. One folder is used to learn one ProMP. If the data are already loaded from a ".mat" file, write the whole path with the extension. The data in ".mat" matches with the output of the Matlab function `loadTrajectory`.

—`nbInput`= D is the dimension of the input vector ξ_t.

—`expNoise` = Σ_ξ^o is the expected noise of the initiated trajectory. The smaller this variable is, the stronger the modification of the ProMP distribution will be, given new observations.

We will now explain more in detail the script. To recover data recorded in a ".txt" file, we call the function :

```
t{1} = loadTrajectory(PATH, nameT, varargin)
```

Its input parameters specify the path of the recorded data, the label of the trajectory. Other information can be added by using the `varargin` variable (for more detail, check the header of the function with the help comments). The output is an object that contains all the information about the demonstrated trajectories. It is composed of `nbTraj`, the number of trajectory ; `realTime`, the simulation time ; `y` (and `yMat`), the vector (and matrix) trajectory set , etc.. Thus, `t{1}.y{i}` contains the i-th trajectory.

The Matlab function `drawRecoverData(t{1}, inputName,'namFig', nFig, varargin)` plots in a Matlab figure (numbered `nFig`) the dataset of loaded trajectories. An example is shown in Figure **??**, on the left. Incidentally, the different duration of the trajectories is visible : on average, it is 1.17 ± 0.42 seconds.

To split the entire dataset of demonstrated trajectories `t{1}` into a training dataset (used for learning the ProMPs) and a test dataset (used for the inference), call the function

```
[train, test] = partitionTrajectory(t{1}, partitionType, percentData, s_ref)
```

where if `partitionType`$= 1$, only one trajectory is in the test set and the others are placed in the training set, and if `partitionType`> 1 it corresponds to the percentage of trajectories that will be included in the training set.

The ProMP can be computed from the training set by using the function :

```
promp = computeDistribution(train, M, s_ref,c,h)
```

The output variable `promp` is an object that contains all the ProMP information. The first three input parameters have been presented before : `train` is the training set, `M` is the number of RBFs, `s_ref` is the number of samples used to rescale all the trajectories. The last two input parameters `c` and `h` shape the RBFs of the ProMP model : `c` $\in \mathbb{R}^M$ is the center of the Gaussians and `h` $\in \mathbb{R}$ their variance.

To visualize this ProMP, as shown in Figure **??**, call the function : `drawDistribution(promp, ... inputName,s_ref)`

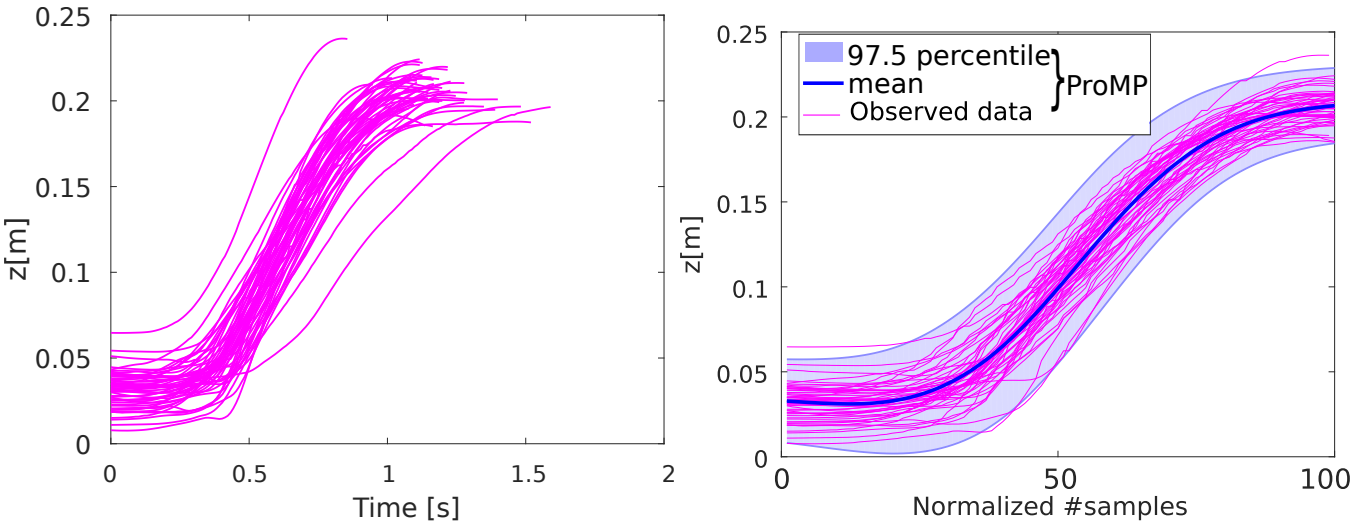

FIGURE 4. The observed trajectories are represented in magenta. The corresponding ProMP is represented in blue. The following parameters are used : $\bar{s} = 100$ for the reference number of samples, $M = 5$ for the number of RBFs spread over time, and $h = 0.04$ (=$\frac{1}{M^2}$) the variance of the RBFs.

For debugging purposes and to understand how to tune the ProMPs' parameters, it is interesting to plot the overlay of the basis functions in time. Choosing an appropriate number of basis functions is important, as too few may be insufficient to approximate the trajectories under consideration, and too many could result in over-fitting problems. To plot the basis functions, simply call :

```
drawBasisFunction(promp.PHI,M)
```

where `promp.PHI` is a set of RBFs evaluated in the normalized time range $t \in [1 : \bar{s}]$.

Figure **??** shows at the top the basis functions before normalization, and at the bottom the ProMP modeled from these basis functions. From left to right, we change the number of basis functions. When there are not enough basis functions (left), the model is not able to correctly represent the shape of the trajectories. In the middle, the trajectories are well represented by the five basis functions. With more basis functions (right), the variance of the distribution decreases because the model is more accurate. However, arbitrarily increasing the number of basis functions is not a good idea, as it may not improve the accuracy of the model and worse it may cause overfitting.

Once the ProMP is learned, the robot can reproduce the movement primitive using the mean of the distribution. Moreover, it can now recognize a movement that has been initiated in this distribution, and predict how to finish it. To do so, given the early n_o observations of a movement, the robot updates the prior distribution to match the early observed data points : through conditioning, it finds the posterior distribution, that can be used by the robot to execute the movement on its own.

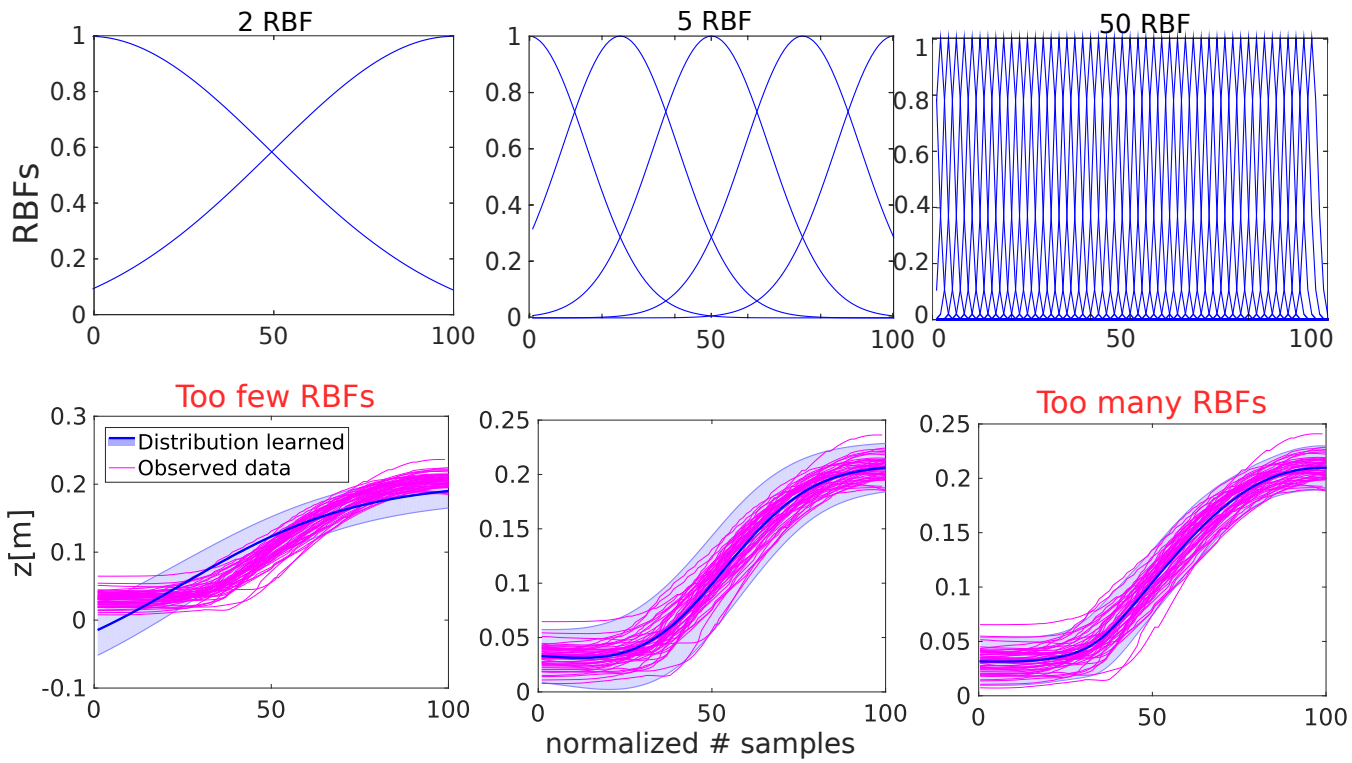

FIGURE 5. The ProMP computed for the test dataset (Figure **??**) using different numbers of basis functions : from left to right : $M = \{2, 5, 50\}$ basis functions before normalization, with a variance $h = \frac{1}{M^2}$.

The first step in predicting the evolution of the trajectory is to infer the duration of this trajectory, which is encoded by the time modulation parameter $\hat{\alpha}$. The computation of this inference, which was detailed in Section **??**, can be done by using the function :

```
[expAlpha,type,x]=inferenceAlpha(promp,test{1},M,s_ref,c,h,test{1}.nbData, ...
expNoise, typeReco)
```

where `typeReco` is the type of criteria used to find the expected time modulation ('MO', 'DI' or 'ML' for model, distance or maximum likelihood methods) ; `expAlpha` $= \hat{\alpha}$ is the expected time modulation ; `type` is the index of the ProMP from which `expAlpha` has been computed, which we note in this paper as k . To predict the evolution of the trajectory, we use Equation **??** from Section **??**. In Matlab, this is done by the function : `infTraj = inference(promp, test{1}, M, s_ref, c, h, ... test{1}.nbData, expNoise, expAlpha)` where `test{1}.nbData` has been computed during the `partitionTrajectory` step. This variable is the number of observations n_o, representing the percentage of observed data (`percentData`) of the test trajectory (*i.e.*, to be inferred) that the robot observes. `infTraj`= $\hat{\Xi}$ is the inferred trajectory. Finally, to draw the inferred trajectory, we can call the function `drawInference(promp,inputName,infTraj, ... test1,s_ref)`.

It can be interesting to plot the quality of the predicted trajectories as a function of the number of observations, as done in Figure **??**.

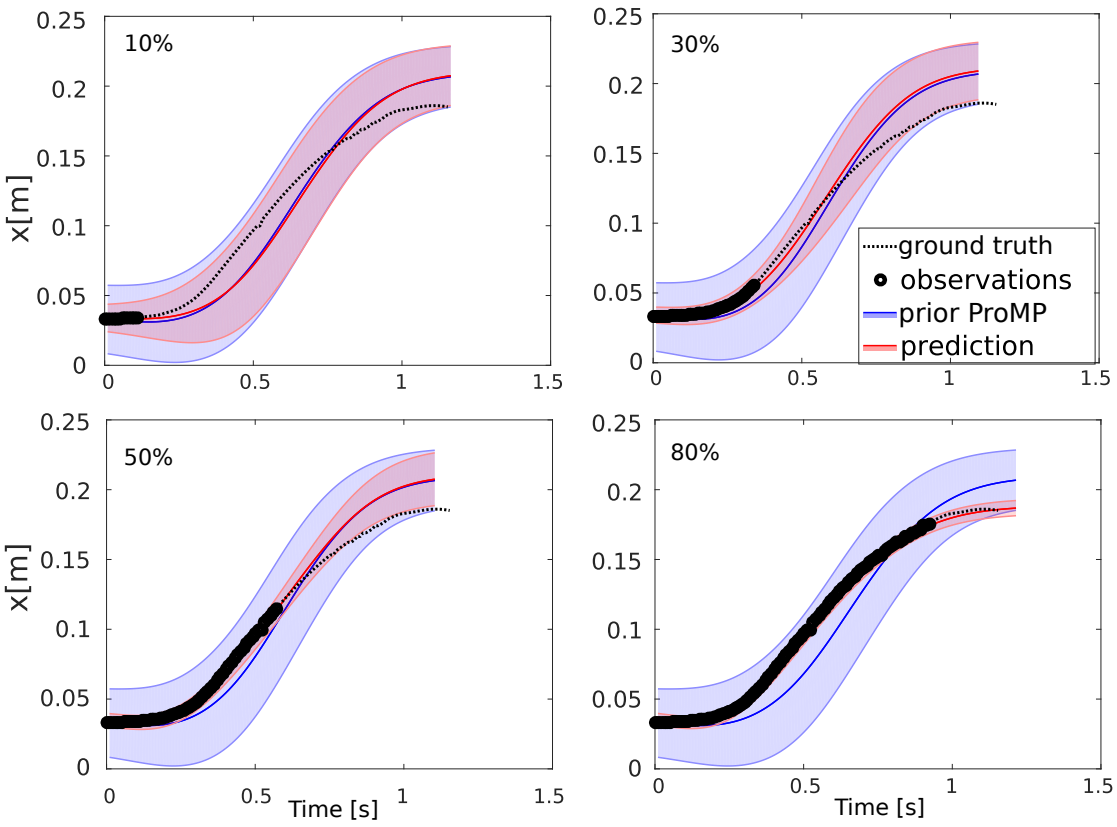

FIGURE 6. The prediction of the future trajectory given early observations, exploiting the information of the learned ProMP (Figure **??**). The plots show the predicted trajectories after 10%, 30%, 50% and 80% of observed data points.

Note that when we have observed a larger portion of the trajectory, the prediction of the remaining portion is more accurate.

Now we want to measure the quality of the prediction. Let $\Xi^* = [\xi^o(1), \ldots, \xi^o(n_o), \xi^*(n_o + 1), \ldots, \xi^*(t_f^*)]$ be the real trajectory expected by the user. To measure the quality of the prediction, we can use :

—The likelihood of having the Ξ^* trajectory given the updated distribution $p(\hat{\boldsymbol{\omega}})$.

—The distance between the Ξ^* trajectory and the $\hat{\Xi}$ inferred trajectory.

However, according to the type of recognition `typeReco` used to estimate the time modulation parameter α from the early observations, a visible mismatch between the predicted trajectory and the real one can be visible even when a lot of observations are used. This is due to the error of the expectation of this time modulation parameter. In Section **??**, we present the different methods used to predict the trajectory duration. These methods select the most likely $\hat{\alpha}$ according to different criteria : distance ; maximum likelihood ; model of the α variable [10] ; and average of the observed α during learning.

Figure **??** shows the different trajectories predicted after $n_o = 40\%$ of the length of the desired trajectory is observed, according to the method used to estimate the time modulation parameter.

10. In this model, we assume that we can find the time modulation parameter according to the global variation of the position during the n_o first observed data.

	Traj. Samples	$\alpha = \frac{s}{\text{Iterations}}, \bar{s} = 100$	Duration [s]
Min	83	1.2048	0.83
Max	115	0.8696	1.15
Mean	100	1	0.99
Std deviation	9	11.1111	0.09

Table 2. information about trajectories' duration

On this simple test, where the variation time is little as shown in Table **??**, the best result is accomplished by the average of time modulation parameter of the trajectories used during the learning step. In more complicated cases, when the time modulation varies, the other methods will be preferable as seen in Section **??**.

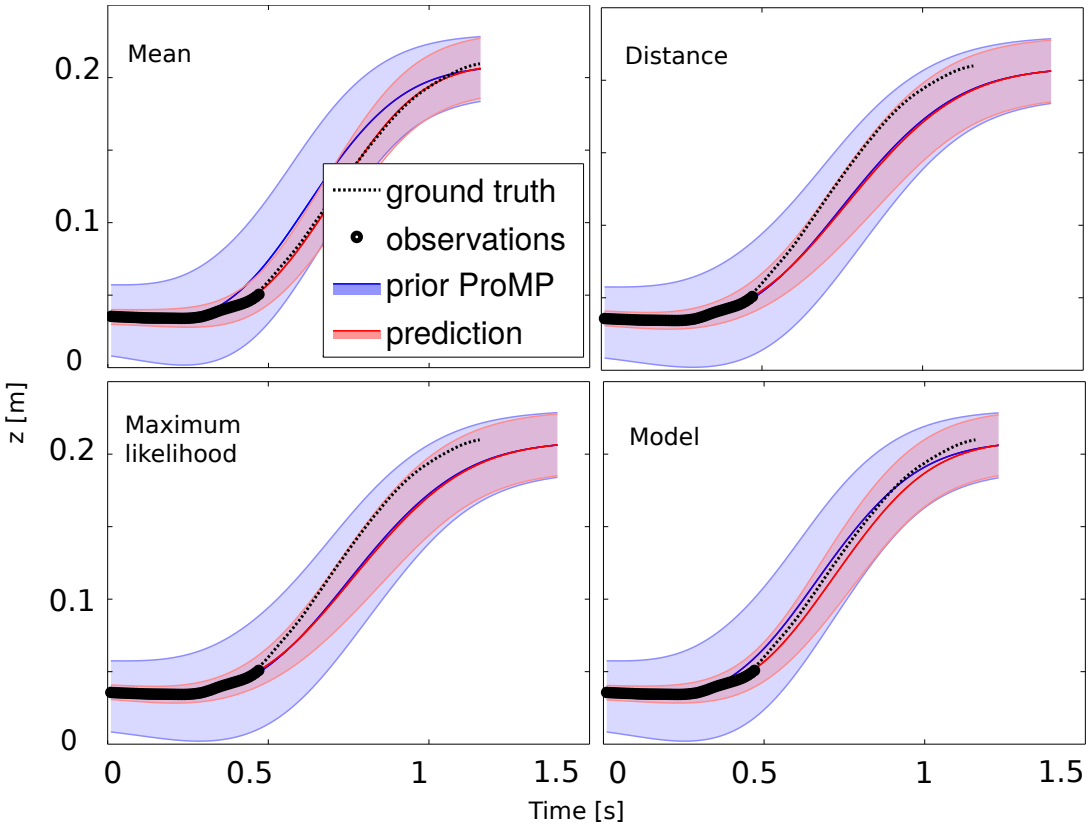

FIGURE 7. The prediction of the future trajectory given $n_o = 40\%$ of early observations from the learned ProMP computed for the test dataset (Figure **??**). The plots show the predicted trajectory, using different criteria to estimate the best phases of the trajectory : using the average time modulation (Equation **??**) ; using the distance criteria (Equation **??**) ; using the maximum log-likelihood (Equation **??**) ; or using a model of time modulation according to the time variation (Equation **??**).

7 APPLICATION ON THE SIMULATED ICUB : LEARNING THREE PRIMITIVES

In this application, the robot learns multiple ProMPs and is able to predict the future trajectory of a movement initiated by the user, assuming the movement belongs to one of the learned primitives. Based on this prediction, it can also complete the movement once it has recognized the appropriate ProMP.

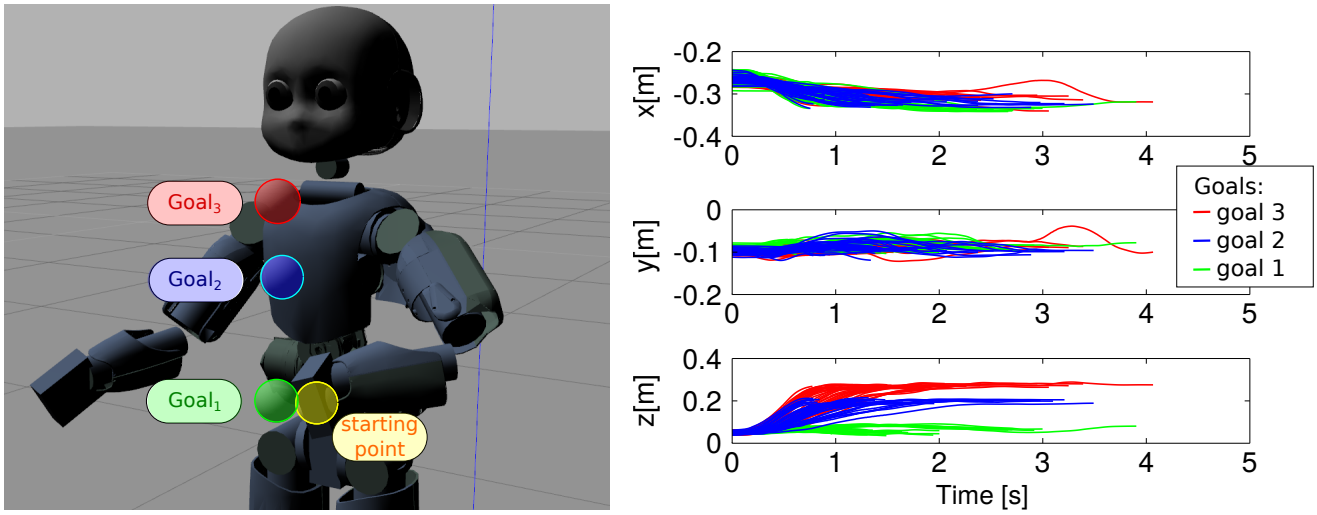

FIGURE 8. Left : the three colored targets that the robot must reach from the starting point ; the corresponding trajectories are used to learn three primitives representing three skills. Right : the Cartesian position information of the demonstrated trajectories for the three reaching tasks.

We simplify the three actions/tasks by reaching three different targets, represented by three colored balls in the reachable workspace of the iCub. The example is performed with the simulated iCub in Gazebo. Figure **??** shows the three targets, placed at different heights in front of the robot.

In Section **??** we formulate the intention recognition problem for the iCub : the problem is to learn the ProMP from trajectories consisting of Cartesian positions in 3D [11] and the 6D wrench information measured by the robot during the movement. In Section **??** we describe the simulated setup of iCub in Gazebo, then in Section **??** we explain how trajectories are recorded, including force information, when we use the simulated robot.

7.1 Predicting intended trajectories by using ProMPs

The model is based on Section 4, but here we want to learn more information during movements. We record this information in a multivariate parameter vector :

$$\forall t, \xi_t = \begin{bmatrix} X_t \\ F_t \end{bmatrix} \in \mathbb{R}^9$$

Were $X_t \in \mathbb{R}^3$ is the Cartesian position of the robot's end effector and $F_t \in \mathbb{R}^6$ the external forces and moments. In particular, F_t contains the user's contact forces and moments. Let us call dim(ξ_t)$= D$, the dimension of this parameter vector.

The corresponding ProMP model is :

$$\xi_t = \begin{bmatrix} X_t \\ F_t \end{bmatrix} = \Phi_{\alpha t} \boldsymbol{\omega} + \epsilon_t$$

11. Note that in that particular example we do not use the orientation because we want the robot's hand to keep the same orientation during the movement. But in principle, it is possible to learn trajectories consisting of the 6D/7D Cartesian position and orientation of the hand, to make the robot change also the orientation of the hand during the task.

Where $\boldsymbol{\omega} \in \mathbb{R}^{D \cdot M}$ is the time independent parameter vector, $\epsilon_t = \begin{bmatrix} \epsilon_{X_t} \\ \epsilon_{F_t} \end{bmatrix} \in \mathbb{R}^D$ is the zero-mean Gaussian i.i.d. observation noise, and $\Phi_{\alpha t} \in \mathbb{R}^{D \times D \cdot M}$ a matrix of Radial Basis Functions (RBFs) evaluated at time αt.
Since we are in the multidimensional case, this $\Phi_{\alpha t}$ block diagonal matrix is defined as :

$$\Phi_{\alpha t} = BlockdiagonalMatrix(\phi_1, \ldots, \phi_D) \in \mathbb{R}^{D \times D \cdot M}$$

It is a diagonal matrix of D Radial Basis Functions (RBFs), where each RBF represents one dimension of the ξ_t vector and it is composed of M Gaussians, spread over same number of samples $\bar{s}$.

7.1.1 Learning motion primitives

During the learning step of each movement primitive $k \in [1:3]$, the robot observes different trajectories $S_k = \{\Xi_1, \ldots, \Xi_n\}_k$, as presented in Section **??**.
For each trajectory $\Xi_{i[1:t_{fi}]} = [\xi_{i(1)}, \ldots, \xi_{i(tf_i)}]^\top$, it computes the optimal $\boldsymbol{\omega}_{ki}$ parameter vector that best approximates the trajectory.

We saw in Section **??** how these computations are done. In our software, we use matrix computation instead of t_{fi} iterative ones done for each observation t (as in Equation **??**). Thus, we have :

$$\boldsymbol{\omega}_{ki} = (\Phi^\top_{\alpha[1:t_{fi}]} \Phi_{\alpha[1:t_{fi}]})^{-1} \Phi^\top_{\alpha[1:t_{fi}]} * \Xi_{i[1:t_{fi}]} \tag{17}$$

with $\Phi_{\alpha[1:t_{fi}]} = [\Phi_{\alpha 1}, \Phi_{\alpha 2} \ldots, \Phi_{\alpha t_{fi}}]^\top$.

7.1.2 Prediction of the trajectory evolution from initial observations

After having learned the three ProMPs, the robot is able to finish an initiated movement on its own. In Sections **??**, **??** and **??** we explained how to compute the future intended trajectory given the early observations.

In this example, we add specificities about the parameters to learn.

Let $\Xi^o = \begin{bmatrix} X^o \\ F^o \end{bmatrix} = [\Xi_1 \ldots \Xi_{n_o}]^\top$ be the early observations of the trajectory.

First, we only consider the partial observations : $X^o = [X_1 \ldots X_{n_o}]^\top$. Indeed, we assume the recognition of a trajectory is done with Cartesian position information only, because the same movement can be done and recognized with different force profiles than the learned ones.

From this partial observation X^o, the robot recognizes the current ProMP $\hat{k} \in [1:3]$, as seen in Section **??**. It also computes an expectation of the time modulation $\hat{t}_f$, as seen in Section **??**. Using the expected value of the time modulation, it approximates the trajectory speed and its total time duration.

Second, we use the total observation Ξ^o to update the ProMP, as seen in Section **??**. This computation is based on equation **??**, but here again, we use matrix computation :

$$\begin{cases} \hat{\mu}_{\boldsymbol{\omega}_k} &= \mu_{\boldsymbol{\omega}_k} + K(\Xi^o - \Phi_{\alpha[1:n_o]} \mu_{\boldsymbol{\omega}_k}) \\ \hat{\Sigma}_{\boldsymbol{\omega}_k} &= \Sigma_{\boldsymbol{\omega}_k} - K(\Phi_{\alpha[1:n_o]} \Sigma_{\boldsymbol{\omega}_k}) \\ K &= \Sigma_{\boldsymbol{\omega}_k} \Phi^\top_{\alpha[1:n_o]} (\Sigma_{\xi^o} + \Phi_{\alpha[1:n_o]} \Sigma_{\boldsymbol{\omega}_k} \Phi^\top_{\alpha[1:n_o]})^{-1} \end{cases}$$

From this posterior distribution, we retrieve the inferred $\hat{\Xi} = \{\hat{\xi}_1, ..., \hat{\xi}_{\hat{t}_f}\}$ trajectory, with :

$$\forall t \in [1 : \hat{t}_f], \hat{\xi}_t = \begin{bmatrix} \hat{X}_t \\ \hat{F}_t \end{bmatrix} = \Phi_{\alpha t} \hat{\mu}_{\boldsymbol{\omega}_k}$$

Note that the inferred wrenches $\hat{F}_t$, here, correspond to the simulated wrenches in Gazebo. In this example there is little use for them in simulation ; the interest for predicting also wrenches will be clearer in Section **??**, with the example on the real robot.

7.2 Setup for simulated iCub

For this application, we created a prototype in Gazebo, where the robot must reach three different targets with the help of a human. To interact physically with the robot simulated in Gazebo, we used the Geomagic touch, a haptic device.

The setup consists of :

—the iCub simulation in Gazebo, complete with the dynamic information provided by *wholeBodyDynamicsTree* (`https://github.com/robotology/codyco-modules/tree/master/src/modules/wholeBodyDynamicsTree`) and the Cartesian information provided by *iKinCartesianController* ;

—the Geomagic Touch, installed following the instructions in `https://github.com/inria-larsen/icub-manual/wiki/Installation-with-the-Geomagic-Touch`, which not only install the SDK and the drivers of the GeoMagic but also point to how to create the yarp drivers for the Geomagic ;

—a C++ module (`https://github.com/inria-larsen/icubLearningTrajectories/tree/master/CppProgram`) that connects the output command from the Geomagic to the iCub in Gazebo, and eventually enables recording the trajectories on a file. A tutorial is included in this software.

The interconnection among the different modules is represented in Figure **??**, where the Matlab module is not used. The tip of the Geomagic is virtually attached to the end-effector of the robot :

$$x_{geo} \rightarrow x_{icub\_hand}$$

When the operator moves the Geomagic, the position of the Geomagic tip x_{geo} is scaled (1 :1 by default) in the iCub workspace as x_{icub_hand}, and the Cartesian controller is used to move the iCub hand around a "home" position, or default starting position :

$$x_{icub\_hand} = hapticDriverMapping(x_0 + x_{geo})$$

where *hapticDriverMapping* is the transformation applied by the haptic device driver, which essentially maps the axis from the Geomagic reference frame to the iCub reference frame. By default, no force feedback is sent back to the operator in this application, as we want to emulate the zero-torque control mode of the real iCub, where the robot is ideally transparent and not opposing any resistance to the human guidance. A default orientation of the hand ("katana" orientation) is set.

7.3 Data acquisition

The dark button of the Geomagic is used to start and stop the recording of the trajectories. The operator must click and hold the button during the whole movement and release the button at the end. The trajectory is saved on a file called *recordX.txt* for the X-th trajectory. The structure of this file is :

```
#time #xgeo #ygeo #zgeo #fx #fy #fz #mx #my #mz #x_icub_hand #y_icub_hand #z_icub_hand
5.96046e-06 -0.0510954 -0.0127809 -0.0522504 0.284382 -0.0659538 -0.0239582 -0.0162418 -0.0290078 ...
    -0.0607215 -0.248905 -0.0872191 0.0477496$
```

A video showing the iCub's arm moved by an user through the haptic device in Gazebo is available in Section **??** (tutorial video). The graph in Figure **??** represents some trajectories recorded with the Geomagic, corresponding to lifting the left arm of the iCub.

Demonstrated trajectories and their corresponding forces can be recorded directly from the robot, by accessing the Cartesian interface and the *wholeBodyDynamicsTree* module. [12]

In our project on Github, we provide the acquired dataset with the trajectories for the interested reader who wishes to test the code with these trajectories. Two datasets are available at `https://github.com/inria-larsen/icubLearningTrajectories/tree/master/MatlabProgram/Data/` : the first dataset called "heights" is composed of three goal-directed reaching tasks, where the targets vary in height ; the second dataset called "FLT" is composed of trajectories recorded on the real robot, whose arms moves forward, to the left and to the top.

A matlab script that learns ProMPs with such kinds of datasets is available in the toolbox, called `demo_plotProMPs.m`. It contains all the following steps.

To load the first "heights" dataset with the three trajectories, write :

```
t{1} = loadTrajectory('Data/heights/bottom', 'bottom', 'refNb', s_bar, 'nbInput',nbInput, ...
    'Specific', 'FromGeom');
t{2} = loadTrajectory('Data/heights/top', 'top', 'refNb', s_bar, 'nbInput',nbInput, 'Specific', ...
    'FromGeom');
t{3} = loadTrajectory('Data/heights/middle', 'forward', 'refNb', s_bar, 'nbInput',nbInput, ...
    'Specific', 'FromGeom');
```

Figure **??** shows the three sets of demonstrated trajectories. In the used dataset called "heights", we have recorded 40 trajectories per movement primitive.

7.4 Learning the ProMPs

We need to first learn the ProMPs associated to the three observed movements. First, we partition the collected dataset into a training set and test dataset for the inference. One random trajectory for the inference is used :

```
[train{i},test{i}] = partitionTrajectory(t{i},1,percentData,s_bar);
```

12. In our example, we do not use the simulated wrench information as it is very noisy. However, we provide the code and show how to retrieve it and use it, in case the readers should not have access to the real iCub.

The second input parameter specifies that we select only one trajectory, randomly selected, to test the ProMP.

Now, we compute the three ProMPs with :

```
promp{1} = computeDistribution(train{1}, M, s_bar,c,h);
promp{2} = computeDistribution(train{2}, M, s_bar,c,h);
promp{3} = computeDistribution(train{3}, M, s_bar,c,h)
```

We set the following parameters :

—`s_bar=100` : reference number of samples, which we note in this paper as $\bar{s}$.

—`nbInput(1)= 3; nbInput(2)= 6` : dimension of the generic vector containing the state of the trajectories. It is composed of 3D Cartesian position and 6D forces and wrench information. [13]

—`M(1)= 5; M(2)= 5` : number of basis functions for each `nbInput` dimension.

—`c = 1/M;h = 1/(M*M)` : RBF parameters (see Equation **??**).

—`expNoise = 0.00001` : the expected data noise. We assume this noise to be very low, since this is a simulation.

—`percentData = 40` : this variable specifies the percentage of the trajectory that the robot will be observed, before infering the end.

These parameters can be changed at the beginning of the Matlab script.

Figure **??** shows the three ProMPs of the reaching movements towards the three targets. To highlight the most useful dimension, we only plot the z-axis Cartesian position. However, each dimension is plotted by the Matlab script with :

```
drawRecoverData(t{1}, inputName, 'Specolor','b','namFig', 1);
drawRecoverData(t{1}, inputName, 'Interval', [4 7 5 8 6 9], 'Specolor','b','namFig',2);
drawRecoverData(t{2}, inputName, 'Specolor','r','namFig',1);
drawRecoverData(t{2}, inputName, 'Interval', [4 7 5 8 6 9], 'Specolor','r','namFig',2);
drawRecoverData(t{3}, inputName, 'Specolor','g','namFig',1);
drawRecoverData(t{3}, inputName, 'Interval', [4 7 5 8 6 9], 'Specolor','g','namFig',2);
```

7.5 Predicting the desired movement

Now that we have learned the different ProMPs, we can predict the end of a trajectory according to the early observation n_o. This number is computed from the variable `percentData` written at the beginning of the trajectory by : $n_o = |\frac{percentData}{100} * t_{fi}|$, where i is the index of the test-trajectory

To prepare the prediction, the model the time modulation of each trajectory is computed with :

```
    w = computeAlpha(test.nbData,t, nbInput);
    promp{1}.w_alpha= w{1};
    promp{2}.w_alpha= w{2};
    promp{3}.w_alpha= w{3};
```

13. Note that in our example wrenches are separated from the Cartesian position, because they are not used to recognize the index of the current ProMP during the inference.

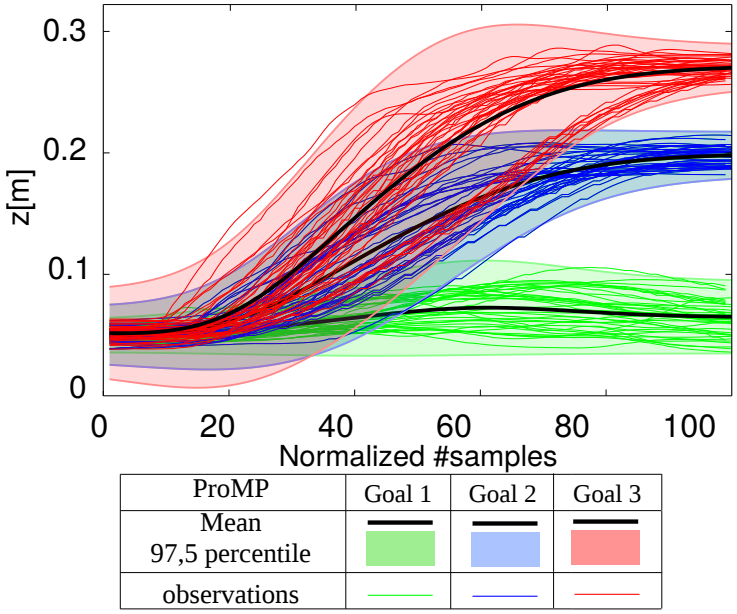

FIGURE 9. The Cartesian position in the z-axis of the three ProMPs corresponding to reaching three targets. There are 39 trajectory demonstrations per each ProMPm with M=5 basis functions, $c = \frac{1}{M}, h = \frac{1}{M^2}$ and $\bar{s} = 100$.

This model relies on the global variation of Cartesian position during the first n_o observations. The model's computations are explained in Section **??**.

Now, to estimate the time modulation of the trajectory, call the function :

```
[alphaTraj,type, x] = inferenceAlpha(promp,test{1},M,s_bar,c,h,test{1}.nbData, expNoise, 'MO');
```

Where `alphaTraj` contains the estimated time modulation $\hat{\alpha}$ and `type` gives the index of the recognized ProMP. The last parameter `x` is used for debbuging purposes.

Using this estimation of the time modulation, the end of the trajectory is inferred with :

```
infTraj = inference(promp, test{1}, M, s_bar, c, h, test{1}.nbData, expNoise, alphaTraj);
```

As shown in the previous example, the quality of the prediction of the future trajectory depends on the accuracy of the time modulation estimation. This estimation is computed by calling the function :

```
%Using the model:
[alphaTraj,type, x] = inferenceAlpha(promp,test{1},M,s_bar,c,h,test{1}.nbData, expNoise, 'MO');
%Using the distance criteria:
[alphaTraj,type, x] = inferenceAlpha(promp,test{1},M,s_bar,c,h,test{1}.nbData, expNoise, 'DI');
%Using the Maximum likelihood criteria:
[alphaTraj,type, x] = inferenceAlpha(promp,test{1},M,s_bar,c,h,test{1}.nbData, expNoise, 'ML');
%Using the mean of observed temporal modulation during learning:
alphaTraj = (promp{1}.mu_alpha + promp{2}.mu_alpha + promp{3}.mu_alpha) /3.0;
```

7.6 Predicting the time modulation

In Section **??** we presented four main methods for estimating the time modulation parameter, discussing why this is crucial for a better estimation of the trajectory. Here, we compare the methods on the three goals experiment. We recorded 40 trajectories for each movement primitive, for a total of 120 trajectories. After having computed the corresponding ProMPs, we tested the inference by providing early observations of a trajectory that the robot must finish. For that purpose, it recognizes the correct ProMP among the three precedently learned (see Section **??**) and then it estimates the time modulation parameter $\hat{\alpha}$. Figure **??** represents the average error of the $\hat{\alpha}$ during inference for 10 trials according to the number of observations (from 30% to 90% of observed data) and according to the used method. These methods are the ones we have just presented before that we called mean (Equation **??**), maximum likelihood (Equation **??**), minimum distance (Equation **??**) or model (Equation **??**). Each time, the tested trajectory is chosen randomly from the data set of observed trajectories (of course, the test trajectory does not belong to the training set, so it was not used in the learning step). The method that takes the average of α observed during learning is

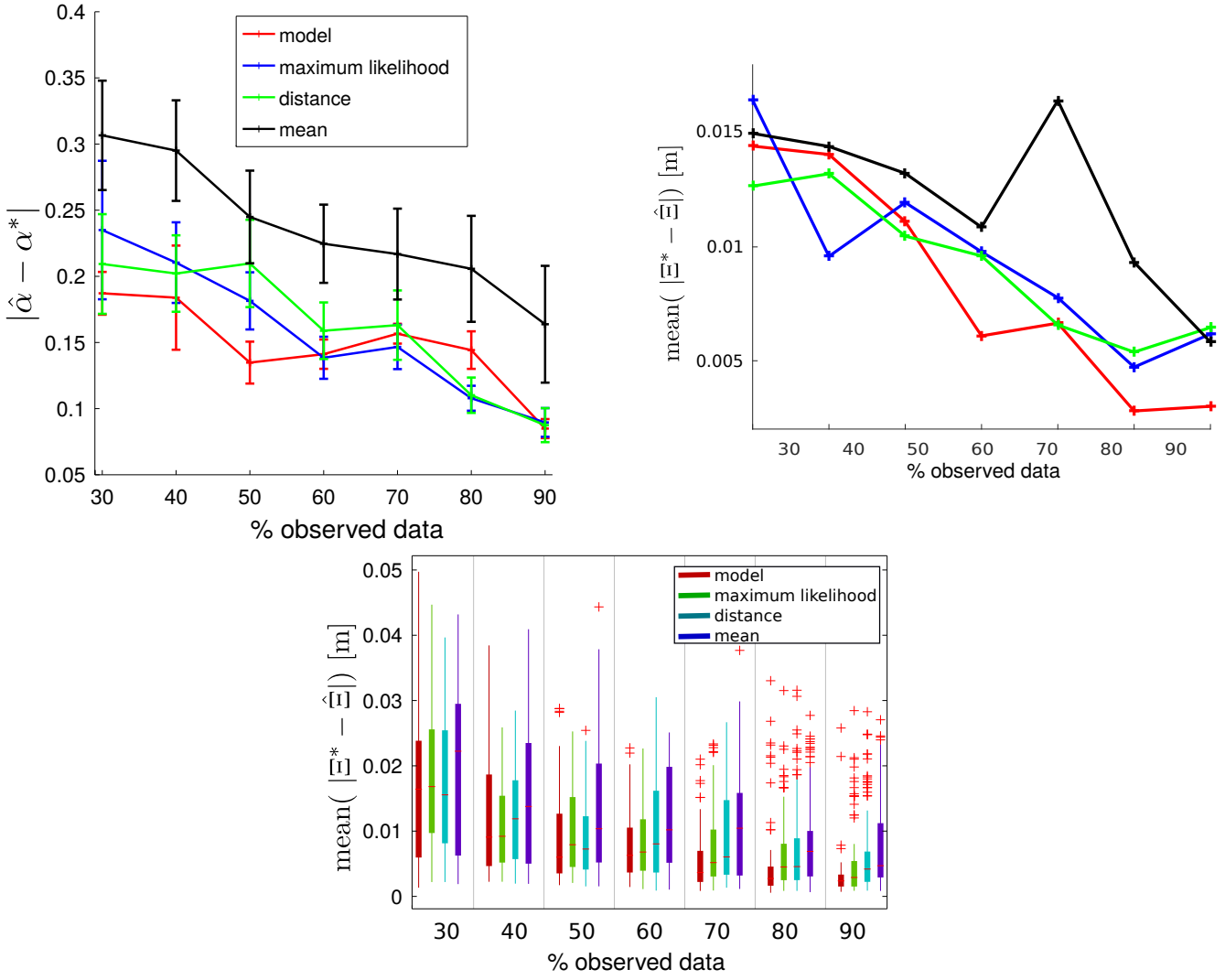

FIGURE 10. (top left) Error of α estimation ; (top right and bottom) error of trajectory prediction according to the number of known data and the method used. We executed 10 different trials for each case.

taken as comparison (in black). We can see that other methods are more accurate. The *maximum likelihood*

is increasingly more accurate, as expected. The fourth method (*model*) that models the α according to the global variation of the trajectory's positions during the early observations is the best performing when the portion of observed trajectory is small (*e.g.*, 30%-50%). Since it is our interest to predict the future trajectory as early as possible, we adopted the *model* method for our experiments.

8 APPLICATION ON THE REAL ICUB

In this section we present and discuss two experiments with the real robot iCub.

In the first, we take inspiration from the experiment of the previous Section **??**, where the "tasks" are exemplified by simple point-to-point trajectories demonstrated by a human tutor. In this experiment we explore how to use wrench information and use known demonstrations as ground truth, to evaluate the quality of our prediction.

In the second experiment, we set up a more realistic collaborative scenario, inspired by collaborative object sorting. In such applications, the robot is used to lift an object (heavy, or dangerous, or that the human cannot manipulate, as for some chemicals or food), the human inspects the object and then decides if it is accepted or rejected. Depending on this decision, the object goes on a tray or bin in front of the robot, or on a bin located on the robot side. Dropping the objects in two cases must be done in a different way. Realizing this application with iCub is not easy, as iCub cannot lift heavy objects and has a limited workspace. Therefore, we simplify the experiment with small objects and two bins. The human simply starts the robots movement with physical guidance, and then the robot finishes the movement on its own. In this experiment the predicted trajectories are validated on-the-fly by the human operator.

In a more complex collaborative scenario, tasks could be elementary tasks such as pointing, grasping, reaching, manipulating tools (the type of task here is not important, as long as it can be represented by a trajectory).

8.1 Three simple actions with wrench information

Task trajectories, in this example, have both position and wrench information. In general, it is a good idea to represent collaborative motion primitives in terms of both position and wrenches, as this representation enables using them in the context of physical interaction. Contrarily to the simulated experiment, here the inferred wrenches $\hat{F}_t$ correspond to the wrenches the robot should perceive if the partner was manually guiding the robot to perform the entire movement : indeed, these wrenches are computed from the demonstrations used to learn the primitive. The predicted wrenches can be used in different ways, depending on the application. For example, if the partner breaks the contact with the robot, the perceived wrenches will be different. If the robot is not equipped with tactile or contact sensors, this information can be used by the robot to "perceive" the contact breaking and interpret it, for example, as the sign that the human wants the robot to continue the task on its own. Another use for the demonstrated wrenches is for detecting abnormal forces while the robot is moving : this use can have different applications, from adapting the motion to new environment to automatically detecting new demonstrations. Here, they are simply used to detect when the partner breaks the contact with the robot, and the latter must continue the movement on its own.

In the following, we present how to realize the experiment for predicting the user intention with the real iCub, using our software. The robot must learn three task trajectories represented in Figure **??**. In red, the first trajectory goes from an initial position in front of the robot to its left (task A). In green, the second

trajectory goes from the same initial position to the top (task C). In blue, the last trajectory goes from the top position to the position on the left (task B).

To provide the demonstrations for the tasks, the human tutor used three visual targets shown on the iCub_GUI, a basic module of the iCub code that provides a real-time synthetic and augmented view of the robot status, with arrows for the external forces and colored objects for the targets. One difficulty for novice users of iCub is to be able to drive the robot's arm making it perform desired complex 3D trajectories **?**, but after some practice in moving the robot's arm the operator recorded all the demonstrations. We want to highlight that having variations in the starting or ending points of the trajectories is not at all a problem, since the ProMPs are able to deal with this variability.

We will see that by using the ProMPs method and by learning the end-effector Cartesian position, the robot will be able to learn distributions over trajectories, recognize when a movement belongs to one of these distributions and infer the end of the movement.

In this experiment, the robot received 10 demonstrated trajectories per movement primitive, all provided by the same user. We recorded the Cartesian end-effector position and the wrenches of the robot's left arm. Data are retrieved using the function `used_functions/retrieveRealDataWithoutOrientation.m`. The output parameters of this function are three objects (one per ProMP) that contain all the required information to learn the ProMPs.

In this function, the wrench information are filtered using a Matlab function called `envelope.m` [14] : for each trajectory `traj` and its subMatrix $M = F([1:t])$:

```
1 [envHigh, envLow] = envelope(traj.M);
2 traj.M = (envHigh+envLow)/2;
```

These three objects are saved in `'Data/realIcub.mat'`. A Matlab script called `demo_plotProMPsIcub.m` recovers these data, using the function `load('Data/realIcub.mat')`. This script follows the same organization as the ones we previously explained in Sections **??** and **??**. By launching this script, the recovered data are plotted first.

Then, the ProMPs are computed and plotted, as presented in Figure **??**. In this figure, the distributions are visibly overlaid :
—during the whole trajectories duration for the wrench information ;
—during the 40% first samples of the trajectories for the Cartesian position information.
After this learning step, the user chooses which ProMP to test. Using a variable that represents the percentage of observed data to be used for the inference, the script computes the number of early observations n_o [15] that will be measured by the robot. Using this number, the robot models the time modulation parameter α [16] of each ProMP, as explained in Section **??**. Using this model, the time modulation of the test trajectory is estimated and the corresponding ProMP is identified.

Then, the inference of the trajectory's target is performed. Figure **??** represents the inference of the three tested trajectories when wrench information is not used by the robot to infer the trajectory. To realize this figure, with the comparison between the predicted trajectory and the ground truth, we applied our algorithm

14. Information about this function can be found here : `https://fr.mathworks.com/help/signal/ref/envelope.html?requestedDomain=www.mathworks.com`
15. n_o is not the same for each trajectory test, because it depends on the total duration of the trajectory to be inferred.
16. Since the model uses the n_o parameter, its computation cannot be performed before this step.

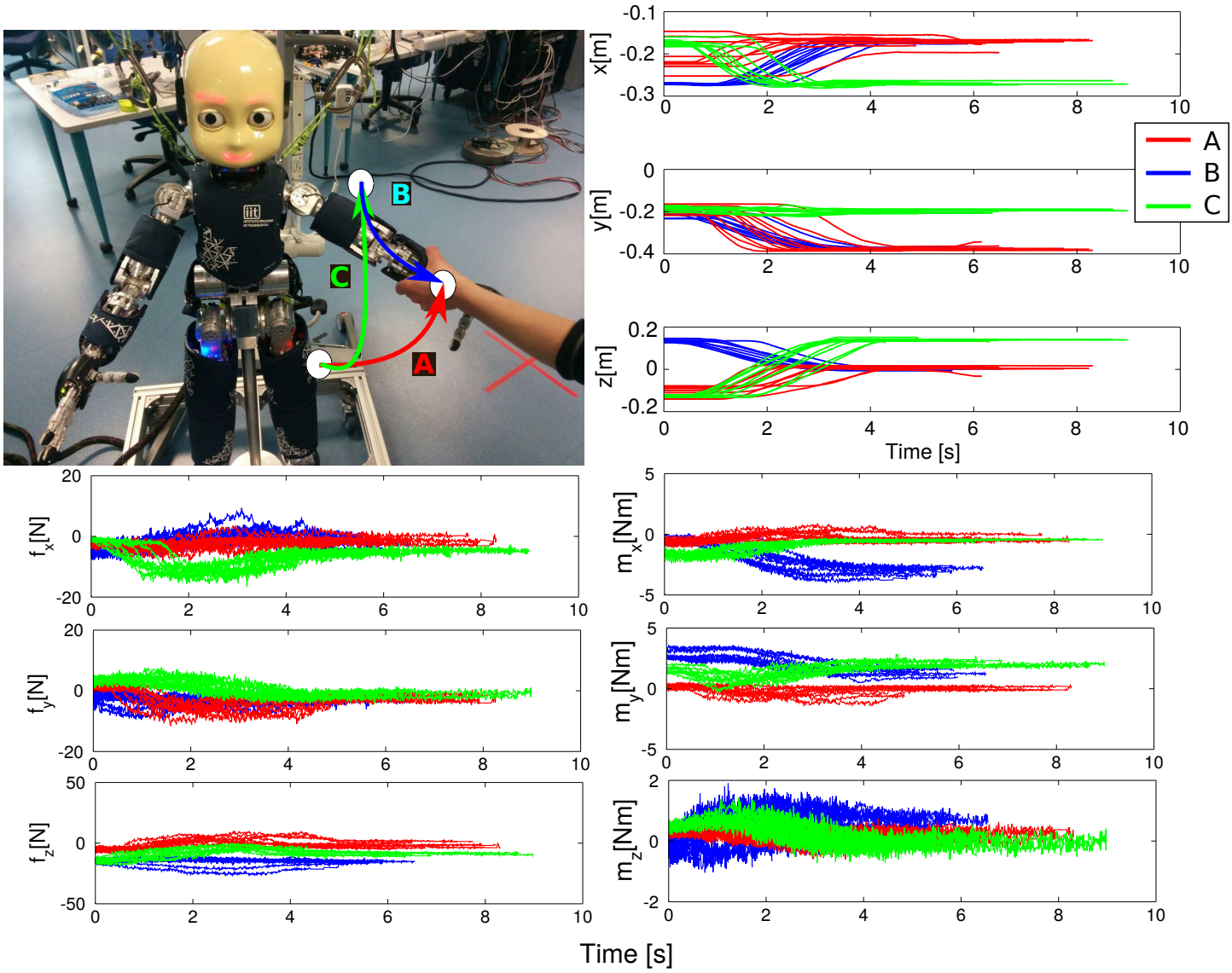

FIGURE 11. Top left : the iCub and the visualization of the three targets in its workspace, defining the three tasks A-B-C. Top right : Cartesian position information of the demonstrated trajectories for the three tasks. Bottom left and right : wrench (force and moment) information of the demonstrated trajectories.

offline. In fact, it is not possible at time t to have the ground truth of the trajectory intended by the human from $t+1$ to t_f : even if we would tell to the human in advance the goal that he/she must reach for, the trajectory to reach that goal could vary. So, for the purpose of these figures and comparisons with the ground truth, we show here the offline evaluation : we select one demonstrated task trajectory from the test set (not the training set used to learn the ProMP) as ground truth, and imagine that this is the intended trajectory. In Figure **??**, the ground truth is shown in black, whereas the portion of this trajectory that is fed to the inference, and that corresponds to the "early observations", is represented with bigger black circles. We can see that the inference of the Cartesian position is correct, although we can see an error of about 1 second of the estimated duration time for the last trial. Also, the wrench inference is not accurate. We can assume that it is : because the robot infers the trajectory using only position information without wrench information, or because the wrenches' variation is not correlated to the position variation. To improve this result, we can make the inference using wrench in addition to Cartesian position information, as shown

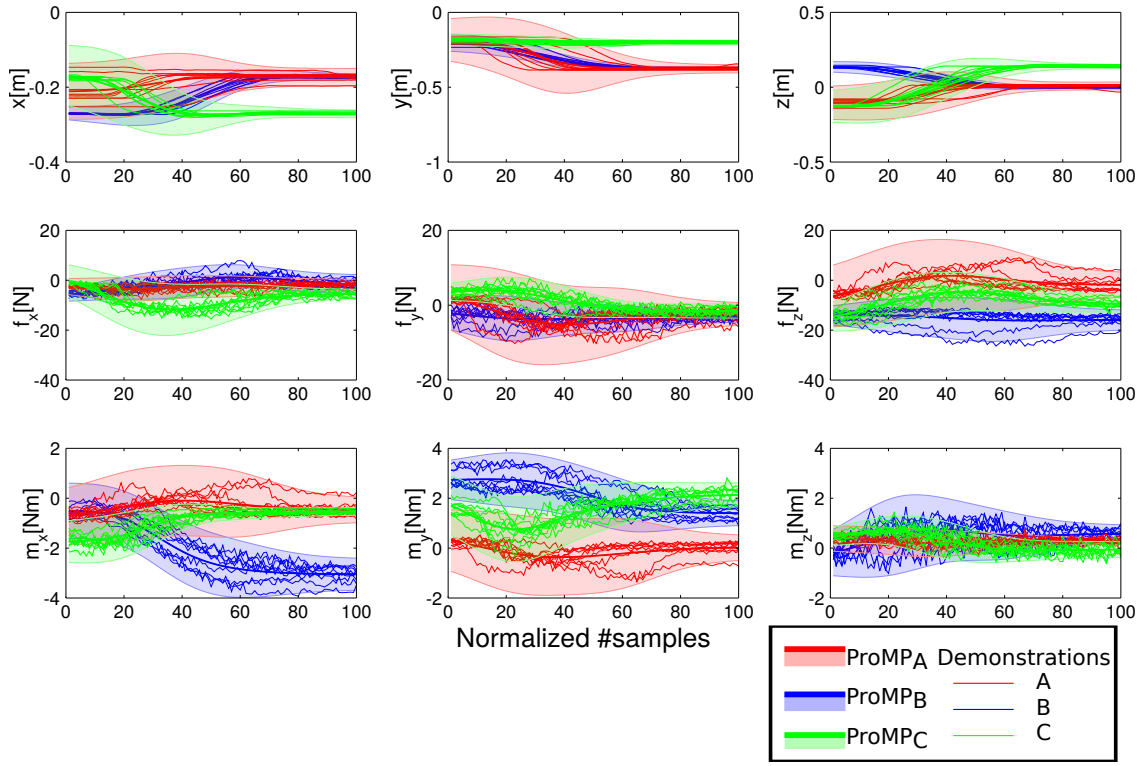

FIGURE 12. The ProMPs learned by the robot from the demonstrations of Figure **??**.

in Figure **??**. We can see in this Figure that the estimation of the trajectory's duration is accurate. The disavantage is that the inference of the Cartesian position is less accurate because the posterior distribution computation makes a trade-off between fitting Cartesian position and wrench early observations. Moreover, to allow a correct inference using wrench information, the noise expectation must be increased to consider forces. [17]

To confirm these results, we analyzed the trajectory inference and α estimation considering different percentages of each trajectory as observed data (30 to 90%). For each percentage, we performed 20 tests, with and without force information.

In Figure **??**, each box-plot represents errors for 20 tests. On the top, the error criterion is the average distance between the inferred trajectory and the real one. We can see that the inference of Cartesian end-effector trajectory is more accurate without wrench information. On the bottom, the error criterion is the distance between the estimated α and the real one. We can see that using wrench information, the estimation of the α is more accurate. Thus, these two graphs confirm what we assumed from Figures **??** and **??**.

Median, mean and variance of the prediction errors, computed with the normalized root-mean-square error (NRMSE) are reported in Table **??**. The prediction error for the time modulation is a scalar : $|\alpha_{\text{prediction}} - \alpha_{\text{real}}|$. The prediction error for the trajectory is computed by the NRMSE of $|\Xi_{\text{prediction}} - \Xi_{\text{real}}|$.

In future upgrades for this application, we will probably use the wrench information only to estimate the time modulation parameter α, to have both the best inference of the intended trajectory and the best

17. In future versions, we will include the possibility to have different noise models for the observations, *e.g.* we will have $\Sigma_\Xi^o = \begin{bmatrix} \Sigma_X & 0 \\ 0 & \Sigma_F \end{bmatrix}$. We will therefore set a bigger covariance for the wrench information than for the position information.

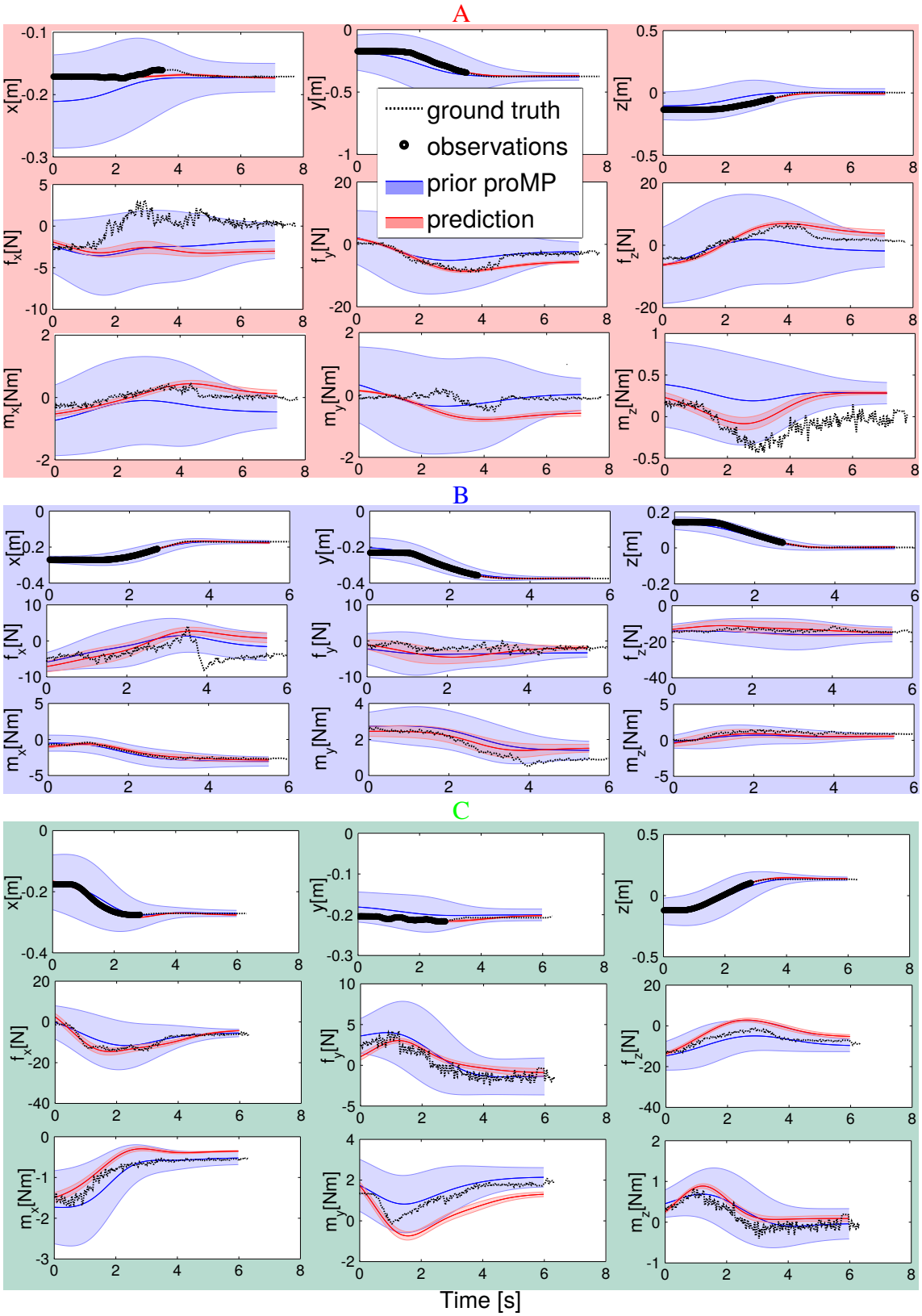

FIGURE 13. The prediction of the future trajectory from the learned ProMPs computed from the position information for the 3-targets dataset on the real iCub (Figure **??**) after 40% of observations.

With wrenches

% of observed data (≃ n. of samples)		30(≃ 180)	50(≃ 300)	70(≃ 419)	90(≃ 539)
Prediction error of time modulation	median	2.26e-08	1.35e-08	3.61e-09	8.95e-09
	mean	4.26e-02	7.81e-09	2.19e-08	2.09e-08
	variance	1.08e-02	1.09e-16	5.44e-16	2.24e-16
Prediction error for the trajectory (NRMSE) [m]	median	2.73e-01	5.51e-01	4.52e-01	5.38e-01
	mean	8.11e-01	5.86e-01	5.29e-01	3.42e-01
	variance	7.45e-01	1.36e-01	7.74e-02	2.93e-02
Computation time [s]	mean	0.25	0.74	1.92	3.59
	variance	0.01	0.27	2.77	4.81

Without wrenches

% of observed data (≃ n. of samples)		30(≃ 180)	50(≃ 300)	70(≃ 419)	90(≃ 539)
Prediction error of time modulation	median	1.19e-02	1.76e-02	1.74e-02	1.62e-02
	mean	2.81e-02	1.92e-02	2.45e-02	1.43e-02
	variance	9.51e-04	1.00e-04	3.37e-04	1.82e-04
Prediction error for the trajectory (NRMSE) [m]	median	4.53e-02	4.73e-02	7.20e-02	6.20e-02
	mean	1.56e-01	7.36e-02	5.75e-02	4.29e-02
	variance	5.04e-02	1.80e-03	1.80e-03	6.0e-04
Computation time [s]	mean	6.89e-02	8.49e-02	1.43e-01	2.58e-01
	variance	2.83e-03	1.19e-03	7.31e-03	2.45e-03

Table 3. Mean and stdev of the NRMSE of the prediction errors plotted in Figure **??**, and average time for computing both predictions (time modulation and trajectory via update of the posterior distribution). The computation were performed in Matlab, on a single core (no parallelization).

estimation of the time modulation parameter to combine the benefits of inference with and without wrench information.

Table **??** also reports the average time for computing the prediction of both time modulation and posterior distribution. The computation were performed in Matlab, on a single core laptop (no parallelization). While the computation time for the case "without wrenches" is fine for real-time application, using the wrench information delays the prediction and represents a limit for real-time applications if fast decisions have to taken by the robot. Computation time will be improved in the future works, with the implementation of the prediction in an iterative way.

8.2 Collaborative object sorting

We realized another experiment with iCub, where the robot has to sort some objects in different bins (see Figure **??**). We have two main primitives : one for a bin located on the left of the robot, and one for the bin to the front. Dropping the object is done at different heights, with a different gesture that also has a different orientation of the hand. For this reason, the ProMP model consists of the Cartesian position of the hand $X_t = [x_t, y_t, z_t] \in R^3$ and its orientation $A_t \in R^4$, expressed as a quaternion :

$$\xi_t = \begin{bmatrix} X_t \\ A_t \end{bmatrix} = \Phi_{\alpha t}\, \boldsymbol{\omega} + \epsilon_t$$

As in the previous experiment, we first teach the robot the primitives by kinesthetic teaching, with a dozen of demonstrations. Then we start the robot movement : the human operator physically grabs the robot's arm and start the movement towards one of the bins. The robot' skin is used twice. First, to detect the

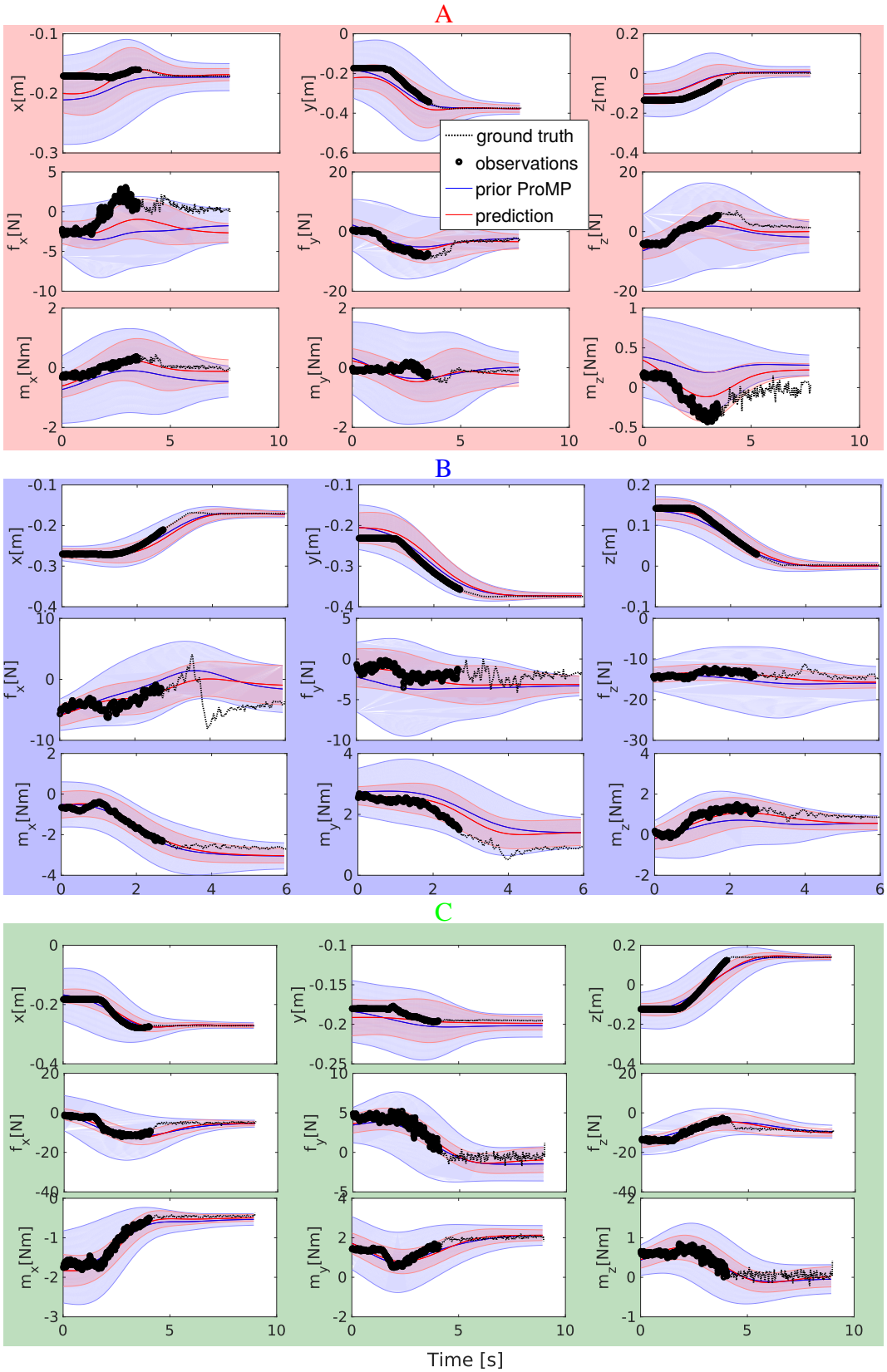

FIGURE 14. The prediction of the future trajectory from the learned ProMPs computed from the position and wrench information for the 3-targets dataset on the real iCub (Figure **??**) after 40% of observations.

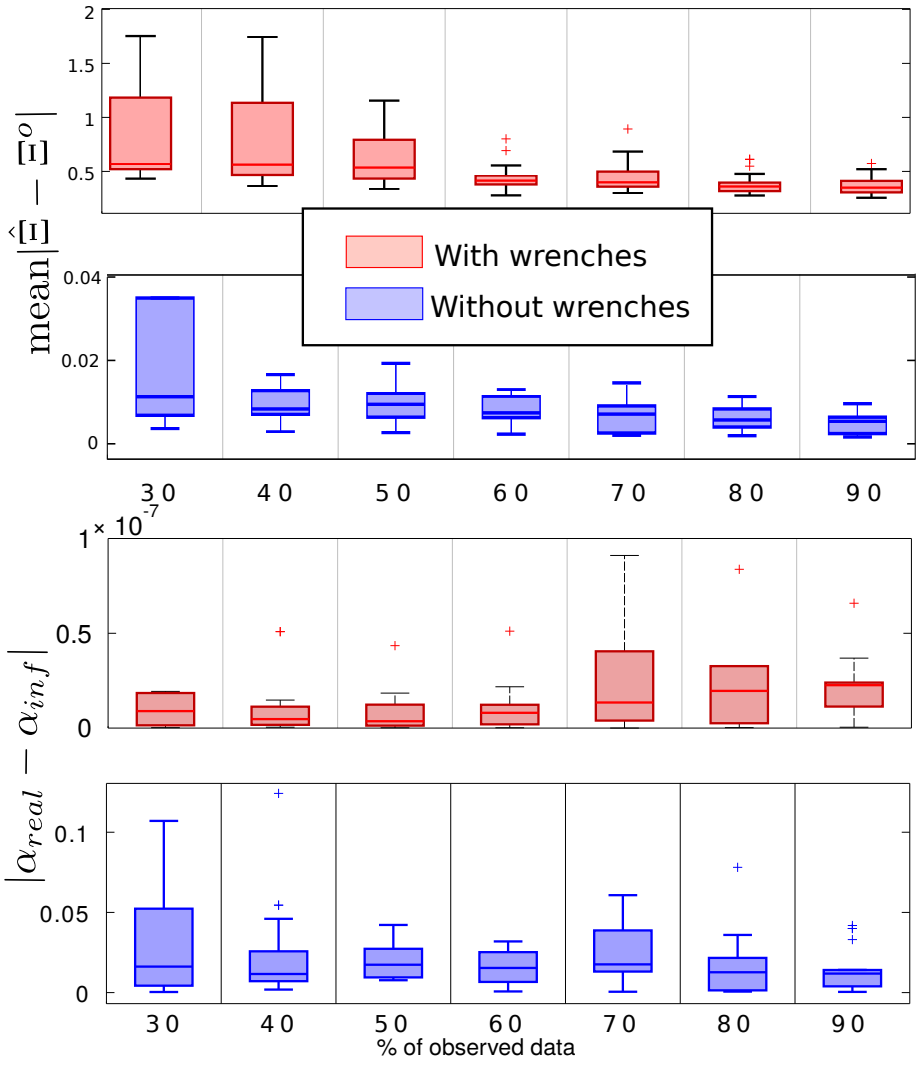

FIGURE 15. Trajectory prediction error (top) and time modulation estimation error (bottom) of the future trajectory with and without wrench information, for the 3-targets dataset on the real iCub (Figure **??**) with respect to the number of observed data points.

contact when the human grabs the arm, which marks the beginning of the observations. Second, when the human breaks the contact with the arm, which marks the end of the observations. Using the first portion of the observed movement, the robot recognize the current task that is being executed, predicts the future movement that is intended by the human and then executes it on its own. In the video (see link in Section **??**) we artificially introduced a pause to let the operator "validate" the predicted trajectory, using a visual feedback on the iCubGui. Figure **??** shows one of the predictions made by the robot after the human releases the arm. Of course in this case we do not have a "ground truth" for the predicted trajectory, only a validation of the predicted trajectory by the operator.

9 VIDEOS

We recorded several videos that complement the tutorials. The videos are presented in the github repository of our software : `https://github.com/inria-larsen/icubLearningTrajectories/tree/master/Videos`.

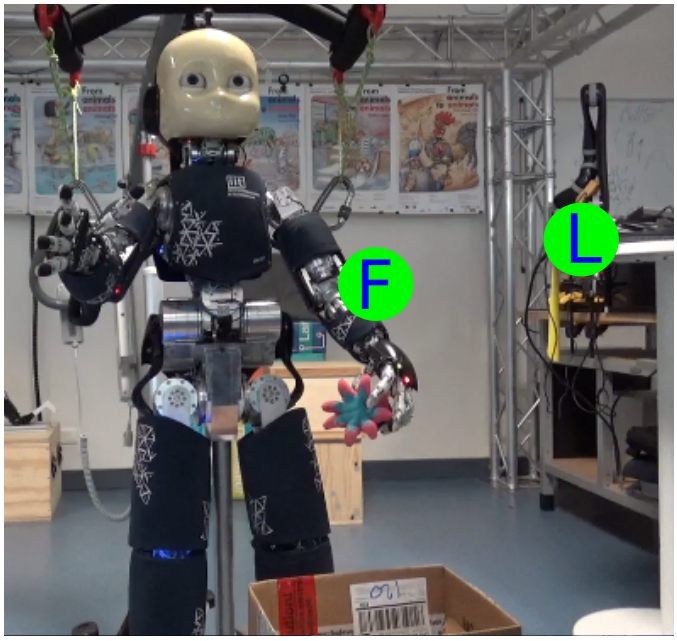

FIGURE 16. The second experiment with the robot : iCub must sort the objects into two bins, guided by the human. If the object is good, the robot has to put the object in the "front bin" ; if the object is not good, the robot has to put the object in the "left bin". The gestures to put the objects into the two bins are different. To simplify, the drop locations for the two bins are represented by the targets F and L. After inspecting the object, the human drives the robot towards the front of the left.

10 DISCUSSION

While we believe that our proposed method is principled and has several advantages for predicting intention in human-robot interaction, there are numerous improvements that can be done. Some will be object of our future works.

Improving the estimation of the time modulation - Our experiments showed that estimating the time modulation parameter α, determining the duration of the trajectory, greatly improves the prediction of the trajectory in terms of difference with the human intended trajectory (*i.e.*, our ground truth). We proposed four simple methods in Section **??**, and in the iCub experiment we showed that the method that maps the time modulation and the variation of the trajectory in the first n_o observations provides a good estimate of the time modulation α for our specific application. However, it is an *ad hoc* model that cannot be generalized to all possible cases. Overall, the estimation of the time modulation (or phase) can be improved. For example, **?** used Dynamic Time Warping, while **?** proposed to improve the estimation by having local estimations of the speed in the execution of the trajectory, to comply with cases where the velocity of task trajectory may not be constant throughout the task execution. In the future, we plan to explore more solutions and integrate them into our software.

Improving prediction - Another point that needs further investigation and improvement is how to improve the prediction of the trajectories exploiting different information. In our experiment with iCub, we improved the estimation of the time modulation using position and wrench information ; however, we observed that the noisy wrench information does not help in improving the prediction of the position trajectory. One improvement is to certainly exploit more information from the demonstrated trajectories, such as estimating the different noise of every trajectory component and exploiting this information to improve the prediction. Another possible improvement would consist in using contextual information about

the task trajectories. Finally, it would be interesting to try to identify automatically the characteristic such as velocity profiles or accelerations, that are renown to play a key role in attributing intentions to human movements. For example, in goal-directed tasks such as reaching, the arm velocity profile and the hand configuration are cues that helps us detect intentions. Extracting these cues automatically, leveraging the estimation of the time modulation, would probably improve the prediction of the future trajectory. This is a research topic on its own, outside the scope of this paper, with strong links to human motor control.

Continuous prediction - In Section **??** we described how to compute the prediction of the future trajectory after recognizing the current task. However, we did not explore what happens if the task recognition is wrong : this may happen, if there are two or more task with a similar trajectory at the beginning (*e.g.*, moving the object from the same initial point towards one of four possible targets), or simply because there were not enough observed points. So what happens if our task recognition is wrong ? How to re-decide on a previously identified task ? And how should the robot decide if its current prediction is finally correct (in statistical terms) ? While implementing a continuous recognition and prediction is easy with our framework (one has simply to do the estimation at each time step), providing a generic answer to these question may not be straightforward. Re-deciding about the current task implies also changing the prediction of the future trajectory. If the decision does not come with a confidence level greater than a desired value, then the robot could face a stall : if asked to continue the movement but unsure about the future trajectory, should it continue or stop ? The choice may be application-dependent. We will address these issues and the continuous prediction in future works.

Improving computational time - Finally, we plan to improve the computational time for the inference and the portability of our software by porting the entire framework in C++.

Learning tasks with objects - In many collaborative scenarios, such as object carrying and cooperative assembly, the physical interaction between the human and the robot is mediated by objects. In these cases, if specific manipulations must be done on the objects, our method still applies, but not only on the robot. It must be adapted to the new "augmented system" consisting of robot and object. Typically, we could image a trajectory for some frame or variable or point of interest for the object, and learn the corresponding task. Since ProMPs support multiplication and sequencing of primitives, we could exploit the properties of the ProMPs to learn the joint distribution of the robot task trajectories and the object task trajectories.

11 CONCLUSION

In this paper we propose a method for predicting the intention of a user physically interacting with the iCub in a collaborative task. We formalize the intention prediction as predicting the target and "future" intended trajectory from early observations of the task trajectory, modeled by Probabilistic Movement Primitives (ProMPs). We use ProMPs because they capture the variability of the task, in the form of a distribution of trajectories coming from several demonstrations of the task. From the information provided by the ProMP, we are able to compute the future trajectory by conditioning the ProMP to match the early observed data points. Additional features of our method are the estimation of the duration of the intended movement, the recognition of the current task among the many known in advance, and multimodal prediction.

Section 4 described the theoretical framework, whereas Sections **??**–**??** presented the open-source software that provides the implementation of the proposed method. The software is available on `github`, and tutorials and videos are provided.

We used three examples of increasing complexity to show how to use our method for predicting the intention of the human in collaborative tasks, exploiting the different features. We presented experiments with both the real and the simulated iCub. In our experiments, the robot learns a set of motion primitives corresponding to different tasks, from several demonstrations provided by a user. The resulting ProMPs are the prior information that is later used to make inferences about human intention. When the human starts a new collaborative task, the robot uses the early observations to infer which task the human is executing, and predicts the trajectory that the human intends to execute. When the human releases the robot, the predicted trajectory is used by the robot to continue executing the task on its own.

In Section **??** we discussed some current issues and challenges for improving the proposed method and make it applicable to a wider repertoire of collaborative human-robot scenarios. In our future works, our priority would be in accelerating the time for computing the inference, and finding a principled way to do continuous estimation, by letting the robot re-decide continuously about the current task and future trajectory.

Appendices

A DETAIL OF THE INFERENCE FORMULA

In this appendices, we explain how to obtain the inference formulae used in our software. First, let us recall the Marginal and Conditional Gaussians laws [18] Given a marginal Gaussian distribution for x and a Gaussian distribution for y given x in the form :

$$p(x) = \mathcal{N}\left(x|\mu, \Delta^{-1}\right) p(y|x) = \mathcal{N}\left(Ax + b, L^{-1}\right) \tag{18}$$

the marginal distribution of y and the conditional distribution of x given y are given by

$$p(y) = \mathcal{N}\left(y|A\mu + b, L^{-1} + A\Delta^{-1}A^{\top}\right) \tag{19}$$

$$p(x|y) = \mathcal{N}\left(x|\Sigma A^{\top} L(y - b) + \Delta\mu, \Sigma\right) \tag{20}$$

where

$$\Sigma = (\Delta + A^{\top} LA)^{-1}$$

We computed the parameter's marginal Gaussian distribution from the set of observed movements :

$$p(\boldsymbol{\omega}) \sim \mathcal{N}(\mu_{\boldsymbol{\omega}}, \Sigma_{\boldsymbol{\omega}}) \tag{21}$$

From the model $\Xi_t = \Phi_{[1:t_f]}\boldsymbol{\omega} + \epsilon_{\Xi}$, we have the conditional Gaussian distribution for Ξ given $\boldsymbol{\omega}$:

$$p(\Xi|\boldsymbol{\omega}) = \mathcal{N}(\Xi|\Phi_{[1:t_f]}\boldsymbol{\omega}, \Sigma_{\Xi}) \tag{22}$$

18. From the book **?**

Then, using Equation **??** :

$$p(\Xi) = \mathcal{N}(\Xi|\Phi_{[1:t_f]}\mu_w, \Sigma_\Xi + \Phi_{[1:t_f]}\Sigma_{\boldsymbol{\omega}}\Phi^\top_{[1:t_f]}) \tag{23}$$

that is the prior distribution of the ProMP.

Let $\Xi^o = [\xi^o(1), \dots, \xi^o(n_o)]$ be the first n_o observations of the trajectory to predict with the first n_o elements corresponding to the early observations.

Let $\hat{\Xi} = [\xi^o(1), \dots, \xi^o(n_o), \hat{\xi}(n_o+1), \dots, \hat{\xi}(t_{\hat{t}_f})]$ be the whole trajectory we have to predict. We can then compute the posterior distribution of the ProMP by using the conditional Gaussians Equation **??** :

$$p(\boldsymbol{\omega}|\Xi^o) = \mathcal{N}(\boldsymbol{\omega}|\mu_{\boldsymbol{\omega}} + K(\Xi^o - \Phi_{[1:n_o]}\mu_{\boldsymbol{\omega}}), \Sigma_{\boldsymbol{\omega}} - K\Phi_{[1:n_o]}\Sigma_{\boldsymbol{\omega}}) \tag{24}$$

$$\text{with } K = \Sigma_{\boldsymbol{\omega}}\Phi^\top_{[1:n_o]}(\Sigma_\Xi + \Phi_{[1:n_o]}\Sigma_{\boldsymbol{\omega}}\Phi^\top_{[1:n_o]})^{-1} \tag{25}$$

Thus, we have the posterior distribution of the ProMP $p(\boldsymbol{\omega}|\Xi^o) = \mathcal{N}(\boldsymbol{\omega}|\hat{\boldsymbol{\mu}}_\omega, \hat{\boldsymbol{\Sigma}}_\omega)$ with :

$$\begin{cases} \hat{\mu}_{\boldsymbol{\omega}} &= \mu_{\boldsymbol{\omega}} + K(\Xi^o - \Phi_{[1:n_o]}\mu_{\boldsymbol{\omega}}) \\ \hat{\Sigma}_{\boldsymbol{\omega}} &= \Sigma_{\boldsymbol{\omega}} - K(\Phi_{[1:n_o]}\Sigma_{\boldsymbol{\omega}}) \\ K &= \Sigma_{\boldsymbol{\omega}}\Phi^\top_{[1:n_o]}(\Sigma^o_\xi + \Phi_{[1:n_o]}\Sigma_{\boldsymbol{\omega}}\Phi^\top_{[1:n_o]})^{-1} \end{cases} \tag{26}$$

CONFLICT OF INTEREST STATEMENT

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

AUTHOR CONTRIBUTIONS

Designed study : OD, AP, FC, SI. Wrote software : OD, ME, AP, SI. Wrote paper : OD, AP, ME, FC, JP, SI.

FUNDING

This paper was partially funded by the European projects CoDyCo (no. 600716 ICT211.2.1) and AnDy (no. 731540 H2020-ICT-2016-1), and the French CPER project SCIARAT.

ACKNOWLEDGMENTS

The authors wish to thank the IIT researchers of the CoDyCo project for their support with iCub, Ugo Pattacini and Olivier Rochel for their help with the software for the Geomagic, and Iñaki Fernández Pérez for all his relevant feedback.

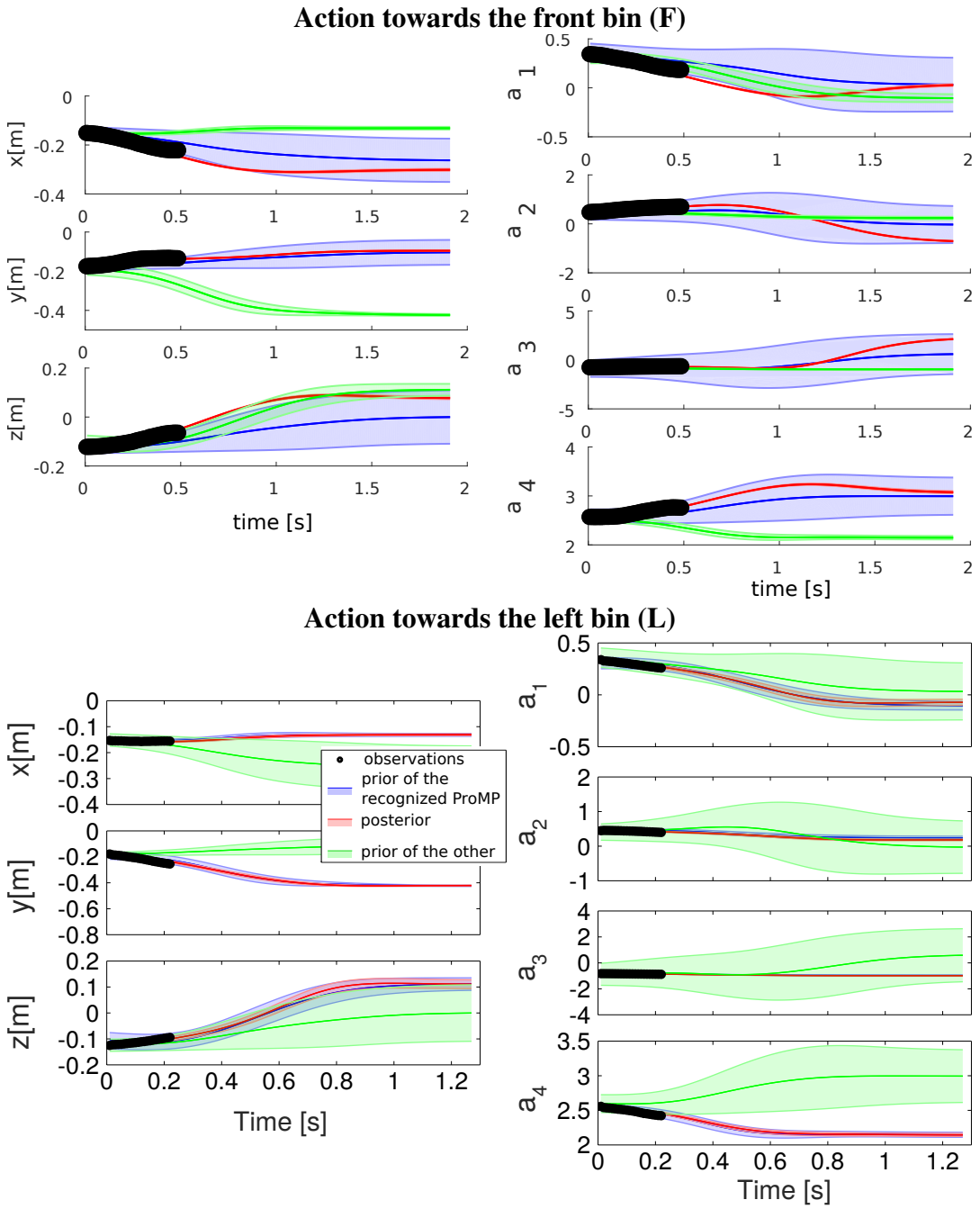

FIGURE 17. Predicted trajectories for the second experiment with the robot (Figure **??**). The black circles represent the observations acquired while the human is physically moving the iCub's arm. When the human breaks the contact and releases the arm, the robot predicts the future trajectory and continues the movement. The prior of the recognized ProMP is blue, the posterior ProMP used for prediction is red, the prior ProMP of the other task (i.e., the one that is recognized as not the one currently being executed) is green.**Top, F** : the human moves the arm towards the front bin. After few observations ($\sim$ 0.5s) the robot recognizes that the movement corresponds to the "F" action. The prior of the F actions is blue, the posterior/prediction is red, the L action is green. **Bottom, L** : the human moves the arm towards the left bin. After few observations ($\sim$ 0.25s) the robot has recognized the L action. The prior of the L action is blue, the posterior red, the F action (not recognized) is green.